

# A Timed Temporal Logic for Specifying Scheduling Problems (Extended Report)

TECHNICAL REPORT CSRG-629, Department of Computer Science, University of Toronto

Roy Luo\*, J. Christopher Beck†, Sheila A. McIlraith\*

\*Department of Computer Science, University of Toronto

†Department of Mechanical and Industrial Engineering, University of Toronto

\*{royluo,sheila}@cs.toronto.edu, †jcb@mie.utoronto.ca

## Abstract

As real-world scheduling applications diversify in scope and grow in complexity, there is an increasing need for domain-independent languages and reasoning systems that support their specification and analysis. In this paper we explore Metric Temporal Logic (MTL) as a language to support the specification of complex scheduling patterns including repeated and conditional occurrences of activities and rich temporal relationships among them. We modify the standard MTL semantics to better suit scheduling problems, and explore a series of natural restrictions to the language to gain tractability. In so doing, we provide a standard framework through which a variety of scheduling problems, from temporal networks and job shop scheduling to temporal planning, can be related and generalized. We also provide an algorithm to find a schedule specified as an MTL formula, and establish the equivalence between a fragment of MTL and simple temporal networks (STNs).

## 1 Introduction

The need and opportunity for fully automated and mixed-initiative scheduling has exploded as automation plays an increasing role in end-to-end business practices, and as sophisticated calendaring tools become broadly adopted. The availability and ease of sharing of computer-interpretable scheduling constraints and partial schedules enables a broad array of new scheduling problems that go beyond classic problems such as job shop scheduling (Fox 1983) to include sport team rotations, car pooling, and other personal and commercial applications. With these new problems come new challenges in the specification and realization of scheduling problems.

Scheduling problems have traditionally been specified via a finite set of activities, temporal constraints, and resource requirements and capacities. Increasingly, there is a desire to express richer problems involving repeated patterns of occurrence of various activities, some conditional, some quantity bounded (e.g. *always schedule nurses four days on followed by two days off or check on each patient every two to three hours*). While specialized approaches to some classes of patterns exist, for example, as global grammar constraints in constraint-based scheduling (Pesant 2004), there does not appear to be a formal language with the flexibility to allow specification of a variety of rich scheduling patterns.

There is also a desire to characterize activities and resources in terms of their properties (e.g. *always schedule at least one Mandarin-speaking nurse on each rotation*). Such properties support ad hoc and implicit groupings of activities, the specification of constraints by referring to unnamed activities and resources, and lifted reasoning about groups rather than explicit individual objects.

Although many advances have been made in efficiently solving a wide range of *specialized* scheduling problems, real-world situations are often too complex to be easily or accurately represented as instances of such specialized problems. Burke et al. [2004] present an extensive review of approaches to the myriad variants of ‘the nurse rostering problem’, but conclude that “there is a definite gap between much of the current state of the art in nurse scheduling research and the demanding and challenging requirements of today’s hospital environments.” A key challenge pointed to is the ability to specify and reason about the tremendous variety of constraints encountered in real-world nurse rostering problems.

In this paper, we adapt Metric Temporal Logic (MTL) (Koymans 1990), an extension of Linear Temporal Logic (LTL) used in model-checking and verification of concurrent and reactive systems, to the scheduling domain. We augment the syntax and modify the semantics of MTL to more easily represent rich scheduling patterns and properties of named and unnamed activities. In Sections 2 and 3 we present the syntax and altered semantics of MTL. To trade off expressiveness for tractability, in Section 4 we explore novel restrictions to MTL that are conducive for scheduling. We then describe an algorithm for finding satisfying schedules under these restrictions in Section 5. In Section 6 we demonstrate the utility of the language to express several well-known scheduling problems and prove a novel equivalence between a fragment of MTL and simple temporal networks. Finally, we summarize our complexity results and related work in Section 7.

## 2 A Language for Scheduling

First, we define a general template for what we consider to be a scheduling problem.

**Definition 2.1.** *A scheduling problem is a tuple  $\langle A, D, R, C, TC \rangle$ , where:*

- $\mathcal{A}$  is a set of durative activities,
- $\mathcal{D}$  is a function from activities to their durations,
- $\mathcal{R}$  is a relation defining activity resource requirements,
- $\mathcal{C}$  is a function from resources to maximum capacities,
- $\mathcal{TC}$  is a set of temporal constraints between activities.

Our intention is to create a logical formula that is satisfiable iff the corresponding scheduling problem has a solution, and we prove this approach correct for a few existing scheduling problems. A solution to a scheduling problem is a schedule, i.e. an assignment of start and end times to each activity such that the durations, resource capacities, and temporal constraints are all satisfied. While the activities and durations are fairly standard across all scheduling problems, resource and temporal constraints (i.e.,  $\mathcal{R}$ ,  $\mathcal{C}$ , and  $\mathcal{TC}$ ), vary depending on the type of problem. For example, in job shop scheduling there are unary capacity, differentiated resources (a machine can process one job at a time and a job may require a particular machine) and simple precedence constraints between jobs. This paper will demonstrate a method of modeling unary capacity, differentiated resources, but will not explore resource constraints beyond that. Instead, we focus on extending the expressivity of the temporal constraints to model rich temporal relationships.

To this end, we extend our concept of activities to allow them to occur more than once. Our basic object is the (possibly repeating) activity  $A \in \mathcal{A}$  (denoted by upper case), which the schedule maps to a set of instances  $\{a_1 \dots a_k\} \subseteq \mathcal{I}$  (denoted by lower case), where  $\mathcal{I}$  is the set of all instances of  $\mathcal{A}$ . For example, in the nurse rostering domain, an activity could refer to a particular nurse’s shifts over a time horizon, with each instance representing an actual scheduled shift.

In order to represent these concepts in temporal logic, we begin with a review of the standard propositional MTL.

**Definition 2.2.** Let  $\mathbb{D}$  be a domain ( $\mathbb{Z}$  or  $\mathbb{R}$ ), and  $I$  be an interval in  $\mathbb{D} \cup \{-\infty, \infty\}$ . A Metric Temporal Logic formula is a well-formed formula in the following grammar:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \mathcal{U}_I \phi$$

where  $p$  is an atomic proposition (Koymans 1990).

We now introduce our own specialization of MTL and follow with an intuitive explanation of the various operators mentioned; a formal semantics is given in the next section.

**Definition 2.3.** Let  $\mathcal{A}$  be a set of activities. A Scheduling MTL formula is a MTL formula such that the set of atomic predicates consists of propositions of the form  $\{\text{start}(A), \text{end}(A), P(A)\}$  for some activity  $A$  or property  $P$  from a set of properties  $\mathcal{P}$ . Additionally, well-formed formulae include those of the form  $\forall x.\phi$ ; if  $\phi$  is in the scope of a quantified variable  $x$ , atomic predicates also include  $\text{start}(x)$ ,  $\text{end}(x)$ , and  $P(x)$  (we omit a formal treatment of the syntax of quantified variables here).

From the repeating activity  $A$  we define the predicates  $\text{start}(A)$  and  $\text{end}(A)$ . Their truth value is dependent on the time the predicate is evaluated:  $\text{start}(A)$  (resp.  $\text{end}(A)$ ) is true at a given time if some instance of  $A$  starts (resp. ends) at that time. This notion is formalized below.

We associate properties to activities in order to speak generally about all activities that share some attribute. A property  $P$  is defined as a subset of  $\mathcal{A}$ ; the elements of  $P$  are understood to share that property. These properties translate to unary predicates  $P(A)$  in the language. Given a set of activities representing nurses’ shifts, one could define a subset of them to have the property “trainee” signifying that the shift is performed by a nurse in training. Then, we could formulate a Scheduling MTL constraint stating that if a trainee shift is scheduled, a shift from an experienced nurse must also be scheduled at the same time. The semantics in the following section will show that the formula  $\Box(\exists x.(Trainee(x) \wedge Currently(x)) \supset \exists y.(\neg Trainee(y) \wedge Currently(y)))$  captures this rule. Properties may also be used for specifying the resource that a set of activities require. One can then represent resource capacity by limiting the number of activities with the corresponding property that can occur at any given time.

In order to utilize properties and repeating activities in a meaningful way, it is useful to quantify over instances of activities. For example, if we would like to say that every instance of a checkup activity  $A$  takes 10-15 minutes, we can state that *for all* instances  $a$  of  $A$ , the  $\text{start}(a)$  is followed in 10-15 time units by  $\text{end}(a)$ . Quantifiers also allow us to make statements on events that all share a property, as illustrated in the trainee shift example above. This is not provided in existing scheduling specification languages but is seamlessly accommodated in MTL.

Finally, the MTL operator “until” augmented with a time interval ( $\phi \mathcal{U}_I \psi$ ) states that  $\phi$  is true from now until some point within  $I$  time units, when  $\psi$  becomes true. We also make liberal use of two operators derived from  $\mathcal{U}_I$ :  $\Diamond_I \phi$  and  $\Box_I \phi$ , which state that “at some point within interval  $I$  after the current time,  $\phi$  will be true” and “at every point within interval  $I$  after the current time,  $\phi$  is true”, respectively.

It is worth noting that in addition to serving as a problem specification language that can be used as input to a scheduling algorithm, MTL’s logical foundation enables us to reason about properties of a MTL-specified scheduling problem at the logical level and to answer queries about the schedules it entails without necessarily calculating specific solutions. This is critical tool in scheduling requirements engineering and verification.

### 3 Semantics

Here, we formally define the notions introduced above.

A schedule  $\mathcal{T}$  consists of a tuple  $(Instances, T)$  and is the object we will use to evaluate the truth value of an Scheduling MTL formula.  $Instances : \mathcal{A} \mapsto 2^{\mathcal{I}}$  is a function that maps an activity to the set of instances of that activity that actually occur.  $T$  is a function that assigns times to the start and end of each instance of each activity, i.e.  $T : \{a \mid a \in \{Instances(A) \mid A \in \mathcal{A}\}\} \mapsto \mathbb{D} \times \mathbb{D}$ . Let  $T_s(a)$  and  $T_e(a)$  return the first and second values of  $T(a)$  respectively. A formula  $\phi$  is satisfied by a schedule  $\mathcal{T}$  at a time  $t_i \in \mathbb{D}$ , which is written as  $\langle \mathcal{T}, t_i \rangle \models \phi$ .  $\top$  (true),  $\perp$  (false), and boolean operators are defined in the standard way. The rest of the operators are defined as follows:

- $\langle \mathcal{T}, t_i \rangle \models \text{start}(A)$  (resp.  $\text{end}(A)$ ) iff  $\exists x \in \text{Instances}(A)$  such that  $T_s(x) = t_i$  (resp.  $T_e(x) = t_i$ ).
- $\langle \mathcal{T}, t_i \rangle \models \text{start}(a)$  (resp.  $\text{end}(a)$ ) iff  $T_s(a) = t_i$  (resp.  $T_e(a) = t_i$ ).
- $\langle \mathcal{T}, t_i \rangle \models P(A)$  iff  $A \in P$ .
- $\langle \mathcal{T}, t_i \rangle \models P(a)$  iff for some  $A \in \mathcal{A}$ ,  $a \in \text{Instances}(A)$  and  $A \in P$ .
- $\langle \mathcal{T}, t_i \rangle \models \neg\psi$  iff  $\langle \mathcal{T}, t_i \rangle \not\models \psi$ .
- $\langle \mathcal{T}, t_i \rangle \models \psi \wedge \chi$  iff  $\langle \mathcal{T}, t_i \rangle \models \psi$  and  $\langle \mathcal{T}, t_i \rangle \models \chi$ .
- $\langle \mathcal{T}, t_i \rangle \models \phi \mathcal{U}_I \psi$  iff  $\exists t_j$  such that  $\langle \mathcal{T}, t_j \rangle \models \psi$ ,  $t_j - t_i \in I$ , and  $\forall t_k$  such that  $t_i \leq t_k \leq t_j$  (or  $t_j \leq t_k \leq t_i$ ),  $\langle \mathcal{T}, t_k \rangle \models \phi$ .
- $\langle \mathcal{T}, t_i \rangle \models \forall x.\phi$  iff for every activity instance  $a \in \mathcal{I}$ ,  $\langle \mathcal{T}, t_i \rangle \models \phi_{x=a}$ , where  $\phi_{x=a}$  is the formula  $\phi$  with all occurrences of  $x$  replaced with  $a$ .

Semantically, a schedule as it is defined here is equivalent (although expressed in a different manner) to the signal function used to evaluate formulae in the continuous (as opposed to pointwise) semantics of MTL (Ouaknine and Worrell 2008).

We can also define other operators from the basic ones:

- $\exists, \vee, \supset$  (logical implication), and  $\equiv$  (logical equivalency) are defined as usual.
- The metric-valued “eventually” operator  $\diamond_I \phi$  is defined as  $\top \mathcal{U}_I \phi$ , and the metric valued “always” operator  $\square_I \phi$  is its dual  $\neg \diamond_I \neg \phi$ .
- Let  $\phi \rightarrow_I \psi$  be equivalent to  $\diamond_{(-\infty, \infty)}(\phi \wedge \diamond_I \psi)$ . This is semantically analogous to a simple temporal constraint:  $T \models \phi \rightarrow_I \psi$  iff  $\exists t_j, t_k$  such that  $\langle T, t_j \rangle \models \phi$ ,  $\langle T, t_k \rangle \models \psi$ , and  $t_k - t_j \in I$ . In other words,  $\psi$  is true at some time within interval  $I$  after  $\phi$  is true.
- Precedence constraint predicates *Before*( $\phi$ ), *After*( $\phi$ ), and *Between*( $\phi, \psi$ ) can be constructed as  $\diamond_{(0, \infty)} \phi$ ,  $\diamond_{(-\infty, 0)} \phi$ , and *After*( $\phi$ )  $\wedge$  *Before*( $\psi$ ), respectively. We also define *Currently*( $x$ ) as *Between*(*start*( $x$ ), *end*( $x$ )); this predicate signifies that an instance  $x$  is currently happening.
- For convenience, we will use  $\diamond \phi$  and  $\square \phi$  without the subscript to represent the LTL-style unconstrained definitions,  $\diamond_{[0, \infty)} \phi$  and  $\square_{[0, \infty)} \phi$ .
- Past-LTL operators are also easy to define. For example, the past version of  $\diamond$  (the “once” operator) can be defined as  $\diamond_{(-\infty, 0]}$ .
- For every activity  $A$ , let property  $P_A = \{A\}$ , so only instances of  $A$  share this property. Let predicate *InstanceOf*( $x, A$ )  $\equiv P_A(x)$ , so that *InstanceOf*( $x, A$ ) is true iff  $x$  is an instance of  $A$ .

Note that some of the above operators are already made redundant by others; i.e.  $\diamond^I \phi \equiv \text{true} \mathcal{U}^I \phi$ ,  $\phi \vee \psi \equiv \neg \phi \wedge \neg \psi$ ,  $\square^I \phi \equiv \neg \diamond^I \neg \phi$ , and  $\exists x \in P.\phi \equiv \neg \forall x \in P.\neg \phi$ . However, we will be considering fragments of this language that exclude some of these operators, so the semantics of all of them are given independently. A schedule  $\mathcal{T}$  *models* or *satisfies* a formula  $\phi$ , written as  $\mathcal{T} \models \phi$ , iff  $\langle \mathcal{T}, 0 \rangle \models \phi$  (0 is the initial time point).

## 4 Restrictions to Scheduling MTL

We now consider the complexity of this language, which we will measure by the difficulty of the following problem.

**Definition 4.1.** *The Scheduling MTL satisfiability problem, or SMTL-SAT, is the task of taking a Scheduling MTL formula  $\phi$  and finding a schedule  $\mathcal{T}$  such that  $\mathcal{T} \models \phi$ .*

Since Scheduling MTL is an extension of the standard propositional MTL, the fact that the latter is in EXPSPACE for integer values (Ouaknine and Worrell 2008) and undecidable for real values (Alur, Feder, and Henzinger 1996) implies SMTL-SAT is at least as hard. We therefore consider restrictions on the language to make satisfying Scheduling MTL formulae more tractable. The first restriction we impose is to bound the number of times an activity can occur.

**Definition 4.2.** *Bounded-instance MTL, or BI-MTL, is Scheduling MTL with the additional restriction that for every  $A \in \mathcal{A}$ , the problem specification includes a bound  $\sim k_A$ , where  $\sim$  is either  $=$  or  $\leq$ , and  $k_A \in \mathbb{N}$ . BI-MTL-SAT is defined analogously to SMTL-SAT. A satisfying schedule for a BI-MTL formula must additionally satisfy the bound  $|\text{Instances}(A)| \sim k_A$  for all  $A \in \mathcal{A}$ .*

Although we lose the ability to specify infinite schedules, bounded-instance MTL is a useful language that is capable of expressing many types of constraints. To our knowledge, restricting MTL in this way has never been studied before, perhaps due to the importance of infinite traces in most applications of MTL. However, for scheduling, it is natural to consider only finite numbers of events. By way of illustration, an  $=k$ -bounded activity can be used to represent the shifts of a nurse stipulated to work  $k$  shifts within the scheduling period. This model can be augmented with a  $\leq j$ -bounded activity representing  $j$  overtime shifts that this nurse could, but not necessarily, work. A coverage constraint requiring at least one nurse to be on shift at any given time can be represented as  $\square \exists x. \text{Currently}(x)$ , and in an integer domain where each time step is one day, a simple shift pattern for a nurse  $A$  stating that every four days on shift is followed by two days off can be expressed as  $\square(\square_{[0,3]} \text{start}(A) \supset \square_{[4,5]} \neg \text{start}(A))$ . This formula is easily adaptable to other  $x$ -on/ $y$ -off patterns, unique patterns for each nurse, pattern avoidance (“never schedule isolated days off”), and patterns that change over the course of the scheduling horizon. It is important to note that while these are typical scheduling patterns in nurse rostering, there is no scheduling language that can collectively encode them, a deficit that Burke openly laments (Burke et al. 2004).

The complexity of BI-STL-SAT is PSPACE-complete. To show that it is PSPACE-hard, one can reduce True Quantified Boolean Formula, the canonical PSPACE-complete problem, to BI-STL satisfiability by keeping the boolean operators and quantifiers as-is, and mapping each proposition to an activity. A proposition is assigned to be true if and only if the corresponding activity is assigned a start time of 0. Each quantified variable corresponds to an activity that repeats exactly twice, once at time zero and once after, mimicking the quantified variable’s domain of true and false. For a full proof, see Section 9.2 of the appendix. We demonstrate that it is still in PSPACE by showing that verifying a

schedule satisfies a formula can be computed in polynomial space, and giving a finite set of possible schedules to search through that are guaranteed to include a satisfying schedule if it exists. The full proof can be found in Section 9.6.

Having shown that BI-MTL-SAT is PSPACE-hard and also in PSPACE, we reach the following result:

**Theorem 4.1.** *BI-MTL satisfiability is PSPACE-complete.*

The PSPACE-completeness of satisfying bounded-instance MTL formulae may prove to be too complex for some scheduling problems. Analogously to the difference between TQBF and propositional SAT, removing quantifiers from our language reduces the complexity from PSPACE to NP. We define another fragment of our language as follows:

**Definition 4.3.** *Bounded-instance quantifier-free MTL, or BI-QF-MTL, is the set of BI-MTL formulae in which no subformula has the form  $\exists x.\phi$  or  $\forall x.\phi$ .*

We can prove that BI-QF-MTL satisfiability is NP-hard by reducing standard SAT to it in a simplified version of the reduction mentioned for TQBF above. Additionally, we describe a polynomial time verification algorithm for BI-QF-MTL here. Given a BI-QF-MTL formula  $\phi$  and a schedule  $\mathcal{T}$ , this algorithm assigns a set of *true times* to each subformula  $\psi$  such that for every time  $t$  in this set,  $\langle \mathcal{T}, t \rangle \models \psi$ . The true times of each atomic predicate  $\text{start}(a)$  and  $\text{end}(a)$  are the times the schedule assigns to each of its instances; then each additional operator combines or modifies these sets. For example, if  $\mathcal{T}$  assigns times 5 and 10 to a  $\leq$ -bounded activity  $A$ ,  $\text{start}(A)$  has true times  $\{5, 10\}$  and  $\diamond_{[0,3]}\text{start}(A)$  has true times  $[2, 5] \cup [7, 10]$ . Negation and conjunction correspond to set complementation and intersection, respectively. The algorithm returns true iff the true times of  $\phi$  include 0. This process is described in algorithm **BIQFSMTL-Verify**. The correctness of this algorithm is trivial after proving the following statement:

*Claim:* For any time  $t$ ,  $\langle \mathcal{T}, t \rangle \models \phi$  iff  $t \in \text{trueTimes}(\mathcal{T}, \phi)$ .

*Proof:* By structural induction over bounded-instance quantifier-free STL formulae; full proof in Section 9.8.

Now that we have shown that BI-QF-MTL-SAT is NP-hard and also in NP, we conclude the following result:

**Theorem 4.2.** *BI-QF-MTL satisfiability is NP-complete.*

We discuss applications of BI-QF-MTL in Section 6.

## 5 Solving BI-MTL-SAT

We now present a brief overview of an algorithm to find a schedule for a given BI-MTL formula. It proceeds as follows: first, the formula is transformed into an equivalent Scheduling MTL formula in which each activity occurs exactly once. Then, this formula is converted into a first-order logic formula with constants from  $\mathbb{D}$  and the  $+$  and  $\leq$  symbols. Finally, quantifiers are eliminated, resulting in a propositional formula with linear inequalities as atoms, which can be solved by an SMT solver.

### 5.1 Reduction to Exactly-Once Activities

**Definition 5.1.** *An exactly-once STL formula is a BI-STL formula in which no subformula is of the form  $\exists x.\phi$  or  $\forall x.\phi$ ,*

---

```

1 Algorithm: BIQFSMTL-Verify( $\mathcal{T}, \phi$ )
   Input: A schedule  $\mathcal{T} = (\text{Instances}, T)$ , and a bounded,
           quantifier-free STL formula  $\phi$ .
   Output: Whether or not  $\mathcal{T} \models \phi$ .
2 begin
3   | return true iff  $0 \in \text{trueTimes}(\mathcal{T}, \phi)$ ;
4 end
5 trueTimes( $\mathcal{T}, \phi$ ) :
6 begin
7   | if  $\phi = \text{start}(A)$  or  $\text{end}(A)$  :
8     | return  $\bigcup_{a \in \text{Instances}(A)} T_s(a)$  or  $T_e(a)$ , respectively
9   | else if  $\phi = P(A)$  :
10    | return  $\mathbb{D}$  if  $A \in P$ ,  $\emptyset$  if not
11  | else if  $\phi = \psi \wedge \chi$  :
12    | return  $\text{trueTimes}(\mathcal{T}, \psi) \cap \text{trueTimes}(\mathcal{T}, \chi)$ 
13  | else if  $\phi = \neg\psi$  :
14    | return  $\text{trueTimes}(\mathcal{T}, \psi)^C$ 
15  | else if  $\phi = \psi \mathcal{U}_I \chi$  :
16    | return  $\{t - i \mid i \in I, t \in \text{trueTimes}(\mathcal{T}, \chi), \text{ and } \forall t_k \text{ such that } t - i \leq t_k \leq t \text{ (or } t \leq t_k \leq t - i), t_k \in \text{trueTimes}(\mathcal{T}, \psi)\}$ 
17  | end
18 end

```

---

Note:  $^C$  in line 14 is the set complement operation.

---

and every activity used has bound  $=1$ .

We first transform  $\phi$  over bounded instance activities to an equivalent exactly-once formula  $\phi^{=1}$ .

- For each activity  $A$  with exact bound  $=k$ , replace it with  $k$  activities  $B_1, \dots, B_k$ , all with bound  $=1$ . In  $\phi$ , replace all occurrences of  $\text{start}(A)$  with  $\bigvee_{1 \leq i \leq k} \text{start}(B_i)$  and similarly for  $\text{end}(A)$ . For every property  $P$ , if  $A \in P$ , let  $P' = P \cup \{B_1, \dots, B_k\}$ , and replace all occurrences of  $P$  in  $\phi$  with  $P'$ .
- For each activity  $A$  with upper bound  $\leq k$ , replace it with  $k$  activities  $C_1, \dots, C_k$ , all with bound  $\leq 1$ . In  $\phi$ , replace all occurrences of  $\text{start}(A)$  with  $\bigvee_{1 \leq i \leq k} \text{start}(C_i)$  and similarly for  $\text{end}(A)$ . For every property  $P$ , if  $A \in P$ , let  $P' = P \cup \{C_1, \dots, C_k\}$ , and replace all occurrences of  $P$  in  $\phi$  with  $P'$ .
- For each subformula of the form  $\forall x.\psi$  in  $\phi$ , replace it with  $\bigwedge_{Y \in \mathcal{A}} (\diamond_{(-\infty, \infty)} \text{start}(Y) \supset \psi_{x=Y})$ , where  $\psi_{x=Y}$  is  $\psi$  with all occurrences of  $x$  replaced by  $Y$ . Note that at this point,  $Y$  is either a  $=1$ -bounded activity  $B_i$  or a  $\leq 1$ -bounded activity  $C_i$ .

Now,  $\phi$  has no quantifiers and consists of only  $=1$ -bounded activities and  $\leq 1$ -bounded activities. Call this formula  $\phi^{\leq 1}$ . We then replace  $\leq 1$ -bounded activities as follows:

- For each activity  $C$  with bound  $\leq 1$ , replace it with two activities  $\alpha_C$  and  $\beta_C$  with bound  $=1$ .
- Replace every instance of  $\text{start}(C)$  with  $\text{start}(\alpha_C) \wedge \text{start}(\beta_C)$  (similarly for  $\text{end}(C)$  and  $P(C)$ ).

- Append by conjunction the formulae  $\diamond(\text{start}(\alpha_C) \wedge \text{start}(\beta_C)) \equiv \diamond(\text{end}(\alpha_C) \wedge \text{end}(\beta_C))$  (for every  $\leq 1$  bounded  $C$ ).

This is our  $\phi^{\leq 1}$ .

Let  $\mathcal{T} = (\text{Instances}, T)$  be a schedule over the original activities in  $\phi$ , let  $\mathcal{T}' = (\text{Instances}', T')$  be a schedule over the  $\leq 1$ -bounded activities in  $\phi^{\leq 1}$ , and let  $\mathcal{T}'' = (\text{Instances}'', T'')$  be a schedule over the  $=1$ -bounded activities in  $\phi^{\leq 1}$ . We will define  $\mathcal{T}'$  in terms of  $\mathcal{T}''$  and  $\mathcal{T}$  in terms of  $\mathcal{T}'$  as follows. Given  $\mathcal{T}''$ , we define a  $\leq 1$ -bounded activity  $C$  to occur in  $\mathcal{T}'$  (i.e.  $|\text{Instances}'(C)| = 1$ ) iff  $\mathcal{T}'' \models \diamond(\text{start}(\alpha_C) \wedge \text{start}(\beta_C)) \wedge \diamond(\text{end}(\alpha_C) \wedge \text{end}(\beta_C))$ , and define  $T'(C) = (T''_s(\alpha_C), T''_e(\alpha_C))$ . For  $=1$ -bounded activities,  $\text{Instances}'$  and  $T'$  give the same value as  $\text{Instances}''$  and  $T''$ . In other words, given a time assignment for the  $=1$ -bounded activities, a  $\leq 1$ -bounded activity occurs in  $\mathcal{T}'$  iff the two corresponding  $=1$ -bounded activities occur simultaneously in  $\mathcal{T}''$ .

Then, for a  $\sim k$ -bounded activity  $A$  from the original formula, let  $\text{Instances}(A) = \bigcup_{1 \leq i \leq k} \text{Instances}'(B_i)$  (or  $C_i$ , depending on  $\sim$ ), and the time of each instance be the time assigned by  $T'$  to the corresponding  $B_i$  (or  $C_i$ ) instance.

*Claim:* Let  $\mathcal{T}''$  be a schedule over the activities of an exactly-once STL formula  $\phi^{\leq 1}$  resulting from reducing a bounded STL formula  $\phi$  by the method above, and let  $\mathcal{T}$  be the schedule over activities of  $\phi$  derived from  $\mathcal{T}''$  as just defined. Then  $\mathcal{T}'' \models \phi^{\leq 1}$  iff  $\mathcal{T} \models \phi$ .

*Proof (sketch):* Quantified instances in  $\phi$  are grounded over all possible instances and replaced with a conditional formula that checks if that particular instance actually occurs (it may not, if it is the instance of a  $\leq 1$ -bounded activity). A  $\leq 1$ -bounded activity  $C$  in  $\mathcal{T}'$  has either 0 or 1 instances, which corresponds to whether or not  $=1$ -bounded activities  $\alpha_C$  and  $\beta_C$  occur simultaneously in  $\mathcal{T}''$ . The union of the instances of  $k \leq 1$ -bounded activities yields a set of instances whose size is  $\leq k$ , which simulates a  $\leq k$ -bounded activity in  $\mathcal{T}$ . The rest of the proof follows from the semantics - see Section 9.3 in the appendix for a full proof.

## 5.2 Conversion to First-order Logic

The remainder of the algorithm proceeds in a manner similar to the one used by (Cimatti, Micheli, and Roveri 2012). In this step, we take the STL formula  $\phi^{\leq 1}$  and convert it to first-order logic via the algorithm **SMTL-to-FOL**. Note that since each activity has exactly one instance, we will equate activities with their instances for the remainder of this section.

As an example, recall that  $\text{end}(a) \rightarrow_{[2,3]} \text{start}(b)$  is shorthand for the formula  $\diamond_{(-\infty, \infty)}(\text{end}(a) \wedge \diamond_{[2,3]} \text{start}(b))$ . This formula has an equivalent first-order formula  $\exists i((i = \text{end}(a)) \wedge \exists j((2 \leq j) \wedge (j \leq 3) \wedge (i + j = \text{start}(b))))$ .

*Claim:* A schedule  $\mathcal{T} = (\text{Instances}, T)$  satisfies an exactly-once SMTL formula  $\phi$  iff variable assignment  $T$ , along with the standard interpretation of nonlogical symbols, satisfies  $\Phi_{FOL}$ .

*Proof:* By structural induction over well-formed exactly-once SMTL formulae. See Section 9.4 in the appendix.

---

### 1 Algorithm: SMTL-to-FOL( $\phi$ )

**Input:** An exactly-once SMTL formula over activities  $\mathcal{A} = \{a_1 \dots a_n\}$ .

**Output:** A first-order formula  $\Phi_{FOL}$  over the language  $\{+, \leq, \mathbb{D}\}$  with free variables in  $\{\text{start}(a), \text{end}(a) \mid a \in \mathcal{A}\}$ .  $\mathbb{D}$  is the set of constant symbols corresponding to the values in  $\mathbb{D}$ .

```

2 begin
3   | return recursiveFOL( $\phi, 0$ );
4 end
5 recursiveFOL( $\phi, x$ ) :
6 begin
7   | if  $\phi = \text{start}(a)$  :
8     | return  $\text{start}(a) = x$ 
9   | else if  $\phi = \text{end}(a)$  :
10    | return  $\text{end}(a) = x$ 
11  | else if  $\phi = P'(a)$  :
12    | return  $\top$  if  $a \in P'$ ,  $\perp$  if not
13  | else if  $\phi = \psi \wedge \chi$  :
14    | return
15    |   recursiveFOL( $\psi, x$ )  $\wedge$  recursiveFOL( $\chi, x$ )
16  | else if  $\phi = \neg\psi$  :
17    | return  $\neg$ recursiveFOL( $\psi, x$ )
18  | else if  $\phi = \psi \mathcal{U}_{[l,u]} \chi$  :
19    | return
20    |    $\exists i(l \leq i \wedge i \leq u \wedge$  recursiveFOL( $\chi, x + i$ )
21    |      $\wedge (i \geq 0 \supset \forall k((0 \leq k \wedge k \leq i) \supset$ 
22    |       recursiveFOL( $\psi, x + k)))$ )
23    |      $\wedge (i < 0 \supset \forall k((i \leq k \wedge k \leq 0) \supset$ 
24    |       recursiveFOL( $\psi, x + k)))$ )
25    |   recursiveFOL( $\psi, x + k$ )
26  | end
27 end

```

---

Note: If the interval  $I$  in  $\mathcal{U}_I$  is open, strict inequalities should be used. The equality  $x = a$  will actually be realized as  $a \leq x \wedge x \leq a$ .

---

The relationship between MTL and first-order logic is well-studied in Hirshfeld and Rabinovich [2004], and this conversion is similar to theirs, except the  $=1$  bound on all activities allows us to replace MTL propositions with variables rather than monadic predicates.

Note that while there are no quantifiers in  $\phi^{\leq 1}$ , there are first-order quantifiers in  $\Phi_{FOL}$ . At this point, the formula can be given to any SMT solver that supports such quantifiers, such as Z3 (De Moura and Bjørner 2008). However, if a standard SMT solver is preferred, quantifier elimination can be performed to propositionalize  $\Phi_{FOL}$ . One method of doing this is to convert all  $\forall$  quantifiers into  $\neg\exists\neg$ , then starting with the innermost quantified subformula, convert it to DNF, and for each clause perform Fourier-Motzkin elimination on the quantified variable (Schrijver 1986). This procedure is repeated until no more quantifiers appear in the formula. Section 9.5 expands on this topic.

To continue the previous example, performing quantifier elimination on  $\exists i((i = \text{end}(a)) \wedge \exists j((2 \leq j) \wedge (j \leq 3) \wedge$

$(i + j = \text{start}(b)))$  returns  
 $(2 \leq 3) \wedge (\text{start}(b) - 3 \leq \text{start}(b) - 2) \wedge (\text{end}(a) \leq \text{start}(b) - 2) \wedge (\text{start}(b) - 3 \leq \text{end}(a)) \wedge (\text{end}(a) \leq \text{end}(a))$ .

Discarding the trivially true clauses simplifies the formula to  $(\text{end}(a) \leq \text{start}(b) - 2) \wedge (\text{start}(b) - 3 \leq \text{end}(a))$ , i.e.  $2 \leq \text{start}(b) - \text{end}(a) \leq 3$ , which agrees with the semantics of the original scheduling MTL formula.

While it is clear that the literals of the final quantifier-free formula are linear equalities, it is possible to strengthen the result and show it is a difference logic formula: all the atomic propositions are inequalities of the form  $x - y \leq k$  or  $x - y < k$ , where  $x$  and  $y$  are variables and  $k$  is a constant. Now that  $\phi$  is in the form of a difference logic formula, a schedule can be found by any standard SMT solver (with a difference logic or linear arithmetic theory solver). We note that the entire algorithm uses space exponential in the input although we have proven the problem is in PSPACE. However, this approach elucidates several relationships with existing problems and also exploits advances in SMT. In particular, recent developments in SMT solvers allow SMT problems to be optimized with respect to a linear objective function (Sebastiani and Tomasi 2012; Li et al. 2014). This allows us to optimize a schedule in various ways (schedule some activities as early as possible and another as late as possible, minimize makespan, etc.).

## 6 MTL Applied to Existing Problems

A natural question to ask now is what kind of constraints can be expressed using only BI-QF-MTL. The most straightforward uses of this fragment are for constraining events that happen exactly or at most once. This captures the needs of many scheduling problems where there is a fixed set of activities that need to be scheduled, such as job shop scheduling or temporal network problems. We explore both these applications in this section.

### 6.1 Job Shop Scheduling

The job shop scheduling problem (JSP) is one of the most well-studied scheduling paradigms (Fox 1983). We demonstrate here that BI-QF-MTL is capable of modeling JSPs in a straightforward fashion. In a standard JSP, a set of jobs must be completed on a set of machines, which can process at most one job at a time. A job may require the use of multiple machines in a fixed order: job  $J$  is associated with a sequence of machines  $S(J, 1) \dots S(J, n)$  and processing times on these machines,  $P(J, 1) \dots P(J, n)$ . The solution to the problem is an assignment of times to jobs that respects the processing times of each job, the sequence ordering of the machines for each job, and the capacity of each machine.

Our formulation consists of a set of BI-QF-MTL formulae  $\Phi$  over  $=1$ -bounded activities for each machine in a job's sequence:  $\mathcal{A} = \{a_{J,i} \mid \text{job } J, 1 \leq i \leq n\}$ . For each machine  $M$ , define property  $U_M$  to be shared by all the elements of all job sequences that use machine  $M$ , i.e.  $U_M = \{a_{J,i} \mid S(J,i) = M\}$ . For each processing time  $P(J,i)$ , add constraint  $\text{start}(a_{J,i}) \rightarrow_{[P(J,i), P(J,i)]} \text{end}(a_{J,i})$ . In order to preserve the order of each job sequence, for each job  $J$  with job sequence of length  $n$ , for  $1 \leq k < n$ , add constraint

$\text{end}(a_{J,k}) \rightarrow_{[0, \infty)} \text{start}(a_{J,k+1})$ . Finally, to allow only one job to be processed on a machine at a time, for each machine  $M$ , add the formula  $\bigwedge_{a \in U_M, b \in U_M, a \neq b} \Box \neg (\text{Currently}(a) \wedge$

$\text{Currently}(b))$  to  $\Phi$  (recall that  $\text{Currently}(a)$  is defined as  $\text{Between}(\text{start}(a), \text{end}(a))$ , i.e. it is true when activity  $a$  is currently happening). See Section 9.9 for a proof of the correctness of this reduction.

To pose the decision problem (i.e., does there exist a schedule within a makespan  $D$ ), two activities,  $S$  and  $T$ , which require no resources, are created.  $S$  is constrained to execute before the first activity in each job and  $T$  is constrained to execute after the last. Then, the following constraint is posed:  $\text{start}(S) \rightarrow_{[0, D]} \text{start}(T)$ . Optimization problems cannot be directly expressed in MTL but must be handled separately, as in the SMT component of the algorithm in Section 5.

This formulation can also be extended to model variations of JSP: idle time between jobs, sequence dependent setups, precedence constraints between jobs, and variable processing times can all be expressed as Scheduling MTL formulae. While specialized methods already exist to solve these problems, Scheduling MTL provides a unifying language for expressing these constraints and opens up possibilities for more complex and expressive ones in the future.

### 6.2 Temporal Networks

While MTL is a timed temporal logic, its semantics allow it to model the well-studied class of temporal network problems simply and directly, using only exactly-once formulae. We begin with some definitions.

**Definition 6.1.** *Given a set of instantaneous events, a simple temporal constraint has the form  $[l, u]_{xy}$ , where  $x, y$  are times of events and  $l, u$  are constant bounds; it represents the inequality  $l \leq y - x \leq u$ . A simple temporal network (STN) is a set of simple temporal constraints, and the simple temporal problem (STP) is the task of assigning times to events that satisfy all the constraints in a STN (Dechter, Meiri, and Pearl 1991).*

**Definition 6.2.** *A disjunctive temporal constraint is a disjunction of simple temporal constraints. A disjunctive temporal network (DTN) is a set of disjunctive temporal constraints, and the disjunctive temporal problem (DTP) is the task of assigning times to events that satisfy all the constraints in a DTN (Stergiou and Koubarakis 1998).*

The “network” terminology derives from the fact that temporal networks are often represented as graphs with events as nodes and constraints as edges. STP is known to be NL-complete and DTP is known to be NP-complete (Planken 2013; Stergiou and Koubarakis 1998).

It is straightforward to model a given DTN with an exactly-once MTL formula  $\phi_{DTN}$ . As mentioned in Section 3, the derived operator  $\phi \rightarrow_I \psi$  exactly corresponds to a simple temporal constraint between  $\phi$  and  $\psi$  when they are events in a temporal network. Therefore, a STN can be modeled as a conjunction of  $\rightarrow_I$  formulae, with one  $=1$ -bounded activity for each event, and a DTN can be expressed as a

conjunction of disjunctions of these formulae. See Section 9.10 for a full proof.

DTNs were originally created with the intent of extending the expressivity of STNs; many people in turn have augmented DTNs with additional machinery to continue this extension. We consider two of these extensions and demonstrate that they can also be modeled as BI-QF-MTL formulae.

A generalized temporal network (GTN) is an extension of DTNs that allows constraints to be disjunctions of *conjunctions* of simple temporal constraints, rather than disjunctions of constraints (Staab 1998). The conversion from DTNs to MTL formulae in the previous section can be easily modified to show that GTNs are also expressible as exactly-once MTL formulae, by adding one additional layer of conjunctions.

A conditional temporal problem (CTP) extends DTPs or STPs by adding variables whose truth value must be observed at an event, and temporal constraints that only activate if a particular set of variables has been observed to be true (Tsamardinos, Vidal, and Pollack 2003). This problem has been a subject of recent interest where the observation variables are treated as fully controllable decision variables (Yu and Williams 2013). It is possible to model such a controllable CTP using BI-QF-MTL, with the use of  $\leq 1$  bounded activities (activities that either happen or not) to represent the truth value of observation variables, and temporal constraints that are conditioned on which of these activities occur.

The benefits of Scheduling MTL are clear with respect to these extensions: while entirely new formulations were developed to handle specific forms of disjunction and condition for temporal networks, even the BI-QF-MTL fragment can specify disjunctions and conditions of arbitrary subformulae.

We now consider the relationship between MTL and STPs. To do so, we first define the following restriction.

**Definition 6.3.** For a set of MTL operators  $\mathcal{O}$ ,  $\mathcal{O}$ -formulae are the set of MTL formulae using only operators in  $\mathcal{O}$ .  $\mathcal{O}$ -SAT is the satisfiability problem for  $\mathcal{O}$ -formulae.

We first observe that  $\phi_{DTN}$  is an exactly-once  $\{\wedge, \vee, \diamond_I\}$ -formula by construction (since  $\rightarrow_I$  is composed of  $\wedge$  and  $\diamond_I$ ). Furthermore, for STNs, the same transformation results in an exactly-once  $\{\wedge, \diamond_I\}$  formula, since  $\vee$  is only used in  $\phi_{DTN}$  for disjoining simple temporal constraints. While this demonstrates that exactly-once  $\{\wedge, \diamond_I\}$  formulae are at least as expressive as STNs, it turns out the converse is true as well.

To accomplish this for a given exactly-once  $\{\wedge, \diamond_I\}$  formula  $\phi$ , an event is created for each start and end predicate, as well as one for each subformula of  $\phi$ , which intuitively represents the time at which that subformula is made true. We assume no atomic predicates are property predicates, because without quantifiers we can immediately evaluate the truth value of  $P(A)$  for any property  $P$  and activity  $A$  and replace it with  $\top$  or  $\perp$  accordingly. This process is described in algorithm **SMTL-to-STN** and demonstrated for a small example in Figure 1.

---

#### 1 Algorithm: SMTL-to-STN( $\phi$ )

**Input:** An exactly-once  $\{\wedge, \diamond_I\}$ -formula  $\phi$  over a set of activities  $\mathcal{A}$

**Output:** A set of STN constraints on  $\{start(A) \mid A \in \mathcal{A}\} \cup \{end(A) \mid A \in \mathcal{A}\} \cup \Psi \cup START$ , where  $\Psi$  is the set of all subformulae of  $\phi$  and  $START$  is an event signifying the initial time point of the schedule.

```

2 begin
3   return AddExplicitConstraints( $\phi$ ,  $[0, 0]_{START, \phi}$ );
4 end
5 AddExplicitConstraints( $\phi$ , constraints)
6 begin
7   if  $\phi = start(A)$  or  $end(A)$  :
8     return constraints;
9   else if  $\phi = \psi \wedge \chi$  :
10    return  $\{[0, 0]_{\phi, \psi}, [0, 0]_{\phi, \chi}\} \cup$ 
11         AddExplicitConstraints( $\psi$ , constraints)  $\cup$ 
12         AddExplicitConstraints( $\chi$ , constraints);
13  else if  $\phi = \diamond_{[l, u]} \psi$  :
14    return  $\{[l, u]_{\phi, \psi}\} \cup$ 
15         AddExplicitConstraints( $\psi$ , constraints);
16 end
17 end

```

---

Figure 1 shows the STN resulting from converting  $\diamond(end(A) \wedge \diamond_{[2,3]}(start(B) \wedge start(C)))$ , which states that  $B$  and  $C$  simultaneously begin 2-3 time units after  $A$  ends. Observe that any satisfying schedule must assign  $\phi_2, \phi_3$ , and  $end(A)$  to the same time due to the  $[0, 0]$  constraints between them; the same holds for  $\phi_4, start(B)$ , and  $start(C)$ . The constraint  $[2, 3]_{\phi_2, \phi_3}$  then forces  $end(A)$  to occur 2-3 time units before  $start(B)$  and  $start(C)$ , which agrees with the semantics of the original formula. The  $[0, \infty)$  constraint between  $\phi_1$  and  $\phi_2$  allows this gap to occur at any positive time.

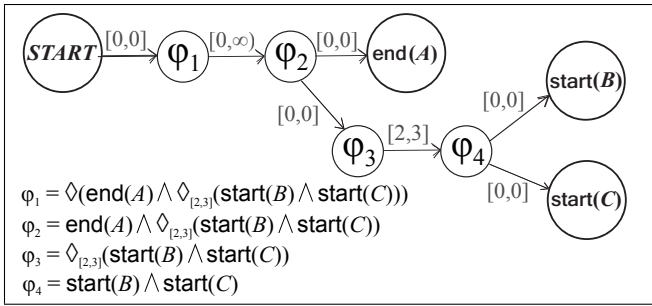
Note that duplicates of auxiliary nodes may occur if the same subformula appears multiple times in the same formula. For example, calling the algorithm on  $\diamond(A \wedge Before(C)) \wedge \diamond(B \wedge Before(C))$  will return constraints on two *different* nodes both labeled  $Before(C)$  because  $Before(C)$  occurs twice in the formula. However, all nodes corresponding to atomic propositions refer to the same event.

**SMTL-to-STN** is easily proven correct after proving the following claim:

*Claim:* Let  $STN_\phi$  be the set of constraints returned by **AddExplicitConstraints**( $\phi, \emptyset$ ), and let  $T$  be a variable assignment over the events in  $STN_\phi$ . Define  $\mathcal{T}_T = (Instances, T)$ , where  $Instances$  is the identity function (since all activities are =1-bounded). Then,  $T$  satisfies the constraints returned by **AddExplicitConstraints**( $\phi, \emptyset$ ) iff  $\langle \mathcal{T}, T(\phi) \rangle \models \phi$  (where  $T(\phi)$  is the time the auxiliary node  $\phi$  is scheduled).

*Proof:* By structural induction over  $\{\wedge, \diamond_I\}$ -formulae. A full proof can be found in Section 9.12.

Since STNs are expressible as exactly-once  $\{\wedge, \diamond_I\}$ -



**Figure 1:** The STN equivalent to  $\diamond(\text{end}(A) \wedge \diamond_{[2,3]}(\text{start}(B) \wedge \text{start}(C)))$ .

formulae and exactly-once  $\{\wedge, \diamond_I\}$ -formulae can be converted to STNs via the above algorithm (both in log-space), we have the following result:

**Theorem 6.1.** *Exactly-once  $\{\wedge, \diamond_I\}$ -formulae and STNs are equivalently expressive.*

We find it remarkable that this simple syntactic restriction, combined with the exactly-once bound, results in a temporal logic expressible as a temporal network. There is a known relationship between the timed-word model of MTL and timed automata (Alur, Feder, and Henzinger 1996), but the continuous semantics of these logics (such as the one used here) have not previously been known to be expressible with such a graph-based model. This may also shed some light on the connection between timed automata and temporal networks, a recently explored topic (Cimatti et al. 2014).

The mutual expressibility between STNs and this MTL fragment also allows us to state its complexity:

**Theorem 6.2.** *Exactly-once  $\{\wedge, \diamond_I\}$ -SAT is in P.*

*Proof.* A  $\{\wedge, \diamond_I\}$ -formulae can be converted to an equivalent STN in linear time via the algorithm described above, which can then be solved in polynomial time by any standard STP algorithm (Planken 2013).  $\square$

## 7 Related Work and Summary

Few attempts have been made to apply temporal logic to scheduling. Dorn [1993] is an early example, although its approach is very different—the temporal logic used is based on Allen’s interval algebra and resources are handled in an object-oriented manner.

A common application of temporal logic constraints is for augmenting an existing problem representation with a temporal logic formula and later compiling the formula into the language of the model. In scheduling, both LTL and MTL have been applied to the vehicle routing problem by compiling the state-sequence based formula into mixed integer linear programming (MILP) constraints, which were added to the existing MILP model (Karaman, Sanfelice, and Frazzoli 2008; Karaman and Frazzoli 2008). This approach has also been applied to discrete-time nonlinear systems (Wolff, Topcu, and Murray 2014). Another method of compilation is to convert an LTL formula into an equivalent finite-state automaton; this method is used in temporally extended goals in planning problems (Baier and McIlraith 2006). Similarly,

timed temporal logics can be converted into timed automata (Ostroff 1989).

While our approach also compiles MTL formulae into another language, it differs in using temporal logic as the native language to specify the entire problem. This difference allows Scheduling MTL to be used in formal logical reasoning to answer queries about the entire problem, rather than just the fragment represented in logic. Additionally, the previous examples all rely on a state sequence model for satisfying formulae; ours uses a continuous timeline model instead, which results in a slightly different semantics (Hirshfeld and Rabinovich 2004).

In general, restrictions of MTL are more common than extensions in the literature. Most of these have sought to increase tractability through restricting the types of intervals allowed in the  $\mathcal{U}_I$  operator, such as Bounded MTL, which restricts all intervals to be of finite length; satisfiability for this logic in  $\mathbb{R}$  is EXPSpace-complete (Bouyer et al. 2008). The strongest syntactic restriction on MTL intervals that is still more expressive than LTL results in a satisfiability problem that is PSPACE-complete (Ouaknine and Worrell 2008). In contrast, using an approach slightly different from ours, Bacchus and Kabanza [1998] augmented MTL with first-order quantifiers to express temporally extended goals in TLPlan planner.

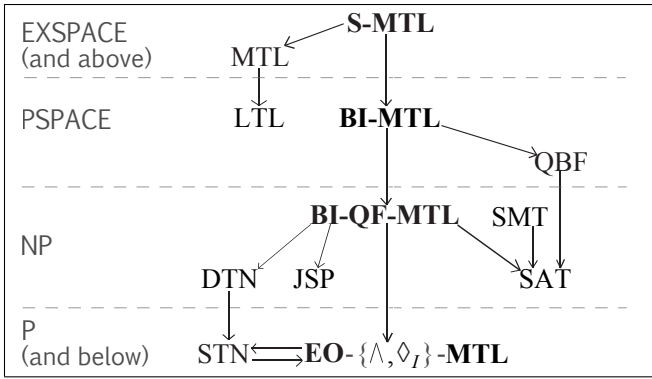
Combining capacity constraints with LTL was explored by Dixon, Fisher, and Konev [2007], by limiting the number of propositions that could be true at any given time. Metric-valued temporal constraints are not considered, however, and again the state-sequence model differs from ours. LTL restricted to finite traces has also been studied (De Giacomo and Vardi 2013; De Giacomo, De Masellis, and Montali 2014).

Other extensions to DTPs and STPs have also been previously studied. Some of these are also expressible in BI-QF-MTL. For example, Generalized Temporal Networks (Staab 1998) allow constraints composed of disjunctions of *conjunctions* of simple temporal constraints. Conditional Temporal Problems (CTPs) extend DTNs by only requiring constraints to be satisfied under certain conditions (Tsamardinos, Vidal, and Pollack 2003). The original formulation of CTPs defines these conditions as uncontrollable, but a controllable version (which is expressible in Scheduling MTL) has also been examined (Yu and Williams 2013). The benefits of Scheduling MTL are clear here: while entirely new formulations were developed to handle specific forms of disjunction and condition for temporal networks, even the BI-QF-MTL fragment can specify disjunctions and conditions of arbitrary subformulae.

## 8 Concluding Remarks

To summarize our complexity results, Figure 2 illustrates the complexity of each successive restriction to Scheduling MTL we have introduced and its expressiveness relative to other logics and scheduling problems. An arrow from A to B indicates that problem A is polytime reducible to problem B. For all the vertical arrows in the diagram, the reduction is trivial; methods of reduction for the rest of them have been addressed in this paper. Note that the algorithm described in





**Figure 2:** Complexity classes for satisfiability of Scheduling MTL fragments and related problems, in the spirit of [Planken, 2013]. Arrows represent polytime reductions. “S-MTL” is Scheduling MTL, and “EO” is exactly-once.

Section 5 converts a BI-MTL formula to a BI-QF-MTL formula, and finally into a SMT formula; however, neither of these conversions takes polynomial time.

As a proof of concept, we built a Z3-based implementation of our algorithm that confirms the straightforward nature of the conversion to SMT. Empirical results are not reported here since we did not do extensive benchmarking. The culture of the scheduling community is to build problem-specific solvers. Any such solver would demonstrate superior performance. The value and significance of an expressive language such as Scheduling MTL lies rather in the ability to help us specify, understand, compare, and solve a myriad of scheduling problems with varying complex constraints that are one-off and may not exist in the scheduling literature.

In conclusion, we have introduced Scheduling Metric Temporal Logic, an adaptation of MTL for scheduling problems, and demonstrated the expressive power of several natural restrictions of the language. In particular, the ability for BI-QF-MTL to model classic problems such as job shop scheduling as well as newer developments such as conditional temporal problems suggests that it is a highly versatile language. We believe the utility of MTL does not only lie in its capability to specify existing problems, but also to augment and generalize such problems. Additionally, its logical basis allows us to reason over constraints to answer queries about problems.

We have described an algorithm to find a schedule satisfying a BI-MTL (or BI-QF-MTL) formula, and characterized the complexity of this problem under various restrictions. Finally, we have shown that a simple syntactic restriction of Scheduling MTL is equivalent to simple temporal networks, providing, for the first time to our knowledge, a connection between temporal logic and STNs.

## Acknowledgements

This work was performed in partial fulfillment of Roy Luo’s Masters of Science (Computer Science) degree in the Department of Computer Science, University of Toronto, February, 2015. Professors Chris Beck and Sheila McIl-

raith were Roy’s co-supervisors. We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC).

## 9 Appendix

### 9.1 MTL $\leq$ Scheduling MTL

First, we begin with the syntax and semantics of MTL.

MTL formulae are constructed by the following grammar:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \mathcal{U}_I \phi$$

Where  $p$  is an atomic proposition from a set of propositions  $P$ , and  $\mathcal{U}_I$  is defined in the same manner as the SMTL version.

Under the *continuous semantics* of MTL, a *signal* is a function  $f : \mathbb{R}_+ \mapsto 2^P$ , where  $\mathbb{R}_+$  is the set of nonnegative real numbers.  $f$  maps every time point to the set of atomic propositions true at that time, and then the truth of a formula at a time point with respect to  $f$  is defined inductively following the standard semantics of the given operators.

Given a MTL formula  $\phi_{MTL}$  over propositions  $P$  and a signal  $f$ , we can construct a SMTL formula  $\phi_{SMTL}$  over activities  $\mathcal{A}$  and schedule  $\mathcal{T}$  as follows:

- For each  $p \in P$ , add activity  $A_p$  to  $\mathcal{A}$ , and an uncountably infinite set of possible instances of  $A_p$ ,  $a_{p,1}, \dots$  to the instance space *Instances*.
- Construct  $\phi_{SMTL}$  by replacing each  $p$  in  $\phi_{MTL}$  with  $\text{start}(A_p) \wedge \text{end}(A_p)$  (keep all operators the same).
- Let  $\text{Times}_p$  be the set of times proposition  $p$  is true in signal  $f$ , i.e.  $\text{Times}_p = \{t \mid p \in f(t)\}$ . We will have one instance of activity  $A_p$  for each point in this set, i.e.  $|\text{Instances}(A_p)| = |\text{Times}_p|$ .
- The timing function  $T$  of  $\mathcal{T}$  maps the start and end of each instance of  $A_p$  to a time in  $\text{Times}_p$ . More formally, for any proposition  $p$ , fix  $g_p$  to be some isomorphism from  $\text{Instances}(A_p)$  to  $\text{Times}_p$ , and for every  $a_p \in \text{Instances}(A_p)$ , define  $T(a_p) = (g_p(a_p), g_p(a_p))$ .

*Claim:*  $\forall t \in \mathbb{R}, \langle \mathcal{T}, t \rangle \models \phi_{SMTL}$  iff  $f(t) \models \phi_{MTL}$ .

*Proof:* By structural induction over MTL formulae.

- **Base Case:**  $\phi_{MTL} = p$ .  
 $f(t) \models p \Leftrightarrow p \in f(t) \Leftrightarrow \exists a_p \in \text{Instances}(A_p) \text{ s.t. } T(a_p) = (t, t) \Leftrightarrow \langle \mathcal{T}, t \rangle \models \text{start}(A_p) \wedge \text{end}(A_p) \Leftrightarrow \langle \mathcal{T}, t \rangle \models \phi_{SMTL}$ .
- **Inductive Case:**  $\phi_{MTL} = \neg\psi_{MTL}$ .  
 $f(t) \models \phi_{MTL} \Leftrightarrow f(t) \models \neg\psi_{MTL} \Leftrightarrow f(t) \not\models \psi_{MTL} \Leftrightarrow \langle \mathcal{T}, t \rangle \not\models \psi_{SMTL}$  (by inductive hypothesis)  $\Leftrightarrow \langle \mathcal{T}, t \rangle \models \neg\psi_{SMTL} \Leftrightarrow \langle \mathcal{T}, t \rangle \models \phi_{SMTL}$ .
- **Inductive Case:**  $\phi_{MTL} = \psi_{MTL} \wedge \chi_{MTL}$ .  
 $f(t) \models \phi_{MTL} \Leftrightarrow f(t) \models \psi_{MTL} \wedge \chi_{MTL} \Leftrightarrow f(t) \models \psi_{MTL} \text{ and } f(t) \models \chi_{MTL} \Leftrightarrow \langle \mathcal{T}, t \rangle \models \psi_{SMTL} \text{ and } \langle \mathcal{T}, t \rangle \models \chi_{SMTL}$  (by inductive hypothesis)  $\Leftrightarrow \langle \mathcal{T}, t \rangle \models \psi_{SMTL} \wedge \chi_{SMTL} \Leftrightarrow \langle \mathcal{T}, t \rangle \models \phi_{SMTL}$ .
- **Inductive Case:**  $\phi_{MTL} = \psi_{MTL} \mathcal{U}_I \chi_{MTL}$ .  
 $f(t) \models \phi_{MTL} \Leftrightarrow f(t) \models \psi_{MTL} \mathcal{U}_I \chi_{MTL} \Leftrightarrow \exists t_j \text{ s.t. } t_j - t_i \in I, f(t_j) \models \psi_{MTL}, \text{ and } \forall t_k \text{ s.t. } t \leq t_k \leq t_j,$

$f(t_k) \models \chi_{SMTL} \Leftrightarrow \exists t_j \text{ s.t. } t_j - t_i \in I, \langle \mathcal{T}, t_j \rangle \models \psi_{SMTL}$ , and  $\forall t_k \text{ s.t. } t \leq t_k \leq t_j, \langle \mathcal{T}, t_k \rangle \models \chi_{SMTL}$  (by inductive hypothesis)  $\Leftrightarrow \langle \mathcal{T}, t \rangle \models \psi_{SMTL} \mathcal{U} \chi_{SMTL} \Leftrightarrow \langle \mathcal{T}, t \rangle \models \phi_{SMTL}$ .

Finally, we present the full result:

*Claim:*  $\mathcal{T} \models \phi_{SMTL}$  iff  $f \models \phi_{MTL}$ .

*Proof:*  $\mathcal{T} \models \phi_{SMTL} \Leftrightarrow \langle \mathcal{T}, 0 \rangle \models \phi_{SMTL} \Leftrightarrow f(0) \models \phi_{MTL}$  (by the above result)  $\Leftrightarrow f \models \phi_{MTL}$ .

## 9.2 QBF $\leq$ BI-SMTL

We assume the reader is familiar with the syntax and semantics of QBF formulae. We will assume all variables are bound and the formula is in prenex normal form. Determining whether such a formula is true is PSPACE-complete; we will use this fact to show that determining whether a BI-SMTL formula with no free variables is true is PSPACE-hard. From this result, we can deduce that finding a schedule satisfying a BI-SMTL formula is also PSPACE-hard since doing so requires evaluating the truth of a BI-SMTL formula.

Given a prenex normal form QBF formula  $\phi_{QBF} = \exists x_1. \forall x_2. \exists x_3. \dots Q x_n. \psi$ , where  $Q$  is either  $\exists$  or  $\forall$  and  $\psi$  is a boolean formula over  $x_1 \dots x_n$ , let  $P$  be the set of all existentially quantified variables and let  $Q$  be the set of all universally quantified variables. Construct a BI-SMTL formula  $\phi_{BISMTL}$  as follows:

- For each variable  $p$  in the set of all QBF variables, add an  $=2$ -bounded activity  $A_p$  to  $\mathcal{A}$ .
- For every atomic proposition  $p$  in  $\phi_{QBF}$ , replace it with  $\text{start}(p)$ .
- Let  $\chi_{\exists} = \bigwedge_{p \in P} \text{InstanceOf}(p, A_p)$  and  $\chi_{\forall} = \bigwedge_{q \in Q} \text{InstanceOf}(q, A_q)$ . Replace  $\psi$  with  $\chi_{\exists} \wedge (\chi_{\forall} \supset \psi)$ .
- For every  $p$  in the set of all QBF variables, conjoin  $\phi_{BISMTL}$  with the formula  $\text{start}(A_p) \wedge \diamond_{[1,1]} \text{start}(A_p)$ .

The intuition of this construction is that a proposition is true in  $\phi_{QBF}$  iff the corresponding activity in  $\phi_{BISMTL}$  occurs at time 0. Each activity occurs twice, once at 0 and once at 1, so each quantified instance ranges over these two values, which corresponds to quantified variables in  $\phi_{QBF}$  ranging over the values true and false. The exact time of 1 is not important; we simply need to fix the second instance to occur at some time other than 0.

*Claim:* Let  $\mathcal{T} = (\text{Instances}, T)$  be defined such that for all  $A \in \mathcal{A}$ ,  $\text{Instances}(A) = \{a_0, a_1\}$ ,  $T(a_0) = (0, 0)$ , and  $T(a_1) = (1, 1)$ . Then  $\phi_{QBF}$  is true iff  $\mathcal{T} \models \phi_{BISMTL}$ .

*Proof:*

Assume  $\phi_{QBF}$  is true. Then there is a winning strategy  $S$  for Player E when  $\phi_{QBF}$  is treated as a formula game (**todo: cite Sipser text**); this strategy consists of truth assignments for every existentially quantified variable for every universally quantified variable preceding it such that the formula evaluates to true in all cases. Using this strategy, we can create a corresponding one  $S'$  for  $\phi_{BISMTL}$  in which the players take turns assigning times to activities. Whenever  $S$  assigns an existentially quantified variable  $p$  to be true,  $S'$

assigns the time of  $\text{start}(a_p)$  to be 0; whenever  $S$  assigns  $p$  to be false,  $S'$  assigns  $\text{start}(a_p)$  to be 1 (the end time is also assigned to the same value to match  $\mathcal{T}$  as defined above). Assignments to universally quantified variables follow the same pattern, so that after all quantifiers have had truth assignments, the propositional formula  $\psi_{BISMTL}$  in  $\phi_{BISMTL}$  corresponding to  $\psi$  in  $\phi_{QBF}$  is a boolean formula over predicates  $\text{start}(A_p)$  for  $p \in P \cup Q$ , accompanied by a time assignment to these start events. By the semantics of SMTL, for a schedule  $\mathcal{T}$ ,  $\mathcal{T} \models \text{start}(A_p)$  iff  $T_s(a_p) = 0$  (i.e.  $S'$  assigned a time of 0); therefore  $\text{start}(A_p) \equiv \top$  if  $S$  assigned  $p$  to true and  $\text{start}(A_p) \equiv \perp$  if  $S$  assigned  $p$  to false. Since the atomic predicates of  $\phi_{BISMTL}$  are always equivalent to the atomic propositions in  $\phi_{QBF}$ , the boolean operators are semantically identical in both domains. The quantifiers are also forced to behave identically by the additional constraints in  $\phi_{QBF}$  imposed by the last two bullet points in the above construction: each quantified variable can only range of instances of a single activity by the addition of the *InstanceOf* formula, and each activity has two instances, one corresponding to true (0) and one to false (1), so quantified instances range over exactly the values 0 and 1, which mirrors the behavior of the quantified variables in  $\phi_{QBF}$ . Therefore the truth value of  $\phi_{QBF}$  and  $\phi_{BISMTL}$  must be equal as well. The proof in the other direction (assuming  $\mathcal{T} \models \phi_{BISMTL}$ ) is analogous; the strategy for  $\phi_{QBF}$  is constructed from  $\phi_{BISMTL}$  by setting  $p$  to be true iff  $\text{start}(A_p)$  is assigned time 0.

## 9.3 $\phi \equiv \phi^{=1}$

We proceed in two parts; first we prove that  $\mathcal{T}'' \models \phi^{=1}$  iff  $\mathcal{T}' \models \phi^{\leq 1}$ , and then we prove that  $\mathcal{T}' \models \phi^{\leq 1}$  iff  $\mathcal{T} \models \phi$ .

*Claim:*  $\mathcal{T}'' \models \phi^{=1}$  iff  $\mathcal{T}' \models \phi^{\leq 1}$ .

*Proof:*

Suppose  $\mathcal{T}'' \models \phi^{=1}$ . For each  $\leq 1$ -bounded activity  $C$  in  $\phi^{\leq 1}$ ,  $C$  occurs ( $|\text{Instances}'(C)| = 1$ ) iff  $\mathcal{T}'' \models \diamond(\text{start}(\alpha_C) \wedge \text{start}(\beta_C)) \wedge \diamond(\text{end}(\alpha_C) \wedge \text{end}(\beta_C))$ . Say  $\mathcal{T}''$  does satisfy this formula. Then  $T''(\alpha_C) = T''(\beta_C) = T'(c)$ , where  $\text{Instances}'(C) = \{c\}$ . Therefore,  $\forall t \in \mathbb{D}, \langle \mathcal{T}'', t \rangle \models \text{start}(\alpha_C) \wedge \text{start}(\beta_C)$  iff  $\langle \mathcal{T}', t \rangle \models \text{start}(C)$  and similarly for  $\text{end}(\alpha_C) \wedge \text{end}(\beta_C)$  and  $\text{end}(C)$ . Since  $\alpha_C$  and  $\beta_C$  only appear in  $\phi^{=1}$  exactly where there was a start/end predicate for  $C$  in  $\phi^{\leq 1}$ , they are semantically equivalent, and the rest of the formula was unchanged,  $\mathcal{T}' \models \phi^{\leq 1}$ . This proof can be reversed to prove the converse.

Now we consider the case where  $\mathcal{T}'' \not\models \diamond(\text{start}(\alpha_C) \wedge \text{start}(\beta_C)) \wedge \diamond(\text{end}(\alpha_C) \wedge \text{end}(\beta_C))$ . Then  $\mathcal{T}'' \models \neg(\diamond(\text{start}(\alpha_C) \wedge \text{start}(\beta_C)) \wedge \diamond(\text{end}(\alpha_C) \wedge \text{end}(\beta_C)))$ , which is equivalent to  $\mathcal{T}'' \models (\neg \diamond(\text{start}(\alpha_C) \wedge \text{start}(\beta_C))) \vee (\neg \diamond(\text{end}(\alpha_C) \wedge \text{end}(\beta_C)))$ . Since we know that  $\mathcal{T}'' \models \diamond(\text{start}(\alpha_C) \wedge \text{start}(\beta_C)) \equiv \diamond(\text{end}(\alpha_C) \wedge \text{end}(\beta_C))$  (since this formula was conjoined to  $\phi^{=1}$  in our construction) and we know at least one of the two sides is false, both must be false, i.e.  $\mathcal{T}'' \models (\neg \diamond(\text{start}(\alpha_C) \wedge \text{start}(\beta_C)))$  and  $\mathcal{T}'' \models (\neg \diamond(\text{end}(\alpha_C) \wedge \text{end}(\beta_C)))$ . This implies that  $T''_s(\alpha_C) \neq T''_s(\beta_C)$  and  $T''_e(\alpha_C) \neq T''_e(\beta_C)$ . By definition of  $\mathcal{T}'$ ,  $C$  does not occur in  $\phi^{\leq 1}$  i.e.  $|\text{Instances}'(C)| = \emptyset$ . Therefore,  $\forall t \in \mathbb{D}, \langle \mathcal{T}'', t \rangle \models \text{start}(\alpha_C) \wedge \text{start}(\beta_C)$  iff  $\langle \mathcal{T}', t \rangle \models \text{start}(C)$  and similarly for  $\text{end}(\alpha_C) \wedge \text{end}(\beta_C)$  and

$\text{end}(C)$  (because both sides are always false). Since  $\alpha_C$  and  $\beta_C$  only appear in  $\phi^{\leq 1}$  exactly where there was a start/end predicate for  $C$  in  $\phi^{\leq 1}$ , they are semantically equivalent, and the rest of the formula was unchanged,  $\mathcal{T}' \models \phi^{\leq 1}$ . This proof can be reversed to prove the converse.

*Claim:*  $\mathcal{T} \models \phi$  iff  $\mathcal{T}' \models \phi^{\leq 1}$ .

*Proof:* Consider a  $= k$ -bounded activity  $A$  in  $\phi$ . For all  $t \in \mathbb{D}$ ,  $\langle \mathcal{T}, t \rangle \models \text{start}(A)$  iff  $\exists x \in \text{Instances}(A)$  such that  $T_s(x) = t$ . Since  $\text{Instances}(A) = \bigcup_{1 \leq i \leq k} \text{Instances}'(B_i)$ , where  $B_1 \dots B_k$  are the  $= 1$ -bounded activities  $A$  is replaced with in  $\phi^{\leq 1}$ ,  $\langle \mathcal{T}, t \rangle \models \text{start}(A)$  iff  $\exists x \in \bigcup_{1 \leq i \leq k} \text{Instances}'(B_i)$  such that  $T_s(x) = t$ .

This is equivalent to the statement  $\langle \mathcal{T}, t \rangle \models \text{start}(A)$  iff  $\bigvee_{1 \leq i \leq k} (T_s(\text{Instances}'(B_i)) = t)$ , which is equivalent (by construction of  $\phi^{\leq 1}$  and  $\mathcal{T}'$ ) to  $\langle \mathcal{T}', t \rangle \models \bigvee_{1 \leq i \leq k} \text{start}(B_i)$ .

Therefore, for all  $t \in \mathbb{D}$ ,  $\langle \mathcal{T}, t \rangle \models \text{start}(A)$  iff  $\langle \mathcal{T}', t \rangle \models \bigvee_{1 \leq i \leq k} \text{start}(B_i)$ . The same argument can be applied to  $\text{end}(A)$  and  $P(A)$  for any property  $P$ , and also for all  $\leq k$ -bounded activities in  $\phi$ .

Now that we have shown the atomic predicates of  $\phi$  are correctly represented by  $\phi^{\leq 1}$ , it remains to show that the operators all function identically as well.  $\wedge$ ,  $\neg$ , and  $\mathcal{U}_I$  operators are all left unchanged; only quantified formulae are modified. Consider a subformula of  $\phi$  of the form  $\forall x.\psi$ . For all  $t \in \mathbb{D}$ ,  $\langle \mathcal{T}, t \rangle \models \forall x.\psi$  iff for every activity instance  $a \in \mathcal{I}$ ,  $\langle \mathcal{T}, t_i \rangle \models \phi_{x=a}$ , where  $\phi_{x=a}$  is the formula  $\phi$  with all occurrences of  $x$  replaced with  $a$ . This is equivalent to the statement  $\langle \mathcal{T}, t \rangle \models \forall x.\psi$  iff  $\langle \mathcal{T}, t \rangle \models \bigwedge_{a \in \mathcal{I}} \phi_{x=a}$  (1).

We now show this statement is equivalent to  $\langle \mathcal{T}', t \rangle \models \bigwedge_{Y \in \mathcal{A}} (\diamond_{(-\infty, \infty)} \text{start}(Y) \supset \psi_{x=Y})$  (2). Note that the subscript in (2) ranges over all activities in  $\phi^{\leq 1}$  (recall that these are all  $= 1$  or  $\leq 1$ -bounded), whereas the subscript in (1) ranges over all instances in  $\phi$ . We created one activity in  $\phi^{\leq 1}$  for every possible instance of an activity in  $\phi$ , but the two sets may not be equal if some of the  $\leq 1$ -bounded activities do not occur - then the set of all instances  $\mathcal{I}$  is smaller than the set of activities  $\mathcal{A}$  in  $\phi^{\leq 1}$ . However, observe that  $\langle \mathcal{T}', t \rangle \models \diamond_{(-\infty, \infty)} \text{start}(Y)$  iff  $\text{Instances}'(Y) \neq \emptyset$ ; this is because as long as  $Y$  occurs (i.e.  $\text{Instances}'(Y) \neq \emptyset$ ), there must be some time at which  $Y$  starts, so  $\diamond_{(-\infty, \infty)} \text{start}(Y)$  must be true. Therefore, if  $\text{Instances}'(Y) = \emptyset$ ,  $\diamond_{(-\infty, \infty)} \text{start}(Y) \supset \psi_{x=Y}$  is equivalent to  $\perp \supset \psi_{x=Y}$  i.e.  $\top$ , and if  $\text{Instances}'(Y) \neq \emptyset$ , then  $\diamond_{(-\infty, \infty)} \text{start}(Y) \supset \psi_{x=Y}$  is equivalent to  $\top \supset \psi_{x=Y}$  i.e.  $\psi_{x=Y}$ . Now it is clear that  $\bigwedge_{Y \in \mathcal{A}} (\diamond_{(-\infty, \infty)} \text{start}(Y) \supset \psi_{x=Y})$  is in fact the conjunction of  $\psi_{x=Y}$  over all activities  $Y$  that actually occur, which is equivalent by construction of  $\phi^{\leq 1}$  and  $\mathcal{T}'$  to  $\langle \mathcal{T}', t \rangle \models \bigwedge_{a \in \mathcal{I}} \phi_{x=a}$ , which is true iff

$\langle \mathcal{T}, t \rangle \models \forall x.\psi$  from the previous paragraph.

Since the truth value of all atomic predicates at all times is preserved in the translation from  $\phi$  to  $\phi^{\leq 1}$  and the behavior of all operators, including quantifiers, is also preserved,  $\mathcal{T} \models$

$\phi$  iff  $\mathcal{T}' \models \phi^{\leq 1}$ .

## 9.4 Correctness of SMTL-to-FOL

Since this proof only concerns exactly-once SMTL formulae, each activity has exactly one instance, so we will treat the two as equivalent.

*Claim:* Let  $\mathcal{T} = (\text{Instances}, T)$  be a schedule and  $T'$  be a first-order variable assignment over variables  $\{\text{start}(a), \text{end}(a) \mid a \in A\}$  such that  $T'(\text{start}(a)) = t$  iff  $T_s(a) = t$  and  $T'(\text{end}(a)) = t$  iff  $T_e(a) = t$  for all  $a \in A$ . Then  $\mathcal{T}$  satisfies an exactly-once SMTL formula  $\phi$  iff variable assignment  $T$ , along with the standard interpretation of nonlogical symbols, satisfies  $\Phi_{\text{FOL}}$ .

*Proof:* First we begin by proving a statement about the helper method, **recursiveFOL**.

*Claim:* Let  $\mathcal{T} = (\text{Instances}, T)$  be a schedule,  $\phi$  be an exactly-once SMTL formula, and  $\mathcal{I}$  be the standard interpretation for the symbols  $+, \leq$  and all constants in  $\mathbb{D}$  in first-order logic. For all  $t \in \mathbb{D}$ ,  $\langle \mathcal{T}, t \rangle \models \phi$  iff  $\langle \mathcal{I}, T' \rangle \models \text{recursiveFOL}(\phi, t)$ . *Proof:* By structural induction over well-formed exactly-once SMTL formulae.

• **Base case:**  $\phi = \text{start}(a)$ .

Assume  $\langle \mathcal{T}, t \rangle \models \text{start}(a)$ . Then by definition  $T_s(a) = t$ . **recursiveFOL**( $\phi, t$ ) returns the formula  $\text{start}(a) = t$ .  $\langle \mathcal{I}, T' \rangle$  satisfies this formula because  $T_s(a) = t$  as stated, so  $T'(\text{start}(a)) = t$ . Therefore  $\langle \mathcal{I}, T' \rangle \models \text{recursiveFOL}(\phi, t)$ . The proof can be reversed to prove the converse.

• **Base case:**  $\phi = \text{end}(a)$ .

Analogous to the  $\text{start}(a)$  case.

• **Inductive Case:**  $\phi = \psi \wedge \chi$ .

Assume  $\langle \mathcal{T}, t \rangle \models \psi \wedge \chi$ . Then  $\langle \mathcal{T}, t \rangle \models \psi$ , so  $\langle \mathcal{I}, T' \rangle \models \text{recursiveFOL}(\psi, t)$  by the inductive hypothesis. Similarly,  $\langle \mathcal{I}, T' \rangle \models \text{recursiveFOL}(\chi, t)$ . Therefore,  $\langle \mathcal{I}, T' \rangle \models \text{recursiveFOL}(\psi, t) \wedge \text{recursiveFOL}(\chi, t)$ , so  $\langle \mathcal{I}, T' \rangle \models \text{recursiveFOL}(\phi, t)$ . The proof can be reversed to prove the converse.

• **Inductive Case:**  $\phi = \neg\psi$ .

Assume  $\langle \mathcal{T}, t \rangle \models \neg\psi$ . Then  $\langle \mathcal{T}, t \rangle \not\models \psi$ , so  $\langle \mathcal{I}, T' \rangle \not\models \text{recursiveFOL}(\psi, t)$  by the inductive hypothesis. Therefore,  $\langle \mathcal{I}, T' \rangle \models \neg \text{recursiveFOL}(\psi, t)$ , so  $\langle \mathcal{I}, T' \rangle \models \text{recursiveFOL}(\phi, t)$ . The proof can be reversed to prove the converse.

• **Inductive Case:**  $\phi = \psi \mathcal{U}_I \chi$ .

Assume  $\langle \mathcal{T}, t \rangle \models \psi \mathcal{U}_I \chi$ . Then by definition there exists  $t_i$  such that  $\langle \mathcal{T}, t_i \rangle \models \chi$  and  $t_i - t \in I$ , and  $\forall t_k$  such that  $t \leq t_k \leq t_i$  (or  $t_i \leq t_k \leq t$ ),  $\langle \mathcal{T}, t_k \rangle \models \psi$ . Using the induction hypothesis yields the statement: there exists  $t_i$  such that  $\langle \mathcal{I}, T' \rangle \models \text{recursiveFOL}(\chi, t_i)$  and  $t_i - t \in I$ , and  $\forall t_k$  such that  $t \leq t_k \leq t_i$  (or  $t_i \leq t_k \leq t$ ),  $\langle \mathcal{I}, T' \rangle \models \text{recursiveFOL}(\psi, t_k)$ . Let  $I = [l, u]$ ,  $i = t_i - t$ , and  $k = t_k - t$ . The semantics of FOL combined with some simple substitutions using the variables just defined allow us to convert this statement to the formula

$$\exists i (l \leq i \wedge i \leq u \wedge \text{recursiveFOL}(\chi, t + i) \wedge (i \geq 0 \supset \neg \exists k (0 \leq k \wedge k \leq i \wedge$$

$\neg\text{recursiveFOL}(\psi, t + k))$   
 $\wedge(i < 0 \supset \neg\exists k(i \leq k \wedge k \leq 0 \wedge$   
 $\neg\text{recursiveFOL}(\psi, t + k)))$ .

( $\forall$  is replaced with  $\neg\exists\neg$ ). Now we can conclude that  $\langle \mathcal{I}, T' \rangle \models \text{recursiveFOL}(\psi \mathcal{U}_I \chi, t)$ , so  $\langle \mathcal{I}, T \rangle \models \text{recursiveFOL}(\phi, t)$ . The proof can be reversed to prove the converse.

With this result, we can now easily prove our main statement:  $\mathcal{T} \models \phi$  iff  $\langle \mathcal{T}, 0 \rangle \models \phi$ , which is true iff  $\langle \mathcal{I}, T' \rangle \models \text{recursiveFOL}(\phi, 0)$  (by the above claim), which is exactly  $\Phi_{\text{FOL}}$ , so  $\langle \mathcal{I}, T' \rangle \models \Phi_{\text{FOL}}$ .

## 9.5 Fourier-Motzkin Elimination is applicable

Fourier-Motzkin elimination is used to eliminate variables from a system of linear equalities. We will assume all quantifiers are existential (all subformulae of the form  $\forall x.\psi$  can be replaced with  $\neg\exists x.\neg\psi$ ). Observe that the most deeply nested quantified subformula of  $\Phi_{\text{FOL}}$  consists of a single quantifier over a propositional formula with linear inequalities as atomic propositions. This formula must have an equivalent DNF formula; each of these clauses is a conjunction of linear inequalities (the negation of a linear inequality is also a linear inequality). Each clause is therefore a system of linear inequalities, so Fourier-Motzkin elimination can be used in each clause to eliminate the one quantified variable, which allows us to drop the quantifier over the entire subformula. This results in a new most-deeply-nested quantified subformula with the same format as before (boolean formula of linear equalities), so the method can be repeated until no quantifiers remain.

## 9.6 BI-SMTL-SAT is in PSPACE

We begin with a preliminary result:

**Theorem 9.1.** *Given a BI-SMTL formula  $\phi$  and a schedule  $\mathcal{T}$ , determining whether  $\mathcal{T} \models \phi$  is in PSPACE.*

*Proof.* We base our method on the standard approach for showing that evaluating the truth value of a fully quantified QBF formula is in PSPACE. This algorithm simply recursively evaluates the truth of each subformula; for a  $\exists$ -quantified subformula, it takes the disjunction of the truth value of that subformula with the quantified variable replaced with  $\top$  and the truth value of that subformula with the quantified variable replaced with  $\perp$ ; for  $\forall$ -quantified subformulae a conjunction is used instead. This algorithm stores a value for each nested quantifier, so the total space used is linear in the length of the formula.

The same approach can be applied to BI-SMTL formulae. The verifying algorithm simply recursively evaluates the truth of each subformula (e.g.  $\text{Verify}(\mathcal{T} \models \psi \wedge \chi)$  returns  $\text{Verify}(\mathcal{T} \models \psi) \wedge \text{Verify}(\mathcal{T} \models \chi)$ ); on a quantified subformula the verifier evaluates the subformula with every instance substituted into the subformula and conjoins (for  $\forall$ ) or disjoins (for  $\exists$ ) the results one at a time. The amount of space required is linear in the number of operators, which is linear in the size of the formula, so the entire algorithm runs in polynomial space.  $\square$

With this knowledge in hand, one naive method of solving BI-SMTL-SAT in polynomial space is to simply guess (or iterate through all) possible schedules, and check if each one satisfies the given formula. The rest of this section will be devoted to showing that we only need to search through a finite set of these, and hence it is possible to determine a formula is unsatisfiable when the search space is exhausted.

**Definition 9.1.** *Given a schedule  $\mathcal{T} = (\text{Instances}, T)$ , the makespan of  $\mathcal{T}$  is the latest time assigned to any instance, i.e.  $\max_{a \in \mathcal{I}} T_e(a)$ .*

Given a BI-SMTL formula  $\phi$  over  $n$  activities with bounds  $b_1 \dots b_n$ , let  $B_\phi = \sum_{1 \leq i \leq n} b_i$ . Let  $\mathcal{U}_{I_1} \dots \mathcal{U}_{I_k}$  be the set of

all  $\mathcal{U}_I$  operators that appear in  $\phi$ , where  $I_i = [l_i, u_i]$  for all  $i$ . Let  $m_\phi = \sum \{u_i \mid u_i \neq \infty\} + \{l_i \mid u_i = \infty \text{ and } l_i \neq \infty\}$ , i.e.  $m_\phi$  is the sum of all finite upper bounds of intervals that appear in  $\phi$ , added to the sum of all finite lower bounds for intervals where the upper bound is infinite. Finally, let  $M_\phi = m_\phi \dot{B}_\phi$ .

We claim that  $M_\phi$  is an upper bound on the makespan required to find a satisfying schedule. This idea is formalized in the following statement.

**Theorem 9.2.** *For every BI-SMTL formula  $\phi$ , if  $\mathcal{T} \models \phi$  and the makespan of  $\mathcal{T}$  is greater than  $M_\phi$ , there exists a schedule  $\mathcal{T}'$  such that  $\mathcal{T}' \models \phi$  and the makespan of  $\mathcal{T}'$  is less than or equal to  $M_\phi$ .*

*Proof.* The intuition behind this statement is that  $\phi$  can only constrain the timing of activities within a certain distance from each other; that distance is governed by all the intervals used in the temporal operators. If activities are scheduled beyond a certain distance, then there is “slack” in the schedule that can be tightened without changing the truth value of the formula.  $\square$

## 9.7 SAT $\leq$ BI-QF-SMTL

Given a propositional boolean formula  $\phi_{\text{SAT}}$ , construct a BI-QF-SMTL formula  $\phi_{\text{SMTL}}$  as follows:

- For each proposition  $p$  in the set of all propositions  $P$  in  $\phi_{\text{SAT}}$ , add activity  $A_p$  to  $\mathcal{A}$  with bound =1.
- For each proposition  $p$  in  $\phi_{\text{SAT}}$ , replace all occurrences of  $p$  with  $\text{start}(p)$ .
- All boolean operators remain unchanged.

This formula is clearly in BI-QF-SMTL since no quantifiers are introduced and all activities have bound =1.

*Claim:*  $\phi_{\text{SAT}}$  is satisfiable iff  $\phi_{\text{SMTL}}$  is satisfiable.

*Proof:*

$\Rightarrow$ : Assume  $\phi_{\text{SAT}}$  is satisfiable. Then there is a truth assignment  $V : P \mapsto \{\top, \perp\}$  over the propositions in  $\phi_{\text{SAT}}$  that evaluates to true. Define  $\mathcal{T}^V = (\text{Instances}^V, T^V)$  as follows:  $\text{Instances}^V$  is any 1-to-1 function from activities to instances (since all activities are =1-bounded). For each activity  $A_p$  with instance  $a_p$ , let  $T^V(a_p) = (0, 2)$  if  $V(p) = \top$  and  $T^V(a_p) = (1, 2)$  if  $V(p) = \perp$  (the end time 2 is not important; this is an arbitrary value). We claim  $\mathcal{T}^V \models \phi_{\text{SMTL}}$ .

By the semantics of SMTL,  $\mathcal{T}^V \models \text{start}(A_p)$  iff  $T_s^V(a_p) = 0$ ; therefore  $\text{start}(A_p) \equiv \top$  if  $V(p) = \top$  and  $\text{start}(A_p) \equiv \perp$  if  $V(p) = \perp$ . Since the atomic predicates of  $\phi_{SMTL}$  under  $\mathcal{T}^V$  are equivalent to the propositions of  $\phi_{SAT}$  under  $V$ , and the semantics for  $\wedge$  and  $\neg$  are identical in both domains, if  $V$  is an assignment that satisfies  $\phi_{SAT}$ , then  $\mathcal{T}^V \models \phi_{SMTL}$ .  $\Leftarrow$ : Given a schedule  $\mathcal{T} = (\text{Instances}, T)$  such that  $\mathcal{T} \models \phi_{SMTL}$ , construct a truth assignment  $V^T$  for the propositions in  $\phi_{SAT}$  as follows: for each proposition  $p$ ,  $V^T(p) = \top$  if  $T_s(a_p) = 0$  and  $V^T(p) = \perp$  if  $T_s(a_p) \neq 0$ . The rest of the proof is the same as above; using the fact that  $\mathcal{T} \models \text{start}(A_p)$  iff  $T_s(a_p) = 0$ , it follows that  $V^T(p) = \top$  iff  $\text{start}(A_p) \equiv \top$ , and the identical boolean operators guarantee that if  $\mathcal{T} \models \phi_{SMTL}$ , then  $V^T$  satisfies  $\phi_{SAT}$ .

## 9.8 Correctness of BIQFSMTL-Verify

*Claim:* **BIQFSMTL-Verify**( $\mathcal{T}, \phi$ ) returns true iff  $\mathcal{T} \models \phi$ .

First we begin with a statement about the helper method.

*Claim:* For any time  $t \in \mathbb{D}$ ,  $\langle \mathcal{T}, t \rangle \models \phi$  iff  $t \in \text{trueTimes}(\mathcal{T}, \phi)$ .

*Proof:* By structural induction over BI-QF-SMTL formulae.

- **Base Case:**  $\phi = \text{start}(A)$ .

Then **trueTimes**( $\mathcal{T}, \phi$ ) returns the set of points  $\{T_s(a) \mid a \in \text{Instances}(A)\}$ . By definition,  $\langle \mathcal{T}, t \rangle \models \text{start}(A)$  iff  $\exists a \in \text{Instances}(A)$  such that  $t = T(a)$ , so this is correct.

- **Inductive Case:**  $\phi = \psi \wedge \chi$ .  $\langle \mathcal{T}, t \rangle \models \psi \wedge \chi \Leftrightarrow \langle \mathcal{T}, t \rangle \models \psi$  and  $\langle \mathcal{T}, t \rangle \models \chi \Leftrightarrow t \in \text{trueTimes}(\mathcal{T}, \psi)$  and  $t \in \text{trueTimes}(\mathcal{T}, \chi)$  (by induction hypothesis)  $\Leftrightarrow t \in \text{trueTimes}(\mathcal{T}, \psi) \cap \text{trueTimes}(\mathcal{T}, \chi) \Leftrightarrow t \in \text{trueTimes}(\mathcal{T}, \phi)$ .
- **Inductive Case:**  $\phi = \neg\psi$ .  $\langle \mathcal{T}, t \rangle \models \neg\psi \Leftrightarrow \langle \mathcal{T}, t \rangle \not\models \psi \Leftrightarrow t \notin \text{trueTimes}(\mathcal{T}, \psi)$  (by induction hypothesis)  $\Leftrightarrow t \in \text{trueTimes}(\mathcal{T}, \psi)^C \Leftrightarrow t \in \text{trueTimes}(\mathcal{T}, \neg\psi)$ .
- **Inductive Case:**  $\phi = \psi \mathcal{U}_I \chi$ .  $\langle \mathcal{T}, t \rangle \models \psi \mathcal{U}_I \chi \Leftrightarrow \exists t_j$  such that  $\langle \mathcal{T}, t_j \rangle \models \psi$ ,  $t_j - t \in I$ , and for all  $t \leq t_k \leq t_j$  (or  $t_j \leq t_k \leq t$ ),  $\langle \mathcal{T}, t_k \rangle \models \chi \Leftrightarrow \exists t_j \in \text{trueTimes}(\mathcal{T}, \psi)$ ,  $t_j - t \in I$ , and for all  $t \leq t_k \leq t_j$  (or  $t_j \leq t_k \leq t$ ),  $t_k \in \text{trueTimes}(\mathcal{T}, \psi)$  (by induction hypothesis)  $\Leftrightarrow t \in \text{trueTimes}(\mathcal{T}, \psi \mathcal{U}_I \chi)$  (let the  $i$  in line 16 of the algorithm be  $t_j - t$ ).

With this result we can then prove the full claim:

**BIQFSMTL-Verify**( $\mathcal{T}, \phi$ ) returns true  $\Leftrightarrow$

$0 \in \text{trueTimes}(\mathcal{T}, \phi) \Leftrightarrow \langle \mathcal{T}, 0 \rangle \models \phi$  (by the above lemma)  $\Leftrightarrow \mathcal{T} \models \phi$ .

## 9.9 JSP $\leq$ BI-QF-SMTL

First we repeat the definition of a job shop scheduling problem (JSP):

A set of jobs must be completed on a set of machines, which can process at most one job at a time. A

job may require the use of multiple machines in a fixed order: job  $J$  is associated with a sequence of machines  $S(J, 1) \dots S(J, n)$  and processing times on these machines,  $P(J, 1) \dots P(J, n)$ . The solution to the problem is an assignment of times to jobs that respects the processing times of each job, the sequence ordering of the machines for each job, and the capacity of each machine.

Let  $JSP$  be an instance of a job shop scheduling problem, and let  $\Phi_{JSP}$  be the set of BI-QF-SMTL formula resulting from the construction given in Section 6.1.

**Theorem 9.3.** *JSP is satisfiable iff  $\Phi_{JSP}$  is satisfiable.*

*Proof.* Assume  $JSP$  is satisfiable. Then there exists an assignment of times to the machine sequence of each job that respects the processing times and order of machines. Construct a schedule  $\mathcal{T}$  from this assignment by setting  $T(a_{J,i}) = (s_{J,i}, e_{J,i})$ , where  $s_{J,i}$  is the start time assigned to the  $i$ th machine in job  $J$ 's sequence, and  $e_{J,i}$  is its end time. We claim  $\mathcal{T} \models \Phi_{JSP}$ . First, all the processing time formulae are satisfied: the formula  $\text{start}(a_{J,i}) \rightarrow_{[P(J,i), P(J,i)]} \text{end}(a_{J,i})$  is equivalent to  $\diamond(\text{start}(a_{J,i}) \wedge \diamond_{[P(J,i), P(J,i)]} \text{end}(a_{J,i}))$ , which translates via the semantics to  $T_e(a_{J,i}) - T_s(a_{J,i}) \in [P(J,i), P(J,i)]$  i.e.  $e_{J,i} - s_{J,i} = P(J,i)$ , which is true by the assumption that the original time assignment satisfies  $JSP$ . Therefore  $\mathcal{T} \models \text{start}(a_{J,i}) \rightarrow_{[P(J,i), P(J,i)]} \text{end}(a_{J,i})$ .

The formulae added for preserving the machine sequence of each job are also satisfied similarly:  $\Phi_{JSP}$  includes formulae  $\text{end}(a_{J,k}) \rightarrow_{[0, \infty)} \text{start}(a_{J,k+1})$ , which are true iff  $T_s(J, k+1) - T_e(J, k) \geq 0$ ; this is true because the solution to  $JSP$  respects the machine sequence of each job, i.e.  $s_{J,k+1} - e_{J,k} \geq 0$ . Therefore  $\mathcal{T} \models \text{end}(a_{J,k}) \rightarrow_{[0, \infty)} \text{start}(a_{J,k+1})$ .

Finally, the formula  $\bigwedge_{a \in U_M, b \in U_M, a \neq b} \square \neg(\text{Currently}(a) \wedge \text{Currently}(b))$  is true iff for every two jobs that require the same machine in their sequence,  $\square \neg(\text{Currently}(a) \wedge \text{Currently}(b))$  is true, where  $a$  is the interval the first job uses that machine and  $b$  is the interval the second job uses the same machine. This subformula is true iff  $\forall t$ , it is not the case that  $s_a \leq t \leq e_a$  and  $s_b \leq t \leq e_b$  (where  $s_a, e_a$  are the start and end times of  $a$  and similarly for  $b$ ). This is equivalent to stating that the intervals  $[s_a, e_a]$  and  $[s_b, e_b]$  do not overlap for every  $a$  and  $b$  on the same machine, which is true because the solution to  $JSP$  respects this constraint. Therefore  $\mathcal{T} \models \bigwedge_{a \in U_M, b \in U_M, a \neq b} \square \neg(\text{Currently}(a) \wedge \text{Currently}(b))$ .

Since all the formulae in  $\Phi_{JSP}$  are satisfied by  $\mathcal{T}$ ,  $\mathcal{T} \models \Phi_{JSP}$ . The proof in the other direction is analogous; assuming there is some schedule that satisfies  $\Phi_{JSP}$ , an assignment of times to jobs in  $JSP$  is constructed in a straightforward manner and the same arguments for each formula in  $\Phi_{JSP}$  are applied.  $\square$

## 9.10 DTN $\leq$ BI-QF-SMTL

Given a DTN, we can model it via an exactly-once STL formula  $\phi_{DTN}$  as follows:

- The set of activities of the STL formula corresponds to the set of events of the STN. They are bounded to occur

exactly once. The events in the STN are instantaneous, so we include a formula  $\diamond(\text{start}(A) \wedge \text{end}(A))$  for each activity  $A$  to force the start and end of an activity to be simultaneous. We use  $A$  in place of  $\text{start}(A)$  and  $\text{end}(A)$  for the remainder of this section for convenience.

- Let the DTN be the set of disjunctive temporal constraints  $\{D_1 \dots D_n\}$ . For each constraint  $D_i = C_{i_1} \vee \dots \vee C_{i_k}$ , where each  $C_{i_j}$  is a simple temporal constraint  $[l, u]_{XY}$ , let  $\chi_{i_j} = X \rightarrow^{[l, u]} Y$  for  $1 \leq j \leq k$ .
- Let  $\psi_i = \chi_{i_1} \vee \dots \vee \chi_{i_k}$ .
- Let  $\phi_{DTN} = \psi_1 \wedge \dots \wedge \psi_n$ , conjoined with the instantaneous-activity constraints mentioned above.

Given some assignment of times  $S$  to the events in  $D$ , define schedule  $\mathcal{T}_S = (\text{Instances}_S, T_S)$  as follows:  $\text{Instances}_S$  is any isomorphism (recall that the set of activities is identical to the set of events, and they are all  $\leq 1$ -bounded) and for any event  $e$  in  $D$  with corresponding activity  $A_e$  in  $\phi_{DTN}$ ,  $T_S(A_e) = (S(e), S(e))$  (the start and end are identical because we are modeling an instantaneous event).

**Theorem 9.4.** *For any DTN  $D$  and time assignment  $S$  over the events in  $D$ ,  $S$  satisfies the constraints in  $D$  iff  $\mathcal{T}_S \models \phi_{DTN}$ .*

*Proof.*  $\Leftarrow$ : Let  $D = \{D_1 \dots D_n\}$ , where  $D_i$  is a single disjunctive temporal constraint.  $D_i$  is a disjunction of simple temporal constraints  $C_{i_1} \vee \dots \vee C_{i_k}$ . If  $\mathcal{T}_S \models \phi_{DTN}$ ,  $\langle \mathcal{T}_S, 0 \rangle \models \phi_{DTN}$ , and by the semantics of  $\wedge$ ,  $\langle \mathcal{T}_S, 0 \rangle \models \psi_i$ , i.e.  $\langle \mathcal{T}_S, 0 \rangle \models \chi_{i_1} \vee \dots \vee \chi_{i_k}$  for every  $i$ .

By the semantics of  $\vee$ ,  $\langle \mathcal{T}_S, 0 \rangle \models \chi_{i_j}$  for some  $j$ , where  $\chi_{i_j} = X \rightarrow^{[l, u]} Y$ .  $X$  is an activity and technically should either be  $\text{start}(X)$  or  $\text{end}(X)$ , but since  $\phi_{DTN} \models \diamond(\text{start}(X) \wedge \text{end}(X))$ , either predicate can be used without changing the schedule. The same is true for  $Y$ .

By the semantics of  $\rightarrow^I$ , this means  $\exists t_j, t_k > 0$  such that  $\langle \mathcal{T}_S, t_j \rangle \models X$ ,  $\langle \mathcal{T}_S, t_k \rangle \models Y$ , and  $t_k - t_j \in [l, u]$ .  $\langle \mathcal{T}_S, t_j \rangle \models X$  iff  $T_S(X) = (t_j, t_j)$  and similarly  $\langle \mathcal{T}_S, t_k \rangle \models Y$  iff  $T_S(Y) = (t_k, t_k)$ . This implies that  $T_S(Y) - T_S(X) \in [l, u]$ , which mean  $S(Y) - S(X) \in [l, u]$ , which is the definition of  $C_{i_j}$ , so constraint  $C_{i_j}$  is satisfied. Consequently,  $D_i$  is satisfied as well. This is true for all  $1 \leq i \leq n$ , so  $\mathcal{T}_S$  satisfies the entire DTN.

$\Rightarrow$ : Let  $S$  be a valid schedule for the DTN  $\{D_1, \dots, D_n\}$ . Consider  $D_i = C_{i_1} \vee \dots \vee C_{i_k}$ . Since  $S$  satisfies  $D_i$ , it must satisfy one of  $C_{i_1} \vee \dots \vee C_{i_k}$ ; call this  $C_{i_j}$ . Let  $C_{i_j}$  be the STN constraint  $[l, u]_{XY}$ ; then  $S$  must have  $l \leq S(Y) - S(X) \leq u$ . This also means that  $T_S(Y) - T_S(X) \in [l, u]$ , so  $\mathcal{T}_S \models X \rightarrow^{[l, u]} Y$  i.e.  $\mathcal{T}_S \models \chi_{i_j}$ . Therefore  $\mathcal{T}_S \models \psi_i$ . Since this is true for all  $1 \leq i \leq n$ ,  $\mathcal{T}_S \models \psi_1 \wedge \dots \wedge \psi_n$ , and since the start and end of each activity is identical in  $T_S$ ,  $\mathcal{T}_S \models \diamond(\text{start}(X) \wedge \text{end}(X))$  for every activity  $X$ . Therefore,  $\mathcal{T}_S \models \phi_{DTN}$ .  $\square$

## 9.11 CTP $\leq$ BI-QF-SMTL

We define a controllable conditional temporal problem (CCTP) in the following way (identical to the definition in (Yu and Williams 2013), but without preferences and relaxations):

- $P$ , a set of controllable finite domain discrete variables (we will additionally assume this domain is a subset of  $\mathbb{N}$ ).
- $V$ , the set of (instantaneous) events to be scheduled.
- $E$ , the set of (simple or disjunctive) temporal constraints between events in  $V$ .
- $L_v : V \mapsto Q$ , a labeling function that assigns a conjunction of variable assignments in  $P$  to an event in  $V$ .
- $L_p : P \mapsto Q$ , a labeling function that assigns a conjunction of variable assignments in  $P$  to a variable in  $P$ .

Given an assignment of values to the variables in  $P$ , we will call a variable  $p \in P$  *activated* if all the variables in  $L_p(p)$  are activated, and the assignment satisfies all the assignments in  $L_p(p)$ . Similarly, an event  $v \in V$  is activated if all the variables in  $L_v(v)$  are activated, and the variable assignment satisfies all the assignments in  $L_v(v)$ . Finally, a constraint  $c \in E$  is activated if all the events involved in the constraint are activated. A solution to the CCTP is an assignment of values to the variables in  $P$  and times to the events in  $V$  such that the times of all the activated events satisfy all the activated temporal constraints.

In order to represent a CCTP in BI-QF-SMTL, we will create one  $\leq 1$ -bounded activity for each event  $v \in V$ ; call it  $A_v$ . Since these events are instantaneous, we will simply use  $A_v$  to represent both the start and end of an activity; this can be formalized by including the formula  $\Box \text{start}(a) \supset \text{end}(a)$  for each activity  $a$ . Additionally, we introduce a  $\leq 1$ -bounded activities for each decision variable in  $p \in P$ ; call it  $A_p$  (these will also be instantaneous). The time assigned to this activity corresponds to the value assigned to it; if the variable is not activated, it will not occur, hence the use of the  $\leq 1$  bound.

We now describe a set of BI-QF-SMTL formulae  $\Phi_{CCTP}$  to represent the CCTP.

- For each decision variable  $p \in P$ , let  $L_p(p)$  be the variable assignment  $p_1 = k_1, p_2 = k_2, \dots, p_n = k_n$ . Add the formula  $\bigwedge_{1 \leq i \leq n} \diamond_{[k_i, k_i]} A_{p_i} \equiv \diamond A_p$  to  $\Phi_{CCTP}$ . This formula forces variable  $p$  to occur (i.e. it is activated) iff all the variables in its label are activated and are assigned the appropriate value.
- For each event  $v \in V$ , let  $L_v(v)$  be the variable assignment  $p_1 = k_1, p_2 = k_2, \dots, p_n = k_n$ . Add the formula  $\bigwedge_{1 \leq i \leq n} \diamond_{[k_i, k_i]} A_{p_i} \equiv \diamond A_v$  to  $\Phi_{CCTP}$ . This formula forces event  $v$  to occur (i.e. it is activated) iff all the variables in its label are activated and are assigned the appropriate value.
- For each temporal constraint  $c \in E$ , let  $c_{SMTL}$  be the SMTL representation of  $c$  as described in Section 9.10, and let  $\text{Events}(c)$  be the set of all events mentioned in constraint  $c$ . Add the formula  $(\bigwedge_{x \in \text{Events}(c)} \diamond x) \supset c_{SMTL}$  to  $\Phi_{CCTP}$ , which states that if all the events in a constraint are activated, then that constraint must be satisfied.

**Theorem 9.5.** *A CCTP has a solution iff  $\Phi_{CCTP}$  is satisfiable.*

*Proof.* Assume the CCTP has a solution  $(S_p, S_v)$  where  $S_p$  is the variable assignment for  $P$  and  $S_v$  is the time assignment for  $V$ . We construct a schedule  $\mathcal{T} = (\text{Instances}, T)$  using the  $\leq 1$ -bounded activities described above. For each variable  $p \in P$  that is activated and is assigned value  $S_p(p)$ , let  $\text{Instances}(A_p) = \{a_p\}$  and  $T(a_p) = (S_p(p), S_p(p))$ . For each variable in  $q \in P$  that is not activated, let  $\text{Instances}(A_p) = \emptyset$ . For each event  $v \in V$  that is activated and is assigned value  $S_v(v)$ , let  $\text{Instances}(A_v) = \{a_v\}$  and  $T(a_v) = (S_v(v), S_v(v))$ . For each variable in  $w \in V$  that is not activated, let  $\text{Instances}(A_w) = \emptyset$ . Now, we claim that  $\mathcal{T} \models \Phi_{\text{CCTP}}$ . Note that a variable or event  $x$  is activated in the CCTP iff  $\mathcal{T} \models \diamond A_x$ , by the formulae in the first two bullet points. This is because by the semantics of SMTL,  $\mathcal{T} \models \diamond_{[k_i, k_i]} A_{p_i}$  iff  $\text{Instances}(A_{p_i}) = a_{p_i}$  (i.e. the activity occurs) and  $T(a_{p_i}) = (k_i, k_i)$ , so an activity corresponding to a decision variable occurs iff all the decision variables in its label occur and are assigned the values that satisfy the label. The same is true for events. Finally, the formulae added by the last bullet point are satisfied when all the constraints are satisfied in which all the activities corresponding to the events in the constraint occur (the correctness of converting  $c$  to  $c_{\text{SMTL}}$  was proved in Section 9.10). This is true iff every constraint with all its events activated is satisfied by the CCTP solution, which it is.

Assuming a schedule  $\mathcal{T} = (\text{Instances}, T)$  satisfies  $\Phi_{\text{CCTP}}$ , one can construct a solution to the CCTP in the same manner as above (an event or variable is activated if the activity corresponding to it occurs, and its value is the time assigned to it) and argue the correctness similarly.

Note that the SMTL formulation of CCTP can be easily expanded to include labels that use ranges of variable assignments (e.g.  $2 \leq p \leq 5$ ) rather than specific values, as well as continuous decision variables and durative actions.  $\square$

## 9.12 Correctness of SMTL-to-STN

Let  $\phi$  be an exactly-once  $\{\wedge, \diamond_I\}$ -formula, and  $\text{STN}_\phi$  be the STN returned by **SMTL-to-STN**( $\phi$ ); recall that this STN is over the set of events  $E = \{\text{start}(a) \mid a \in \mathcal{A}\} \cup \{\text{end}(a) \mid a \in \mathcal{A}\} \cup \{\psi \mid \psi \text{ is a subformula of } \phi\}$ . Let  $T$  be a variable assignment over  $E$  that satisfies the constraints in  $\text{STN}_\phi$ , and let  $T'$  assign times to activities such that  $T'_s(a) = t$  iff  $T(\text{start}(a)) = t$  and  $T'_e(a) = t$  iff  $T(\text{end}(a)) = t$  for all  $a \in \mathcal{A}$ . Let  $\mathcal{T} = (\text{Instances}, T')$ , where  $\text{Instances}$  is any isomorphism.

**Theorem 9.6.** *For every exactly-once  $\{\wedge, \diamond_I\}$ -formula  $\phi$ , the STN returned by **SMTL-to-STN**( $\phi$ ) is consistent iff  $\phi$  is satisfiable.*

*Proof.*  $\Rightarrow$ : Let  $\text{STN}_\phi$  be the STN returned by **SMTL-to-STN**( $\phi$ ); recall that this STN is over the set of events  $E = \{\text{start}(a) \mid a \in \mathcal{A}\} \cup \{\text{end}(a) \mid a \in \mathcal{A}\} \cup \{\psi \mid \psi \text{ is a subformula of } \phi\}$ . Assume  $\text{STN}_\phi$  is consistent; then there exists a variable assignment  $T$  over  $E$  that satisfies the constraints in  $\text{STN}_\phi$ . Let  $\mathcal{T} = (\text{Instances}, T')$ , where  $\text{Instances}$  is any isomorphism and  $T'$  assigns times to activities such that  $T'_s(a) = t$  iff  $T(\text{start}(a)) = t$  and  $T'_e(a) = t$  iff  $T(\text{end}(a)) = t$  for all  $a \in \mathcal{A}$ . We claim that  $\mathcal{T} \models \phi$ .

In order to prove this, we begin with a claim about the helper function.

**Lemma 9.1.** *For every subformula  $\alpha$  of  $\phi$ , if time assignment  $T$  satisfies the constraints returned by **AddExplicitConstraints**( $\alpha, \emptyset$ ), then there exists a schedule  $\mathcal{T}$  such that  $\mathcal{T} \models \alpha$  and for every subformula  $\beta$  of  $\alpha$ ,  $\langle \mathcal{T}, T(\beta) \rangle \models \beta$ .*

( $\mathcal{T}$  is as defined above;  $T(\beta)$  is the time the auxiliary node corresponding to  $\beta$  is scheduled.)

*Proof:* We will proceed by induction of the structure of well-formed exactly-once  $\{\wedge, \diamond_I\}$ -formulae.

- **Base Case:**  $\alpha = \text{start}(a)$ . **AddExplicitConstraints**( $\alpha, \emptyset$ ) does not return any constraints in this case, so any time assignment  $T$  will satisfy it; pick some value  $t$  and let  $T(\text{start}(a)) = t$ . Then  $T'_s(a) = t$ , so  $\langle \mathcal{T}, t \rangle \models \alpha$ .
- **Base Case:**  $\alpha = \text{end}(a)$ . Analogous to the  $\text{start}(a)$  case.
- **Inductive Case:**  $\alpha = \psi \wedge \chi$ . By the induction hypothesis, (1)  $\langle \mathcal{T}, T(\psi) \rangle \models \psi$  and (2)  $\langle \mathcal{T}, T(\chi) \rangle \models \chi$ . **AddExplicitConstraints**( $\alpha, \emptyset$ ) returns constraints  $\{[0, 0]_{\alpha, \psi}, [0, 0]_{\alpha, \chi}\}$  (in addition to the explicit constraints induced by  $\psi$  and  $\chi$ ). Therefore,  $T(\alpha) = T(\psi) = T(\chi)$ . These can be substituted into (1) and (2) to yield  $\langle \mathcal{T}, T(\alpha) \rangle \models \psi$  and  $\langle \mathcal{T}, T(\alpha) \rangle \models \chi$ , i.e.  $\langle \mathcal{T}, T(\alpha) \rangle \models \psi \wedge \chi$ .
- **Inductive Case:**  $\alpha = \diamond_I \psi$ . By the induction hypothesis,  $\langle \mathcal{T}, T(\psi) \rangle \models \psi$ . **AddExplicitConstraints**( $\alpha, \emptyset$ ) returns constraint  $I_{\alpha, \psi}$  (among the rest of the constraints). This implies  $T(\psi) - T(\alpha) \in I$ . Therefore, by the definition of  $\diamond_I \psi$  ( $t_i$  from the definition is  $T(\alpha)$  and  $t_j$  is  $T(\psi)$ ),  $\langle \mathcal{T}, T(\alpha) \rangle \models \phi$ .

With this claim proven, we can prove one direction of our main statement. Let  $T$  be a solution to the constraints given by **SMTL-to-STN**( $\phi$ ). It is clear from the algorithm that these constraints are the union of the explicit constraints returned by **AddExplicitConstraints**( $\phi, \emptyset$ ) and the initial constraint  $[0, 0]_{\text{START}, \phi}$ . From the above result, we know that  $\langle \mathcal{T}, T(\phi) \rangle \models \phi$ .  $T$  also satisfies the constraint  $[0, 0]_{\text{START}, \phi}$ , so  $T(\text{START}) = T(\phi)$ . Therefore,  $\langle \mathcal{T}, T(\text{START}) \rangle \models \phi$  i.e.  $\mathcal{T} \models \phi$ .

Now we prove the other direction.

$\Leftarrow$ : Assume  $\phi$  is satisfiable; let  $\mathcal{T} = (\text{Instances}, T)$  be a schedule witnessing this statement. Let  $T_{\mathcal{T}}$  be a time assignment for the STN constraints returned by **SMTL-to-STN**( $\phi$ ) constructed out of  $\mathcal{T}$  by setting  $T_{\mathcal{T}}(\text{start}(a)) = t$  iff  $T_s(a) = t$  and  $T_{\mathcal{T}}(\text{end}(a)) = t$  iff  $T'_e(a) = t$ . For each subformula  $\alpha$  of  $\phi$ ,  $T_{\mathcal{T}}(\alpha) = t$  for some  $t$  such that  $\langle \mathcal{T}, t \rangle \models \alpha$  (recall that the STN is defined over the start and end events of  $\phi$  as well as all its subformulae). Such a  $t$  is guaranteed to exist because the  $\neg$  is not permitted in  $\phi$  and thus the semantics of any exactly-once  $\{\wedge, \diamond_I\}$ -formula require all of its subformulae to be true at some time.

Again, we will prove a statement about the helper function:

**Lemma 9.2.** *If  $\mathcal{T} \models \phi$ , then time assignment  $T_{\mathcal{T}}$  satisfies the constraints returned by **AddExplicitConstraints** $(\alpha, \emptyset)$ .*

*Proof:* by induction over exactly-once  $\{\wedge, \diamond_I\}$ -formulae.

- **Base Case:**  $\alpha = \text{start}(a)$ .  
Since  $\mathcal{T} \models \phi$ , there is some  $t$  such that  $\langle \mathcal{T}, t \rangle \models \text{start}(a)$ . **AddExplicitConstraints** $(\alpha, \emptyset)$  does not return any constraints in this case, so  $T_{\mathcal{T}}$  will vacuously satisfy it.
- **Base Case:**  $\alpha = \text{end}(a)$ .  
Analogous to the  $\text{start}(a)$  case.
- **Inductive Case:**  $\alpha = \psi \wedge \chi$ .  
By the induction hypothesis,  $T_{\mathcal{T}}$  satisfies the constraints returned by **AddExplicitConstraints** $(\psi, \emptyset)$  and **AddExplicitConstraints** $(\chi, \emptyset)$ , so all that remains is to show that  $T_{\mathcal{T}}$  also satisfies  $[0, 0]_{\alpha, \psi}$  and  $[0, 0]_{\alpha, \chi}$ . By assumption, there exists a time  $t$  such that  $\langle \mathcal{T}, t \rangle \models \alpha$ , and since  $\alpha = \psi \wedge \chi$ ,  $\langle \mathcal{T}, t \rangle \models \psi$  and  $\langle \mathcal{T}, t \rangle \models \chi$  as well. This implies that  $T_{\mathcal{T}}(\alpha) = T_{\mathcal{T}}(\psi) = T_{\mathcal{T}}(\chi)$ , and therefore  $[0, 0]_{\alpha, \psi}$  and  $[0, 0]_{\alpha, \chi}$  are satisfied by  $T_{\mathcal{T}}$  as well.
- **Inductive Case:**  $\alpha = \diamond_I \psi$ . By the induction hypothesis,  $T_{\mathcal{T}}$  satisfies the constraints returned by **AddExplicitConstraints** $(\psi, \emptyset)$ , so all that remains is to show that  $T_{\mathcal{T}}$  also satisfies constraint  $I_{\alpha, \psi}$ . By assumption, there exists a time  $t$  such that  $\langle \mathcal{T}, t \rangle \models \alpha$ , and since  $\alpha = \diamond_I \psi$ , there exists a time  $t_j$  such that  $\langle \mathcal{T}, t_j \rangle \models \psi$  and  $t_j - t \in I$ . Therefore, we can construct  $T_{\mathcal{T}}$  such that  $T_{sch}(\alpha) = t$  and  $T_{\mathcal{T}}(\psi) = t_j$ , which allows us to conclude that  $T_{\mathcal{T}}(\psi) - T_{\mathcal{T}}(\alpha) \in I$ , which by definition means that  $T_{\mathcal{T}}$  satisfies constraint  $I_{\alpha, \psi}$ .

Now that the lemma has been proven, we can finish our original claim. If  $\mathcal{T} \models \phi$ , then  $\langle \mathcal{T}, T(\text{START}) \rangle \models \phi$ . This means  $T_{\mathcal{T}}(\text{START}) = T_{\mathcal{T}}(\phi)$ , so the constraint  $[0, 0]_{\text{START}, \phi}$  is satisfied. All the other constraints in the STN are satisfied by the above lemma, so  $T_{\mathcal{T}}$  satisfies the entire STN.  $\square$

## References

- Alur, R.; Feder, T.; and Henzinger, T. A. 1996. The benefits of relaxing punctuality. *J. ACM* 43(1):116–146.
- Bacchus, F., and Kabanza, F. 1998. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence* 22(1-2):5–27.
- Baier, J. A., and McIlraith, S. A. 2006. Planning with temporally extended goals using heuristic search. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling, ICAPS 2006, Cumbria, UK, June 6-10, 2006*, 342–345.
- Bouyer, P.; Markey, N.; Ouaknine, J.; and Worrell, J. 2008. On expressiveness and complexity in real-time model checking. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part II, ICALP '08*, 124–135. Berlin, Heidelberg: Springer-Verlag.
- Burke, E. K.; De Causmaecker, P.; Berghe, G. V.; and Van Landeghem, H. 2004. The state of the art of nurse rostering. *J. of Scheduling* 7(6):441–499.
- Cimatti, A.; Hunsberger, L.; Micheli, A.; and Roveri, M. 2014. Using timed game automata to synthesize execution strategies for simple temporal networks with uncertainty.
- Cimatti, A.; Micheli, A.; and Roveri, M. 2012. Solving temporal problems using smt: Strong controllability. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming, CP'12*, 248–264. Berlin, Heidelberg: Springer-Verlag.
- De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, 854–860.
- De Giacomo, G.; De Masellis, R.; and Montali, M. 2014. Reasoning on LTL on finite traces: Insensitivity to infiniteness. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 1027–1033.
- De Moura, L., and Bjørner, N. 2008. Z3: An efficient smt solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'08/ETAPS'08*, 337–340. Berlin, Heidelberg: Springer-Verlag.
- Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artif. Intell.* 49(1-3):61–95.
- Dixon, C.; Fisher, M.; and Konev, B. 2007. Temporal logic with capacity constraints. In *Frontiers of Combining Systems, 6th International Symposium, FroCoS 2007, Liverpool, UK, September 10-12, 2007, Proceedings*, 163–177.
- Dorn, J. 1993. Supporting scheduling with temporal logic. In *Proceedings of the IJCAI'93 Workshop on Production Planning, Scheduling and Control, Chambry*, 113–124.
- Fox, M. S. 1983. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.
- Hirshfeld, Y., and Rabinovich, A. M. 2004. Logics for real time: Decidability and complexity. *Fundam. Inform.* 62(1):1–28.
- Karaman, S., and Frazzoli, E. 2008. Vehicle routing problem with metric temporal logic specifications. In *Proceedings of the 47th IEEE Conference on Decision and Control, CDC 2008, December 9-11, 2008, Cancún, México*, 3953–3958.
- Karaman, S.; Sanfelice, R. G.; and Frazzoli, E. 2008. Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In *Proceedings of the 47th IEEE Conference on Decision and Control, CDC 2008, December 9-11, 2008, Cancún, México*, 2117–2122.



- Koymans, R. 1990. Specifying real-time properties with metric temporal logic. *Real-Time Syst.* 2(4):255–299.
- Li, Y.; Albarghouthi, A.; Kincaid, Z.; Gurfinkel, A.; and Chechik, M. 2014. Symbolic optimization with smt solvers. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '14, 607–618. New York, NY, USA: ACM.
- Ostroff, J. S. 1989. *Temporal Logic for Real Time Systems*. New York, NY, USA: John Wiley & Sons, Inc.
- Ouaknine, J., and Worrell, J. 2008. Some recent results in metric temporal logic. In *Formal Modeling and Analysis of Timed Systems, 6th International Conference, FORMATS 2008, Saint Malo, France, September 15-17, 2008. Proceedings*, 1–13.
- Pesant, G. 2004. A regular language membership constraint for finite sequences of variables. In *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004. Proceedings*, 482–495.
- Planken, L. R. 2013. *Algorithms for Simple Temporal Reasoning*. Ph.D. Dissertation, Delft University of Technology.
- Schrijver, A. 1986. *Theory of Linear and Integer Programming*. New York, NY, USA: John Wiley & Sons, Inc.
- Sebastiani, R., and Tomasi, S. 2012. Optimization in smt with  $la(q)$  cost functions. In *Proceedings of the 6th International Joint Conference on Automated Reasoning, IJCAR'12*, 484–498. Berlin, Heidelberg: Springer-Verlag.
- Staab, S. 1998. On non-binary temporal relations. In *In Proc. of ECAI-98*, 567–571. Wiley Chichester UK.
- Stergiou, K., and Koubarakis, M. 1998. Backtracking algorithms for disjunctions of temporal constraints. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98, July 26-30, 1998, Madison, Wisconsin, USA.*, 248–253.
- Tsamardinos, I.; Vidal, T.; and Pollack, M. E. 2003. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints* 8(4):365–388.
- Wolff, E. M.; Topcu, U.; and Murray, R. M. 2014. Optimization-based trajectory generation with linear temporal logic specifications. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, 5319–5325.
- Yu, P., and Williams, B. C. 2013. Continuously relaxing over-constrained conditional temporal problems through generalized conflict learning and resolution. In Rossi, F., ed., *IJCAI. IJ-CAI/AAAI*.