Timelines with Temporal Uncertainty*

Alessandro Cimatti[1] and **Andrea Micheli**[1,2] and **Marco Roveri**[1]
[1]Fondazione Bruno Kessler – Italy,
[2]University of Trento – Italy
{cimatti,amicheli,roveri}@fbk.eu

Abstract

Timelines are a formalism to model planning domains where the temporal aspects are predominant, and have been used in many real-world applications. Despite their practical success, a major limitation is the inability to model temporal uncertainty, i.e. the fact that the plan executor cannot decide the actual duration of some activities.

In this paper we make two key contributions. First, we propose a comprehensive, semantically well founded framework that (conservatively) extends with temporal uncertainty the state of the art timeline approach.

Second, we focus on the problem of producing time-triggered plans that are robust with respect to temporal uncertainty, under a bounded horizon. In this setting, we present the first complete algorithm, and we show how it can be made practical by leveraging the power of Satisfiability Modulo Theories.

Introduction

Timelines are a comprehensive formalism to model planning domains where the temporal aspects are predominant. The framework builds on a quantitative extension of Allen's temporal operators (Allen 1983; Angelsmark and Jonsson 2000). For example, it is possible to state that a certain activity must last no longer than 5 seconds, and must be carried out during another activity. The key difference with respect to (Allen 1983) is in the fact that, with timelines, the number and type of activities is not known a priori — they are the result of unrolling over time the domain description (similar to the instantiation of operators into actions in classical planning).

Timelines have been used in many real-world applications. The research line pioneered by NASA, that resulted in the Europa planner (Barreiro et al. 2012), is based on a timeline framework. APSI is a timeline-based framework that has been developed in the European Space Agency (ESA) since 2008 (Donati et al. 2008; Cesta et al. 2009a). The framework is very expressive, and it has been used to describe real-world planning and scheduling domains and problems (Donati et al. 2008; Cesta et al. 2008), and as a core for several practical applications (Cesta et al. 2009b; 2010a; 2011).

Despite the practical success of the timeline approach, a major limitation is the inability to express temporal uncertainty. Temporal uncertainty is needed to model situations in which some activities have a duration that cannot be controlled by the plan executor. Such phenomenon is pervasive in several application domains, including transportation, production planning, and aerospace. In fact, in 2010 ESA issued an Invitation to Tender aiming at the extension of the APSI timeline-based framework with uncertainty.

In this paper, we make the following contributions. First, we propose a comprehensive framework, that (conservatively) extends the state of the art timeline approach with temporal uncertainty. We provide a semantic foundation to the strong controllability problem, that is the problem of producing time-triggered plans that are robust with respect to temporal uncertainty. In practice, this is useful to generate plans that are guaranteed to fulfill the goal under any possible behavior of the uncertain components of the domain.

Second, we present the first complete algorithm for timeline planning under temporal uncertainty. The approach is based on the logical encoding of the problem into the problem of satisfiability of a first order formula with respect to a background theory. The approach is made practical by leveraging the power of Satisfiability Modulo Theories (Barrett et al. 2009) (SMT). In addition to the direct encoding, we propose a lazy algorithm that relies on the incremental use of the underlying SMT solver. We experimented on various problems, and the results confirm the potential of the approach.

This paper is structured as follows. In Section we present some background. In Section we model timelines with uncertainty; in Section we show how to encode the strong controllability problem into SMT. In Section we compare our approach with related work, and in Section we experimentally evaluate it. In Section we draw some conclusions, and outline directions for future research.

Background

Allen's Algebra. Allen's algebra is a well known formalism to reason about the temporal properties of a finite set of activities (Allen 1983). The algebra is defined by 13 operators, representing all the possible relations between a pair of intervals, by a transitivity table that allows for con-

*The research was partly funded by the Autonomous Province of Trento, project ACube (Grandi Progetti 2006).

straint propagation and by an inverse function. The problem of checking the temporal consistency of a set of Allen constraints is known to be NP-hard (Allen 1983).

Allen's algebra has been extended in a number of works to express quantitative information (Angelsmark and Jonsson 2000; Cheng and Smith 1994; Drakengren and Jonsson 1997; Wetprasit and Sattar 1998; Cesta et al. 2009a). In this paper, we use the extension proposed in (Cesta et al. 2009a), where operators are annotated with intervals: for example, the expression *"A contains [10,20] [2,5] B"* states that the interval A contains the interval B, and the start of A precedes the start of B by no less than 10 and no more than 20 time units; similarly the end of B precedes the end of A by no less than 2 and no more than 5 time units. The semantics of the operators is given in terms of structures describing the mutual relations between the start/end point of each interval. Clearly, this quantitative formalism subsumes the qualitative version: each "classical" Allen operator can be obtained by setting the quantitative intervals to $[0, \infty]$.

In our setting we consider the time to be dense. Time points are interpreted over real values.

Satisfiability Modulo Theory. Given a first-order formula ψ in a decidable background theory T, *Satisfiability Modulo Theory* (SMT) (Barrett et al. 2009) is the problem of deciding whether there exists a satisfying assignment to the free variables in ψ.

In this work we concentrate on the theory of Linear Arithmetic over the Real numbers (LRA). A formula in LRA obtained from atoms by applying Boolean connectives (negation $\neg$, conjunction $\wedge$, disjunction $\vee$), and universal ($\forall$) and existential ($\exists$) quantification. Atoms are 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. We denote with QF_LRA the quantifier-free fragment of LRA.

As an example, consider the QF_LRA formula $(x \leq y) \wedge (x + 3 = z) \vee (z \geq y)$ with x, y, z being real variables. In the theory of real arithmetic, numerical constants are interpreted as the corresponding real numbers, and $+, =, <, >, \leq, \geq$ as the corresponding operations and relations over $\mathbb{R}$. The formula is satisfiable, and a satisfying assignment is $\{x := 5,\ y := 6,\ z := 8\}$.

An SMT solver is a decision procedure which solves the satisfiability problem for a formula expressed in a decidable subset of First-Order Logic. Currently, the most efficient implementations of SMT solvers use the so-called "lazy approach", where a SAT solver is tightly integrated with a T-solver. See (Barrett et al. 2009) for a survey.

Several techniques have been developed for removing quantifiers from an LRA formula (e.g. Fourier-Motzkin (Schrijver 1998), Loos-Weispfenning (Loos and Weispfenning 1993; Monniaux 2008)): they transform an LRA formula into a QF_LRA formula that is *logically equivalent* modulo the LRA theory. These techniques enable for the solution of quantified formulae at a cost that is doubly exponential in time and space in the original formula size (Schrijver 1998; Monniaux 2008; Loos and Weispfenning 1993).

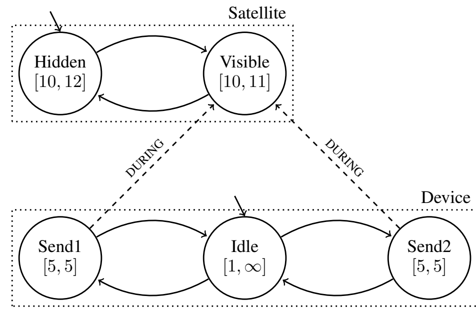

Figure 1: Running example

It is possible to reduce the consistency problem for quantitative Allen's algebra to SMT(QF_LRA). Intuitively, for each activity, two (start/end) real variables are introduced; each constraint over two activities is mapped to an SMT formula over the corresponding variables. For example, A before B by at least 10 time units is expressed as $B.start - A.end \geq 10$. The disjunction in SMT is essential to express "non-convex" Allen's constraints.

In the following we will use the following shorthands. Let V be a finite set $\{v_1, v_2, \ldots, v_n\}$. We write $t \in V$ for the formula $t = v_1 \vee t = v_2 \vee \ldots \vee t = v_n$. Let I be an interval $[l, h)$. We write $t \in I$ for the formula $t \geq l \wedge t < h$. We write $\mathbb{I}$ for the set of all possible intervals. Let $I_1 = [s_1, e_1)$ and $I_2 = [s_2, e_2)$ be two intervals, we write $I_1 \subseteq I_2$ if $s_1 \geq s_2$ and $e_1 \leq e_2$.

Timelines with Uncertainty

Example. Consider a communication device that can send data packets of two different types to a satellite during the time period in which the satellite is visible. The visibility window of the satellite is not controllable by the communication device and it ranges between 10 and 11 hours, while the satellite remains hidden in the following 10-12 hours (also uncontrollably). The device needs 5 hours to send each packet of data and a transmission has to happen *during* the visibility window. Notice that both the satellite and the device can be in each state more than once. The satellite is initially hidden, and the device is idle. The goal is to send one data packet per type. The situation is depicted in Figure 1.

Syntax. We introduce an abstract notation for timeline-based domain descriptions. We retain all the features of the concrete languages used in the applications. Intuitively, the timeline framework can be thought of as a "sequential version" of Allen's algebra, where the same activity can be instantiated multiple times. The instantiations are obtained by means of generators.

Definition 1. *A* generator *G is a tuple (V, T, δ) such that V is a finite set of values, $T \subseteq V \times V$ is a transition relation, $\delta : V \to \mathbb{I}$ is a temporal labeling function.*

A generator represents a *state variable* over values V in a timeline framework[1]. The transition relation T is used to logically describe the evolution of the generator. If $T(v_i, v_j)$, then the end of (an instance of) activity v_i can be followed by the start of (an instance of) activity v_j. In our satellite example, the system is composed of two generators: the Satellite and the Communicator. The Satellite generator has values $\{Visible, Hidden\}$, the transition relation imposes the alternation of $Visible$ and $Hidden$ values, and the δ function imposes the minimal and maximal duration of each value ($Visible \rightarrow [10, 11)$, $Hidden \rightarrow [10, 12)$). The Communicator generator is three-valued ($Idle, Send1, Send2$), the transition relation imposes the automaton shape depicted in Figure 1 and the duration constraints are $Idle \rightarrow [1, \infty)$, $Send1 \rightarrow [5, 5)$ and $Send2 \rightarrow [5, 5)$.

In order to express the constraints between different generators, we introduce the notion of synchronization.

Definition 2. *Let $G_i = (V_i, T_i, \delta_i)$ be generators, with $i \in \{0, \ldots, n\}$. An n-ary* synchronization *σ is a triple $((G_0, v_0), \{(G_1, v_1), \ldots, (G_n, v_n)\}, C)$, such that, for all $i \in \{0, \ldots, n\}$, $v_i \in V_i$, and C is a set of Allen constraints in the form $v_h \bowtie v_k$, with $h, k \in \{0, \ldots, n\}$.*

The synchronizations are based on Allen's temporal operators applied to generator values. The interpretation, however, is quite different from (Allen 1983). For example, "$Send1\ during\ [0, \infty)\ Visible$" means that *every* instance of $Send1$ occurs during *some* instance of $Visible$; similarly, "$Visible\ during^{-1}\ [0, \infty)\ Send1$" means that during *every* visibility window *some* $Send1$ occurs. Therefore, the algebraic properties of (Allen 1983) are not retained here. In Figure 1, we indicated two synchronizations using dashed arrows. These synchronizations are used to require that the packet of data is sent *during* the visibility window.

A set of generators and a set of synchronizations are sufficient to define a planning domain. For what concerns the planning problem, we do not distinguish between facts and goals: we just require an execution that exhibits a set of (temporally-extended and temporally constrained) *facts*.

Definition 3. *Let $G = (V, T, \delta)$ be a generator. A* unary fact *is a tuple (G, v, I_s, I_e), where $v \in V$ and $I_s, I_e \in \mathbb{I}$. Let f_1 and f_2 be two unary facts and let $\bowtie$ be a quantified Allen operator. A* binary fact *is a constraint in the form $f_1 \bowtie f_2$.*

A unary fact prescribes the existence of a value v in the execution of G, that starts during I_s and ends during I_e. A binary fact is useful to impose constraints (e.g. precedence, containment) between the intervals in the corresponding unary facts. In our satellite example, we use two unary facts to force the initial condition of the system: $f_1 = (Satellite, Hidden, [0, 0], [0, \infty))$ forces the satellite to be in $Hidden$ state at 0. Similarly, $f_2 = (Communicator, Idle, [0, 0], [0, \infty))$ constrains the initial state of the communicator to be idle.

Similarly, to express the goals we introduce $g_1 = (Communicator, Send1, [0, \infty), [0, \infty))$ and $g_2 = (Communicator, Send1, [0, \infty), [0, \infty))$, that require the communicator to be eventually in $Send1$ state and in $Send2$ state. If we need to order the goals, prescribing that the packet 1 must be sent before packet 2, we can impose a binary fact $g1\ before[0, \infty)\ g2$. Notice that the goals are temporally extended, i.e. they do not simply require to reach a final condition.

[1]Without loss of generality, we disregard the parametrization used in some timeline languages.

The above definitions characterize timelines in the classical sense. In order to deal with temporal uncertainty, we now introduce an annotation to distinguish controllable and uncontrollable elements.

Definition 4. *A* CU-annotation for a set of generators *$\mathcal{G} = \{G_i = (V_i, T_i, \delta_i)\}$ is a function $\chi : \mathcal{G} \times \bigcup_i V_i \rightarrow \{\mathtt{C}, \mathtt{U}\} \times \{\mathtt{C}, \mathtt{U}\}$. A* CU-annotation for a set of synchronizations *S is a function $\chi : S \rightarrow \{\mathtt{C}, \mathtt{U}\}$.*

With a slight abuse of notation, we overload the χ function. The $\mathtt{U}$ flag identifies an uncontrollable element, therefore the flagged time instant is not under the control of the agent. Instead, the $\mathtt{C}$ flag identifies controllable elements. Consider again the running example. If we flag both the states of the satellite with $(\mathtt{U}, \mathtt{U})$ and all the rest as controllable, we are modeling a situation in which the satellite visibility is not decidable by the communicator, the only possible assumption is the minimal and maximal durations.

We now define what is a planning problem.

Definition 5. *Let $\mathcal{G}$ be a generator set, Σ a set of synchronizations over the generators in $\mathcal{G}$, $\mathcal{F}$ and $\mathcal{R}$ be sets of unary and binary facts, respectively. Let χ be a CU-annotation. A* timeline controllability problem *P is a tuple $(\mathcal{G}, \Sigma, \mathcal{F}, \mathcal{R}, \chi)$.*

In this work possible solutions are time-triggered plans, defined as follows.

Definition 6. *A* time-triggered plan *is a (possibly infinite) sequence $(G_1, v_1, cmd_1, t_1); (G_2, v_2, cmd_2, t_2); \ldots$ where, for all $i \geq 1$, v_i is a value for G_i, $cmd_i \in \{\mathtt{S}, \mathtt{E}\}$, and $t_i \leq t_{i+1}$.*

Intuitively, at a specific time point, a time triggered plan may specify one or more start/end commands to be executed on a specific generator and value. This definition is syntactic; the executability of a time-triggered plan is defined at the semantic level.

Semantics. In the following we assume that a timeline description is given. We provide an interpretation of timelines by means of streams, i.e. possibly infinite sequences of time-labeled activity instances.

Definition 7. *Let $G = (V, T, \delta)$ be a generator. A* stream *S for G is a (possibly infinite) sequence $(v_1, d_1); (v_2, d_2); \ldots$ such that, for all $i \geq 1$, $v_i \in V$, $(v_i, v_{i+1}) \in T$, $d_i \in \delta(v_i)$.*

Given a stream S, we use the following notation: $Value(S, i) = v_i$; $StartTime(S, i) = \sum_{j=1}^{i-1} d_j$; $EndTime(S, i) = StartTime(S, i) + d_i$; $Interval(S, i) = (StartTime(S, i), EndTime(S, i))$.

We can now define the compatibility of a stream with the problem constraints.

Definition 8. *Let $G_0, \ldots, G_n$ be generators, and let $\sigma = ((G_0, v_0), \{(G_1, v_1), \ldots, (G_n, v_n)\}, C)$ be a synchronization. For $0 \leq i \leq n$, let S_i be a stream for G_i. $\{S_0, \ldots, S_n\}$* fulfills *σ iff for all j_0 such that $(Value(S_0, j_0) = v_0)$, there exist $j_1, \ldots, j_n$ such that for every constraint $(v_h \bowtie v_k) \in C$, $Interval(S_h, j_h) \bowtie Interval(S_k, j_k)$ holds.*

Notice that, in general, n-ary synchronizations, cannot be expressed in terms of binary synchronizations only. This is true only in the case where each Allen constraint involves one value from G_0 and one from another G_i. In the case of constraints between G_i and G_j, with $i, j > 0$ a binding between the activities in G_i and G_j is introduced, but the binding is further constrained by G_0.

Definition 9. *Let G be a generator, and let S be a stream for G. S fulfills the unary fact (G, v, I_s, I_e) at i iff $Value(S, i) = v$, $StartTime(S, i) \in I_s$ and $EndTime(S, i) \in I_e$.*

Definition 10. *Let $f_1 \bowtie f_2$ be a binary fact, where $f_i \stackrel{\text{def}}{=} (G_i, v_i, I_{s_i}, I_{e_i})$. Let S_1 and S_2 be streams for G_1 and G_2 respectively. S_1 and S_2 fulfill $f_1 \bowtie f_2$ iff S_1 fulfills f_1 at i_1, S_2 fulfills f_2 at i_2, and $Interval(S_1, i_1) \bowtie Interval(S_2, i_2)$.*

Definition 11. *A time-triggered plan $(G_1, v_1, cmd_1, t_1); (G_2, v_2, cmd_2, t_2); \ldots$ induces a stream S on $G = (V, T, \delta)$ iff for all $i \geq 1$, when $G = G_i$, there exists $j \geq 1$ such that (1) if $cmd_i = \mathrm{S}$ then $StartTime(S, j) = t_i$, and (2) if $cmd_i = \mathrm{E}$ then $EndTime(S, j) = t_i$.*

Definition 12. *A time triggered plan $(G_1, v_1, cmd_1, t_1); (G_2, v_2, cmd_2, t_2); \ldots$ obeys a CU-annotation χ iff for each $i \geq 1$, (1) if $cmd_i = \mathrm{S}$ then $\chi(G_i, v_i) \in \{(\mathrm{C}, \mathrm{C}), (\mathrm{C}, \mathrm{U})\}$, and (2) if $cmd_i = \mathrm{E}$ then $\chi(G_i, v_i) \in \{(\mathrm{C}, \mathrm{C}), (\mathrm{U}, \mathrm{C})\}$.*

Intuitively, this means that each assigned time point is labeled as controllable.

Definition 13. *Let π be a time-triggered plan, χ a CU-annotation and $\mathcal{G}$ the set of generators controlled by π. π is* complete *with respect to χ iff for each $G \in \mathcal{G}$, for each stream $S = (v_1, d_1); (v_2, d_2); \ldots$ of G induced by π and for each i: (1) if $\chi(G, v_i) \in \{(\mathrm{C}, \mathrm{C}), (\mathrm{C}, \mathrm{U})\}$ then $(G, v_i, \mathrm{S}, StartTime(S, i)) \in \pi$; (2) if $\chi(G, v_i) \in \{(\mathrm{C}, \mathrm{C}), (\mathrm{U}, \mathrm{C})\}$ then $(G, v_i, \mathrm{E}, EndTime(S, i)) \in \pi$.*

In other words, if π is complete, each controllable time point of an induced stream S is assigned by π.

Definition 14. *Given the CU-annotation χ, a stream $(v_1, d_1); (v_2, d_2); \ldots$ for generator $G = (V, T, \delta)$ is said to* satisfy contingencies *of G iff for each $i \geq 1$, $v_i \in V$, $(v_i, v_{i+1}) \in T$ and if $\chi(G_i, v_i) \in \{(\mathrm{U}, \mathrm{U}), (\mathrm{U}, \mathrm{C})\}$ then $d_i \in \delta(v_i)$.*

In other words, a stream satisfies the contingencies of a generator if it is compatible with the generator constraints on the uncontrollable values.

Definition 15 (Solution to strong controllability problem)**.** *A time-triggered plan π is a strong* solution *for $P = (\mathcal{G}, \Sigma, \mathcal{F}, \mathcal{R}, \chi)$ iff it obeys and is complete w.r.t χ, and all the streams induced by π that are compatible with the consistencies of the generators in $\mathcal{G}$ and that fulfill the synchronizations labeled as u, also fulfill each generator, the rest of Σ, $\mathcal{F}$ and $\mathcal{R}$.*

Intuitively we are searching for a plan that constrains the execution in such a way that for every possible evolution of the uncontrollable parts (fulfilling the assumed contingencies), all the problem constraints are satisfied.

In practice, we are interested in finding solutions to a strong controllability problem within a given temporal horizon H.

Definition 16 (Bounded solution to strong controllability problem)**.** *A finite time-triggered plan π is a* strong bounded solution *for $P = (\mathcal{G}, \Sigma, \mathcal{F}, \mathcal{R}, \chi)$ for a time horizon $H \in \mathbb{R}^+$ iff the following conditions hold: (1) π obeys and is complete w.r.t χ; (2) all the streams compatible with π finish after H; (3) each stream S that is compatible with the contingencies of the generators in $\mathcal{G}$ and that satisfies the synchronizations labeled as u, also satisfies the generator constraints, $\mathcal{F}$, $\mathcal{R}$, and the rest of Σ is satisfied for every interval of S that ends before H.*

Note that we chose to impose no constraint on interval that end after the horizon, but other semantics are possible. We highlight that searching for a time-triggered plan means searching for a fixed assignment of controllable decisions in time. For instance, in the satellite example it is possible to produce a time triggered plan for sending each packet once as shown in Figure 2. However, it is not possible to send more packets, because the uncertainty in the satellite compresses the guaranteed visibility window. Consider again Figure 2, the next guaranteed visibility window of the satellite would be $[58, 60)$ that is too short for sending another packet.

Bounded Encoding in FOL

We now reduce the problem of finding a solution for a bounded strong controllability problem to a SMT problem. Intuitively, we aim at finding a *finite* sequence of intervals, that completely covers the time-span between 0 and the horizon H, that fulfill all the problem constraints. Note that no synchronization constraints are imposed on intervals that end after the horizon bound. The underlying idea is to logically model a set of bounded streams and to impose the problem constraints on the streams. If the resulting formula is satisfiable, it means that a model for the formula codifies a stream that witnesses a solution for the original problem.

Let $H \in \mathbb{R}^+$ be the horizon, a generator $G = (V, T, \delta)$ is associated with a maximum number of intervals (assuming each $\delta(v) > 0$). A coarse upper bound M_G is given dividing H by the minimal duration associated with any value in V: $M_G = \lceil \frac{H}{\min_{v \in V} start(\delta(v))} \rceil$.

We use two set of variables for each generator G: $ValueOf^G(j)$ and $EndOf^G(j)$, whose interpretation defines the stream for G. $ValueOf^G(j)$ gives the value of the j-th interval, while $EndOf^G(j)$ encodes the end time point of the j-th interval. Thus, for each generator $G = (V, T, \delta)$, we can use M_G variables $ValueOf^G(j)$ ranging over the domain V, and M_G variables $EndOf^G(j)$ of type $\mathbb{R}^+$ to model a bounded stream that is guaranteed to cover the interval $[0, H]$.

$EndOf^G(j)$ defines time points in which the stream changes its value. Unfortunately, whether a time point is controllable or not cannot be detected statically in general. In fact, depending on the discrete path encoded in the assignments to $ValueOf^G(j)$, the j-th time point can be either controllable or uncontrollable. For this reason, we have to introduce M_G new variables, called $U^G(j)$, that model the uncertain values (analogous to $EndOf^G(j)$). In order to properly capture the strong controllability of the execution, we consider $EndOf^G(j-1)$ and $ValueOf^G(j)$ as existentially-quantified variables, and $U^G(j)$ as universally quantified variables. We indicate with U^G the set of all the $U^G(j)$ variables. In order to impose the proper constraints on either $EndOf^G(j)$ or $U^G(j)$ we have to condition the constraint on the controllability of the j-th interval that is decided at solving time. Therefore we introduce two macros $S^G(j, U^G)$ and $E^G(j, U^G)$ that encapsulate this conditioning and return the proper value that encodes the start or the end of the j-interval respectively. The first formula, $S^G(j, U^G)$, is defined as[2] $ite(j = 0, 0, ite(\chi(G, ValueOf^G(j)) \in \{(\mathrm{C}, \mathrm{C}), (\mathrm{C}, \mathrm{U})\}, EndOf^G(j-1), U^G(j-1)))$. Similarly, $E^G(j, U^G)$ is defined as $ite(\chi(G, ValueOf^G(j)) \in \{(\mathrm{C}, \mathrm{C}), (\mathrm{U}, \mathrm{C})\}, EndOf^G(j), U^G(j))$.

Let $Used^G(j)$ be the predicate defined as $E^G(j, U^G) \leq H$. The encoding is defined as follows. For each generator $G = (V, T, \delta)$, we define $Value_G \stackrel{\text{def}}{=} \bigwedge_{j=1}^{M_G} ValueOf^G(j) \in V$ to force the domain of $ValueOf^G(j)$ and $Trans_G \stackrel{\text{def}}{=} \bigwedge_{j=1}^{M_G-1} T(ValueOf^G(j), ValueOf^G(j+1))$ to codify the transition relation of G.

We split the constraints encoding the interval durations in two distinct formulae as follows.

$$\begin{aligned}\Gamma_G(U^G) = \bigwedge_{j=1}^{M_G}((\chi(G, ValueOf^G(j)) \in \{(\mathrm{C}, \mathrm{U}), (\mathrm{U}, \mathrm{U})\}) \rightarrow \\ (E^G(j, U^G) - S^G(j, U^G) \in \delta(ValueOf^G(j)))) \\ \Psi_G(U^G) = \bigwedge_{j=1}^{M_G}((\chi(G, ValueOf^G(j)) \in \{(\mathrm{C}, \mathrm{C}), (\mathrm{U}, \mathrm{C})\}) \rightarrow \\ (E^G(j, U^G) - S^G(j, U^G) \in \delta(ValueOf^G(j))))\end{aligned}$$

For every uncontrollable synchronization $\sigma = ((G_0, v_0), \{(G_1, v_1), \ldots, (G_n, v_n)\}, C)$ $(\chi(\sigma) = \mathrm{U})$, we define $\Gamma_\sigma(U^{G_0}, \ldots, U^{G_n})$ as follows.

$$\begin{aligned}&\bigwedge_{j_0=1}^{M_{G_0}}(ValueOf^{G_0}(j_0) = v_0 \wedge Used^{G_0}(j_0)) \rightarrow \\ &\quad (\bigvee_{j_1=1}^{M_{G_1}}(ValueOf^{G_1}(j_1) = v_1 \wedge Used^{G_1}(j_1)) \wedge \ldots \\ &\quad\quad (\bigvee_{j_n=1}^{M_{G_n}}(ValueOf^{G_n}(j_n) = v_n \wedge Used^{G_n}(j_n)) \wedge \\ &\quad\quad\quad \bigwedge_{v_k \bowtie v_h \in C} \xi(\bowtie, S^{G_k}(j_k, U^{G_k}), E^{G_k}(j_k, U^{G_k}), \\ &\quad\quad\quad\quad S^{G_h}(j_h, U^{G_h}), E^{G_h}(j_h, U^{G_h}))) \ldots)\end{aligned}$$

Where $\xi(\bowtie, s_1, e_1, s_2, e_2)$ is the LRA encoding of the Allen constraint $I_1 \bowtie I_2$ with the interval I_i being (s_i, e_i). We also define $\Psi_\sigma(U^{G_0}, \ldots, U^{G_n})$ in the very same way for each controllable synchronization $(\chi(\sigma) = \mathrm{C})$. The formula encoding unary facts is obtained by imposing the existence of a compatible interval in the considered stream.

[2] $ite(\psi, \phi_1, \phi_2)$ is a shortcut for $(\psi \rightarrow \phi_1) \wedge (\neg\psi \rightarrow \phi_2)$.

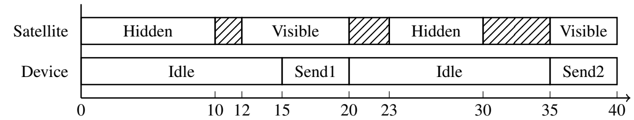

Figure 2: An execution of the satellite example that fulfills the problem constraints. The striped regions are uncertain: depending on the actual duration of the intervals the satellite can be either in Hidden or in Visible state.

For each unary fact $f = (G, v, I_s, I_e)$ we define $\Psi_f(U^G)$ as $\bigvee_{j=1}^{M_G} Fact(U^G, j)$ where $Fact(U^G, j)$ is $Used^G(j) \wedge (ValueOf^G(j) = v) \wedge S^G(j, U^G) \in I_s \wedge E^G(j, U^G) \in I_e$

For every binary fact requirement $r = f_1 \bowtie f_2$, where $f_i = (G_i, v_i, I_{s_i}, I_{e_i})$ we define $\Psi_r(U^{G_1}, U^{G_2})$ as $\bigvee_{j_1=1}^{M_{G_1}} \bigvee_{j_2=1}^{M_{G_2}}(Fact(U^{G_1}, j_1) \wedge Fact(U^{G_2}, j_2) \wedge \xi(\bowtie, S^{G_1}(j_1, U^{G_1}), E^{G_1}(j_1, U^{G_1}), S^{G_2}(j_2, U^{G_2}), E^{G_2}(j_2, U^{G_2})))$.

Finally, let Σ_u be the subset of Σ of the uncontrollable synchronizations and let Σ_c be Σ / Σ_u. The overall encoding for the problem is:

$$\begin{aligned}&\bigwedge_{G \in \mathcal{G}} Value_G \wedge \bigwedge_{G \in \mathcal{G}} Trans_G \wedge \forall U^{G_0}, \ldots, U^{G_n}. \\ &\quad ((\bigwedge_{G \in \mathcal{G}} \Gamma_G(U^G) \wedge \bigwedge_{\sigma \in \Sigma_u} \Gamma_\sigma(U^{G_0}, \ldots, U^{G_n})) \rightarrow \\ &\quad\quad (\bigwedge_{G \in \mathcal{G}} \Psi_G(U^G) \wedge \bigwedge_{\sigma \in \Sigma_c} \Psi_\sigma(U^{G_0}, \ldots, U^{G_n}) \wedge \\ &\quad\quad\quad \bigwedge_{f=(G, v, I_s, I_e) \in \mathcal{F}} \Psi_f(U^G) \wedge \\ &\quad\quad\quad \bigwedge_{r=(G_1, v_1, I_{s_1}, I_{e_1}) \bowtie (G_2, v_2, I_{s_2}, I_{e_2}) \in \mathcal{R}} \Psi_r(U^{G_1}, U^{G_2}))).\end{aligned}$$

The universal quantification captures the "universality" of the solution: for each possible allocation of the uncontrollables, given by $U^{G_0}, \ldots, U^{G_n}$, we impose that the contingent part of the problem implies the requirements. The encoding admits a model iff there exist a bounded solution to the original problem and the model can be used to build a complete time-triggered plan for the original bounded strong controllability problem. This formula is a first-order quantification over a finite set of real variables. Therefore, it can be decided by a SMT(LRA) solver equipped with a quantifier elimination procedure.

Related Work

This work is most closely related to two research lines: timeline-based planning, and temporal problems with uncertainty.

The literature on timeline-based planning is extensive, starting from the seminal work described in (Muscettola 1993), and including the APSI framework (Cesta et al. 2009a; Cesta, Fratini, and Pecora 2008; Cesta et al. 2009b), the EUROPA framework (Frank and Jónsson 2003) with its formalization in (Bernardini 2008), and (Verfaillie, Pralet, and Lemaître 2010). The key difference of our work is that we provide a full formal account of timelines with temporal uncertainty, while the frameworks mentioned above assume controllable duration of activities. The problem addressed in this paper, requiring universal quantifications to model the effect of an "adversarial" environment, is significantly harder than the "consistency" problem, where quantification over (start and end) time points is only existential. It is important to emphasize the that problem of finding flexible timelines (Cesta et al. 2010b; 2010a) is very different from the one solved here: timeline flexibility demands the scheduling of the activities to the executor, but does not guarantee goal achievement in a temporally uncertain domain, with uncontrollable durations. Finally, we mention that we use a dense-time interpretation: we represent time points as real variables, while APSI and EUROPA use integers.

There are various extensions of temporal problems with uncertainty, starting from (Vidal and Fargier 1999), to strong (Peintner, Venable, and Yorke-Smith 2007; Cimatti, Micheli, and Roveri 2012a), weak (Venable et al. 2010; Cimatti, Micheli, and Roveri 2012b), and dynamic controllability (Morris, Muscettola, and Vidal 2001). In temporal problems, the number of instances of activities is known a priori. This is a key difference with the work discussed here, where determining the right type and number of activities is part of the problem.

IxTeT (Ghallab and Laruelle 1994) is a temporal planning system that is able to deal with temporal uncertainty. Differently from our approach, IxTeT does not produce robust plans. The approach separates planning and scheduling, by demanding to the plan executor the on-line solution of the dynamic controllability of a temporal problem with uncertainty.

For completeness, we also contrast our work with the (less related) work on planning for durative actions based on PDDL (Coles et al. 2012; 2009). The first difference is implicit in the two modeling paradigms – for example, timelines can naturally express temporally extended goals. More importantly, planning for durative actions (Coles et al. 2012; 2009) assumes that the duration of actions is controllable.

Evaluation

Implementation. The approach described in previous sections was implemented in the first (sound and complete) decision procedure for timelines with uncertainty. We developed a tool chain that uses an APSI-like syntax for specifying the planning domain and problem, with an extension for CU-annotations of generator states and synchronizations.

The implementation uses the state-of-the-art MathSAT SMT solver (Cimatti et al. 2012) as a backend, and a quantifier elimination procedure based on the Loos-Weispfenning method (Loos and Weispfenning 1993).

The tool implements the encoding described in Section , in the following referred to as "Monolithic". We also implemented an "Incremental" approach, where the incrementality feature of the SMT solver is exploited[3]. The idea is to limit the number of considered intervals for a generator (M_G), thus resulting in a smaller and easier formula to decide. If the check returns a plan, then the algorithm can terminate, otherwise more intervals are considered, until we reach the M_G for each generator. This solution often avoids

Type	Problem	Monolithic		Incremental	
		Time(s)	Memory(Mb)	Time(s)	Memory(Mb)
Sat	Satellite	6.87	111.5	1.88	31.9
	Machinery1	TO	TO	360.15	611.5
	Meeting	MO	MO	182.52	1897.0
Unsat	Satellite	7.17	126.2	171.25	147.6
	Machinery2	104.86	253.7	113.53	284.4
	Meeting	23.12	630.8	105.17	776.9

Table 1: Experimental results.

submitting to the solver the (bigger) formula corresponding to the whole problem.

Both approaches were optimized by applying a rewriting similar to the one described in (Cimatti, Micheli, and Roveri 2012a): the formula of the bounded encoding has the form $\forall \vec{x}.\Gamma(\vec{x}) \rightarrow \bigwedge_h \psi_h(\vec{x})$ and can be equivalently rewritten as $\bigwedge_h \forall \vec{x}.\Gamma(\vec{x}) \rightarrow \psi_h(\vec{x})$. This is often useful, since a large number of small quantifications may be preferable to a single Monolithic one.

Experiments. No competitor approach is available for the problem that we address. For this reason, we compare the performances of the Monolithic approach and the Incremental one.

We considered problems on three different domains, and ran several (solvable and unsolvable) problems, with various time horizons. The results, reported in Table 1, show that the Incremental approach outperforms the Monolithic one in all the instances that have a solution. On unsolvable instances, the Monolithic approach is superior. This is because the Incremental version has to solve increasingly difficult problems, and in order to prove the unsatisfiability the algorithm ends up solving the same problem the Monolithic approach solved in the first place. In sat-instances, instead, the Incremental algorithm can early-terminate without having to solve the big quantifications needed for the termination proof. For the lack of space we omit the description of the used domains and problems. An archive containing the runnable tool, the tested instances and the relative explanations can be found at `https://es.fbk.eu/people/amicheli/resources/aaai13`.

Conclusions

We presented a formal foundation for planning with timelines under temporal uncertainty, where the duration of actions can not be controlled by the executor. We proposed the first decision procedure that is able to produce time-triggered plans satisfying the problem constraints regardless of temporal uncertainty, and implemented it leveraging the power of SMT solvers.

In the future, we plan to extend the framework to encompass resource consumption, to investigate more efficient quantifier-elimination techniques and other forms of lazy encoding, and to address the production of conditional plans that are robust with respect to temporal uncertainty.

[3]In an incremental setting, a solver instance can be queried for the satisfiability of a formula and then clauses can be pushed or popped to obtain another formula that can be decided, possibly recycling parts of the previous search.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communication of the ACM* 26(11):832–843.

Angelsmark, O., and Jonsson, P. 2000. Some observations on durations, scheduling and allen's algebra. In *CP*, 484–488.

Barreiro, J.; Boyce, M.; Do, M.; Frank, J.; Iatauro, M.; Kichkaylo, T.; Morris, P.; Ong, J.; Remolina, E.; Smith, T.; and Smith, D. 2012. EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. In *Proc. of 4th International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS)*.

Barrett, C. W.; Sebastiani, R.; Seshia, S. A.; and Tinelli, C. 2009. Satisfiability modulo theories. In *Handbook of Satisfiability*. IOS Press. 825–885.

Bernardini, S. 2008. *Constraint-based Temporal Planning: Issues in Domain Modelling and Search Control*. Ph.D. Dissertation, Faculty of Computer Science – University of Trento (Italy).

Cesta, A.; Fratini, S.; Oddi, A.; and Pecora, F. 2008. APSI Case#1: Pre-planning Science Operations in Mars Express. In *Proceedings of i-SAIRAS-08, the 9th International Symposium in Artificial Intelligence, Robotics and Automation in Space*.

Cesta, A.; Cortellessa, G.; Fratini, S.; Oddi, A.; and Rasconi, R. 2009a. The APSI Framework: a Planning and Scheduling Software Development Environment. In *Working Notes of the ICAPS-09 Application Showcase Program*.

Cesta, A.; Cortellessa, G.; Fratini, S.; and Oddi, A. 2009b. Developing an end-to-end planning application from a timeline representation framework. In Haigh, K. Z., and Rychtyckyj, N., eds., *IAAI*. AAAI.

Cesta, A.; Finzi, A.; Fratini, S.; Orlandini, A.; and Tronci, E. 2010a. Analyzing flexible timeline-based plans. In Coelho, H.; Studer, R.; and Wooldridge, M., eds., *ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, 471–476. IOS Press.

Cesta, A.; Finzi, A.; Fratini, S.; Orlandini, A.; and Tronci, E. 2010b. Validation and verification issues in a timeline-based planning system. *Knowledge Eng. Review* 25(3):299–318.

Cesta, A.; Cortellessa, G.; Fratini, S.; and Oddi, A. 2011. Mrspock - steps in developing an end-to-end space application. *Computational Intelligence* 27(1):83–102.

Cesta, A.; Fratini, S.; and Pecora, F. 2008. Planning with multiple-components in omps. In Nguyen, N. T.; Borzemski, L.; Grzech, A.; and Ali, M., eds., *IEA/AIE*, volume 5027 of *Lecture Notes in Computer Science*, 435–445. Springer.

Cheng, C.-C., and Smith, S. F. 1994. Generating feasible schedules under complex metric constraints. In Hayes-Roth, B., and Korf, R. E., eds., *AAAI*, 1086–1091. AAAI Press / The MIT Press.

Cimatti, A.; Griggio, A.; Sebastiani, R.; and Schaafsma, B. 2012. The MathSAT5 SMT solver. `http://mathsat.fbk.eu`.

Cimatti, A.; Micheli, A.; and Roveri, M. 2012a. Solving temporal problems using smt: Strong controllability. In *CP*, 248–264.

Cimatti, A.; Micheli, A.; and Roveri, M. 2012b. Solving temporal problems using smt: Weak controllability. In *AAAI*.

Coles, A.; Fox, M.; Halsey, K.; Long, D.; and Smith, A. 2009. Managing concurrency in temporal planning using planner-scheduler interaction. *Artif. Intell.* 173(1):1–44.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2012. Colin: Planning with continuous linear numeric change. *J. Artif. Intell. Res. (JAIR)* 44:1–96.

Donati, A.; Policella, N.; Cesta, A.; Fratini, S.; Oddi, A.; Cortellessa, G.; Pecora, F.; Schulster, J.; Rabenau, E.; Niezette, M.; and Steel, R. 2008. Science Operations Pre-Planning & Optimization using AI constraint-resolution - the APSI Case Study 1. In *Proceedings of SpaceOps-08, the 10th International Conference on Space Operations*.

Drakengren, T., and Jonsson, P. 1997. Eight maximal tractable subclasses of allen's algebra with metric time. *J. Artif. Intell. Res. (JAIR)* 7:25–45.

Frank, J., and Jónsson, A. 2003. Constraint-based Attribute and Interval Planning. *Constraints* 8(4):339–364.

Ghallab, M., and Laruelle, H. 1994. Representation and control in ixtet, a temporal planner. In *AIPS*, 61–67.

Loos, R., and Weispfenning, V. 1993. Applying linear quantifier elimination. *Computer Journal* 36(5):450–462.

Monniaux, D. 2008. A Quantifier Elimination Algorithm for Linear Real Arithmetic. In *Logic for Programming, Artificial Intelligence, and Reasoning - LPAR*, 243–257.

Morris, P. H.; Muscettola, N.; and Vidal, T. 2001. Dynamic control of plans with temporal uncertainty. In *International Joint Conference on Artificial Intelligence - IJCAI*, 494–502.

Muscettola, N. 1993. Hsts: Integrating planning and scheduling. Technical report, DTIC Document.

Peintner, B.; Venable, K. B.; and Yorke-Smith, N. 2007. Strong controllability of disjunctive temporal problems with uncertainty. In Bessiere, C., ed., *Principles and Practice of Constraint Programming - CP*, volume 4741 of *LNCS*, 856–863. Springer.

Schrijver, A. 1998. *Theory of Linear and Integer Programming*. J. Wiley & Sons.

Venable, K. B.; Volpato, M.; Peintner, B.; and Yorke-Smith, N. 2010. Weak and dynamic controllability of temporal problems with disjunctions and uncertainty. In *Workshop on Constraint Satisfaction Techniques for Planning & Scheduling*, 50–59.

Verfaillie, G.; Pralet, C.; and Lemaître, M. 2010. How to model planning and scheduling problems using constraint networks on timelines. *Knowledge Eng. Review* 25(3):319–336.

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

Wetprasit, R., and Sattar, A. 1998. Temporal reasoning with qualitative and quantitative information about points and durations. In Mostow, J., and Rich, C., eds., *AAAI/IAAI*, 656–663. AAAI Press / The MIT Press.