Topics in Monitoring and Planning for Embedded Real-Time Systems

Hsi-Ming Ho
St Anne's College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Hilary 2015

Acknowledgements

First of all, let me thank my supervisor Joël Ouaknine. Without his great help, guidance and support in all aspects of my graduate life, this thesis would not be possible.

I am also grateful to James Worrell who helped me tremendously in revising manuscripts and provided many invaluable insights.

I thank Daniel Kroening and Tom Melham for their useful suggestions on early drafts of this thesis.

I thank Thomas Brihaye and Stefan Kiefer for their time and effort in helping me to improve this thesis.

I am grateful to Thomas Gibson-Robinson and Bill Roscoe (for their help on CSP and FDR3), Paul Hunter (for his patience in answering my questions) and Ofer Strichman (for introducing me to the CR-UAV Problem and many fruitful discussions).

I thank my colleagues Daniel Bundala and Björn Wachter for their encouragement throughout the years. Also, I am indebited to Ventsi Chonev for helping me with a conference presentation.

Finally, my doctorate study was financially supported by Clarendon Fund.

Abstract

The verification of real-time systems has gained much interest in the formal verification community during the past two decades. In this thesis, we investigate two real-time verification problems that benefit from the techniques normally used in untimed verification.

The first part of this thesis is concerned with the monitoring of real-time specifications. We study the expressiveness of metric temporal logics over timed words, a problem that dates back to early 1990s. We show that the logic obtained by extending *Metric Temporal Logic* (MTL) with two families of new modalities is expressively complete for the *Monadic First-Order Logic of Order and Metric* (FO[$<, +1$]) in time-bounded settings. Furthermore, by allowing rational constants, expressive completeness also holds in the general (time-unbounded) setting. Finally, we incorporate several notions and techniques from LTL monitoring to obtain the first trace-length independent monitoring procedure for this logic.

The second part of this thesis concerns a decision problem regarding UAVs: given a set of targets (each ascribed with a relative deadline) and flight times between each pair of targets, is there a way to coordinate a flock of k identical UAVs so that all targets are visited infinitely often and no target is ever left unvisited for a time longer than its relative deadline? We show that the problem is PSPACE-complete even in the single-UAV case, thereby corrects an erroneous claim from the literature. We then complement this result by proposing an efficient antichain-based approach where a delayed simulation is used to prune the state space. Experimental results clearly demonstrate the effectiveness of our approach.

Contents

Chapter 1

Introduction

1.1 Background

Ever since their emergence in the last century, computers have played an important role in the human civilisation. Apart from Desktop PCs and laptops, an even more common use of computer systems is to act as integral parts of larger mechanical or electrical systems. In this case, they are more aptly referred to as *embedded systems*. In today's world they are practically everywhere: from avionic systems to smartphones in everyone's hands.

Currently, the design phase of most embedded systems still requires a high degree of human intervention.[1] For this reason, they are prone to contain errors, or *bugs* as they are often called in technical contexts. Intuitively, a bug is a flaw in the system that causes inconsistencies between the actual system behaviour and the desired behaviour described in the specification. In some applications, e.g., control systems in washing machines, bugs can only cause some inconvenience or annoyance and may possibly be tolerated if they do not happen too often. In certain other scenarios, however, a simple bug may lead to catastrophic consequences. Some notable examples are Intel's $475 million recall of Pentium processors for a circuit-design bug, the explosion of the Ariane 5 rocket caused by an error in floating-point conversion, and more tragically, the death of six patients due to a race condition in Therac-25.

For very simple and insignificant systems, e.g., a programming assignment in a first-year undergraduate course, manual debugging may be considered as a viable option to find errors. However, the level of sophistication of virtually any system in real-world applications can easily render manual debugging infeasible. For such

[1] An approach aimed at overcoming this limitation is *synthesis*, i.e., build systems automatically from specifications.

systems, and especially those are safety-critical, we definitely need a better alternative. Unfortunately, Rice's theorem [Ric53] already tells us that it is mathematically impossible in general to decide *automatically* whether a given system satisfies a given specification. Still, it is possible to first abstract the system under scrutiny into a *model* specified in a weaker formalism (e.g., an automaton) and check automatically whether the model satisfies the given specification. The latter part of this process is called *model checking* [CE81, QS82], a core technique in the field of *formal verification*.

Similar to the case of manual debugging, a major challenge in model checking is the so-called state-space explosion problem: if the number of states in the model gets too large, running verification algorithms on the model may not be feasible. In the past 25 years, extensive research efforts have been devoted to alleviate this problem. For example, in *symbolic model checking* [McM93] one represents a set of states as a logic formula rather than an explicit list of states, thus reducing the memory needed in storing and manipulating them. A more recent proposal is the *antichain approach* [WDHR06] where one compactly represents a set of states by its 'minimal representatives' in a partial order. There are also techniques that exploit special structures of state spaces, such as *symmetry reduction* [ES96] and *partial-order reduction* [HP94]. Owing to these and other related developments, model checking is now widely used in industry, especially in hardware design. At this point, it is fair to say that model checking is one of the greatest success stories of computer science. For this achievement, Clarke, Emerson and Sifakis received the prestigious ACM Turing Award in 2007.

Traditionally, specifications used in verification are purely *qualitative*, e.g., one may require the system to satisfy the property "every request is eventually followed by an acknowledgement". In many applications, however, it would be preferable to be able to talk about the timing aspects of the system. For example, one may want to check whether the system satisfies "every request is followed by an acknowledgement *in less than 3 seconds*". Verification in this *quantitative* setting[2] was first considered by Alur *et al.* in the early 1990s. In particular, a real-time formalism called *timed automata* [AD94] was proposed. Timed automata extend finite automata by real-valued *clocks* and allow *clock constraints* to be specified on edges. It is now widely accepted as the standard formalism for modelling real-time systems.

A foundational result in real-time verification is that the emptiness problem for timed automata (is there an accepting run in the given timed automaton?) is decidable.

[2]Other quantitative information have been considered in verification as well, e.g., probabilities, rates and energy consumption.

However, since timed automata are not closed under complementation, a necessary step in most model-checking algorithms, they are not suitable as specifications by themselves. A natural alternative is to extend the existing (untimed) logical formalisms with timing constraints and use them as specification formalisms. This strategy worked very well for some formalisms; for example, *Timed Computation Tree Logic* (TCTL) [ACD90] enjoys an efficient (PSPACE) model-checking algorithm and excellent tool support. Unfortunately, adding timing constraints to *linear-time* formalisms[3] can make their model-checking problems undecidable. In particular, one can encode the halting problem of a given Turing machine [Tur36] into the satisfiability problem (is there a behaviour satisfying the formula?) of a formula in this case. For this reason, it is necessary to impose certain syntactic or semantic restrictions on either models or specifications to make the verification even possible. We mention some of these approaches here:

- Limit the expressive powers of specification languages, thus the aforementioned encoding of the halting problem for Turing machines becomes impossible (cf., e.g., [AFH96, RS97, OW06b, BMOW07]).

- Under certain conditions, it suffices to consider only the discrete-time behaviours of models, i.e., assume time advances in 'ticks' and thus all actions happen at integer time points. Existing algorithms and tools can then be used without much difficulty (cf., e.g., [HMP92, GPV94, BMT99, OW03, KP05, CLT07]).

- Consider only time domains of finite length; for example, one may check whether the model satisfies the specification *in the first 20 seconds* (cf., e.g., [ORW09, OW10]). This idea is useful in dealing with other modelling formalisms as well; for example, new decidability results have been obtained for hybrid automata in time-bounded settings [BDG+11].

In recent years, there also has been interest in other verification techniques. For example, *runtime verification* has emerged as a complement to model checking (see [LS09, SHL11] for surveys). Roughly speaking, while in model checking one considers *all* behaviours of the model, in runtime verification one focusses on *one* particular behaviour. Such techniques are particularly useful in situations in which we wish to evaluate a system that is either too complex to model or whose internal details are not accessible.

[3] TCTL, on the other hand, is a *branching-time* formalism. For a thorough comparison and discussion of the two paradigms, see [Var01].

1.2 Scope and Contribution of this Thesis

1.2.1 Expressiveness of Metric Temporal Logics

Context. One of the most prominent specification formalisms used in verification is *Linear Temporal Logic* (LTL), which is typically interpreted over the non-negative integers or reals. A celebrated result of Kamp [Kam68] states that, in either case, LTL has precisely the same expressive power as the *Monadic First-Order Logic of Order* (FO[<]). These logics, however, are inadequate to express specifications for systems whose correct behaviour depends on quantitative timing requirements. Over the last three decades, much work has therefore gone into lifting classical verification formalisms and results to the real-time setting. *Metric Temporal Logic* (MTL),[4] which extends LTL by constraining the modalities by time intervals, was introduced by Koymans [Koy90] in 1990 and has emerged as a central real-time specification formalism.

MTL enjoys two main semantics, depending intuitively on whether atomic formulas are interpreted as *state predicates* or as (instantaneous) *events*. In the former, the system is assumed to be under observation at every instant in time, leading to a 'continuous' semantics based on *flows* (or *signals*), whereas in the latter, observations of the system are taken to be (finite or infinite) sequences of timestamped snapshots, leading to a 'pointwise' semantics based on *timed words*. Timed words are the leading interpretation, for example, for systems modelled as timed automata [AD94]. In both cases, the time domain is usually taken to be the non-negative real numbers. Both semantics have been extensively studied; see, e.g., [OW08] for a historical account.

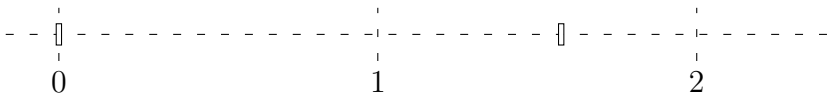


Figure 1.1: The formula $\Diamond_{(0,1)}(\Diamond_{=1}\mathbf{true})$ holds at time 0 in the continuous semantics, but not in the pointwise semantics.

Alongside these developments, researchers proposed the *Monadic First-Order Logic of Order and Metric* (FO[$<, +1$]) as a natural quantitative extension of FO[$<$]. Like MTL, FO[$<, +1$] can be interpreted over signals [HR04] or timed words [Wil94]. An obvious question to ask is whether MTL has the same expressive power as FO[$<, +1$], i.e., an analogue of Kamp's theorem holds in the real-time setting. Unfortunately,

[4] In this thesis, we refer to the standard logic with constrained 'Until' and 'Since' modalities exclusively as 'MTL' and use the term 'metric temporal logic' in a broader sense to refer to a temporal logic with modalities definable in FO[$<, +1$].

Hirshfeld and Rabinovich [HR07] showed that no 'finitary' extension of MTL—and *a fortiori* MTL itself—could have the same expressive power as FO[$<, +1$] over the reals.[5] Still, in the continuous semantics, MTL can be made expressively complete for FO[$<, +1$] by extending the logic with an infinite family of '*counting modalities*' [Hun13] or considering only *bounded* time domains [ORW09, OW10]. Nonetheless, and rather surprisingly, MTL with counting modalities remains strictly less expressive than FO[$<, +1$] over bounded time domains in the pointwise semantics, i.e., over timed words of bounded duration, as we will see in Section 3.2.

Contributions. We study the expressiveness of various fragments and extensions of MTL, with the goal of achieving expressive completeness in the pointwise semantics, in Chapters 3 and 4. In particular, we identify a fundamental deficiency in expressiveness in the pointwise interpretation of MTL. To amend this, we propose new (first-order definable) modalities 'generalised Until' ($\mathfrak{U}$) and 'generalised Since' ($\mathfrak{S}$). With the help of these new modalities and the techniques developed in [PS11, ORW09, HOW13], we establish the following results in the pointwise semantics:

(i). There is a strict hierarchy of metric temporal logics based on their expressiveness over bounded timed words.

(ii). The metric temporal logic with the new modalities $\mathfrak{U}$ and $\mathfrak{S}$ (denoted MTL[$\mathfrak{U}, \mathfrak{S}$]) is expressively complete for FO[$<, +1$] over bounded timed words.

(iii). The time-bounded satisfiability and model-checking problems for MTL[$\mathfrak{U}, \mathfrak{S}$] are EXPSPACE-complete, the same as that of MTL.

(iv). Any MTL[$\mathfrak{U}, \mathfrak{S}$] formula is equivalent to a syntactically separated one.

(v). MTL[$\mathfrak{U}, \mathfrak{S}$] is expressively complete for FO[$<, +\mathbb{Q}$] (the rational variant of FO[$<, +1$]) over unbounded (i.e., infinite non-Zeno) timed words if we allow the use of rational endpoints in specifying constraining intervals of the modalities.

Related work. Bouyer, Chevalier and Markey [BCM05] showed that $\mathsf{MTL}_{\mathsf{fut}}$ (the future-only fragment of MTL) is strictly less expressive than MTL in both the

[5] Hirshfeld and Rabinovich's result was only stated and proved for the continuous semantics, but we believe that their approach would also carry through for the pointwise semantics. In any case, using different techniques Prabhakar and D'Souza [PD06] and Pandya and Shah [PS11] independently showed that MTL is strictly weaker than FO[$<, +1$] in the pointwise semantics.

continuous and pointwise semantics. This, together with the aforementioned results [HR07, PS11], form a strict hierarchy of expressiveness that holds in both the continuous and pointwise semantics:

$$\mathsf{MTL}_{\mathsf{fut}} \subsetneq \mathsf{MTL} \subsetneq \mathsf{FO}[<, +1]\,.$$

Ouaknine, Rabinovich and Worrell [ORW09] showed that the hierarchy collapses in the continuous semantics when one considers bounded time domains of the form $[0, N)$. Our results in Chapter 3 show that this is not the case in the pointwise semantics.

Another possible way to obtain expressive completeness is by allowing rational endpoints. In this setting, counting modalities can readily be expressed in MTL in both semantics [BCM05]. Exploiting this observation, Hunter, Ouaknine and Worrell [HOW13] showed that MTL with rational endpoints is expressively complete for $\mathsf{FO}[<, +\mathbb{Q}]$ (the rational version of $\mathsf{FO}[<, +1]$) in the continuous semantics. However, as can be immediately derived from a result of Prabhakar and D'Souza [PD06], this pleasant result does not hold in the pointwise semantics. On the other hand, D'Souza and Tabareau [DT04] showed that MTL with rational endpoints is expressively complete for $\mathsf{rec\text{-}TFO}[\Diamond, \Diamond\!\!\!-]$ (an 'input-determined' fragment of $\mathsf{FO}[<, +\mathbb{Q}]$) in the pointwise semantics. In Chapter 4, we complement these results by extending MTL with the new modalities introduced in Chapter 3 (with rational endpoints) to make it expressively complete for $\mathsf{FO}[<, +\mathbb{Q}]$ in the pointwise semantics.

1.2.2 Monitoring of Real-Time Specifications

Context. At the very heart of runtime verification is the *monitoring* problem. Given a specification φ (e.g., an MTL formula) and a finite trace ρ (e.g., a finite timed word), the *prefix* problem asks whether all infinite traces extending ρ satisfy φ. The monitoring problem can be seen as an *online* version of the prefix problem where ρ *grows* incrementally (e.g., one event at a time). A monitoring procedure is meant to be executed in parallel with the system under scrutiny and is required to output an answer when either (i) all infinite extensions of the current trace satisfy the specification, or (ii) no infinite extension of the current trace can possibly meet the specification.

Ideally, we would also like to require a monitoring procedure to be *trace-length independent* [Roş12, BKV13] in the sense that the total space requirement should not depend on the length of the input trace (this is important since input traces in practical applications tend to be very long; cf., e.g., [BCE+14]). In the untimed case this is

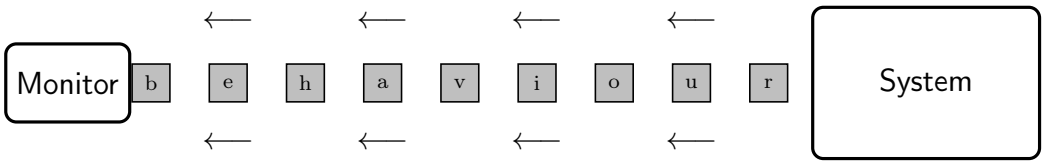


Figure 1.2: A monitor receives the trace in an incremental fashion.

easily achievable: one can translate LTL formulas into Büchi automata [GO03, DKL10] and then turn them into efficient trace-length independent monitors [ABLS05].[6] Unfortunately, a number of obstacles hinder the applicability of the approach above in the real-time setting: it is known that MTL is expressively incomparable with timed automata; even though certain fragments of MTL can be translated into timed automata, the latter are not always determinisable as required for the purpose of monitoring. For this reason, researchers purposed automata-free monitoring procedures that work directly with metric temporal logic formulas (e.g., [MN04,BKZ11]). However, it proved difficult to maintain trace-length independence while being able to handle MTL in its full generality, i.e., with unbounded intervals and nesting of future and past modalities. Almost all monitoring procedures for metric temporal logics in the literature have certain syntactic or semantic limitations, e.g., only allowing bounded future modalities[7] or assuming integer time. A notable exception is [BN12] which handles the full logic MTL over flows, but which unfortunately fails to be trace-length independent.

Contributions. We study a restricted version of the monitoring problem of MTL[$\mathfrak{U}, \mathfrak{S}$], based on the notion of *informative prefixes* [KV01], in Chapter 5. The main idea of our approach is to work with MTL[$\mathfrak{U}, \mathfrak{S}$] formulas of a special form: LTL formulas over atoms comprising bounded MTL[$\mathfrak{U}, \mathfrak{S}$] formulas. By the syntactic separation result in Chapter 4, no expressiveness is sacrificed in restricting ourselves to this fragment. The truth values of bounded MTL[$\mathfrak{U}, \mathfrak{S}$] formulas can be computed and stored efficiently through dynamic programming techniques. The remaining untimed component is then handled via translation to deterministic finite automata. As a result, we obtain the first trace-length independent monitoring procedure for a metric logic that is at least as expressive as the full MTL. The procedure is free of dynamic memory allocations, linked lists, etc., and hence can be implemented efficiently (the *amortised* running time

[6] This methodology applies to any Büchi automaton.

[7] More precisely, they can only monitor formulas of the form $\Box\psi$ where ψ is not allowed to have unbounded future modalities.

per event is linear in the number of subformulas in all bounded formulas). Moreover, we show that our approach can also handle arbitrary $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formulas via syntactic rewriting. To be more precise:

(i). We give a trace-length independent monitoring procedure (which detects informative good/bad prefixes) for LTL formulas over atoms comprising bounded $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formulas.

(ii). For an arbitrary $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula, we show that its informative good/bad prefixes are preserved by the syntactic rewriting rules in Chapter 4 (and thus can be monitored in a trace-length independent fashion).

Related work. In a pioneering work, Thati and Roşu [TR05] propose a rewriting-based monitoring procedure for MTL over discrete-time traces. Their procedure is trace-length independent and amenable to efficient implementations. However, as we will see in Section 5.2, trace-length independent monitoring is not possible in dense-time settings—a monitor would have to 'remember' an infinite number of timestamps. For this reason, researchers often impose a *bounded-variability* assumption, i.e., only a bounded number of events may occur in any time unit. Under such an assumption, Nickovic and Piterman [NP10] showed that $\mathsf{MTL}_{\mathsf{fut}}$ formulas can be translated into deterministic timed automata. Unfortunately, their approach does not extend to MTL as past modalities necessitate the use of *two-way timed automata*; while the two-wayness of such automata can be removed [AH92], the subset construction used in [NP10] no longer applies to the resulting (one-way) automata.

It is known that the non-punctual fragment of MTL, called MITL, can be translated into timed automata. Since the standard constructions [AH92, AFH96] are notoriously complicated, there have been some proposals for simplified or improved constructions [MNP06, KKP11, DM13, BEG14]. The difficulty in using these constructions for monitoring lies in the fact that timed automata cannot be determinised in general. In principle one can carry out on-the-fly determinisation for timed words of bounded variability (cf., e.g., [Tri02, BBBB09]); however, it is not clear that this approach can yield an efficient procedure.

Online monitoring of real-time properties is a very active topic of research. Recently, there have been some attempts to extend temporal logics with (restricted) first-order quantifiers for monitoring (see, e.g., [BKMP08, BKV13, CS13, GT14, dMPPPP14]). Our work can be seen as orthogonal to these advances.

1.2.3 The Cyclic-Routing UAV Problem

Context. Unmanned aerial vehicles (UAVs) have many uses, ranging from civilian to military operations. Like other autonomous systems, they are particularly well-suited to 'dull, dirty, and/or dangerous' missions [UAV]. A common scenario in such missions is that a set of targets have to be visited by a limited number of UAVs. This has given rise to a large body of research on *path planning* for UAVs.[8] Depending on the specific application at hand, paths of UAVs may be subject to various complex constraints, e.g., related to kinematics or fuel (see, e.g., [AKH03, SR12, YK02, RH02]).

In the second part of this thesis, we consider the *Cyclic-Routing UAV* (CR-UAV) *Problem* [DPS10]: the decision version of a simple *recurrent* UAV path-planning problem in which each target must be visited not only once but repeatedly, i.e., at intervals of prescribed maximal duration ('*relative deadline*').

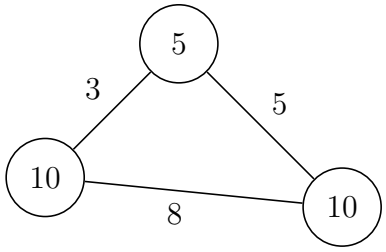


Figure 1.3: Targets in an example instance of the CR-UAV Problem. The number on each vertex denotes the relative deadline of the target; the number on each edge denotes the time required for a UAV to travel between the two targets ('*flight time*').

The CR-UAV Problem is trivially NP-hard as it generalises the decision version of the *Travelling Salesman Problem* (TSP). It is also easy to see that the CR-UAV Problem lies in PSPACE since it can be encoded straightforwardly as the existence of infinite paths in a network of timed automata. The exact complexity of the problem, however, remained unsettled before our work. A special case of the problem (with a single UAV) is considered in [BGA09, BGA12, LFHKG13] and claimed to be NP-complete in [BGA12]. However, the proof of NP-membership in [BGA12] is not detailed. In particular, a crucial claim in [BGA09, BGA12] (which essentially asserts that it suffices to consider infinite paths of 'periods' bounded by the magnitude of the largest relative deadline) is incorrect.

Contributions. We study the complexity of the CR-UAV Problem in Chapter 6. As we will see, by setting the timing constraints carefully, one can devise problem

[8] `http://scholar.google.com/` lists thousands of papers on the subject.

instances that admit only infinite paths of a very constrained form, namely an infinite concatenation of similar subpaths. We use this idea to show that the period of a feasible infinite path (i.e., a *solution*) can be exponential in the magnitude of the largest relative deadline, refuting the claims from [BGA09,BGA12]. We then show that the problem is indeed PSPACE-hard by reducing from the PERIODIC SAT Problem, known to be PSPACE-complete [Orl81]. The proof is quite technical yet conceptually simple: we encode a polynomial amount of information in each subpath and ensure that the information contained in adjacent subpaths are related, effectively forcing any algorithm to keep track of a polynomial amount of information.

The task of solving the CR-UAV Problem in practice is considered in Chapter 7. We first note that, by a standard digitisation argument [HMP92], the CR-UAV Problem can be modelled as the emptiness problem for Büchi automata; moreover, a natural *direct* simulation relation can be defined on the state space of these automata. This suggests that we can apply the well-established *antichain approach* [DR10]. The standard antichain algorithm, however, turned out to be too slow in our case. We therefore adopt a portfolio approach: use a constraint solver for finding solutions and an antichain semi-algorithm for proving that no solution exists. An advantage of this approach is that we can use a coarser *delayed* simulation relation to prune the state space even further. Experimental results demonstrate that our prototype implementation outperforms state-of-the-art model checkers. In summary, we show that:

(i). The CR-UAV Problem is PSPACE-complete (even in the single-UAV case).[9]

(ii). The antichain approach can be applied to solve the CR-UAV Problem efficiently.

Related work. Problems similar to the CR-UAV Problem have long been considered in many other fields such as transportation [Orl82,Wol90] and robotics [CVDK97,KL97]. More recently, a number of game-theoretic frameworks have been developed to study similar problems in the context of security [TKO$^+$09, JKK$^+$10, BGA12]. These works are mostly focused on modelling techniques or approximation algorithms. We, on the other hand, focus on a very simple decision problem.

Orlin [Orl81] characterised the class PSPACE as languages with 'periodic certificates'. In particular, the periodic versions of many NP-complete problems can be shown to be PSPACE-complete. Marathe *et al.* [MHSR95] showed that PSPACE-hardness extends to other kinds of *succinctly* specified problems as well. We note that

[9]Our result holds irrespective of whether the numbers are encoded in unary or binary.

the CR-UAV Problem is not specified in any of these senses. Its PSPACE-hardness, therefore, does not follow from these results.

Antichain techniques have been used very successfully in dealing with PSPACE-complete problems, e.g., universality of non-deterministic finite automata [WDHR06], QBF [BBD+11], and multiprocessor scheduling of real-time tasks [GGL13]. The only work we are aware of that uses delayed simulations in conjunction with antichain techniques is [SL14]. However, as we point out in Section 7.4.1, delayed simulations cannot (in general) be used directly in the standard antichain algorithms.

1.3 Joint Work

The results presented in this thesis are mainly based on jointly authored papers. The results in Chapter 3 have been published in the proceedings of RP'14 [Ho14]. Some parts of Chapters 4 and 5 have been published in the proceedings of RV'14 [HOW14]. The paper was co-authored by Joël Ouaknine and James Worrell. A conference version of Chapter 6 has been accepted to appear in the proceedings of FoSSaCS'15 [HO15]. The paper was co-authored by Joël Ouaknine. Finally, Chapter 7 reports an on-going work, of which we are currently preparing a journal version for a submission to *Operations Research* [DHO+15]. The paper is co-authored by Nir Drucker, Joël Ouaknine, Michal Penn and Ofer Strichman.

Chapter 2

Preliminaries

In this chapter, we briefly recall some standard concepts and define some notations and conventions used throughout this thesis. More specific notions will be introduced in relevant chapters as necessary.

2.1 Automata and Logics

Finite automata. Given a finite alphabet Σ, a *finite word* u' over Σ is a finite sequence of symbols in Σ. We write $|u'|$ for the length of u', i.e., the number of symbols in u'. The set of all finite words over Σ is denoted by Σ^*. A *finite-word language* over Σ is a subset of Σ^*. A common formalism for defining finite-word languages are *finite automata*, which we define formally as follows.

Definition 2.1.1. *A non-deterministic finite automaton $\mathcal{A}$ is a tuple $\langle \Sigma, S, S_0, \Delta, F \rangle$ where*

- *Σ is a finite alphabet*
- *S is a finite set of states*
- *$S_0 \subseteq S$ is the set of initial states*
- *$\Delta \subseteq S \times \Sigma \times S$ is the transition relation*
- *F is the set of accepting states.*

We say that $\mathcal{A}$ is *deterministic* if (i) S_0 consists of a single state and (ii) for each $s \in S$ and $a \in \Sigma$, there is at most one transition $\langle s, a, s' \rangle$. $\mathcal{A}$ is *complete* if there is at least one transition $\langle s, a, s' \rangle$ for each $s \in S$ and $a \in \Sigma$. A *run* of $\mathcal{A}$

on $u' = u'_0 u'_1 \dots u'_{|u|-1}$ ($u'_i \in \Sigma$ for all i, $0 \leq i < |u'|$) is the following sequence of transitions with $s_0 \in S_0$:

$$s_0 \xrightarrow{u'_0} s_1 \xrightarrow{u'_1} \cdots \xrightarrow{u'_{|u|-1}} s_{|u'|}\,.$$

A run is *accepting* if it ends in an accepting state. A finite word u' is *accepted* by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on u'. The finite-word language defined by $\mathcal{A}$ is the set of all finite words u' accepted by $\mathcal{A}$.

It is well-known that for each NFA, one can construct an equivalent DFA defining the same finite-word language with an exponential blow-up [RS59].

Alternating automata. The non-determinism in NFA can be seen as the power to specify successors as disjunctions over states. A generalisation of this idea leads to the definition of *alternating automata* [CS76], in which we define the translation relation Δ as a (partial) mapping from $S \times \Sigma$ to the set of positive Boolean combinations over states.[1] However, alternation does not add expressiveness: each AFA can be translated into an equivalent NFA with an exponential blow-up [FJY90].

Büchi automata. Given a finite alphabet Σ, an *infinite word* u over Σ is an infinite sequence of symbols in Σ. We denote the set of all infinite words over Σ by Σ^ω. A *language* over Σ is a subset of Σ^ω. The definition of *Büchi automata* [Büc60] is the same as that of NFA; the definition of runs is also similar except that the sequence of transitions is infinite. A run of a Büchi automaton $\mathcal{B}$ is *accepting* if it visits the set of accepting states *infinitely often*. An infinite word u is *accepted* by $\mathcal{B}$ if there is an accepting run of $\mathcal{B}$ on u. The language defined by $\mathcal{B}$ is the set of all infinite words accepted by $\mathcal{B}$.

We can also define alternating automata with Büchi acceptance conditions. It is known that ABA and NBA are equally expressive [MH84]; however, unlike the finite-word case, NBA is strictly more expressive than DBA.

Monadic First-Order Logic of Order. For dealing with words, another common formalism is the *Monadic First-Order Logic of Order* ($\mathsf{FO}[<]$).

Definition 2.1.2. *Given a set of monadic predicates* $\mathbf{P}$*, the set of* $\mathsf{FO}[<]$ *formulas is generated by the grammar*

$$\vartheta ::= P(x) \mid x < x' \mid \mathbf{true} \mid \vartheta_1 \wedge \vartheta_2 \mid \neg\vartheta \mid \exists x\, \vartheta\,,$$

[1] We omit the definition of runs of alternating automata, which can be found in, e.g., [Var96].

where $P \in \mathbf{P}$ and x, x' are variables.

We will use the usual syntactic sugar, e.g., $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\mathbf{false} \equiv \neg\mathbf{true}$, $x = x' \equiv \neg(x < x') \wedge \neg(x' < x)$, etc. We interpret $\mathsf{FO}[<]$ over (finite or infinite) words as follows: for each word u over $\Sigma_{\mathbf{P}} = 2^{\mathbf{P}}$, we associate a structure M_u. Its universe U_u is $\{0, 1, \ldots, |u| - 1\}$. The order relation $<$ and monadic predicates in $\mathbf{P}$ are interpreted in the expected way. The satisfaction relation is defined inductively in the standard fashion. For an $\mathsf{FO}[<]$ formula $\vartheta(x_0, \ldots, x_{n-1})$ and a word u, We write $M_u, i_0, \ldots, i_{n-1} \models \vartheta(x_0, \ldots, x_{n-1})$ (or simply $u, i_0, \ldots, i_{n-1} \models \vartheta(x_0, \ldots, x_{n-1})$) if $i_0, \ldots, i_{n-1} \in U_u$ and $\vartheta(i_0, \ldots, i_{n-1})$ holds in M_u.

Linear Temporal Logic. A even more popular formalism in verification is the *Linear Temporal Logic* (LTL) [Pnu77], which itself is a fragment of $\mathsf{FO}[<]$.

Definition 2.1.3. *Given a set of monadic predicates $\mathbf{P}$, the set of LTL formulas is generated by the grammar*

$$\varphi ::= P \mid \mathbf{true} \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2 \mid \varphi_1 \,\mathcal{S}\, \varphi_2$$

where $P \in \mathbf{P}$.

The (future-only) fragment $\mathsf{LTL}_{\mathsf{fut}}$ is obtained by banning subformulas of the form $\varphi_1 \,\mathcal{S}\, \varphi_2$. We allow the usual syntactic sugar, e.g., $\Diamond\varphi \equiv \mathbf{true}\,\mathcal{U}\,\varphi$, $\boxminus\varphi \equiv \neg\big(\mathbf{true}\,\mathcal{S}\,(\neg\varphi)\big)$, etc. Each LTL formula φ can be seen as an $\mathsf{FO}[<]$ formula $\vartheta(x)$ with a single free variable x. For example, $\varphi_1 \,\mathcal{U}\, \varphi_2$ is equivalent to

$$\exists x' \Big(x < x' \wedge \vartheta_2(x') \wedge \forall x'' \big(x < x'' \wedge x'' < x' \implies \vartheta_1(x'') \big) \Big)$$

where ϑ_1, ϑ_2 are the $\mathsf{FO}[<]$ equivalents of φ_1, φ_2.[2] A word u *satisfies* φ (written as $u \models \varphi$) if and only if $u, 0 \models \vartheta(x)$ where ϑ is the $\mathsf{FO}[<]$ equivalent of φ. We denote by $|\varphi|$ the number of subformulas in φ.

A seminal result by Kamp [Kam68] states that each $\mathsf{FO}[<]$ formula with one free variable is equivalent to an LTL formula. Gabbay [GPSS80] showed that if we consider only *initial equivalence* (i.e., equivalent when evaluated at the beginning of words), then $\mathsf{LTL}_{\mathsf{fut}}$ (the fragment of LTL without 'Since' $\mathcal{S}$) is already as expressive as $\mathsf{FO}[<]$.

[2] In this thesis we adopt the *strict-future* semantics for $\mathcal{U}$ and $\mathcal{S}$, i.e., φ_1 is not required to hold at the current instant.

Safety and liveness. A language specified by a logic formula or an automaton is sometimes referred to as a *property*. Two classes of properties are of special interest in verification: *safety properties* and *liveness properties* [AS87]. Conceptually, a safety property asserts that '*something bad never happens*' and a liveness property asserts that '*something good eventually happens*'. Formally, for any property Φ,

$$\begin{aligned} \Phi \text{ is a safety property} &\iff \forall u \notin \Phi \left(\exists u' \left(\exists w \left(u = u'w\right) \wedge \forall v \left(u'v \notin \Phi\right)\right)\right) \\ \Phi \text{ is a liveness property} &\iff \forall u' \left(\exists v \left(u'v \in \Phi\right)\right) \end{aligned}$$

where u' is a finite word and u, v, w are infinite words.

2.2 Reasoning about Time

Timed words. Let the time domain $\mathbb{T}$ be a subinterval of $\mathbb{R}_{\geq 0}$ that contains 0. A *time sequence* $\tau = \tau_0\tau_1\ldots$ is a non-empty finite or infinite sequence over $\mathbb{T}$ (called *timestamps*) that satisfies the requirements below (we denote the length of τ by $|\tau|$):

- *Initialisation*[3]: $\tau_0 = 0$
- *Strict monotonicity*: For all i, $0 \leq i < |\tau| - 1$, we have $\tau_i < \tau_{i+1}$.

If τ is infinite we require it to be unbounded, i.e., we disallow so-called Zeno sequences. A $\mathbb{T}$-*timed word* over finite alphabet Σ is a pair $\rho = (\sigma, \tau)$, where $\sigma = \sigma_0\sigma_1\ldots$ is a non-empty finite or infinite word over Σ and τ is a time sequence over $\mathbb{T}$ of the same length. We usually refer to a $\mathbb{T}$-timed word simply as a *timed word* when $\mathbb{T} = \mathbb{R}_{\geq 0}$.[4] We refer the pair (σ_i, τ_i) as the i^{th} *event* in ρ, and define the *distance* between i^{th} and j^{th} $(i \leq j)$ events to be $\tau_j - \tau_i$. In this sense, a timed word can be equivalently regarded as a sequence of events. We denote by $|\rho|$ the number of events in ρ. A *position* in ρ is a number i such that $0 \leq i < |\rho|$. The *duration* of ρ is defined as $\tau_{|\rho|-1}$ if ρ is finite. We write $t \in \rho$ if t is equal to one of the timestamps in ρ. For a finite alphabet Σ, we write $T\Sigma^*$ and $T\Sigma^\omega$ for the respective sets of finite and infinite timed words over Σ. A *timed (finite-word) language* over Σ is a subset of $T\Sigma^\omega$ $(T\Sigma^*)$.

Timed automata. *Timed automata* [AD94] was first introduced by Alur and Dill in the early 1990s and has since emerged as the standard model for real-time systems.

[3]This requirement is natural in the present context as all the logics we consider in this thesis are *translation invariant*: two timed words are indistinguishable by formulas (of these logics) if they differ only by a fixed delay.

[4]By the non-Zeno requirement, if $\mathbb{T}$ is bounded then a $\mathbb{T}$-timed word must be a finite timed word.

Definition 2.2.1. *Given a set of clocks X, the set $\Phi(X)$ of clock constraints δ is defined inductively by*

$$\delta ::= \varphi_1 \wedge \varphi_2 \mid x \bowtie c$$

where $x \in X$, $c \in \mathbb{N}$ and $\bowtie \in \{<, \leq, >, \geq\}$.

Definition 2.2.2. *A (non-deterministic) timed automaton $\mathcal{A}$ is a tuple $\langle \Sigma, S, S_0, X, I, E, F \rangle$ where*

- *Σ is a finite alphabet*
- *S is a finite set of locations*
- *$S_0 \subseteq S$ is the set of initial locations*
- *X is a finite set of clocks*
- *$I : S \mapsto \Phi(X)$ is a mapping that labels each location in S with some clock constraint in $\Phi(X)$ (an 'invariant')*[5]
- *$E \subseteq S \times S \times \Sigma \times 2^X \times \Phi(X)$ is the set of edges. An edge $\langle s, s', a, \lambda, \delta \rangle$ denotes an a-labelled edge from location s to location s' where δ (a 'guard') specifies when the edge is enabled and $\lambda \subseteq X$ is the set of clocks to be reset with this edge*
- *F is the set of accepting locations.*

We say that $\mathcal{A}$ is *deterministic* if it (i) has only one initial location and (ii) for each $s \in S$, $a \in \Sigma$ and every pair of edges $\langle s, s_1, a, \lambda_1, \delta_1 \rangle$, $\langle s, s_2, a, \lambda_2, \delta_2 \rangle$, δ_1 and δ_2 are mutually exclusive (i.e., $\delta_1 \wedge \delta_2$ is unsatisfiable). We say that $\mathcal{A}$ is *complete* if for each $s \in S$ and $a \in \Sigma$, the disjunction of the clock constraints of the a-labelled edges starting at s is a valid formula.

Let $\mathcal{A}$ has n clocks. We define its set of clock values as $\mathsf{Val} = [0, c_{max}] \cup \{\top\}$ where c_{max} is the maximum constant appearing in $\mathcal{A}$. A *state* of $\mathcal{A}$ as a pair $(s, \mathbf{v})$ where $s \in S$ is a location and $\mathbf{v} \in \mathsf{Val}^n$ is a *clock valuation*. Write $\mathbf{v}(x)$ for the value of clock x in $\mathbf{v}$. We denote by $Q = S \times \mathsf{Val}^n$ the set of all states of $\mathcal{A}$. A *run* of $\mathcal{A}$ on a timed word may be described as follows: it takes some edge when an event arrives, otherwise it stays in the same location as time elapses. More specifically, $\mathcal{A}$ induces a labelled transition system $\mathcal{T}_\mathcal{A} = \langle Q, \rightsquigarrow, \rightarrow \rangle$ where $\rightsquigarrow \subseteq Q \times \mathbb{R}_{>0} \times Q$ is the *delay-step relation* and $\rightarrow \subseteq Q \times \Sigma \times Q$ is the *discrete-step relation*. In these steps, corresponding invariants and guards must be met (define $\top > c$ for all constants c):

[5] In the literature there are also definitions of timed automata without invariants on locations, though the expressiveness remains the same [AM04].

- For $(s, \mathbf{v}) \overset{t}{\rightsquigarrow} (s', \mathbf{v}')$, $s' = s$, $\mathbf{v}' = \mathbf{v} + t$ and $\mathbf{v} + t' \models I(s)$ for all $0 \leq t' \leq t$.
- For $(s, \mathbf{v}) \overset{a}{\rightarrow} (s', \mathbf{v}')$, there is an edge $\langle s, a, s', \lambda, \delta \rangle \in E$ such that $\mathbf{v}' = \mathbf{v}[\lambda := 0]$ and $\mathbf{v} \models \delta$.

The clock valuation $\mathbf{v} + t$ maps each clock x to $\mathbf{v}(x) + t$ if $\mathbf{v}(x) + t \leq c_{max}$, otherwise $\top$. $\mathbf{v}[\lambda := 0]$ maps x to $\mathbf{v}(x)$ if $x \notin \lambda$, otherwise 0. Formally, a run of $\mathcal{A}$ on $\rho = (\sigma, \tau)$ is an alternating sequence of delay steps and discrete steps

$$(s_0, \mathbf{v}_0) \overset{\sigma_0}{\rightarrow} (s_1, \mathbf{v}_1) \overset{d_0}{\rightsquigarrow} (s_2, \mathbf{v}_2) \overset{\sigma_1}{\rightarrow} (s_3, \mathbf{v}_3) \overset{d_1}{\rightsquigarrow} (s_4, \mathbf{v}_4) \overset{\sigma_2}{\rightarrow} \cdots$$

where $d_i = \tau_{i+1} - \tau_i$ for $i \geq 0$, $s_0 \in S_0$ and $\mathbf{v}_0 = 0^n$. A finite timed word ρ' is *accepted* by $\mathcal{A}$ if there is an *accepting* run (i.e., ending in an accepting location) of $\mathcal{A}$ on u'. We can also equip $\mathcal{A}$ with a Büchi acceptance condition; in this case, a run is *accepting* if it visits an accepting location infinitely often, and an infinite timed word ρ is *accepted* by $\mathcal{A}$ if there is such a run of $\mathcal{A}$ on ρ. The *timed (finite-word) language* defined by $\mathcal{A}$ is the set of (finite) timed words accepted by $\mathcal{A}$.

Example 2.2.3 ([AM04]). Consider the timed automaton with $\Sigma = \{a, b\}$ in Figure 2.1. The automaton accepts timed words containing an a event at some time t such that no event occurs at time $t + 1$.

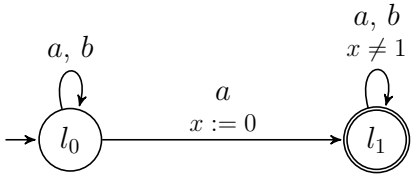


Figure 2.1: A timed automaton.

One can construct a *network of timed automata* that comprises a number of timed automata in a way similar to the standard product construction for finite automata. This enables the modelling of a complex system in terms of its components.

Monadic First-Order Logic of Order and Metric. We now define the *Monadic First-Order Logic of Order and Metric* ($\mathsf{FO}[<, +1]$) [Wil94] which subsumes all other logics discussed in this thesis.

Definition 2.2.4. *Given a set of monadic predicates* $\mathbf{P}$*, the set of* $\mathsf{FO}[<, +1]$ *formulas is generated by the grammar*

$$\vartheta ::= P(x) \mid x < x' \mid d(x, x') \sim c \mid \mathbf{true} \mid \vartheta_1 \wedge \vartheta_2 \mid \neg\vartheta \mid \exists x\, \vartheta\,,$$

where $P \in \mathbf{P}$*,* x, x' *are variables,* $\sim \in \{<, >\}$ *and* $c \in \mathbb{N}$.[6]

Metric temporal logics. Formulas of metric temporal logics are built from monadic predicates using Boolean connectives and *modalities*. A k-ary modality is defined by an $\mathsf{FO}[<, +1]$ formula $\varphi(x, X_1, \ldots, X_k)$ with a single free variable x and k free monadic predicates $X_1, \ldots, X_k$. For example, the MTL [Koy90] modality $\mathcal{U}_{(0,5)}$ is defined by the $\mathsf{FO}[<, +1]$ formula

$$\mathcal{U}_{(0,5)}(x, X_1, X_2) = \exists x' \Big(x < x' \wedge d(x, x') < 5 \wedge X_2(x') \\ \wedge \forall x'' \left(x < x'' \wedge x'' < x' \implies X_1(x'') \right) \Big) .$$

The MTL formula $\varphi_1 \, \mathcal{U}_{(0,5)} \, \varphi_2$ (using infix notation) is obtained by substituting MTL formulas φ_1, φ_2 for X_1, X_2, respectively.

Definition 2.2.5. *Given a set of monadic predicates* $\mathbf{P}$*, the set of* MTL *formulas is generated by the grammar*

$$\varphi ::= P \mid \mathbf{true} \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \varphi_1 \, \mathcal{U}_I \, \varphi_2 \mid \varphi_1 \, \mathcal{S}_I \, \varphi_2 \, ,$$

where $P \in \mathbf{P}$ *and* $I \subseteq (0, \infty)$ *is an interval with endpoints in* $\mathbb{N} \cup \{\infty\}$.

The (future-only) fragment $\mathsf{MTL}_{\mathsf{fut}}$ is obtained by banning subformulas of the form $\varphi_1 \, \mathcal{S}_I \, \varphi_2$. We write $|I|$ for $\sup(I) - \inf(I)$. If I is not present as a subscript to a given modality then it is assumed to be $(0, \infty)$. We sometimes use pseudo-arithmetic expressions to denote intervals, e.g., '≥ 1' denotes $[1, \infty)$ and '$= 1$' denotes the singleton $\{1\}$. We also employ the usual syntactic sugar, e.g., $\mathbf{false} \equiv \neg\mathbf{true}$, $\Diamond_I \varphi \equiv \mathbf{true} \, \mathcal{U}_I \, \varphi$, $\Diamond\!\!\!-_I \varphi \equiv \mathbf{true} \, \mathcal{S}_I \, \varphi$, $\Box_I \varphi \equiv \neg\Diamond_I \neg\varphi$ and $\bigcirc_I \varphi \equiv \mathbf{false} \, \mathcal{U}_I \, \varphi$, etc. For convenience, we also allow 'weak' temporal operators as syntactic sugar, e.g., $\varphi_1 \, \mathcal{U}^w_I \, \varphi_2 \equiv \varphi_1 \wedge (\varphi_1 \, \mathcal{U}_I \, \varphi_2)$ if $0 \notin I$ and $\varphi_1 \, \mathcal{U}^w_I \, \varphi_2 \equiv \varphi_2 \vee (\varphi_1 \wedge (\varphi_1 \, \mathcal{U}_I \, \varphi_2))$ if $0 \in I$.[7] We denote by $|\varphi|$ the number of subformulas in φ.

The pointwise semantics. With each $\mathbb{T}$-timed word $\rho = (\sigma, \tau)$ over $\Sigma_{\mathbf{P}}$ we associate a structure M_ρ. Its universe U_ρ is the subset $\{\tau_i \mid 0 \leq i < |\rho|\}$ of $\mathbb{T}$. The order relation $<$ and monadic predicates in $\mathbf{P}$ are interpreted in the expected way, e.g., $P(\tau_i)$ holds in M_ρ iff $P \in \sigma_i$. The binary *distance predicate* $d(x, x') \sim c$

[6] Note that whilst we refer to the logic as $\mathsf{FO}[<, +1]$, we adopt here an equivalent definition using a binary distance predicate $d(x, x')$ (as in [Wil94]) in place of the usual $+1$ function symbol.

[7] We allow $0 \in I$ (i.e., $I \subseteq [0, \infty)$) in the case of weak temporal operators.

holds iff $|x - x'| \sim c$. The satisfaction relation is defined inductively as usual. We write $M_\rho, t_0, \ldots, t_{n-1} \models \vartheta(x_0, \ldots, x_{n-1})$ (or $\rho, t_0, \ldots, t_{n-1} \models \vartheta(x_0, \ldots, x_{n-1})$) if $t_0, \ldots, t_{n-1} \in U_\rho$ and $\vartheta(t_0, \ldots, t_{n-1})$ holds in M_ρ. We say that two $\mathsf{FO}[<, +1]$ formulas $\vartheta_1(x)$ and $\vartheta_2(x)$ are *equivalent* over $\mathbb{T}$-timed words if for all $\mathbb{T}$-timed words ρ and $t \in U_\rho$,

$$\rho, t \models \vartheta_1(x) \iff \rho, t \models \vartheta_2(x)\,.$$

We say that a metric logic L' is *expressively complete* for metric logic L over $\mathbb{T}$-timed words iff for any formula $\vartheta(x) \in L$, there is an equivalent formula $\varphi(x) \in L'$ over $\mathbb{T}$-timed words. We say that L' is *at least as expressive* as (or *more expressive* than) L over $\mathbb{T}$-timed words (written $L \subseteq L'$) iff for any formula $\vartheta \in L$, there is an *initially equivalent* formula $\varphi \in L'$ over $\mathbb{T}$-timed words (i.e., ϑ and φ evaluates to the same truth value at the beginning of any $\mathbb{T}$-timed word). If $L \subseteq L'$ but $L' \nsubseteq L$ then we say that L' is *strictly more expressive* than L (or L is *strictly less expressive* than L') over $\mathbb{T}$-timed words.

As we have seen earlier, each MTL formula can be defined as an $\mathsf{FO}[<, +1]$ formula with a single free variable. Here, for the sake of completeness we give an (equivalent) traditional inductive definition of the satisfaction relation for MTL over timed words. We write $\rho \models \varphi$ if $\rho, 0 \models \varphi$.

Definition 2.2.6. *The satisfaction relation $\rho, i \models \varphi$ for an MTL formula φ, a timed word $\rho = (\sigma, \tau)$ and a position i in ρ is defined as follows:*

- $\rho, i \models P$ *iff* $P(\tau_i)$ *holds in* M_ρ
- $\rho, i \models \mathbf{true}$
- $\rho, i \models \varphi_1 \wedge \varphi_2$ *iff* $\rho, i \models \varphi_1$ *and* $\rho, i \models \varphi_2$
- $\rho, i \models \neg\varphi$ *iff* $\rho, i \not\models \varphi$
- $\rho, i \models \varphi_1 \,\mathcal{U}_I\, \varphi_2$ *iff there exists* j, $i < j < |\rho|$ *such that* $\rho, j \models \varphi_2$, $\tau_j - \tau_i \in I$ *and* $\rho, k \models \varphi_1$ *for all* k *with* $i < k < j$
- $\rho, i \models \varphi_1 \,\mathcal{S}_I\, \varphi_2$ *iff there exists* j, $1 \leq j < i$ *such that* $\rho, j \models \varphi_2$, $\tau_i - \tau_j \in I$ *and* $\rho, k \models \varphi_1$ *for all* k *with* $j < k < i$.

Example 2.2.7. The $\mathsf{MTL}_{\mathsf{fut}}$ formula

$$\varphi = \Box(P \implies \Diamond_{<3} Q) \tag{2.1}$$

is satisfied by a timed word ρ if and only if, whenever there is a P-event in ρ (say at time t), there is a Q-event in ρ with timestamp in $(t, t+3)$.

Safety and liveness relative to the divergence of time. Recall that we require the timestamps of any infinite timed word to be a strictly increasing divergent sequence. Under this assumption, we can define *safety properties* and *liveness properties* in the timed case in exactly the same way as in the untimed case. For example, (2.1) is a safety property as any infinite timed word u violating φ must have a bad prefix u' such that there is a P-event in u' with no Q-event in the following three time units. On the other hand, had we allowed Zeno timed words, φ would not be safety as

$$(\{P\}, 0)(\{P\}, 1)(\{P\}, 1 + \frac{1}{2})(\{P\}, 1 + \frac{1}{2} + \frac{1}{4}) \ldots$$

violates φ without having a prefix that cannot be extended into an infinite timed word satisfying φ. The notion we adopt here is called *safety and liveness relative to the divergence of time* in the literature [HMP92].

The continuous semantics. Another way to interpret metric logics is to regard time as a continuous entity; a behaviour of a system can therefore be viewed as a continuous function. Formally, a *flow* over finite alphabet Σ is a function $f : \mathbb{T} \mapsto \Sigma$ that is *finitely variable*, i.e., the restriction of f to a subinterval of $\mathbb{T}$ of finite length has only finite number of discontinuities. We sometimes write $\mathbb{T}$-flows for flows with time domain $\mathbb{T}$. With each flow f over $\Sigma_{\mathbf{P}}$ we associate a structure M_f. Its universe U_f is $\mathbb{T}$. The order relation $<$ and monadic predicates in $\mathbf{P}$ are interpreted in the expected way, e.g., $P(x)$ holds in M_f iff $P \in f(x)$. The binary *distance predicate* $d(x, x') \sim c$ holds iff $|x - x'| \sim c$. We write $M_f, t_0, \ldots, t_{n-1} \models \vartheta(x_0, \ldots, x_{n-1})$ (or $f, t_0, \ldots, t_{n-1} \models \vartheta(x_0, \ldots, x_{n-1})$) if $t_0, \ldots, t_{n-1} \in U_f$ and $\vartheta(t_0, \ldots, t_{n-1})$ holds in M_f. We say that $\mathsf{FO}[<, +1]$ formulas $\vartheta_1(x)$ and $\vartheta_2(x)$ are equivalent over $\mathbb{T}$-flows if for all $\mathbb{T}$-flows f and $t \in U_f$,

$$f, t \models \vartheta_1(x) \iff f, t \models \vartheta_2(x)\,.$$

A metric logic L' is *expressively complete* for metric logic L over $\mathbb{T}$-flows iff for any formula $\vartheta(x) \in L$, there is an equivalent formula $\varphi(x) \in L'$ over $\mathbb{T}$-flows. We say that L' is *at least as expressive* as (or *more expressive* than) L over $\mathbb{T}$-flows (written $L \subseteq L'$) iff for any formula $\vartheta \in L$, there is an *initially equivalent* formula $\varphi \in L'$ over $\mathbb{T}$-flows (i.e., ϑ and φ evaluates to the same truth value at the beginning of any $\mathbb{T}$-flow). If $L \subseteq L'$ but $L' \not\subseteq L$ then we say that L' is *strictly more expressive* than L

(or L is *strictly less expressive* than L') over $\mathbb{T}$-flows.

The satisfaction relation for MTL over flows is defined as follows. We write $f \models \varphi$ if $f, 0 \models \varphi$.

Definition 2.2.8. *The satisfaction relation $f, t \models \varphi$ for an MTL formula φ, a flow f and $t \in U_f$ is defined as follows:*

- $f, t \models P$ *iff* $P(t)$ *holds in* M_f
- $f, t \models \mathbf{true}$
- $f, t \models \varphi_1 \wedge \varphi_2$ *iff* $f, t \models \varphi_1$ *and* $f, t \models \varphi_2$
- $f, t \models \neg\varphi$ *iff* $f, t \not\models \varphi$
- $f, t \models \varphi_1 \,\mathcal{U}_I\, \varphi_2$ *iff there exists* $t' > t$, $t' \in \mathbb{T}$ *such that* $f, t' \models \varphi_2$, $t' - t \in I$ *and* $f, t'' \models \varphi_1$ *for all* t'' *with* $t < t'' < t'$
- $f, t \models \varphi_1 \,\mathcal{S}_I\, \varphi_2$ *iff there exists* $t' < t$, $t' \in \mathbb{T}$ *such that* $f, t' \models \varphi_2$, $t - t' \in I$ *and* $f, t'' \models \varphi_1$ *for all* t'' *with* $t' < t'' < t$.

Relating Two Semantics Note that timed words can be regarded as a particular kind of flow: for a given $\mathbb{T}$-timed word ρ over Σ_P, we can introduce a 'silent' monadic predicate P_ϵ and construct the corresponding $\mathbb{T}$-flow f^ρ over $\Sigma_{\mathbf{P}'}$, where $\mathbf{P}' = \mathbf{P} \cup \{P_\epsilon\}$, as follows:

- $f^\rho(\tau_i) = \sigma_i$ for all i, $0 \leq i < |\rho|$
- $f^\rho(\tau_i) = \{P_\epsilon\}$.

This enables us to interpret metric logics over timed words 'continuously'. We can thus compare the expressiveness of metric logics in both semantics by restricting the models of the continuous interpretations of metric logics to flows of this form (i.e., f^ρ for some timed word ρ). For example, we say that continuous $\mathsf{FO}[<, +1]$ is at least as expressive as pointwise $\mathsf{FO}[<, +1]$ since for each $\mathsf{FO}[<, +1]$ formula $\vartheta_{pw}(x)$, there is an 'equivalent' $\mathsf{FO}[<, +1]$ formula $\vartheta_{cont}(x)$ such that $\rho, t \models \vartheta_{pw}(x)$ iff $f^\rho, t \models \vartheta_{cont}(x)$.

Example 2.2.9. Consider the timed word ρ illustrated in Figure 2.2 where the red boxes denote P-events. The $\mathsf{MTL}_{\mathsf{fut}}$ formula

$$\varphi = \Diamond(\Diamond_{=1} P)$$

does not hold at the beginning of ρ in the pointwise semantics (i.e., $\rho \not\models \varphi$) since there is no event at exactly one time unit before the second event in ρ. On the other hand, φ holds at the beginning of ρ in the continuous semantics (i.e., $f^\rho \models \varphi$) since there is a point (at which P_ϵ holds) at exactly one time unit before the second event in f^ρ. We can, however, simulate the pointwise semantics with

$$\varphi' = \Diamond\Big(\neg P_\epsilon \wedge \big(\Diamond_{=1}(\neg P_\epsilon \wedge P)\big)\Big)$$

such that for all timed words π we have $\pi \models \varphi$ iff $f^\pi \models \varphi'$.

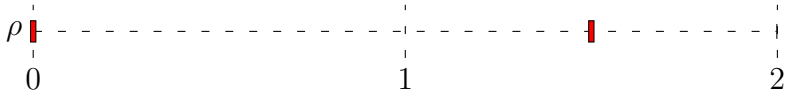


Figure 2.2: The timed word ρ.

As the example above shows, the pointwise and continuous interpretations of metric logics differs in the range of first-order quantifiers. While the ability to quantify over time points between events appears to add to the expressiveness of metric logics, this is not the case for $\mathsf{FO}[<,+1]$ as both interpretations indeed have the same expressiveness (when one considers only flows of the form f^ρ) [DHV07].[8] In contrast, it is known that MTL is strictly more expressive in the continuous semantics than in the pointwise semantics [PD06].

2.3 Model Checking

A key advantage in using $\mathsf{LTL}_{\mathsf{fut}}$ (or LTL) in verification is that its *model-checking* problem is PSPACE-complete [SC85], much better than the complexity of the same problem for $\mathsf{FO}[<]$ (non-elementary [Sto74]). Given a Büchi automaton $\mathcal{A}$ that models the system under scrutiny, and a specification expressed as an $\mathsf{LTL}_{\mathsf{fut}}$ formula Φ, the corresponding model-checking problem asks whether the language defined by $\mathcal{A}$ is included in the language defined by Φ. By a fundamental result in verification—$\mathsf{LTL}_{\mathsf{fut}}$ formulas can be translated into Büchi automata [WVS83]—this reduces to the *emptiness* problem on the product Büchi automaton of $\mathcal{A}$ and the Büchi automaton $\mathcal{B}_{\neg\Phi}$ translated from $\neg\Phi$. The latter problem can be solved, e.g., by a standard fixed-point algorithm [EL86]. This is sometimes called the *automata-theoretic approach* to $\mathsf{LTL}_{\mathsf{fut}}$ model checking.

[8] The translation in [DHV07] also holds in a time-bounded setting with trivial modifications.

Definition 2.3.1 (The emptiness problem for Büchi automata)**.** *Given a Büchi automaton $\mathcal{A}$, does $\mathcal{A}$ have an accepting run?*

In the real-time setting, given a timed (Büchi) automaton $\mathcal{A}$ and a specification φ (e.g., a formula of some metric logic), the corresponding model-checking problem asks whether the timed (finite-word) language defined by $\mathcal{A}$ is included in the timed (finite-word) language defined by φ. By analogy with the untimed case, one may solve this problem by first translating $\neg\varphi$ into a timed automaton $\mathcal{A}_{\neg\varphi}$ and then checking the emptiness of the product of $\mathcal{A}$ and $\mathcal{A}_{\neg\varphi}$. This procedure does work for certain metric logics; for example, each formula of $\mathsf{MITL}_{\mathsf{fut}}$ (the non-punctual fragment of $\mathsf{MTL}_{\mathsf{fut}}$) can be translated into a timed automaton, and the model-checking problem for timed automata against $\mathsf{MITL}_{\mathsf{fut}}$ is EXPSPACE-complete [AFH96]. On the other hand, it does not apply to $\mathsf{MTL}_{\mathsf{fut}}$ as $\mathsf{MTL}_{\mathsf{fut}}$ formulas cannot be translated into timed automata in general. We summarise the decidability/complexity of model checking timed automata against $\mathsf{MTL}_{\mathsf{fut}}$ in Table 2.1.

	finite timed words	infinite timed words
in the pointwise semantics	Non-Prim.-Rec. [OW05]	Undecidable [OW06a]
in the continuous semantics	Undecidable [AFH96]	Undecidable [AFH96]

Table 2.1: Model checking timed automata against $\mathsf{MTL}_{\mathsf{fut}}$.

2.4 Monitoring

We now formalise the notion of *monitoring*.

Definition 2.4.1 (The prefix problem for $\mathsf{LTL}_{\mathsf{fut}}$ [BKV13])**.** *Given an $\mathsf{LTL}_{\mathsf{fut}}$ formula Φ and a finite word u', do all infinite extensions of u' satisfy Φ?*

If the answer is 'yes', then we say that u' is a *good prefix* for Φ. Similarly, u' is a *bad prefix* for Φ if the answer to the dual problem is 'yes', i.e., none of its infinite extensions satisfies Φ. The monitoring problem for $\mathsf{LTL}_{\mathsf{fut}}$ takes two inputs: an $\mathsf{LTL}_{\mathsf{fut}}$ formula Φ and an infinite word u. In contrast to standard decision problems, the latter input is given *incrementally*, i.e., one symbol at a time; a monitor (a procedure that 'solves' the $\mathsf{LTL}_{\mathsf{fut}}$ monitoring problem) is required to continuously check whether the currently accumulated finite word u' (a prefix of u) is a good/bad prefix for Φ.

For $\mathsf{LTL}_{\mathsf{fut}}$, there is a well-known monitor construction closely related to the automata-theoretic approach to $\mathsf{LTL}_{\mathsf{fut}}$ model checking [ABLS05]. For example, a DFA $\mathcal{D}$ that accepts all bad prefixes for an $\mathsf{LTL}_{\mathsf{fut}}$ formula Φ can be constructed as follows:

1. Translate Φ into a Büchi automaton $\mathcal{B}_\Phi$.
2. Perform emptiness checking *per state* on $\mathcal{B}_\Phi$, i.e., check whether the language accepted from s is empty for each state s of $\mathcal{B}_\Phi$.
3. Regard $\mathcal{B}_\Phi$ as an NFA $\mathcal{A}$ and define its set of accepting states as the set of all 'non-empty' states of $\mathcal{B}_\Phi$.
4. Determinise $\mathcal{A}$ with the standard subset construction.
5. Complement the resulting automaton to obtain $\mathcal{D}$.

The same construction can also be used for LTL. Unfortunately, this methodology does not carry over directly to the timed case as (i) $\mathsf{MTL}_{\mathsf{fut}}$ and time automata are expressively incomparable; (ii) it is not clear how to check emptiness *per state* on a timed automaton; (iii) timed automata are not determinisable in general.

Part I

Metric Temporal Logics: Expressiveness and Monitoring

Chapter 3

Expressive Completeness over Bounded Timed Words

Contrary to the situation in the usual case where time is unbounded, $\mathsf{MTL}_{\mathsf{fut}}$ is as expressive as $\mathsf{FO}[<,+1]$ over time domains of the form $[0,N)$ in the *continuous* semantics [ORW09]. In this chapter, we study the expressiveness of MTL (and its various fragments and extensions) in a time-bounded *pointwise* setting, i.e., all timed words are assumed to have durations less than a positive integer N.

We first recall MTL EF games [PS11], which serves as our main tool in proving expressiveness results. Then we demonstrate a strict hierarchy of metric temporal logics (based on their expressiveness over bounded timed words) as we extend $\mathsf{MTL}_{\mathsf{fut}}$ incrementally towards $\mathsf{FO}[<,+1]$. Finally, we show that MTL, equipped with both the forwards and backwards temporal modalities 'generalised Until' ($\mathfrak{U}_I^c$) and 'generalised Since' ($\mathfrak{S}_I^c$), has precisely the same expressive power as $\mathsf{FO}[<,+1]$ over bounded time domains in the pointwise semantics (and also, trivially, in the continuous semantics). This extended version of MTL, written $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$, therefore yields a definitive real-time analogue of Kamp's theorem over bounded domains. For the time-bounded satisfiability and model-checking problems, we show that the relevant constructions (and hence the complexity bounds) for MTL in [ORW09] carry over to our new logic $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$.

3.1 MTL EF Games

Ehrenfeucht-Fraïssé games are a handy tool in proving the inexpressibility of certain properties in first-order logics. In many proofs in this chapter, we resort to (extended

versions of) Pandya and Shah's MTL **EF *games*** on timed words [PS11], which itself is a timed generalisation of Etessami and Wilke's LTL EF games [EW96].

An m-round MTL EF game starts with round 0 and ends with round m. The game is played by two players (*Spoiler* and *Duplicator*) on a pair of timed words ρ and ρ'.[1] A *configuration* is a pair of positions (i, j), respectively in ρ and ρ'. In each round r $(0 \leq r \leq m)$, the game proceeds as follows. *Spoiler* first checks whether the two events that correspond to the current configuration (i_r, j_r) in ρ and ρ' satisfy the same set of monadic predicates. If this is not the case then he wins the game. Otherwise if $r < m$, *Spoiler* chooses an interval $I \subseteq (0, \infty)$ with endpoints in $\mathbb{N} \cup \{\infty\}$ and plays either of the following moves:

- $\mathcal{U}_I$*-move*: *Spoiler* chooses one of the two timed words (say ρ). He then picks i'_r such that $\tau_{i'_r} - \tau_{i_r} \in I$ where $\tau_{i'_r}$ and τ_{i_r} are the corresponding timestamps in ρ (if there is no such i'_r then *Duplicator* wins the game). *Duplicator* must choose a position j'_r in ρ' such that the difference of the corresponding timestamps in ρ' is in I. If she cannot find such a position then *Spoiler* wins the game. Otherwise, *Spoiler* plays either of the following 'parts':
  - $\Diamond$*-part*: The game proceeds to the next round with $(i_{r+1}, j_{r+1}) = (i'_r, j'_r)$.
  - $\mathcal{U}$*-part*: If $j'_r = j_r + 1$ the game proceeds to the next round with $(i_{r+1}, j_{r+1}) = (i'_r, j'_r)$. If $i'_r = i_r + 1$ but $j'_r \neq j_r + 1$ then *Spoiler* wins the game. Otherwise *Spoiler* picks another position j''_r in ρ' such that $j_r < j''_r < j'_r$. *Duplicator* have to choose a position i''_r in ρ such that $i_r < i''_r < i'_r$ in response. If she cannot find such a position then *Spoiler* wins the game; otherwise the game proceeds to the next round with $(i_{r+1}, j_{r+1}) = (i''_r, j''_r)$.
- $\mathcal{S}_I$*-move*: Defined symmetrically.

We say that *Duplicator* has a *winning strategy* for the m-round MTL EF game on ρ, ρ' that starts from configuration (i, j) if and only if, no matter how *Spoiler* plays, he cannot win the m-round MTL EF game on ρ, ρ' with $(i_0, i_0) = (i, j)$. If this is not the case then we say that *Spoiler* has a winning strategy.

It is not hard to see that the moves in MTL EF games are closely related to the usage of modalities in MTL formulas. For example, the $\mathcal{U}_I$-move can be seen as *Spoiler*'s attempt to verify that a formula of the form $\varphi_1 \,\mathcal{U}_I\, \varphi_2$ holds at i_r in ρ if and only if it holds at j_r in ρ': the $\Diamond$-part and the remaining rounds verify that φ_2 holds

[1] We follow the convention that *Spoiler* is male and *Duplicator* is female.

at i'_r in ρ iff it holds at j'_r in ρ', whereas the $\mathcal{U}$-part and the remaing rounds verify that φ_1 holds at all i''_r, $i_r < i''_r < i'_r$ in ρ iff it holds at all j''_r, $j_r < j''_r < j'_r$ in ρ'. Formally, the following theorem relates the number of rounds of MTL EF games to the *modal depth* (i.e., the maximal depth of nesting of modalities) of MTL formulas.

Theorem 3.1.1 (MTL EF Theorem [PS11])**.** *For (finite) timed words ρ, ρ' and an* MTL *formula φ of modal depth $\leq m$, if* Duplicator *has a winning strategy for the m-round* MTL *EF game on ρ, ρ' with $(i_0, j_0) = (0, 0)$, then*

$$\rho \models \varphi \iff \rho' \models \varphi\,.$$

In other words, ρ, ρ' can be distinguished by an MTL formula of modal depth $\leq m$ if and only if *Spoiler* has a winning strategy for the m-round MTL EF game on ρ, ρ' with $(i_0, j_0) = (0, 0)$. Note that specialised versions of Theorem 3.1.1 also hold for sublogics of MTL; for example, the corresponding theorem for $\mathsf{MTL}_{\mathsf{fut}}$ is obtained by banning the $\mathcal{S}_I$-move in the game.

Example 3.1.2. Consider the timed words ρ and ρ' illustrated in Figure 3.1 where the white, red and blue boxes represent events at which no monadic predicate holds, P-events and Q-events, respectively. The positions are labelled above the events.

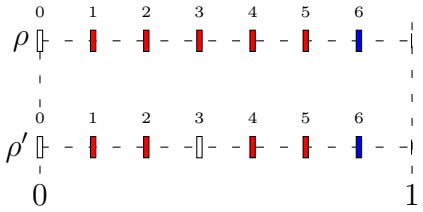


Figure 3.1: ρ and ρ' can be distinguished by $P\,\mathcal{U}\,Q$.

In the 1-round MTL EF game on ρ, ρ' with $(i_0, j_0) = (0, 0)$, a way for *Spoiler* to win the game is as follows:

1. The two events that correspond to $(i_0, j_0) = (0, 0)$ in ρ and ρ' satisfy the same set of monadic predicates, so *Spoiler* does not win here.

2. *Spoiler* chooses $I = (0, \infty)$ and $i'_0 = 6$ in ρ.

3. If *Duplicator* chooses $j'_0 \neq 6$ in ρ', she will lose at the beginning of round 1. So she chooses $j'_0 = 6$.

4. *Spoiler* plays the $\mathcal{U}$-part and chooses $j''_0 = 3$ in ρ'.

5. *Duplicator* can only choose i_0'' in ρ such that $1 \leq i_0'' \leq 5$. But she will then lose at the beginning of round 1.

It follows that there is an MTL formula of modal depth 1 that distinguishes ρ and ρ'. One such formula is $P \,\mathcal{U}\, Q$, as can be obtained from *Spoiler*'s winning strategy above.

3.2 A Hierarchy of Expressiveness

In this section, we present a sequence of successively more expressive extensions of $\mathsf{MTL}_{\mathsf{fut}}$ over bounded timed words. The technique we use here is to construct two *families* of models—parametrised by m—such that there is a certain formula of the more expressive logic telling them apart for all m, yet they cannot be distinguished by any formula of the less expressive logic with modal depth $\leq m$ (i.e., *Duplicator* has a winning strategy in the corresponding m-round MTL EF game). Along the way we highlight the key features that give rise to the differences in expressiveness. The necessity of a 'new' extension (such as the one in the next section) is justified by the fact that no known extension can lead to expressive completeness.

3.2.1 Definability of Time 0

Recall that $\mathsf{MTL}_{\mathsf{fut}}$ and $\mathsf{FO}[<, +1]$ have the same expressiveness over $[0, N)$-flows [ORW09]. This result fails in the pointwise semantics.

Proposition 3.2.1 (Corollary of [PD06, Section 8])**.** MTL *is strictly more expressive than* $\mathsf{MTL}_{\mathsf{fut}}$ *over* $[0, N)$*-timed words.*[2]

To account for this difference between the two semantics, observe that a distinctive feature of the continuous interpretation of $\mathsf{MTL}_{\mathsf{fut}}$ is exploited in [ORW09]: in any $[0, N)$-flow, the formula $\Diamond_{=(N-1)}\mathbf{true}$ holds in $[0, 1)$ and nowhere else. One can make use of conjunctions of similar formulas to determine the integer part of the current instant (where the relevant formula is being evaluated). Unfortunately, since the duration of a given bounded timed word is not known *a priori*, this trick does not work for $\mathsf{MTL}_{\mathsf{fut}}$ in the pointwise semantics. For example, the formula $\Diamond_{=1}\mathbf{true}$ does not hold at any position in the $[0, 2)$-timed word $\rho = (\sigma_0, 0)(\sigma_1, 0.5)$. However, the same effect can be achieved in MTL by using past modalities. Let

$$\varphi_{i,i+1} = \Diamond\!\!\!-_{[i,i+1)}(\neg \Diamond\!\!\!-\mathbf{true})$$

[2]The models constructed in [PD06, Section 8] are bounded timed words.

and $\Phi_{int} = \{\varphi_{i,i+1} \mid i \in \mathbb{N}\}$. Note that the subformula $\neg\diamondminus\mathbf{true}$ can only hold at the very first event (with timestamp 0), thus $\varphi_{i,i+1}$ holds only at events with timestamps in $[i, i+1)$. Denote by $\mathsf{MTL}_{\mathsf{fut}}[\Phi_{int}]$ the extension of $\mathsf{MTL}_{\mathsf{fut}}$ obtained by allowing these formulas as subformulas. It turned out that this very restrictive use of past modalities strictly increases the expressiveness of $\mathsf{MTL}_{\mathsf{fut}}$. Indeed, the main result of this chapter (Theorem 3.4.3 on page 49) crucially depends on the use of these formulas.

Proposition 3.2.2. $\mathsf{MTL}_{\mathsf{fut}}[\Phi_{int}]$ *is strictly more expressive than* $\mathsf{MTL}_{\mathsf{fut}}$ *over* $[0, N)$*-timed words.*

Proof. For a given $m \in \mathbb{N}$, we construct the following models:

$$\mathcal{A}_m = (\emptyset, 0)(\emptyset, 1 - \frac{2.5}{2m+5})(\emptyset, 1 - \frac{1.5}{2m+5})(\emptyset, 1 - \frac{0.5}{2m+5}) \dots (\emptyset, 1 + \frac{m+2.5}{2m+5}),$$
$$\mathcal{B}_m = (\emptyset, 0)(\emptyset, 1 - \frac{1.5}{2m+5})(\emptyset, 1 - \frac{0.5}{2m+5})(\emptyset, 1 + \frac{0.5}{2m+5}) \dots (\emptyset, 1 + \frac{m+3.5}{2m+5}).$$

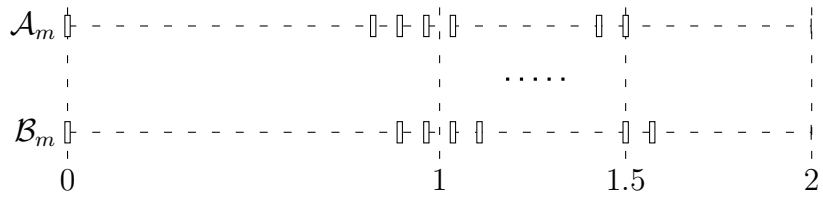


Figure 3.2: Models $\mathcal{A}_m$ and $\mathcal{B}_m$.

The models are illustrated in Figure 3.2, where each white box represents an event (at which no monadic predicate holds).

We play an m-round MTL EF game on $\mathcal{A}_m$, $\mathcal{B}_m$ and allow only $\mathcal{U}_I$-move. After round 0, either (i) $i_1 = j_1 \geq 1$ (in which case *Duplicator* can win the remaining rounds) or (ii) $(i_1, j_1) = (3, 2)$ (*Spoiler* chooses position 3 in $\mathcal{A}_m$) or $(i_1, j_1) = (4, 3)$ (*Spoiler* chooses position 3 in $\mathcal{B}_m$). In the latter case, it is easy to verify that in any remaining round r, *Duplicator* can make $i_{r+1} = j_{r+1} \geq 1$ or $(i_{r+1}, j_{r+1}) = (i_r + 1, j_r + 1)$. It follows from the MTL EF Theorem that no $\mathsf{MTL}_{\mathsf{fut}}$ formula of modal depth $\leq m$ can distinguish $\mathcal{A}_m$ and $\mathcal{B}_m$; however, the formula

$$\diamondsuit_{(0,1)}\big(\varphi_{0,1} \wedge \bigcirc(\varphi_{0,1} \wedge \bigcirc\varphi_{0,1})\big),$$

which says "in the next time unit there are three events with timestamps in $[0, 1)$", distinguishes $\mathcal{A}_m$ and $\mathcal{B}_m$ for any $m \in \mathbb{N}$ (when evaluated at position 0). □

3.2.2 Past Modalities

The conservative extension in the last subsection uses past modalities in a very restricted way. This is not sufficient for obtaining the full expressiveness of MTL: the following proposition says that non-trivial nesting of future modalities and past modalities provides more expressiveness.

Proposition 3.2.3. MTL *is strictly more expressive than* $\mathsf{MTL}_{\mathsf{fut}}[\Phi_{int}]$ *over* $[0, N)$*-timed words.*

Proof. For a given $m \in \mathbb{N}$, we construct

$$\mathcal{C}_m = (\emptyset, 0)(\emptyset, \frac{0.5}{2m+3})(\emptyset, \frac{1.5}{2m+3}) \dots (\emptyset, 2 - \frac{0.5}{2m+3}) .$$

$\mathcal{D}_m$ is constructed as $\mathcal{C}_m$ except that the event at time $\frac{m+1.5}{2m+3}$ is missing.

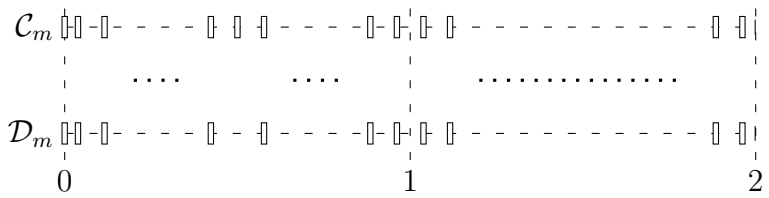


Figure 3.3: Models $\mathcal{C}_m$ and $\mathcal{D}_m$.

The models are illustrated in Figure 3.3, where each white box represents an event (at which no monadic predicate holds).

We play an m-round MTL EF game on $\mathcal{C}_m$ and $\mathcal{D}_m$, allowing only $\mathcal{U}_I$-move. For simplicity, assume that we can use special monadic predicates to refer to formulas in Φ_{int}. The proof is similar to the proof of Proposition 3.2.2: in each round r, *Duplicator* can either make (i) $i_{r+1} = j_{r+1} + 1$ and $i_{r+1} \geq m + 3$ (in which case she can win the remaining rounds) or (ii) $i_{r+1} = j_{r+1}$ and $i_{r+1} \neq 2m + 3$. It follows from the MTL EF Theorem that no $\mathsf{MTL}_{\mathsf{fut}}[\Phi_{int}]$ formula of modal depth $\leq m$ can distinguish $\mathcal{C}_m$ and $\mathcal{D}_m$; but the formula

$$\Box_{(1,2)}(\diamondsuit_{=1}\mathbf{true}) ,$$

which says "for each event in $(1, 2)$ from now, there is a corresponding event that is exactly 1 time unit earlier", distinguishes $\mathcal{C}_m$ and $\mathcal{D}_m$ for any $m \in \mathbb{N}$ (when evaluated at position 0). □

3.2.3 Counting Modalities

The modality $C_n(x, X)$ asserts that X holds at least at n points in the open interval $(x, x+1)$. The modalities C_n for $n \geq 2$ are called *counting modalities*. It is well-known that these modalities are not expressible in MTL over $\mathbb{R}_{\geq 0}$-flows [HR07]. For this reason, they (and variants thereof) are often used to separate the expressiveness of various metric logics (cf., e.g., [BCM05, PD06, PS11]). For example, the following property

- P holds at an event at time y in the future
- Q holds at an event at time $y' > y$
- R holds at an event at time $y'' > y' > y$
- Both the Q-event and the R-event are within $(1, 2)$ from the P-event

can be expressed as the $\mathsf{FO}[<, +1]$ formula

$$\vartheta_{pqr}(x) = \exists y \Big(x < y \wedge P(y) \wedge \exists y' \Big(y < y' \wedge d(y, y') > 1 \wedge d(y, y') < 2 \wedge Q(y') \\ \wedge \exists y'' \left(y' < y'' \wedge d(y, y'') > 1 \wedge d(y, y'') < 2 \wedge R(y'') \right) \Big) \Big),$$

yet it has no equivalent in MTL over $\mathbb{R}_{\geq 0}$-timed words [PS11]. The difficulty here is that while we can easily write 'there is a Q-event within $(1, 2)$ from a P-event in the future' as $\Diamond(P \wedge \Diamond_{(1,2)} Q)$, it is not possible to express 'there is a R-event after the Q-event' and 'that R-event is within $(1, 2)$ from the P-event' at the same time in MTL. Indeed, it was shown recently that in the continuous semantics, MTL extended with counting modalities and their past counterparts (which we denote by $\mathsf{MTL}[\{C_n, \overleftarrow{C}_n\}_{n=2}^{\infty}]$) is expressively complete for $\mathsf{FO}[<, +1]$ [Hun13]. In other words, counting modalities fill exactly the expressiveness gap between MTL and $\mathsf{FO}[<, +1]$ in the continuous semantics. However, they add no expressiveness to MTL in the time-bounded setting. To see this, observe that the following formula is equivalent to $\vartheta_{pqr}(x)$ over $[0, N)$-timed words (we make use of the formulas in Φ_{int} defined in

Section 3.2.1 on page 29):

$$\Diamond\Bigg(\bigvee_{0\le i\le N-1}\Big(P\wedge\varphi_{i,i+1}\wedge\Big(\underbrace{\Diamond_{>1}\big(Q\wedge\Diamond(R\wedge\varphi_{i+1,i+2})\big)}_{\text{Case (i)}}\vee\underbrace{\Diamond_{<2}\big(R\wedge\varphi_{i+2,i+3}\wedge\Diamond(Q\wedge\varphi_{i+2,i+3})\big)}_{\text{Case (ii)}}\vee\underbrace{\big(\Diamond_{>1}(Q\wedge\varphi_{i+1,i+2})\wedge\Diamond_{<2}(R\wedge\varphi_{i+2,i+3})\big)}_{\text{Case (iii)}}\Big)\Big)\Bigg).$$

The three cases that correspond to the subformulas are illustrated in Figure 3.4 where time is measured relative to the very first event (with timestamp 0). Intuitively, we use the 'integer boundaries' as an alternative distance measure and thus ensure that both the Q-event and the R-event are within $(1, 2)$ from the P-event.

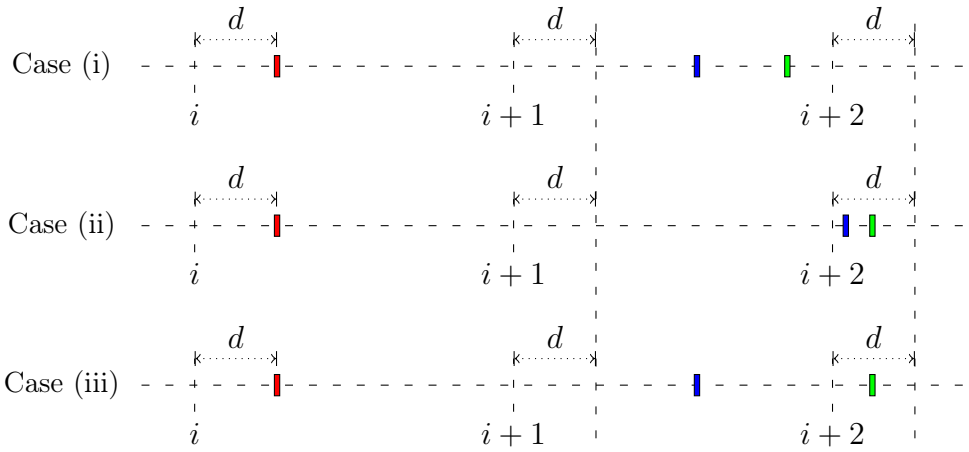


Figure 3.4: Counting modalities is expressible in MTL over $[0, N)$-timed words. The red, blue and green boxes represent P-events, Q-events and R-events respectively.

The same idea can readily be generalised to handle counting modalities and their past counterparts. We therefore have the following proposition.

Proposition 3.2.4. MTL *is expressively complete for* $\mathsf{MTL}[\{C_n, \overleftarrow{C}_n\}_{n=2}^{\infty}]$ *over* $[0, N)$*-timed words.*

3.2.4 Non-Local Properties: One Reference Point

Proposition 3.2.4 shows that a part of the expressiveness hierarchy of metric logics over $\mathbb{R}_{\ge 0}$-timed words collapses in the time-bounded setting. Nonetheless, MTL is

still not expressive enough to capture $\mathsf{FO}[<,+1]$. Recall that another feature of the continuous interpretation of $\mathsf{MTL}_{\mathsf{fut}}$ used in the proof in [ORW09] is that $\Diamond_{=k}\varphi$ holds at t *iff* φ holds at $t+k$. Suppose that we want to specify the following property over $\mathbf{P} = \{P, Q\}$ for some positive integer a (let the current instant be t_1):

- There is an event at time $t_2 > t_1 + a$ where Q holds
- P holds at all events in $(t_1 + a, t_2)$.

In the continuous semantics, the property can easily be expressed as the following $\mathsf{MTL}_{\mathsf{fut}}$ formula

$$\varphi_{cont1} = \Diamond_{=a}\big((P \vee P_\epsilon)\,\mathcal{U}\,Q\big)$$

over flows of the form f^ρ (over $\Sigma_{\mathbf{P}'}$ where $\mathbf{P}' = \mathbf{P} \cup \{P_\epsilon\}$). See Figure 3.5 for an example where the formula φ_{cont1} holds at t_1 in the continuous semantics.

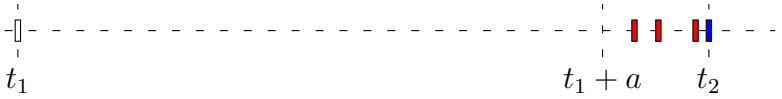


Figure 3.5: φ_{cont1} holds at t_1 in the continuous semantics. The red boxes denote P-events whereas the blue boxes denote Q-events.

In essence, when the current instant is t_1, the continuous interpretation of MTL allows one to speak of events 'around' $t_1 + a$ regardless of whether there is an actual event at $t_1 + a$. As we will show in a moment, it is not possible to do the same with the pointwise interpretation of MTL when there is no event at $t_1 + a$. To remedy this issue within the pointwise semantic framework, we introduce a relatively simple family of modalities $\mathcal{B}_I^{\rightarrow}$ ('Beginning') and their past versions $\mathcal{B}_I^{\leftarrow}$. They can be used to reference to the *first* (earliest or latest, respectively) event in a given interval. More precisely, we define the modality that asserts "X holds at the first event in (a, b) relative to the current instant" as the following $\mathsf{FO}[<,+1]$ formula:

$$\begin{aligned}\mathcal{B}^{\rightarrow}_{(a,b)}(x, X) = \exists x' \Big(&x < x' \wedge d(x, x') > a \wedge d(x, x') < b \wedge X(x') \\ &\wedge \nexists x'' \left(x < x'' \wedge x'' < x' \wedge d(x, x'') > a\right)\Big).\end{aligned}$$

The property above can now be written as $\mathcal{B}^{\rightarrow}_{(a,\infty)}\big(Q \vee (P\,\mathcal{U}\,Q)\big)$ in the pointwise semantics. We refer to the extension of MTL with $\mathcal{B}_I^{\rightarrow}, \mathcal{B}_I^{\leftarrow}$ as $\mathsf{MTL}[\mathcal{B}^{\leftrightarrows}]$.[3]

[3]Readers may find the modalities $\mathcal{B}_I^{\rightarrow}$ similar to the modalities $\triangleright_I$ in *Event-Clock Logic* [HRS98]. The difference is that the formula $\mathcal{B}_I^{\rightarrow}\varphi$ requires φ to hold at the *first* event in I, whereas the formula $\triangleright_I\varphi$ requires (i) φ to hold at *some* event in I and that (ii) φ does not hold from the current instant to the time of that event.

The following proposition states that this extension is indeed non-trivial.

Proposition 3.2.5. $\mathsf{MTL}[\mathcal{B}^{\leftrightarrows}]$ *is strictly more expressive than* MTL *over* $[0, N)$*-timed words.*

Proof. The proof we give here is inspired by a proof in [PS11, Section 5]. Given $m \in \mathbb{N}$, we describe models $\mathcal{E}_m$ and $\mathcal{F}_m$ that are indistinguishable by MTL formulas of modal depth $\leq m$ but distinguishing in $\mathsf{MTL}[\mathcal{B}^{\leftrightarrows}]$.

We first describe $\mathcal{F}_m$. Let $g = \frac{1}{2m+6}$ and pick $\varepsilon < \frac{g}{\frac{1}{g}-1}$. The first event (at time 0) satisfies $\neg P \wedge \neg Q$. Then, a sequence of overlapping segments (arranged as described below) starts at time $\frac{0.5}{2m+5}$. See Figure 3.6 for an illustration of a segment. Each segment consists of an event satisfying $P \wedge \neg Q$ and an event satisfying $\neg P \wedge Q$. Here for ease of presentation we will simply refer to them as P-events and Q-events respectively. If the P-event in the i^{th} segment is at time t, then its Q-event is at time $t + 2g + \frac{1}{2}\varepsilon$. All P-events in neighbouring segments are separated by $g - \frac{g}{\frac{1}{g}-1}$. We put a total of $4m + 12$ segments.

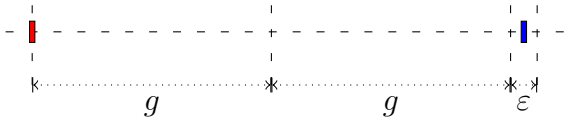


Figure 3.6: A single segment in $\mathcal{F}_m$. The red box denotes a P-event and the blue box denotes a Q-event.

$\mathcal{E}_m$ is almost identical to $\mathcal{F}_m$ except the $(3m+9)^{th}$ segment. Let this segment start at t_{3m+9}. In $\mathcal{E}_m$, we move the corresponding Q-event to $t + 2g - \frac{1}{2}\varepsilon$ (see Figure 3.7). Note that there are P-events at time 0.5 in both models (in their $(m+4)^{th}$ segment).

The only difference in two models is a pair of Q-events. We denote this pair of events by x and y respectively and write their corresponding timestamps as t_x and t_y (see Figure 3.7). It is easy to verify that no two events are separated by an integer distance. We say a configuration (i, j) is *identical* if $i = j$. For $i \geq 1$, we denote by $seg(i)$ the segment that the i^{th} event belongs to, and we write $P(i)$ if the i^{th} event is a P-event and $Q(i)$ if its a Q-event.

Proposition 3.2.6. Duplicator *has a winning strategy for* m*-round* MTL *EF game on* $\mathcal{E}_m$ *and* $\mathcal{F}_m$ *that starts from* $(0, 0)$*. In particular, she has a winning strategy such that for each round* $0 \leq r \leq m$*, the* i_r^{th} *event in* $\mathcal{E}_m$ *and the* j_r^{th} *event in* $\mathcal{F}_m$ *satisfy the same set of propositions and*

- *if* $i_r \neq j_r$*, then*

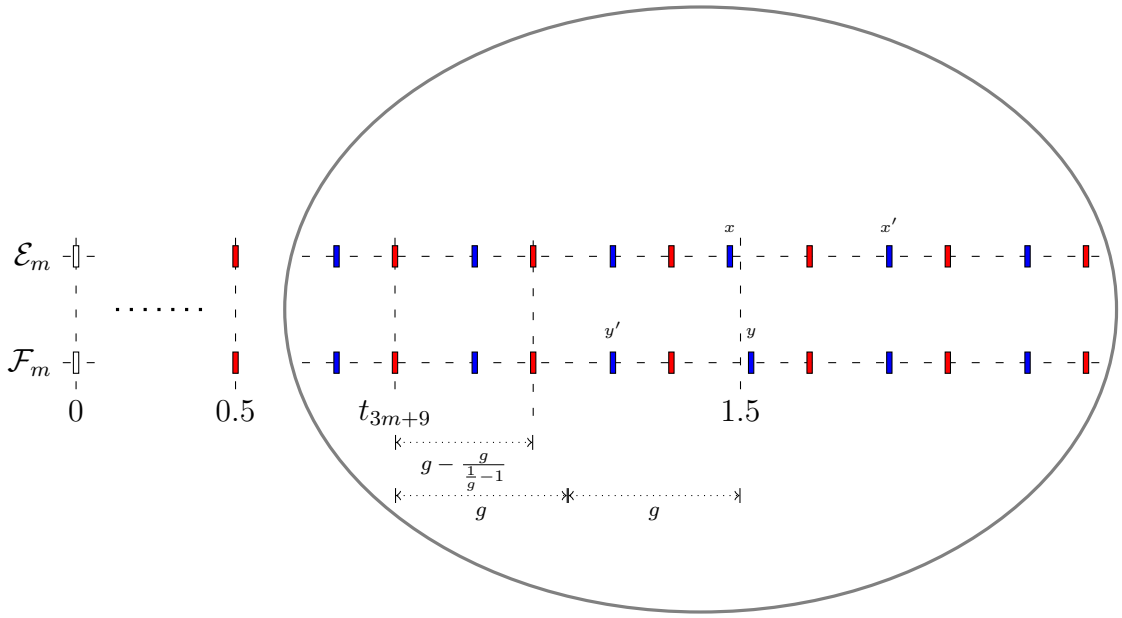


Figure 3.7: A close-up near the $(3m+9)^{th}$-segments in $\mathcal{E}_m$ and $\mathcal{F}_m$.

- $seg(i_r) - seg(j_r) < r$
- $(m{+}1{-}r) < seg(i_r), seg(j_r) < (m{+}5{+}r)$ *or* $(3m{+}8{-}r) < seg(i_r), seg(j_r) < (3m+12+r)$.

We prove the proposition by induction on r. The idea is to try to make the resulting configurations identical. When this is not possible *Duplicator* simply imitates what *Spoiler* does.

- *Base step.* The proposition holds trivially for $(i_0, j_0) = (0, 0)$.
- *Induction step.* Suppose that the claim holds for $r < m$. We prove it also holds for $r + 1$.
  - $(i_r, j_r) = (0, 0)$:
    Duplicator can always make (i_{r+1}, j_{r+1}) identical.
  - $(i_r, j_r) \neq (0, 0)$ is identical:
    Duplicator tries to make (i'_r, j'_r) identical. This may only fail when
    * $P(i_r)$, $P(j_r)$ and $seg(i_r) = seg(j_r) = m + 4$.
    * $Q(i_r)$, $Q(j_r)$ and $seg(i_r) = seg(j_r) = 3m + 9$, i.e., x and y.

    In these cases, *Duplicator* chooses another event in a neighbouring segment that minimises $|seg(i'_r) - seg(j'_r)|$. For example, if (i_r, j_r) corresponds to x and y and *Spoiler* chooses j'_r such that $P(j'_r)$ and $seg(j'_r) = m + 4$ in a

$\mathcal{S}_{(1,\infty)}$-move, *Duplicator* chooses i'_r with $seg(i'_r) = m + 3$. If *Spoiler* then plays $\diamondsuit$-part, the resulting configuration (i_{r+1}, j_{r+1}) will clearly satisfy the claim. If she plays $\mathcal{S}$-part, *Duplicator* makes (i''_r, j''_r) identical whenever possible. Otherwise she chooses the appropriate event that minimises $|seg(i''_r) - seg(j''_r)|$. For instance, if $Q(i''_r)$ and $seg(i''_r) = m + 1$, *Duplicator* chooses j''_r such that $Q(j''_r)$ and $seg(j''_r) = m + 2$.

– (i_r, j_r) is not identical:
Duplicator tries to make (i'_r, j'_r) identical. If this is not possible, then *Duplicator* chooses an event that minimises $|seg(i'_r) - seg(j'_r)|$. For example, consider $seg(i_r) = m + 4$, $seg(j_r) = m + 3$ such that $P(i_r)$ and $P(j_r)$, and *Spoiler* chooses x in an $\mathcal{U}_{(0,1)}$-move. In this case, *Duplicator* cannot choose y' but the first Q-event that happens before y'. *Duplicator* responds to $\mathcal{U}$-parts and $\mathcal{S}$-parts in similar ways as before. It is easy to see that the claim holds.

Proposition 3.2.5 now follows from Proposition 3.2.6, the MTL EF Theorem, and the fact that $\mathcal{E}_m \models \diamondsuit(P \wedge \mathcal{B}^{\rightarrow}_{(1,2)}P)$ but $\mathcal{F}_m \not\models \diamondsuit(P \wedge \mathcal{B}^{\rightarrow}_{(1,2)}P)$. □

3.2.5 Non-Local Properties: Two Reference Points

Adding modalities $\mathcal{B}^{\rightarrow}_I, \mathcal{B}^{\leftarrow}_I$ to MTL allows one to specify properties with respect to a distant time point even when there is no event at that point. However, the following proposition shows that this is still not enough for expressive completeness.

Proposition 3.2.7. $\mathsf{FO}[<, +1]$ *is strictly more expressive than* $\mathsf{MTL}[\mathcal{B}^{\leftrightarrows}]$ *over* $[0, N)$*-timed words.*

Proof. This is similar to a proof in [PD06, Section 7]. Given $m \in \mathbb{N}$, we construct two models as follows. Let

$$\mathcal{G}_m = (\emptyset, 0)(\emptyset, \frac{0.5}{2m+3})(\emptyset, \frac{1.5}{2m+3}) \dots (\emptyset, 1 - \frac{0.5}{2m+3})$$
$$(\emptyset, 1 + \frac{0.5}{2m+2})(\emptyset, 1 + \frac{1.5}{2m+2}) \dots\dots (\emptyset, 2 - \frac{0.5}{2m+2}).$$

$\mathcal{H}_m$ is constructed as $\mathcal{G}_m$ except that the event at time $\frac{m+1.5}{2m+3}$ is missing.

Figure 3.8 illustrates the models for the case $m = 2$ where white boxes represent events at which no monadic predicate holds. Observe that no two events are separated by an integer distance. We say that a configuration (i, j) is *synchronised* if they

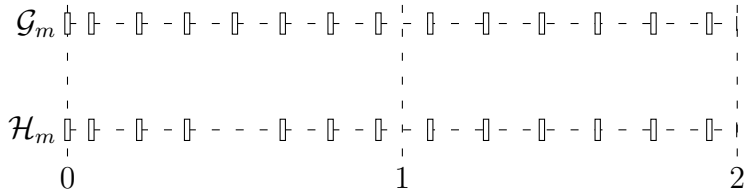


Figure 3.8: Models $\mathcal{G}_m$ and $\mathcal{H}_m$ for $m = 2$.

correspond to events with the same timestamp. Here we extend MTL EF games with the following moves to obtain $\mathsf{MTL}[\mathcal{B}^{\leftrightarrows}]$ **EF *games***.

- $\mathcal{B}_I^{\rightarrow}$*-move*: *Spoiler* chooses one of the two timed words (say ρ) and picks i'_r such that (i) $\tau_{i'_r} - \tau_{i_r} \in I$ in ρ and (ii) there is no position $i' < i'_r$ in ρ such that $\tau_{i'} - \tau_{i_r} \in I$. *Duplicator* must choose a position j'_r in ρ' such that j'_r is the first position in I relative to j_r in ρ'. If she cannot find such a position then *Spoiler* wins the game.

- $\mathcal{B}_I^{\leftarrow}$*-move*: Defined symmetrically.

Theorem 3.2.8 ($\mathsf{MTL}[\mathcal{B}^{\leftrightarrows}]$ EF Theorem)**.** *For (finite) timed words ρ, ρ' and an $\mathsf{MTL}[\mathcal{B}^{\leftrightarrows}]$ formula φ of modal depth $\leq m$, if* Duplicator *has a winning strategy for the m-round $\mathsf{MTL}[\mathcal{B}^{\leftrightarrows}]$ EF game on ρ, ρ' with $(i_0, j_0) = (0, 0)$, then*

$$\rho \models \varphi \iff \rho' \models \varphi\,.$$

Proposition 3.2.9. Duplicator *has a winning strategy for m-round $\mathsf{MTL}[\mathcal{B}^{\leftrightarrows}]$ EF game on $\mathcal{G}_m$ and $\mathcal{H}_m$ that starts from $(0, 0)$. In particular, she has a winning strategy such that for each round $0 \leq r \leq m$, the i_r^{th} event in $\mathcal{G}_m$ and the j_r^{th} event in $\mathcal{H}_m$ satisfy the same set of propositions and*

- *if (i_r, j_r) is not synchronised, then*
  - $|i_r - j_r| = 1$
  - $(m + 2 - r) < i_r, j_r < (m + 4 + r)$ *or* $(3m + 5 - r) < i_r, j_r < (3m + 6 + r)$.

We prove the proposition by induction on r. The idea, again, is to try to make the resulting configurations identical.

- *Base step.* The proposition holds trivially for $(i_0, j_0) = (0, 0)$.

- *Induction step.* Suppose that the claim holds for $r < m$. We prove it also holds for $r + 1$.

– $(i_r, j_r) = (0, 0)$:
Duplicator tries to make (i'_r, j'_r) synchronised. If *Spoiler* chooses $i'_r = m+3$, *Duplicator* chooses $j'_r = m + 2$.

– $(i_r, j_r) \neq (0, 0)$ is synchronised:
Duplicator tries to make (i'_r, j'_r) synchronised. If this is not possible then *Duplicator* chooses the event that minimises $|i'_r - j'_r|$. It is easy to see that the resulting configuration (i_{r+1}, j_{r+1}) satisfies the claim regardless of how *Spoiler* plays.

– (i_r, j_r) is not synchronised:
The strategy of *Duplicator* is same as the case above.

Proposition 3.2.7 now follows from Proposition 3.2.9, Theorem 3.2.8, and the fact that the $\mathsf{FO}[<, +1]$ formula

$$\exists x' \Big(d(x, x') > 1 \wedge d(x, x') < 2 \wedge \exists x'' \Big(x' < x'' \wedge \nexists y' \, (x' < y' \wedge y' < x'') \\ \wedge \nexists y'' \, \big(d(x', y'') < 1 \wedge d(x'', y'') > 1 \big) \Big) \Big)$$

distinguishes $\mathcal{G}_m$ and $\mathcal{H}_m$ for any $m \in \mathbb{N}$ (when evaluated at position 0). This formula asserts that there is a pair of neighbouring events in $(1, 2)$ such that there is no event between them if they are both mapped to exactly one time unit earlier. □

One way to understand why $\mathsf{MTL}[\mathcal{B}^{\leftrightarrows}]$ is still less expressive than $\mathsf{FO}[<, +1]$ is to consider the arity of modalities.[4] Let the current instant be t_1. Suppose that we want to specify the following property for some positive integers a and c, $a > c$:

- There is an event at $t_2 > t_1 + a$ where Q holds
- P holds at all events in $\big(t_1 + c, t_1 + c + (t_2 - t_1 - a)\big)$.

See Figure 3.9 for an example. In the continuous semantics, this property can be expressed as the following simple formula over flows of the form f^ρ:

$$\varphi_{cont2} = \big(\Diamond_{=c} (P \vee P_\epsilon) \big) \, \mathcal{U} \, (\Diamond_{=a} Q) \, .$$

Observe how this formula (effectively) talks about events around two time points: $t_1 + c$ and $t_1 + a$. In the same vein, the following formula can be used to distinguish

[4] We remark that a closely related yet different property is used in [LW08] to show that one-clock alternating timed automata and timed automata are expressively incomparable.

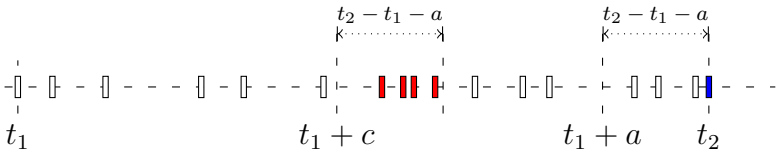


Figure 3.9: φ_{cont2} holds at t_1 in the continuous semantics. The red boxes denote P-events and the blue boxes denote Q-events.

$\mathcal{G}_m$ and $\mathcal{H}_m$ (defined in the proof of Proposition 3.2.7) in the continuous semantics:

$$\varphi_{cont3} = \Diamond_{(1,2)}\big(\neg P_\epsilon \wedge (\diamondsuit_{=1} P_\epsilon)\,\mathcal{U}\,(\neg P_\epsilon)\big)\,.$$

In the next section, we propose new modalities that add this ability to MTL in the pointwise semantics. We then show that this ability is exactly the 'missing piece' of expressiveness.

3.3 New Modalities

We introduce a family of modalities which can be understood as generalisations of the usual 'Until' and 'Since' modalities. Roughly speaking, the definition of these new modalities closely mimic the meaning of formulas of the form $(\Diamond_{=k_1}\varphi_1)\,\mathcal{U}_{<k_3}\,(\Diamond_{=k_2}\varphi_2)$ or $(\Diamond_{=k_1}\varphi_1)\,\mathcal{S}_{<k_3}\,(\Diamond_{=k_2}\varphi_2)$ in the continuous semantics.

3.3.1 Generalised 'Until' and 'Since'

Let $I \subseteq (0,\infty)$ be an interval with endpoints in $\mathbb{N} \cup \{\infty\}$ and $c \in \mathbb{N}$, $c < \inf(I)$. The formula $\varphi_1\,\mathfrak{U}^c_I\,\varphi_2$ (using infix notation), when imposed at t_1, asserts that

- There is an event at t_2 where φ_2 holds and $t_2 - t_1 \in I$
- φ_1 holds at all events in the open interval $\Big(t_1 + c, t_1 + c + \big(t_2 - \big(t_1 + \inf(I)\big)\big)\Big)$.

For example, the formula $P\,\mathfrak{U}^c_{(a,\infty)}\,Q$ (which is 'equivalent' to φ_{cont2} when the latter is interpreted over flows of the form f^ρ) holds at time t_1 in Figure 3.9. Formally, for $I = (a,b) \subseteq (0,\infty)$, $a \in \mathbb{N}$, $b \in \mathbb{N} \cup \{\infty\}$ and $c \in \mathbb{N}$ such that $c \leq a$, we define the

generalised 'Until' modality $\mathfrak{U}^c_{(a,b)}$ by the following $\mathsf{FO}[<,+1]$ formula:

$$\mathfrak{U}^c_{(a,b)}(x, X_1, X_2) = \exists x' \Big(x < x' \wedge d(x,x') > a \wedge d(x,x') < b \wedge X_2(x') \\ \wedge \forall x'' \big(x < x'' \wedge d(x,x'') > c \wedge x'' < x' \\ \wedge d(x',x'') > (a-c) \implies X_1(x'') \big) \Big).$$

Symmetrically, we define the ***generalised 'Since'*** modality $\mathfrak{S}^c_{(a,b)}$ as

$$\mathfrak{S}^c_{(a,b)}(x, X_1, X_2) = \exists x' \Big(x' < x \wedge d(x,x') > a \wedge d(x,x') < b \wedge X_2(x') \\ \wedge \forall x'' \big(x'' < x \wedge d(x,x'') > c \wedge x' < x'' \\ \wedge d(x',x'') > (a-c) \implies X_1(x'') \big) \Big).$$

We also define the modalities for $I \subseteq (0,\infty)$ being a half-open interval or a closed interval in the expected way. We will refer to the logic obtained by adding these modalities to MTL as $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$. Note that the usual 'Until' and 'Since' modalities can be written in terms of the generalised modalities. For instance,

$$\varphi_1 \,\mathcal{U}_{(a,b)}\, \varphi_2 = \varphi_1 \,\mathfrak{U}^a_{(a,b)}\, \varphi_2 \wedge \neg\big(\mathbf{true}\,\mathfrak{U}^0_{(0,a]}\,(\neg\varphi_1)\big).$$

3.3.2 More Liberal Bounds

In defining modalities $\mathfrak{U}^c_{(a,b)}$ and $\mathfrak{S}^c_{(a,b)}$ in the last subsection we stressed that $c \leq a$. We now show that more liberal uses of bounds (constraining intervals and superscript 'c') are merely syntactic sugars, and we therefore allow them in the sequel. For instance, suppose that we want to to assert the following property (which translates to $\big(\Diamond_{=10}(P \vee P_\epsilon)\big)\,\mathcal{U}_{<3}\,(\Diamond_{=2}Q)$ in the continuous semantics) at t_1:

- There is an event at t_2 where φ_2 holds and $t_2 - t_1 \in (2,5)$
- φ_1 holds at all events in $\big(t_1+10, t_1+10+(t_2-t_1-2)\big)$.

This can be expressed in $\mathsf{FO}[<,+1]$ as

$$\exists x' \Big(x < x' \wedge d(x,x') > 2 \wedge d(x,x') < 5 \wedge X_2(x') \\ \wedge \forall x'' \big(x < x'' \wedge d(x,x'') > 10 \wedge d(x',x'') < 8 \implies X_1(x'') \big) \Big).$$

Whilst we could have defined a modality $\mathfrak{U}^{10}_{(2,5)}(x, X_1, X_2)$ by this formula, this is not necessary as the formula is indeed equivalent to

$$\Diamond_{(2,5)}\varphi_2 \wedge \neg\Big((\neg\varphi_2)\,\mathfrak{U}^{2}_{(10,13)}\,\big(\neg\varphi_1 \wedge \neg(\diamondsuit_{=8}\varphi_2)\big)\Big)$$

if we substitute φ_1, φ_2 for X_1, X_2.

In the continuous semantics we can also write formulas like $(\diamondsuit_{=k_1}\varphi_1)\mathcal{U}_{<k_3}(\Diamond_{=k_2}\varphi_2)$. We now generalise the idea above to handle these cases.

Proposition 3.3.1. *Let the current instant be t_1. The property (and its past counterpart):*

- *There is an event at t_2 where φ_2 holds and $t_2 - t_1 \in I$*
- *φ_1 holds at all events in $\Big(t_1 + c, t_1 + c + \big(t_2 - \big(t_1 + \inf(I)\big)\big)\Big)$*

where $I \subseteq (-\infty, \infty)$, $\inf(I) \in \mathbb{Z}$, $\sup(I) \in \mathbb{Z} \cup \{\infty\}$ and $c \in \mathbb{Z}$, can be expressed with the modalities defined in Section 3.3.1.

Proof. Without loss of generality, we shall only focus on expressing the future version of the property for the case of I being an open interval (we will write them as formulas of the form $\varphi_1\,\mathfrak{U}^{c}_{(a,b)}\,\varphi_2$). To ease the presentation, we will use the following convention in all the illustrations given in this proof: the red boxes denote φ_1-events, blue boxes denote φ_2-events and white boxes denote events where neither φ_1 nor φ_2 hold. We prove the claim in each of the following cases:

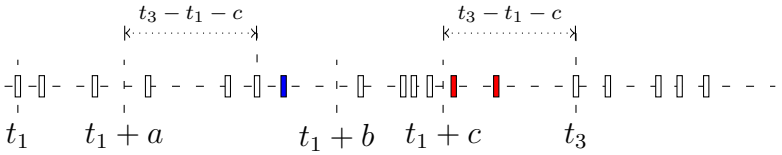
- *$a \geq 0$ and $0 \leq c \leq a$*: Simply use the modalities defined in Section 3.3.1.
- *$a \geq 0$ and $c > a$*: $\varphi_1\,\mathfrak{U}^{c}_{(a,b)}\,\varphi_2$ does not hold at t_1 if and only if one of the following holds at t_1:
  - *There is no φ_2-event in $(t_1 + a, t_1 + b)$*: This can be enforced by

    $$\neg(\Diamond_{(a,b)}\varphi_2)\,.$$

  - *$\neg\varphi_1$ holds at an event at $t_3 \in \big(t_1 + c, t_1 + c + (b - a)\big)$ and there is no φ_2-event in $(t_1 + a, t_1 + a + (t_3 - t_1 - c)]$*: This can be enforced by

    $$(\neg\varphi_2)\,\mathfrak{U}^{a}_{\left(c,c+(b-a)\right)}\,\big(\neg\varphi_1 \wedge \underbrace{\neg(\diamondsuit_{=(c-a)}\varphi_2)}_{\psi}\big)\,.$$

We need the subformula ψ to ensure that there is no φ_2-event at $t_1 + a + (t_3 - t_1 - c)$ (see below for an illustration).



The desired formula is the conjunction of the negations of these two formulas.

- $a \geq 0$ *and* $c < 0$: Let t_2 be the first time instant in $(t_1 + a, t_1 + b)$ where there is a φ_2-event. Consider the following subcases:

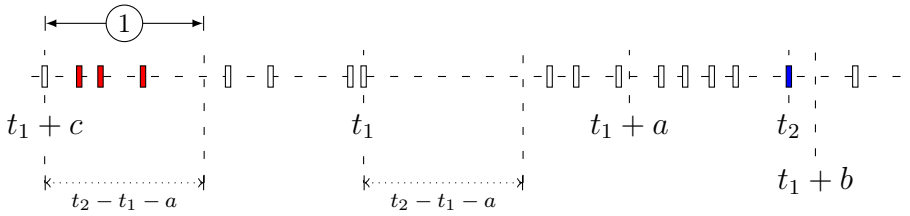
  – *There is no event in* $\big(t_1, t_1 + (t_2 - t_1 - a)\big)$: This can be enforced by

  $$\varphi = \textbf{false}\ \mathfrak{U}^0_{(a,b)}\ \varphi_2\,.$$

  Then we can enforce that φ_1 holds at all events in ① in the illustration below by

  $$\varphi' = (\neg\varphi_2)\ \mathfrak{U}^a_{(a,b)}\ \big(\varphi_2 \wedge \underbrace{(\varphi_1\ \mathfrak{S}^{a+|c|}_{(a,b)}\ \textbf{true})}_{\psi'}\big)\,.$$

  Note that the subformula ψ' must hold at t_2 if φ_1 holds at all events in ①. This is because, by assumption, there must be an event at t_1.



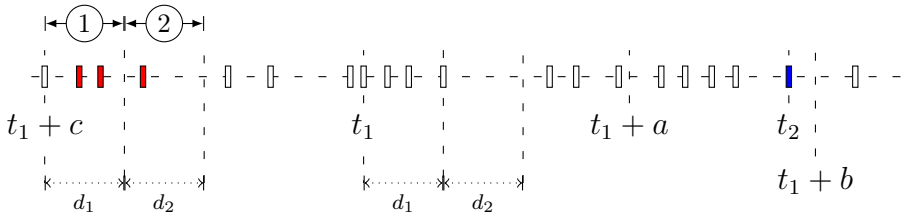
  – *There are events in* $\big(t_1, t_1 + (t_2 - t_1 - a)\big)$: In this case, φ' can only ensure that φ_1 holds at all events in ② (see the illustration below where $d_1 + d_2 = t_2 - t_1 - a$). We can enforce that φ_1 holds at all events in ① by

  $$\varphi'' = \psi''\ \mathcal{U}\ (\varphi \wedge \psi'')$$

  where

  $$\psi'' = \underbrace{(\varphi_1\ \mathfrak{S}^{|c|}_{(0,b-a)}\ \textbf{true})}_{\psi'''} \wedge \neg(\diamondminus_{=|c|}\neg\varphi_1)\,.$$

It is easy to see that φ must hold at the last event in $\big(t_1, t_1 + (t_2 - t_1 - a)\big)$. The correctness of our use of the subformula ψ''' here again depends on the fact that there is an event at t_1.



The desired formula is $\varphi' \wedge (\varphi \vee \varphi'')$.

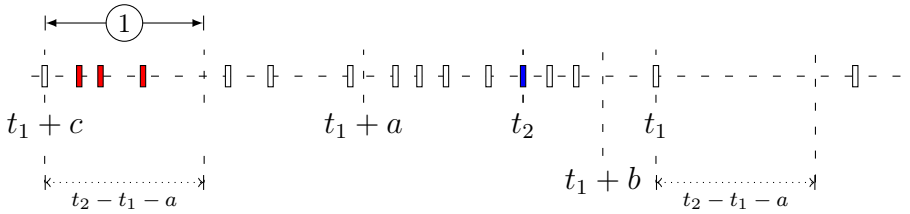
- $a < 0$ *and* $c \geq 0$: Without loss of generality we assume $a < b < 0$. Similar to the case $a \geq 0$ *and* $c > a$ above, the desired formula is

$$\diamondsuit_{(|b|,|a|)}\varphi_2 \wedge \neg\Big((\neg\varphi_2)\ \mathfrak{U}^{a}_{\left(c,c+(b-a)\right)}\ \big(\neg\varphi_1 \wedge \neg(\diamondsuit_{=(c-a)}\varphi_2)\big)\Big)\,.$$

- $a < 0$ *and* $c \leq a$: Without loss of generality we assume $a < b < 0$. Let t_2 be the first time instant in $(t_1 + a, t_1 + b)$ where there is a φ_2-event. Similar to the case $a \geq 0$ and $c < 0$ above, consider the following subcases:

  - *There is no event in* $\big(t_1, t_1 + (t_2 - t_1 - a)\big)$: We enforce that φ_1 holds at all events in (1) in the illustration below by

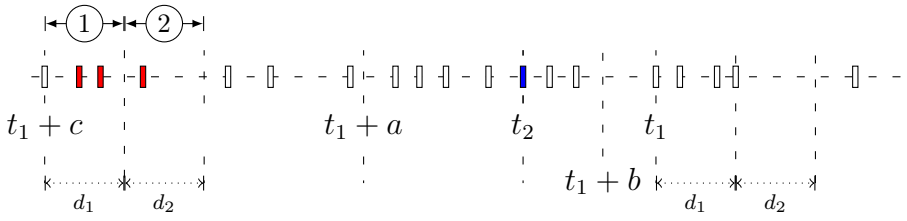
$$\varphi''' = \textbf{false}\ \mathfrak{U}^{0}_{(a,b)}\ \big(\varphi_2 \wedge (\varphi_1\ \mathfrak{S}^{a+|c|}_{(a,b)}\ \textbf{true})\big)\,.$$



  - *There are events in* $\big(t_1, t_1 + (t_2 - t_1 - a)\big)$: We enforce that φ_1 holds at all events in (1) and (2) in the illustration below (in which $d_1 + d_2 = t_2 - t_1 - a$) by

$$\diamondsuit_{(|b|,|a|)}\varphi_2 \wedge \big(\psi''\,\mathcal{U}\,(\varphi''' \wedge \psi'')\big)\,,$$

    where ψ'' is defined in the case $a \geq 0$ and $c < 0$ above.



The desired formula is the disjunction of these two formulas.

- $a < 0$ *and* $a < c < 0$: Without loss of generality we assume $a < b < 0$. The desired formula is identical to the formula in the case $a < 0$ and $c \geq 0$ above. □

We can now give an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula that distinguishes, in the pointwise semantics, the models $\mathcal{G}_m$ and $\mathcal{H}_m$ in Section 3.2.5 (on page 37):

$$\Diamond_{(1,2)}\big(\mathbf{true} \wedge (\mathbf{false}\, \mathfrak{U}^{-1}_{(0,\infty)}\, \mathbf{true})\big)\,.$$

This formula is 'equivalent' to the formula φ_{cont3} defined in Section 3.2.5, which distinguishes $\mathcal{G}_m$ and $\mathcal{H}_m$ in the continuous semantics.

3.4 The Translation

We give a translation from an arbitrary $\mathsf{FO}[<, +1]$ formula with one free variable into an equivalent $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula (over $[0, N)$-timed words). Our proof strategy is similar to that in [ORW09]: we convert a metric formula into a non-metric one, translate the formula into LTL, and then construct an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula equivalent to the original formula. The basis of these steps is a ***'stacking' bijection*** between $[0, N)$-timed words over $\Sigma_{\mathbf{P}}$ and a set of $[0, 1)$-timed words over a different yet closely related alphabet. Roughly speaking, since the time domain is bounded, we can encode the integer parts of timestamps with a bounded number of new monadic predicates. This enables us to work instead with 'stacked' $[0, 1)$-timed words, in which only the ordering of events are relevant.

3.4.1 Eliminating the Metric

Stacking bounded timed words. For each monadic predicate $P \in \mathbf{P}$, we introduce fresh monadic predicates P_i, $0 \leq i \leq N - 1$ and let the set of all these new monadic predicates be $\overline{\mathbf{P}}$. Intuitively, for $x \in [0, 1)$, $P_i(x)$ holds in a stacked $[0, 1)$-timed word iff P holds at time $i + x$ in the corresponding $[0, N)$-timed word, We also introduce

$\overline{\mathbf{Q}} = \{Q_i \mid 0 \leq i \leq N-1\}$. For $x \in [0,1)$, $Q_i(x)$ holds in a stacked $[0,1)$-timed word iff there is an event at time $i+x$ in the corresponding $[0,N)$-timed word, regardless of whether any $P \in \mathbf{P}$ holds there. Let $\vartheta_{event} = \forall x \left(\bigvee_{0\leq i\leq N-1} Q_i(x)\right) \wedge \forall x \left(\bigwedge_{0\leq i\leq N-1} \left(P_i(x) \implies Q_i(x)\right)\right)$ and $\vartheta_{init} = \exists x \left(\nexists x' \left(x' < x\right) \wedge Q_0(x)\right)$. There is an obvious 'stacking' bijection (indicated by overlining) between $[0,N)$-timed words over $\Sigma_{\mathbf{P}}$ and $[0,1)$-timed words over $\Sigma_{\overline{\mathbf{P}}\cup\overline{\mathbf{Q}}}$ satisfying $\vartheta_{event} \wedge \vartheta_{init}$. For a concrete example, the stacked counterpart of the $[0,2)$-timed word

$$\rho = (\{A\},0)(\{A,C\},0.3)(\{B\},1)(\{B,C\},1.5)$$

with $\mathbf{P} = \{A,B,C\}$ is the $[0,1)$-timed word:

$$\overline{\rho} = (\{Q_0,Q_1,A_0,B_1\},0)(\{Q_0,A_0,C_0\},0.3)(\{Q_1,B_1,C_1\},0.5)\,.$$

Stacking $\mathsf{FO}[<,+1]$ formulas. Let $\vartheta(x)$ be an $\mathsf{FO}[<,+1]$ formula with one free variable and in which each quantifier uses a fresh new variable. Without loss of generality, we assume that $\vartheta(x)$ contains only existential quantifiers (this can be achieved by syntactic rewriting). Replace the formula by

$$\big(Q_0(x) \wedge \vartheta[x/x]\big) \vee \big(Q_1(x) \wedge \vartheta[x+1/x]\big) \vee \ldots \vee \big(Q_{N-1}(x) \wedge \vartheta[x+(N-1)/x]\big)$$

where $\vartheta[e/x]$ denotes the formula obtained by substituting all free occurrences of x in ϑ by (an expression) e. Then, similarly, recursively replace every subformula $\exists x'\,\theta$ by

$$\exists x' \Big(\big(Q_0(x') \wedge \theta[x'/x']\big) \vee \ldots \vee \big(Q_{N-1}(x') \wedge \theta[x'+(N-1)/x']\big)\Big)\,.$$

Note that we do not actually have the $+k$ functions in the pointwise version $\mathsf{FO}[<,+1]$; they only serve as annotations here and will be removed later, e.g., $x'+k$ means that $Q_k(x')$ holds. We then carry out the following syntactic substitutions:

- For each inequality of the form $x_1 + k_1 < x_2 + k_2$, replace it with
  - $x_1 < x_2$ if $k_1 = k_2$
  - **true** if $k_1 < k_2$
  - **false** if $k_1 > k_2$
- For each distance formula, e.g., $d(x_1+k_1, x_2+k_2) < 2$, replace it with
  - **true** if $|k_1 - k_2| \leq 1$

- $x_2 < x_1$ if $k_2 - k_1 = 2$
- $x_1 < x_2$ if $k_1 - k_2 = 2$
- **false** if $|k_1 - k_2| > 2$

- Replace terms of the form $P(x_1 + k)$ with $P_k(x_1)$.

This gives a non-metric first-order formula $\overline{\vartheta}(x)$ over $\overline{\mathbf{P}} \cup \overline{\mathbf{Q}}$. Denote by $frac(t)$ the fractional part of a non-negative real t. It is not hard to see that for each $[0, N)$-timed word $\rho = (\sigma, \tau)$ over $\Sigma_{\mathbf{P}}$ and its stacked counterpart $\overline{\rho}$, the following holds:

- $\rho, t \models \vartheta(x)$ implies $\overline{\rho}, \overline{t} \models \overline{\vartheta}(x)$ where $\overline{t} = frac(t)$
- $\overline{\rho}, \overline{t} \models \overline{\vartheta}(x)$ implies there exists $t \in \rho$ with $frac(t) = \overline{t}$ s.t. $\rho, t \models \vartheta(x)$.

Moreover, if $\rho, t \models \vartheta(x)$, then the integer part of t indicates which disjunct in $\overline{\vartheta}(x)$ is satisfied when x is substituted with $\overline{t} = frac(t)$, and vice versa.

By Kamp's theorem [Kam68] (applied respectively on each $\vartheta[x + i/x]$), $\overline{\vartheta}(x)$ is equivalent to an LTL formula $\overline{\varphi}$ of the following form:

$$(Q_0 \wedge \overline{\varphi}_0) \vee (Q_1 \wedge \overline{\varphi}_1) \vee \ldots \vee (Q_{N-1} \wedge \overline{\varphi}_{N-1}) .$$

3.4.2 From Non-Metric to Metric

We construct inductively an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula ψ for each subformula $\overline{\psi}$ of $\overline{\varphi}_i$ (for some $i \in \{0, \ldots, N-1\}$). Note that we make heavy use of the formulas in Φ_{int} defined in Section 3.2.1 (on page 29).

- $\overline{\psi} = P_j$: Let

$$\psi = (\varphi_{0,1} \wedge \Diamond_{=j} P) \vee \ldots \vee (\varphi_{j,j+1} \wedge P) \vee \ldots \vee (\varphi_{N-1,N} \wedge \Diamond\!\!\!\!-_{=((N-1)-j)} P) .$$

- $\overline{\psi} = Q_j$: Similarly, let

$$\psi = (\varphi_{0,1} \wedge \Diamond_{=j} \mathbf{true}) \vee \ldots \vee (\varphi_{j,j+1} \wedge \mathbf{true}) \vee \ldots \vee (\varphi_{N-1,N} \wedge \Diamond\!\!\!\!-_{=((N-1)-j)} \mathbf{true}) .$$

- $\overline{\psi} = \overline{\psi}_1 \,\mathcal{U}\, \overline{\psi}_2$: Let $\psi^{j,k,l} = \psi_1 \,\mathfrak{U}^k_{(j,j+1)}\, (\psi_2 \wedge \varphi_{l,l+1})$. The desired formula is

$$\psi = \bigvee_{0 \le i \le N-1} \left(\varphi_{i,i+1} \wedge \bigvee_{\substack{-i \le j \le (N-1)-i \\ l = i+j}} \left(\bigwedge_{-i \le k \le (N-1)-i} \psi^{j,k,l} \right) \right) .$$

- $\overline{\psi} = \overline{\psi}_1 \,\mathcal{S}\, \overline{\psi}_2$: This is symmetric to the case for $\overline{\psi}_1 \,\mathcal{U}\, \overline{\psi}_2$.

The construction for other cases are as expected and therefore omitted.

Proposition 3.4.1. *Let $\overline{\psi}$ be a subformula of $\overline{\varphi}_i$ for some $i \in \{0, \ldots, N-1\}$. There is an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula ψ such that for any $[0, N)$-timed word ρ, $t \in \rho$ and $frac(t) = \bar{t} \in \overline{\rho}$, we have*

$$\overline{\rho}, \bar{t} \models \overline{\psi} \iff \rho, t \models \psi \,.$$

Proof. Induction on the structure of $\overline{\psi}$ and ψ, where the latter is constructed as described above.

- $\overline{\psi} = P_j$: Assume $\overline{\rho}, \bar{t} \models \overline{\psi}$. If $t = j + \bar{t}$, the disjunct $(\varphi_{j,j+1} \wedge P)$ of ψ clearly holds at t in ρ. If $t = j' + \bar{t}$ where $j' \neq j$, since there is a P-event at time $j + \bar{t}$ in ρ, the j'-th disjunct of ψ must hold at t in ρ. The proof for the other direction is similar.

- $\overline{\psi} = \overline{\psi}_1 \,\mathcal{U}\, \overline{\psi}_2$: Assume $\overline{\rho}, \bar{t} \models \overline{\psi}$ and let the witness be at $\overline{t'}$. By construction and the induction hypothesis, there is an event at $t' = l + \bar{t}$ in ρ for some $l \in \{0, \ldots, N-1\}$ such that $\rho, t' \models \psi_2$. Moreover, since we have $\overline{\rho}, \overline{t''} \models \overline{\psi_1}$ for all $\overline{t''}$, $\bar{t} < \overline{t''} < \overline{t'}$, we must have $\rho, t'' \models \psi_1$ for all $t'' \in \rho$ with $t'' = k' + \overline{t''}$ for some $\bar{t} < \overline{t''} < \overline{t'}$ and $0 \leq k' \leq N-1$. Now let $t = i + \bar{t}$ for some $i \in \{0, \ldots, N-1\}$ be a timestamp in ρ and let $j = l - i$. It is clear that $\rho, t \models \varphi_{i,i+1}$ and $\rho, t \models \bigwedge_{\substack{0 \leq k' \leq N-1 \\ k = k' - i}} \psi^{j,k,l}$, as required. For the other direction, let $t = i + \bar{t}$ for some $i \in \{0, \ldots, N-1\}$ and let $\rho, t \models \bigwedge_{-i \leq k \leq (N-1)-i} \psi^{j,k,l}$ for some $j \in \{-i, \ldots, (N-1) - i\}$ and $l = i + j$. It follows that there is a (minimal) $\overline{t'} > \bar{t}$ such that $\rho, l + \overline{t'} \models \psi_2$ and $\rho, k' + \overline{t''} \models \psi_1$ for all $t'' \in \rho$ with $t'' = k' + \overline{t''}$ for some $\bar{t} < \overline{t''} < \overline{t'}$ and $0 \leq k' \leq N-1$. The claim follows by construction and the induction hypothesis.

Other cases are trivial or symmetric to the above ones. □

Using the construction above, we obtain an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula φ_i for each $\overline{\varphi}_i$. Substitute them into $\overline{\varphi}$ and replace all remaining Q_i by $\varphi_{i,i+1}$ to obtain our final formula φ. We now claim that φ is equivalent to the original $\mathsf{FO}[<, +1]$ formula $\vartheta(x)$ over $[0, N)$-timed words.

Proposition 3.4.2. *For any $[0, N)$-timed word ρ and $t \in \rho$, we have*

$$\rho, t \models \varphi(x) \iff \rho, t \models \vartheta(x)\,.$$

Proof. Follows directly from Section 3.4.1 and Proposition 3.4.1. □

We are now ready to state the main result of this chapter.

Theorem 3.4.3. $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *is expressively complete for* $\mathsf{FO}[<, +1]$ *over $[0, N)$-timed words.*

3.5 Time-Bounded Verification

We claim that the ***timed-bounded satisfiability*** and ***time-bounded model-checking*** problems for $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ are EXPSPACE-complete in both the pointwise and continuous semantics.

Theorem 3.5.1. *The time-bounded satisfiability problem for* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *(in both the pointwise and continuous semantics) is* EXPSPACE*-complete.*

Proof. The claim follows easily from [ORW09]. To see this, note that for a given $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula, one can replace all subformulas of the form $\varphi_1\, \mathfrak{U}^c_{(a,b)}\, \varphi_2$ by the MTL formula

$$(\Diamond_{=c}\varphi_1)\, \mathcal{U}_{<b}\, (\Diamond_{=a}\varphi_2)$$

in the continuous semantics (this can incur at most a linear blow-up). Since for each $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula over timed words one can construct, in linear time, an 'equivalent' $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula over flow of the form f^ρ, it follows that the claim is also true in the pointwise semantics. However, we give a direct proof for the case of pointwise semantics along the lines of [ORW09] here (see Section 3.6 for a discussion on the practical implication). The idea is again based on stacking: for a given $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula φ, we can build an $\mathsf{LTL}_{\mathsf{fut}}$ formula $\overline{\varphi}$ (of exponential size) such that $\overline{\rho} \models \overline{\varphi}$ iff $\rho \models \varphi$ for all $[0, N)$-timed words ρ.

For each subformula ψ of φ and every $i \in \{0, \ldots, N\}$, we introduce a monadic predicate F_i^ψ. We then add suitable subformulas into $\overline{\varphi}$ to ensure that F_i^ψ holds at $\bar{t}$ in $\overline{\rho}$ iff ψ holds at $t = \bar{t} + i$ in ρ. The construction in [ORW09] readily carries over to our pointwise setting when the outermost operator of ψ is Boolean or a standard MTL modality. We now demonstrate how to handle the new modalities by an example. For

$i \leq N-4$ and a subformula $A\,\mathfrak{U}^1_{(2,3)}\,B$ of φ, we require (in $\overline{\varphi}$) the following formula to hold at every point in time:

$$F_i^{A\mathfrak{U}^1_{(2,3)}B} \iff \big((F^Q_{i+1} \implies F^A_{i+1})\,\mathcal{U}\,F^B_{i+2}\big) \vee \Big(\Box(F^Q_{i+1} \implies F^A_{i+1}) \wedge \diamondsuit\big(F^B_{i+3} \wedge \boxminus(F^Q_{i+2} \implies F^A_{i+2})\big)\Big).$$

We also add the $\mathsf{LTL}_{\mathsf{fut}}$ equivalents of ϑ_{event} and ϑ_{init} (defined in Section 3.4.1 on page 45) into $\overline{\varphi}$ as conjuncts. It is clear that $\overline{\varphi}$ is of size exponential in the size of φ. EXPSPACE-hardness follows from the corresponding result of Bounded-MTL (in the pointwise semantics) in [BMOW07]. □

Since the time-bounded model-checking problem and satisfiability problem are inter-reducible in both the pointwise and continuous semantics [Wil94, HRS98], we have the following theorem.

Theorem 3.5.2. *The time-bounded satisfiability problem for timed automata against* $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$ *(in both the pointwise and continuous semantics) is* EXPSPACE*-complete.*

3.6 Discussion

Our main result in this chapter is that over bounded timed words, MTL extended with our new modalities 'generalised Until' and 'generalised Since' ($\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$) is expressively complete for $\mathsf{FO}[<,+1]$. Moreover, the time-bounded satisfiability and model-checking problems for $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$ remain EXPSPACE-complete, same as that of MTL. Our results hold both when the logics are interpreted over timed words or flows, the two most popular models of real-time behaviour. We remark that the situation here is, in a certain sense, similar to LTL over general (possibly non-Dedekind complete) linear orders (e.g., the rationals): in this case, LTL can be made expressively complete (for $\mathsf{FO}[<]$) by adding the Stavi modalities [GHR94], yet the complexity of the satisfiability problem remains PSPACE-complete [Rab10]. Along the way we also obtain a strict hierarchy of metric temporal logics, based on their expressiveness over bounded timed words (see Figure 3.10 for an illustration where the arrows indicate 'strictly more expressive than' and the edges indicate 'equally expressive').

One drawback of the modalities $\mathfrak{U}^c_I$ and $\mathfrak{S}^c_I$ is that they are not very intuitive. However, as we proved that simpler versions of these modalities ($\mathcal{B}^{\rightarrow}_I$ and $\mathcal{B}^{\leftarrow}_I$) are strictly less expressive over bounded timed words, we believe it is unlikely that another

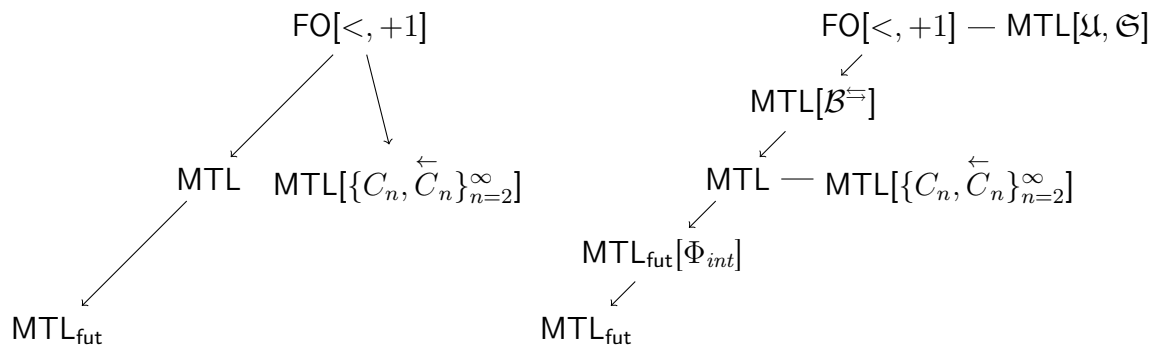


Figure 3.10: Summary of the expressiveness results of this chapter.

reasonable expressively complete extension of MTL, if one exists, could be much simpler than our extensions.

In this chapter we focused on expressive completeness, i.e., the question of whether for every first-order formula $\vartheta(x)$ (with a single free variable x), there is a temporal logic formula φ such that φ and $\vartheta(x)$ are equivalent at *every point in time*. For most practical purposes, it suffices to compare the relative expressiveness of logics with respect to initial equivalence, i.e., whether we can always find a formula equivalent to $\vartheta(x)$ at *time* 0. In this sense, $\mathsf{MTL}_{\mathsf{fut}}$ is as expressive as $\mathsf{FO}[<,+1]$ over $[0,N)$-flows [ORW09], analogous to Gabbay's classic result that $\mathsf{LTL}_{\mathsf{fut}}$ is as expressive as $\mathsf{FO}[<]$ over finite or infinite (untimed) words [GPSS80]. An interesting question is whether we can find a future metric temporal logic (i.e., all its modalities are defined by formulas whose truth values at any point depend only on the future) that is as expressive as $\mathsf{FO}[<,+1]$ over bounded timed words. We have shown that $\mathsf{MTL}_{\mathsf{fut}} \subsetneq \mathsf{MTL}$ over bounded timed words, but this does not rule out the possibility that more complex future modalities might suffice. For example, the models $\mathcal{C}_m$ and $\mathcal{D}_m$ defined in Section 3.2.2 (on page 31) can be distinguished by

$$\Box\big((\mathbf{false}\,\mathfrak{U}^1_{(0,1)}\,\mathbf{true}) \vee \neg\Diamond\mathbf{true}\big)$$

where $\mathfrak{U}^1_{(0,1)}$ can be defined as a stand-alone modality by a future $\mathsf{FO}[<,+1]$ formula.

The satisfiability and model-checking procedures for MTL in [ORW09] are based on the satisfiability procedure for LTL over flows in [Rey10]. While the satisfiability problem for LTL remains PSPACE-complete when interpreted over flows, very few implementations are currently available [FMDR13]. This is in contrast with the discrete case, where a number of highly-optimised tools (e.g., SPIN [Hol97]) are readily

available. A possible route to make use of these tools is to first reduce the problem to a discrete one (this is possible since we assume flows to be finitely variable). However, this may introduce some unnecessary complications. For example, the $\mathsf{FO}[<, +1]$ formula $\forall x \forall y \big(x < y \implies \exists z (x < z \wedge z < y)\big)$ is satisfied by any $[0, N)$-flow but not by any $[0, N)$-timed word. Our results, on the other hand, allow one to leverage these tools in a more direct manner. Whether this approach yields efficiency gains in practice can only be evaluated by implementation, which we leave as future work.

Chapter 4

Expressive Completeness over Unbounded Timed Words

Recall that the counting modality $C_2(x, X)$ asserts that X holds at at least two points in $(x, x+1)$. While the modality is not expressible in MTL, it is equivalent to the following MTL formula with *rational* endpoints:

$$\Diamond_{(0,\frac{1}{2})}(X \wedge \Diamond_{(0,\frac{1}{2})}X) \vee \Diamond_{(\frac{1}{2},1)}(X \wedge \Diamond\!\!\!-_{(0,\frac{1}{2})}X) \vee (\Diamond_{(0,\frac{1}{2})}X \wedge \Diamond_{(\frac{1}{2},1)}X)\,.$$

Indeed, MTL with rational endpoints is expressively complete for $\mathsf{FO}[<, +\mathbb{Q}]$ (the rational version of $\mathsf{FO}[<, +1]$) over $\mathbb{R}_{\geq 0}$-flows [HOW13]. Unfortunately, even with rational endpoints, MTL is still less expressive than $\mathsf{FO}[<, +1]$ in the pointwise semantics [PD06]. We show in this chapter that expressive completeness of MTL over (infinite) timed words can be recovered by adding (the rational versions of) the modalities 'generalised Until' ($\mathfrak{U}_I^c$) and 'generalised Since' ($\mathfrak{S}_I^c$)[1] we introduced in the last chapter.

Our presentation in this chapter closely follows [HOW13]. We first give a set of rewriting rules that 'extract' unbounded temporal operators from the scopes of bounded ones. Then we invoke Gabbay's separation theorem [GPSS80] to obtain a analogous syntactic separation result for $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$. Exploiting a normal form for $\mathsf{FO}[<, +1]$ in [GPSS80], we detail how to express any bounded $\mathsf{FO}[<, +\mathbb{Q}]$ formula in $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$. Finally, we combine these ideas to obtain our desired result.

[1] To simplify notations, we still refer to the logic (allowing rational endpoints) as $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$.

4.1 Syntactic Separation of $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$

In this section we present a series of logical equivalences that can be used to rewrite a given $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula into an equivalent formula in which no unbounded temporal operators occurs within the scope of a bounded temporal operator. Only the rules for open intervals are given, as the rules for other types of intervals are straightforward variants.

4.1.1 A Normal Form for $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$

We say an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula is in *normal form* if it satisfies the following.

(i) All occurrences of unbounded temporal operator are of the form $\mathcal{U}_{(0,\infty)}$, $\mathcal{S}_{(0,\infty)}$, $\Box_{(0,\infty)}$, $\boxminus_{(0,\infty)}$.

(ii) All other occurrences of temporal operators are of the form $\mathcal{U}_I$, $\mathcal{S}_I$, $\mathfrak{U}^c_I$, $\mathfrak{S}^c_I$ with bounded I.

(iii) Negation is only applied to monadic predicates or bounded temporal operators.

(iv) In any subformula of the form $\varphi_1 \mathcal{U}_I \varphi_2$, $\varphi_1 \mathcal{S}_I \varphi_2$, $\Diamond_I \varphi_2$, $\diamondminus_I \varphi_2$, $\varphi_1 \mathfrak{U}^c_I \varphi_2$, $\varphi_1 \mathfrak{S}^c_I \varphi_2$ where I is bounded, φ_1 is a disjunction of subformulas and φ_2 is a conjunction thereof.

We now describe how to rewrite a given formula into normal form. To satisfy (i) and (ii), apply the usual rules (e.g., $\Box_I \varphi \iff \neg \Diamond_I \neg \varphi$) and the rules:

$$\begin{aligned}
\varphi_1 \,\mathcal{U}_{(a,\infty)}\, \varphi_2 &\iff \varphi_1 \,\mathcal{U}\, \varphi_2 \wedge \Box_{(0,a]}(\varphi_1 \wedge \varphi_1 \,\mathcal{U}\, \varphi_2) \\
\varphi_1 \,\mathfrak{U}^c_{(a,\infty)}\, \varphi_2 &\iff \varphi_1 \,\mathfrak{U}^c_{(a,2a]}\, \varphi_2 \vee \left(\Diamond^w_{[0,c]} \big(\varphi_1 \,\mathcal{U}_{(a,\infty)}\, (\varphi_2 \vee \Diamond_{\leq a-c} \varphi_2) \big) \right).
\end{aligned}$$

To satisfy (iii), use the usual rules and the rule:

$$\neg(\varphi_1 \,\mathcal{U}\, \varphi_2) \iff \Box \neg \varphi_2 \vee \big(\neg \varphi_2 \,\mathcal{U}\, (\neg \varphi_2 \wedge \neg \varphi_1) \big).$$

For (iv), use the usual rules of Boolean algebra and the rules below:

$$\begin{aligned}
\phi \,\mathcal{U}_I\, (\varphi_1 \vee \varphi_2) &\iff (\phi \,\mathcal{U}_I\, \varphi_1) \vee (\phi \,\mathcal{U}_I\, \varphi_2) \\
(\varphi_1 \wedge \varphi_2) \,\mathcal{U}_I\, \phi &\iff (\varphi_1 \,\mathcal{U}_I\, \phi) \wedge (\varphi_2 \,\mathcal{U}_I\, \phi) \\
\phi \,\mathfrak{U}^c_I\, (\varphi_1 \vee \varphi_2) &\iff (\phi \,\mathfrak{U}^c_I\, \varphi_1) \vee (\phi \,\mathfrak{U}^c_I\, \varphi_2) \\
(\varphi_1 \wedge \varphi_2) \,\mathfrak{U}^c_I\, \phi &\iff (\varphi_1 \,\mathfrak{U}^c_I\, \phi) \wedge (\varphi_2 \,\mathfrak{U}^c_I\, \phi).
\end{aligned}$$

The rules for past temporal operators are as symmetric.

Proposition 4.1.1. *The following equivalence holds over infinite timed words:*

$$\varphi_1 \,\mathfrak{U}^c_{(a,\infty)}\, \varphi_2 \iff \varphi_1 \,\mathfrak{U}^c_{(a,2a]}\, \varphi_2 \vee \Big(\Diamond^w_{[0,c]}\big(\varphi_1 \,\mathcal{U}_{(a,\infty)}\, (\varphi_2 \vee \Diamond_{\leq a-c}\varphi_2)\big)\Big)\,.$$

Proof. Let the current position be i and the 'witness' position where φ_2 holds be w. Consider the following cases:

- $\tau_w \in (\tau_i + a, \tau_i + 2a]$: $\varphi_1 \,\mathfrak{U}^c_{(a,2a]}\, \varphi_2$ clearly holds.
- $\tau_w \in (\tau_i + 2a, \infty)$: Consider the following subcases:
  - φ_1 holds at all positions $j < w$ such that $\tau_j > \tau_i + c$: $\varphi_1 \,\mathcal{U}_{(a,\infty)}\, \varphi_2$ holds at the maximal position j' such that $\tau_{j'} \in [\tau_i, \tau_i + c]$.
  - φ_1 holds at all positions $j < w$ such that $\tau_j > \tau_i + c$ and $\tau_w - \tau_j > a - c$: By assumption, there is a position j' at which φ_1 does not hold and $\tau_w - \tau_{j'} \leq a - c$. Since $\tau_w > \tau_i + 2a$, we have $\tau_{j'} > \tau_i + a + c$. It follows that $\varphi_1 \,\mathcal{U}_{(a,\infty)}\, (\Diamond_{\leq a-c}\varphi_2)$ holds at the maximal position in $[\tau_i, \tau_i + c]$.

The other direction is obvious. □

4.1.2 Extracting Unbounded Operators from Bounded Operators

We now provide a set of rewriting rules that extract unbounded temporal operators from the scopes of bounded temporal operators. In what follows, let $\varphi_{xlb} = \mathbf{false}\,\mathcal{U}_{(0,b)}\,\mathbf{true}$, $\varphi_{ylb} = \mathbf{false}\,\mathcal{S}_{(0,b)}\,\mathbf{true}$ and

$$\begin{aligned}
\varphi_{ugb} = {} & \Big((\varphi_{xlb} \implies \Box_{(b,2b)}\varphi_1) \wedge \big(\neg\varphi_{ylb} \implies (\varphi_1 \wedge \Box_{(0,b]}\varphi_1)\big)\Big) \\
& \mathcal{U}\Big(\big(\varphi_1 \wedge (\varphi_1 \,\mathcal{U}_{(b,2b)}\, \varphi_2)\big) \vee \Big(\neg\varphi_{ylb} \wedge \Big(\varphi_2 \vee \big(\varphi_1 \wedge (\varphi_1 \,\mathcal{U}_{(0,b]}\, \varphi_2)\big)\Big)\Big)\Big)\,, \\
\varphi_{ggb} = {} & \Box\Big((\varphi_{xlb} \implies \Box_{(b,2b)}\varphi_1) \wedge \big(\neg\varphi_{ylb} \implies (\varphi_1 \wedge \Box_{(0,b]}\varphi_1)\big)\Big)\,.
\end{aligned}$$

The purpose of formulas φ_{ugb} and φ_{ggb} are similar to $\varphi_1 \,\mathfrak{U}^b_{>b}\, \varphi_2$ and $\neg\big(\mathbf{true}\,\mathfrak{U}^b_{>b}\,(\neg\varphi_1)\big)$, respectively. Indeed, the equivalences in the following proposition still hold if we replace all occurrences of φ_{ugb} and φ_{ggb} by these simpler formulas. We, however, have to use these complicated formulas here as we aim to pull the unbounded 'Until'

operator to the outermost level. The subformulas $\Box_{(b,2b)}\varphi_1$ and $\Box_{(0,b]}\varphi_1$ assert that φ_1 holds continuously in short 'strips', and we use the subformulas φ_{xlb} and φ_{ylb} to ensure that each event before the point where φ_2 holds is covered by such a strip.

Proposition 4.1.2. *The following equivalences hold over infinite timed words.*

$$
\begin{aligned}
\theta\,\mathcal{U}_{(a,b)}\,\big((\varphi_1\,\mathcal{U}\,\varphi_2)\wedge\chi\big) &\iff \theta\,\mathcal{U}_{(a,b)}\,\big((\varphi_1\,\mathcal{U}_{(0,2b)}\,\varphi_2)\wedge\chi\big)\\
&\quad\vee\Big(\big(\theta\,\mathcal{U}_{(a,b)}\,(\Box_{(0,2b)}\varphi_1\wedge\chi)\big)\wedge\varphi_{ugb}\Big)\\
\theta\,\mathcal{U}_{(a,b)}\,(\Box\varphi\wedge\chi) &\iff \big(\theta\,\mathcal{U}_{(a,b)}\,(\Box_{(0,2b)}\varphi\wedge\chi)\big)\wedge\varphi_{ggb}\\
\theta\,\mathcal{U}_{(a,b)}\,\big((\varphi_1\,\mathcal{S}\,\varphi_2)\wedge\chi\big) &\iff \theta\,\mathcal{U}_{(a,b)}\,\big((\varphi_1\,\mathcal{S}_{(0,b)}\,\varphi_2)\wedge\chi\big)\\
&\quad\vee\Big(\big(\theta\,\mathcal{U}_{(a,b)}\,(\boxminus_{(0,b)}\varphi_1\wedge\chi)\big)\wedge\varphi_1\,\mathcal{S}\,\varphi_2\Big)\\
\theta\,\mathcal{U}_{(a,b)}\,(\boxminus\varphi\wedge\chi) &\iff \big(\theta\,\mathcal{U}_{(a,b)}\,(\boxminus_{(0,b)}\varphi\wedge\chi)\big)\wedge\boxminus\varphi\\
\big((\varphi_1\,\mathcal{U}\,\varphi_2)\vee\chi\big)\,\mathcal{U}_{(a,b)}\,\theta &\iff \big((\varphi_1\,\mathcal{U}_{(0,2b)}\,\varphi_2)\vee\chi\big)\,\mathcal{U}_{(a,b)}\,\theta\\
&\quad\vee\Bigg(\Big(\big((\varphi_1\,\mathcal{U}_{(0,2b)}\,\varphi_2)\vee\chi\big)\,\mathcal{U}_{(0,b)}\,(\Box_{(0,2b)}\varphi_1)\Big)\\
&\qquad\wedge\\
&\qquad\Diamond_{(a,b)}\theta\wedge\varphi_{ugb}\Bigg)\\
\big((\Box\varphi)\vee\chi\big)\,\mathcal{U}_{(a,b)}\,\theta &\iff \chi\,\mathcal{U}_{(a,b)}\,\theta\\
&\quad\vee\big(\chi\,\mathcal{U}_{(0,b)}\,(\Box_{(0,2b)}\varphi_1)\wedge\Diamond_{(a,b)}\theta\wedge\varphi_{ggb}\big)\\
\big((\varphi_1\,\mathcal{S}\,\varphi_2)\vee\chi\big)\,\mathcal{U}_{(a,b)}\,\theta &\iff \big((\varphi_1\,\mathcal{S}_{(0,b)}\,\varphi_2)\vee\chi\big)\,\mathcal{U}_{(a,b)}\,\theta\\
&\quad\vee\Bigg(\Big(\big(\boxminus_{(0,b)}\varphi_1\vee(\varphi_1\,\mathcal{S}_{(0,b)}\,\varphi_2)\vee\chi\big)\,\mathcal{U}_{(a,b)}\,\theta\Big)\\
&\qquad\wedge\\
&\qquad\varphi_1\,\mathcal{S}\,\varphi_2\Bigg)\\
\big((\boxminus\varphi)\vee\chi\big)\,\mathcal{U}_{(a,b)}\,\theta &\iff \chi\,\mathcal{U}_{(a,b)}\,\theta\vee\Big(\big((\boxminus_{(0,b)}\varphi\vee\chi)\,\mathcal{U}_{(a,b)}\,\theta\big)\wedge\boxminus\varphi\Big)\,.
\end{aligned}
$$

Proof. We sketch the proof for the first rule as the proofs for the other rules are similar. In the following, let the current position be i.

For the forward direction, let the witness position where φ_2 holds be w. If $\tau_w < \tau_j + 2b$ for some j such that $\tau_j \in (\tau_i + a, \tau_i + b)$, the subformula $\varphi_1\,\mathcal{U}_{(0,2b)}\,\varphi_2$ clearly holds at j and we are done. Otherwise, let j be the maximal position such that $\tau_j \in (\tau_i + a, \tau_i + b)$. We know that $\Box_{(0,2b)}\varphi_1$ must hold at j, so $(\varphi_{xlb} \implies \Box_{(b,2b)}\varphi_1)$, φ_{ylb}, and hence $\big(\neg\varphi_{ylb} \implies (\varphi_1 \wedge \Box_{(0,b]}\varphi_1)\big)$ must hold at all positions j', $i < j' < j$. Let $l > j$ be the minimal position such that $\tau_w \in (\tau_l + b, \tau_l + 2b)$. Consider the following cases:

- There exists such l: It is clear that $\big(\varphi_1 \wedge (\varphi_1 \mathcal{U}_{(b,2b)} \varphi_2)\big)$ holds at l. Since $\Box_{(b,2b)}\varphi_1$ holds at all positions j'', $j \leq j'' < l$ by the minimality of l, $(\varphi_{xlb} \implies \Box_{(b,2b)}\varphi_1)$ also holds at these positions. For the other conjunct, note that φ_{ylb} holds at j and $\varphi_1 \wedge \Box_{(0,b]}\varphi_1$ holds at all positions j''', $j < j''' < l$.

- There is no such l: Consider the following cases:

  - $\neg\varphi_{ylb}$ and $\neg\diamondsuit_{=b}\mathbf{true}$ hold at w: By assumption, there is no event in $(\tau_w - 2b, \tau_w)$. The proof is similar to the case where l exists.

  - $\neg\varphi_{ylb}$ and $\diamondsuit_{=b}\mathbf{true}$ hold at w: Let l' be the position such that $\tau_{l'} = \tau_w - b$. By assumption, there is no event in $(\tau_{l'} - b, \tau_{l'})$. It follows that $\neg\varphi_{ylb}$ and $\big(\varphi_1 \wedge (\varphi_1 \mathcal{U}_{(0,b]} \varphi_2)\big)$ hold at l'. The proof is similar.

  - φ_{ylb} holds at w: By assumption, there is no event in $(\tau_w - 2b, \tau_w - b)$. It is easy to see that there is a position such that $\neg\varphi_{ylb} \wedge \big(\varphi_1 \wedge (\varphi_1 \mathcal{U}_{(0,b]} \varphi_2)\big)$ holds. The proof is again similar.

We prove the other direction by contraposition. Consider the interesting case where $\Box_{(0,2b)}\varphi_1$ holds at the maximal position j such that $j \in (\tau_i + a, \tau_i + b)$, yet $\varphi_1 \mathcal{U} \varphi_2$ does not hold at j. By assumption, there is no φ_2-event in $(\tau_j, \tau_j + 2b)$. If φ_2 never holds in $[\tau_j + 2b, \infty)$ then we are done. Otherwise, let $l > j$ be the minimal position such that both φ_1 and φ_2 do not hold at l (note that $\tau_l \geq \tau_j + 2b$). It is clear that $\Bigg(\big(\varphi_1 \wedge (\varphi_1 \mathcal{U}_{(b,2b)} \varphi_2)\big) \vee \Big(\neg\varphi_{ylb} \wedge \big(\varphi_2 \vee \big(\varphi_1 \wedge (\varphi_1 \mathcal{U}_{(0,b]} \varphi_2)\big)\big)\Big)\Bigg)$ does not hold at all positions j', $i < j' \leq l$. Consider the following cases:

- $\neg\varphi_{ylb}$ holds at l: $\varphi_1 \wedge \Box_{(0,b]}\varphi_1$ does not hold at l, and therefore φ_{ugb} fails to hold at i.

- φ_{ylb} holds at l: Consider the following cases:

  - There is an event in $(\tau_l - 2b, \tau_l - b)$: Let j'' be the maximal position of such an event. We have $j'' + 1 < l$, $\tau_{j''+1} - \tau_{j''} \geq b$ and $\tau_l - \tau_{j''+1} < b$. However, it follows that φ_{ylb} does not hold at $j'' + 1$ and $\varphi_1 \wedge \Box_{(0,b]}\varphi_1$ holds at $j'' + 1$, which is a contradiction.

  - There is no event in $(\tau_l - 2b, \tau_l - b)$: Let j'' be the minimal position such that $\tau_{j''} \in [\tau_l - b, \tau_l)$. It is clear that φ_{ylb} does not hold at j'' and $\varphi_1 \wedge \Box_{(0,b]}\varphi_1$ must hold at j'', which is a contradiction. □

Proposition 4.1.3. *The following equivalences hold over infinite timed words.*

$$
\begin{aligned}
\theta\, \mathfrak{U}^c_{(a,b)}\, \big((\varphi_1 \,\mathcal{U}\, \varphi_2) \wedge \chi\big) &\iff \theta\, \mathfrak{U}^c_{(a,b)}\, \big((\varphi_1 \,\mathcal{U}_{(0,2b)}\, \varphi_2) \wedge \chi\big) \\
&\quad \vee \Big(\big(\theta\, \mathfrak{U}^c_{(a,b)}\, (\Box_{(0,2b)}\varphi_1 \wedge \chi)\big) \wedge \varphi_{ugb} \Big) \\
\theta\, \mathfrak{U}^c_{(a,b)}\, (\Box\varphi \wedge \chi) &\iff \big(\theta\, \mathfrak{U}^c_{(a,b)}\, (\Box_{(0,2b)}\varphi \wedge \chi)\big) \wedge \varphi_{ggb} \\
\theta\, \mathfrak{U}^c_{(a,b)}\, \big((\varphi_1 \,\mathcal{S}\, \varphi_2) \wedge \chi\big) &\iff \theta\, \mathfrak{U}^c_{(a,b)}\, \big((\varphi_1 \,\mathcal{S}_{(0,b)}\, \varphi_2) \wedge \chi\big) \\
&\quad \vee \Big(\big(\theta\, \mathfrak{U}^c_{(a,b)}\, (\boxminus_{(0,b)}\varphi_1 \wedge \chi)\big) \wedge \varphi_1 \,\mathcal{S}\, \varphi_2 \Big) \\
\theta\, \mathfrak{U}^c_{(a,b)}\, (\boxminus\varphi \wedge \chi) &\iff \big(\theta\, \mathfrak{U}^c_{(a,b)}\, (\boxminus_{(0,b)}\varphi \wedge \chi)\big) \wedge \boxminus\varphi \\
\big((\varphi_1 \,\mathcal{U}\, \varphi_2) \vee \chi\big)\, \mathfrak{U}^c_{(a,b)}\, \theta &\iff \big((\varphi_1 \,\mathcal{U}_{(0,2b)}\, \varphi_2) \vee \chi\big)\, \mathfrak{U}^c_{(a,b)}\, \theta \\
&\quad \vee \Bigg(\Big(\big((\varphi_1 \,\mathcal{U}_{(0,2b)}\, \varphi_2) \vee \chi\big)\, \mathfrak{U}^c_{\left(c,c+(b-a)\right)}\, (\Box_{(0,2b)}\varphi_1) \Big) \\
&\qquad \wedge \\
&\qquad \Diamond_{(a,b)}\theta \wedge \varphi_{ugb} \Bigg) \\
\big((\Box\varphi) \vee \chi\big)\, \mathfrak{U}^c_{(a,b)}\, \theta &\iff \chi\, \mathfrak{U}^c_{(a,b)}\, \theta \\
&\quad \vee \big(\chi\, \mathfrak{U}^c_{\left(c,c+(b-a)\right)}\, (\Box_{(0,2b)}\varphi_1) \wedge \Diamond_{(a,b)}\theta \wedge \varphi_{ggb}\big) \\
\big((\varphi_1 \,\mathcal{S}\, \varphi_2) \vee \chi\big)\, \mathfrak{U}^c_{(a,b)}\, \theta &\iff \big((\varphi_1 \,\mathcal{S}_{(0,b)}\, \varphi_2) \vee \chi\big)\, \mathfrak{U}^c_{(a,b)}\, \theta \\
&\quad \vee \Bigg(\Big(\big(\boxminus_{(0,b)}\varphi_1 \vee (\varphi_1 \,\mathcal{S}_{(0,b)}\, \varphi_2) \vee \chi\big)\, \mathfrak{U}^c_{(a,b)}\, \theta \Big) \\
&\qquad \wedge \\
&\qquad \varphi_1 \,\mathcal{S}\, \varphi_2 \Bigg) \\
\big((\boxminus\varphi) \vee \chi\big)\, \mathfrak{U}^c_{(a,b)}\, \theta &\iff \chi\, \mathfrak{U}^c_{(a,b)}\, \theta \vee \Big(\big((\boxminus_{(0,b)}\varphi \vee \chi)\, \mathfrak{U}^c_{(a,b)}\, \theta\big) \wedge \boxminus\varphi \Big).
\end{aligned}
$$

Lemma 4.1.4. *For any* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *formula* φ*, we can use the rules above to obtain an equivalent* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *formula* $\hat{\varphi}$ *in which no unbounded temporal operator appears in the scope of a bounded temporal operator. In particular, all occurrences of* $\mathfrak{U}^c_I$*,* $\mathfrak{S}^c_I$ *have* I *bounded.*

Proof. Define the *unbounding depth* $ud(\varphi)$ of an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula φ to be the modal depth of φ counting only unbounded operators. We demonstrate a rewriting process on φ which terminates in an equivalent formula $\hat{\varphi}$ such that any subformula $\hat{\psi}$ of $\hat{\varphi}$ with outermost operator bounded has $ud(\hat{\psi}) = 0$.

Assume that the input formula φ is in normal form. Let k be the largest unbounding depth among all subformulas of φ with bounded outermost operators. We pick all minimal (wrt. subformula order) such subformulas ψ with $ud(\psi) = k$. By applying the rules in Section 4.1.2, we can rewrite ψ into ψ' where all subformulas of ψ' with

bounded outermost operators have unbounded depths strictly less than k. We then substitute these ψ' back into φ to obtain φ'. We repeat this step until there remain no bounded temporal operators with unbounding depth k. The rules that rewrite a formula into normal form are used whenever necessary on relevant subformulas—this will never affect their unbounding depths, and note that we never introduce $\mathfrak{U}_I^c$ or $\mathfrak{S}_I^c$. It is easy to see that we will eventually obtain such a formula φ^*. Now rewrite φ^* into normal form and start over again. This is to be repeated until we reach $\hat{\varphi}$. □

4.1.3 Completing the Separation

We now have an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula $\hat{\varphi}$ in which no unbounded temporal operator appears in the scope of a bounded temporal operator. If we regard each bounded subformula as a new proposition, the formula $\hat{\varphi}$ can be seen as an LTL formula Φ. We can then apply Gabbay's separation theorem on Φ.

Theorem 4.1.5 ([GPSS80, Theorem 3])**.** *Every* LTL *formula is equivalent (over discrete complete models) to a Boolean combination of*

- *atomic formulas,*
- *formulas of the form* $\varphi_1 \,\mathcal{U}\, \varphi_2$ *such that* φ_1 *and* φ_2 *use only* $\mathcal{U}$*,*
- *formulas of the form* $\varphi_1 \,\mathcal{S}\, \varphi_2$ *such that* φ_1 *and* φ_2 *use only* $\mathcal{S}$*.*

Combining the theorem above with Lemma 4.1.4, we obtain the following lemma.

Lemma 4.1.6. *Every* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *formula is equivalent to a Boolean combination of*

- *bounded formulas,*
- *formulas that use arbitrary* $\mathcal{U}_I$ *but only bounded* $\mathcal{S}_I$*,* $\mathfrak{U}_I^c$*,* $\mathfrak{S}_I^c$*,*
- *formulas that use arbitrary* $\mathcal{S}_I$ *but only bounded* $\mathcal{U}_I$*,* $\mathfrak{U}_I^c$*,* $\mathfrak{S}_I^c$*.*

We now prove the main theorem of this section which states that each $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula is equivalent to a ***syntactically separated*** one.

Theorem 4.1.7. *Every* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *formula can be written as a Boolean combination of*

- *bounded formulas*
- *formulas of the form* $\mathbf{false}\, \mathfrak{U}_{\geq M}^M \varphi$ *for some* $M \in \mathbb{N}_{>0}$

- *formulas of the form* $\mathbf{false}\, \mathfrak{S}^{M}_{\geq M}\, \varphi$ *for some* $M \in \mathbb{N}_{>0}$.

Proof. Suppose that we have an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula φ with no unbounded $\mathcal{S}$. If φ is bounded then we are done. Otherwise we can apply Lemma 4.1.4 (note in particular that it does not introduce new unbounded $\mathcal{U}$ operators) and further assume that $\varphi = \varphi_1 \,\mathcal{U}\, \varphi_2$. Following the idea used in the rewriting rules in Section 4.1.2, for any $M \in \mathbb{N}_{>0}$, we can rewrite φ into the following $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula:

$$\varphi_1 \,\mathcal{U}_{<M}\, \varphi_2 \vee \left(\Box_{<M} \varphi_1 \wedge \left(\mathbf{false}\, \mathfrak{U}^{M}_{\geq M} \left(\varphi_2 \vee \left(\varphi_1 \wedge (\varphi_1 \,\mathcal{U}\, \varphi_2) \right) \right) \right) \right).$$

It is clear that φ_1 and φ_2, and therefore $\varphi_1 \,\mathcal{U}_{<M}\, \varphi_2$ and $\Box_{<M}\varphi_1$, have strictly fewer unbounded $\mathcal{U}$ operators than φ. By the induction hypothesis, φ is equivalent to a syntactically separated $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula. The case for formulas with no unbounded $\mathcal{U}$ is symmetric. □

4.2 Expressing Bounded $\mathsf{FO}[<, +1]$ Formulas

In this section, we describe how to express bounded $\mathsf{FO}[<, +1]$ formulas with one free variable in $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$. The use of rational endpoints is crucial here; $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ cannot express a certain counting modality (which can be written as a bounded $\mathsf{FO}[<, +1]$ formula) if only integer endpoints are allowed [HR07]. As some techniques here are exactly similar to that of Section 3.4.1 (on page 45), we will omit certain explanations.

Suppose that we are given such a formula $\vartheta(x)$. Without loss of generality, we assume that each quantifier in $\vartheta(x)$ uses a fresh new variable and $\vartheta(x)$ contains only existential quantifiers. We say that $\vartheta(x)$ is *N-bounded* if each subformula $\exists x'\, \psi$ of $\vartheta(x)$ is of the form

$$\exists x' \left((x' > x \implies d(x, x') < N) \wedge (x' < x \implies d(x, x') \leq N) \wedge \dots \right).$$

In particular, $\vartheta(x)$ only refers to the events in the half-open interval $[x - N, x + n)$. Similarly, we say that $\vartheta(x)$ is a *unit formula* if each subformula $\exists x'\, \psi$ of $\vartheta(x)$ is of the form

$$\exists x' \left(x' \geq x \wedge d(x, x') < 1 \wedge \dots \right).$$

In this case, $\vartheta(x)$ only refers to the events in $[x, x + 1)$.

Stacking events around a point. Let ρ be an infinite timed word over $\Sigma_{\mathbf{P}}$, $\overline{\mathbf{P}} = \{P_i \mid P \in \mathbf{P}, -N \leq i < N\}$ and $\overline{\mathbf{Q}} = \{Q_i \mid N \leq i < N\}$. For each $t \in \rho$, we can construct a (finite) $[0,1)$-timed word $\overline{\rho_t}$ over $\Sigma_{\overline{\mathbf{P}} \cup \overline{\mathbf{Q}}}$ that satisfies the following:

- For all $\bar{t} \in [0,1)$ and $-N \leq i < N$, P_i holds at $\bar{t} \in \overline{\rho_t}$ iff P holds at $i + \bar{t} \in \rho$.
- For all $\bar{t} \in [0,1)$ and $-N \leq i < N$, Q_i holds at $\bar{t} \in \overline{\rho_t}$ iff $i + \bar{t} \in \rho$.

Stacking N-bounded $\mathsf{FO}[<,+1]$ formulas. Now let $\vartheta(x)$ be an N-bounded $\mathsf{FO}[<,+1]$ formula. Recursively replace every subformula $\exists x'\, \theta$ by

$$\exists x' \Big(\big(Q_{-N}(x') \wedge \theta[x' + (-N)/x']\big) \vee \ldots \vee \big(Q_{N-1}(x') \wedge \theta[x' + (N-1)/x']\big) \Big)$$

where $\vartheta[e/x]$ denotes the formula obtained by substituting all free occurrences of x in ϑ by (an expression) e. We then carry out the following syntactic substitutions:

- For each inequality of the form $x_1 + k_1 < x_2 + k_2$, replace it with
  - $x_1 < x_2$ if $k_1 = k_2$
  - **true** if $k_1 < k_2$
  - **false** if $k_1 > k_2$
- For each distance formula, e.g., $d(x_1 + k_1, x_2 + k_2) < 2$, replace it with
  - **true** if $|k_1 - k_2| \leq 1$
  - $x_2 < x_1$ if $k_2 - k_1 = 2$
  - $x_1 < x_2$ if $k_1 - k_2 = 2$
  - **false** if $|k_1 - k_2| > 2$
- Replace terms of the form $P(x_1 + k)$ with $P_k(x_1)$.

Finally, recursively replace every subformula $\exists x'\, \theta$ by $\exists x' \big(x' \geq x \wedge d(x,x') < 1 \wedge \theta\big)$. This gives a unit formula $\overline{\vartheta}(x)$ such that for each $t \in \rho$,

$$\rho, t \models \vartheta(x) \iff \overline{\rho_t}, 0 \models \overline{\vartheta}(x)\,.$$

From non-metric to metric. For each $\overline{\rho_t}$, we add an event at time 1 (at which no monadic predicate holds) and call the resulting $[0,1]$-timed word $\overline{\rho_t}'$. It is clear that

$$\overline{\rho_t}, 0 \models \overline{\vartheta}(x) \iff \overline{\rho_t}', 0, 1 \models \overline{\vartheta}'(x, y)$$

where $\overline{\vartheta}'(x,y)$ is a non-metric $\mathsf{FO}[<]$ formula obtained by replacing all distance formulas of the form $d(x,x') < 1$ with $x' < y$ in $\overline{\vartheta}(x)$. We now invoke a normal form lemma from [GPSS80] to rewrite $\overline{\vartheta}'(x,y)$ into a disjunction of ***decomposition formulas***.

Lemma 4.2.1 ([GPSS80])**.** *Every* $\mathsf{FO}[<]$ *formula* $\theta(x,y)$ *in which all quantifications are of the form* $\exists x' \, (x' \geq x \wedge x' < y \wedge \ldots)$ *is equivalent to a disjunction of decomposition formulas, i.e.,* $\mathsf{FO}[<]$ *formulas of the form*

$$\begin{aligned} x < y \wedge \exists z_0 \ldots \exists z_n \, (x = z_0 < \cdots < z_n = y) \\ \wedge \bigwedge \{ \Phi_i(z_i) : 0 \leq i < n \} \\ \wedge \bigwedge \{ \forall u \, \big(z_i < u < z_{i+1} \implies \Psi_i(u) \big) : 0 \leq i < n \} \end{aligned}$$

where Φ_i *and* Ψ_i *are* $\mathsf{LTL}_{\mathsf{fut}}$ *formulas.*[2]

A careful inspection of the proof of the lemma above in [Dam94] reveals that we can further assume that Φ_i and Ψ_i are Boolean combinations of atomic formulas.[3] It follows that $\overline{\vartheta}(x)$ is equivalent to a disjunction of unit formulas $\overline{\delta}(x)$ of the form

$$\begin{aligned} \exists z_0 \ldots \exists z_{n-1} \, (x = z_0 < \cdots < z_{n-1}) \wedge d(x, z_{n-1}) < 1 \\ \wedge \bigwedge \{ \Phi_i(z_i) : 0 \leq i < n \} \\ \wedge \bigwedge \{ \forall u \, \big(z_i < u < z_{i+1} \implies \Psi_i(u) \big) : 0 \leq i < n-1 \} \\ \wedge \forall u \, \big(z_{n-1} < u \wedge d(x,u) < 1 \implies \Psi_{n-1}(u) \big) \end{aligned}$$

where Φ_i and Ψ_i are Boolean combinations of atomic formulas.

It remains to show that for each such unit formula $\overline{\delta}(x)$ and each $t \in \rho$, we can construct an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula φ such that

$$\overline{\rho_t}, 0 \models \overline{\delta}(x) \iff \rho, t \models \varphi \, .$$

It turned out to be more convenient to prove a stronger claim, i.e., we can handle

[2] This version of the lemma follows from Lemma 4 and Main Lemma in [GPSS80].

[3] This observation is not necessary for the results of this chapter to hold, but it greatly simplifies our proof.

$\mathsf{FO}[<, +\mathbb{Q}]$ formulas of the following form for any rational number r, $0 \leq r < 1$:

$$\begin{aligned}\exists z_0 \ldots \exists z_{n-1}\, (x = z_0 < \cdots < z_{n-1}) \wedge d(x, z_1) > r \wedge d(x, z_{n-1}) < 1 \\ \wedge \bigwedge \{\Phi_i(z_i) : 1 \leq i < n\} \\ \wedge \forall u \left(x < u \wedge u < z_1 \wedge d(x, u) > r \implies \Psi_0(u)\right) \\ \wedge \bigwedge \{\forall u \left(z_i < u < z_{i+1} \implies \Psi_i(u)\right) : 1 \leq i < n-1\} \\ \wedge \forall u \left(z_{n-1} < u \wedge d(x, u) < 1 \implies \Psi_{n-1}(u)\right).\end{aligned}$$

The proof is by induction on the number of existential quantifiers in $\overline{\delta}(x)$. Before we proceed with the proof, we first define a function f that maps a Boolean combination Ψ of atomic formulas over $\overline{\mathbf{P}} \cup \overline{\mathbf{Q}}$ and i, $-N \leq i < N$ to an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula $f(\Psi, i)$ over $\mathbf{P}$:

- $f(P_j, i) = \begin{cases} \diamondminus_{=(i-j)} P & \text{if } i > j \\ P & \text{if } i = j \\ \diamondsuit_{=(j-i)} P & \text{if } i < j \end{cases}$
- $f(Q_j, i) = \begin{cases} \diamondminus_{=(i-j)} \mathbf{true} & \text{if } i > j \\ \mathbf{true} & \text{if } i = j \\ \diamondsuit_{=(j-i)} \mathbf{true} & \text{if } i < j \end{cases}$
- $f(\mathbf{true}, i) = \mathbf{true}$
- $f(\Psi_1 \wedge \Psi_2, i) = f(\Psi_1, i) \wedge f(\Psi_2, i)$
- $f(\neg\Psi, i) = \neg f(\Psi, i)$.

Now consider the base step. We have

$$\overline{\delta}(x) = \forall u \left(x < u \wedge d(x, u) > r \wedge d(x, u) < 1 \implies \Psi(u)\right)$$

where Ψ is a Boolean combination of atomic formulas. It is clear that

$$\varphi = \bigwedge_{0 \leq i < N} \left(\square_{(i+r, i+1)} f(\Psi, i)\right) \wedge \bigwedge_{-N \leq i < 0} \left(\boxminus_{(|i+1|, |i+r|)} f(\Psi, i)\right).$$

The main idea in the induction step is to consider the ways in which z_1, …, z_{n-1} are scattered in $(r, 1)$. To this end, let us split $(r, 1)$ into an open interval

$(r, r+\frac{1-r}{2n})$ and $2n-1$ half-open intervals $[r+\frac{1-r}{2n}, r+\frac{2(1-r)}{2n})$, $[r+\frac{2(1-r)}{2n}, r+\frac{3(1-r)}{2n})$, $\ldots$, $[r+\frac{(2n-1)(1-r)}{2n}, 1)$. Consider the following cases:

(i). $\{z_1, \ldots, z_{n-1}\} \subseteq (r, r+\frac{1-r}{2n})$ or $\{z_1, \ldots, z_{n-1}\} \subseteq [r+\frac{k(1-r)}{2n}, r+\frac{(k+1)(1-r)}{2n})$ for some k, $1 \leq k < n$.

(ii). $\{z_1, \ldots, z_{n-1}\} \subseteq [r+\frac{k(1-r)}{2n}, r+\frac{(k+1)(1-r)}{2n})$ for some k, $n \leq k < 2n$.

(iii). There exists k, $1 \leq k < 2n$ and l, $1 \leq l < n-1$ such that $z_l < r+\frac{k(1-r)}{2n} \leq z_{l+1}$ (i.e., z_1, $\ldots$, z_{n-1} are not in a single interval).

We now detail the construction of a formula ψ in each case; the desired formula φ is the disjunction of these ψ. The proofs are omitted as they are similar to the proof of Proposition 3.4.1 (on page 48).

- Case (i): Consider the subcase $z_1 > r+\frac{k(1-r)}{2n}$. Let

$$\overrightarrow{\varphi}^i_{n-1} = \bigwedge_{0 \leq j < N-i} \left(\Box_{(j, j+\frac{1-r}{2n})} f(\Psi_{n-1}, i+j)\right) \wedge \bigwedge_{-N-i \leq j < 0} \left(\boxminus_{(|j+\frac{1-r}{2n}|, |j|)} f(\Psi_{n-1}, i+j)\right)$$

for all i, $-N \leq i < N$ and recursively define

$$\overrightarrow{\varphi}^i_m = \bigvee_{-N-i \leq j < N-i} \left(\bigwedge_{-N-i \leq h < N-i} \left(\left(f(\Psi_m, i+h)\right) \mathfrak{U}^h_{(j, j+\frac{1-r}{2n})} \left(f(\Phi_{m+1}, i+j) \wedge \overrightarrow{\varphi}^{i+j}_{m+1}\right)\right)\right)$$

for all i, $-N \leq i < N$ and m, $1 \leq m < n-1$. Let α_k be the conjunction of

$$\bigwedge_{0 \leq i < N} \left(\Box_{(i+r, i+r+\frac{k(1-r)}{2n}]} f(\Psi_0, i)\right) \wedge \bigwedge_{-N \leq i < 0} \left(\boxminus_{[|i+r+\frac{k(1-r)}{2n}|, |i+r|)} f(\Psi_0, i)\right)$$

and

$$\bigvee_{-N \leq j < N} \left(\bigwedge_{-N \leq h < N} \left(\left(f(\Psi_0, h)\right) \mathfrak{U}^{h+r+\frac{k(1-r)}{2n}}_{(j+r+\frac{k(1-r)}{2n}, j+r+\frac{(k+1)(1-r)}{2n})} \left(f(\Phi_1, j) \wedge \overrightarrow{\varphi}^j_1\right)\right)\right)$$

and

$$\bigwedge_{0 \leq i < N} \left(\Box_{[i+r+\frac{(k+1)(1-r)}{2n}, i+1)} f(\Psi_{n-1}, i)\right) \wedge \bigwedge_{-N \leq i < 0} \left(\boxminus_{(|i+1|, |i+r+\frac{(k+1)(1-r)}{2n}|]} f(\Psi_{n-1}, i)\right).$$

Similarly, we construct α'_k to handle the subcase $z_1 = r+\frac{k(1-r)}{2n}$. The formula ψ is the disjunction of these formulas for k, $0 \leq k < n$.

- Case (ii): Let

$$\overleftarrow{\varphi}_1^i = \bigwedge_{0<j<N-i} \big(\Box_{(j-\frac{1-r}{2n},j)} f(\Psi_0, i+j)\big) \wedge \bigwedge_{-N-i\le j\le 0} \big(\boxminus_{(|j|,|j-\frac{1-r}{2n}|)} f(\Psi_0, i+j)\big)$$

for all i, $-N \le i < N$ and recursively define

$$\overleftarrow{\varphi}_m^i = \bigvee_{-N-i\le j<N-i} \Big(\bigwedge_{-N-i\le h<N-i} \Big(\big(f(\Psi_{m-1}, i+h)\big) \mathfrak{S}^h_{(j,j+\frac{1-r}{2n})} \big(f(\Phi_{m-1}, i+j) \wedge \overleftarrow{\varphi}_{m-1}^{i+j}\big)\Big)\Big)$$

for all i, $-N \le i < N$ and m, $1 < m \le n-1$. Let β_k be the conjunction of

$$\bigwedge_{0\le i<N} \big(\Box_{[i+r+\frac{(k+1)(1-r)}{2n},i+1)} f(\Psi_{n-1}, i)\big) \wedge \bigwedge_{-N\le i<0} \big(\boxminus_{(|i+1|,|i+r+\frac{(k+1)(1-r)}{2n}|]} f(\Psi_{n-1}, i)\big)$$

and

$$\bigvee_{-N\le j<N} \Big(\bigwedge_{-N\le h<N} \Big(\big(f(\Psi_{n-1}, h)\big) \mathfrak{S}^{-|h+r+\frac{(k+1)(1-r)}{2n}|}_{(-|j+r+\frac{(k+1)(1-r)}{2n}|,-|j+r+\frac{k(1-r)}{2n}|]} \big(f(\Phi_{n-1}, j) \wedge \overleftarrow{\varphi}_{n-1}^{j}\big)\Big)\Big)$$

and

$$\bigwedge_{0\le i<N} \big(\Box_{(i+r,i+r+\frac{k(1-r)}{2n})} f(\Psi_0, i)\big) \wedge \bigwedge_{-N\le i<0} \big(\boxminus_{(|i+r+\frac{k(1-r)}{2n}|,|i+r|)} f(\Psi_0, i)\big)\,.$$

The formula ψ is the disjunction of β_k, $n \le k < 2n$.

- Case (iii): Suppose that $z_l < r + \frac{k(1-r)}{2n} \le z_{l+1}$ for some k, $1 \le k < 2n$ and l, $1 \le l < n-1$. Consider the following subcases:
  - $r + \frac{k(1-r)}{2n} < z_{l+1}$: This can be handled by the conjunction of the formulas that correspond to the following:
    - $\{z_1, \ldots, z_l\} \subseteq (r, r+\frac{k(1-r)}{2n})$: We can scale the corresponding $\mathsf{FO}[<, +\mathbb{Q}]$ formula by $\frac{1}{r+\frac{k(1-r)}{2n}}$, apply the induction hypothesis (with $r' = \frac{r}{r+\frac{k(1-r)}{2n}}$) and scale the resulting $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula by $r + \frac{k(1-r)}{2n}$.
    - $\{z_{l+1}, \ldots, z_{n-1}\} \subseteq (r + \frac{k(1-r)}{2n}, 1)$: We can set $r' = r + \frac{k(1-r)}{2n}$ and apply the induction hypothesis.
  - $r + \frac{k(1-r)}{2n} = z_{l+1}$: Exactly similar except that we also use the following

formula as a conjunct:

$$\bigvee_{0\leq i<N} \left(\Diamond_{=i+r+\frac{k(1-r)}{2n}} f(\Phi_{l+1}, i)\right) \vee \bigvee_{-N\leq i<0} \left(\Diamond_{=|i+r+\frac{k(1-r)}{2n}|} f(\Phi_{l+1}, i)\right).$$

The formula ψ is the disjunction of these formulas for all k, $1 \leq k < 2n$ and l, $1 \leq l < n-1$.

Finally, observe that the original claim can be achieved by setting $r = 0$ and using the conjunct $f(\Phi_0, 0)$. We are now ready to state the main result of this section.

Theorem 4.2.2. *For every N-bounded $\mathsf{FO}[<,+1]$ formula $\vartheta(x)$ there exists an equivalent $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$ formula φ (with rational endpoints).*

4.3 Expressive Completeness of $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$

In this section, we show that any $\mathsf{FO}[<,+\mathbb{Q}]$ formula with one free variable can be expressed as an $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$ formula. The crucial idea here is that we can separate formulas far enough so that all references to a certain variable become vacuous. To this end, we define $fr(\varphi)$ and $pr(\varphi)$ (*future-reach* and *past-reach*) for an $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$ formula φ as follows (let the set of monadic predicates be $\mathbf{P}$):

- $fr(\mathbf{true}) = pr(\mathbf{true}) = fr(P) = pr(P) = 0$ for all $P \in \mathbf{P}$
- $fr(\varphi_1 \,\mathcal{U}_I\, \varphi_2) = \sup(I) + \max\big(fr(\varphi_1), fr(\varphi_2)\big)$
- $pr(\varphi_1 \,\mathcal{S}_I\, \varphi_2) = \sup(I) + \max\big(pr(\varphi_1), pr(\varphi_2)\big)$
- $fr(\varphi_1 \,\mathcal{S}_I\, \varphi_2) = \max\big(fr(\varphi_1), fr(\varphi_2) - \inf(I)\big)$
- $pr(\varphi_1 \,\mathcal{U}_I\, \varphi_2) = \max\big(pr(\varphi_1), pr(\varphi_2) - \inf(I)\big)$
- $fr(\varphi_1 \,\mathfrak{U}^c_I\, \varphi_2) = \max\big(c + |I| + fr(\varphi_1), \sup(I) + fr(\varphi_2)\big)$
- $pr(\varphi_1 \,\mathfrak{S}^c_I\, \varphi_2) = \max\big(c + |I| + pr(\varphi_1), \sup(I) + pr(\varphi_2)\big)$
- $fr(\varphi_1 \,\mathfrak{S}^c_I\, \varphi_2) = \max\big(fr(\varphi_1) - c, fr(\varphi_2) - \inf(I)\big)$
- $pr(\varphi_1 \,\mathfrak{U}^c_I\, \varphi_2) = \max\big(pr(\varphi_1) - c, pr(\varphi_2) - \inf(I)\big)$.

The cases for Boolean connectives are defined in the expected way. Clearly, they are over-approximations of the lengths of the time horizons needed to determine the truth value of φ.

Theorem 4.3.1. *For every* $\mathsf{FO}[<,+1]$ *formula* $\vartheta(x)$ *there exists an equivalent* $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$ *formula* φ *(with rational endpoints).*

Proof. The proof is by induction on the quantifier depth of $\vartheta(x)$. In what follows, Let the set of monadic predicates be $\mathbf{P}$. Without loss of generality, we assume that each quantifier in $\vartheta(x)$ uses a fresh new variable.

- *Base step.* $\vartheta(x)$ is a Boolean combination of atomic formulas $P(x)$, $x < x$, $d(x,x) \sim c$, **true**. We can replace them by P, **false**, $0 \sim c$ and **true** respectively to obtain φ.

- *Induction step.* Without loss of generality assume that $\vartheta(x) = \exists y\, \theta(x,y)$. Our goal here is to remove x from $\theta(x,y)$. To this end, we can remove $x < x$ and $d(x,x) \sim c$ as before and use a mapping $\gamma : \mathbf{P} \mapsto \{\mathbf{true}, \mathbf{false}\}$ to determine the truth value of $P(x)$ for each $P \in \mathbf{P}$. Thus we can rewrite $\exists y\, \theta(x,y)$ as

$$\bigvee_{\gamma} \big(\eta_\gamma(x) \wedge \exists y\, \theta_\gamma(x,y)\big) \tag{4.1}$$

  where $\eta_\gamma = \bigwedge_{P \in \mathbf{P}} \big(P(x) \iff \gamma(P)\big)$ and $\theta_\gamma(x,y)$ is obtained by replacing $P(x)$ with $\gamma(P)$ for all $P \in \mathbf{P}$ in $\theta(x,y)$.

  Observe that in each $\theta_\gamma(x,y)$, x only appears in atomic formulas of the form $x < z$, $z < x$, $d(x,z) \sim c$ and $d(z,x) \sim c$ where $\sim \in \{<,>\}$. We now introduce new monadic predicates $P_<$, $P_>$, and $P_{\sim c}$ for each $d(x,z) \sim c$ or $d(z,x) \sim c$ that correspond to these atomic formulas. Specifically, we write $x < z$ as $P_<(z)$, $z < x$ as $P_>(z)$ and $d(x,z) \sim c$ or $d(z,x) \sim c$ as $P_{\sim c}(z)$. Apply these substitutions to (4.1) yields

$$\bigvee_{\gamma} \big(\eta_\gamma(x) \wedge \exists y\, \theta'_\gamma(y)\big) \tag{4.2}$$

  where x does not occur in each $\theta'_\gamma(y)$. In particular, (4.1) and (4.2) have the same truth value at any given point if $P_<$, $P_>$ and all $P_{\sim c}$ are interpreted correctly with respect to that point.

  Observe that each $\eta_\gamma(x)$ is clearly equivalent to an $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$ formula ψ_γ. By the induction hypothesis, each $\theta'_\gamma(y)$ is also equivalent to an $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$ formula φ_γ. It follows that (4.2) is equivalent to the following $\mathsf{MTL}[\mathfrak{U},\mathfrak{S}]$ formula:

$$\varphi' = \bigvee_{\gamma} \big(\psi_\gamma(x) \wedge (\diamondsuit \varphi_\gamma \vee \varphi_\gamma \vee \diamondsuit \varphi_\gamma)\big)\,.$$

By Theorem 4.1.7 (on page 59) and noting that $M \in \mathbb{N}_{>0}$ can be chosen arbitrarily, φ' is equivalent to a Boolean combination φ'' of

- bounded formulas
- formulas of the form $\mathbf{false}\, \mathfrak{U}^M_{\geq M}\, \psi$ such that $M > c_{max} + pr(\psi)$
- formulas of the form $\mathbf{false}\, \mathfrak{S}^M_{\geq M}\, \psi$ such that $M > c_{max} + fr(\psi)$

where c_{max} are the largest constants appearing in monadic predicates of the form $P_{\sim c}$ in respective formulas ψ.

Now suppose that φ'' is evaluated at t_1. For the formulas of the second type, since all references to $P_<$, $P_>$ and all $P_{\sim c}$ must happen at time strictly greater than $t_1 + c_{max}$, we can simply replace them with **true**, **false** and $c_{max} + 1 \sim c$ to obtain equivalent $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formulas over $\mathbf{P}$. The formulas of the third type can be dealt with similarly. Finally, for the formulas of the first type, we replace $P_<$, $P_>$ and all $P_{\sim c}$ with $x < z$, $z < x$ and $d(x, z) \sim c$. The resulting formulas are clearly bounded $\mathsf{FO}[<, +\mathbb{Q}]$ formulas. We can scale them to bounded $\mathsf{FO}[<, +1]$ formulas, apply Theorem 4.2.2 (on page 66) and then scale back to obtain equivalent $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formulas over $\mathbf{P}$. □

The main result of this chapter now follows from a simple scaling argument.

Theorem 4.3.2. $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *with rational endpoints is expressively complete for* $\mathsf{FO}[<, +\mathbb{Q}]$ *over infinite timed words.*

4.4 Discussion

In this chapter, we continued our study on the expressiveness of metric logics in the pointwise semantics. Building on a previous work of Hunter, Ouaknine and Worrell [HOW13], we showed that the *rational* version of $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ is expressively complete for $\mathsf{FO}[<, +\mathbb{Q}]$ over infinite timed words. The result answers an implicit open question in a long line of research started in [AH90] and further developed in [BCM05, PD06, DP06, PS11].

It is known that the *integer* version of MTL extended with counting modalities (and their past counterparts) is expressively complete for $\mathsf{FO}[<, +1]$ over the reals [Hun13].[4] We conjecture that the analogous result holds in the pointwise semantics, i.e., the

[4]This result is stronger than [HOW13] as counting modalities (and their past counterparts) can be expressed in MTL with rational endpoints.

integer version of $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ becomes expressively complete for $\mathsf{FO}[<, +1]$ when we add counting modalities. Adapting the proof in [Hun13] to the pointwise case, however, is not a straight-forward task. In particular, the proof relies on the expressive completeness of MITL with counting modalities for Q2MLO [HR04] (one of the most expressive yet decidable fragment of $\mathsf{FO}[<, +1]$ in the continuous semantics), a result that itself requires a highly non-trivial proof [HR06] and seems to hold only in the continuous semantics.

Besides expressiveness, another major concern in the study of metric logics is *decidability*. While our new modalities strictly increase the expressiveness of MTL, it is not clear how they can be used to extend the decidability frontier of metric logics. Indeed, adding modalities $\mathfrak{U}^c_I$ to $\mathsf{MITL}_{\mathsf{fut}}$, even if we require all constraining intervals I to be non-singular, renders it undecidable: the encoding in [OW06a] carries over with some trivial modifications. It is possible that we may recover the decidability by introducing fuzziness into the superscript 'c' of $\mathfrak{U}^c_I$, but we do not know how this can be done meaningfully and whether this helps. On the other hand, we may add $\mathcal{B}^{\rightarrow}_I$ with bounded I (defined in Section 3.2.4 on page 33) into $\mathsf{Safety\text{-}MTL}_{\mathsf{fut}}$ [OW06b] (one of the most expressive yet decidable fragments of $\mathsf{FO}[<, +1]$ in the pointwise semantics[5]) without affecting its decidability, as $\mathcal{B}^{\rightarrow}_I$ is readily expressible in the class of one-clock alternating timed automata used in [OW06b].

From a practical viewpoint, since the satisfiability problem for $\mathsf{Safety\text{-}MTL}_{\mathsf{fut}}$ is non-primitive recursive [LOW13], one might also be interested in extending $\mathsf{MITL}_{\mathsf{fut}}$ or MITL (with the hope that the satisfiability problem remains EXPSPACE-complete). In this regard, we would like to answer the following question: what is the complexity of the satisfiability problem for the logic obtained by adding $\mathcal{B}^{\rightarrow}_I$ (with non-singular I) into $\mathsf{MITL}_{\mathsf{fut}}$? Since $\mathcal{B}^{\rightarrow}_I$ can be expressed in one-clock alternating timed automata, it may possibly be handled in the framework of [BEG14]. More generally, we would also like to consider MITL extended with $\mathcal{B}^{\rightarrow}_I$ and $\mathcal{B}^{\leftarrow}_I$ (with non-singular I). Note that allowing these modalities simultaneously does not automatically lead to undecidability; for example, $\mathcal{B}^{\rightarrow}_{(a,b)}P$ can be expressed (in the continuous semantics) as the following Q2MLO formula:

$$(\exists x')^{<x+b}_{>x+a}\big(P(x') \wedge \neg(\exists x'')^{<x+b}_{>x+a}(\neg P_\epsilon(x'') \wedge x'' < x')\big)\,.$$

[5] We believe that $\mathsf{Safety\text{-}MTL}_{\mathsf{fut}}$ is not subsumed by Q2MLO as (i) $\mathsf{Safety\text{-}MTL}_{\mathsf{fut}}$ allows punctuality (ii) $\mathsf{Safety\text{-}MTL}_{\mathsf{fut}}$ is undecidable over the reals.

Chapter 5

Monitoring of Real-Time Specifications

While the results in Chapter 4 (regarding $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ and $\mathsf{FO}[<, +\mathbb{Q}]$) might be interesting from a theoretical perspective, it is not obvious how they can benefit model checking, as the corresponding problem is undecidable already for $\mathsf{MTL}_{\mathsf{fut}}$ [OW06a]. Nonetheless, we show in this chapter that those results can indeed be very useful in *monitoring*, a core interest in *runtime verification*.

We first define some basic notions used throughout the chapter, and justify the necessity of a *bounded-variability* assumption. Then we give a bi-linear path-checking algorithm for $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$,[1] based on which we devise a monitoring procedure for an expressively complete fragment of $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$. Among other advantages, our procedure is *trace-length independent*, i.e., its space usage does not depend on the length of the (growing) trace. Finally, we show that our approach extends to arbitrary $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formulas by employing the syntactic rewriting rules in Chapter 4.

5.1 Preliminaries

5.1.1 Truncated Semantics

Since in monitoring one naturally deals with truncated paths, it is useful to define satisfaction relations of $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formulas over finite timed words. To this end, we adopt a timed version of the ***truncated semantics*** [EFHL03] which incorporates *strong* and *weak* views on satisfaction over truncated paths. These views indicate

[1] In this chapter we assume that all timestamps are rational, can be finitely represented (e.g., with a built-in data type), and additions and subtractions on them can be done in constant time.

whether the evaluation of the formula 'has completed', i.e., whether the truth value of the formula on the whole path is already determined by the prefix. Intuitively speaking, in the strong view one is 'pessimistic' on satisfaction—for example, $\Box P$ can never be strongly satisfied by any finite timed word as such may be extended into an infinite timed word that violates the formula. On the other hand, one is 'optimistic' on satisfaction in the weak view; e.g., $\Diamond_{<5} P$ is weakly satisfied by any finite timed word whose timestamps are all strictly less than 5 since one can always extend such into an infinite timed word that satisfies the formula. We also consider the *neutral* view, which extends to MTL the traditional LTL semantics over finite words [MP95].

The respective strong, neutral and weak satisfaction relations will be denoted by $\models_f^+$, $\models_f$ and $\models_f^-$ respectively. We write $\rho \models_f^+ \varphi$ (respectively $\rho \models_f \varphi$, $\rho \models_f^- \varphi$) if $\rho, 0 \models_f^+ \varphi$ ($\rho, 0 \models_f \varphi$, $\rho, 0 \models_f^- \varphi$). In what follows, $P \in \mathbf{P}$ is a proposition (monadic predicate) and $I \subseteq (0, \infty)$ is an interval with endpoints in $\mathbb{Q}_{\geq 0} \cup \{\infty\}$.

Definition 5.1.1. *The satisfaction relation $\rho, i \models_f^+ \varphi$ for an* MTL$[\mathfrak{U}, \mathfrak{S}]$ *formula φ, a finite timed word $\rho = (\sigma, \tau)$ and a position i, $0 \leq i < |\rho|$ is defined as follows:*

- $\rho, i \models_f^+ P$ *iff* $P \in \sigma_i$
- $\rho, i \models_f^+ \mathbf{true}$
- $\rho, i \models_f^+ \varphi_1 \wedge \varphi_2$ *iff* $\rho, i \models_f^+ \varphi_1$ *and* $\rho, i \models_f^+ \varphi_2$
- $\rho, i \models_f^+ \neg\varphi$ *iff* $\rho, i \not\models_f^- \varphi$
- $\rho, i \models_f^+ \varphi_1 \,\mathcal{U}_I\, \varphi_2$ *iff there exists* j, $i < j < |\rho|$ *such that* $\rho, j \models_f^+ \varphi_2$, $\tau_j - \tau_i \in I$, *and* $\rho, k \models_f^+ \varphi_1$ *for all* k *with* $i < k < j$
- $\rho, i \models_f^+ \varphi_1 \,\mathcal{S}_I\, \varphi_2$ *iff there exists* j, $0 \leq j < i$ *such that* $\rho, j \models_f^+ \varphi_2$, $\tau_i - \tau_j \in I$, *and* $\rho, k \models_f^+ \varphi_1$ *for all* k *with* $j < k < i$
- $\rho, i \models_f^+ \varphi_1 \,\mathfrak{U}_I^c\, \varphi_2$ *iff there exists* j, $i < j < |\rho|$ *such that* $\rho, j \models_f^+ \varphi_2$, $\tau_j - \tau_i \in I$, *and* $\rho, k \models_f^+ \varphi_1$ *for all* k, $i < k < j$ *such that* $\tau_k - \tau_i > c$ *and* $\tau_j - \tau_k > a - c$ *where* $a = \inf(I)$
- $\rho, i \models_f^+ \varphi_1 \,\mathfrak{S}_I^c\, \varphi_2$ *iff there exists* j, $0 \leq j < i$ *such that* $\rho, j \models_f^+ \varphi_2$, $\tau_i - \tau_j \in I$, *and* $\rho, k \models_f^+ \varphi_1$ *for all* k, $j < k < i$ *such that* $\tau_i - \tau_k > c$ *and* $\tau_k - \tau_j > a - c$ *where* $a = \inf(I)$.

Definition 5.1.2. *The satisfaction relation $\rho, i \models_f \varphi$ for an* MTL$[\mathfrak{U}, \mathfrak{S}]$ *formula φ, a finite timed word $\rho = (\sigma, \tau)$ and a position i, $0 \leq i < |\rho|$ is defined as follows:*

- $\rho, i \models_f P$ *iff* $P \in \sigma_i$
- $\rho, i \models_f \mathbf{true}$
- $\rho, i \models_f \varphi_1 \wedge \varphi_2$ *iff* $\rho, i \models_f \varphi_1$ *and* $\rho, i \models_f \varphi_2$
- $\rho, i \models_f \neg\varphi$ *iff* $\rho, i \not\models_f \varphi$
- $\rho, i \models_f \varphi_1 \,\mathcal{U}_I\, \varphi_2$ *iff there exists* j, $i < j < |\rho|$ *such that* $\rho, j \models_f \varphi_2$, $\tau_j - \tau_i \in I$, *and* $\rho, k \models_f \varphi_1$ *for all* k *with* $i < k < j$
- $\rho, i \models_f \varphi_1 \,\mathcal{S}_I\, \varphi_2$ *iff there exists* j, $0 \leq j < i$ *such that* $\rho, j \models_f \varphi_2$, $\tau_i - \tau_j \in I$, *and* $\rho, k \models_f \varphi_1$ *for all* k *with* $j < k < i$
- $\rho, i \models_f \varphi_1 \,\mathfrak{U}_I^c\, \varphi_2$ *iff there exists* j, $i < j < |\rho|$ *such that* $\rho, j \models_f \varphi_2$, $\tau_j - \tau_i \in I$, *and* $\rho, k \models_f \varphi_1$ *for all* k, $i < k < j$ *such that* $\tau_k - \tau_i > c$ *and* $\tau_j - \tau_k > a - c$ *where* $a = \inf(I)$
- $\rho, i \models_f \varphi_1 \,\mathfrak{S}_I^c\, \varphi_2$ *iff there exists* j, $0 \leq j < i$ *such that* $\rho, j \models_f \varphi_2$, $\tau_i - \tau_j \in I$, *and* $\rho, k \models_f \varphi_1$ *for all* k, $j < k < i$ *such that* $\tau_i - \tau_k > c$ *and* $\tau_k - \tau_j > a - c$ *where* $a = \inf(I)$.

Definition 5.1.3. *The satisfaction relation* $\rho, i \models_f^- \varphi$ *for an* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *formula* φ, *a finite timed word* $\rho = (\sigma, \tau)$ *and a position* i, $0 \leq i < |\rho|$ *is defined as follows:*

- $\rho, i \models_f^- P$ *iff* $P \in \sigma_i$
- $\rho, i \models_f^- \mathbf{true}$
- $\rho, i \models_f^- \varphi_1 \wedge \varphi_2$ *iff* $\rho, i \models_f^- \varphi_1$ *and* $\rho, i \models_f^- \varphi_2$
- $\rho, i \models_f^- \neg\varphi$ *iff* $\rho, i \not\models_f^+ \varphi$
- $\rho, i \models_f^- \varphi_1 \,\mathcal{U}_I\, \varphi_2$ *iff either of the following holds:*
  - *there exists* j, $i < j < |\rho|$ *such that* $\rho, j \models_f^- \varphi_2$, $\tau_j - \tau_i \in I$, *and* $\rho, k \models_f^- \varphi_1$ *for all* k *with* $i < k < j$
  - $\tau_{|\rho|-1} - \tau_i < \sup(I)$ *and* $\rho, k \models_f^- \varphi_1$ *for all* k *with* $i < k < |\rho|$
- $\rho, i \models_f^- \varphi_1 \,\mathcal{S}_I\, \varphi_2$ *iff there exists* j, $0 \leq j < i$ *such that* $\rho, j \models_f^- \varphi_2$, $\tau_i - \tau_j \in I$, *and* $\rho, k \models_f^- \varphi_1$ *for all* k *with* $j < k < i$
- $\rho, i \models_f^- \varphi_1 \,\mathfrak{U}_I^c\, \varphi_2$ *iff either of the following holds:*

– *there exists* j, $i < j < |\rho|$ *such that* $\rho, j \models_f^- \varphi_2, \tau_j - \tau_i \in I$, *and* $\rho, k \models_f^- \varphi_1$ *for all* k, $i < k < j$ *such that* $\tau_k - \tau_i > c$ *and* $\tau_j - \tau_k > a - c$ *where* $a = \inf(I)$

– $\tau_{|\rho|-1} - \tau_i < \sup(I)$ *and* $\rho, k \models_f^- \varphi_1$ *for all* k, $i < k < |\rho|$ *such that* $\tau_k - \tau_i > c$ *and* $\tau_{|\rho|-1} - \tau_k \geq a - c$ *where* $a = \inf(I)$

- $\rho, i \models_f^- \varphi_1 \, \mathfrak{S}_I^c \, \varphi_2$ *iff there exists* j, $0 \leq j < i$ *such that* $\rho, j \models_f^- \varphi_2$, $\tau_i - \tau_j \in I$, *and* $\rho, k \models_f^- \varphi_1$ *for all* k, $j < k < i$ *such that* $\tau_i - \tau_k > c$ *and* $\tau_k - \tau_j > (a - c)$ *where* $a = \inf(I)$.

The following proposition which helps explain the terms strong, neutral and weak, can be proved by a simple induction on the structure of φ.

Proposition 5.1.4. *For a finite timed word* ρ, *a position* i *in* ρ *and an* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *formula* φ,

$$\rho, i \models_f^+ \varphi \implies \rho, i \models_f \varphi \text{ and } \rho, i \models_f \varphi \implies \rho, i \models_f^- \varphi \,.$$

5.1.2 Informative Prefixes

The notion of *informative prefixes* [KV01] has been adopted in several works on LTL monitoring, e.g., [Gei01, AKT+06]. Intuitively, an informative prefix for a formula φ is a finite timed word that 'tells the whole story' about the fulfilment or violation of φ.[2] This notion is formalised in terms of the truncated semantics: we say that ρ is ***informative*** for φ if either of the following holds:[3]

- ρ strongly satisfies φ, i.e., $\rho \models_f^+ \varphi$. In this case we say that ρ is an *informative good prefix* for φ; or
- ρ fails to weakly satisfy φ, i.e., $\rho \not\models_f^- \varphi$. In this case we say that ρ is an *informative bad prefix* for φ.

The following proposition follows immediately from the definition of informative prefixes. In words, negating (syntactically) a formula essentially exchanges its set of informative good prefixes and informative bad prefixes.

Proposition 5.1.5. *For an* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *formula, a finite timed word* ρ *is an informative good prefix for* φ *iff* ρ *is an informative bad prefix for* $\neg\varphi$.

[2] Our usage of the term *informative* slightly deviates from [KV01] as in that paper the term refers exclusively to bad prefixes.

[3] Note that we have adopted here the semantic characterisation of informative prefixes in terms of the truncated semantics from [EFHL03], rather than the original syntactic definition [KV01].

Example 5.1.6. Consider the following formula over $\{P\}$:

$$\varphi = \Diamond\Box(\neg P) \wedge \Box(P \implies \Diamond_{<3} P)\,.$$

We say that the finite timed word $\rho = (\{P\}, 0)(\{P\}, 2)(\emptyset, 5.5)$ is an informative bad prefix for φ as the second conjunct has 'already' been violated, i.e., there is a P-event not followed by another P-event in three time units ($\rho \models_f^+ \neg\varphi$ or, equivalently, $\rho \not\models_f^- \varphi$). On the other hand, while $\rho' = (\{P\}, 0)(\{P\}, 2)(\{P\}, 4)$ is a bad prefix for φ, it can be extended so that each P-event is followed by another P-event in three time units but the first conjunct is always 'not yet' violated ($\rho' \not\models_f^+ \neg\varphi$ or equivalently $\rho' \models_f^- \varphi$). Thus we do not consider it an informative bad prefix.

Example 5.1.7. Consider the following formula over $\{P\}$:

$$\varphi' = \Box(\neg P) \wedge \Box(P \implies \Diamond_{<3} P)\,.$$

This formula is equivalent to the formula φ in the previous example. However, all the bad prefixes ρ for φ' are informative ($\rho \models_f^+ \neg\varphi$ or equivalently $\rho \not\models_f^- \varphi$).

5.2 Bounded Variability

Conceptually, we can regard a monitor (say, which accepts the bad prefixes for a metric formula φ) as a *deterministic* timed automaton (over finite timed words). The monitoring process can thus be seen as simply executing the automaton. However, it is long known that in a dense-time setting, such a monitor needs an unbounded number of clocks and therefore cannot be realised [AH92]. The proof for the case of continuous semantics can be found in [MNP05, Rey14] where non-standard versions of timed automata (defined over flows) are considered. In this section, we give a proof for the case of pointwise semantics using the standard version of timed automata (defined over timed words). In particular, the claim is already true for fairly basic metric logics, e.g., $\mathsf{MITL}_{\mathsf{fut}}$. For simplicity, we will work with a variant of timed automata with no location invariants (it is well-known that this variant is equally expressive as the variant with location invariants [AM04]).

Proposition 5.2.1. *Consider a clock constraint δ and a clock valuation $\mathbf{v}$ in a timed automaton. If δ is satisfied by $(\mathbf{v}+t)$ for $t \in [0, t_1)$ but not by $(\mathbf{v}+t_1)$ for some $t_1 > 0$, then there is a constraint of the form $x < c$ in δ such that $(\mathbf{v}+t_1)(x) = c$. Similar claims hold for other cases; for example, if δ is satisfied by $(\mathbf{v}+t)$ for $t \in (t_1, t_2)$*

($0 < t_1 < t_2$) but not by $(\mathbf{v} + t_1)$, then there is a constraint of the form $x > c$ in δ such that $(\mathbf{v} + t_1)(x) = c$.

Proof. Recall that by definition, δ is a conjunction of constraints of the form $x \bowtie c$ where $c \in \mathbb{N}$ and $\bowtie \in \{<, \leq, >, \geq\}$. Consider the following cases:

- There is no constraint of the form $x < c$ in δ: It is clear that for all constraints of the form $y > c$ or $y \geq c$ are satisfied by $(\mathbf{v} + t_1)$. For each constraint of the form $y \leq c$, if there is some t', $0 < t' < t_1$ such that $(\mathbf{v} + t')(y) = c$, then there is some t'', $t' < t'' < t_1$ such that $(\mathbf{v} + t'')(y) > c$. Therefore, the constraint is satisfied by $(\mathbf{v} + t_1)$.

- For all constraints of the form $x < c$ in δ we have $(\mathbf{v} + t_1)(x) \neq c$: Consider such a constraint. Since $(\mathbf{v} + t)(x) < c$ for $t \in [0, t_1)$, there must be $t' \geq t_1$ such that $(\mathbf{v} + t')(x) = c$. It follows that $(\mathbf{v} + t_1)(x) < c$. All constraints of the form $y > c$, $y \geq c$ or $y \leq c$ are satisfied by $(\mathbf{v} + t_1)$ as argued in the case above.

We must have $(\mathbf{v} + t_1) \models \delta$ in both cases, which is a contradiction. The other cases can be proved similarly. □

Proposition 5.2.2. *There is an* $\mathsf{MITL}_{\mathsf{fut}}$ *formula such that its bad prefixes cannot be recognised by any deterministic timed automaton.*

Proof. Suppose that the bad prefixes for $\varphi = \Box_{(0,1)}(P \implies \Diamond_{[1,2]} Q)$ is recognised by some deterministic timed automaton $\mathcal{D}$ with $\Sigma = 2^{\{P,Q\}}$. By definition, let the only initial location of $\mathcal{D}$ be s_0. Without loss of generality we assume that $\mathcal{D}$ is complete, i.e., it has a run on any finite timed word over Σ.

Now consider the finite timed word

$$\rho = (\{P\}, \tau_0)(\{P\}, \tau_1) \dots (\{P\}, \tau_{m+1})$$

where $0 = \tau_0 < \tau_1 < \cdots < \tau_{m+1} = 1$. We pick a P-event with timestamp τ_k ($0 < k < m + 1$) and let $t_2 = \tau_k + 1$ (see Figure 5.1 for an illustration). The run of $\mathcal{D}$ on ρ is (recall that $d_i = \tau_{i+1} - \tau_i$ for all $i \geq 0$):

$$(s_0, \mathbf{v}_0) \xrightarrow{\sigma_0} (s_1, \mathbf{v}'_1) \overset{d_0}{\rightsquigarrow} (s_1, \mathbf{v}_1) \xrightarrow{\sigma_1} \cdots \xrightarrow{\sigma_{m+1}} (s_{m+2}, \mathbf{v}'_{m+2}) .$$

Since $\mathcal{D}$ is total, there must be an edge $e_a = \langle s_{m+2}, s_a, \{Q\}, \lambda_a, \delta_a \rangle$ and $0 < \epsilon_a < \tau_k - \tau_{k-1}$ such that $(\mathbf{v}'_{m+2} + t - 1)$ satisfies δ_a for all $t \in [\tau_k + 1 - \epsilon_a, \tau_k + 1)$.[4] Choose

[4]Such ϵ_a must exist as the number of clocks in $\mathcal{D}$ is finite.

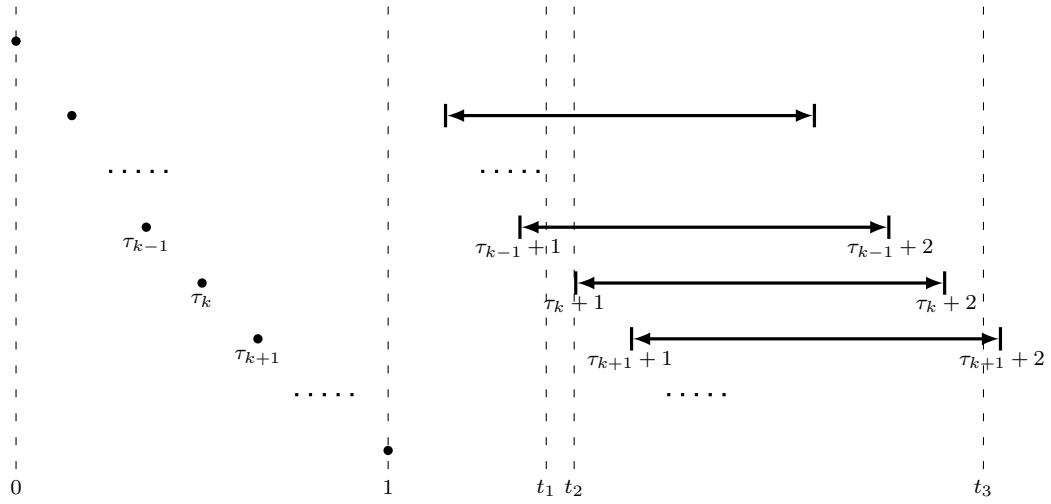


Figure 5.1: Obligations imposed by ρ.

$t_1 \in [\tau_k + 1 - \epsilon_a, \tau_k + 1)$ and let $\zeta = (\{Q\}, t_1)$, then the run of $\mathcal{D}$ on $\rho\zeta$ is:

$$(s_0, \mathbf{v}_0) \overset{\sigma_0}{\to} \cdots \overset{\sigma_{m+1}}{\to} (s_{m+2}, \mathbf{v}'_{m+2}) \overset{d_{m+1}}{\rightsquigarrow} (s_{m+2}, \mathbf{v}_{m+2}) \overset{\sigma_{m+2}}{\to} (s_a, \mathbf{v}'_a)\,.$$

Similarly, there is an edge $e_b = \langle s_a, s_b, \{Q\}, \lambda_b, \delta_b \rangle$ and $0 < \epsilon_b < \tau_{k+1} - \tau_k$ such that δ_b is satisfied by $(\mathbf{v}'_a + t - t_1)$ for all $t \in (\tau_k + 2, \tau_k + 2 + \epsilon_b]$. Choose $t_3 \in (\tau_k + 2, \tau_k + 2 + \epsilon_b]$ and let $\zeta' = (\{Q\}, t_1)(\{Q\}, t_3)$. The run of $\mathcal{D}$ on $\rho\zeta'$ is

$$(s_0, \mathbf{v}_0) \overset{\sigma_0}{\to} \cdots \overset{d_{m+1}}{\rightsquigarrow} (s_{m+2}, \mathbf{v}_{m+2}) \overset{\sigma_{m+2}}{\to} (s_a, \mathbf{v}'_a) \overset{d_{m+2}}{\rightsquigarrow} (s_a, \mathbf{v}_a) \overset{\sigma_{m+3}}{\to} (s_b, \mathbf{v}'_b)\,.$$

Note that s_b is accepting. For all suffixes $\zeta'' = (\{Q\}, t_1)(\{Q\}, t)$ where $\tau_{m+1} + 1 = 2 \leq t \leq \tau_k + 2$, the run of $\mathcal{D}$ on $\rho\zeta''$ should end in a non-accepting state since $\rho\zeta''$ is not a bad prefix. Clearly, e_b cannot be the last edge taken in the run on $\rho\zeta''$. Namely,

$$\begin{cases} (\mathbf{v}'_a + t - t_1) \not\models \delta_b & \text{if } t \in [2, \tau_k + 2] \\ (\mathbf{v}'_a + t - t_1) \models \delta_b & \text{if } t \in (\tau_k + 2, \tau_k + 2 + \epsilon_b] \end{cases}$$

By Proposition 5.2.1, there is a clock x and a clock constraint of the form $x > c$ in δ_b such that $(\mathbf{v}'_a + \tau_k + 2 - t_1)(x) = c$. Since $\tau_k + 2 - t_1 \notin \mathbb{N}$ and $\tau_k + 2 \notin \mathbb{N}$, it is not hard to see that the clock x had been reset at time instant τ_k.

We have shown that $\mathcal{D}$ resets a clock at time instant τ_k for each $0 < k < m + 1$. Suppose that $\mathcal{D}$ resets the same clock x on reading two distinct P-events in $(0, 1)$. Let τ_{k_1}, τ_{k_2} be the timestamps of the last pair of such P-events in $(0, 1)$. Along similar lines we have $\tau_{k_1} + 2 - \tau_{k_2} \in \mathbb{N}$. But the way we chose the timestamps of P-events

have made this impossible. This implies that $\mathcal{D}$ resets distinct clocks at τ_k for each $0 < k < m+1$. Since m is arbitrary, the number of clocks of $\mathcal{D}$ cannot be bounded. $\square$

The bottom line is that a monitor for bad prefixes has to remember the precise instants of all P-events. In particular, note that all the bad prefixes in the example above are informative. This implies that a ***bounded-variability*** assumption [Wil94] is essential for the monitoring of a logic strong enough to express the property above. For notational simplicity, henceforth in this chapter we simply assume that all input timed words have variability at most k_{var} for some absolute constant k_{var}, i.e., there are at most k_{var} events in any (open) unit time interval.

5.3 Path-Checking Algorithm

Recall that the model-checking problem is essentially a language inclusion problem: is the language of the model a subset of the language of the specification? A much more restricted case of this scheme is ***path checking***, i.e., the language of the model consists of a single word (or timed word). Formally, we define the path-checking problem for $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ as follows.

Definition 5.3.1 (The path-checking problem for $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$)**.** *Given a finite timed word ρ and an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula φ, is $\rho \models_f \varphi$?*

In this section, we give an (offline) path-checking algorithm for $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$. For given ρ and φ, the algorithm maintains a two-dimensional array `table` of $|\psi|$ rows and $|\rho|$ columns. Each row is used to store the truth values of a subformula at all positions. The algorithm proceeds by filling up the array `table` in a bottom-up manner, starting from minimal subformulas. We only detail the cases for $\varphi_1 \,\mathcal{U}_I\, \varphi_2$ and $\varphi_1 \,\mathfrak{U}^{\mathrm{c}}_I\, \varphi_2$ as other cases are either symmetric or trivial. In what follows, we write $x \leq I$ for $x < \sup(I)$ if I is right-open and for $x \leq \sup(I)$ otherwise. To ease the presentation we omitted the checks for array bounds, e.g., the code should stop when *ptr* (*ptr1*) reaches -1.

Algorithm 1 FILLTABLE($\texttt{table}, \varphi_1 \,\mathcal{U}_I\, \varphi_2$)

```
1  ptr ← |ρ| − 1
2  for j = |ρ| − 1 to 0 do
3      if ptr = j then
4          table[φ1 U_I φ2][ptr] ← ⊥
5          ptr ← ptr − 1
6      if table[φ2][j] then
7          if ptr = j − 1 then
8              table[φ1 U_I φ2][ptr] ← (τ_j − τ_ptr ∈ I)
9              ptr ← ptr − 1
10         while table[φ1][ptr + 1] ∧ τ_j − τ_ptr ≤ I do
11             if τ_j − τ_ptr ∈ I then
12                 table[φ1 U_I φ2][ptr] ← ⊤
13             else
14                 table[φ1 U_I φ2][ptr] ← ⊥
15             ptr ← ptr − 1
```

Proposition 5.3.2. *After executing* FILLTABLE($\texttt{table}, \varphi_1 \,\mathcal{U}_I\, \varphi_2$)*, we have*

$$\texttt{table}[\varphi_1 \,\mathcal{U}_I\, \varphi_2][i] \iff \rho, i \models_f \varphi_1 \,\mathcal{U}_I\, \varphi_2$$

for all $0 \leq i < |\rho|$ *if* $\texttt{table}[\varphi_1]$ *and* $\texttt{table}[\varphi_2]$ *were both correct.*

Proof. Suppose that $\texttt{table}[\varphi_1 \,\mathcal{U}_I\, \varphi_2][i] = \top$. Since each entry in $\texttt{table}[\varphi_1 \,\mathcal{U}_I\, \varphi_2]$ is filled exactly once, it must be filled at either line 8 or line 12. In the former case it is clear that $\rho, i \models_f \varphi_1 \,\mathcal{U}_I\, \varphi_2$. In the latter case, first observe that we must have $ptr \leq j - 2$. If $ptr = j - 2$ then we are done. So we assume $ptr < j - 2$. If there is a maximal position ptr', $ptr + 1 < ptr' < j$ such that $\texttt{table}[\varphi_1][ptr'] = \bot$, we must have $ptr + 1 = ptr'$, which is a contradiction. Therefore $\rho, i \models_f \varphi_1 \,\mathcal{U}_I\, \varphi_2$.

For the other direction assume $\rho, i \models_f \varphi_1 \,\mathcal{U}_I\, \varphi_2$ and $j' > i$ be the witness position, i.e., $\tau_{j'} - \tau_i \in I$, $\texttt{table}[\varphi_2][j'] = \top$ and $\texttt{table}[\varphi_1][j''] = \top$ for all j'', $i < j'' < j'$. Now consider $j = j'$. If $ptr \geq i$ then we are done. So we assume $ptr < i$. If we already have $\texttt{table}[\varphi_1 \,\mathcal{U}_I\, \varphi_2][i] = \bot$, then it must be the case that $\tau_{j'} - \tau_i \notin I$, which is a contradiction. Therefore we must have $\texttt{table}[\varphi_1 \,\mathcal{U}_I\, \varphi_2][i] = \top$. □

Algorithm 2 FILLTABLE($\texttt{table}, \varphi_1 \,\mathfrak{U}_I^c\, \varphi_2$)

```
1  ptr1, ptr2 ← |ρ| − 1
2  for j = |ρ| − 1 to 0 do
3      while τ_j − τ_ptr2 ≤ inf(I) − c ∨ table[φ1][ptr2] do
4          ptr2 ← ptr2 − 1
5      if ptr1 = j then
6          table[φ1 U^c_I φ2][ptr1] ← ⊥
7          ptr1 ← ptr1 − 1
8      if table[φ2][j] then
9          if ptr1 = j − 1 then
10             table[φ1 U^c_I φ2][ptr1] ← (τ_j − τ_ptr1 ∈ I)
11             ptr1 ← ptr1 − 1
12         while τ_j − τ_ptr1 ≤ I ∧ τ_ptr2 − τ_ptr1 ≤ c do
13             if τ_j − τ_ptr1 ∈ I then
14                 table[φ1 U^c_I φ2][ptr1] ← ⊤
15             else
16                 table[φ1 U^c_I φ2][ptr1] ← ⊥
17             ptr1 ← ptr1 − 1
```

Proposition 5.3.3. *After executing* FILLTABLE($\texttt{table}, \varphi_1 \,\mathfrak{U}_I^c\, \varphi_2$), *we have*

$$\texttt{table}[\varphi_1 \,\mathfrak{U}_I^c\, \varphi_2][i] \iff \rho, i \models_f \varphi_1 \,\mathfrak{U}_I^c\, \varphi_2$$

for all $0 \leq i < |\rho|$ *if* $\texttt{table}[\varphi_1]$ *and* $\texttt{table}[\varphi_2]$ *were both correct.*

Proof. Observe that after line 5, *ptr2* is equal to the maximal position such that $\tau_j - \tau_{ptr2} > a - c$ and $\texttt{table}[\varphi_1][ptr2] = \bot$. The proof is very similar to the case of $\varphi_1 \,\mathcal{U}_I\, \varphi_2$. □

Note that the time and space complexity of the algorithm (linear in $|\rho| \cdot |\varphi|$) do not depend on the variability of ρ. However, a bounded-variability assumption will be needed in the next section where we adapt the algorithm to work in an *online* fashion.

5.4 Monitoring Procedure

In this section, we describe a monitoring procedure for $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formulas (over $\mathbf{P}$) of the form

$$\hat{\varphi} = \Phi(\psi_1, \ldots, \psi_m)$$

where ψ_1, ..., ψ_m are *bounded* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formulas and Φ is an LTL formula. By Lemma 4.1.4 (on page 58), this fragment is already expressively complete for $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ (and therefore for $\mathsf{FO}[<, +\mathbb{Q}]$). The main idea is similar to the one used in the previous chapter: we introduce new propositions $\mathbf{Q} = \{Q_1, \ldots, Q_m\}$ that correspond to ψ_1, ..., ψ_m. In this way, we can monitor Φ as an LTL property over $\mathbf{Q}$.[5] Since these propositions correspond to bounded formulas, their truth values can be obtained by running the path-checking algorithm on subpaths: as all input (infinite) timed words have variability at most k_{var}, we only need to store a 'sliding window' of a certain size on the input timed word.

5.4.1 Untimed LTL Part

We recall briefly the standard methodology to construct finite automata that accept exactly the informative prefixes for a given $\mathsf{LTL}_{\mathsf{fut}}$ formula [KV01]. Given such a formula Ψ, first use a standard construction [Var96] to obtain a language-equivalent alternating Büchi automaton $\mathcal{A}_\Psi$. Then redefine its set of accepting states to be the empty set and treat it as an automaton over finite words. The resulting automaton $\mathcal{A}_\Psi^{true}$ accepts exactly all informative good prefixes for Ψ. One can further determinise $\mathcal{A}_\Psi^{true}$ with the usual subset construction. The same can be done for $\neg\Psi$ to obtain a deterministic automaton that accepts exactly the informative bad prefixes for Ψ.

In our case, we first translate the LTL formulas Φ and $\neg\Phi$ into a pair of *two-way* alternating Büchi automata [GO03]. With the same 'tweaks', we obtain two automata that accept informative good prefixes and informative bad prefixes for Φ. We then apply existing procedures that translate two-way alternating automata over finite words into deterministic automata, e.g., [Bir93]. In the rest of this chapter, we refer to the resulting automata as $\mathcal{D}_{good}$ and $\mathcal{D}_{bad}$. In order to detect the two types of prefixes simultaneously, we will execute $\mathcal{D}_{good}$ and $\mathcal{D}_{bad}$ in parallel.

Proposition 5.4.1. *For an* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *formula* $\hat{\varphi}$ *of the form described above, the automata* $\mathcal{D}_{good}$ *and* $\mathcal{D}_{bad}$ *have size* $2^{2^{O(|\Phi|)}}$ *where* Φ *is the 'backbone'* LTL *formula.*

[5] A similar idea is used in [FK09] to synthesise smaller monitor circuits for $\mathsf{LTL}_{\mathsf{fut}}$ formulas.

5.4.2 Bounded Metric Part

We now describe how to obtain the truth values of bounded formulas $\psi_1, \ldots, \psi_m$. In what follows, let $l_{fr}(\psi) = k_{var} \cdot \lceil fr(\psi) \rceil$ and $l_{pr}(\psi) = k_{var} \cdot \lceil pr(\psi) \rceil$ (the functions fr and pr were defined in Section 4.3 on page 66).

Naïve evaluation. Suppose that we want to obtain the truth value of ψ_i at position j in the input timed word $\rho = (\sigma, \tau)$. Since ψ_i is bounded, only the events occurring between $\tau_j - pr(\psi_i)$ and $\tau_j + fr(\psi_i)$ can affect the truth value of ψ_i at j. This implies that $\rho, j \models \psi_i \iff \rho', j \models_f \psi_i$ where ρ' is a prefix of ρ that contains all the events between $\tau_j - pr(\psi_i)$ and $\tau_j + fr(\psi_i)$ in ρ. Since ρ is of bounded variability k_{var}, there can be at most $l_{pr}(\psi_i) + 1 + l_{fr}(\psi_i)$ events between $\tau_j - pr(\psi_i)$ and $\tau_j + fr(\psi_i)$. It follows that we can simply 'record' all events in this interval with a two-dimensional array of $l_{pr}(\psi_i) + 1 + l_{fr}(\psi_i)$ columns and $1 + |\psi_i|$ rows: the first row is used to store the timestamps whereas the last $|\psi_i|$ rows are used to store the truth values. Intuitively, the two-dimensional array acts as a sliding window on ρ.

Now consider all the propositions in $\mathbf{Q}$. Their truth values at position j can be evaluated using a two-dimensional array of $l^{\mathbf{Q}}_{pr} + 1 + l^{\mathbf{Q}}_{fr}$ columns and $1 + |\psi_1| + \cdots + |\psi_m|$ rows where $l^{\mathbf{Q}}_{pr} = \max\limits_{1 \le i \le m} l_{pr}(\psi_i)$ and $l^{\mathbf{Q}}_{fr} = \max\limits_{1 \le i \le m} l_{fr}(\psi_i)$. Each row can be filled in time $O\big(k_{var} \cdot (l^{\mathbf{Q}}_{pr} + 1 + l^{\mathbf{Q}}_{fr})\big)$ using the path-checking algorithm. Overall, we need a two-dimensional array of size $O(k_{var} \cdot c_{sum} \cdot |\hat{\varphi}|)$ where c_{sum} is the sum of the constants in $\hat{\varphi}$; for each position j, we need time $O(k_{var} \cdot c_{sum} \cdot |\hat{\varphi}|)$ to obtain the truth values of all propositions in $\mathbf{Q}$.

Incremental evaluation. While the method we just described uses only bounded space, it is clearly inefficient as for each j, we have to fill the whole two-dimensional array from scratch. This is because the filled entries, as they are computed without the knowledge of the events outside of the interval, may become incorrect later. We now describe an optimisation which allows effective reuse of previously filled entries. The idea is to regard entries that depend on future events as 'unknown' and not fill them right away. By construction, these unknown entries are not needed in the evaluation.

We first deal with the simpler case of past subformulas. Observe that as the path-checking algorithm is filling a row for $\varphi_1 \, \mathcal{S}_I \, \varphi_2$ or $\varphi_1 \, \mathfrak{S}^c_I \, \varphi_2$, the variables *ptr*, *ptr1* and *ptr2* all increases *monotonically*. This implies that for past subformulas, we can use the path-checking algorithm in an online manner: simply suspend the path-checking algorithm when we have filled all entries using the truth values of φ_1

and φ_2 (at various positions) that are currently known, and resume the algorithm when the truth values of φ_1 and φ_2 (at some other positions) that are previously unknown become available.

The case of future subformulas is more involved. Suppose that we want to evaluate the truth value of a subformula $P_1 \,\mathcal{U}_{(a,b)}\, P_2$ at position j in the input timed word $\rho = (\sigma, \tau)$. It is clear that the truth value may depend on future events if $\tau_j + b$ is greater than the timestamp of the last acquired event. However, observe that even when this is the case, we may still do the evaluation if any of the following holds:

- P_1 does not hold at some position j' such that $\tau_{j'}$ is less or equal than the timestamp of the last acquired event. In this case, we know that all the truth values of $P_1 \,\mathcal{U}_{(a,b)}\, P_2$ at positions $< j'$ cannot depend on the events at positions $> j'$.

- P_2 holds at some position $j' > j$ and P_1 holds at all positions j'', $j < j'' < j'$. In this case, the truth values of $P_1 \,\mathcal{U}_{(a,b)}\, P_2$ at positions $k < j'$ where $\tau_{j'} - \tau_k \in (a, b)$ do not depend on the events at positions $> j'$.

We generalise this observation to handle the general case of $\varphi_1 \,\mathcal{U}_{(a,b)}\, \varphi_2$. First of all, we keep an index j_φ to the first unknown entry in the row corresponding to φ for each subformula φ. Let $t_{max} = \min(\tau_{(j_{\varphi_1}-1)}, \tau_{(j_{\varphi_2}-1)})$ and update its value as j_{φ_1} or j_{φ_2} are changed. Whenever t_{max} is updated to a new value, we also update the following indices:

- j_1 is the maximal position such that $\tau_{j_1} + b \leq t_{max}$.

- j_2 is the maximal position such that $\tau_{j_2} \leq t_{max}$ and φ_2 holds at j_2.

- j_3 is the maximal position such that $\tau_{j_3} + a < \tau_{j_2}$.

- j_4 is the maximal position such that $\tau_{j_4} \leq t_{max}$ and φ_1 does not hold at j_4.

Now, if any of $j_1, j_3, j_4 - 1$ is updated to a value greater than $j_{\varphi_1 \mathcal{U}_I \varphi_2}$, let the maximal such value be j_5 and start Algorithm 1 (on page 78) from line 3 with $ptr = j_5$ and $j = j_2$. We run the algorithm till all the entries at positions $< j_5$ in the row corresponding to $\varphi_1 \,\mathcal{U}_I\, \varphi_2$ have been filled.

The crucial observation here is that j_1, j_2, j_3, j_4 all increase monotonically, and therefore can be maintained using time linear in the length of ρ. Also observe that the truth value of any subformula at any position will be filled only once. The case of $\varphi_1 \mathfrak{U}^c_{(a,b)} \varphi_2$ is similar (albeit more involved): we let $t_{max} = \min(\tau_{(j_{\varphi_1}-1)} + (a - c), \tau_{(j_{\varphi_2}-1)})$

and maintain some different indices. These observations imply that each entry in the two-dimensional array can be filled in amortised constant time. Assuming that moving a token on a deterministic finite automaton takes constant time, we can state the following theorem.

Theorem 5.4.2. *For an* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *formula* $\hat{\varphi}$ *of the form described earlier and an infinite timed word of bounded variability* k_{var}*, our monitoring procedure uses two DFAs of size* $2^{2^{O(|\Phi|)}}$*, a two-dimensional array of size* $O(k_{var} \cdot c_{sum} \cdot |\hat{\varphi}|)$ *where* c_{sum} *is the sum of the constants in* $\hat{\varphi}$*, and amortised time* $O(|\hat{\varphi}|)$ *per event.*

5.4.3 Correctness

We now show that our procedure is sound and complete for detecting informative prefixes. The following proposition is immediate since the three views of the truncated semantics coincide in this case.

Proposition 5.4.3. *For a bounded* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *formula* ψ*, a finite timed word* $\rho = (\sigma, \tau)$ *and a position* $0 \leq i < |\rho|$ *such that* $\tau_i + fr(\psi) \leq \tau_{|\rho|-1}$ *and* $\tau_i - pr(\psi) \geq 0$*, we have*

$$\rho, i \models_f^+ \psi \iff \rho, i \models_f \psi \iff \rho, i \models_f^- \psi \,.$$

The following proposition says 'something good remains good' and 'something bad remains bad'.

Proposition 5.4.4. *For an* $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ *formula* φ*, a finite timed word* ρ *and a position* i *in* ρ*, if* ρ *is a prefix of a longer finite timed word* ρ'*, then*

$$\rho, i \models_f^+ \varphi \implies \rho', i \models_f^+ \varphi \text{ and } \rho, i \not\models_f^- \varphi \implies \rho', i \not\models_f^- \varphi \,.$$

Theorem 5.4.5 (Soundness)**.** *In our procedure, if we ever reach an accepting state of* $\mathcal{D}_{good}$ *(*$\mathcal{D}_{bad}$*) via a finite word* $u \in \Sigma_Q^*$*, then we must eventually read an informative good (bad) prefix for* $\hat{\varphi}$*.*

Proof. For such u and a corresponding $\rho = (\sigma, \tau)$ such that $\tau_{|u|-1} + l_{fr}^Q \leq \tau_{|\rho|-1}$,

$$\forall i \in [0, |u|) \left((u, i \models_f^+ \Psi \implies \rho, i \models_f^+ \psi) \wedge (u, i \not\models_f^- \Psi \implies \rho, i \not\models_f^- \psi) \right)$$

where Ψ is a subformula of Φ and $\psi = \Psi(\psi_1, \ldots, \psi_m)$. This can easily be proved by structural induction and Proposition 5.4.3. If u is accepted by $\mathcal{D}_{good}$, we have $u, 0 \models_f^+ \Phi$

by construction. By the above we have $\rho, 0 \models_f^+ \Phi(\psi_1, \ldots, \psi_m)$, as desired. The case of $\mathcal{D}_{bad}$ is symmetric. □

Theorem 5.4.6 (Completeness)**.** *Whenever we read an informative good (bad) prefix $\rho = (\sigma, \tau)$ for $\hat{\varphi}$, $\mathcal{D}_{good}$ ($\mathcal{D}_{bad}$) must eventually reach an accepting state.*

Proof. For the finite word u' obtained a bit later with $|u'| = |\rho|$,

$$\forall i \in [0, |u'|) \left((\rho, i \models_f^+ \psi \implies u', i \models_f^+ \Psi) \wedge (\rho, i \not\models_f^- \psi \implies u', i \not\models_f^- \Psi) \right)$$

where Ψ is a subformula of Φ and $\psi = \Psi(\psi_1, \ldots, \psi_m)$. This can be proved by structural induction and Proposition 5.4.4. The theorem follows. □

5.5 Preservation of Informative Prefixes

As we have seen in Example 5.1.6 and 5.1.7 (on page 74), it is possible for two equivalent $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formulas to have different sets of informative good/bad prefixes. In this section, we show that this is cannot be the case when the two formulas are related by one of the rewriting rules in Section 4.1 (on page 54). In other words, the rewriting process outlined in Section 4.1 preserves the 'informativeness' of prefixes.

Lemma 5.5.1. *For an $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula φ, let φ' be the formula obtained from φ by applying one of the rules in Section 4.1 on some of its subformula. We have*

$$\rho \models_f^+ \varphi \iff \rho \models_f^+ \varphi' \text{ and } \rho \models_f^- \varphi \iff \rho \models_f^- \varphi' .$$

Given the lemma above, we can state the following theorem on any MTL formula φ and the equivalent formula $\hat{\varphi}$ (of our desired form) obtained from φ by applying the rewriting rules in Section 4.1.

Theorem 5.5.2. *The set of informative good prefixes of φ coincides with the set of informative good prefixes of $\hat{\varphi}$. The same holds for informative bad prefixes.*

We now have a way to monitor the informative good/bad prefixes for an arbitrary $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula φ: use the rewriting rules to obtain $\hat{\varphi}$ and apply the monitoring procedure we described in the last section. This suggests that a monitor for informative prefixes only needs a bounded amount of memory, even for complicated $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formulas with arbitrary nestings of (bounded and unbounded) past and future operators. This is somewhat surprising as the truth value of such a formula at some point seems

to depend on an arbitrary amount of information in the past (e.g., this is the case for the procedure in [BN12]).

Proof of Lemma 5.5.1. Since the satisfaction relations are defined inductively, we can work directly on the relevant subformula. We would like to prove that for a finite timed word ρ and a position i in ρ,

$$\rho, i \models_f^+ \phi \iff \rho, i \models_f^+ \phi' \text{ and } \rho, i \models_f^- \phi \iff \rho, i \models_f^- \phi'$$

where $\phi \iff \phi'$ matches one of the rules in Section 4.1. For a group of similar rules we will only prove a representative one, as the proof for others follow similarly. In the following we let the LHS be ϕ and RHS be ϕ'.

- $\varphi_1 \,\mathcal{U}_{(a,\infty)}\, \varphi_2 \iff \varphi_1 \,\mathcal{U}\, \varphi_2 \wedge \Box_{(0,a]}(\varphi_1 \wedge \varphi_1 \,\mathcal{U}\, \varphi_2)$:

  - $\rho, i \models_f^+ \phi \iff \rho, i \models_f^+ \phi'$:
    Assume $\rho, i \models_f^+ \phi$. By definition we have $\rho, i \models_f^+ \varphi_1 \,\mathcal{U}\, \varphi_2$. If there is no event in $(\tau_i, \tau_i + a]$, since there must be an event in $(\tau_i + a, \tau_{|\rho|-1}]$, we are done. If there are events in $(\tau_i, \tau_i + a]$, then for all j such that $\tau_j - \tau_i \in (0, a]$ we have $\rho, j \not\models_f^- \neg\varphi_1$. Also for all such j we have $\rho, j \not\models_f^- \neg\varphi_1 \,\mathcal{U}\, \varphi_2$ since it is obvious that $\rho, j \models_f^+ \varphi_1 \,\mathcal{U}\, \varphi_2$. For the other direction, if the witness (for $\rho, i \models_f^+ \varphi_1 \,\mathcal{U}\, \varphi_2$) is in $(\tau_i + a, \tau_{|\rho|-1})$ then we are done. If this is not the case, since $\rho, i \not\models_f^- \Diamond_{(0,a]}\big(\neg\varphi_1 \vee \neg(\varphi_1 \,\mathcal{U}\, \varphi_2)\big)$, we must have $\tau_{|\rho|-1} \geq a$. Now for all j such that $\tau_j - \tau_i \in (0, a]$ we have $\rho, j \models_f^+ \varphi_1$ and $\rho, j \models_f^+ \varphi_1 \,\mathcal{U}\, \varphi_2$, which imply $\rho, i \models_f^+ \phi$.
  - $\rho, i \models_f^- \phi \iff \rho, i \models_f^- \phi'$:
    Assume $\rho, i \models_f^- \phi$. This holds if there is a witness in (a, ∞) or $\rho, i \models_f^- \Box\varphi_1$. In both cases we have $\rho, i \models_f^- \varphi_1 \,\mathcal{U}\, \varphi_2$. If there is no event in $(\tau_i, \tau_i + a]$ then we are done. If there is a witness, then for all such j that $\tau_j - \tau_i \in (0, a]$ we have $\rho, j \models_f^- \varphi_1$ and $\rho, j \models_f^- \varphi_1 \,\mathcal{U}\, \varphi_2$. If there is no witness then for all such j we again have $\rho, j \models_f^- \varphi_1$ and $\rho, j \models_f^- \varphi_1 \,\mathcal{U}\, \varphi_2$. For the other direction, if there is no event in $(\tau_i, \tau_i + a]$ we are done. If there are events in $(\tau_i, \tau_i + a]$, all j such that $\tau_j - \tau_i \in (0, a]$ will satisfy $\rho, j \models_f^- \varphi_1$ and $\rho, j \models_f^- \varphi_1 \,\mathcal{U}\, \varphi_2$. This clearly gives $\rho, i \models_f^- \phi$.

- $\varphi_1 \,\mathfrak{U}^c_{(a,\infty)}\, \varphi_2 \iff \varphi_1 \,\mathfrak{U}^c_{(a,2a]}\, \varphi_2 \vee \Big(\Diamond^w_{[0,c]}\big(\varphi_1 \,\mathcal{U}_{(a,\infty)}\, (\varphi_2 \vee \Diamond_{\leq a-c}\varphi_2)\big)\Big)$:
  The proof is very similar to the proof of Proposition 4.1.1 (on page 55).

- $\neg(\varphi_1 \,\mathcal{U}\, \varphi_2) \iff \Box\neg\varphi_2 \vee \big(\neg\varphi_2 \,\mathcal{U}\, (\neg\varphi_2 \wedge \neg\varphi_1)\big)$:

  - $\rho, i \models^+_f \phi \iff \rho, i \models^+_f \phi'$:
    Assume $\rho, i \models^+_f \phi \iff \rho, i \not\models^-_f \varphi_1 \,\mathcal{U}\, \varphi_2$. This implies that φ_1 fails to hold before φ_2 holds, and we have $\rho, i \models^+_f \neg\varphi_2 \,\mathcal{U}\, (\neg\varphi_2 \wedge \neg\varphi_1)$. For the other direction note that $\rho, i \not\models^+_f \Box\neg\varphi_2$, the second disjunct must be satisfied, and it is easy to see that $\rho, i \models^+_f \phi$.
  - $\rho, i \models^-_f \phi \iff \rho, i \models^-_f \phi'$:
    Assume $\rho, i \models^-_f \neg(\varphi_1 \,\mathcal{U}\, \varphi_2) \iff \rho, i \not\models^+_f \varphi_1 \,\mathcal{U}\, \varphi_2$. This implies either $\rho, j \not\models^+_f \varphi_2 \iff \rho, j \models^-_f \neg\varphi_2$ for all $j > i$ in ρ (this gives $\rho, i \models^-_f \Box\neg\varphi_2$) or φ_1 fails to hold before φ_2 holds—$\rho, i \models^-_f \neg\varphi_2 \,\mathcal{U}\, (\neg\varphi_2 \wedge \neg\varphi_1)$. For the other direction, if $\rho, i \models^-_f \Box\neg\varphi_2 \iff \rho, i \not\models^+_f \Diamond\varphi_2$ then $\rho, i \models^+_f \varphi_1 \,\mathcal{U}\, \varphi_2$ cannot hold. If $\rho, i \models^-_f \neg\varphi_2 \,\mathcal{U}\, (\neg\varphi_2 \wedge \neg\varphi_1)$ then either $\rho, i \models^-_f \Box\neg\varphi_2$ or there is a witness, and it is easy to see that $\rho, i \models^+_f \varphi_1 \,\mathcal{U}\, \varphi_2$ cannot hold.

- $\theta \,\mathcal{U}_{(a,b)}\, \big((\varphi_1 \,\mathcal{U}\, \varphi_2) \wedge \chi\big) \iff \theta \,\mathcal{U}_{(a,b)}\, \big((\varphi_1 \,\mathcal{U}_{(0,2b)}\, \varphi_2) \wedge \chi\big)$
  $\vee \Big(\big(\theta \,\mathcal{U}_{(a,b)}\, (\Box_{(0,2b)}\varphi_1 \wedge \chi)\big) \wedge \varphi_{ugb}\Big)$:
  The proof is very similar to the proof of Proposition 4.1.2 (on page 56).

- $\big((\varphi_1 \,\mathcal{U}\, \varphi_2) \vee \chi\big) \,\mathcal{U}_{(a,b)}\, \theta \iff \big((\varphi_1 \,\mathcal{U}_{(0,2b)}\, \varphi_2) \vee \chi\big) \,\mathcal{U}_{(a,b)}\, \theta$
  $\vee \Bigg(\Big(\big((\varphi_1 \,\mathcal{U}_{(0,2b)}\, \varphi_2) \vee \chi\big) \,\mathcal{U}_{(0,b)}\, (\Box_{(0,2b)}\varphi_1)\Big) \wedge \Diamond_{(a,b)}\theta \wedge \varphi_{ugb}\Bigg)$:

  - $\rho, i \models^+_f \phi \iff \rho, i \models^+_f \phi'$:
    Assume $\rho, i \models^+_f \phi$. It is obvious that $\rho, i \models^+_f \Diamond_{(a,b)}\theta$ holds. If the first disjunct of ϕ' does not hold, then $\rho, i \models^+_f \big((\varphi_1 \,\mathcal{U}_{(0,2b)}\, \varphi_2) \vee \chi\big) \,\mathcal{U}_{(0,b)}\, (\Box_{(0,2b)}\varphi_1)$ must hold. The last conjunct holds by an argument similar to the proof of Proposition 4.1.2. For the other direction, if the first disjunct of ϕ' holds then we are done. If it does not hold, then there must be a witness (at which φ_2 holds) in $[\tau_i + 2b, \tau_{|\rho|-1}]$, and it is easy to see that $\rho, i \models^+_f \phi$.
  - $\rho, i \models^-_f \phi \iff \rho, i \models^-_f \phi'$:
    Assume $\rho, i \models^-_f \phi$. If the first disjunct of ϕ' does not hold then there must

be events in $[\tau_i + 2b, \tau_{|\rho|-1}]$. It follows that $\rho, i \models_f^- \big((\varphi_1 \,\mathcal{U}_{(0,2b)}\, \varphi_2) \vee \chi\big) \,\mathcal{U}_{(0,b)}$ $(\Box_{(0,2b)}\varphi_1)$ and $\rho, i \models_f^- \Diamond_{(a,b)}\theta$ must hold. The rest is similar to the proof to Proposition 4.1.2. For the other direction, if the first disjunct of ϕ' holds then we are done. Otherwise if $\tau_{|\rho|-1} < b$, it is easy to see that $\rho, i \models_f^- \phi$. If this is not the case then the proof again follows Proposition 4.1.2. □

5.6 An Informative Fragment

As we have also seen in Example 5.1.6 (on page 74), is possible that some of the bad prefixes for an MTL[$\mathfrak{U}, \mathfrak{S}$] formula φ are not informative. To avoid such an situation, we can impose certain syntactic restrictions on φ. In particular, we show in this section that all bad prefixes for Safety-MTL [OW06b] formulas inevitably extend to informative bad prefixes.[6] That is, we can detect all bad prefixes for Safety-MTL formulas, whether informative or not, with our monitoring procedure (albeit possibly with some delays).

We say an MTL formula φ is in *positive normal form*[7] if all the negations in φ apply directly to propositions. Using syntactic sugars, we can easily rewrite any MTL formula into positive normal form. Safety-MTL is defined as the fragment of MTL such that, when written in positive normal form, all subformulas $\varphi_1 \,\mathcal{U}_I\, \varphi_2$ are bounded (there are no restrictions on other types of subformulas, e.g., $\varphi_1 \widetilde{\mathcal{U}}_I \varphi_2$ and $\varphi_1 \,\mathcal{S}_I\, \varphi_2$).

Proposition 5.6.1. *For a* Safety-MTL *formula φ, an infinite timed word ρ and a position i in ρ, $\rho, i \not\models \varphi$ implies that there is a prefix ρ' of ρ such that for any infinite timed word $\rho'\zeta$ we have $\rho'\zeta, i \not\models \varphi$.*

Proof. Simple induction on the structure of φ. □

Proposition 5.6.2. *For a* Safety-MTL *formula φ, a finite timed word ρ and $i \geq 0$, if $\rho\zeta, i \not\models \varphi$ for all ζ, then for any ζ there exists a prefix ρ' of $\rho\zeta$ such that $\rho', i \models_f^+ \neg\varphi$.*

Proof. We prove this by structural induction. The base step is trivial. For the induction step:

- $\varphi_1 \vee \varphi_2$:
  For any ζ we have $\rho\zeta, i \not\models \varphi_1$ and $\rho\zeta, i \not\models \varphi_2$. Hence for any ζ there is a ρ' such that $\rho', i \models_f^+ \neg\varphi_1$ and $\rho', i \models_f^+ \neg\varphi_2$ (by the induction hypothesis and

[6] In the words of Kupferman and Vardi [KV01], all Safety-MTL properties are either *intentionally safe* or *accidentally safe*.

[7] Not to be confused with the normal form introduced in Section 4.1 (on page 54).

Proposition 5.4.4 (on page 83)). We then have $\rho', i \models^+_f \neg\varphi_1 \wedge \rho', i \models^+_f \neg\varphi_2 \iff \rho', i \models^+_f (\neg\varphi_1 \wedge \neg\varphi_2) \iff \rho', i \models^+_f \neg(\varphi_1 \vee \varphi_2)$.

- $\varphi_1 \wedge \varphi_2$:
  For any ζ we have $\rho\zeta, i \not\models \varphi_1$ or $\rho\zeta, i \not\models \varphi_2$. Hence for any ζ there is a ρ' such that $\rho', i \models^+_f \neg\varphi_1$ or $\rho', i \models^+_f \neg\varphi_2$ (by the induction hypothesis). We then have $\rho' \not\models_f \varphi_1 \vee \rho' \not\models_f \varphi_2 \iff \rho' \not\models_f \varphi_1 \wedge \varphi_2 \iff \rho' \models^+_f \neg(\varphi_1 \wedge \varphi_2)$.

- $\varphi_1 \,\mathcal{U}_I\, \varphi_2$:
  $\rho\zeta, i \not\models \varphi_1 \,\mathcal{U}_I\, \varphi_2$ for any ζ iff for any ζ, φ_1 fails to hold before φ_2 holds or there is no φ_2 (later than τ_i) in I. By the non-Zenoness of $\rho\zeta$ and the fact that I is bounded, in the first case we have, for a given ζ, a prefix ρ' of $\rho\zeta$ such that $\exists j \in (i, |\rho'| - 1]\,\big(\tau_j - \tau_i \leq I \wedge \rho', j \models^+_f \neg\varphi_1 \wedge \forall l \in (i, j]\,(\rho', l \models^+_f \neg\varphi_2)\big)$ by the induction hypothesis. In the second case we have ρ' such that $\forall j \in (i, |\rho'| - 1]\,(\tau_j - \tau_i \in I \implies \rho', j \models^+_f \neg\varphi_2)$. In either case it is easy to see that $\rho', i \models^+_f \neg(\varphi_1 \,\mathcal{U}_I\, \varphi_2)$.

- $\varphi_1 \widetilde{\mathcal{U}}_I \varphi_2$: $\rho\zeta, i \not\models \varphi_1 \widetilde{\mathcal{U}}_I \varphi_2$ for any ζ iff for any ζ we have $\rho\zeta, i \models \neg\varphi_1 \,\mathcal{U}_I\, \neg\varphi_2$. We have, for a given ζ, a prefix ρ' of $\rho\zeta$ such that $\exists j \in (i, |\rho'| - 1]\,\big(\tau_j - \tau_i \in I \wedge \rho', j \models^+_f \neg\varphi_2 \wedge \forall l \in (i, j)\,(\rho', l \models^+_f \neg\varphi_1)\big)$ by the induction hypothesis. Now, of course, $\rho', i \models^+_f \neg\varphi_1 \,\mathcal{U}_I\, \neg\varphi_2 \iff \rho', i \models^+_f \neg(\varphi_1 \widetilde{\mathcal{U}}_I \varphi_2)$.

The cases for $\varphi_1 \,\mathcal{S}_I\, \varphi_2$ and $\varphi_1 \widetilde{\mathcal{S}}_I \varphi_2$ are exactly similar. □

5.7 Discussion

We identified an 'easy-to-monitor' fragment of $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$, for which we proposed an efficient trace-length independent monitoring procedure. This fragment is much more expressive than the fragments previously considered in runtime verification literature; in fact, it is already expressively complete for $\mathsf{FO}[<, +\mathbb{Q}]$. Moreover, we showed that informative good/bad prefixes are preserved by the syntactic rewriting rules in Section 4.1 (on page 54). It follows that the informative good/bad prefixes for an arbitrary $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula can be monitored in a trace-length independent fashion, overcoming a long-standing barrier to the runtime verification of real-time properties.

An practical issue is that for an arbitrary $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ formula, the syntactic rewriting process could potentially induce a non-elementary blow-up. However, such behaviour does not seem to be realised in practice. In our experience, the resulting formula is

often of comparable size to the original specification, which itself is typically small. For example, consider the following formula:

$$\Box\big(\texttt{ChangeGear} \implies \Diamond_{(0,30)}(\texttt{InjectFuel} \wedge \Diamond\!\!\!-\texttt{InjectLubricant})\big)\,.$$

The resulting formula after rewriting is

$$\begin{aligned}\Box\big(\texttt{ChangeGear} \implies \Diamond_{(0,30)}(\texttt{InjectFuel} \wedge \Diamond\!\!\!-_{(0,30)}\texttt{InjectLubricant}) \\ \vee(\Diamond_{(0,30)}\texttt{InjectFuel} \wedge \Diamond\!\!\!-\texttt{InjectLubricant})\big)\,.\end{aligned}$$

Indeed, it can be argued that most common real-time specification patterns [KC05] belong syntactically to our 'easy-to-monitor' fragment and thus need no rewriting. Another way to alleviate the issue is to allow more liberal syntax (or more derived operators) to make writing specifications easier; no blow-up occurs as long as they can be handled efficiently in the 'sliding window'. For example, the procedures described in Section 5.4.2 (on page 81) can handle subformulas with unbounded past without modification.

To detect informative bad prefixes, our monitoring procedure uses a deterministic finite automaton of size doubly-exponential in the size of the input formula. While such a blow-up is rarely a problem in practice (see [BLS11, Section 2.5]), it would be better if it could at all be avoided. In the untimed setting, it is known that if a safety property can be written as an LTL formula, then it is equivalent to a formula of the form $\Box\psi$ where ψ is a past-only LTL formula [LPZ85]. Therefore, if we restrict our attention to safety properties,[8] it is sufficient to consider formulas of this form, for which there is an efficient monitoring procedure that runs in linear time (for each event) and linear space [HR01]. Unfortunately, the question of whether the corresponding result holds for MTL (or similar metric temporal logics) is still open.

Our procedure detects only *informative* good/bad prefixes, which themselves can be seen as easily-checkable certificates for the fulfilment/violation of the property. Whilst we believe this limitation is in no way severe—indeed, the limitation is implicit in almost all current approaches to monitoring real-time properties—there might be certain practical scenarios where detecting *all* good/bad prefixes is preferred. We could have used two deterministic finite automata that detect all good/bad prefixes for the backbone LTL formula, but still they cannot detect all good/bad prefixes for the

[8] Note that there exist LTL formulas which are neither safety nor co-safety yet still monitorable; see [BLS11, Lemma 3.8] for an example.

whole formula (consider Example 5.1.6 on page 74). A working yet limited approach is as follows: we first translate the property into a non-deterministic timed automaton (assuming this is possible). Then, under a bounded-variability assumption, we can determinise this timed automaton on-the-fly. To see whether the current prefix is a good/bad prefix, we run a universality/emptiness check from the current state in the determinised timed automaton. However, each of the steps above is very expensive and therefore unlikely to be practical for monitoring, in which execution speed is critical. We leave as future work the search for a better methodology.

Finally, we remark that the (offline) path-checking problem is of independent theoretical interest [MS03]. It is known that the path-checking problem for LTL [KF09] and MTL [BO14] are both in $\mathrm{AC}^1[\log \mathrm{DCFL}]$, yet their precise complexity is still open. It would be interesting to see whether the construction for MTL in [BO14] carries over to $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$.

Part II

The Cyclic-Routing UAV Problem: Complexity and Implementation

Chapter 6

The Complexity of the Cyclic-Routing UAV Problem

In this chapter, we show that the CR-UAV Problem is PSPACE-complete even in the single-UAV case, regardless of whether constants are encoded succinctly; this contradicts an incorrect claim in the literature.

We first formally define the problem as the emptiness problem on a network of timed automata, from which PSPACE-membership follows immediately. Then we show that the period of a solution can be exponential in the magnitude of the largest constant, contradicting a relevant claim in the literature. Finally, we detail a reduction from the PERIODIC SAT Problem, known to be PSPACE-complete.

6.1 Preliminaries

6.1.1 Scenario

Let there be a set of targets and a number of identical UAVs. Each target has a ***relative deadline***: an upper bound requirement on the time between successive visits by UAVs. The UAVs are allowed to fly freely between targets, with a ***flight time*** given for each pair of targets: the least amount of time required for a UAV to fly from one of the targets to the other. We assume that flight times are symmetric, that they obey the triangle inequality, and that the flight time from target v to target v' is zero iff v and v' denote the same target. In other words, flight times are a metric on the set of targets. The goal is to decide whether there is a way to coordinate UAVs such that no relative deadline is ever violated. We make a few further assumptions:

- The first visit to each target must take place at the latest by the expiration time of its relative deadline.
- More than one UAV may visit the same target simultaneously.
- Time units are chosen so that all relative deadlines and flight times are integers, and moreover all relative deadlines are interpreted as closed constraints (i.e., using non-strict inequalities).

6.1.2 Modelling via Networks of Timed Automata

We now formally define the CR-UAV Problem in terms of the existence of infinite non-Zeno runs in a network of timed automata.

Consider $G = \langle V, RD, FT \rangle$ where V is a set of $n \geq 2$ vertices, a strictly positive integer weight $RD(v)$ is assigned to each vertex $v \in V$ (the relative deadline of target v) and a strictly positive integer weight $FT(v, v')$ is assigned to each pair of vertices (v, v') with $v \neq v'$ (the flight time from v to v'). In addition we let $FT(v, v) = 0$ for all $v \in V$ and require that FT be symmetric and satisfy the triangle inequality. Intuitively, G can be seen as a (vertex- and edge-) weighted undirected clique. An instance of the CR-UAV Problem is specified by such a clique G and $k < n$ (the number of UAVs).

For each such instance, we can construct a corresponding network $\mathcal{A}_{G,k}$ of k timed automata according to the scenario we described earlier. Intuitively, a particular timed automaton is ascribed to each UAV, and for each target there is a single clock (shared by all UAV-automata) that keeps track of time elapsed since the last visit by some UAV. Each UAV-automaton keeps track of the location of its associated UAV, and enforces flight times by means of a single clock, which is reset the instant the UAV visits any target. A designated initial location allows the UAV to start its route at any target and any time. Finally, suitable invariants are imposed to assure that no relative deadline is violated.

We now give an example using the clique G illustrated in Figure 1.3 (on page 9). Note in particular that in each UAV-automaton there are self-loops on locations that correspond to targets (they will play a significant role in the next chapter). To simplify the notation, in the rest of this chapter we assume $V = \{0, \ldots, n-1\}$ and use zero-based matrices to represent RD and FT.

Example 6.1.1. Consider $G = \langle V, RD, FT \rangle$ where $V = \{0, 1, 2\}$, $RD = \begin{bmatrix} 10 \\ 5 \\ 10 \end{bmatrix}$, $FT = \begin{bmatrix} 0 & 3 & 8 \\ 3 & 0 & 5 \\ 8 & 5 & 0 \end{bmatrix}$ and $k = 2$. One of the two UAV-automata comprising $\mathcal{A}_{G,k}$ is illustrated in Figure 6.1 where all the locations have the invariant $TC_0 \leq 10 \wedge TC_1 \leq 5 \wedge TC_2 \leq 10$ and all the edges also reset clock UC.

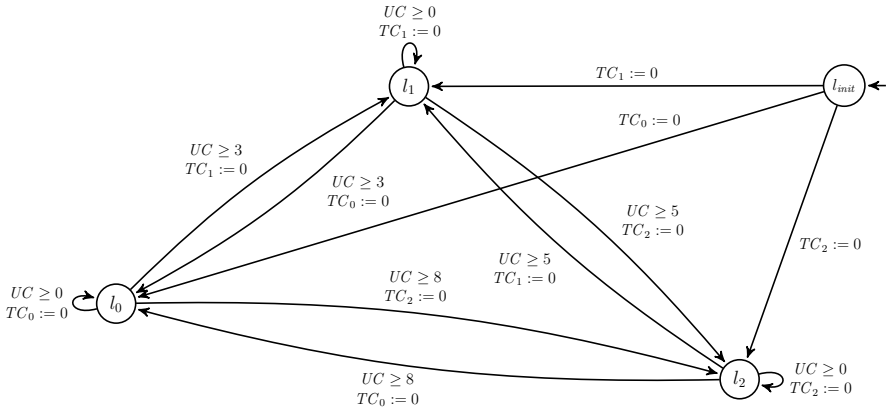


Figure 6.1: A UAV-automaton.

Observe that the presence of self-loops is crucial in this example: there are Zeno runs in the constructed network $\mathcal{A}_{G,k}$, yet no such run is possible if we remove all the self-loops in UAV-automata.

The problem is defined as the existence of non-Zeno runs in $\mathcal{A}_{G,k}$. By [Alu98, Theorem 7], this can be decided in PSPACE.

Definition 6.1.2 (The CR-UAV Problem)**.** *Given G and k as described above, is there a non-Zeno run in $\mathcal{A}_{G,k}$?*

It is worth noting that, since all timing constraints are closed by assumption, standard digitisation results apply (cf. [HMP92]) and it is sufficient to consider integer (i.e., discrete) time.[1]

6.1.3 The Single-UAV Case

Suppose that we are given an instance of the CR-UAV Problem with $k = 1$. In this case, we may clearly assume that the UAV never 'lingers' at any given target, i.e., the edges

[1]Strictly speaking, for digitisation to work we should allow *weakly monotonic* infinite timed words. However, to keep the presentation consistent we neglect this difference here.

between targets are taken exactly when UC is equal to the corresponding flight times, and the UAV never takes self-loops. Thus, a non-Zeno run in $\mathcal{A}_{G,k}$ readily corresponds to an infinite path in G in which each target is visited infinitely often at time intervals never greater than the target's relative deadline (recall that cliques have no self-loop). Following this, we now give a discrete graph-based (and timed-automaton independent) formulation of the problem specialised to the single-UAV case of the problem.

Given a finite path u in G, the *duration* of u, which we denote by $dur(u)$, is defined to be the sum of the weights of the edges in u. A ***solution*** to G is an infinite path s through G with the following properties:

- s visits every vertex in V infinitely often;
- Any finite subpath of s that starts and ends at consecutive occurrences of a given vertex v must have duration at most $RD(v)$.

Definition 6.1.3 (The CR-UAV Problem with a single UAV)**.** *Given G as described above, does G have a solution?*

As pointed out in [LFHKG13], if a solution exists at all then a *periodic* solution can be found, i.e., an infinite path in which the targets are visited repeatedly in the same order.

6.1.4 The PERIODIC SAT Problem

PERIODIC SAT is one of the many PSPACE-complete problems introduced in [Orl81]. In the following definition (and in the rest of this paper), let $\overline{x}$ be a finite set of variables and let $\overline{x}^j$ be the set of variables obtained from $\overline{x}$ by adding a superscript j to each variable.

Definition 6.1.4 (The PERIODIC SAT Problem [Orl81])**.** *Consider a CNF formula $\varphi(0)$ over $\overline{x}^0 \cup \overline{x}^1$. Let $\varphi(j)$ be the formula obtained from $\varphi(0)$ by replacing all variables $x_i^0 \in \overline{x}^0$ by x_i^j and all variables $x_i^1 \in \overline{x}^1$ by x_i^{j+1}. Is there an assignment of $\bigcup_{j\geq 0} \overline{x}^j$ such that $\bigwedge_{j\geq 0} \varphi(j)$ is satisfied?*

Since $|\overline{x}|$ is finite, if such an assignment exists, there must be a *periodic* satisfying assignment to $\bigwedge_{j\geq 0} \varphi(j)$ whose period N is at most exponential in $|\overline{x}|$ (i.e., there is a positive integer $N \leq 2^{|\overline{x}|}$ such that for all non-negative integers j_1 and j_2, the truth values of $\overline{x}^{j_1}$ and $\overline{x}^{j_2}$ are identical if $j_1 \equiv j_2 \pmod{N}$).

Example 6.1.5. Let $\overline{x} = \{x_1, x_2\}$ and

$$\varphi(0) = (x_1^0 \vee x_1^1) \wedge (x_2^0 \vee x_2^1) \wedge (\neg x_1^0 \vee \neg x_1^1) \wedge (\neg x_2^0 \vee \neg x_2^1)\,.$$

Then

$$\varphi(1) = (x_1^1 \vee x_1^2) \wedge (x_2^1 \vee x_2^2) \wedge (\neg x_1^1 \vee \neg x_1^2) \wedge (\neg x_2^1 \vee \neg x_2^2)\,,$$
$$\varphi(2) = (x_1^2 \vee x_1^3) \wedge (x_2^2 \vee x_2^3) \wedge (\neg x_1^2 \vee \neg x_1^3) \wedge (\neg x_2^2 \vee \neg x_2^3)\,,$$

and so on. The infinite conjunction $\bigwedge_{j\geq 0} \varphi(j)$ is satisfied by the following periodic assignment of variables $\bigcup_{j\geq 0} \overline{x}^j$ (with period $N = 2$):

$$(x_1^i, x_2^i) \mapsto \begin{cases} (\mathbf{true}, \mathbf{true}) & \text{if } i \text{ is even} \\ (\mathbf{false}, \mathbf{false}) & \text{otherwise.} \end{cases}$$

6.2 Periods of Solutions

In [BGA12] it is claimed that the CR-UAV Problem with a single UAV is in NP. The claim is based on the following bound on the periods of solutions:

Claim 6.2.1 ([BGA12, Theorem 4.5]). Consider an instance G of the CR-UAV Problem with a single UAV. If G has a solution, then G has a solution of the form u^ω where u is a finite path through G with $|u| \leq \dfrac{\max_{v\in V} RD(v)}{\min_{\substack{v,v'\in V \\ v\neq v'}} FT(v,v')}$.

If constants are encoded in unary, the claim above would immediately imply NP-membership of the problem (with a single UAV). However, the claim turned out to be incorrect, as we now give a counterexample below. Consider the problem instance G in Figure 6.2 (we number the vertices in clockwise order, starting from bottom left). The shortest possible period of a solution is 11 (this can be verified with a model checker, e.g., NuSMV [CCG+02]) whereas the claim above gives a bound of 10.

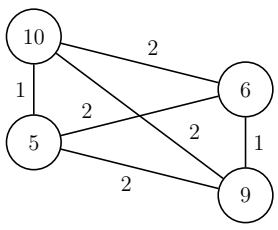


Figure 6.2: A periodic solution with the shortest period: $(32010230210)^\omega$.

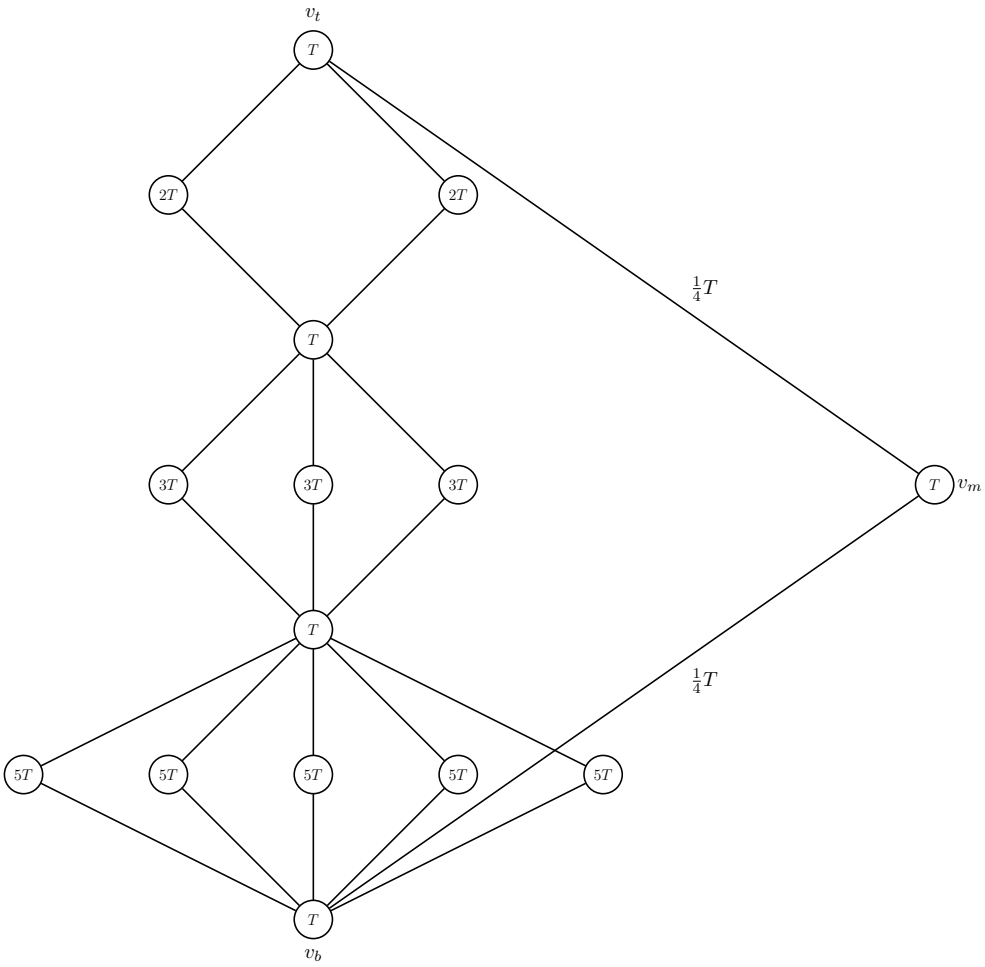


Figure 6.3: The clique G_3.

In fact, we can state a stronger result here. The following proposition says that, the shortest period of a solution can indeed be exponential (and not linear) in the magnitude of the largest relative deadline.

Proposition 6.2.2. *There is a family of instances $\{G_n\}_{n>0}$ (of the* CR-UAV *Problem with a single UAV) such that the shortest possible period of a solution to G_n is exponential in the magnitude of the largest constant in G_n.*[2]

Proof. (*Sketch.*) See Figure 6.3 for an illustrated example where $T = 4n$. The i-th 'diamond' (in top-down order) has p_n branches where p_n is the n-th prime. The relative deadlines are set as indicated, each unlabelled edge has FT set to 1, and each missing edge has FT set to the 'shortest distance' between the two relevant vertices. It can be shown that a solution must be an infinite repetition of either (i) from v_t through all

[2]The proof of this proposition is due to Daniel Bundala.

the diamonds to v_b, to v_m and to v_t again, or (ii) from v_b through all the diamonds to v_t, to v_m and to v_b again. Furthermore, in each diamond one must go straight down, and only the edges shown in the figure can be used. It can be shown that the shortest period of a solution to G_n is bounded below by $\prod_{i=1}^{n} p_i = \Omega(e^n)$. On the other hand, the number of vertices and the largest constant in G_n are both $O(n^2 \ln n)$. □

6.3 PSPACE-Hardness

In this section, we give a reduction from the PERIODIC SAT Problem to the CR-UAV Problem with a single UAV. Consider a CNF formula $\varphi(0) = c_1 \wedge \cdots \wedge c_h$ over $\overline{x}^0 = \{x_1^0, \ldots, x_m^0\}$ and $\overline{x}^1 = \{x_1^1, \ldots, x_m^1\}$. Without loss of generality, we assume that each clause c_j of $\varphi(0)$ is non-trivial (i.e., c_j does not contain both positive and negative occurrences of a variable) and $m > 2$, $h > 0$. We can construct an instance G of the CR-UAV Problem with a single UAV (with the largest constant having magnitude $O(m^2 h)$ and $|V| = O(mh)$) such that $\bigwedge_{j \geq 0} \varphi(j)$ is satisfiable if and only if G has a solution.

The general idea of the reduction is inspired by the textbook reduction from 3SAT to HAMILTONIAN PATH [Sip12] and can be described as follows. We construct *variable gadgets* that can be traversed in two 'directions' (corresponding to assignments **true** and **false** to variables). A *clause vertex* is visited if the corresponding clause is satisfied by the assignment. Crucially, we use *consistency gadgets*, in which we set the relative deadlines of the vertices carefully to ensure that the directions of traversals of the variable gadgets for $\overline{x}^1$ (corresponding to a particular assignment of variables) in a given iteration is consistent with the directions of traversals of the variable gadgets for $\overline{x}^0$ in the next iteration.

6.3.1 The Construction

We describe and explain each part of G in detail. The reader is advised to glance ahead to Figure 6.8 (on page 102) to form an impression of G. Note that for ease of presentation, we temporarily relax the requirement that FT be a metric and describe G as an incomplete graph.[3] In what follows, let $l = 24h + 34$ and

$$T = 2\Big(m\big(2(3m+1)l + l\big) + m\big(2(3m+2)l + l\big) + l + 2h\Big).$$

[3]In the single-UAV case, if the FT of some edge is greater than any value in RD, that edge can simply be seen as non-existent.

Variable gadgets. For each variable x_i^0, we construct (as a subgraph of G) a *variable gadget*. It consists of the following vertices (see Figure 6.4):

- Three vertices on the left side ($LS_i = \{v_i^{t,L}, v_i^{m,L}, v_i^{b,L}\}$)
- Three vertices on the right side ($RS_i = \{v_i^{t,R}, v_i^{m,R}, v_i^{b,R}\}$)
- A '*clause box*' ($CB_i^j = \{v_i^{a,j}, v_i^{b,j}, v_i^{c,j}, v_i^{d,j}, v_i^{e,j}, v_i^{f,j}\}$) for each $j \in \{1, \ldots, h\}$
- A '*separator box*' ($SB_i^j = \{v_i^{\bar{a},j}, v_i^{\bar{b},j}, v_i^{\bar{c},j}, v_i^{\bar{d},j}, v_i^{\bar{e},j}, v_i^{\bar{f},j}\}$) for each $j \in \{0, \ldots, h\}$
- A vertex at the top (v_{top} if $i = 0$, v_{i-1} otherwise)
- A vertex at the bottom (v_i).

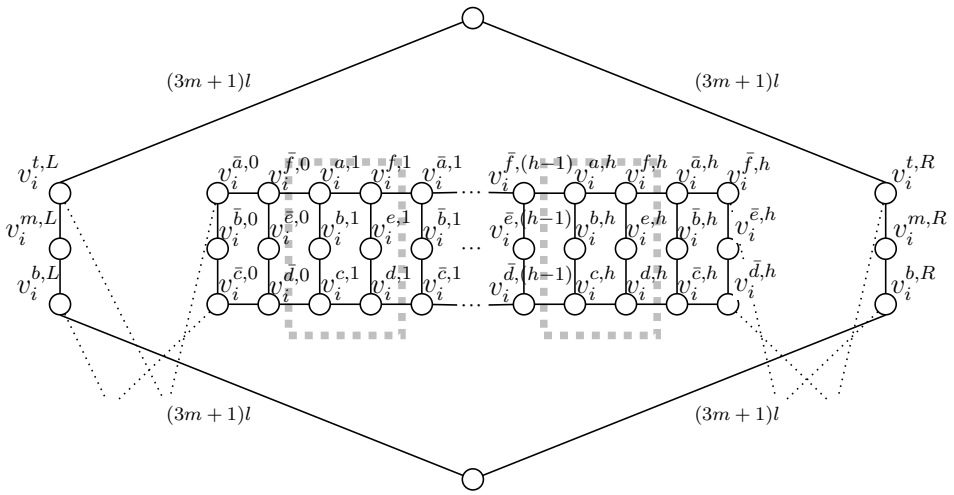


Figure 6.4: The variable gadget for x_i^0.

The clause boxes for $j \in \{1, \ldots, h\}$ are aligned horizontally in the figure. A separator box is laid between each adjacent pair of clause boxes and at both ends. This row of boxes ($Row_i = \bigcup_{j \in \{1,\ldots,h\}} CB_i^j \cup \bigcup_{j \in \{0,\ldots,h\}} SB_i^j$) is then put between LS_i and RS_i. The RD of all vertices $v \in LS_i \cup RS_i \cup Row_i$ are set to $T + l + 2h$.

The vertices are connected as indicated by solid lines in the figure. The four 'long' edges in the figure have their FT set to $(3m+1)l$ while all other edges have FT equal to 2, e.g., $FT(v_{top}, v_1^{t,L}) = (3m+1)l$ and $FT(v_1^{b,1}, v_1^{c,1}) = 2$. There is an exception though: $FT(v_m^{b,L}, v_m)$ and $FT(v_m^{b,R}, v_m)$ (in the variable gadget for x_m^0) are equal to $(3m+2)l$.

The variable gadgets for variables x_i^1 are constructed almost identically. The three vertices on the left and right side are now LS_{i+m} and RS_{i+m}. The set of vertices in

the row is now $Row_{i+m} = \bigcup_{j\in\{1,\ldots,h\}} CB^j_{i+m} \cup \bigcup_{j\in\{0,\ldots,h\}} SB^j_{i+m}$. The vertex at the top is v_{i+m-1} and the vertex at the bottom is v_{i+m} ($i \neq m$) or v_{bot} ($i = m$). The RD of vertices in $LS_{i+m} \cup RS_{i+m} \cup Row_{i+m}$ are set to $T + l + 2h$, and the FT of the edges are set as before, except that all the 'long' edges now have FT equal to $(3m + 2)l$.

Now consider the following ordering of variables:

$$x^0_1, x^0_2, \ldots, x^0_m, x^1_1, x^1_2, \ldots, x^1_m\,.$$

Observe that the variable gadgets for two 'neighbouring' variables (with respect to this ordering) have a vertex in common. To be precise, the set of shared vertices is $S = \{v_1, \ldots, v_{2m-1}\}$. We set the RD of all vertices in S to $T + 2h$ and the RD of v_{top} and v_{bot} to T.

Clause vertices. For each clause c_j in $\varphi(0)$, there is a *clause vertex* v^{c_j} with RD set to $\frac{3}{2}T$. If x^0_i occurs in c_j as a literal, we connect the j-th clause box in the variable gadget for x^0_i to v^{c_j} as shown in Figure 6.5 and set the FT of these new edges to 2 (e.g., $FT(v^{c_j}, v^{c,j}_i) = FT(v^{c_j}, v^{d,j}_i) = 2$). If instead $\neg x^0_i$ occurs in c_j, then v^{c_j} is connected to $v^{a,j}_i$ and $v^{f,j}_i$ (with FT equal to 2). Likewise, the variable gadget for x^1_i may be connected to v^{c_j} via $\{v^{c,j}_{i+m}, v^{d,j}_{i+m}\}$ (if x^1_i occurs in c_j) or $\{v^{a,j}_{i+m}, v^{f,j}_{i+m}\}$ (if $\neg x^1_i$ occurs in c_j).

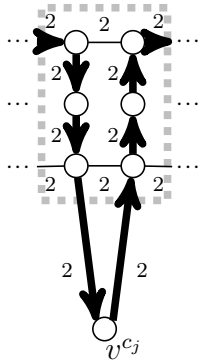


Figure 6.5: The variable occurs positively in c_j.

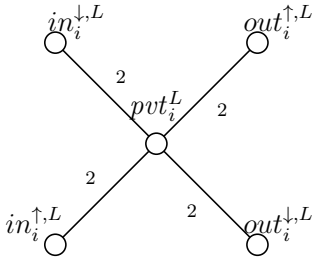


Figure 6.6: A consistency gadget LCG_i.

Consistency gadgets. For each $i \in \{1, \ldots, m\}$, we construct two *consistency gadgets* LCG_i (see Figure 6.6) and RCG_i. In LCG_i, the vertex at the centre ($pvt^{t,L}_i$) has RD equal to $\frac{1}{2}T + m\big(2(3m + 2)l + l\big) - (2i - 1)l + 4h$. The other four vertices ($in^{\downarrow,L}_i$, $out^{\uparrow,L}_i$, $in^{\uparrow,L}_i$ and $out^{\downarrow,L}_i$) have RD equal to $\frac{3}{2}T$. The FT from $pvt^{t,L}_i$ to any

of the other four vertices is 2. RCG_i is identical except that the subscripts on the vertices change from L to R.

LCG_i and RCG_i are connected to the variable gadgets for x_i^0 and x_i^1 as in Figure 6.7. The vertices $in_i^{\downarrow,L}, out_i^{\uparrow,L}, in_i^{\downarrow,R}, out_i^{\uparrow,R}$ are connected to certain vertices in the variable

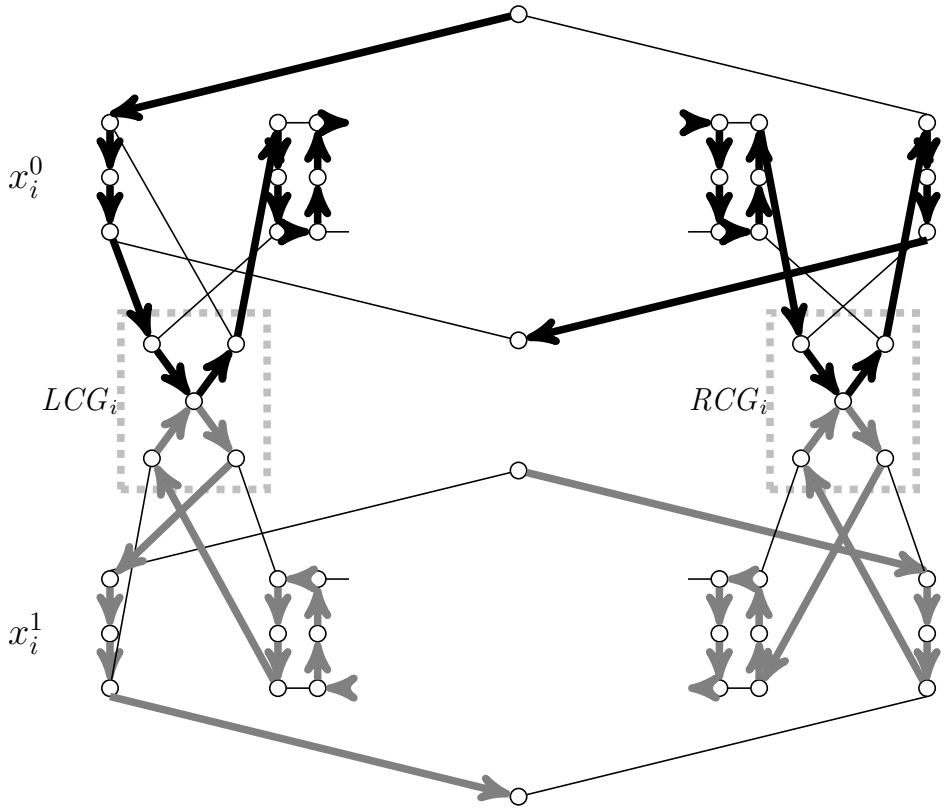


Figure 6.7: Connecting the variable gadgets for x_i^0 and x_i^1 to LCG_i and RCG_i.

gadget for x_i^0—this allows pvt_i^L and pvt_i^R to be traversed 'from above'. Similarly, the edges connected to $in_i^{\uparrow,L}, out_i^{\downarrow,L}, in_i^{\uparrow,L}, out_i^{\downarrow,L}$ allow pvt_i^L and pvt_i^R to be traversed 'from below'. Formally, $FT(v, v') = 2$ if

- $v = in_i^{\downarrow,L}, v' \in \{v_i^{b,L}, v_i^{\bar{c},0}\}$ or $v = in_i^{\downarrow,R}, v' \in \{v_i^{\bar{f},h}, v_i^{b,R}\}$
- $v = out_i^{\uparrow,L}, v' \in \{v_i^{t,L}, v_i^{\bar{a},0}\}$ or $v = out_i^{\uparrow,R}, v' \in \{v_i^{\bar{d},h}, v_i^{t,R}\}$
- $v = in_i^{\uparrow,L}, v' \in \{v_{(i+m)}^{b,L}, v_{(i+m)}^{\bar{c},0}\}$ or $v = in_i^{\uparrow,R}, v' \in \{v_{(i+m)}^{\bar{f},h}, v_{(i+m)}^{b,R}\}$
- $v = out_i^{\downarrow,L}, v' \in \{v_{(i+m)}^{t,L}, v_{(i+m)}^{\bar{a},0}\}$ or $v = out_i^{\downarrow,R}, v' \in \{v_{(i+m)}^{\bar{d},h}, v_{(i+m)}^{t,R}\}$.

Two parts of an intended path, which we will explain in more detail later, is also illustrated in Figure 6.7.

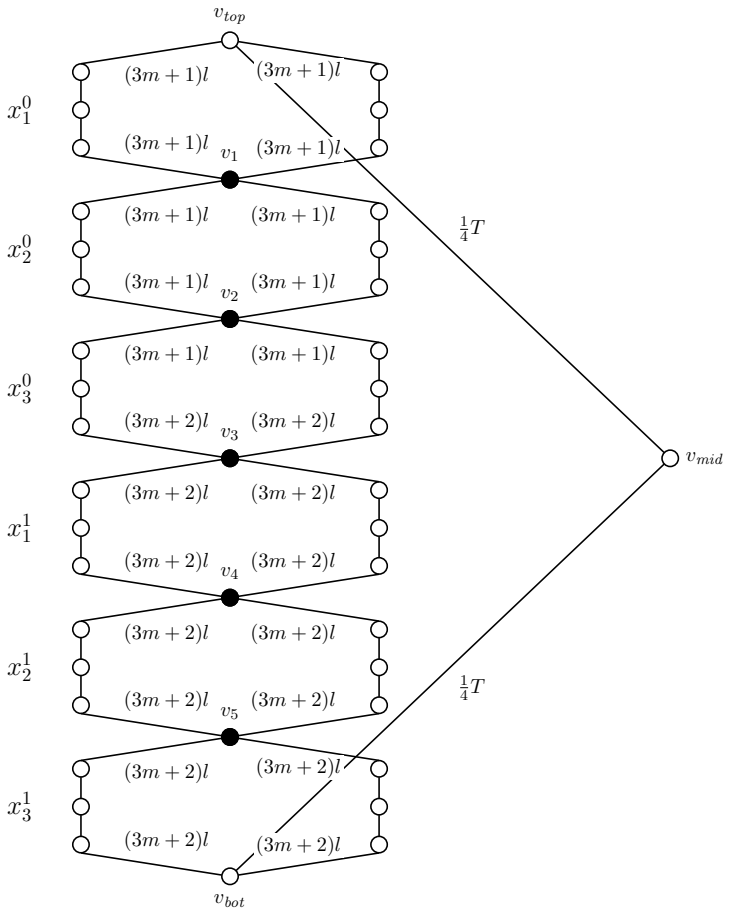


Figure 6.8: An example with $m = 3$. Solid circles denote shared vertices $S = \{v_1, \ldots, v_5\}$.

Finally, there is a vertex v_{mid} with $RD(v_{mid}) = T$ connected to v_{bot} and v_{top} with two edges, both with FT equal to $\frac{1}{4}T$. The FT of all the missing edges are $2T$ (note that the largest value in RD is less than $2T$, so these edges can never be taken). This completes the construction of G. An example with $m = 3$ is given in Figure 6.8, where vertices in S (shared by two variable gadgets) are depicted as solid circles.

Proposition 6.3.1. *$\bigwedge_{j \geq 0} \varphi(j)$ is satisfiable iff G has a solution.*

6.3.2 The Main Proof

The rest of this chapter is devoted to the proof of Proposition 6.3.1. We first prove the forward direction. Given a satisfying assignment to $\bigwedge_{j \geq 0} \varphi(j)$, we construct a solution s as follows: s starts from v_{top} and goes through the variable gadgets for $x_1^0, x_2^0, \ldots, x_m^0, x_1^1, x_2^1, \ldots, x_m^1$ in order, eventually reaching v_{bot}. Each variable gadget is traversed according to the truth value assigned to its corresponding variable. In such

a traversal, both pvt_i^L and pvt_i^R are visited once (see the thick arrows in Figure 6.7 for the situation when x_i^0 is assigned **true** and x_i^1 is assigned **false**). Along the way from v_{top} to v_{bot}, s detours at certain times and 'hits' each clause vertex exactly once as illustrated by the thick arrows in Figure 6.5 (this can be done as $\varphi(0)$ is satisfied by the assignment). Then s goes back to v_{top} through v_{mid} and starts over again, this time following the truth values assigned to variables in $\overline{x}^1 \cup \overline{x}^2$, and so on. One can verify that this describes a solution to G.

Now consider the other direction. Let

$$s = (v_{mid} s_1 v_{mid} \dots v_{mid} s_p)^\omega$$

be a periodic solution to G where each *segment* s_j, $j \in \{1, \dots, p\}$ is a finite subpath visiting only vertices in $V \setminus \{v_{mid}\}$. We further assume that s satisfies the first case of the following proposition (this is sound as a periodic solution can be 'reversed' while remaining a valid solution). Let $s_{j-1} = s_p$ if $j = 1$ and $s_{j+1} = s_1$ if $j = p$.

Proposition 6.3.2. *In $s = (v_{mid} s_1 v_{mid} \dots v_{mid} s_p)^\omega$, either of the following holds:*

- *All s_j, $j \in \{1, \dots, p\}$ starts with v_{top} and ends with v_{bot}*
- *All s_j, $j \in \{1, \dots, p\}$ starts with v_{bot} and ends with v_{top}.*

We prove this proposition below.

Lemma 6.3.3. *Each segment s_j must start with and end with v_{top} or v_{bot}.*

Lemma 6.3.4. *The time needed from v_{top} or v_{bot} to any other vertex is at least $(3m + 1)l$.*

Lemma 6.3.5. *The time needed from v_{mid} to any other vertex is at least $\frac{1}{4}T$.*

Lemma 6.3.6. *Each segment s_j must contain more than one vertex.*

Proof. By Lemma 6.3.3, without loss of generality let $s_j = v_{bot}$, a single vertex. It is easy to see that s_{j-1} must end with v_{top} and s_{j+1} must start with v_{top}, otherwise the relative deadline of v_{top} will be violated. Now consider v_1 (with $RD(v_1) = T + 2h$). By Lemma 6.3.4 and the fact that $dur(v_{top} v_{mid} v_{bot} v_{mid} v_{top}) = T$, the relative deadline of v_1 is violated for sure even if s visits v_1 immediately after v_{top}. This is a contradiction. □

Proposition 6.3.7. *For each segment s_j, $0 < dur(s_j) \leq \frac{1}{2}T$.*

Proof. By Lemma 6.3.6 we have $dur(s_j) > 0$. For the upper bound, note that $dur(v_{mid}s_jv_{mid}) = \frac{1}{2}T + dur(s_j)$ and $RD(v_{mid}) = T$. □

Proposition 6.3.8. *Each segment s_j contains all vertices in $V \setminus \{v_{mid}\}$ with relative deadlines less or equal than $T + l + 2h$.*

Proof. Let $v \in V \setminus \{v_{mid}\}$ be a vertex missing in s_j with $RD(v) \leq T + l + 2h$. By Lemmas 6.3.3, 6.3.4 and 6.3.6, $dur(s_j) \geq 2(3m + 1)l > l + l > l + 2h$. We have $dur(v_{mid}s_jv_{mid}) = \frac{1}{2}T + dur(s_j) > \frac{1}{2}T + l + 2h$. By Lemma 6.3.5, $dur(vv_{mid}s_jv_{mid}v)$ must be greater than $T + l + 2h$ for any $v \in V \setminus \{v_{mid}\}$, which is a contradiction. □

By Proposition 6.3.8, we first derive a (crude) lower bound on $dur(s_j)$. The sum of the minimum times needed to enter and leave every $v \in S$ and the minimum times needed to enter and leave both ends of s_j gives

$$dur(s_j) \geq (m-1)\big(2(3m+1)l\big) + m\big(2(3m+2)l\big) + 2(3m+1)l\,. \tag{6.1}$$

Proposition 6.3.9. *v_{top}, v_{bot} and each $v \in S$ appears once in each segment s_j.*

Proof. Without loss of generality, assume one of these vertices appears more than once in s_j. By a similar argument as above, we derive that $dur(s_j)$ is at least $(m-1)\big(2(3m+1)l\big) + m\big(2(3m+2)l\big) + 2(3m+1)l + 2(3m+1)l > \frac{1}{2}T$. This contradicts Proposition 6.3.7. □

By the proposition above, we can revise our lower bound in Eq.(6.1) by noting that s_j must start and end with different vertices. This gives

$$dur(s_j) \geq (m-1)\big(2(3m+1)l\big) + m\big(2(3m+2)l\big) + (3m+1)l + (3m+2)l\,. \tag{6.2}$$

Now without loss of generality let s_j ends with v_{top} and s_{j+1} starts with v_{top}. By Eq.(6.2), $dur(s_j) + dur(s_{j+1}) \geq 2\Big((m-1)\big(2(3m+1)l\big) + m\big(2(3m+2)l\big) + (3m+1)l + (3m+2)l\Big) > \frac{1}{2}T$, and hence $dur(s_jv_{mid}s_{j+1}) > T$. By Proposition 6.3.9, v_{bot} can only appear at both ends of $s_jv_{mid}s_{j+1}$, hence its relative deadline must be violated. This is a contradiction. Proposition 6.3.2 is hence proved.

We now argue that s 'witnesses' a satisfying assignment to $\bigwedge_{j\geq 0}\varphi(j)$.

Lemma 6.3.10. *In each segment s_j, each vertex in $\bigcup_{i\in\{1,\ldots,m\}}\{pvt_i^L, pvt_i^R\}$ appears twice whereas other vertices in $V \setminus \{v_{mid}\}$ appear once.*

To prove this lemma, we first refine our lower bound in Eq.(6.2) by taking into account other vertices in variable gadgets and consistency gadgets with RD less or equal to $T + l + 2h$ (by Proposition 6.3.8). As many of these vertices are adjacent, we only accumulate the minimum times needed to enter them. This gives an extra time of $m(24h + 22) + 4m + m(24h + 22)$ (note that by Proposition 6.3.9, only one of the four vertices connected to a shared vertex has been entered and cannot be included in the calculation). In total, we have

$$dur(s_j) \geq \frac{1}{2}T - 20m - 2h\,. \tag{6.3}$$

Proposition 6.3.11. *Each segment s_j contains all vertices with relative deadlines equal to $\frac{3}{2}T$, i.e., clause vertices and vertices in $\bigcup_{i\in\{1,\ldots,m\}} \big((LCG_i \setminus \{pvt_i^L\}) \cup (RCG_i \setminus \{pvt_i^R\})\big)$.*

Proof. Assume that there is such a vertex v not appearing in s_j. By Eq.(6.3), we have $dur(v_{bot}v_{mid}s_jv_{mid}v_{top}) \geq \frac{3}{2}T - 20m - 2h$. By Lemma 6.3.4, the relative deadline of v must be violated as $dur(vv_{bot}v_{mid}s_jv_{mid}v_{top}v) \geq \frac{3}{2}T - 20m - 2h + 2(3m + 1)l > \frac{3}{2}T$. This is a contradiction. □

Based on the previous proposition, we can further refine our lower bound on the duration of a segment. The minimum times needed to enter

- clause vertices v^{c_j}, $j \in \{1, \ldots, h\}$
- vertices in $\bigcup_{i\in\{1,\ldots,m\}} \big((LCG_i \setminus \{pvt_i^L\}) \cup (RCG_i \setminus \{pvt_i^R\})\big)$

can now be included in the calculation. We have

$$dur(s_j) \geq \frac{1}{2}T - 4h\,. \tag{6.4}$$

Proposition 6.3.12. *In each segment s_j, each vertex in $\bigcup_{i\in\{1,\ldots,m\}}\{pvt_i^L, pvt_i^R\}$ appears more than once.*

Proof. Let there be such a vertex v appearing only once in a segment. By Lemma 6.3.4, there are two occurrences of v in s separated by at least $\frac{1}{2} \cdot \big(\frac{1}{2}T + (\frac{1}{2}T - 4h) + \frac{1}{2}T\big) + (3m + 1)l$. This exceeds all possible values of $RD(v)$. □

By the proposition above, we assume that each vertex in $\bigcup_{i\in\{1,\ldots,m\}}\{pvt_i^L, pvt_i^R\}$ appears twice in a segment. Counting each such vertex once again gives an extra time

of $4h$. The sum of this with Eq.(6.4) matches the upper bound in Proposition 6.3.7. Any more visit to a vertex in $V \setminus \{v_{mid}, v_{top}, v_{bot}, v_1, \ldots, v_{2m-1}\}$ will immediately contradict Proposition 6.3.7. Lemma 6.3.10 is hence proved.

Based on Lemma 6.3.10, we show that s cannot 'jump' between variable gadgets via clause vertices. It follows that the traversal of each Row_i must be done in a single pass.

Proposition 6.3.13. *In each segment s_j, if v^{c_k} is entered from a clause box (in some variable gadget), the edge that immediately follows must go back to the same clause box.*

Proof. Consider a 3×3 'box' formed by a separator box and (the left- or right-) half of a clause box. Note that except for the four vertices at the corners, no vertex in this 3×3 box is connected to the rest of the graph. Recall that if each vertex in this 3×3 box is to be visited only once (as enforced by Lemma 6.3.10), it must be traversed in the patterns illustrated in Figures 6.9 and 6.10.

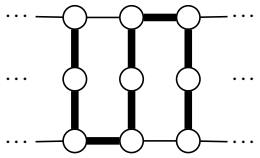


Figure 6.9: Pattern '⊔⊓'.

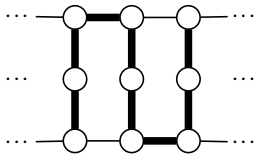


Figure 6.10: Pattern '⊓⊔'.

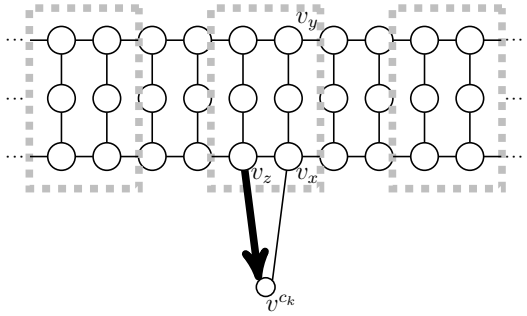


Figure 6.11: x_i^0 occurs positively in c_k.

Now consider the situation in Figure 6.11 where s_j goes from v_z to v^{c_k}. The 3×3 box with v_z at its lower-right must be traversed in Pattern '⊔⊓' (as otherwise v_z will be visited twice). Assume that s_j does not visit v_x immediately after v^{c_k}. As v_x cannot be entered or left via v_z and v^{c_k}, the 3×3 box with v_x at its lower-left must also be traversed in Pattern '⊔⊓'. However, there is then no way to enter or leave v_y. This is a contradiction. □

Note that in Figure 6.11, the three clause boxes (framed by dotted lines) are all traversed in Pattern '⊓' or they are all traversed in Pattern '⊔'. More generally, we have the following proposition.

Proposition 6.3.14. *In each segment s_j, clause boxes in a given variable gadget are all traversed in Pattern '⊓' or they are all traversed in Pattern '⊔' (with possible detours via clause vertices).*

Write $v \to v'$ for the edge from v to v' and $v \rightsquigarrow v'$ for a finite path that starts with v and ends with v'. By Lemma 6.3.10, each segment s_j can be written as $v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots \rightsquigarrow v_{b_{2m-1}} \rightsquigarrow v_{bot}$ where $b_1, \ldots, b_{2m-1}$ is a permutation of $1, \ldots, 2m-1$. We show that each subpath $v \rightsquigarrow v'$ of s_j with distinct $v, v' \in S \cup \{v_{top}, v_{bot}\}$ and no $v'' \in S \cup \{v_{top}, v_{bot}\}$ in between must be of a very restricted form. For convenience, we call such a subpath $v \rightsquigarrow v'$ a *fragment*.

Proposition 6.3.15. *In each segment $s_j = v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots \rightsquigarrow v_{b_{2m-1}} \rightsquigarrow v_{bot}$, a fragment $v \rightsquigarrow v'$ visits pvt_i^L and pvt_i^R (once for each) for some $i \in \{1, \ldots, m\}$. Moreover, each fragment $v \rightsquigarrow v'$ in $v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots \rightsquigarrow v_{b_m}$ visits a different set $\{pvt_i^L, pvt_i^R\}$. The same holds for $v_{b_m} \rightsquigarrow v_{b_{m+1}} \rightsquigarrow \cdots \rightsquigarrow v_{bot}$.*

Proof. It is clear that $dur(v \rightsquigarrow v') \geq 2(3m+1)l$, and hence $dur(v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots v_{b_m}) \geq m\big(2(3m+1)l\big)$. Let there be a vertex $v \in \bigcup_{i \in \{1,\ldots,m\}} \{pvt_i^L, pvt_i^R\}$ missing in $v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots v_{b_m}$. Since the time needed from v_{b_m} to v is greater than $(3m+1)l$, even if s_j visits v as soon as possible after v_{b_m}, the duration from v_{bot} in s_{j-1} to v in s_j will still be greater than $\frac{1}{2}T + m\big(2(3m+1)l\big) + (3m+1)l > RD(v)$, which is a contradiction. Therefore, all vertices in $\bigcup_{i \in \{1,\ldots,m\}} \{pvt_i^L, pvt_i^R\}$ must appear in the subpath from v_{top} to v_{b_m}. The same holds for the subpath from v_{b_m} to v_{bot} by similar arguments. Now note that by Proposition 6.3.13, a fragment $v \rightsquigarrow v'$ may visit at most two vertices—$\{pvt_i^L, pvt_i^R\}$ for some $i \in \{1, \ldots, m\}$. The proposition then follows from Lemma 6.3.10. □

Proposition 6.3.16. *In each segment s_j, a fragment $v \rightsquigarrow v'$ visits all vertices in either Row_i or Row_{i+m} for some $i \in \{1, \ldots, m\}$ but not a single vertex in $\bigcup_{j \in \{1,\ldots,m\}, j \neq i} (Row_j \cup Row_{j+m})$.*

Now consider a fragment $v \rightsquigarrow v'$ that visits pvt_i^L and pvt_i^R (by Proposition 6.3.15). By Lemma 6.3.10, $v \rightsquigarrow v'$ must also visit exactly two vertices other than pvt_i^L in LCG_i and exactly two vertices other than pvt_i^R in RCG_i (once for each). It is not hard to see that $v \rightsquigarrow v'$ must contain, in order, the following subpaths (together with some obvious choices of edges connecting these subpaths):

(i). A long edge, e.g., $v_i \to v_i^{b,R}$.

(ii). A 'side', e.g., $v_i^{b,R} \to v_i^{m,R} \to v_i^{t,R}$.

(iii). A subpath consisting of a *pvt* vertex and two other vertices in the relevant consistency gadget, e.g., $out_i^{\uparrow,R} \to pvt_i^R \to in_i^{\downarrow,R}$.

(iv). A traversal of a row with detours.

(v). A subpath consisting of a *pvt* vertex and two other vertices in the relevant consistency gadget.

(vi). A side.

(vii). A long edge.

The following proposition is then immediate. In particular, the exact value of $dur(v \rightsquigarrow v')$ is decided by:

- FT of the long edges taken in (i) and (vii)
- detours to clause vertices in (iv).

Proposition 6.3.17. *In each segment s_j, the following holds for all fragments $v \rightsquigarrow v'$:*

$$2(3m+1)l + l \leq dur(v \rightsquigarrow v') \leq 2(3m+2)l + l + 2h.$$

Proposition 6.3.18. *The order the sets $\{pvt_i^L, pvt_i^R\}$ are visited (regardless of which vertex in the set is first visited) in the first m fragments of each segment s_j is identical to the order they are visited in the last m fragments of s_{j-1}.*

Proof. By Proposition 6.3.17, if this does not hold then there must be a *pvt* vertex having two occurrences in s separated by more than $\frac{1}{2}T + m\big(2(3m+1)l + l\big) + 2(3m+1)l$. This is a contradiction. □

For each segment s_j, we denote by $first(s_j)$ the 'first half' of s_j, i.e., the subpath of s_j that consists of the first m fragments of s_j and by $second(s_j)$ the 'second half' of s_j. Write $\exists(v \rightsquigarrow v') \subseteq u$ if u has a subpath of the form $v \rightsquigarrow v'$.

Proposition 6.3.19. *In each segment $s_j = v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots \rightsquigarrow v_{b_{2m-1}} \rightsquigarrow v_{bot}$, we have $b_i = i$ for all $i \in \{1, \ldots, 2m-1\}$.*

Proof. First note that by construction and Proposition 6.3.15, $\{pvt_m^L, pvt_m^R\}$ must be the last set of *pvt* vertices visited in $second(s_{j-1})$. By Proposition 6.3.18, it must also be the last set of *pvt* vertices visited in $first(s_j)$. Now assume that a long edge of flight time $(3m+2)l$ is taken before pvt_m^L and pvt_m^R are visited in $first(s_j)$. Consider the following cases:

- $\exists(pvt_m^L \rightsquigarrow pvt_m^R) \subseteq second(s_{j-1})$ and $\exists(pvt_m^R \rightsquigarrow pvt_m^L) \subseteq first(s_j)$: Note that the last edge taken in s_{j-1} is a long edge of flight time $(3m+2)l$, and hence there are two occurrences of pvt_m^L in s separated by at least $\frac{1}{2}T + m\big(2(3m+1)l + l\big) + 2l > \frac{1}{2}T + m\big(2(3m+1)l + l\big) + l + 4h = RD(pvt_m^L)$.
- $\exists(pvt_m^R \rightsquigarrow pvt_m^L) \subseteq second(s_{j-1})$ and $\exists(pvt_m^L \rightsquigarrow pvt_m^R) \subseteq first(s_j)$: The same argument shows that pvt_m^R must miss its relative deadline.
- $\exists(pvt_m^L \rightsquigarrow pvt_m^R) \subseteq second(s_{j-1})$ and $\exists(pvt_m^L \rightsquigarrow pvt_m^R) \subseteq first(s_j)$: The same argument shows that both pvt_m^L and pvt_m^R must miss their relative deadlines.
- $\exists(pvt_m^R \rightsquigarrow pvt_m^L) \subseteq second(s_{j-1})$ and $\exists(pvt_m^R \rightsquigarrow pvt_m^L) \subseteq first(s_j)$: The same argument shows that both pvt_m^L and pvt_m^R must miss their relative deadlines.

We therefore conclude that in $first(s_j)$, all long edges taken before pvt_m^L and pvt_m^R are visited must have FT equal to $(3m+1)l$. Furthermore, all such long edges must be traversed 'downwards' (by Lemma 6.3.10). It follows that $b_i = i$ for $i \in \{1, \ldots, m-1\}$. By Proposition 6.3.18, Lemma 6.3.10 and $m > 2$, we easily derive that $b_m = m$ and then $b_i = i$ for $i \in \{m+1, \ldots, 2m-1\}$. □

By Proposition 6.3.19, the long edges in each variable gadget must be traversed in the ways shown in Figures 6.12 and 6.13.

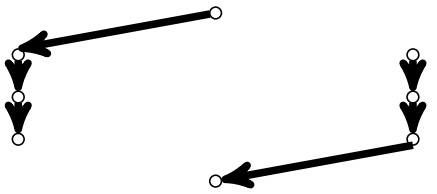


Figure 6.12: The variable is assigned to **true**.

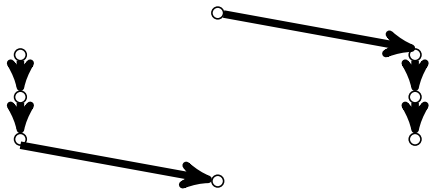


Figure 6.13: The variable is assigned to **false**.

Proposition 6.3.20. *For each segment s_j, the ways in which the long edges are traversed in the last m fragments of s_j are consistent with the ways in which the long edges are traversed in the first m fragments of s_{j+1}.*

Proof. Without loss of generality, consider the case that $\exists(pvt_i^L \rightsquigarrow pvt_i^R) \subseteq second(s_j)$ and $\exists(pvt_i^R \rightsquigarrow pvt_i^L) \subseteq first(s_{j+1})$. By Proposition 6.3.19, these two occurrences of pvt_i^L in s are separated by, at least, the sum of $\frac{1}{2}T + m\big(2(3m+2)l + l\big) - (2i-1)l$ and the duration of the actual subpath $pvt_i^R \rightsquigarrow pvt_i^L$ in $first(s_{j+1})$. It is clear that pvt_i^L must miss its relative deadline. □

Proposition 6.3.21. *In each segment s_j, if a variable gadget is traversed as in Figure 6.12 (Figure 6.13), then all of its clause boxes are traversed in Pattern '⊔' (Pattern '⊓').*

Consider a segment s_j. As each clause vertex is visited once in s_j (by Lemma 6.3.10), the ways in which the long edges are traversed in all fragments $v \rightsquigarrow v'$ of s_j (i.e., as in Figure 6.12 or Figure 6.13) can be seen as a satisfying assignment to $\varphi(0)$ (by construction and Proposition 6.3.21). By the same argument, the ways in which the long edges are traversed in all fragments of s_{j+1} can be seen as a satisfying assignment to $\varphi(1)$. Now by Proposition 6.3.20, the assignment of variables $\overline{x}^1$ is consistent in both segments. By the induction hypothesis, s witnesses a (periodic) satisfying assignment to $\bigwedge_{j\geq 0} \varphi(j)$. Proposition 6.3.1 is hence proved.

Finally, note that FT can easily be modified into a metric over V by replacing each entry of value $2T$ with the 'shortest distance' between the two relevant vertices. It is easy to see that Proposition 6.3.1 still holds. Our main result, which holds for the metric case, follows immediately from Section 6.1.2.

Theorem 6.3.22. *The* CR-UAV *Problem is* PSPACE*-complete.*[4]

6.4 Discussion

We have shown that the CR-UAV Problem is PSPACE-complete even in the single-UAV case. Our result corrects an erroneous claim in the literature.

Our PSPACE-hardness proof crucially depends on the freedom to set flight times and relative deadlines. In [LFHKG13], it is claimed that the *Euclidean* version of the problem, i.e., in which targets can be realised as points in a two-dimensional plane (with discretised distances between points), is NP-complete (with a single UAV). In the view of our result, we would like to investigate whether this claim is indeed true. It is conceivable that the techniques used in the well-known NP-hardness proof of

[4]Our result holds irrespective of whether the numbers are encoded in unary or binary.

EUCLIDEAN TSP [Pap77] might be useful in this regard, but we were unfortunately unable to leverage them in the case at hand.

As we mentioned earlier, introducing *periodicity* into many NP-complete problems renders them PSPACE-complete [Orl81]. The CR-UAV Problem, on the other hand, can be seen as a recurrent variant of the decision version of the *Travelling Salesman Problem with Time Windows* (TSPTW) with only upper bounds (or *TSP with Deadlines* [BHKK07]). Its PSPACE-hardness hence stems from *recurrence*: the decision version of the (non-recurrent) TSPTW Problem is NP-complete [Sav85]. The main difference between the two notions is that, in the former, a problem instance consists of a number of parts that correspond to neighbouring periods (e.g., in the case of PERIODIC SAT, the input is a formula that can be divided into two parts), whereas this is not the case in the latter. Our reduction reveals a connection between these two types of problem specifications. Moreover, since the latter type is much more common in practice [LKdPC10], we would like to know whether it is possible to obtain a similar 'rule of thumb' as in [Orl81].

Another possibility to extend the result here is to do a more refined complexity analysis of the CR-UAV Problem. For example, could it be fixed-parameter tractable in some parameters? Is there a PTAS under certain conditions (see [MHSR98] for some related results)? We leave these questions as future work.

Finally, we note that a number of crucial problems in other domains, e.g., the generalised pinwheel scheduling problem [FC05] and the message ferrying problem [ZAZ04], share similarities with the CR-UAV Problem—namely, they have relative deadlines and therefore 'contexts'. Most of these problems are only known to be NP-hard. It would be interesting to investigate whether our construction can be adapted to establish PSPACE-hardness of these problems.

Chapter 7

An Antichain-based Approach to the Cyclic-Routing UAV Problem

The main result of the previous chapter indicates that the CR-UAV Problem is computationally hard. However, in the realm of verification, solving a PSPACE-complete problem is rarely considered infeasible: there are harder problems shown to be efficiently solvable in practice (e.g., $\mathsf{LTL}_{\mathsf{fut}}$ realisability and synthesis [BBF+12]). In this chapter, we investigate empirically whether this is the case for the CR-UAV Problem.

We first describe how to model the (general) CR-UAV Problem as the emptiness problem for Büchi automata. Then we define a *direct* simulation relation on the state space of these automata, which allows us to exploit a standard antichain algorithm. As this turned out to be too slow in our case, we define a coarser *delayed* simulation relation and describe how to exploit it in a portfolio approach. Finally, we report some promising experimental results.

7.1 Non-Zeno Networks of Timed Automata

Recall that in Section 6.1.2 (on page 93) we defined the CR-UAV Problem in terms of the existence of infinite non-Zeno runs in a network of timed automata. However, there is an obstacle in using this formulation directly in practice: most current tools (such as UPPAAL [BDL+06]) do not automatically rule out Zeno runs. A standard solution to this issue is to add a new clock which ensures the progress of time [AM04, TYB05], but it is known that this may lead to large overhead in practice [HSW10]. Fortunately, we can turn a constructed network into a *strongly non-Zeno* one by the following modification on the component UAV-automata:

- Replace the guards '$UC \geq 0$' on all self-loops by '$UC \geq 1$'.

This modification does not affect the correctness of the construction since when we consider only integer time, self-loops can change neither locations nor clock values when $UC = 0$. In the rest of this chapter, we call this modified construction the ***standard*** timed automata modelling.

It is not hard to see that a further modification can be made on the standard timed automata modelling without affecting its correctness:

- Replace all guards of the form '$UC \geq c$' by '$UC = c$'.

We call this construction the ***exact*** timed automata modelling. While the number of admitted runs is clearly reduced with this modelling, we will see in Section 7.5 that the performance of tools do not necessarily improve. In fact, the simulation relations that we introduce later depend crucially on being able to take such edges when $UC > c$.

7.2 Büchi Automata

In this section, we explain how to construct Büchi automata that correspond to the networks of timed automata just described. Recall that we use $\top$ to represent all clock values greater than the largest constant in a timed automaton $\mathcal{A}$ as they are indistinguishable by $\mathcal{A}$. Together with the fact that it suffices to consider integer time in our case, it follows that the problem of the existence of infinite non-Zeno runs in a constructed network reduces to the emptiness problem for a Büchi automaton $\mathcal{B}$. In particular, all states are accepting in $\mathcal{B}$. This special case of Büchi automata is sometimes referred to as ***looping automata*** [WVS83].[1]

State space. Let the number of targets be $n \geq 2$ and the number of UAVs be $k < n$. Each state of $\mathcal{B}$ is a tuple

$$s = \langle TC_0, \ldots, TC_{n-1}, UC_0, \ldots, UC_{k-1}, latest_0, \ldots, latest_{k-1} \rangle .$$

The intended meanings of the components are described below.

- For each $i \in \{0, \ldots, n-1\}$, TC_i ('target clock' for the i-th target) is the time elapsed since the i-th target was last visited ($0 \leq TC_i \leq RD(i)$).

[1] The class of looping automata characterises exactly the class of safety properties that are expressible as Büchi automata [AS87].

- For each $j \in \{0, \ldots, k-1\}$, UC_j ('UAV clock' for the j-th UAV) is the time elapsed since the j-th UAV last visited a target ($0 \leq UC_j \leq FT_{max}$ where FT_{max} is the largest constant in FT).
- For each $j \in \{0, \ldots, k-1\}$, $latest_j$ is the last target visited by the j-th UAV ($0 \leq latest_j \leq n-1$).

We now describe two sets of transitions which, when combined with the set of states described above, give two different looping automata that correspond to the standard and exact timed automata modelling, respectively.

Standard modelling. Each transition of $\mathcal{B}$ is labelled by a triple

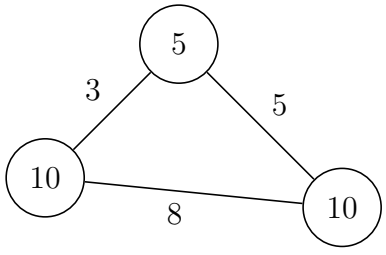
$$\langle turn, heading, step \rangle$$

with $turn \in \{0, \ldots, k-1\}$, $heading \in \{0, \ldots, n-1\}$ and $step \in \{0, \ldots, RD_{max}\}$ where RD_{max} is the largest constant in RD. Intuitively, we use a transition to describe an 'action' of a UAV. In a transition

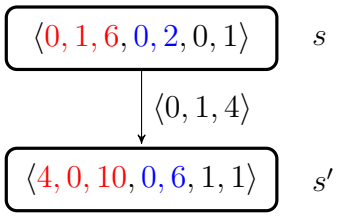
$$s \xrightarrow{\langle turn, heading, step \rangle} s',$$

$turn$ is the acting UAV, $heading$ indicates which target it is to visit and $step$ is a value greater or equal than $\max(0, FT(latest_{turn}, heading) - UC_{turn})$—the least time needed for UAV $turn$ to reach target $heading$. In s', UC_{turn} and $TC_{heading}$ are reset to 0 while the corresponding clocks of all other targets and UAVs are increased by $step$.

Example 7.2.1. Consider the following instance of the CR-UAV Problem with $k = 2$ (we number the targets in clockwise order, starting from bottom left):



A transition in the corresponding looping automaton is illustrated below, where the values of target clocks are highlighted in red and the values of UAV clocks are highlighted in blue.



In this example we have $turn = 0$ and $heading = 1$, so we know that UAV 0 must reach target 1 in s'. Note that the value of $step$ is greater than

$$FT(latest_0, heading) - UC_0 = FT(0, 1) - 0 = 3.$$

The clocks UC_0 and TC_1 are reset to 0 while TC_0, TC_2, UC_1 are increased by $step = 4$.

Formally, we define the transition relation as follows

forall $i \in \{0, \ldots, n-1\}$
  if $(i = heading)$ // target i is being visited
    $next(TC_i) := 0;$
  else
    $next(TC_i) := TC_i + step;$

forall $j \in \{0, \ldots, k-1\}$
  if $(j = turn)$ // UAV j is acting
    $next(UC_j) := 0;$
  else
    $next(UC_j) := UC_j + step;$

where

$$\max\left(0, FT(latest_{turn}, heading) - UC_{turn}\right) \leq step \leq RD_{min}$$

where RD_{min} is the smallest constant in RD. In addition, we exclude all transitions that result in $TC_i + step > RD(i)$ for some $i \in \{0, \ldots, n-1\}$—this amounts to violation of the relative deadline of target i—and let all addition operations on UC_j to saturate at FT_{max} for all $j \in \{0, \ldots, k-1\}$.

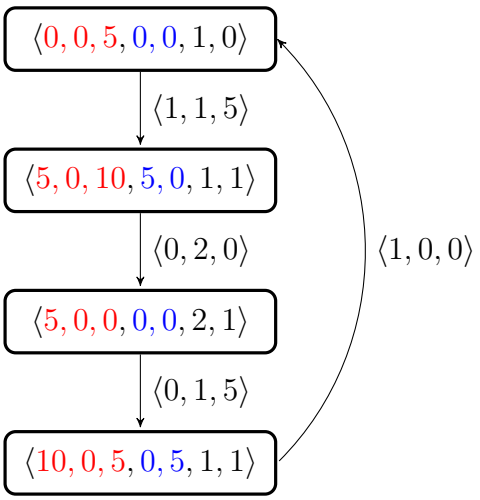
Exact modelling. The transition relation in this case is very similar except that we now require

- $FT(latest_{turn}, heading) \geq UC_{turn}$

- $step = FT(latest_{turn}, heading) - UC_{turn}$

to hold in each transition.

Example 7.2.2. Consider again the instance of the CR-UAV Problem in Example 7.2.1. A solution is witnessed by the following loop in the corresponding looping automaton obtained with standard modelling:



In contrast, the shortest loop in the corresponding looping automaton obtained with exact modelling consists of 6 states.

Finally, we let all states of $\mathcal{B}$ be both initial and accepting.

7.3 Antichains and Direct Simulations

Let $\mathcal{B}$ be a looping automaton obtained with standard modelling. In this section, we recall some notions from [DR10] and define a ***direct simulation*** induced naturally by the structure of $\mathcal{B}$. This enables us to use the standard antichain ***repeated reachability*** algorithm for the emptiness problem for $\mathcal{B}$.

7.3.1 Antichains

Let $\mathcal{T} = \langle S, \mathsf{Init}, E, \mathsf{Final} \rangle$ be a finite-state transition system where S is a finite set of states, E is the transition relation, Init and Final are the set of initial and accepting states, respectively.[2] The set of successors of a set of states $S_1 \subseteq S$ is denoted by $\mathsf{post}(S_1) = \{s' \mid \exists s \in S \big((s, s') \in E\big)\}$. Let $\mathsf{post}^0(S_1) = S_1$ and

[2]We keep the notations consistent with [DR10] whenever appropriate.

$\mathsf{post}^i(S_1) = \mathsf{post}\big(\mathsf{post}^{i-1}(S_1)\big)$ for $i \geq 1$. We denote by $\mathsf{post}^*(S_1)$ the set $\bigcup_{i\geq 0} \mathsf{post}^i(S_1)$ and by $\mathsf{post}^+(S_1)$ the set $\bigcup_{i\geq 1} \mathsf{post}^i(S_1)$. The classical fixed-point algorithm for repeated reachability (is there an infinite path that starts from Init and visits Final infinitely often?) is often employed when $\mathcal{T}$ is not given explicitly:

- $\mathsf{FF}(0) = \mathsf{Final} \cap \mathsf{post}^*(\mathsf{Init})$
- $\mathsf{FF}(i) = \mathsf{post}^+\big(\mathsf{FF}(i-1)\big) \cap \mathsf{Final}$ for all $i \geq 1$.

The algorithm computes this sequence until a fixed point FF^* and check if it is empty.

While the algorithm explores the state space of $\mathcal{T}$ *on-the-fly*, in many practical cases it is still infeasible as $\mathcal{T}$ can be of (say) exponential size. Fortunately, the standard algorithm can be improved by exploiting direct simulations on $\mathcal{T}$. In essence, for any set of states used in the algorithm above, we can manipulate only its most '*promising*' states rather than the set itself.[3] This idea is particularly useful when the structure of $\mathcal{T}$ admits an easily-computable direct simulation.

Definition 7.3.1. *A pre-order $\preceq$ over S is a direct simulation on $\mathcal{T} = \langle S, \mathsf{Init}, E, \mathsf{Final}\rangle$ if for all $s_1, s_2, s_3 \in S$ such that $s_2 \preceq s_1$ ('s_2 direct-simulates s_1') and $(s_1, s_3) \in E$, there exists $s_4 \in S$ such that $(s_2, s_4) \in E$ and $s_4 \preceq s_3$.*

Let $S_1, S_2 \subseteq S$. We say that a direct simulation $\preceq$ on $\mathcal{T}$ is *compatible* with S_1 if for all $s \in S_1$ and $s' \preceq s$, we must have $s' \in S_1$. Denote the set of minimal elements of S_1 by $\mathsf{Min}(\preceq, S_1) = \{s \in S_1 \mid \forall s' \in S_1\,(s' \preceq s \implies s \preceq s')\}$. We write $S_1 \sqsubseteq S_2$ if $\forall s \in S_1\,\big(\exists s' \in S_2\,(s' \preceq s)\big)$ and $S_1 \approx S_2$ if $S_1 \sqsubseteq S_2$ and $S_2 \sqsubseteq S_1$. Assuming that $\preceq$ is a direct simulation on $\mathcal{T}$ that is compatible with Final, the aforementioned algorithm can be replaced by an algorithm that computes the following sequence (the *forward repeated reachability sequence of promising states* [DR10]):

Sequence 1

- $\widehat{\mathsf{FF}}(0) = \mathsf{Min}\big(\preceq, \mathsf{Final} \cap \mathsf{post}^*(\mathsf{Init})\big)$
- $\widehat{\mathsf{FF}}(i) = \mathsf{Min}\Big(\preceq, \mathsf{post}^+\big(\widehat{\mathsf{FF}}(i-1)\big) \cap \mathsf{Final}\Big)$ for all $i \geq 1$.

In particular, $\mathsf{post}^+\big(\widehat{\mathsf{FF}}(i-1)\big)$ can be replaced by the fixed point of the following sequence:

[3] Another perspective (and hence a different theory) is to regard antichains as symbolic representations of closed sets; we refer the reader to [DR10] for details.

- $\widehat{\mathsf{F}}(0) = \mathsf{Min}\Big(\preceq, \mathsf{post}\big(\widehat{\mathsf{FF}}(i-1)\big)\Big)$
- $\widehat{\mathsf{F}}(j) = \mathsf{Min}\Big(\preceq, \mathsf{post}\big(\widehat{\mathsf{F}}(j-1)\big) \cup \widehat{\mathsf{F}}(j-1)\Big)$ for all $j \geq 1$.

The algorithm computes Sequence 1 until $\widehat{\mathsf{FF}}(i) \approx \widehat{\mathsf{FF}}(i+1)$ for some $i \geq 0$. Let $\widehat{\mathsf{FF}}^{\natural}$ denote the set $\widehat{\mathsf{FF}}(i)$ for the smallest such i.

Theorem 7.3.2 ([DR10]). *If $\preceq$ is a direct simulation on $\mathcal{T}$, then FF^* is non-empty iff $\widehat{\mathsf{FF}}^{\natural}$ is non-empty.*

7.3.2 A Direct Simulation

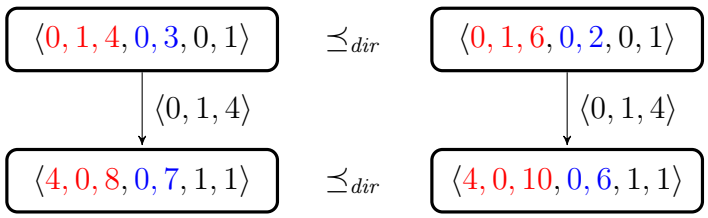
By dropping all the labels on the transitions of $\mathcal{B}$, we can obtain a corresponding finite-state transition system $\mathcal{T}_{\mathcal{B}} = \langle S_{\mathcal{B}}, \mathsf{Init}_{\mathcal{B}}, E_{\mathcal{B}}, \mathsf{Final}_{\mathcal{B}} \rangle$. We now define a direct simulation $\preceq_{dir}$ on $\mathcal{T}_{\mathcal{B}}$ based on the intuition that states with smaller values of target clocks and larger values of UAV clocks are more promising.

Definition 7.3.3. *Let $\mathcal{T}_{\mathcal{B}}$ be the finite-state transition system described above. For all states s and s' in $S_{\mathcal{B}}$, $s' \preceq_{dir} s$ iff all of the following holds (the relevant states are shown as superscripts):*

- *For all $i \in \{0, \ldots, n-1\}$, $TC_i^{s'} \leq TC_i^{s}$*
- *For all $j \in \{0, \ldots, k-1\}$, $UC_j^{s'} \geq UC_j^{s}$*
- *For all $j \in \{0, \ldots, k-1\}$, $latest_j^{s'} = latest_j^{s}$.*

Observe that for each $s_1, s_2, s_3 \in S_{\mathcal{B}}$ such that $s_2 \preceq_{dir} s_1$ and $(s_1, s_3) \in E_{\mathcal{B}}$, there is a transition $(s_2, s_4) \in E_{\mathcal{B}}$ that corresponds to a transition in $\mathcal{B}$ *labelled with the same symbol* as the corresponding transition of (s_1, s_3) in $\mathcal{B}$; furthermore, we must have $s_4 \preceq_{dir} s_3$.

Example 7.3.4. Consider the following states and transitions in the finite-state transition system $\mathcal{T}_{\mathcal{B}}$ obtained from Example 7.2.1 (on page 114):



7.4 Antichains and Delayed Simulations

7.4.1 Antichains + Constraint Solving

Let $\mathcal{T} = \langle S, \mathsf{Init}, E, \mathsf{Final} \rangle$ be a finite-state transition system. As in the case of standard algorithms [CP03], we can simplify the antichain algorithm in the last section if $\mathcal{T}$ is of some special type. In particular, the proposition below clearly holds.

Proposition 7.4.1. *If $\preceq$ is a direct simulation on $\mathcal{T}$ and $\mathsf{Init} = \mathsf{Final} = S$, then $\mathsf{post}^{i+1}(S) \sqsubseteq \mathsf{post}^{i}(S)$ for all $i \geq 0$.*

It follows from the transitivity of $\preceq_{dir}$ that, in place of Sequence 1, we can compute the simplified sequence below:

Sequence 2

- $\overline{\mathsf{FF}}(0) = \mathsf{Min}(\preceq, \mathsf{Init})$
- $\overline{\mathsf{FF}}(i) = \mathsf{Min}\Big(\preceq, \mathsf{post}\big(\overline{\mathsf{FF}}(i-1)\big)\Big)$ for all $i \geq 1$.

Since $\overline{\mathsf{FF}}(i+1) \sqsubseteq \overline{\mathsf{FF}}(i)$ for all $i \geq 0$, we can use $\overline{\mathsf{FF}}(i) \sqsubseteq \overline{\mathsf{FF}}(i+1)$ as the termination condition.

Proposition 7.4.2. *If $\preceq$ is a direct simulation on $\mathcal{T}$ and $\mathsf{Init} = \mathsf{Final} = S$, then $\overline{\mathsf{FF}}(i) = \widehat{\overline{\mathsf{FF}}}(i)$ for all $i \geq 0$.*

Unfortunately, even with these simplifications, solving the CR-UAV Problem with the approach described so far turned out to be too slow in practice. As we were unable to come up with a direct simulation coarser than $\preceq_{dir}$, we have to consider alternative ways to improve the performance.

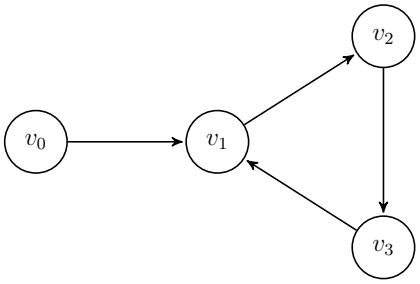
Constraint solving. For randomly generated instances (such as those used later in Section 7.5), it is often the case that solutions are very simple and short. This suggests that it might be practically advantageous to solve the problem with a constraint solver such as Z3 [dMB08]. However, as is the case with bounded model checking [BCCZ99], this approach is clearly impractical for *proving that no solution exists*: as we have seen in the last chapter, a loop in $\mathcal{T}_{\mathcal{B}}$ (which corresponds directly to a solution) can be exponential in the size of the original problem instance.

Delayed simulations. It is suggested in [WDMR08] that weaker notions of simulation (cf. [EWS05]) can possibly be used in place of direct simulations to further improve the performance of antichain algorithms. Intuitively, coarser simulation relations allow us to manipulate smaller antichains. One of such notions, called ***delayed simulation***, is defined as follows.

Definition 7.4.3. *A pre-order $\preceq$ over S is a delayed simulation on $\mathcal{T} = \langle S, \mathsf{Init}, E, \mathsf{Final} \rangle$ if for all $s_1, s_2, s_3 \in S$ such that $s_2 \preceq s_1$ ('s_2 delayed-simulates s_1') and $(s_1, s_3) \in E$, there exists $s_4 \in S$ such that s_4 is reachable from s_2 and $s_4 \preceq s_3$.*

As delayed simulations are more general and can be coarser than direct simulations, one would hope that they can be adopted to further improve the performance. However, the following examples show that delayed simulations cannot be used straightforwardly with the algorithms above.

Example 7.4.4. Consider the finite-state transition system $\mathcal{T} = \langle S, \mathsf{Init}, E, \mathsf{Final} \rangle$ with $\mathsf{Init} = \mathsf{Final} = S$ illustrated below:

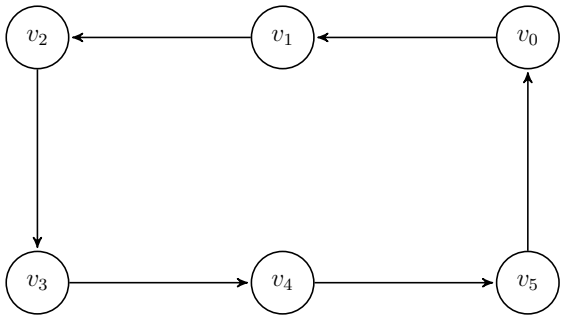


Let $\preceq$ be the reflexive transitive closure of $\{(v_0, v_1)\}$. The pre-order $\preceq$ can be seen as a delayed simulation as $v_1 \to v_2$ can be mimicked by $v_0 \to v_1 \to v_2$. On the other hand, $\preceq$ cannot be a direct simulation as if it is, we must have $(v_1, v_2) \in \preceq$ for v_0 to direct-simulate v_1. If we use $\preceq$ in the computation of Sequence 2, the algorithm would not terminate as

- $\overline{\mathsf{FF}}(0) = \{v_0, v_2, v_3\}$
- $\overline{\mathsf{FF}}(1) = \{v_1, v_3\}, \overline{\mathsf{FF}}(2) = \{v_1, v_2\}, \overline{\mathsf{FF}}(3) = \{v_2, v_3\}, \overline{\mathsf{FF}}(4) = \{v_1, v_3\}, \ldots$,

and therefore $\overline{\mathsf{FF}}(i) \sqsubseteq \overline{\mathsf{FF}}(i+1)$ never holds.

Example 7.4.5. Consider the finite-state transition system $\mathcal{T} = \langle S, \mathsf{Init}, E, \mathsf{Final} \rangle$ with $\mathsf{Init} = \mathsf{Final} = S$ illustrated below:



Let $\preceq$ be the reflexive transitive closure of $\{(v_0, v_1), (v_1, v_2), (v_3, v_4), (v_4, v_5)\}$. This can be seen as a delayed simulation as $v_1 \to v_2$ can be mimicked by $v_0 \to v_1 \to v_2$ and $v_2 \to v_3$ can be mimicked by $v_0 \to v_1 \to v_2 \to v_3$, etc. If we use $\preceq$ in the computation of Sequence 1 and, in particular, substitute the fixed point of the corresponding sequence $\widehat{\mathsf{F}}(j)$, $j \geq 0$ for $\mathsf{post}^+\big(\widehat{\overline{\mathsf{FF}}}(i-1)\big)$ in each step, we have

- $\widehat{\overline{\mathsf{FF}}}(0) = \{v_0, v_3\}$
- $\widehat{\overline{\mathsf{FF}}}(1) = \{v_1, v_4\}, \widehat{\overline{\mathsf{FF}}}(2) = \{v_0, v_3\}, \ldots$.

The algorithm would not terminate as $\widehat{\overline{\mathsf{FF}}}(i) \approx \widehat{\overline{\mathsf{FF}}}(i+1)$ never holds.

A portfolio approach. In order to obtain the supposed benefits of constraint solving and delayed simulations, we propose a very simple approach: let the constraint solver handle 'yes' instances (i.e., instances that have solutions) and the algorithm that computes Sequence 2 handle 'no' instances. More precisely, we can solve the problem by computing Sequence 2 and running the constraint solver (for which we increase incrementally the supposed length of solution) in two parallel threads; if any of the two threads returns an answer then we kill both threads. In the view of the examples above, we can simply suppress the (expensive) check of $\overline{\mathsf{FF}}(i) \sqsubseteq \overline{\mathsf{FF}}(i+1)$ and terminate the computation of Sequence 2 only when $\overline{\mathsf{FF}}(i)$ becomes empty. It is easy to see that the proposition below also holds for the case of delayed simulations.

Proposition 7.4.6. *If* $\mathsf{Final} = S$*, then* FF^* *is empty iff* $\overline{\mathsf{FF}}(i)$ *is empty for some* $i \geq 0$.

Proof. As the state space is finite, the forward direction clearly holds. For the other direction, it can be shown that if there is a finite path of length l starting from Init in $\mathcal{T}$, we must have $\overline{\mathsf{FF}}(i) \neq \emptyset$ for all i, $0 \leq i < l$. □

We remark that such a portfolio approach is very common in modern SAT-based model checkers, e.g., ABC [BM10] and IImc [HBS12].

7.4.2 A Delayed Simulation

We now define a delayed simulation $\preceq_{del}$ on $\mathcal{T}_{\mathcal{B}}$ on the basis of $\preceq_{dir}$ and the following observation: a state s_2 may delayed-simulate another state s_1 via a transition that corresponds to a self-loop in the network of timed automata $\mathcal{A}$. In other words, we can define $\preceq_{del}$ so that $s_2 \preceq_{del} s_1$ whenever a state s_1' such that $s_1' \preceq_{dir} s_1$ is reachable from s_2 by letting a UAV to 'stay at a target'. This idea is formalised in the following definition.

Definition 7.4.7. *Let $\mathcal{T}_{\mathcal{B}}$ be the finite-state transition system described above. For all states s and s' in $S_{\mathcal{B}}$, $s' \preceq_{del} s$ iff all of the following holds (the relevant states are shown as superscripts):*

- *For all $i \in \{0, \ldots, n-1\}$, $TC_i^{s'} \leq TC_i^s$*
- *$UC_j^{s'} \geq UC_j^s$ for all $j \in \{0, \ldots, k-1\}$ or there exists $i \in \{0, \ldots, n-1\}$, $j \in \{0, \ldots, k-1\}$ and $d \in \mathbb{N}_{>0}$ such that*
  - *$latest_j^{s'} = i$*
  - *$TC_i^{s'} = 0$*
  - *$UC_j^s = 0$*
  - *For all $i' \neq i$, $TC_{i'}^{s'} + d \leq TC_{i'}^s$*
  - *For all $j' \neq j$, $UC_{j'}^{s'} + d \geq UC_{j'}^s$*
- *For all $j \in \{0, \ldots, k-1\}$, $latest_j^{s'} = latest_j^s$.*

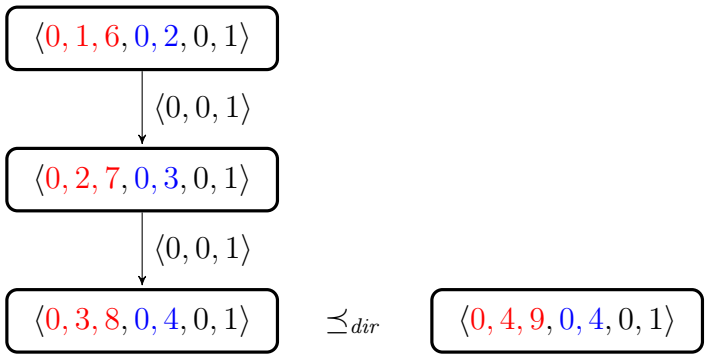
Proposition 7.4.8. *$\preceq_{del}$ is a pre-order over $S_{\mathcal{B}}$.*

Proof. It is clear that $\preceq_{del}$ is reflexive. For the transitivity, let $s_1 \preceq_{del} s_2 \preceq_{del} s_3$ and consider the following cases:

- $s_1 \preceq_{dir} s_2$ and $s_2 \preceq_{dir} s_3$: We have $s_1 \preceq_{dir} s_3$ and trivially $s_1 \preceq_{del} s_3$.
- $s_1 \preceq_{del} s_2$, $s_1 \not\preceq_{dir} s_2$ and $s_2 \preceq_{dir} s_3$: It is clear that by staying at a target (say, using UAV j), a state s_2' such that $s_2' \preceq_{dir} s_2$ is reachable from s_1. Since $s_2' \preceq_{dir} s_3$ and $UC_j^{s_3} = 0$, we also have $s_1 \preceq_{del} s_3$.
- $s_1 \preceq_{dir} s_2$, $s_2 \preceq_{del} s_3$ and $s_2 \not\preceq_{dir} s_3$: By staying at a target (say, at target i), a state s_3' such that $s_3' \preceq_{dir} s_3$ is reachable from s_2. Since $s_1 \preceq_{dir} s_2$ and $TC_i^{s_1} = 0$, we can do the same at s_1 to reach a state s_3'' such that $s_3'' \preceq_{dir} s_3'$. It follows that $s_3'' \preceq_{dir} s_3$ and thus $s_1 \preceq_{del} s_3$.

- $s_1 \preceq_{del} s_2$, $s_1 \not\preceq_{dir} s_2$, $s_2 \preceq_{del} s_3$ and $s_2 \not\preceq_{dir} s_3$: The claim follows by observing that in s_2, there must be a unique $i \in \{0, \ldots, n-1\}$ such that $TC_i^{s_2} = 0$. □

Example 7.4.9. Consider the following states and transitions in the finite-state transition system $\mathcal{T}_{\mathcal{B}}$ obtained from Example 7.2.1 (on page 114):



We have $\langle 0, 1, 6, 0, 2, 0, 1\rangle \preceq_{del} \langle 0, 4, 9, 0, 4, 0, 1\rangle$ as illustrated above.

7.5 Experimental Results

To evaluate the effectiveness of the proposed approach, we implemented a prototype tool in C++ using the AaPAL library [Boh14]. To emphasise the use of antichains, we focus on comparing the performance in solving 'no' instances, i.e., proving that no solution exists. In practice, we use a parallel thread to run Z3 incrementally; in our experience, this is clearly the fastest way to solve 'yes' instances. All experiments were conducted on an IBM x3550 (Intel Xeon X5680 Processor clocked at 3.33GHz (12C/24T) + 64GB RAM) with a timeout of 1200 seconds.

Test instances. We generated two sets of test instances as benchmarks. The first set ('easier instances') consists of 217 'no' instances (out of 500 randomly-generated instances) with parameters randomly selected as follows:

- *type* = {'Euclidean', 'Metric', 'Random'}
- $n \in \{4, 5, 6, 7, 8\}$
- $k \in \{1, 2, 3\}$.

For the 'Euclidean' type, we generate grid points in a 10000×10000 box and calculate the Euclidean distances between them. For the 'Metric' type, we generate latitude and longitude values and calculate the distances between them with the haversine

formula. These distances are then scaled down and filled into FT to make $FT_{max} = 20$. For the 'Random' type, we generate the values in FT (except '1's on the diagonal) from $\{1, \ldots, 20\}$. Finally, for all these types of instances, we generate the values in RD from $\{1, \ldots, 3FT_{max}\}$. For the second set ('harder instances'), we first generate 1000 instances with FT_{max} randomly selected from $\{20, 40, 60, 80, 100\}$ and other parameters selected as above. Then we pick the 'no' instances that require more than a second and less than 1200 seconds to solve by our semi-algorithm using $\preceq_{del}$; this amounts to 124 instances.

Tools and modelling. We compare the performance in solving the two benchmarks above with the following tools:

- FDR3 [GRABR14] is the standard refinement checker for CSP [Hoa85, Ros98, Ros10]. We model $\mathcal{B}$ in *tock*-CSP and formulate the emptiness problem as a divergence check.
- NuSMV [CCG+02] is the standard BDD-based symbolic model checker. We model $\mathcal{B}$ in its modelling language and formulate the problem as an $\mathsf{LTL}_{\mathsf{fut}}$ model-checking query.
- ABC [BM10] is a SAT-based model checker. For our purpose, we use it as follows: we translate the NuSMV models into AIGER format [Bie07] with a customised version of SMVFLatten [Bie14] and run ABC on them. We use the version that won the liveness track of the Hardware Model Checking Competition 2014 (HWMCC'14) [Ste14].
- IImc [HBS12] is the winner of the liveness track of HWMCC'13. We use it in the same way as ABC.
- Uppaal [BDL+06] is the standard real-time model checker. We model $\mathcal{A}$ in its modelling language and formulate the problem as a TCTL model-checking query.
- opaal + LTSmin [LOD+13] is an $\mathsf{LTL}_{\mathsf{fut}}$ model checker for timed automata. We use the Uppaal models and formulate the problem as an $\mathsf{LTL}_{\mathsf{fut}}$ model-checking query.

For each of the tools above, we build two sets of models that correspond respectively to the standard and exact (timed automata) modelling. Some of the tools must run on multiple cores (ABC and IImc); for the other tools, we set the number of available cores to 1.

Delayed simulation. We first evaluate the effect of using $\preceq_{del}$ in place of $\preceq_{dir}$ in the computation of Sequence 2. Since using $\preceq_{dir}$ by itself is still too slow, for this comparison we simplify the transition relation by forcing $step = \max\big(0, FT(latest_{turn}, heading) - UC_{turn}\big)$.[4] It is easy to see that this modification does not affect correctness. However, $\preceq_{dir}$ is no longer a direct simulation but a delayed simulation. The results are illustrated in Figures 7.1 and 7.2.[5]

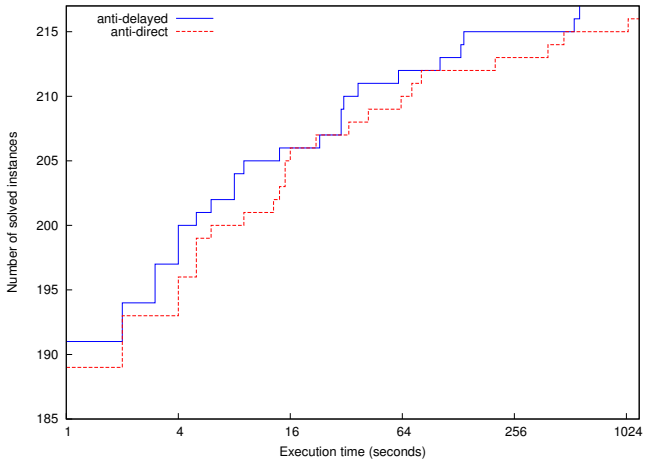


Figure 7.1: Delayed vs. direct simulation for the easier instances.

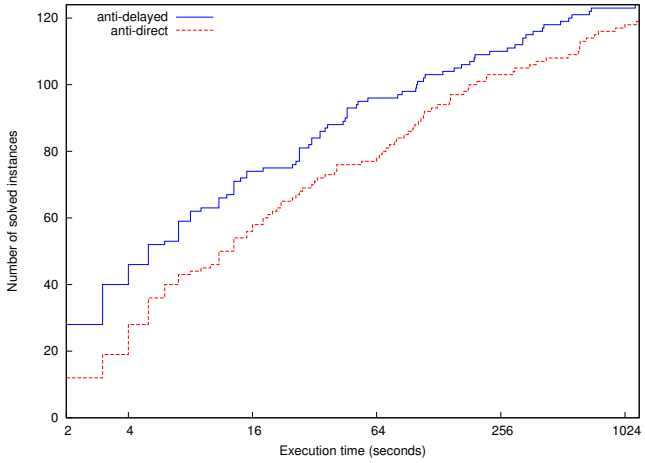


Figure 7.2: Delayed vs. direct simulation for the harder instances.

We see that for the easier instances, using $\preceq_{del}$ is only marginally faster than using $\preceq_{dir}$. For the harder instances, using $\preceq_{del}$ is often faster by more than 100%.

[4]This simplification is already implicit in the definition of $\preceq_{del}$.

[5]Following [CS12], we put the logarithmic-scale time axis at bottom; this allows one to roughly estimate the ratio between runtimes.

Comparison with untimed model checkers. The second comparison is between the semi-algorithm (using $\preceq_{del}$) and the untimed model checkers. The results are illustrated in Figure 7.3. We remark that none of the harder instances can be solved in 1200 seconds by any of the untimed model checkers considered here.

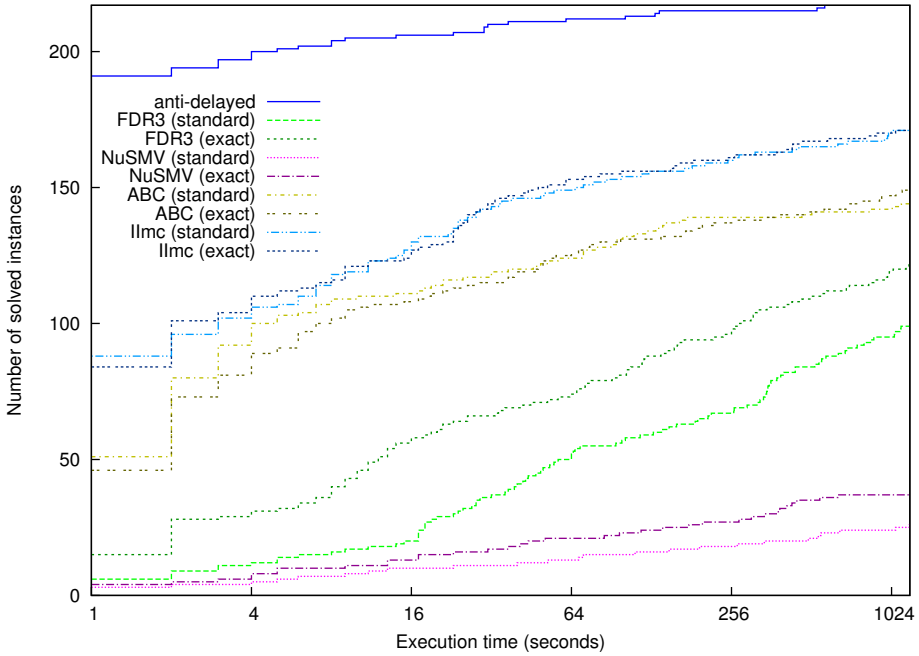


Figure 7.3: Comparison with untimed model checkers for the easier instances.

We see that FDR3 significantly outperforms NuSMV for these instances. This is somewhat surprising as FDR3 uses explicit state representations internally. We also note that FDR3 is faster with the exact modelling; this is expected since the number of possible paths is reduced. It is also clear from the figure that IImc performs better than ABC for these instances. However, since both ABC and IImc use three to four solver engines, it is difficult to draw deeper conclusions about their underlying algorithms here. Finally, using $\preceq_{del}$ is clearly much faster than the untimed model checkers considered here.

Comparison with timed model checkers. We compare the performance of the semi-algorithm (using $\preceq_{del}$) and that of UPPAAL and opaal + LTSMIN. The results are illustrated in Figures 7.4 and 7.5. Detailed execution times (for the harder instances) are given in Figures 7.6 and 7.7 where '—' represents a time-out.

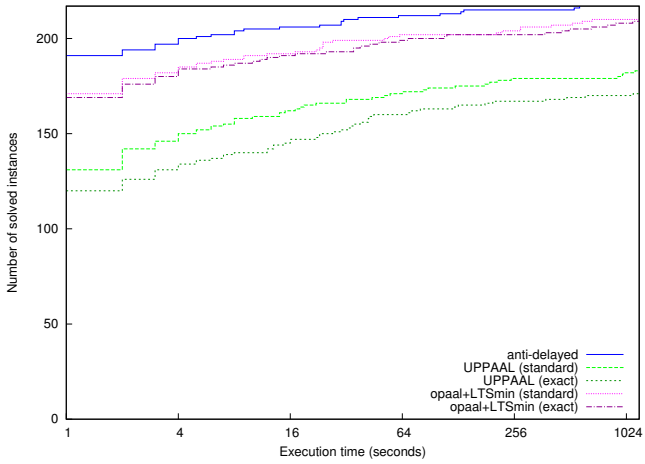


Figure 7.4: Comparison with timed model checkers for the easier instances.

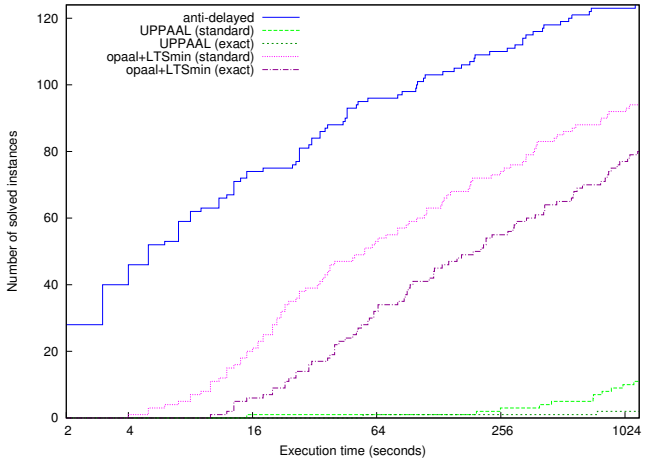


Figure 7.5: Comparison with timed model checkers for the harder instances.

Comparing Figure 7.4 with Figure 7.3, we conclude that the timed model checkers are generally faster than the untimed model checkers in our case. For the easier instances, opaal + LTSMIN is faster than UPPAAL by at least an order of magnitude. For the harder instances, this gap is more evident: UPPAAL is only able to solve a little number of hard instances, yet opaal + LTSMIN can solve most of them. Finally, observe that using $\preceq_{del}$ is more than four times faster than opaal + LTSMIN for the harder instances. We also see that in contrast to the case of FDR3, UPPAAL and opaal + LTSMIN work better with the standard modelling. A possible explanation is that guards of the form '$UC \geq 0$' are more amenable to the abstraction techniques employed in these tools (e.g., zone subsumption in opaal + LTSMIN).

instance	anti-direct	anti-delayed	UPPAAL (standard)	UPPAAL (exact)	opaal + LTSMIN (standard)	opaal + LTSMIN (exact)
0-020-5-3-848	33	13	—	—	373	418
0-020-6-2-427	2	2	—	—	27	40
0-020-6-2-975	11	5	—	—	20	30
0-020-6-3-851	—	367	—	—	—	—
0-020-7-2-569	542	488	—	—	—	—
0-020-8-3-564	66	26	—	—	384	938
0-040-5-2-481	9	4	—	—	33	133
0-040-5-2-515	25	13	—	—	465	—
0-040-5-2-970	18	7	—	—	38	121
0-040-5-3-383	—	688	—	—	—	—
0-040-5-3-509	169	45	—	—	517	878
0-040-5-3-586	8	4	—	—	230	413
0-040-5-3-893	68	27	—	—	—	—
0-040-6-2-743	22	11	—	—	36	157
0-040-6-3-344	40	15	—	—	1038	1180
0-040-7-2-198	299	181	—	—	—	—
0-040-7-2-247	2	2	—	—	18	26
0-040-7-2-542	2	2	720	—	4	12
0-040-8-2-881	146	85	—	—	—	—
0-060-4-2-121	2	2	—	—	5	90
0-060-5-2-229	180	108	—	—	—	—
0-060-5-2-753	10	4	—	—	17	37
0-060-6-2-35	219	110	—	—	1077	—
0-060-6-3-306	16	8	793	—	187	478
0-060-7-2-369	3	2	394	—	14	52
0-060-7-2-690	5	2	—	—	29	122
0-060-7-2-691	13	5	—	—	20	26
0-080-4-3-803	196	46	—	—	—	—
0-080-5-2-389	3	2	—	—	6	13
0-080-5-2-449	6	3	—	—	17	20
0-080-5-2-774	16	9	—	—	81	1081
0-080-5-2-898	2	2	195	—	5	13
0-080-6-2-191	144	44	—	—	112	297
0-080-6-2-327	4	3	—	—	20	87
0-080-6-2-418	666	327	—	—	—	—
0-080-6-2-59	54	25	—	—	255	—
0-080-6-2-957	4	2	—	—	10	23
0-080-6-3-551	96	81	—	—	—	—
0-080-6-3-858	41	18	—	—	—	—
0-080-8-2-591	5	3	—	—	27	62
0-080-8-2-977	425	152	—	—	266	—
0-100-4-2-307	26	11	—	—	65	563
0-100-4-2-459	105	37	—	—	326	—
0-100-4-2-597	13	6	—	—	37	376
0-100-4-2-827	4	2	—	—	10	40
0-100-5-2-44	98	27	—	—	92	305
0-100-5-3-216	87	31	717	—	339	—
0-100-6-2-24	13	5	—	—	56	293
0-100-6-2-422	294	134	—	—	—	—
0-100-6-2-448	126	58	—	—	340	—
0-100-6-2-66	108	51	—	—	780	—
0-100-7-2-744	2	3	—	—	63	23
1-020-6-2-641	3	2	—	—	18	30
1-020-7-3-217	—	703	—	—	—	—
1-020-7-3-465	2	2	15	55	23	45
1-020-7-3-50	618	275	—	—	—	—
1-020-7-3-997	3	2	—	—	56	65
1-040-4-2-809	2	2	—	—	15	31
1-040-5-2-223	5	3	—	—	26	165
1-040-5-2-93	5	2	—	—	12	49
1-040-8-2-155	6	5	—	—	71	115
1-040-8-2-437	4	2	—	—	9	18

Figure 7.6: Execution times for the harder instances (seconds).

instance	anti-direct	anti-delayed	UPPAAL (standard)	UPPAAL (exact)	opaal + LTSMIN (standard)	opaal + LTSMIN (exact)
1-040-8-2-813	15	7	—	—	140	341
1-040-8-3-698	1023	546	—	—	—	—
1-060-4-2-588	4	2	—	—	12	81
1-060-5-2-80	5	2	—	—	49	211
1-060-5-3-134	761	300	—	—	—	—
1-060-6-2-262	178	100	—	—	591	—
1-060-6-2-271	11	5	—	—	38	217
1-060-7-2-908	608	226	—	—	—	—
1-060-7-3-171	36	15	—	—	368	1054
1-060-8-2-109	920	414	—	—	—	—
1-060-8-3-30	—	1149	—	—	—	—
1-080-4-2-453	2	2	—	—	21	143
1-080-5-2-283	31	13	—	—	109	821
1-080-5-2-645	80	34	—	—	480	—
1-080-6-2-368	20	8	—	—	131	966
1-080-6-2-519	28	11	—	—	81	637
1-080-6-2-899	146	46	—	—	178	839
1-080-7-2-302	5	3	—	—	40	205
1-080-8-2-532	91	52	—	—	181	—
1-080-8-2-54	1166	408	—	—	858	—
1-080-8-2-666	617	339	—	—	—	—
1-080-8-2-712	11	12	—	—	286	277
1-080-8-2-846	729	327	—	—	—	—
1-100-4-2-61	71	30	—	—	842	—
1-100-4-2-739	4	2	—	—	21	91
1-100-4-2-923	7	4	—	—	22	96
1-100-6-2-420	380	165	—	—	—	—
1-100-6-2-714	18	7	—	—	89	600
1-100-6-3-873	354	99	—	—	—	—
1-100-7-2-640	4	2	—	—	7	13
1-100-8-3-615	109	46	—	—	—	—
1-100-8-3-632	27	13	—	—	784	853
2-020-7-2-347	5	3	1001	—	24	65
2-020-7-2-775	4	3	—	—	34	58
2-020-8-3-832	5	4	255	752	100	188
2-040-4-3-817	102	34	—	—	—	—
2-040-5-3-267	6	3	—	—	133	232
2-040-5-3-725	620	192	—	—	—	—
2-040-6-2-230	22	7	—	—	50	54
2-060-4-2-64	2	2	880	—	11	62
2-060-7-2-494	218	190	—	—	—	—
2-060-7-2-78	32	27	—	—	—	780
2-060-7-2-791	11	4	—	—	112	24
2-080-4-2-14	21	8	—	—	183	—
2-080-6-2-428	15	5	—	—	22	59
2-080-6-2-782	4	2	—	—	10	15
2-080-6-3-202	118	101	—	—	—	—
2-080-7-2-278	—	566	—	—	—	—
2-080-7-2-438	2	2	451	—	8	38
2-080-7-2-520	3	2	—	—	12	10
2-080-7-2-74	6	3	—	—	16	42
2-080-7-2-927	3	2	—	—	14	51
2-100-5-2-228	2	2	1131	—	8	20
2-100-5-2-270	78	27	—	—	136	579
2-100-5-2-761	7	3	—	—	23	88
2-100-5-3-419	13	7	—	—	146	418
2-100-6-2-617	41	14	—	—	35	217
2-100-6-2-720	3	2	—	—	13	121
2-100-7-2-98	7	3	—	—	15	40
2-100-8-2-235	19	7	—	—	61	93
2-100-8-2-482	74	36	—	—	569	—
2-100-8-2-993	64	31	—	—	380	573

Figure 7.7: Execution times for the harder instances (seconds).

7.6 Discussion

We proposed an antichain-based approach to the CR-UAV problem. The crux of the approach is that it can be used with coarser *delayed* simulations. The effectiveness of our approach is demonstrated experimentally: our prototype tool clearly outperforms the state-of-the-art model checkers.

An obvious way to improve the performance is to divide all constants in a given instance (except '1's on the diagonal of FT) by their greatest common divisor whenever possible. It is not hard to see that this yields an equivalent instance, i.e., the simplified instance has a solution iff the original instance has one. Furthermore, when all constants are coprime, we can adopt the standard notion of weakening and strengthening specifications [HMP92]. For example, if there is no solution to the instance obtained by (i) dividing all the constants by a number and (ii) rounding up all the relative deadlines and (iii) rounding down all the flight times, then there is also no solution to the original instance.

Another possibility to improve the models (or the algorithm) is to apply symmetry reduction [ES96]. Intuitively, since all UAVs are identical, it is not really necessary to distinguish between, e.g., $\langle 0, 1, 6, 0, 2, 0, 1\rangle$ and $\langle 0, 1, 6, 2, 0, 1, 0\rangle$ (the two states here are of the form $\langle TC_0, TC_1, TC_2, UC_0, UC_1, latest_0, latest_1\rangle$). The reasons we chose not to adopt the idea in the current prototype are (i) the current modelling is simpler and clearer; (ii) the current version of UPPAAL can only utilise symmetry reduction in verifying safety properties [HBL$^+$03], and we do not know whether the other tools use symmetry reduction at all.

It is possible to modify the semi-algorithm (computing Sequence 2 until $\overline{\mathsf{FF}}(i)$ becomes empty for some $i \geq 0$) to make it always terminate by itself. For example, following the idea of the standard liveness-to-safety translation [BAS02], we can add to each state an additional state-recording component to detect loops; this yields a (sound and complete) algorithm for the emptiness problem for looping automata. We have indeed implemented this algorithm; yet, whilst its performance in proving that no solution exists is not seriously degraded (slowed down by about 30%), it is often much slower than Z3 in finding solutions (loops). For this reason, it is not reported here. On the other hand, since the state space is finite, in 'yes' instances we must have $\overline{\mathsf{FF}}(i) \approx \overline{\mathsf{FF}}(j)$ for some $j > i$. However, it is not clear to us how to bound the difference between i and j. We speculate that *a priori* bounds on the 'delay' of delayed simulations might help, but in our case this bound is still too large.

Finally, multi-core architectures have become more popular in recent years. Indeed,

a principal design goal of both FDR3 and LTSmin is to exploit such architectures effectively. We leave as future work to investigate how to combine parallelism with the antichain approach.

Chapter 8

Conclusion and Future Work

In this chapter, we summarise our contributions and pose some open questions as directions for further research.

8.1 Summary

We considered the expressiveness of MTL over bounded timed words in Chapter 3. We showed that, in contrast to the situation in the continuous semantics, the various fragments of MTL form a strict hierarchy of expressiveness in this setting. Moreover, we proposed new first-order definable modalities 'generalised Until' and 'generalised Since' which, when added into MTL, yield a metric temporal logic (which we call $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$) that is expressively complete for $\mathsf{FO}[<, +1]$ over bounded timed words. The time-bounded satisfiability and time-bounded model-checking problems for $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$ were shown to be EXPSPACE-complete, the same as that of MTL. As a side result, we lifted the decision procedures for the time-bounded satisfiability and time-bounded model-checking problems [ORW09] to the pointwise case. This enables direct applications of standard discrete-time tools for LTL/Büchi automata in implementations. We then showed in Chapter 4 that the expressive completeness result extends to the case of infinite timed words if we allow rational endpoints, solving an implicit open problem in the field.

In Chapter 5, we proposed a monitoring procedure for an expressively complete fragment of $\mathsf{MTL}[\mathfrak{U}, \mathfrak{S}]$. In terms of expressiveness, the procedure strictly generalises all known procedures in the literature. More importantly, the procedure is trace-length independent, does not use dynamic memory, recursion, etc., and therefore admits efficient hardware implementations.

In Chapter 6, we considered a recurrent UAV path-planning problem (which we call the CR-UAV Problem) where a number of targets have to be visited infinitely often at intervals of given maximal duration. We proved that the problem is already PSPACE-complete in the metric single-UAV case and thus refuted an erroneous claim in the literature.

In Chapter 7, we proposed an antichain-based approach to the CR-UAV Problem. In contrast to the standard antichain algorithm (where a *direct* simulation is used to prune the state space), we exploited a *delayed* simulation in our approach to prune the state space even further. We showed empirically that for the CR-UAV Problem, our approach is much more efficient than the state-of-the-art model checkers.

8.2 Future Work

We now discuss the main remaining open questions along the lines of research in this thesis. Some other possibilities for future work were addressed in the Discussion sections of relevant chapters.

Open Question 1. *Is the integer version of* MTL *with counting modalities expressively complete for* FO[$<, +1$] *in the pointwise semantics?*

In light of [Hun13], a possible route for establishing a positive answer is to go through Q2MLO. Unfortunately, all the results on Q2MLO are based on the continuous semantics and not easily adapted to the pointwise semantics. Indeed, some of the simplest equivalence rules in [HR04], e.g.,

$$(\exists t)_{>t_0}^{\leq t_0+1} X(t) = (\exists t)_{>t_0}^{\leq t_0+1} X(t) \vee \big((\neg X \,\mathcal{U}\, X) \wedge (\forall t_1)_{>t_0}^{\leq t_0+1} (\exists t)_{>t_1}^{\leq t_1+1} X(t)\big)\,,$$

already fail in the pointwise semantics (consider the case when there is no event in $(t_0, t_0 + 1)$). We, however, suspect there is a way to express decomposition formulas $\delta(x, y)$ using counting modalities and thus avoid going through Q2MLO.

Open Question 2. *Assuming bounded variability k, can we establish a better lower bound on the size of a deterministic timed automaton that detects all good (bad) prefixes for a given* $\mathsf{MTL}_{\mathsf{fut}}$ *formula φ?*

The best lower bound we are aware of is $\Omega(2^{2^{\sqrt{|\varphi|}}})$, inherited from $\mathsf{LTL}_{\mathsf{fut}}$ [KV01]. However, we believe it could be at least an exponential higher, as the complexity of the satisfiability problem for $\mathsf{MTL}_{\mathsf{fut}}$ over infinite timed words of bounded variability k

(assuming a unary encoding of k) is EXPSPACE-hard [FR08], higher by an exponential than that of $\mathsf{LTL}_{\mathsf{fut}}$. Another point of interest is that the lower bound for $\mathsf{LTL}_{\mathsf{fut}}$ holds even when we allow non-deterministic automata; in other words, the power of non-determinism does not help here. Is this also true in the timed case?

Open Question 3. *Is the Euclidean version of the* CR-UAV *Problem* PSPACE-*complete?*

The reduction in Chapter 6 crucially depends on the use of clause vertices and consistency gadgets, which unfortunately are also the main sources of the 'non-Euclideanity' of constructed weighted graphs. The construction used in the proof of (the weaker claim) Proposition 6.2.2 (on page 97) is free of these gadgets, but we still could not adapt it even to the three-dimensional Euclidean case. On the other hand, we note that the counterexample to the erroneous bound (on page 96) is itself Euclidean, so the Euclidean version of the problem is at least not trivially in NP (by that bound) even when constants are encoded in unary. Another possibility would be to devise a reduction from another PSPACE-complete problem.

Open Question 4. *How to exploit delayed simulations in the antichain approach for more general classes of finite automata?*

The difficulties in adopting Büchi acceptance conditions are that we have to detect all loops and check whether any of the loops contains an accepting state, both of which seem non-trivial in our present algorithm in Chapter 7. In fact, we did not even know how detecting a single loop—this amounts to checking termination in the looping case—can be done efficiently. It is conceivable that knowing *a priori* the maximum 'delays' in delayed simulations might be helpful, but it is not clear to us whether this can be faster than using direct simulations.

Bibliography

[ABLS05] Oliver Arafat, Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification revisited. Technical Report TUM-I0518, Technische Universität München, 2005.

[ACD90] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking for real-time systems. In *Proceedings of LICS 1990*, pages 414–427. IEEE Computer Society Press, 1990.

[AD94] Rajeev Alur and David Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[AH90] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. In *Proceedings of LICS 1990*, pages 390–401. IEEE Computer Society Press, 1990.

[AH92] Rajeev Alur and Thomas A. Henzinger. Back to the future: towards a theory of timed regular languages. In *Proceedings of FOCS 1992*, pages 177–186. IEEE Computer Society Press, 1992.

[AKH03] Mehdi Alighanbari, Yoshiaki Kuwata, and Jonathan P How. Coordination and control of multiple uavs with timing constraints and loitering. In *Proceedings of ACC 2003*, volume 6, pages 5311–5316. IEEE Press, 2003.

[AKT+06] Roy Armoni, Dmitry Korchemny, Andreas Tiemeyer, Moshe Y. Vardi, and Yael Zbar. Deterministic dynamic monitors for linear-time assertions. In *Proceedings of FATES/RV 2006*, volume 4262 of *LNCS*, pages 163–177. Springer, 2006.

[Alu98] Rajeev Alur. Timed automata. In *NATO-ASI Summer School on Verification of Digital and Hybrid Systems*. Springer, 1998.

[AM04] Rajeev Alur and Parthasarathy Madhusudan. Decision problems for timed automata: A survey. In *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *LNCS*, pages 1–24. Springer, 2004.

[AS87] Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.

[BAS02] Armin Biere, Cyrille Artho, and Viktor Schuppan. Liveness checking as safety checking. *Electronic Notes in Theoretical Computer Science*, 66(2):160–177, 2002.

[BBBB09] Christel Baier, Nathalie Bertrand, Patricia Bouyer, and Thomas Brihaye. When are timed automata determinizable? In *Proceedings of ICALP 2009*, volume 5556 of *LNCS*, pages 43–54. Springer, 2009.

[BBD+11] Thomas Brihaye, Véronique Bruyère, Laurent Doyen, Marc Ducobu, and Jean-François Raskin. Antichain-based QBF solving. In *Proceedings of ATVA 2011*, volume 6996 of *LNCS*, pages 183–197. Springer, 2011.

[BBF+12] Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. Acacia+, a tool for LTL synthesis. In *Proceedings of CAV 2012*, volume 7358 of *LNCS*, pages 652–657. Springer, 2012.

[BCCZ99] Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In *Proceedings of TACAS 1999*, volume 1579 of *LNCS*, pages 193–207. Springer-Verlag, 1999.

[BCE+14] David A. Basin, Germano Caronni, Sarah Ereth, Matús Harvan, Felix Klaedtke, and Heiko Mantel. Scalable offline monitoring. In *Proceedings of RV 2014*, volume 8734 of *LNCS*, pages 31–47. Springer, 2014.

[BCM05] Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. In *Proceedings of FSTTCS 2005*, volume 3821 of *LNCS*, pages 432–443. Springer, 2005.

[BDG+11] Thomas Brihaye, Laurent Doyen, Gilles Geeraerts, Joël Ouaknine, Jean-François Raskin, and James Worrell. On reachability for hybrid

automata over bounded time. In *Proceedings of ICALP 2011*, volume 6756 of *LNCS*, pages 416–427. Springer, 2011.

[BDL+06] Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Håkansson, Paul Pettersson, Wang Yi, and Martijn Hendriks. UPPAAL 4.0. In *Proceedings of QEST 2006*, pages 125–126. IEEE Computer Society Press, 2006.

[BEG14] Thomas Brihaye, Morgane Estiévenart, and Gilles Geeraerts. On MITL and alternating timed automata over infinite words. In *Proceedings of FORMATS 2014*, volume 8711 of *LNCS*, pages 69–84. Springer, 2014.

[BGA09] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Developing a deterministic patrolling strategy for security agents. In *Proceedings of WI-IAT 2009*, pages 565–572. IEEE Computer Society Press, 2009.

[BGA12] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial Intelligence*, 184–185:78–123, 2012.

[BHKK07] Hans-Joachim Böckenhauer, Juraj Hromkovic, Joachim Kneis, and Joachim Kupke. The parameterized approximability of TSP with deadlines. *Theory of Computing Systems*, 41(3):431–444, 2007.

[Bie07] Armin Biere. The aiger and-inverter graph (aig) format. *Available at fmv.jku.at/aiger*, 2007.

[Bie14] Armin Biere. Available at fmv.jku.at/smvflatten, 2014.

[Bir93] Jean-Camille Birget. State-complexity of finite-state devices, state compressibility and incompressibility. *Mathematical Systems Theory*, 26(3):237–269, 1993.

[BKMP08] David Basin, Felix Klaedtke, Samuel Müller, and Birgit Pfitzmann. Runtime monitoring of metric first-order temporal properties. In *Proceedings of FSTTCS 2008*, volume 2 of *LIPIcs*, pages 49–60. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2008.

[BKV13] A Bauer, JC Küster, and Gil Vegliach. From propositional to first-order monitoring. In *Proceedings of RV 2013*, volume 8174 of *LNCS*, pages 59–75. Springer, 2013.

[BKZ11] David Basin, Felix Klaedtke, and Eugene Zălinescu. Algorithms for monitoring real-time properties. In *Proceedings of RV 2011*, volume 7186 of *LNCS*, pages 260–275. Springer, 2011.

[BLS11] Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology*, 20(4):14, 2011.

[BM10] Robert K. Brayton and Alan Mishchenko. ABC: An academic industrial-strength verification tool. In *Proceedings of CAV 2010*, volume 6174 of *LNCS*, pages 24–40. Springer, 2010.

[BMOW07] Patricia Bouyer, Nicolas Markey, Joël Ouaknine, and James Worrell. The cost of punctuality. In *Proceedings of LICS 2007*, pages 109–120. IEEE Computer Society Press, 2007.

[BMT99] Marius Bozga, Oded Maler, and Stavros Tripakis. Efficient verification of timed automata using dense and discrete time semantics. In *Proceedings of CHARME 1999*, volume 1703 of *LNCS*, pages 125–141. Springer, 1999.

[BN12] Kevin Baldor and Jianwei Niu. Monitoring dense-time, continuous-semantics, metric temporal logic. In *Proceedings of RV 2012*, volume 7687 of *LNCS*, pages 245–259. Springer, 2012.

[BO14] Daniel Bundala and Joël Ouaknine. On the complexity of temporal-logic path checking. In *Proceedings of ICALP 2014*, volume 8573 of *LNCS*, pages 86–97. Springer, 2014.

[Boh14] Aaron Bohy. *Antichain based algorithms for the synthesis of reactive systems*. PhD thesis, University of Mons, 2014.

[Büc60] J. Richard Büchi. Weak second order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.

[CCG+02] Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. NuSMV2: An opensource tool for symbolic model checking. In *Proceedings of CAV 2002*, volume 2404 of *LNCS*, pages 359–364. Springer, 2002.

[CE81] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Proceedings of IBM Workshop on Logic of Programs*, volume 131 of *LNCS*, pages 52–71. Springer-Verlag, 1981.

[CLT07] Edmund M. Clarke, Flavio Lerda, and Muralidhar Talupur. An abstraction technique for real-time verification. In *Next Generation Design and Verification Methodologies for Distributed Embedded Control Systems*, pages 1–17. Springer Netherlands, 2007.

[CP03] Ivana Cerná and Radek Pelánek. Relating hierarchy of temporal properties to model checking. In *Proceedings of MFCS 2003*, volume 2747 of *LNCS*, pages 318–327. Springer, 2003.

[CS76] Ashok K. Chandra and Larry J. Stockmeyer. Alternation. In *Proceedings of FOCS 1976*, pages 98–108. IEEE Computer Society Press, 1976.

[CS12] Koen Claessen and Niklas Sörensson. A liveness checking algorithm that counts. In *Proceedings of FMCAD 2012*, pages 52–59. IEEE Computer Society Press, 2012.

[CS13] Ming Chai and Holger Schlingloff. A rewriting based monitoring algorithm for TPTL. In *Proceedings of CS&P 2013*, volume 1032 of *CEUR Workshop Proceedings*, pages 61–72. CEUR-WS.org, 2013.

[CVDK97] Yves Crama and Joris Van De Klundert. Cyclic scheduling of identical parts in a robotic cell. *Operations Research*, 45(6):952–965, 1997.

[Dam94] Mads Dam. Temporal logic, automata, and classical theories - an introduction, 1994.

[DHO+15] Nir Drucker, Hsi-Ming Ho, Joël Ouaknine, Michal Penn, and Ofer Strichman. Cyclic routing of unmanned aerial vehicle. In preparation, 2015.

[DHV07] Deepak D'Souza, Raveendra Holla, and Deepak Vankadaru. On the expressiveness of TPTL in the pointwise and continuous semantics. Unpublished manuscript, 2007.

[DKL10] Christian Dax, Felix Klaedtke, and Martin Lange. On regular temporal logics with past. *Acta Informatica*, 47(4):251–277, 2010.

[DM13] Deepak D'Souza and Raj Mohan Matteplackel. A clock-optimal hierarchical monitoring automaton construction for MITL. Technical Report 2013-1, Department of Computer Science and Automation, Indian Institute of Science, 2013.

[dMB08] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Proceedings of TACAS 2008*, volume 4963 of *LNCS*, pages 337–340. Springer-Verlag, 2008.

[dMPPPP14] André de Matos Pedro, David Pereira, Luís Miguel Pinho, and Jorge Sousa Pinto. A compositional monitoring framework for hard real-time systems. In *Proceedings of NFM 2014*, volume 8430 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2014.

[DP06] Deepak D'Souza and Pavithra Prabhakar. On the expressiveness of MTL in the pointwise and continuous semantics. *International Journal on Software Tools for Technology Transfer*, 9(1):1–4, 2006.

[DPS10] Nir Drucker, Michal Penn, and Ofer Strichman. Cyclic routing of unmanned air vehicles. Technical Report IE/IS-2014-02, Faculty of Industrial Engineering and Management, Technion, 2010.

[DR10] Laurent Doyen and Jean-François Raskin. Antichains algorithms for finite automata. In *Proceedings of TACAS 2010*, volume 6015 of *LNCS*, pages 2–22. Springer, 2010.

[DT04] Deepak D'Souza and Nicolas Tabareau. On timed automata with input-determined guards. In *Proceedings of FORMATS/FTRTFT 2004*, volume 3253 of *LNCS*, pages 68–83. Springer, 2004.

[EFHL03] Cindy Eisner, Dana Fisman, John Havlicek, and Yoad Lustig. Reasoning with temporal logic on truncated paths. In *Proceedings of CAV 2003*, volume 2725 of *LNCS*, pages 27–39. Springer, 2003.

[EL86] E. Allen Emerson and Chin-Laung Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Proceedings of LICS 1986*, pages 267–278. IEEE Computer Society Press, 1986.

[ES96] E. Allen Emerson and A. Prasad Sistla. Symmetry and model checking. *Formal Methods in System Design*, 9:105–131, 1996.

[EW96] Kousha Etessami and Thomas Wilke. An until hierarchy for temporal logic. In *Proceedings of LICS 1996*, pages 108–117. IEEE Computer Society Press, 1996.

[EWS05] Kousha Etessami, Thomas Wilke, and Rebecca A. Schuller. Fair simulation relations, parity games, and state space reduction for bu"chi automata. *SIAM Journal on Computing*, 34(5):1159–1175, 2005.

[FC05] Eugene A. Feinberg and Michael T. Curry. Generalized pinwheel problem. *Mathematical Methods of Operations Research*, 62(1):99–122, 2005.

[FJY90] Abdelaziz Fellah, Helmut Jürgensen, and Sheng Yu. Constructions for alternating finite automata. *International Journal of Computer Mathematics*, 35(1-4):117–132, 1990.

[FK09] Bernd Finkbeiner and Lars Kuhtz. Monitor circuits for LTL with bounded and unbounded future. In *Proceedings of RV 2009*, volume 5779 of *LNCS*, pages 60–75. Springer, 2009.

[FMDR13] Tim French, John Christopher McCabe-Dansted, and Mark Reynolds. Verifying temporal properties in real models. In *Proceedings of LPAR 2013*, volume 8312 of *LNCS*, pages 309–323. Springer, 2013.

[FR08] Carlo A. Furia and Matteo Rossi. MTL with bounded variability: Decidability and complexity. In *Proceedings of FORMATS 2008*, volume 5215 of *LNCS*, pages 109–123. Springer, 2008.

[Gei01] Marc Geilen. On the construction of monitors for temporal logic properties. *Electronic Notes in Theoretical Computer Science*, 55(2):181–199, 2001.

[GGL13] Gilles Geeraerts, Joël Goossens, and Markus Lindström. Multiprocessor schedulability of arbitrary-deadline sporadic tasks: complexity and antichain algorithm. *Real-Time Systems*, 49(2):171–218, 2013.

[GHR94] Dov M. Gabbay, Ian Hodkinson, and Mark Reynolds. *Temporal Logics: Mathematical Foundations and Computational Aspects, Volume 1*. Oxford University Press, 1994.

[GO03] Paul Gastin and Denis Oddoux. LTL with past and two-way very-weak alternating automata. In *Proceedings of MFCS 2003*, volume 2747 of *LNCS*, pages 439–448. Springer, 2003.

[GPSS80] Dov Gabbay, Amir Pnueli, Sharanon Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of POPL 1980*, pages 163–173. ACM Press, 1980.

[GPV94] Aleks Göllü, Anuj Puri, and Pravin Varaiya. Discretization of timed automata. In *Proceedings of CDC 1994*, pages 957–958, 1994.

[GRABR14] Thomas Gibson-Robinson, Philip J. Armstrong, Alexandre Boulgakov, and A. W. Roscoe. FDR3 - A modern refinement checker for CSP. In *Proceedings of TACAS 2014*, volume 8413 of *LNCS*, pages 187–201. Springer, 2014.

[GT14] Hendra Gunadi and Alwen Tiu. Efficient runtime monitoring with metric temporal logic: A case study in the android operating system. In *Proceedings of FM 2014*, volume 8442 of *LNCS*, pages 296–311. Springer, 2014.

[HBL+03] Martijn Hendriks, Gerd Behrmann, Kim Guldstrand Larsen, Peter Niebert, and Frits W. Vaandrager. Adding symmetry reduction to uppaal. In *Proceedings of FORMATS 2003*, volume 2791 of *LNCS*, pages 46–59. Springer, 2003.

[HBS12] Zyad Hassan, Aaron R. Bradley, and Fabio Somenzi. Incremental, inductive CTL model checking. In *Proceedings of CAV 2012*, volume 7358 of *LNCS*, pages 532–547. Springer, 2012.

[HMP92] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. What good are digital clocks? In *Proceedings of ICALP 1992*, volume 623 of *LNCS*, pages 545–558. Springer, 1992.

[Ho14] Hsi-Ming Ho. On the expressiveness of metric temporal logic over bounded timed words. In *Proceedings of RP 2014*, volume 8762 of *LNCS*, pages 138–150. Springer, 2014.

[HO15] Hsi-Ming Ho and Joël Ouaknine. The CR-UAV problem is pspace-complete. In *Proceedings of FoSSaCS 2015*, volume 9034 of *LNCS*, pages 328–342. Springer, 2015.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[Hol97] Gerard J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.

[HOW13] Paul Hunter, Joël Ouaknine, and James Worrell. Expressive completeness of metric temporal logic. In *Proceedings of LICS 2013*, pages 349–357. IEEE Computer Society Press, 2013.

[HOW14] Hsi-Ming Ho, Joël Ouaknine, and James Worrell. Online monitoring of metric temporal logic. In *Proceedings of RV 2014*, volume 8734 of *LNCS*, pages 178–192. Springer, 2014.

[HP94] Gerard J. Holzmann and Doron Peled. An improvement in formal verification. In *Proceedings of FORTE 1994*, volume 6 of *IFIP Conference Proceedings*, pages 197–211. Chapman & Hall, 1994.

[HR01] Klaus Havelund and Grigore Roşu. Testing linear temporal logic formulae on finite execution traces. Technical Report RIACS 01.08, Research Institute for Advanced Computer Science, 2001.

[HR04] Yoram Hirshfeld and Alexander Moshe Rabinovich. Logics for real time: Decidability and complexity. *Fundamenta Informaticae*, 62(1):1–28, 2004.

[HR06] Yoram Hirshfeld and Alexander Moshe Rabinovich. An expressive temporal logic for real time. In *Proceedings of MFCS 2006*, volume 4162 of *LNCS*, pages 492–504. Springer, 2006.

[HR07] Yoram Hirshfeld and Alexander Rabinovich. Expressiveness of metric modalities for continuous time. *Logical Methods in Computer Science*, 3(1), 2007.

[HRS98] Thomas A. Henzinger, Jean-François Raskin, and Pierre-Yves Schobbens. The regular real-time languages. In *Proceedings of ICALP 1998*, volume 1443 of *LNCS*, pages 580–591. Springer, 1998.

[HSW10] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Efficient emptiness check for timed Büchi automata. In *Proceedings of CAV 2010*, volume 6174 of *LNCS*, pages 148–161. Springer, 2010.

[Hun13] Paul Hunter. When is metric temporal logic expressively complete? In *Proceedings of CSL 2013*, volume 23 of *LIPIcs*, pages 380–394. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[JKK+10] Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordóñez, and Milind Tambe. Security games with arbitrary schedules: A branch and price approach. In *Proceedings of AAAI 2010*, pages 792–797. AAAI Press, 2010.

[Kam68] Johan A. Kamp. *Tense logic and the theory of linear order*. PhD thesis, University of California, Los Angeles, 1968.

[KC05] Sascha Konrad and Betty H. C. Cheng. Real-time specification patterns. In *Proceedings of ICSE 2005*, pages 372–381. ACM Press, 2005.

[KF09] Lars Kuhtz and Bernd Finkbeiner. LTL path checking is efficiently parallelizable. In *Proceedings of ICALP 2009*, volume 5556 of *LNCS*, pages 235–246. Springer, 2009.

[KKP11] Dileep Kini, Shankara N. Krishna, and Paritosh Pandya. On construction of safety signal automata for MITL[U,S] using temporal projections. In *Proceedings of FORMATS 2011*, volume 6919 of *LNCS*, pages 225–239. Springer, 2011.

[KL97] Vladimir Kats and Eugene Levner. Minimizing the number of robots to meet a given cyclic schedule. *Annals of Operations Research*, 69:209–226, 1997.

[Koy90] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[KP05] Pavel Krcál and Radek Pelánek. On sampled semantics of timed systems. In *Proceedings of FSTTCS 2005*, volume 3821 of *LNCS*, pages 310–321. Springer, 2005.

[KV01] Orna Kupferman and Moshe Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.

[LFHKG13] Jonathan Las Fargeas, Baro Hyun, Pierre Kabamba, and Anouck Girard. Persistent visitation under revisit constraints. In *Proceedings of ICUAS 2013*, pages 952–957. IEEE Press, 2013.

[LKdPC10] Eugene Levner, Vladimir Kats, David Alcaide López de Pablo, and T. C. Edwin Cheng. Complexity of cyclic scheduling problems: A state-of-the-art survey. *Computers & Industrial Engineering*, 59(2):352–361, 2010.

[LOD+13] Alfons Laarman, Mads Chr. Olesen, Andreas Engelbredt Dalsgaard, Kim Guldstrand Larsen, and Jaco van de Pol. Multi-core emptiness checking of timed Büchi automata using inclusion abstraction. In *Proceedings of CAV 2013*, volume 8044 of *LNCS*, pages 968–983. Springer, 2013.

[LOW13] Ranko Lazic, Joël Ouaknine, and James Worrell. Zeno, hercules and the hydra: Downward rational termination is ackermannian. In *Proceedings of MFCS 2013*, volume 8087 of *LNCS*, pages 643–654. Springer, 2013.

[LPZ85] Orna Lichtenstein, Amir Pnueli, and Lenore D. Zuck. The glory of the past. In *Proceedings of Logics of Programs 1985*, volume 193 of *LNCS*, pages 196–218. Springer, 1985.

[LS09] Martin Leucker and Christian Schallhart. A brief account of runtime verification. *Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009.

[LW08] Slawomir Lasota and Igor Walukiewicz. Alternating timed automata. *ACM Transactions on Computational Logic*, 9(2), 2008.

[McM93] Kenneth L. McMillan. *Symbolic model checking*. Kluwer, 1993.

[MH84] Satoru Miyano and Takeshi Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32(3):321–330, 1984.

[MHSR95] Madhav V. Marathe, Harry B. Hunt, III, Richard E. Stearns, and Venkatesh Radkakrishnan. Complexity of hierarchically and 1-dimensional periodically specified problems. Technical Report LA-UR–95-3348; CONF-960233–2, Los Alamos National Lab., NM (United States), 1995.

[MHSR98] Madhav V. Marathe, Harry B. Hunt III, Richard E. Stearns, and Venkatesh Radhakrishnan. Approximation algorithms for PSPACE-hard hierarchically and periodically specified problems. *SIAM Journal on Computing*, 27(5):1237–1261, 1998.

[MN04] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Proceedings of FORMATS/FTRTFT 2004*, volume 3253 of *LNCS*, pages 152–166. Springer, 2004.

[MNP05] Oded Maler, Dejan Nickovic, and Amir Pnueli. Real time temporal logic: Past, present, future. In *Proceedings of FORMATS 2005*, volume 3829 of *LNCS*, pages 2–16. Springer, 2005.

[MNP06] Oded Maler, Dejan Nickovic, and Amir Pnueli. From MITL to timed automata. In *Proceedings of FORMATS 2006*, volume 4202 of *LNCS*, pages 274–289. Springer, 2006.

[MP95] Zohar Manna and Amir Pnueli. *Temporal verification of reactive systems: safety*, volume 2. Springer, 1995.

[MS03] Nicolas Markey and Philippe Schnoebelen. Model checking a path (preliminary report). In *Proceedings of CONCUR 2003*, volume 2761 of *LNCS*, pages 251–265. Springer, 2003.

[NP10] Dejan Nickovic and Nir Piterman. From MTL to deterministic timed automata. In *Proceedings of FORMATS 2010*, volume 6246 of *LNCS*, pages 152–167. Springer, 2010.

[Orl81] James B Orlin. The complexity of dynamic languages and dynamic optimization problems. In *Proceedings of STOC 1981*, pages 218–227. ACM Press, 1981.

[Orl82] James B Orlin. Minimizing the number of vehicles to meet a fixed periodic schedule: An application of periodic posets. *Operations Research*, 30(4):760–776, 1982.

[ORW09] Joël Ouaknine, Alexander Rabinovich, and James Worrell. Time-bounded verification. In *Proceedings of CONCUR 2009*, volume 5710 of *LNCS*, pages 496–510. Springer, 2009.

[OW03] J. Ouaknine and J. Worrell. Timed CSP = closed timed epsilon-automata. *Nordic Journal of Computing*, 10(2):1–35, 2003.

[OW05] Joël Ouaknine and James Worrell. On the decidability of metric temporal logic. In *Proceedings of LICS 2005*, pages 188–197. IEEE Computer Society Press, 2005.

[OW06a] Joël Ouaknine and James Worrell. On metric temporal logic and faulty turing machines. In *Proceedings of FoSSaCS 2006*, volume 3921 of *LNCS*, pages 217–230. Springer, 2006.

[OW06b] Joël Ouaknine and James Worrell. Safety metric temporal logic is fully decidable. In *Proceedings of TACAS 2006*, volume 3920 of *LNCS*, pages 411–425. Springer, 2006.

[OW08] Joël Ouaknine and James Worrell. Some recent results in metric temporal logic. In *Proceedings of FORMATS 2008*, volume 5215 of *LNCS*, pages 1–13. Springer, 2008.

[OW10] Joël Ouaknine and James Worrell. Towards a theory of time-bounded verification. In *Proceedings of ICALP 2010*, volume 6199 of *LNCS*, pages 22–37. Springer, 2010.

[Pap77] Christos H. Papadimitriou. The Euclidean traveling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237–244, 1977.

[PD06] Pavithra Prabhakar and Deepak D'Souza. On the expressiveness of MTL with past operators. In *Proceedings of FORMATS 2006*, volume 4202 of *LNCS*, pages 322–336. Springer, 2006.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *Proceedings of FOCS 1977*, pages 46–57. IEEE Computer Society Press, 1977.

[PS11] Paritosh K. Pandya and Simoni S. Shah. On expressive powers of timed logics: Comparing boundedness, non-punctuality and deterministic freezing. In *Proceedings of CONCUR 2011*, volume 6901 of *LNCS*, pages 60–75. Springer, 2011.

[QS82] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In *Proceedings of Symposium on Programming 1982*, volume 137 of *LNCS*, pages 337–351. Springer, 1982.

[Rab10] Alexander Rabinovich. Temporal logics over linear time domains are in PSPACE. In *Proceedings of RP 2010*, volume 6227 of *LNCS*, pages 29–50. Springer, 2010.

[Rey10] Mark Reynolds. The complexity of temporal logic over the reals. *Annals of Pure and Applied Logic*, 161(8):1063–1096, 2010.

[Rey14] Mark Reynolds. Metric temporal logics and deterministic timed automata (long report version). Technical report, University of West Australia, 2014.

[RH02] Arthur Richards and Jonathan P How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of ACC 2002*, volume 3, pages 1936–1941. IEEE Press, 2002.

[Ric53] Henry Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 83, 1953.

[Ros98] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1998.

[Ros10] A. W. Roscoe. *Understanding Concurrent Systems*. Springer, 2010.

[Roş12] Grigore Roşu. On safety properties and their monitoring. *Scientific Annals of Computer Science*, 22(2):327–365, 2012.

[RS59] Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[RS97] Jean-François Raskin and Pierre-Yves Schobbens. State clock logic: A decidable real-time logic. In *Proceedings of HART 1997*, volume 1201 of *LNCS*, pages 33–47. Springer, 1997.

[Sav85] Martin WP Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4(1):285–305, 1985.

[SC85] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[SHL11] Oleg Sokolsky, Klaus Havelund, and Insup Lee. Introduction to the special section on runtime verification. *International Journal on Software Tools for Technology Transfer*, 14(3):243–247, 2011.

[Sip12] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[SL14] Youcheng Sun and Giuseppe Lipari. A weak simulation relation for real-time schedulability analysis of global fixed priority scheduling using linear hybrid automata. In *Proceedings of RTNS 2014*, page 35. ACM, 2014.

[SR12] K. Sundar and S. Rathinam. Route planning algorithms for unmanned aerial vehicles with refueling constraints. In *Proceedings of ACC 2012*, pages 3266–3271. IEEE Press, 2012.

[Ste14] Baruch Sterin. Personal Communication, 2014.

[Sto74] Larry Stockmeyer. The complexity of decision problems in automata theory and logic. PhD thesis, TR 133, M.I.T., Cambridge, 1974.

[TKO+09] Jason Tsai, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Shyamsunder Rathi. IRIS—a tool for strategic security allocation in transportation networks. In *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2009.

[TR05] Prasanna Thati and Grigore Roşu. Monitoring algorithms for metric temporal logic specifications. *Electronic Notes in Theoretical Computer Science*, 113:145–162, 2005.

[Tri02] Stavros Tripakis. Fault diagnosis for timed automata. In *Proceedings of FTRTFT 2002*, volume 2469 of *LNCS*, pages 205–224. Springer, 2002.

[Tur36] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[TYB05] Stavros Tripakis, Sergio Yovine, and Ahmed Bouajjani. Checking timed Büchi automata emptiness efficiently. *Formal Methods in System Design*, 26(3):267–292, 2005.

[UAV] Unmanned air vehicle systems association. `http://www.uavs.org/`.

[Var96] Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency – Structure versus Automata (8th Banff Higher Order Workshop'95)*, volume 1043 of *LNCS*, pages 238–266. Springer, 1996.

[Var01] Moshe Y. Vardi. Branching vs. linear time: Final showdown. In *Proceedings of TACAS 2001*, volume 2031 of *LNCS*, pages 1–22. Springer, 2001.

[WDHR06] Martin De Wulf, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Antichains: A new algorithm for checking universality of finite automata. In *Proceedings of CAV 2006*, volume 4144 of *LNCS*, pages 17–30. Springer, 2006.

[WDMR08] Martin De Wulf, Laurent Doyen, Nicolas Maquet, and Jean-François Raskin. Antichains: Alternative algorithms for LTL satisfiability and model-checking. In *Proceedings of TACAS 2008*, volume 4963 of *LNCS*, pages 63–77. Springer, 2008.

[Wil94] Thomas Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *Proceedings of FTRTFT 1994*, volume 863 of *LNCS*, pages 694–715. Springer, 1994.

[Wol90] Richard D. Wollmer. An airline tail routing algorithm for periodic schedules. *Networks*, 20(1):49–54, 1990.

[WVS83] Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. Reasoning about infinite computation paths. In *Proceedings of FOCS 1983*, pages 185–194. IEEE Computer Society Press, 1983.

[YK02] Guang Yang and Vikram Kapila. Optimal path planning for unmanned air vehicles with kinematic and tactical constraints. In *Proceedings of CDC 2002*, volume 2, pages 1301–1306. IEEE Press, 2002.

[ZAZ04] Wenrui Zhao, Mostafa H. Ammar, and Ellen W. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of MobiHoc 2004*, pages 187–198. ACM Press, 2004.