Proceedings, Fifteenth International Conference on
Principles of Knowledge Representation and Reasoning (KR 2016)

Using Metric Temporal Logic to Specify Scheduling Problems

Roy Luo*, Richard Valenzano*, Yi Li*, J. Christopher Beck†, Sheila A. McIlraith*

*Department of Computer Science, University of Toronto, Canada.
†Department of Mechanical and Industrial Engineering, University of Toronto, Canada.
*{royluo,rvalenzano,liyi,sheila}@cs.toronto.edu, †jcb@mie.utoronto.ca

Abstract

We introduce Scheduling MTL (SMTL) an extension of Metric Temporal Logic that supports the specification of complex scheduling problems with repeated and conditional occurrences of activities, and rich temporal relationships among them. We define the syntax and semantics of SMTL, and explore natural restrictions of the language to gain tractability. We also provide an algorithm for finding a schedule to a problem specified as an SMTL formula, and establish a novel equivalence between a fragment of MTL and simple temporal networks, a widely-used formalism in AI temporal planning.

1 Introduction

The need and opportunity for fully automated and mixed-initiative scheduling has exploded as automation plays an increasing role in business practices, and as sophisticated calendaring tools become broadly adopted. The availability of computer-interpretable scheduling constraints and partial schedules enables a broad array of new scheduling problems including sports team and facilities scheduling, car pooling, and other personal and commercial applications. With these new problems come new challenges in the specification of scheduling problems and the verification of their solutions.

Increasingly, there is a desire to express richer problems involving repeated patterns of occurrence of various activities, some conditional, some quantity bounded. For example, in nurse rostering, a typical scheduling constraint might be to *always schedule nurses four days on followed by two days off*. There is also a desire to characterize activities and resources in terms of properties that support implicit groupings of activities, the specification of constraints by referring to unnamed activities and resources, and lifted reasoning about groups rather than named individuals (e.g. *always schedule at least one Mandarin-speaking nurse on each rotation*). While specialized approaches to some such patterns exist (Pesant 2004), no formal language exists to specify the diversity of rich scheduling patterns manifest in modern applications. Instead, in many academic publications, the formal problem definition is itself an optimization model, conflating specification, modeling, and solving.

With notable exceptions such as constraint-based scheduling, a major focus of scheduling research has been on the study of specific scheduling problems, and the development of efficient problem-specific solutions. This focus means that the task of verifying a specification or proposed solution with respect to a set of properties is not easily achieved.

In this paper, we augment Metric Temporal Logic (MTL) (Koymans 1990) to express the complicated patterns in real-world scheduling applications, while also supporting automated reasoning with a view to querying or verifying properties of problem specifications and solutions. We call the resulting language *Scheduling MTL (SMTL)*. To trade off expressiveness for tractability, we explore novel restrictions to SMTL that are conducive to scheduling, and also identify an approach for finding satisfying schedules under these restrictions. We then demonstrate that these restrictions are rich enough to represent shop scheduling and temporal network problems, key problems in the Operations Research (OR) and Artificial Intelligence (AI) communities. As part of this analysis, we prove a novel equivalence between a tractable fragment of SMTL and *simple temporal networks*, a widely-used formalism in AI planning and scheduling.

2 SMTL: A Language for Scheduling

We define a language for specifying a scheduling problem as a logical formula that is satisfiable if and only if the corresponding scheduling problem has a solution.

Definition 2.1 (Scheduling Problem)**.** *A scheduling problem is a tuple $\langle \mathcal{A}, \mathcal{I}, \mathcal{D}, \mathcal{R}, \mathcal{C}, \mathcal{TC} \rangle$ where $\mathcal{A}$ is a set of activities, $\mathcal{I}$ is the set of all activity instances associated with $\mathcal{A}$, $\mathcal{D}$ 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, and $\mathcal{TC}$ is a set of temporal constraints between activities.*

A solution to a scheduling problem, a *schedule*, is an assignment of start and end times to a set of activity instances $\{a_1 \dots a_k\} \subseteq \mathcal{I}$ such that $\mathcal{D}$, $\mathcal{R}$, $\mathcal{C}$ and $\mathcal{TC}$ are all satisfied. Notationally, activities, A, are denoted using upper case while activity instances, a_i, are denoted using lower case.

To specify a scheduling problem as a logical formula, we use a language based on *Metric Temporal Logic (MTL)* — an extension of Linear Temporal Logic (LTL) in which modalities of LTL are augmented with timing constraints.

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).

The MTL operator "until" ($\mathcal{U}$) augmented with a time interval ($\phi\, \mathcal{U}_I \psi$) states that ϕ is true from now until some point within I time units, when ψ becomes true. As in LTL, further modalities such as "always" ($\Box$) and "eventually" ($\Diamond$) can also be defined in terms these operators (cf. Section 2.2).

We now introduce our own augmentation of MTL and follow with an intuitive explanation.

Definition 2.3. *Let $\mathcal{A}$ be a set of constant symbols denoting activities. A* Scheduling MTL (SMTL) formula *is a MTL formula such that the set of atomic predicates consists of propositions of the form $\{\mathsf{start}(A), \mathsf{end}(A), \mathsf{P}(A)\}$ 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 ϕ is in the scope of a quantified variable x, atomic predicates also include $\mathsf{start}(x)$, $\mathsf{end}(x)$, and $\mathsf{P}(x)$ (we omit a formal treatment of the syntax of quantified variables).*

For a repeating activity A, $\mathsf{start}(A)$ and $\mathsf{end}(A)$ correspond to the start and end times of an instance of activity A. Their truth values are dependent on the time of evaluation: $\mathsf{start}(A)$ (resp. $\mathsf{end}(A)$) is true at time t if some instance of A starts (resp. ends) at that time (cf. Section 2.1).

Quantifiers and properties allow for the compact representation of constraints over instances of a repeating activity, or all activities that share some attribute. A property P is defined as the subset of $\mathcal{A}$ for which that property holds, represented by unary predicates, $\mathsf{P}(A)$. For example, in nurse rostering, an activity might correspond to a particular nurse's shifts and a property could be used to characterize whether the activity involves a "trainee" or "experienced" nurse. Using quantifiers, we can easily construct a constraint stating that a new nurse can only be scheduled whenever a more experienced nurse is also on shift. Properties are also useful for specifying resource constraints, as one can represent resource capacity by limiting the number of activities with the corresponding property that can occur at any time.

In addition to serving as an input to a scheduling algorithm, an SMTL specification enables us to reason logically about properties of the problem and to answer queries about the schedules it entails, without necessarily calculating specific solutions. This is a critical and missing tool in scheduling requirements engineering and verification.

2.1 Semantics

We now formally define the introduced notions. A *schedule* $\mathcal{T}$ consists of a tuple $(\mathit{Instances}, T)$ and is the object we use to evaluate an SMTL formula. $\mathit{Instances} : \mathcal{A} \mapsto 2^{\mathcal{I}}$ 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. $T : \{a \mid a \in \{\mathit{Instances}(A) \mid A \in \mathcal{A}\}\} \mapsto \mathbb{D} \times \mathbb{D}$. Let $T_s(a)$ and $T_e(a)$ return the first and second values of $T(a)$ respectively. A formula ϕ is satisfied by a schedule $\mathcal{T}$ at a time $t_i \in \mathbb{D}$, written as $\langle \mathcal{T}, t_i \rangle \models \phi$. Where $\top$ is true and $\bot$ is false, the rest of the operators are defined as follows:

- $\langle \mathcal{T}, t_i \rangle \models \mathsf{start}(A)$ (resp. $\mathsf{end}(A)$) iff $\exists x \in \mathit{Instances}(A)$ such that $T_s(x) = t_i$ (resp. $T_e(x) = t_i$).
- $\langle \mathcal{T}, t_i \rangle \models \mathsf{start}(a)$ (resp. $\mathsf{end}(a)$) iff $T_s(a) = t_i$ (resp. $T_e(a) = t_i$).
- $\langle \mathcal{T}, t_i \rangle \models \mathsf{P}(A)$ iff $A \in P$.
- $\langle \mathcal{T}, t_i \rangle \models \mathsf{P}(a)$ iff for some $A \in \mathcal{A}$, $a \in \mathit{Instances}(A)$ and $A \in P$.
- $\langle \mathcal{T}, t_i \rangle \models \phi\, \mathcal{U}_I \psi$ 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 ϕ with all occurrences of x replaced with a.

$\neg$, $\wedge$, and other connectives follow a standard interpretation.

Definition 2.4. *Given a scheduling problem $S = \langle \mathcal{A}, \mathcal{I}, \mathcal{D}, \mathcal{R}, \mathcal{C}, \mathcal{TC} \rangle$, where $\mathcal{D}$, $\mathcal{R}$, $\mathcal{C}$, and $\mathcal{TC}$ are expressed as SMTL formulae, then $\mathcal{T}$ is a schedule for S iff $\mathcal{T} \models \phi$ for all $\phi \in \mathcal{D} \cup \mathcal{R} \cup \mathcal{C} \cup \mathcal{TC}$, i.e., $\langle \mathcal{T}, 0 \rangle \models \phi$, where 0 is the initial time point.*

Though expressed in a different manner, our definition of a schedule is semantically equivalent to the signal function used to evaluate formulae in the continuous (as opposed to pointwise) semantics of MTL (Ouaknine and Worrell 2008).

2.2 Further Syntax

To facilitate the encoding of scheduling problems, we also define the following syntax in terms of the operators above.

Logical Operators: $\exists$, $\vee$, $\supset$, and $\equiv$ are defined as usual.
Modal Operators: The metric-valued "eventually" operator $\Diamond_I \phi$ is defined as $\top\, \mathcal{U}_I \phi$, and the metric valued "always" operator $\Box_I \phi$ is its dual $\neg\Diamond_I \neg\phi$. We define $\Diamond\phi$ and $\Box\phi$ without the subscript to represent the LTL-style unconstrained definitions, $\Diamond_{[0,\infty)}\phi$ and $\Box_{[0,\infty)}\phi$, and the past-LTL operator "once" as $\Diamond_{(-\infty,0]}$.
Time Windows and Durations: Let $\phi \rightarrow_I \psi$ be equivalent to $\Diamond_{(-\infty,\infty)}(\phi \wedge \Diamond_I \psi)$. This is semantically analogous to the following 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, ψ is true at some time within interval I after ϕ is true. This operator is useful for defining time windows and activity durations. For example, we can denote that activity A has duration 7 with $\mathsf{start}(A) \rightarrow_{[7,7]} \mathsf{end}(A)$.
Temporal Properties: *Before*(ϕ), *After*(ϕ), and *Between*(ϕ, ψ) denote $\Diamond_{(0,\infty)}\phi$, $\Diamond_{(-\infty,0)}\phi$, and *After*$(\phi)$ $\wedge$ *Before*(ψ), respectively. *Currently*(x), encoded as *Between*$(\mathsf{start}(x), \mathsf{end}(x))$ denotes that instance x is currently happening.

For example, we can denote that activity A must complete before activity B with the formula $\Box(\mathsf{start}(B) \supset$ *After*$(\mathsf{end}(A))$. Alternately, we can denote that A and B do not overlap in their occurrence with the formula $\Box(\neg(\mathit{Currently}(A) \wedge \mathit{Currently}(B)))$.
Activity Instances: For every activity A, let property $\mathsf{P}_A = \{A\}$, so only instances of A share this property. Let predicate *InstanceOf*$(x, A) \equiv \mathsf{P}_A(x)$, so that *InstanceOf*$(x, A)$ is true iff x is an instance of A.

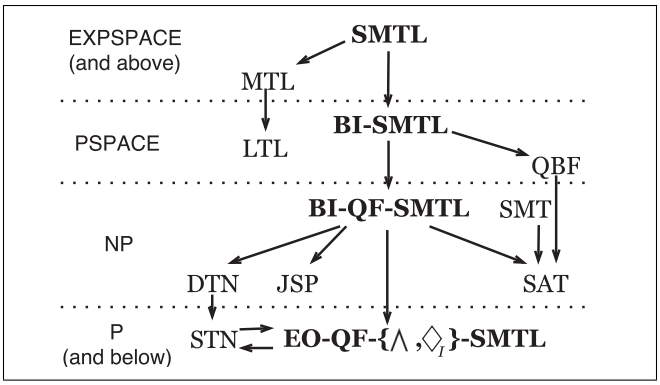

Figure 1: Complexity classes for satisfiability of SMTL fragments and related problems, in the spirit of Planken (2013). Arrows represent polytime reductions.

Notice that SMTL supports the compact encoding of scheduling constraints, while specifying durations and time windows in LTL is impossible for timing that ranges over the real numbers and cumbersome for discrete time.

3 Tractable Fragments of SMTL

We study the complexity of SMTL and more tractable fragments of the language, demonstrating that these fragments are sufficiently expressive to represent important problems in OR and AI scheduling. Finally, we describe an algorithm for solving a fragment of SMTL that is PSPACE-complete.

To measure the complexity of SMTL, we consider the difficulty of solving the following problem:

Definition 3.1. *The* SMTL satisfiability problem *is the task of taking a SMTL formula ϕ and finding a schedule $\mathcal{T}$ such that $\mathcal{T} \models \phi$.*

As an extension of propositional MTL, SMTL satisfiability is at least as hard (i.e., EXPSPACE for integer values (Ouaknine and Worrell 2008) and undecidable for real values (Alur, Feder, and Henzinger 1996)). Figure 1 summarizes our complexity results for the SMTL restrictions that we define, situating them in the context of other logics and scheduling problems discussed below. An arrow from A to B indicates that problem B is polytime reducible to problem A. The vertical arrows denote reductions that are trivial or were previously known. The full proofs of the new results are given in a technical report (Luo et al. 2015).

In our first restriction, we consider problems which have a bound on the number of times an activity can occur:

Definition 3.2. Bounded-instance SMTL*, or BI-SMTL, is SMTL 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 $k_A \in \mathbb{N}$.*

While the bounds mean that it is no longer possible to specify infinite schedules, they are natural in many applications in which there is a finite number of events, like timetabling, shop scheduling, and project scheduling. To evaluate the complexity of this fragment, we consider BI-SMTL satisfiability which can be shown to be PSPACE-hard by reducing True Quantified Boolean Formulae (QBF) to it. As we can also define a satisfiability algorithm that requires polynomial space, the following is true:

Theorem 3.1. *BI-SMTL satisfiability is PSPACE-complete.*

Let us now consider the following fragment, which is given by further removing quantifiers.

Definition 3.3. *Bounded-instance quantifier-free SMTL, or BI-QF-SMTL, is the set of BI-SMTL formulae in which no subformula has the form $\exists x.\phi$ or $\forall x.\phi$.*

BI-QF-SMTL satisfiability can be shown to be NP-hard by reducing standard boolean satisfiability (SAT) to it. As we can also describe a polynomial time verification algorithm for BI-QF-SMTL, the following is true:

Theorem 3.2. *BI-QF-SMTL satisfiability is NP-complete.*

We also define an *exactly-once quantifier-free formula (EO-QF-SMTL)*, as a BI-QF-SMTL formula in which all activities have a =1 bound. While EO-QF-SMTL satisfiability is also NP-complete, we introduce this new fragment for the convenience of the discussion below.

3.1 Representing Existing Problems in SMTL

Here, we demonstrate that the above fragments of SMTL are sufficient to express classical scheduling problems.

Job Shop Scheduling. *Job shop scheduling problems (JSP)* are among the most commonly studied scheduling problems in OR. In JSP, we consider n jobs, which are to be processed on m machines. Each job requires a machine- and job-dependent processing time, and all jobs must be processed by the machines in a specified job-specific order. A JSP can be represented as an EO-QF-SMTL formula by having one activity with a =1 bound for the processing of each job on each machine. The formula is then given by the conjunction of constraints representing processing durations, pair-wise constraints ensuring no two jobs are processed simultaneously by the same machine, and precedence constraints on the order in which the jobs are processed by each machine. Additional JSP variations can also be easily represented as EO-QF-SMTL formulae, including those where the jobs have release times or deadlines, or in which machines have some minimum idle time between jobs.

Temporal Networks. Temporal networks are the foundational representation used in scheduling and temporal planning in AI. For example, *simple temporal networks (STNs)* consist of a set of *simple temporal constraints*, each of the form $[l, u]_{xy}$, where x, y are times of events and l, u are constant bounds. These constraints represent the inequality $l \leq y - x \leq u$. A *simple temporal problem (STP)* requires assigning times to the events such that they satisfy all the constraints of a given STN (Dechter, Meiri, and Pearl 1991).

While SMTL is a timed temporal logic, its semantics still allow it to model temporal network problems simply and directly. To see this, consider the set of EO-QF-SMTL formulae that only allow for the use of the $\wedge$ and $\Diamond_I$ operators. We refer to this fragment as EO-QF-$\{\wedge, \Diamond_I\}$-SMTL, for which we have shown the following:

Theorem 3.3. *EO-QF-$\{\wedge, \Diamond_I\}$-SMTL formulae and STNs are equivalently expressive.*

It is remarkable that the simple syntactic restriction of EO-QF-SMTL results in a temporal logic expressible as a temporal network. Alur, Feder, and Henzinger (1996) established the relationship between the MTL timed-word model and timed automata, but the continuous semantics of logics such as the one used here were not previously known to be expressible with such a graph-based model. This may shed further light on the connection between timed automata and temporal networks (Cimatti et al. 2014).

Given that STPs can be solved in polynomial time (Planken 2013), and the translation from EO-QF-$\{\wedge, \Diamond_I\}$-SMTL to an STN is linear, we can show the following:

Theorem 3.4. *EO-QF-$\{\wedge, \Diamond_I\}$-SMTL satisfiability is in P.*

We have also shown (cf. (Luo et al. 2015)) that disjunctive temporal networks (DTNs) (Stergiou and Koubarakis 2000) and generalized temporal networks (Staab 1998) can be modelled in EO-QF-SMTL, and that conditional temporal networks (Tsamardinos, Vidal, and Pollack 2003) can be expressed in BI-QF-SMTL. The benefits of SMTL are clear: while entirely new formulations were developed to handle these variations, even the BI-QF-SMTL fragment can specify disjunctions and conditions of arbitrary subformulae.

3.2 Solving BI-SMTL Formulae

We now describe a general method for finding a schedule that satisfies a given BI-SMTL formula. First, the formula is translated to an EO-QF-SMTL formula by expanding out the quantifiers and replacing the bounded activities with a set of =1 activities that preserve the semantics of the original formula. The result is then translated to first-order logic, using a similar algorithm to that used by Hirshfeld and Rabinovich (2004) to translate from propositional MTL to first-order logic. Finally, a Satisfiability Modulo Theory (SMT) solver is used to find start and end times for each activity.

To demonstrate the potential of the restricted SMTL languages for specifying scheduling problems, we built a proof-of-concept system that implements the method above. While our system cannot compete with state-of-the-art problem-specific encodings and solvers, it can effectively solve problems from multiple domains. In contrast, the inherent specialization of some state-of-the-art systems means that even small changes to the problem may require major changes to the encoding, which may make the specialized solving techniques ineffective.

4 Concluding Remarks

This paper introduces SMTL, an extension of MTL tailored to the specification of scheduling problems. The value and significance of SMTL lies in its ability to help specify, understand, compare, and solve a myriad of scheduling problems with varying complex constraints that are one-off and may not have crafted solutions in the literature. By using a logic-based language, we also afford the possibility for domain-independent querying and verification. In (Luo et al. 2015) we describe an encoding of SMTL in a SMT solver and our experimental findings.

Few attempts have been made to apply temporal logic to scheduling with Dorn (1993) and later Karaman, Sanfelice, and Frazzoli (2008) being some notable exceptions. LTL on finite traces (e.g., (Baier and McIlraith 2006; De Giacomo and Vardi 2013; Fionda and Greco 2016)) is of increasing interest in AI. By bounding the number of instances of activities, our bounded instance restriction to SMTL indirectly achieves finite traces. We provide all proofs and an extensive discussion of related work in (Luo et al. 2015).

Acknowledgements: The authors gratefully acknowledge funding from NSERC.

References

Alur, R.; Feder, T.; and Henzinger, T. A. 1996. The benefits of relaxing punctuality. *Journal of the ACM* 43(1):116–146.

Baier, J., and McIlraith, S. 2006. Planning with first-order temporally extended goals using heuristic search. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence*, 788–795.

Cimatti, A.; Hunsberger, L.; Micheli, A.; and Roveri, M. 2014. Using timed game automata to synthesize execution strategies for simple temporal networks with uncertainty. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2242–2249.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 854–860.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49(1-3):61–95.

Dorn, J. 1993. Supporting scheduling with temporal logic. In *Proceedings of the IJCAI'93 Workshop on Production Planning, Scheduling and Control*, 113–124.

Fionda, V., and Greco, G. 2016. The complexity of LTL on finite traces: Hard and easy fragments. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Hirshfeld, Y., and Rabinovich, A. 2004. Logics for real time: Decidability and complexity. *Fundamenta Informaticae* 62(1):1–28.

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*, 2117–2122.

Koymans, R. 1990. Specifying real-time properties with metric temporal logic. *Real-Time Systems* 2(4):255–299.

Luo, R.; Valenzano, R.; Li, Y.; Beck, J. C.; and McIlraith, S. A. 2015. A timed temporal logic for specifying scheduling problems (extended report). Technical Report CSRG-629, Department of Computer Science, University of Toronto, Toronto, Canada.

Ouaknine, J., and Worrell, J. 2008. Some recent results in metric temporal logic. In *Formal Modeling and Analysis of Timed Systems, FORMATS 2008*, 1–13.

Pesant, G. 2004. A regular language membership constraint for finite sequences of variables. In *Principles and Practice of Constraint Programming*, 482–495.

Planken, L. R. 2013. *Algorithms for Simple Temporal Reasoning*. Ph.D. Dissertation, Delft University of Technology.

Staab, S. 1998. On non-binary temporal relations. In *Proc. of the 13th European Conference on Artificial Intelligence*, 567–571.

Stergiou, K., and Koubarakis, M. 2000. Backtracking algorithms for disjunctions of temporal constraints. *Artificial Intelligence* 120(1):81–117.

Tsamardinos, I.; Vidal, T.; and Pollack, M. E. 2003. CTP: A new constraint-based formalism for conditional, temporal planning. *Constraints* 8(4):365–388.