MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Monitoring of Systems Behaviour

DIPLOMA THESIS

Jiří Šimša

Brno, 2006

Declaration

I hereby declare that the thesis entitled *Monitoring of Systems Behaviour* is my own, unaided work and has never before been submitted for any degree. Where other sources of information have been used, they have been acknowledged.

Thesis advisor: doc. RNDr. Luboš Brim, CSc.

Acknowledgement

I would like to express my deep sense of gratitude to all those who made a contribution to this thesis.

First, my sincere thanks go to my supervisor Luboš Brim as well as other members of the Laboratory of Parallel and Distributed Systems—ParaDiSe in particular to Jiří Barnat, Ivana Černá, Jakub Chaloupka, Pavel Moravec, Radek Pelánek, and Jan Strejček.

Next, I would like to thank to Gerd Behrman, Kim Larsen, and Jiří Srba for englightning discussions and scientific guidance and Alexander David for technical support during my stay at the Aalborg University in Denmark.

I am also very grateful to my family for their continuos encouragment that was given to me. The acknowledgement episode would not be complete without mentioning my girlfriend Barbora whose patience was tested probably the most. Nevertheless, she has kindly provided her loving support and understanding.

Abstract

Usage of monitoring for the purpose of verification has grown in importance in recent years. In this thesis, we survey monitoring of a system for the purpose of a formal verification of its behaviour. To this aim, we use *temporal logics* and *observation sequences* as formalisms for describing system specification and its actual behaviour.

In particular, we present an unifying view on monitoring of *Linear Temporal Logic* and *Metric Interval Temporal Logic* specifications. Furthermore, we present an extension of this view to handle monitoring of systems with unobservable parts and inaccurate measurement of time.

Keywords

Verification, Testing, Model Checking, Monitoring, Linear Temporal Logic, Metric Interval Temporal Logic, Formula Rewriting, Partial Observability, Measurement Inaccuracies

Contents

Chapter 1

Introduction

"To err is human, but to really foul up requires a computer."
- Dan Rather

It is then no wonder that verification in its widest sense has been a major research topic since the beginings of computer science. Its general purpose is to resolve whether a verified system conforms to its specification, that is to detect errors in its realization.

Verification can take on many different forms—ranging from very simple to very complex methods. An example of a method that has become very popular is *testing* [7], of software in particular, as it is very efficient in detecting likely errors. However, it is not an universal verification technique as it scarcely detects rare errors. Yet these errors, despite the rarity of their occurence, may have serious consequences.

To cope with such errors, other methods such as *model checking* [12] or *theorem proving* [14] have been developed. Benefit of these methods is their ability to *prove* the correctness of a system, or at least of its model. Nevertheless, the applicability of these methods is often limited by their computational complexity and the level of knowledge necessary for their utilization. However, with the ongoing advance of both technologies and techniques, complex verification methods are becoming more and more accessible.

Somewhere in within the field of verification stands *monitoring*. In general, it is a process of observing a system for any changes which may occur over time. In the context of verification, monitoring is typically used for observing a system for errors. From one point of view, monitoring can be regarded as a continuos test of a system. Alternatively, it can be viewed as a light-weight version of model checking useful in situations where a model of a system is either unavailable or too complex.

These views are reflected in the following approaches. *Online monitoring* is a process of observing an activity of a system as it progresses. For example, the Unix command `top` follows the online monitoring paradigm;

observing an activity in the random-access memory. On the contrary, *offline monitoring* is a process of observing an activity of a system after it has occured. A typical example of offline monitoring is accessing log files of a system.

Example Let us consider some mechanical device which can read and write data to and from a data medium. For example a CD-RW mechanic and a CD-RW disc are examples of such a device and data medium. Naturally, we assume the mechanic to perform read or write operations only when a medium is inside. To verify the assumption we can extend the mechanic with coloured diodes or some other monitoring device, which indicate read and write operations. Then we simply monitor the assumption while we operate the device.

In this thesis we focus on—both online and offline—monitoring of a system for the purpose of formal verification of its behaviour. To this aim, we use *temporal logics* and *observation sequences* as formalisms for describing system specification and modeling its actual behaviour.

In particular, we consider two logics, *Linear Temporal Logic* (LTL) and a fragment of *Metric Interval Temporal Logic* (MITL$_{\leq}$). The former has been extensively studied for its utilization in the model checking and is widely used as a specification tool. The latter can be viewed as a real-time extension of the former and is consequently more complicated to handle.

We show that monitoring of formulas from both logics is feasible using the *formula rewriting* approach. In this approach the original formula is iteravely transformed according to the behaviour being observed. Thus all information necessary for resolution of the specification validity is stored in the resulting formula and there is no need to store the observed behaviour.

Our contribution

First of all, we present an unifying view on monitoring of both logics and, to this end, we use an approach based on formula rewriting.

Furthermore, we investigate the problem of monitoring of a system with unobservable parts—an issue reffered to as *partial observability*. For that purpose, we extend our algorithms for monitoring of formulas from both logics. The extension enables resolution of formulas for all potential behaviours of unobservable parts at the same time. Yet it does not increase asymptotical complexity.

Last but not least, we examine real-time monitoring of timing constraints in connection with a limited precision of time measurements—an issue ref-

fered to as *measurement inaccuracies*. We show how to deal with this issue using an abstraction.

Related work

From one point of view, monitoring of a formal specification can be regarded as model checking of a path. This problem has been studied by Markey and Schnoebelen [25]. They investigated complexity of deciding different logics over a single path as a special case of traditional model checking.

Geilen [16] studied possibilities of monitoring of LTL properties using an automata over both infinite and finite words. Later, he has extended his work to real-time logics. His thesis [17] is a comprehensive overview of the model checking of formulas from both LTL and $\text{MITL}_{\leq}$ based on the automata-theoretic approach.

The idea of formula rewriting, used throughout this thesis, has been previously employed by Thati and Grosu [28]. They proposed monitoring techniques for *Metric Temporal Logic* (MTL) and provided a detailed analysis of practical complexity of these techniques.

Thesis structure

In Chapter 2, we lay theoretical foundations for modeling of both desired and observed behaviour of a system. For that purpose, we define observation sequences, their timed extension and consequently syntax and semantics of both LTL and $\text{MITL}_{\leq}$.

Next, we investigate monitoring of LTL in Chapter 3. We review existing algorithms and then thoroughly describe an algorithm based on formula rewriting. Last section of the chapter is devoted to monitoring of LTL with respect to partial observability.

Chapter 4 explores monitoring of $\text{MITL}_{\leq}$. Similarly to LTL, we first review existing algorithms and then thoroughly describe an algorithm based on formula rewriting. In the last section of this chapter we discuss monitoring of $\text{MITL}_{\leq}$ with respect to both partial observability and measurement inaccuracies.

The thesis is concluded in Chapter 5 where we make conclusions and point out directions for future work.

Chapter 2

Theoretical foundations

"All models are wrong. Some are useful."
- George Box

In this chapter we lay theoretical foundations for describing observed as well as desired behaviour of a system. As the quotation above indicates, we cannot expect to have a perfect model. However, we choose an approach which can—under a certain assumption—provide an arbitrarily precise model of a system behaviour.

2.1 Modeling observed behaviour

When observing a system, we can identify its *state*, which can change as time passes, and its *actions*, which may occur in time. These observable features can be used for describing a single behaviour of the system or the system as a whole and give raise to state-based and action-based modeling methods. Naturally, actions can be viewed as changes of a state and states as conseqeunces of actions. Therefore choosing between state-based and action-based methods is just a matter of taste.

For the purpose of this thesis, we choose a state-based method, which identifies a state with a finite set of *boolean propositions*. Assuming that we are factually able to resolve validity of any potential proposition, this approach provides an arbitrary precision of modeling. Benefits of this approach are its universality—ability to model various systems—and its flexibility in implementing various levels of abstraction.

Definition 2.1 *Let $\mathcal{P}$ be a set of boolean propositions. Then any subset of $\mathcal{P}$ is a* state.

Intuitively, a single behaviour of a system could be represented as a sequence of states. By doing so one would abstract away from a very important aspect of the system behaviour. This aspect is time. Clearly, we may

want the system to behave according to certain timing constraints and representing its behaviour simply as a sequence of states makes the verification of such constraints virtually impossible.

Example Now we show how a state of a CD-RW mechanic can be described using boolean propositions. Let o denotes a fact that the mechanic is opened, r denotes a fact that the mechanic performs a read operation, w denotes a fact that the mechanic performs a write operation and lastly m denotes a fact that there is a medium inside the mechanic. Then $\{r, m\}$ denotes a state in which there is a medium inside the mechanic and the mechanic is closed, performs a read operation and does not perform a write operation. Analogously, $\{o, w\}$ denotes a state in which there is no medium inside the mechanic and the mechanic is opened, performs a write operation and does not perform a read operation.

However, incorporation of time into the modeling domain has its disadvantages. Some verification problems may become more complex or even undecidable. Moreover, when observing a system we cannot measure the timing of changes of its state precisely and yet we would like our method to be arbitrarily precise. In the following, we consider both possibilities, that is a model with as well as without time.

Observation sequences

To describe behaviours being monitored, we define an *observation sequence* as a sequence of states and a *timed observation sequence* as an observation sequence augmented with *intervals* of time. These intervals partition the time axis.

Definition 2.2 *An* interval *is a convex subset of reals. Given an interval I its left endpoint – finite or inifinite – is denoted $l(I)$ and its right endpoint – finite or infinite – is denoted $r(I)$. An interval I is* adjacent *to an interval I' if and only if $r(I) = l(I')$ and either $r(I)$ is included in I or $l(I')$ is included in I'.*

Definition 2.3 *An* interval sequence $\mathcal{I} = (I_0, I_1, I_2, \ldots)$ *is a sequence of intervals such that:*

- $I_0 = [0, a)$ *for some* $a \in \mathbb{R}_0^+ \cup \{\infty\}$.
- *For every index i, $l(I_i)$ is included in I_i.*
- *For every index i, I_i is adjacent to I_{i+1}.*
- *For every $t \in \mathbb{R}_0^+$, there exists k such that $t \in I_k$.*

An interval sequence defined above serves as a partitioning of the time axis. Note that we are considering only intervals which are left-closed and right-opened. Although we could consider all types of intervals, it would have no additional benefit. It is not possible to measure the time with infinite precision in practice. Therefore use of all types of intervals would only imply technical complications. The choice we has made reflects our intuition a about change of system state. It takes place at a certain time and until that time the system remains in its original state.

Definition 2.4 *Let $\mathcal{P}$ is a set of* propositions. *An* observation sequence *is a sequence $w = (a_0, a_1, a_2, \ldots)$ of states. A* timed observation sequence *is a pair $\rho = (w, \mathcal{I})$ where w is an observation sequence and $\mathcal{I}$ is an interval sequence and $|w| = |\mathcal{I}|$.*

Clearly, this is only one of many possible approaches we could have used for incorporation of time. For an overview of alternative approaches we refer to a survey by Alur and Henzinger [2].

Example Imagine that you operate a CD-RW mechanic as follows. Initially the mechanic is empty and idle. You open it and insert a medium. Then you close the mechanic and store some data on the medium. Finally you open the mechanic, remove the medium and close the mechanic. This behaviour of the mechanic could be modeled as a finite observation seqeunce $(\emptyset, \{o\}, \{o, m\}, \{m\}, \{m, w\}, \{m\}, \{o, m\}, \{o\}, \emptyset)$.
Now you repeat the scenario and also measure time. You open the mechanic at time 1 and insert a medium at time 3. Then you close the mechanic at time 7 and start a write operation at time 15. However, the mechanic get stuck and you are unable to complete the write operation. This behaviour could be modeled as a finite timed observation sequence $((\emptyset, \{o\}, \{o, m\}, \{m\}, \{m, w\}), ((0, 1], (1, 3], (3, 7], (7, 15], (15, \infty]))$.

We conclude this section by definition of a *suffix* of a (timed) observation sequence. This notion provides for a more compact reasoning in the remainder of the thesis.

Definition 2.5 *Let $\mathcal{I} = (I_0, I_1, \ldots)$ is an interval sequence, $w = (a_0, a_1, \ldots)$ is an observation sequence and $\rho = (w, \mathcal{I})$ is a timed observation sequence.*

- *For every $n = 0, \ldots, |w|$, an observation sequence $w^n = (a_n, a_{n+1}, \ldots)$ is a* suffix *of w starting at a_n.*
- *For every $t \in \mathbb{R}_0^+$, an interval sequence $\mathcal{I}^t = ([0, r(I_n) - t), I_{n+1} - t, \ldots)$ is a* suffix *of an interval sequence $\mathcal{I}$ at time t where $n \in \mathbb{N}_0$ is such that $t \in I_n$.*

- *For every* $t \in \mathbb{R}_0^+$*, a timed observation sequence* $\rho^t = (w^n, \mathcal{I}^t)$ *is a* suffix *of a timed observation sequence* ρ *at time* t *where* $n \in \mathbb{N}_0$ *is such that* $t \in I_n$*.*

2.2 Modeling desired behaviour

Having defined a method for modeling behaviours that a system can exhibit, we now move on to define a method for describing the behaviour that we would like the system to exhibit, that is its *specification*.

As opposed to a single behaviour resulting from an observation of a system, it is quite natural to specify the desired behaviour through an infinite set of behaviours all conforming to certain requirements we put on the system. To this aim, we make use of a commonly and widely used formalism—*temporal logics*.

Temporal logics

The field of temporal logics have a long tradition. It was pioneered by Manna and Pnueli and since then contributions have been made by many others. Temporal logics are particularly useful for specifying behaviour of *reactive* systems, that is systems that maintain an ongoing interaction with their environment and usually are not designed to terminate. Very nice description of temporal logics can be found in Manna and Pnueli's book [24].

Originally, temporal logics could not specify any timing constraints. In the early 1990s, Alur and Henzinger among others made a substantial contribution to the theory of specification and verification of real-time systems. They explored several ways of how to incorporate timing constraints into temporal logic, namely bounded temporal operators. An overview of real-time temporal logics can be found in their paper [3].

We present two temporal logics, one for expressing properties without timing constraints such as safety, liveness or reactivity properties and another that extends the first one, allows for timing constraints, and is able to express properties such as bounded response properties.

The first logic considered is *Linear Temporal Logic* (LTL) that has been widely used for specification [15] and is supported by many existing tools [4, 19]. We use this logic for expressing a (possibly infinite) set of observation sequences and thus specify the desired behaviour of a system.

Definition 2.6 *The syntax of* LTL *is defined as follows*

$$\varphi ::= \mathit{true} \mid \mathit{false} \mid p \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \mathbf{X}\,\varphi_1 \mid \varphi_1\,\mathbf{U}\,\varphi_2$$

where p *is a proposition.*

Formulas *true* and *false* represent a tautology and a contradiction respectively. They do not increase the expressive power of the logic and are included only for technical convenience in proofs. In order to reduce parentheses, unary operators are assigned a higher priority than binary operators. For example $\neg\varphi_1 \mathbf{U} (\mathbf{X}\varphi_1 \vee \varphi_2)$ is the same as $(\neg\varphi_1) \mathbf{U} ((\mathbf{X}\varphi_1) \vee \varphi_2)$. Finally, we define operators $\varphi_1 \mathbf{R} \varphi_2$, $\mathbf{F}\ \varphi$ and $\mathbf{G}\ \varphi$ as a shorthand for $\neg(\neg\varphi_1 \mathbf{U} \neg\varphi_2)$, *true* $\mathbf{U}\ \varphi$ and *false* $\mathbf{R}\ \varphi$ respectively.

Next, we estabilish a connection between formulas of LTL and sets of observation sequences. To this aim, we define the *semantics* of the logic. The semantics describes if an observation sequence w *satifies* a formula φ, a fact denoted as $w \models \varphi$. A formula φ is then identified with a set $\{w \mid w \models \varphi\}$. Note that despite being necessary we explicitly state semantics of operators $\wedge$ and $\mathbf{R}$. The reason for this is that we use (the semantics of) these operators in proofs.

Definition 2.7 *Let $w = (a_0, a_1, \ldots)$ is an infinite observation sequence. Then the semantics of LTL is*

$$
\begin{array}{lcl}
w \models \mathit{true} & & \mathit{always} \\
w \models \mathit{false} & & \mathit{never} \\
w \models p & \Leftrightarrow & p \in a_0 \\
w \models \neg\varphi_1 & \Leftrightarrow & w \not\models \varphi_1 \\
w \models \varphi_1 \wedge \varphi_2 & \Leftrightarrow & w \models \varphi_1 \textit{ and } w \models \varphi_2 \\
w \models \varphi_1 \vee \varphi_2 & \Leftrightarrow & w \models \varphi_1 \textit{ or } w \models \varphi_2 \\
w \models \mathbf{X}\ \varphi_1 & \Leftrightarrow & w^1 \models \varphi_1 \\
w \models \varphi_1 \mathbf{U} \varphi_2 & \Leftrightarrow & \exists(n \in \mathbb{N}_0)(w^n \models \varphi_2 \textit{ and } \forall(m < n)(w^m \models \varphi_1)) \\
w \models \varphi_1 \mathbf{R} \varphi_2 & \Leftrightarrow & \forall(n \in \mathbb{N}_0)(w^n \models \varphi_1 \textit{ or } \exists(m < n)(w^m \models \varphi_2))
\end{array}
$$

Note that the definition of semantics assumes an observation sequence to be infinite. Otherwise, the definition of $\models$ for all temporal operators would be incorrect. For a finite observation the definition of $\models$ is incorrect for all temporal operators and we alter it as follows.

Definition 2.8 *Let $w = (a_0, a_1, \ldots, a_n)$ is a finite observation sequence. Then the semantics of temporal operators is*

$$
\begin{array}{lcl}
w \models \mathbf{X}\ \varphi_1 & \Leftrightarrow & \begin{cases} \mathit{false} & |w| = 1 \\ w^1 \models \varphi_1 & \mathit{otherwise} \end{cases} \\
w \models \varphi_1 \mathbf{U} \varphi_2 & \Leftrightarrow & \exists(0 \leq n \leq |w|)(w^n \models \varphi_1 \textit{ and } \forall(m < n)(w^m \models \varphi_2)) \\
w \models \varphi_1 \mathbf{R} \varphi_2 & \Leftrightarrow & \forall(0 \leq n \leq |w|)(w^n \models \varphi_1 \textit{ or } \exists(m < n)(w^m \models \varphi_2))
\end{array}
$$

Example Here we give examples of some LTL expressible specification requirements for a CD-RW mechanic. First, the mechanic should read or write only when it is closed. This requirement can be expressed as $\mathbf{G}((r \vee w) \Rightarrow \neg o)$. Further the mechanic should never write when there is no medium inside, $\neg\mathbf{F}(w \wedge \neg m)$. Finally, a medium should never be removed when it is read from, $\mathbf{G}((m \wedge r) \Rightarrow (m\ \mathbf{U}\ \neg r))$.

The second logic considered is a fragment of *Metric Interval Temporal Logic* (MITL) [1]. It is denoted MITL$_{\leq}$ and to the best of our knowledge it was identified in [17]. Neither MITL nor MITL$_{\leq}$ has been used as widely as LTL. This is likely implied by a higher practical complexity of verification algorithms for MITL and MITL$_{\leq}$. However, thanks to the technological advance, tools for the verification of real-time systems are becoming more and more popular [6].

We have chosen MITL$_{\leq}$ out of a variety of real-time temporal logics as it allows to express basic timing constraints and yet it has the same asymptotical complexity of the model checking as LTL. Similarly to LTL, we use MITL$_{\leq}$ to express a (possibly infinite) set of timed observation sequences and thus specify the desired behaviour of a system.

Definition 2.9 *The syntax of MITL$_{\leq}$ is defined as follows*

$$\varphi ::= \mathit{true} \mid \mathit{false} \mid p \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1\ \mathbf{U}_{\leq d}\ \varphi_2$$

where $d \in \mathbb{R}_0^+ \cup \{\infty\}$, and p is a proposition.

Again, formulas *true* and *false* represent a tautology and a contradiction respectively and unary operators are assigned a higher priority then binary operators. Further we adopt rules similar to those for LTL. In particular, we define operators $\varphi_1\ \mathbf{R}_{\leq d}\ \varphi_2$, $\mathbf{F}_{\leq d}\ \varphi$ and $\mathbf{G}_{\leq d}\ \varphi$ as a shorthand for formulas $\neg(\neg\varphi_1\ \mathbf{U}_{\leq d}\ \neg\varphi_2)$, $\mathit{true}\ \mathbf{U}_{\leq d}\ \varphi$ and $\mathit{false}\ \mathbf{R}_{\leq d}\ \varphi$ respectively. Finally, in the case of an unbounded temporal operator such as $\varphi_1\ \mathbf{U}_{\leq \infty}\ \varphi_2$ we leave out the subscript and write $\varphi_1\ \mathbf{U}\ \varphi_2$ instead.

We conclude this section by estabilishing a connection between formulas of MITL$_{\leq}$ and sets of timed observation sequences. This is done in a similar fashion to LTL—through the *semantics* of MITL$_{\leq}$. The semantics describes if a timed observation sequence ρ *satifies* a formula φ, a fact denoted as $\rho \models \varphi$. A formula φ is then identified with a set $\{\rho \mid \rho \models \varphi\}$. Note that despite being necessary we explicitly state the semantics of operators $\wedge$ and $\mathbf{R}_{\leq d}$. The reason for this is that we use (the semantics of) these operators in proofs.

Definition 2.10 *Let $\rho = ((a_0, a_1, \ldots), (I_0, I_1, \ldots))$ is a timed observation sequence. Then the semantics of the MITL$_{\leq}$ logic is*

$$
\begin{array}{lcl}
\rho \models \mathit{true} & & \mathit{always} \\
\rho \models \mathit{false} & & \mathit{never} \\
\rho \models p & \Leftrightarrow & p \in a_0 \\
\rho \models \neg\varphi_1 & \Leftrightarrow & \rho \not\models \varphi_1 \\
\rho \models \varphi_1 \wedge \varphi_2 & \Leftrightarrow & \rho \models \varphi_1 \textit{ and } \rho \models \varphi_2 \\
\rho \models \varphi_1 \vee \varphi_2 & \Leftrightarrow & \rho \models \varphi_1 \textit{ or } \rho \models \varphi_2 \\
\rho \models \varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2 & \Leftrightarrow & \exists(0 \leq t \leq d)(\rho^t \models \varphi_1 \textit{ and } \forall(0 \leq t' < t)(\rho^{t'} \models \varphi_2)) \\
\rho \models \varphi_1 \, \mathbf{R}_{\leq d} \, \varphi_2 & \Leftrightarrow & \forall(0 \leq t \leq d)(\rho^t \models \varphi_1 \textit{ or } \exists(0 \leq t' < t)(\rho^{t'} \models \varphi_2))
\end{array}
$$

Note that the Definition 2.10, unlike the Definition 2.7, does not require ρ to be infinite. This is thanks to the underlying interval sequence, which despite being potentially finite represents an infinite time axis.

Example Here we give examples of some MITL$_{\leq}$ expressible specification requirements for a CD-RW mechanic. First, the mechanic should never continuosly read for 5 time units. This requirement can be expressed as $\neg\mathbf{F}(\mathbf{G}_{\leq 5}\ r)$. Further a medium should not be removed during a write operation and the writing should not last for more than 10 time units, $\mathbf{G}((m \wedge w) \Rightarrow (m \, \mathbf{U}_{\leq 10} \, w))$. Finally, the mechanic never stays opened for more than 20 time units $\mathbf{G}(o \Rightarrow \mathbf{F}_{\leq 20}(\neg o))$.

Chapter 3

Monitoring of LTL

"You see, Watson, but you do not observe."
- Sherlock Holmes

We start this chapter with a discussion of monitoring paradigms in the context of LTL. Next, we present a method for monitoring of an LTL specification. Lastly, we extend this method to enable monitoring of a system in the presence of *partial observability*, that is a system with some parts—and thus some propositions from its specification—being unobservable.

3.1 Online vs. offline monitoring

Let us recall that, in the field of monitoring there are two main approaches. *Online monitoring* is a process of watching an activity of a system as it progresses whereas *offline monitoring* is a process of watching an activity of a system after it has occured. In this chapter, we focus on monitoring of a system for the purpose of the formal verification of its LTL specification.

First, let us consider the offline monitoring. Assuming that the behaviour of a system is represented as a finite observation sequence, we can verify the behaviour against an LTL specification in several straightforward ways. One possibility is to exploit an algorithm based on the dynamic programming algorithm for *Computational Tree Logic* (CTL) developed in [11], as CTL and LTL has equal expressivity over paths. This algorithm evaluate all subformulas of the formula syntax tree in a bottom-up manner, that is from leaves to the root, for every state of the observation sequence. Another possibility is to transform the observation sequence into an equivalent Kripke structure [20] and apply any of the variety of LTL model-checking algorithms [5, 8, 9, 29].

Even though these approaches are applicable for offline monitoring, they do not carry over to online monitoring. The above mentioned approaches require an observation sequence to be finite, yet one of basic fea-

tures of online monitoring is that it works with possibly infinite observation sequences. Two natural questions arise. How to deal with infinite observation sequences in finite time and for what purpose?

From our point of view, the benefit of online monitoring should be in detecting violations of LTL specification as soon as they take place. However, not all LTL formulas can be verified using only a finite prefix of an infinite observation sequence. For instance, let us consider a very simple formula $\varphi \equiv \mathbf{GF}\mathit{ready}$ expressing that the state ready is encountered infinitely many times. Clearly, any finite observation sequence can be extended to an infinite observation sequence which either satisfies or violates φ.

Therefore, it is reasonable to use online monitoring only for LTL formulas that can be either satisfied or violated using a finite prefix of an infinite observation sequence. Such formulas fall into a class of properties known as *safety*. This class has been identified by Lamport in [23] and its verification was studied for example by Kupferman and Vardi in [21].

To handle a possibly infinite observation sequence we design an algorithm that works on-the-fly. It processes observation sequence in one pass and thus does need to not store it. The first on-the-fly algorithm for LTL was described in [18] and since then many improvements have been proposed. The original algorithm creates an automata that accepts precisely those observation sequences that do not satisfy the specification. Even though our algorithm is based on the same idea, there are numerous distictions—in use, in complexity, and in presentation.

3.2 Methods

Essence of the algorithm could be phrased as *formula rewriting*. Given a state of an observation sequence and a specification expressed as an LTL formula, the algorithm determines which propositions should hold now and computes an LTL formula that should hold in the next state in order for the original LTL formula to hold in the current state.

Definition 3.1 *An LTL formula φ is in the* positive normal form *(PNF) if negations in φ occur only over propositons. Furthermore, φ is in the* disjunctive *positive normal form (DPNF) if $\varphi \equiv \alpha_1 \vee \cdots \vee \alpha_n$ where each* clause *α_i is of the form $\varphi_1^i \wedge \cdots \wedge \varphi_{m_i}^i$ and each* literal *φ_j^i is in PNF and of the form $\mathit{true}, \mathit{false}, p, \neg p, \mathbf{X}\psi, \psi_1 \mathbf{U} \psi_2$, or $\psi_1 \mathbf{R} \psi_2$.*

A formula φ in DPNF can be alternatively represented as a set of clauses $\{\alpha_1, \ldots, \alpha_n\}$ or as a set of sets of literals $\{\{\varphi_1^1, \ldots, \varphi_{m_1}^1\}, \ldots, \{\varphi_1^n, \ldots, \varphi_{m_n}^n\}\}$.

This notation is justified in the proof of the Lemma 3.8 and is used in the rest of this chapter where we often identify formulas in DPNF with a set of clauses.

Proposition 3.2 *Any LTL formula can be transformed into an equivalent formula in PNF using the following identities:*

- $\neg(\neg\varphi) \equiv \varphi$
- $\neg \mathit{false} \equiv \mathit{true}$
- $\neg \mathit{true} \equiv \mathit{false}$
- $\neg(\varphi_1 \wedge \varphi_2) \equiv \neg\varphi_1 \vee \neg\varphi_2$
- $\neg(\varphi_1 \vee \varphi_2) \equiv \neg\varphi_1 \wedge \neg\varphi_2$
- $\neg\mathbf{X}\varphi \equiv \mathbf{X}\neg\varphi$
- $\neg(\varphi_1 \mathbf{U} \varphi_2) \equiv \neg\varphi_1 \mathbf{R} \neg\varphi_2$
- $\neg(\varphi_1 \mathbf{R} \varphi_2) \equiv \neg\varphi_1 \mathbf{U} \neg\varphi_2$

Proposition 3.3 *Any LTL formula φ in PNF can be transformed into an equivalent DPNF formula $norm(\varphi)$ as follows:*

- $norm(\varphi_1 \wedge \varphi_2) = \{\psi_1 \wedge \psi_2 \mid \psi_1 \in norm(\varphi_1), \psi_2 \in norm(\varphi_2)\}$
- $norm(\varphi_1 \vee \varphi_2) = norm(\varphi_1) \vee norm(\varphi_2)$
- *otherwise* $norm(\varphi) = \varphi$

Example For instance, the formula $\neg\mathbf{F}(w \wedge \neg m)$ expressing that it can never happen that mechanic writes and there is no medium inside is after a transformation to PNF equal to formula $\mathbf{G}(\neg w \vee m)$ expressing the same in other words. Slightly more complex formula such as $\mathbf{G}(\neg w \vee m) \wedge (\mathbf{G}(r \Rightarrow \neg o) \vee \mathbf{F}\, o)$ is after a transformation to DPNF equal to the formula $(\mathbf{G}(\neg w \vee m) \wedge \mathbf{G}(r \Rightarrow \neg o)) \vee (\mathbf{G}(\neg w \vee m) \wedge \mathbf{F}\, o)$.

In the remainder of this chapter we assume that all formulas are in DPNF, unless stated otherwise.

Definition 3.4 *For an observation sequence $w = (a_0, a_1, \ldots)$ and a formula φ we define function $succ(\varphi, w)$ inductively on the structure of φ as follows:*

- $succ(\mathit{true}, w) = \mathit{true}$

- $succ(false, w) = false$
- $succ(p, w) = \begin{cases} true & p \in a_0 \\ false & p \notin a_0 \end{cases}$
- $succ(\neg p, w) = \begin{cases} true & p \notin a_0 \\ false & p \in a_0 \end{cases}$
- $succ(\varphi_1 \wedge \varphi_2, w) = \{\psi_1 \wedge \psi_2 \mid \psi_1 \in succ(\varphi_1, w), \psi_2 \in succ(\varphi_2, w)\}$
- $succ(\varphi_1 \vee \varphi_2, w) = succ(\varphi_1, w) \vee succ(\varphi_2, w)$
- $succ(\mathbf{X}\varphi, w) = \varphi$
- $succ(\varphi_1 \,\mathbf{U}\, \varphi_2, w) = succ(\varphi_2, w) \vee \{\psi \wedge (\varphi_1 \,\mathbf{U}\, \varphi_2) \mid \psi \in succ(\varphi_1, w)\}$
- $succ(\varphi_1 \,\mathbf{R}\, \varphi_2, w) = \{\psi_1 \wedge \psi_2 \mid \psi_1 \in (succ(\varphi_1, w) \cup \{\varphi_1 \,\mathbf{R}\, \varphi_2\}), \psi_2 \in succ(\varphi_2, w)\}$

Example Now we demonstrate the applicability of the function $succ$ on the formula $\mathbf{G}((m \wedge r) \Rightarrow (m \,\mathbf{U}\, \neg r))$ and the observation sequence $w = (\{o, m\}, \{m\}, \{m, r\}, \{m\}, \ldots)$. To keep the intermediate formulas as simple as possible we use reduction rules from the Lemma 3.8.

1. $succ(\mathbf{G}((m \wedge r) \Rightarrow (m \,\mathbf{U}\, \neg r)), w) = \mathbf{G}((m \wedge r) \Rightarrow (m \,\mathbf{U}\, \neg r)$
2. $succ(\mathbf{G}((m \wedge r) \Rightarrow (m \,\mathbf{U}\, \neg r)), w^1) = \mathbf{G}((m \wedge r) \Rightarrow (m \,\mathbf{U}\, \neg r)$
3. $succ(\mathbf{G}((m \wedge r) \Rightarrow (m \,\mathbf{U}\, \neg r)), w^2) = \mathbf{G}((m \wedge r) \Rightarrow (m \,\mathbf{U}\, \neg r)) \wedge (m \,\mathbf{U}\, \neg r)$
4. $succ(\mathbf{G}((m \wedge r) \Rightarrow (m \,\mathbf{U}\, \neg r)) \wedge (m \,\mathbf{U}\, \neg r), w^3) = \mathbf{G}((m \wedge r) \Rightarrow (m \,\mathbf{U}\, \neg r)$

Lemma 3.5 *The function $succ(\varphi, w)$ can be computed using polynomial space with respect to the number of distinct subformulas in φ.*

Proof Directly from the definition of $succ(\varphi, w)$.

■

Lemma 3.6 *Let φ is LTL formula and $w = (a_0, a_1, \ldots)$ is an infinite observation sequence. Then $w \models \varphi \Leftrightarrow \exists(\psi \in succ(\varphi, w))(w^1 \models \psi)$.*

Proof By induction to the structure of formula φ.

- $\varphi \equiv true$

  "$\Rightarrow$": As $succ(true, w) = true$ and $w \models true$ for any observation sequence w, $\exists(\psi \in succ(true, w))(w^1 \models true)$.

  "$\Leftarrow$": By contraposition. Because $w \not\models true$ is not satisfied for any observation sequence w, $w \not\models true \Rightarrow \forall(\psi \in succ(true, w))(w^1 \not\models \psi)$.

- $\varphi \equiv \mathit{false}$

  "$\Rightarrow$": Because $w \models \mathit{false}$ is not satisfied for any observation sequence w, $w \models \mathit{false} \Rightarrow \exists(\psi \in \mathit{succ}(\mathit{false}, w))(w^1 \models \psi)$.

  "$\Leftarrow$": By contraposition. As $w \not\models \mathit{false}$ holds for any observation sequence w, $\forall(\psi \in \mathit{succ}(\mathit{false}, w))(w^1 \not\models \psi)$.

- $\varphi \equiv p$

  "$\Rightarrow$": Because $w \models p \Rightarrow p \in a_0 \Rightarrow \mathit{succ}(p, w) = \mathit{true}$ and $w \models \mathit{true}$ holds for any observation sequence w, $\exists(\psi \in \mathit{succ}(p, w))(w^1 \models \mathit{true})$.

  "$\Leftarrow$": By contraposition. As $w \not\models \mathit{false}$ holds for any observation sequence w, $\forall(\psi \in \mathit{succ}(p, w))(w^1 \not\models \mathit{false})$.

- $\varphi \equiv \neg p$

  Proof of this case is an analogy to the previous one.

- $\varphi \equiv \varphi_1 \wedge \varphi_2$

  "$\Rightarrow$": Clearly, $w \models \varphi_1 \wedge \varphi_2 \Rightarrow w \models \varphi_1$ and $w \models \varphi_2$. By induction hypothesis, $\exists(\psi_1 \in \mathit{succ}(\varphi_1, w))(w^1 \models \psi_1)$ and $\exists(\psi_2 \in \mathit{succ}(\varphi_2, w))(w^1 \models \psi_2)$. Thus exists $\psi_1 \wedge \psi_2$ from $\mathit{succ}(\varphi_1 \wedge \varphi_2, w)$ such that $w^1 \models \psi_1 \wedge \psi_2$. In other words, $\exists(\psi \in \mathit{succ}(\varphi_1 \wedge \varphi_2, w))(w^1 \models \psi)$.

  "$\Leftarrow$": By contraposition. $w \not\models \varphi_1 \wedge \varphi_2$ implies $w \not\models \varphi_1$ or $w \not\models \varphi_2$. Without loss of generality let $w \not\models \varphi_1$. By induction hypothesis $\forall(\psi_1 \in \mathit{succ}(\varphi_1, w))(w^1 \not\models \psi_1)$. Further let $\psi_2 \in \mathit{succ}(\varphi_2, w)$. Clearly, $\psi_1 \wedge \psi_2 \in \mathit{succ}(\varphi_1 \wedge \varphi_2, w)$ and $w^1 \not\models \psi_1 \wedge \psi_2$ for any $\psi_1 \in \mathit{succ}(\varphi_1, w)$. Therefore $\forall(\psi \in \mathit{succ}(\varphi_1 \wedge \varphi_2, w))(w^1 \not\models \psi)$.

- $\varphi \equiv \varphi_1 \vee \varphi_2$

  "$\Rightarrow$": Clearly, $w \models \varphi_1 \vee \varphi_2$ implies $w \models \varphi_1$ or $w \models \varphi_1$. Without loss of generality let $w \models \varphi_1$. Thus, by induction hypothesis, $\exists(\psi_1 \in \mathit{succ}(\varphi_1, w))(w^1 \models \psi_1)$. Further $\psi_1 \in \mathit{succ}(\varphi_1 \vee \varphi_2, w)$ and therefore $\exists(\psi \in \mathit{succ}(\varphi_1 \vee \varphi_2, w))(w^1 \models \psi)$.

  "$\Leftarrow$": By contraposition. $w \not\models \varphi_1 \vee \varphi_2$ implies $w \not\models \varphi_1$ and $w \not\models \varphi_2$. If $\psi \in \mathit{succ}(\varphi_1 \vee \varphi_2, w)$, then either $\psi \in \mathit{succ}(\varphi_1, w)$ or $\psi \in \mathit{succ}(\varphi_2, w)$ and by induction hypothesis $w^1 \not\models \psi$. Therefore $\forall(\psi \in \mathit{succ}(\varphi_1 \vee \varphi_2, w))(w^1 \not\models \psi)$.

- $\varphi \equiv \mathbf{X}\varphi_1$

  Directly from the semantics of LTL.

- $\varphi \equiv \varphi_1 \mathbf{U} \varphi_2$

  "$\Rightarrow$": $w \models \varphi_1 \mathbf{U} \varphi_2$ implies that $\exists(n \in \mathbb{N}_0)(w^n \models \varphi_2$ and $\forall(m < n)(w^m \models \varphi_1))$. If $w \models \varphi_2$ then, by induction hypothesis, $\exists(\psi_2 \in succ(\varphi_2, w))(w^1 \models \psi_2)$ and as $succ(\varphi_2, w) \subseteq succ(\varphi_1 \mathbf{U} \varphi_2, w)$, $\exists(\psi \in succ(\varphi_1 \mathbf{U} \varphi_2, w))(w^1 \models \psi)$.

  Otherwise $w \not\models \varphi_2$ and as $w \models \varphi_1 \mathbf{U} \varphi_2$, $w \models \varphi_1$ and $w^1 \models \varphi_1 \mathbf{U} \varphi_2$ from the semantics of LTL. Further, by induction hypothesis, $\exists(\psi_1 \in succ(\varphi_1, w))(w^1 \models \psi_1)$. Finally, as $\psi_1 \wedge \varphi_1 \mathbf{U} \varphi_2 \in succ(\varphi_1 \mathbf{U} \varphi_2, w)$, $\exists(\psi \in succ(\varphi_1 \mathbf{U} \varphi_2, w))(w^1 \models \psi)$.

  "$\Leftarrow$": In this case, $\exists(\psi \in succ(\varphi_1 \mathbf{U} \varphi_2, w))(w^1 \models \psi)$ implies that $\exists(\psi \in succ(\varphi_2, w))(w^1 \models \psi)$ or $\exists(\psi \in succ(\varphi_1, w)(w^1 \models \psi)$ and $w^1 \models \varphi_1 \mathbf{U} \varphi_2$.

  First, let $\exists(\psi \in succ(\varphi_2, w))(w^1 \models \psi)$. Then $w \models \varphi_2$ using induction hypothesis and thus $w \models \varphi_1 \mathbf{U} \varphi_2$.

  Next, let $\exists(\psi \in succ(\varphi_1, w)(w^1 \models \psi)$ and $w^1 \models \varphi_1 \mathbf{U} \varphi_2$. Then $w \models \varphi_1$ using induction hypothesis and $w \models \varphi_1 \mathbf{U} \varphi_2$ using the semantics of LTL.

- $\varphi \equiv \varphi_1 \mathbf{R} \varphi_2$

  "$\Rightarrow$": $w \models \varphi_1 \mathbf{R} \varphi_2$ implies $\forall(n \in \mathbb{N}_0)(w^n \models \varphi_1$ or $\exists(m < n)(w^m \models \varphi_2))$. Therefore $w \models \varphi_2$. Further $\exists(\psi_2 \in succ(\varphi_2, w))(w^1 \models \psi_2)$ using induction hypothesis.

  First, let $w \models \varphi_1$. Then $\exists(\psi_1 \in succ(\varphi_1, w))(w^1 \models \psi_1)$ using induction hypotehsis and consequently $\exists(\psi \in succ(\varphi_1 \mathbf{R} \varphi_2, w))(w^1 \models \psi)$.

  Otherwise, $w \not\models \varphi_1$ and in this case $w \models \varphi_1 \mathbf{R} \varphi_2$ implies $w^1 \models \varphi_1 \mathbf{R} \varphi_2$. Clearly, $(\varphi_1 \mathbf{R} \varphi_2) \wedge \psi_2 \in succ(\varphi_1 \mathbf{R} \varphi_2, w)$ and thus $\exists(\psi \in succ(\varphi_1 \mathbf{R} \varphi_2, w))(w^1 \models \psi)$.

  "$\Leftarrow$": Assumption $\exists(\psi \in succ(\varphi_1 \mathbf{R} \varphi_2, w))(w^1 \models \psi)$ implies that $\exists(\psi_2 \in succ(\varphi_2, w))(w^1 \models \psi_2)$ and $\exists(\psi_1 \in succ(\varphi_1, w))(w^1 \models \psi_1)$ or $w^1 \models \varphi_1 \mathbf{R} \varphi_2$.

  First, let $\exists(\psi_1 \in succ(\varphi_1, w))(w^1 \models \psi_1)$. Then, by induction hypothesis, $w \models \varphi_1$, $w \models \varphi_2$ and consequently $\rho \models \varphi_1 \mathbf{R} \varphi_2$.

  Otherwise, $\forall(\psi_1 \in succ(\varphi_1, w)(w^1 \not\models \psi_1)$ and $w^1 \models \varphi_1 \mathbf{R} \varphi_2$. By induction hypothesis $w \not\models \varphi_1$ and $w \models \varphi_2$. Clearly, $w \models \varphi_1 \mathbf{R} \varphi_2$ using the semantics of LTL.

■

Thus given an LTL formula φ we can process a possibly infinite observation sequence w, a state by state, and resolve whether $w \models \varphi$ by iteratively computing values of the function $succ$. An advantage of this approach is that it does not need the whole observation sequence w at the same time. Nevertheless, the approach would be of little use if the formula would continuosly grow. Fortunately, we show that it is not the case. To this aim, we measure the size of a formula using characteristics of its DPNF.

Definition 3.7 *Let φ be an LTL formula. Then the* size *of φ, denoted as $|\varphi|$, is the sum of the number of all literals in each clause. Further we say that a subformula of LTL formula φ is* basic *if it is of the form $p, \neg p, \mathbf{X}\varphi_1, \varphi_1 \mathbf{U} \varphi_2$ or $\varphi_1 \mathbf{R} \varphi_2$.*

Lemma 3.8 *Let φ be an LTL formula and w be an infinite observation sequence. Then $|succ^i(\varphi, w)| \in \mathcal{O}(2^{|\varphi|})$ for every $i \in \mathbb{N}_0$.*

Proof We derive the upper bound on the size of $|succ^i(\varphi, w)|$ from the properties of DPNF. A formula in DPNF can be reduced using the following rules. Note that, the last rule justifies alternative representations of a formula in DPNF.

1. $\mathit{false} \wedge \varphi^i_j \equiv \mathit{false}$ and $\mathit{false} \vee \varphi^i_j \equiv \varphi^i_j$
2. $\mathit{true} \wedge \varphi^i_j \equiv \varphi^i_j$ and $\mathit{true} \vee \varphi^i_j \equiv \mathit{true}$
3. if exist distinct i, j such that $\alpha_i \subseteq \alpha_j$ then $\varphi \equiv \varphi \setminus \{\alpha_j\}$
4. duplicate clauses and duplicate literals in a clause can be removed

Let B be the set of all basic subformulas of φ and n be the cardinality of B. First, we prove that the size of a formula φ is at most exponential in n. This is indeed truth as each clause, except for special clauses true and false, is a subset of B and the number of literals in each clause is at most equal to n. Therefore $|\varphi| \in \mathcal{O}(2^n)$.

Next, from definition of function $succ$ it follows that the set of all basic subformulas of $succ(\varphi, w)$ is a subset B. Therefore $|\varphi| \in \mathcal{O}(2^n)$ implies $|succ(\varphi, w)| \in \mathcal{O}(2^n)$. Applying the previous argument iteratively yeilds, $|succ^i(\varphi, w)| \in \mathcal{O}(2^n)$ for every $i \in \mathbb{N}$. Clearly, $n \leq |\varphi|$ which concludes the proof. The bound can be further refined using the last but one reduction rule. The maximum number of clauses in DPNF is in fact the maximum number of pairwise incomparable sets of basic subformulas. In particular, the maximum number of clauses is $\binom{n}{\lfloor n/2 \rfloor}$ (see [27]).

■

The function $succ$ can be utilized for an on-the-fly creation of a non-deterministic automata similar to that from [18]. We describe the connection using a generalization of Büchi automata [10] as defined in [30].

Definition 3.9 Generalized Büchi automata $A = \langle L, \Sigma, \delta, L_0, F \rangle$ *consists of*

- *a set* L *of* locations,
- *an* alphabet Σ,
- *a* transition function $\delta : L \times \Sigma \longrightarrow 2^L$,
- *a set* $L_0 \subseteq L$ *of* initial states,
- *and an* accepting condition $F \subseteq 2^L$.

Further A accepts *an infinite observation sequence* $w = (a_0, a_1, \ldots)$ *if there exists an infinite sequence* $l = (l_0, l_1, \ldots)$ *of locations such that* $l_0 \in L_0$, $l_{i+1} \in \delta(l_i, a_i)$ *for each* $i \in \mathbb{N}_0$, *and* $inf(l) \cap f_i$ *for each* $f_i \in F$, *where* $inf(l)$ *denotes a set of all locations occuring in* l *infinitely many times.*

Definition 3.10 *For an LTL formula* φ *we define a generalized Büchi automata* $A_\varphi = \langle L, \Sigma, \delta, L_0, F \rangle$ *where*

- L *is a set of all subsets of basic subformulas of* φ,
- Σ *is a set of all subsets of atomic propositions in* φ,
- δ *is such that* $\delta(\psi, a) = succ(\psi, a)$ *for each* $\psi \in L$, $a \in \Sigma$,
- $L_0 = \varphi$,
- *and* F *is a set that contains for each subformula* $\varphi_1 \mathbf{U} \varphi_2$ *of* φ *a set of all clauses containing* $\varphi_1 \mathbf{U} \varphi_2$.

Proposition 3.11 *Let* φ *is an LTL formula and* w *is an observation sequence. Then* $w \models \varphi \Leftrightarrow A_\varphi$ *accepts* w.

Last but not least, we argue that our algorithm based on the formula rewriting approach is suitable for offline monitoring as well. Its advantage is the ability to terminate after processing only a part of an observation sequence. To cope with finite observation sequences, the function $succ$ has to be adjusted according to the respective semantics of LTL. Despite this technicality, the algorithm carries over smoothly.

3.3 Partial observability

Now we focus on monitoring of a system whose states are not fully observable. For example consider a system with *black box* modules. We are not allowed to see inside these blacks boxes and thus some propositions about the system might not be resolved.

Example Imagine that we remove diodes from the CD-RW mechanic we have mentioned earlier. Thus we are not able to tell if a read or write operation is taking place and yet we would like to somehow verify formula $\neg\mathbf{F}(w \wedge \neg m)$. Intuitively, this formula can be violated any time there is no medium inside the mechanic, as a write operation can possibly take place. On the other hand, it does not have to be violated if a write operation does not actually take place.

We show that partial observability can be handled using the function $succ$. We pay a particular attention to resolving whether an observation sequence can satisfy or violate its specification. We present a method for resolving the satisfiability and sketch how it can be modified to resolve the validity and other problems.

The idea is quite straightforward. First, we assume that a change of state can take place though no change can be actually observed. Second, when a change of state takes place, we consider all possible valuations of unresolved propositions. After computing all possible results of the function $succ$ we combine them using boolean connectives. The way we combine them is determined by the question we want to answer. For the satisfiability and the validity it is natural to use disjunctions and conjunctions respectively.

In the remainder of this section, let φ is an LTL formula, $w = (a_0, a_1, \ldots)$ is an infinite observation sequence resulting from monitoring of a partially observable system—thus possibly missing some propositions—and U a set of unobservable propositions.

Definition 3.12 *We define function* $enum(\varphi, w, U)$ *as*

$$enum(\varphi, w, U) = \bigvee_{u \in U} succ(\varphi, (a_0 \cup u, a_1, a_2, \ldots)).$$

Next, we generalize the function $succ$ to allow for changes of a state that take place though no change is actually observed. We assume that the number of such intermediate changes is arbitrary large but finite.

Definition 3.13 *We define function* $\widehat{succ}(\varphi, w, U)$ *as*

$$\widehat{succ}(\varphi, w, U) = \bigvee_{i \in \mathbb{N}_0} change^i(\varphi, w, U)$$

where formula $change^i(\varphi, w, U)$ *is inductively defined as*

- $change^0(\varphi, w, U) = enum(\varphi, w, U)$ *and*
- $change^{i+1}(\varphi, w, U) = change^i(enum(\varphi, w, U), w, U)$ *for every* $i \in \mathbb{N}_0$.

Informally, the function $\widehat{succ}$ guesses the number of intermediate changes, that is the number of recursions of the function $change$, and for each change guesses the valuation of unresolved propositions using the function $enum$.

Next, we extend semantics of LTL to allow for partial observability and we prove properties of function $\widehat{succ}$ that provide for its practical use.

Definition 3.14 $w \models_U \varphi$ *if there exists* $i_0, i_1, \ldots \in \mathbb{N}_0$ *such that there exists subsets* $u_0^0, u_1^0, \ldots, u_{i_0}^0, u_0^1, u_1^1, \ldots, u_{i_1}^1, \ldots$ *of* U *such that*

$$((a_0 \cup u_0^0), \ldots, (a_0 \cup u_{i_0}^0), (a_1 \cup u_0^1), \ldots, (a_1 \cup u_{i_1}^1), \ldots) \models \varphi$$

Lemma 3.15 $w \models_U \varphi \Leftrightarrow \exists(\psi \in \widehat{succ}(\varphi, w, U))(w^1 \models_U \psi)$.

Proof

"$\Rightarrow$": If $w \models_U \varphi$ then by the definition of $\models_U$ exists $i_0, i_1, \ldots \in \mathbb{N}_0$ such that there exists subsets $u_0^0, u_1^0, \ldots, u_{i_0}^0, u_0^1, u_1^1, \ldots, u_{i_1}^1, \ldots$ of U such that

$$((a_0 \cup u_0^0), \ldots, (a_0 \cup u_{i_0}^0), (a_1 \cup u_0^1), \ldots, (a_1 \cup u_{i_1}^1), \ldots) \models \varphi. \tag{3.1}$$

Let $w_u = ((a_0 \cup u_0^0), \ldots, (a_0 \cup u_{i_0}^0), (a_1 \cup u_0^1), \ldots, (a_1 \cup u_{i_1}^1), \ldots)$ and $k = i_0 + 1$. An iterative application of the Lemma 3.6 on 3.1 yields 3.2 and consequently 3.3 using the semantics $\models_U$. Further 3.4 using the definition of $enum$ and finally 3.5 using the definition of $\widehat{succ}$.

$$\exists(\psi_1 \in succ(\varphi, w_u))(\ldots \exists(\psi_k \in succ(\psi_{k-1}, w_u^{k-1}))(w_u^k \models \psi_k)\ldots) \tag{3.2}$$

$$\exists(\psi_1 \in succ(\varphi, w_u))(\ldots \exists(\psi_k \in succ(\psi_{k-1}, w_u^{k-1}))(w^1 \models_U \psi_k)\ldots) \tag{3.3}$$

$$\exists(\psi_1 \in enum(\varphi, w, U))(\ldots \exists(\psi_k \in enum(\psi_{k-1}, w, U))(w^1 \models_U \psi_k)\ldots) \tag{3.4}$$

$$\exists(\psi \in \widehat{succ}(\varphi, w, U))(w^1 \models_U \psi) \tag{3.5}$$

"$\Leftarrow$": If $\exists(\psi \in \widehat{succ}(\varphi, w, U))(w^1 \models_U \psi)$ then by the definition of $\widehat{succ}$

$$\exists(i \in \mathbb{N}_0)(\exists(\psi \in change^i(\varphi, w, U))(w^1 \models_U \psi)) \tag{3.6}$$

$$\Leftrightarrow$$

$$\exists(i \in \mathbb{N}_0)(\exists(\psi_1 \in enum(\varphi, w, U))(\ldots \exists(\psi_i \in enum(\psi_{i-1}, w, U)) \text{ such that } (w^1 \models_U \psi_i)\ldots)) \tag{3.7}$$

Then the definition of $enum$ yields 3.8 and 3.9 follows from 3.8 using the semantics $\models_U$ where w_x and w_y are defined as $w_x = ((a_0 \cup u_0^0), \ldots, (a_0 \cup u_i^0), a_1, a_2, \ldots)$ and $w_y = ((a_1 \cup u_0^1), \ldots, (a_1 \cup u_{i_1}^1), (a_2 \cup u_0^2), \ldots, (a_2 \cup u_{i_2}^2), \ldots)$.

Finally, let $w_u = ((a_0 \cup u_0^0), \ldots, (a_0 \cup u_i^0), (a_1 \cup u_0^1), \ldots, (a_1 \cup u_{i_1}^1), \ldots)$. An iterative application of the Lemma 3.6 on 3.9 yeilds 3.10 which is the same as $w \models_U \varphi$.

$$\exists(i \in \mathbb{N}_0)(\exists(u_0^0, u_1^0, \ldots, u_i^0 \subseteq U) \quad (3.8)$$
$$\text{such that } (\exists(\psi_1 \in succ(\varphi, w_x, U))(\ldots \exists(\psi_i \in succ(\psi_{i-1}, w_x^{i-1}, U))$$
$$\text{such that } (w^1 \models_U \psi_i) \ldots)))$$
$$\exists(i, i_1, i_2, \ldots \in \mathbb{N}_0)(\exists(u_0^0, \ldots, u_i^0, u_0^1, \ldots, u_{i_1}^1, u_0^2, \ldots, u_{i_2}^2, \ldots \subseteq U) \quad (3.9)$$
$$\text{such that } (\exists(\psi_1 \in succ(\varphi, w_x, U))(\ldots \exists(\psi_i \in succ(\psi_{i-1}, w_x^{i-1}, U))$$
$$\text{such that } (w_y \models \psi_i) \ldots)))$$
$$\exists(i, i_1, i_2, \ldots \in \mathbb{N}_0)(\exists(u_0^0, \ldots, u_i^0, u_0^1, \ldots, u_{i_1}^1, u_0^2, \ldots, u_{i_2}^2, \ldots \subseteq U) \quad (3.10)$$
$$\text{such that } (w_u \models \varphi) \ldots)))$$

■

If the function $\widehat{succ}$ can be computed, then the Lemma 3.15 provides a method for verification of the satisfiability of LTL formulas in the presence of partial observability. We show that the function $\widehat{succ}$ can be computed with the same asymptotical complexity as the function $succ$.

Lemma 3.16 *Function $\widehat{succ}(\varphi, w, U)$ can be computed using polynomial space with respect to the size of φ.*

Proof First, we show that the function $enum(\varphi, w, U)$ can be computed using polynomial space with respect to the size of φ. As $enum(\varphi, w, U) = \bigvee_{u \in U} succ(\varphi, (a_0 \cup u, a_1, a_2, \ldots))$ the computation enumerates all possibilities one by one and to keep the size polynomial it transforms intermediate results into DPNF.

Next, we show that it is sufficient to consider only first $2^{|\varphi|}$ formulas $change^i(\varphi, w, U)$. This follows from the fact that the number of distinct subformulas in $change^i(\varphi, w, U)$ does not grow with i. Thus there is at most $2^{|\varphi|}$ syntactically distinct formulas $\bigvee_{i=0}^{k} change^i(\varphi, w, U)$ in DPNF. In other words, there exists $j < 2^{|\varphi|}$ such that $\bigvee_{i=0}^{j} change^i(\varphi, w, U) = \bigvee_{i=0}^{k} change^i(\varphi, w, U)$ for all $k > j$.

Finally, we show that the function $\widehat{succ}(\varphi, w, U)$ can be computed using polynomial space with respect to the size of φ. The computation iteratively computes formula $change^i(\varphi, w, U)$ for larger and larger i as long as

$\bigvee_{k=0}^{i} change^k(\varphi, w, U) \neq \bigvee_{k=0}^{i+1} change^k(\varphi, w, U)$. To this aim, it is sufficient to store only constant number of formulas which—after a transformation into DPNF—are of a polynomial size.

■

The way in which the function $\widehat{succ}$ works, provides answers to other questions as well. For instance, we state without a proof that if disjunctions in definitions of functions $enum$ and $\widehat{succ}$ are replaced with conjuctions then function $\widehat{succ}$ would verify the validity of LTL formulas in the presence of partial observability. Further techniques based on other combinations of boolean connectives are possible. However, an exploration of these techniques is out of the scope of this thesis.

Chapter 4

Monitoring of MITL$_{\leq}$

"Thus the whirligig of time brings in his revenges."
- William Shakespear

We start this chapter with a discussion of monitoring paradigms in the context of MITL$_{\leq}$. Next, we present methods for monitoring of an MITL$_{\leq}$ specification. Lastly, we describe how to deal with inaccurate measurement of time and *partial observability*, that is a situation where some parts of a system—and thus some propositions from its specification—are unobservable.

4.1 Online vs. offline monitoring

Let us recall that, in the field of monitoring there are two main approaches. *Online monitoring* is a process of watching an activity of a system as it progresses whereas *offline monitoring* is a process of watching an activity of a system after it has occured. In this chapter, we focus on monitoring of a system for the purpose of the formal verification of its MITL$_{\leq}$ specification.

First, we consider offline monitoring. Assuming that we have a finite timed observation sequence representing the behaviour of a system, we can resolve the validity of an MITL$_{\leq}$ specification using already known algorithms. In particular, we consider algorithms for resolving validity of a formula that use some kind of timed automata. The automata serves as a finite representation of a countable set of timed observation seqeunces.

One possibility is to use the algorithm for deciding MITL described by Alur, Feder, and Henzinger [1]. Although this algorithm is in EXPSPACE and deciding MITL$_{\leq}$ was shown to be PSPACE-complete, it is worth mentioning it as it provides good understanding of MITL and therefore MITL$_{\leq}$.

An asympotically optimal algorithm for deciding MITL$_{\leq}$ has been provided in Geilen's thesis [17]. Given an MITL$_{\leq}$ formula the algorithm builds an automata accepting precisely those timed observation sequences which violate the formula. This automata is intersected with an automata de-

scribing the behaviour being verified and the intersection is checked for the emptiness.

Yet another algorithm for deciding MITL$_{\leq}$ can be based on the formula rewriting approach and it is described in the following section. A similar algorithm has been already designed for *Metric Temporal Logic* (MTL) by Thati and Rosu [28]. The latter logic was defined by Alur and Henziger [2] and has, similarly to MITL, bounded temporal operators. However, it is interpreted over discrete time.

Next, we consider online monitoring of MITL$_{\leq}$. Similarly to LTL we see the benefit of online monitoring in detecting violations of an MITL$_{\leq}$ specification as soon as they take place. From this point of view, it is reasonable to monitor only those MITL$_{\leq}$ formulas which can be either satisfied or violated in a finite time. This allows for, among others, safety properites and formulas of the form $\varphi \equiv \mathbf{G}(p \Rightarrow \mathbf{F}_{\leq d}\, q)$ known as bounded response properties.

To handle possibly infinite timed observation sequences we need an algorithm that processes states and intervals of a timed observation sequence in one pass. To this aim, we can use an on-the-fly version of the algorithm for offline monitoring from [17] or alternatively we can employ, under a certain assumption, an algorithm based on the formula rewriting approach from the following section. We show that both approaches have its pros and cons.

4.2 Methods

First of all, we define the positive normal form (PNF) and the disjunctive positive normal form (DPNF) of MITL$_{\leq}$ formulas analogously to normal forms of LTL formulas. Next, we define fragments of MITL$_{\leq}$ as follows. For a set Φ of temporal operators, MITL$_{\leq}(\Phi)$ denotes a fragment of formulas which after transformation into PNF contain only temporal operators from Φ.

Definition 4.1 *An MITL$_{\leq}$ formula φ is in the* positive normal form *(PNF) if negations in φ occur only over propositons. Further, φ is in the* disjunctive *positive normal form (DPNF) if $\varphi \equiv \alpha_1 \vee \cdots \vee \alpha_n$ where each* clause *α_i is of the form $\varphi_1^i \wedge \cdots \wedge \varphi_{m_i}^i$ and each* literal *φ_j^i is in PNF and of the form $true, false, p, \neg p$, $\psi_1\, \mathbf{U}_{\leq d}\, \psi_2$, or $\psi_1\, \mathbf{R}_{\leq d}\, \psi_2$.*

A formula φ in DPNF can be alternatively represented as a set of clauses $\{\alpha_1, \ldots, \alpha_n\}$ or as a set of sets of literals $\{\{\varphi_1^1, \ldots, \varphi_{m_1}^1\}, \ldots, \{\varphi_1^n, \ldots, \varphi_{m_n}^n\}\}$.

This notation is justified in the proof of the Lemma 4.9 and is used in the rest of this chapter where we often identify formulas in DPNF with a set of clauses.

Proposition 4.2 *Any MITL$_{\leq}$ formula can be transformed into an equivalent formula in PNF using the following identities:*

- $\neg(\neg\varphi) \equiv \varphi$
- $\neg \mathit{false} \equiv \mathit{true}$
- $\neg \mathit{true} \equiv \mathit{false}$
- $\neg(\varphi_1 \wedge \varphi_2) \equiv \neg\varphi_1 \vee \neg\varphi_2$
- $\neg(\varphi_1 \vee \varphi_2) \equiv \neg\varphi_1 \wedge \neg\varphi_2$
- $\neg(\varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2) \equiv \neg\varphi_1 \, \mathbf{R}_{\leq d} \, \neg\varphi_2$
- $\neg(\varphi_1 \, \mathbf{R}_{\leq d} \, \varphi_2) \equiv \neg\varphi_1 \, \mathbf{U}_{\leq d} \, \neg\varphi_2$

Proposition 4.3 *Any MITL$_{\leq}$ formula φ in PNF can be transformed into an equivalent DPNF formula $norm(\varphi)$ as follows:*

- $norm(\varphi_1 \wedge \varphi_2) = \{\psi_1 \wedge \psi_2 \mid \psi_1 \in norm(\varphi_1), \psi_2 \in norm(\varphi_2)\}$
- $norm(\varphi_1 \vee \varphi_2) = norm(\varphi_1) \vee norm(\varphi_2)$
- *otherwise* $norm(\varphi) = \varphi$

Example For instance, the formula $\neg\mathbf{F}(\mathbf{G}_{\leq 5} \, r)$ expressing that it can never happen that a mechanic continuosly continuosly reads for 5 time units is after a transformation to PNF equal to the formula $\mathbf{G}(\mathbf{F}_{\leq 5}\neg r)$ expressing the same in other words. Slightly more complex formula such as $\mathbf{G}(\neg w \vee m) \wedge (\mathbf{G}_{\leq 5}(r \Rightarrow \neg o) \vee \mathbf{F} \, o)$ is after transformation to DPNF equal to the formula $(\mathbf{G}(\neg w \vee m) \wedge \mathbf{G}_{\leq 5}(r \Rightarrow \neg o)) \vee (\mathbf{G}(\neg w \vee m) \wedge \mathbf{F} \, o)$.

In the remainder of this chapter we assume that all formulas are in DPNF, unless stated otherwise.

Definition 4.4 *For a timed observation sequence $\rho = ((a_0, a_1, \ldots), (I_0, I_1, \ldots))$ and an MITL$_{\leq}$ formula φ, we define a set of formulas $succ(\varphi, \rho)$ inductively on the structure of φ as folows:*

- $succ(\mathit{true}, \rho) = \mathit{true}$

- $succ(\mathit{false}, \rho) = \mathit{false}$
- $succ(p, \rho) = \begin{cases} \mathit{true} & p \in a_0 \\ \mathit{false} & p \notin a_0 \end{cases}$
- $succ(\neg p, \rho) = \begin{cases} \mathit{true} & p \notin a_0 \\ \mathit{false} & p \in a_0 \end{cases}$
- $succ(\varphi_1 \wedge \varphi_2, \rho) = \{\psi_1 \wedge \psi_2 \mid \psi_1 \in succ(\varphi_1, \rho), \psi_2 \in succ(\varphi_2, \rho)\}$
- $succ(\varphi_1 \vee \varphi_2, \rho) = succ(\varphi_1, \rho) \vee succ(\varphi_2, \rho)$
- $succ(\varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2, \rho) =$

$$\begin{array}{ll} succ(\varphi_2, \rho) & \mathit{if}\ |I_0| > d \\ succ(\varphi_2, \rho) \cup \{\psi \wedge (\varphi_1 \, \mathbf{U}_{\leq d-|I_0|} \, \varphi_2) \mid \psi \in succ(\varphi_1, \rho)\} & \mathit{otherwise} \end{array}$$

- $succ(\varphi_1 \, \mathbf{R}_{\leq d} \, \varphi_2, \rho) =$

$$\begin{array}{ll} succ(\varphi_2, \rho) & \mathit{if}\ |I_0| > d \\ \{\psi_1 \wedge \psi_2 \mid \psi_1 \in (succ(\varphi_1, \rho) \cup \{\varphi_1 \, \mathbf{R}_{\leq d-|I_0|} \, \varphi_2\}), & \\ \qquad \psi_2 \in succ(\varphi_2, \rho)\} & \mathit{otherwise} \end{array}$$

Example Now we demonstrate the applicability of the function $succ$ on the observation sequence $\rho = (\{\{r, m\}, \{m\}, \{o, m\}, \{o\}, \ldots), (0, 1], (1, 3], (3, 7], (7, 15], \ldots))$ and the formula $\mathbf{G}(o \Rightarrow \mathbf{F}_{\leq 20}(\neg o))$. To keep the intermediate formulas as simple as possible we use reduction rules from the Lemma 4.9.

1. $succ(\mathbf{G}(o \Rightarrow \mathbf{F}_{\leq 20}(\neg o)), \rho) = \mathbf{G}(o \Rightarrow \mathbf{F}_{\leq 20}(\neg o))$
2. $succ(\mathbf{G}(o \Rightarrow \mathbf{F}_{\leq 20}(\neg o)), \rho^1) = \mathbf{G}(o \Rightarrow \mathbf{F}_{\leq 20}(\neg o))$
3. $succ(\mathbf{G}(o \Rightarrow \mathbf{F}_{\leq 20}(\neg o)), \rho^3) = \mathbf{G}(o \Rightarrow \mathbf{F}_{\leq 20}(\neg o)) \wedge \mathbf{F}_{\leq 16}(\neg o)$
4. $succ(\mathbf{G}(o \Rightarrow \mathbf{F}_{\leq 20}(\neg o)) \wedge \mathbf{F}_{\leq 16}(\neg o), \rho^7) = \mathbf{G}(o \Rightarrow \mathbf{F}_{\leq 20}(\neg o)) \wedge \mathbf{F}_{\leq 8}(\neg o)$

Lemma 4.5 *The function $succ(\varphi, \rho)$ can be computed using polynomial space with respect to the number of distinct subformulas in φ.*

Proof Directly from the definition of $succ(\varphi, \rho)$.

■

The function $succ$ defined here is a real-time counterpart of the function $succ$ from the Definition 3.4. To state the key lemma, we need an auxiliary notion that is defined below. It provides for a discretization of time, which turns out to be essential for a practical use of the function $succ$.

Definition 4.6 *Let φ is an MITL$_{\leq}$ formula and $\rho = ((a_0, a_1, \ldots), (I_0, I_1, \ldots))$ is a timed observation sequence. Then ρ is φ-*fine *if and only if for any subformula ψ of φ and any $t \in \mathbb{R}_0^+$ $\rho^t \models \psi$ implies $\rho^{t'} \models \psi$ for all $t' \in I_k$, where $k \in \mathbb{N}_0$ is such that $t \in I_k$.*

Lemma 4.7 *Let φ is an MITL$_{\leq}$ formula and $\rho = ((a_0, a_1, \ldots), (I_0, I_1, \ldots))$ is a φ-fine timed observation sequence. Then $\rho \models \varphi \Leftrightarrow \exists(\psi \in succ(\varphi, \rho))(\rho^{|I_0|} \models \psi)$.*

Proof By induction on the structure of formula φ.

- $\varphi \equiv true$

  "$\Rightarrow$": As $succ(true, \rho) = true$ and $\rho' \models true$ for any timed observation sequence ρ', $\exists(\psi \in succ(true, \rho))(\rho^{|I_0|} \models true)$.

  "$\Leftarrow$": By contraposition. As $\rho' \not\models true$ is not satisfied for any timed observation sequence ρ', $\rho \not\models true \Rightarrow \forall(\psi \in succ(true, \rho))(\rho^{|I_0|} \not\models \psi)$.

- $\varphi \equiv false$

  "$\Rightarrow$": As $\rho' \models false$ is not satisfied for any timed observation sequence ρ', $\rho \models false \Rightarrow \exists(\psi \in succ(false, \rho))(\rho^{|I_0|} \models \psi)$.

  "$\Leftarrow$": By contraposition. As $succ(false, \rho) = false$ and $\rho' \not\models false$ for any timed observation sequence ρ', $\forall(\psi \in succ(false, \rho))(\rho^{|I_0|} \not\models \psi)$.

- $\varphi \equiv p$

  "$\Rightarrow$": Because $\rho \models p \Rightarrow p \in s_0 \Rightarrow succ(p, \rho) = true$ and $\rho' \models true$ for any timed observation sequence ρ', $\exists(\psi \in succ(p, \rho))(\rho^{|I_0|} \models true)$.

  "$\Leftarrow$": By contraposition. As $\rho \not\models p \Rightarrow p \notin s_0 \Rightarrow succ(p, \rho) = false$ and $\rho' \not\models false$ for any ρ', $\forall(\psi \in succ(p, \rho))(\rho^{|I_0|} \not\models false)$.

- $\varphi \equiv \neg p$

  Proof of this case is an analogy to the previous one.

- $\varphi \equiv \varphi_1 \wedge \varphi_2$

  "$\Rightarrow$": Clearly, $\rho \models \varphi_1 \wedge \varphi_2$ implies $\rho \models \varphi_1$ and $\rho \models \varphi_2$. Then $\exists(\psi_1 \in succ(\varphi_1, \rho))(\rho^{|I_0|} \models \psi_1)$ and $\exists(\psi_2 \in succ(\varphi_2, \rho))(\rho^{|I_0|} \models \psi_2)$ using induction hypothesis. Thus exists $\psi_1 \wedge \psi_2$ from $succ(\varphi_1 \wedge \varphi_2, \rho)$ such that $\rho^{|I_0|} \models \psi_1 \wedge \psi_2$. In other words, $\exists(\psi \in succ(\varphi_1 \wedge \varphi_2, \rho))(\rho^{|I_0|} \models \psi)$.

  "$\Leftarrow$": By contraposition. $\rho \not\models \varphi_1 \wedge \varphi_2$ implies $\rho \not\models \varphi_1$ or $\rho \not\models \varphi_2$. Without loss of generality let $\rho \not\models \varphi_1$. By induction hypothesis $\forall(\psi_1 \in succ(\varphi_1, \rho))(\rho^{|I_0|} \not\models \psi_1)$. Further let $\psi_2 \in succ(\varphi_2, \rho)$. Clearly, $\psi_1 \wedge \psi_2 \in$

$succ(\varphi_1 \wedge \varphi_2, \rho)$ and $\rho^{|I_0|} \not\models \psi_1 \wedge \psi_2$ for any $\psi_1 \in succ(\varphi_1, \rho)$. Therefore $\forall(\psi \in succ(\varphi_1 \wedge \varphi_2, \rho))(\rho^{|I_0|} \not\models \psi)$.

- $\varphi \equiv \varphi_1 \vee \varphi_2$

  "$\Rightarrow$": Clearly, $\rho \models \varphi_1 \vee \varphi_2$ implies $\rho \models \varphi_1$ or $\rho \models \varphi_2$. Without loss of generality let $\rho \models \varphi_1$. Then $\exists(\psi_1 \in succ(\varphi_1, \rho))(\rho^{|I_0|} \models \psi_1)$ using induction hypothesis. Further $\psi_1 \in succ(\varphi_1 \vee \varphi_2, \rho)$ and therefore $\exists(\psi \in succ(\varphi_1 \vee \varphi_2, \rho))(\rho^{|I_0|} \models \psi)$.

  "$\Leftarrow$": By contraposition. $\rho \not\models \varphi_1 \vee \varphi_2$ implies $\rho \not\models \varphi_1$ and $\rho \not\models \varphi_2$. If $\psi \in succ(\varphi_1 \vee \varphi_2, \rho)$, then either $\psi \in succ(\varphi_1, \rho)$ or $\psi \in succ(\varphi_2, \rho)$ and by induction hypothesis $\rho^{|I_0|} \not\models \psi$. Therefore $\forall(\psi \in succ(\varphi_1 \vee \varphi_2, \rho))$ $(\rho^{|I_0|} \not\models \psi)$.

- $\varphi \equiv \varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2$

  "$\Rightarrow$": $\rho \models \varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2$ implies $\exists(t \leq d)(\rho^t \models \varphi_2$ and $\forall(0 \leq t' < t)$ $(\rho^{t'} \models \varphi_1))$. First, let $t \in I_0$. As ρ is φ-fine, $\rho \models \varphi_2$. By induction hypothesis, $\exists(\psi_2 \in succ(\varphi_2, \rho))(\rho^{|I_0|} \models \psi_2)$. Finally, as $succ(\varphi_2, \rho) \subseteq succ(\varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2, \rho)$, $\exists(\psi \in succ(\varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2, \rho))(\rho^{|I_0|} \models \psi)$.

  Next, let $t \notin I_0$. Then for all $t' \in I_0$, $\rho^{t'} \models \varphi_1$ and $\rho^{t'} \not\models \varphi_2$. In this case $|I_0| \leq d$ and $\rho \models \varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2$ implies $\rho^{|I_0|} \models \varphi_1 \, \mathbf{U}_{\leq d-|I_0|} \, \varphi_2$. Further, as $\rho \models \varphi_1$, $\exists(\psi_1 \in succ(\varphi_1, \rho))(\rho^{|I_0|} \models \psi_1)$ using induction hypothesis. Clearly, $\psi_1 \wedge (\varphi_1 \, \mathbf{U}_{\leq d-|I_0|} \, \varphi_2) \in succ(\varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2, \rho)$ and therefore $\exists(\psi \in succ(\varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2, \rho))(\rho^{|I_0|} \models \psi)$.

  "$\Leftarrow$": $\exists(\psi \in succ(\varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2, \rho))(\rho^{|I_0|} \models \psi)$ implies $\exists(\psi \in succ(\varphi_2, \rho))$ $(\rho^{|I_0|} \models \psi)$ or $\exists(\psi \in succ(\varphi_1, \rho))(\rho^{|I_0|} \models \psi)$, $\rho^{|I_0|} \models \varphi_1 \, \mathbf{U}_{\leq d-|I_0|} \, \varphi_2$ and $|I_0| \leq d$. First, let $\exists(\psi \in succ(\varphi_2, \rho))(\rho^{|I_0|} \models \psi)$. By induction hypothesis $\rho \models \varphi_2$ and therefore $\rho \models \varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2$.

  Next, let $\forall(\psi \in succ(\varphi_2, \rho))(\rho^{|I_0|} \not\models \psi)$, $\exists(\psi \in succ(\varphi_1, \rho))(\rho^{|I_0|} \models \psi)$, $\rho^{|I_0|} \models \varphi_1 \, \mathbf{U}_{\leq d-|I_0|} \, \varphi_2$ and $|I_0| \leq d$. By induction hypothesis $\rho \models \varphi_1$, $\rho \not\models \varphi_2$. Further, as ρ is φ-fine, $\rho^t \models \varphi_1$ and $\rho^t \not\models \varphi_2$ for all $t \in I_0$. In this case $\rho^{|I_0|} \models \varphi_1 \, \mathbf{U}_{\leq d-|I_0|} \, \varphi_2$ implies $\rho \models \varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2$.

- $\varphi \equiv \varphi_1 \, \mathbf{R}_{\leq d} \, \varphi_2$

  "$\Rightarrow$": $\rho \models \varphi_1 \, \mathbf{R}_{\leq d} \, \varphi_2$ implies $\forall(t \leq d)(\rho^t \models \varphi_1$ or $\exists(0 \leq t' < t)$ $(\rho^{t'} \models \varphi_2))$. Therefore $\rho \models \varphi_2$ and as ρ is φ-fine $\rho^t \models \varphi_2$ for all $t \in I_0$. By induction hypothesis $\exists(\psi_2 \in succ(\varphi_2, \rho))(\rho^{|I_0|} \models \psi_2)$.

  First, let $|I_0| > d$. Then $succ(\varphi_1 \, \mathbf{R}_{\leq d} \, \varphi_2, \rho) = succ(\varphi_2, \rho)$ and therefore $\exists(\psi \in succ(\varphi_1 \, \mathbf{R}_{\leq d} \, \varphi_2, \rho))(\rho^{|I_0|} \models \psi)$.

Next, let $|I_0| \leq d$ and $\exists(t \in I_0)(\rho^t \models \varphi_1)$. Then $\rho \models \varphi_1$ because ρ is φ-fine. Further $\exists(\psi_1 \in succ(\varphi_1, \rho))(\rho^{|I_0|} \models \psi_1)$ using induction hypothesis . Clearly, $\psi_1 \wedge \psi_2 \in succ(\varphi_1 \mathbf{R}_{\leq d} \varphi_2, \rho)$ and $\rho^{|I_0|} \models \psi_1 \wedge \psi_2$. Therefore $\exists(\psi \in succ(\varphi_1 \mathbf{R}_{\leq d} \varphi_2, \rho))(\rho^{|I_0|} \models \psi)$.

Finally, let $|I_0| \leq d$ and $\forall(t \in I_0)(\rho^t \not\models \varphi_1)$. In this case $\rho \models \varphi_1 \mathbf{R}_{\leq d} \varphi_2$ implies $\rho^{|I_0|} \models \varphi_1 \mathbf{R}_{\leq d-|I_0|} \varphi_2$. Clearly, $\rho^{|I_0|} \models \psi_1 \wedge (\varphi_1 \mathbf{R}_{\leq d-|I_0|} \varphi_2)$ and $\psi_1 \wedge (\varphi_1 \mathbf{R}_{\leq d-|I_0|} \varphi_2) \in succ(\varphi_1 \mathbf{R}_{\leq d} \varphi_2, \rho)$. Therefore $\exists(\psi \in succ(\varphi_1 \mathbf{R}_{\leq d} \varphi_2, \rho))$ $(\rho^{|I_0|} \models \psi)$.

"$\Leftarrow$": First, let $|I_0| > d$. Then $\exists(\psi_2 \in succ(\varphi_2, \rho))(\rho^{|I_0|} \models \psi_2)$. Further $\rho \models \varphi_2$ using induction hypothesis and, as ρ is φ-fine, $\rho^t \models \varphi_2$ for all $t \in I_0$. Namely $\rho^t \models \varphi_2$ for all $t \leq d$. Therefore $\rho \models \varphi_1 \mathbf{R}_{\leq d} \varphi_2$.

Next, let $|I_0| \leq d$. Then again $\exists(\psi_2 \in succ(\varphi_2, \rho))(\rho^{|I_0|} \models \psi_2)$ and $\exists(\psi_1 \in succ(\varphi_1, \rho))(\rho^{|I_0|} \models \psi_1)$ or $\rho^{|I_0|} \models \varphi_1 \mathbf{R}_{\leq d-|I_0|} \varphi_2$. First, let $\exists(\psi_1 \in succ(\varphi_1, \rho))(\rho^{|I_0|} \models \psi_1)$. Then $\rho \models \varphi_1$ and $\rho \models \varphi_2$ using induction hypothesis. Thus $\rho \models \varphi_1 \mathbf{R}_{\leq d} \varphi_2$. Finally, let $\forall(\psi_1 \in succ(\varphi_1, \rho))$ $(\rho^{|I_0|} \not\models \psi_1)$ and $\rho^{|I_0|} \models \varphi_1 \mathbf{R}_{\leq d-|I_0|} \varphi_2$. By induction hypothesis $\rho \not\models \varphi_1$ and $\rho \models \varphi_2$. As ρ is φ-fine, $\rho^t \not\models \varphi_1$ and $\rho^t \models \varphi_2$ for all $t \in I_0$. In this case $\rho^{|I_0|} \models \varphi_1 \mathbf{R}_{\leq d-|I_0|} \varphi_2$ implies $\rho \models \varphi_1 \mathbf{R}_{\leq d} \varphi_2$.

■

Thus given an MITL$_{\leq}$ formula φ and a φ-fine timed observation sequence ρ we can resolve the validity of $\rho \models \varphi$ by iteratively computing values of the function $succ$. This suggest an algorithm for offline monitoring of MITL$_{\leq}$ as any timed observation sequence can be easily transformed to a formula-fine timed observation sequence (see [17], page 161).

To enable a practical use, we need to show that the formulas resulting from an iterative computation of the function $succ$ may not continuosly grow. To this aim, we measure the size of a formula using characteristics of its DPNF.

Definition 4.8 *Let φ be an MITL$_{\leq}$ formula. Then the* size *of φ, denoted as $|\varphi|$, is the sum of the number of all literals in each clause. Further we say that a subformula of MITL$_{\leq}$ formula φ is* basic *if it is of the form $p, \neg p, \varphi_1 \mathbf{U}_{\leq d} \varphi_2$ or $\varphi_1 \mathbf{R}_{\leq d} \varphi_2$.*

Lemma 4.9 *Let φ be an MITL$_{\leq}$ formula and ρ be a timed observation sequence. Then $|succ^i(\varphi, \rho)| \in \mathcal{O}(2^{|\varphi|})$ for every $i \in \mathbb{N}_0$.*

Proof We derive the upper bound on the size of $|succ^i(\varphi, \rho)|$ from the properties of DPNF. A formula in DPNF can be reduced using the follow-

ing rules. Note that, the last rule justifies alternative representations of a formula in DPNF.

1. $\mathit{false} \wedge \varphi^i_j \equiv \mathit{false}$ and $\mathit{false} \vee \varphi^i_j \equiv \varphi^i_j$
2. $\mathit{true} \wedge \varphi^i_j \equiv \varphi^i_j$ and $\mathit{true} \vee \varphi^i_j \equiv \mathit{true}$
3. if exist distinct i, j such that
   - $p \in \alpha_i \Rightarrow p \in \alpha_j$,
   - $\neg p \in \alpha_i \Rightarrow \neg p \in \alpha_j$,
   - $\psi_1 \, \mathbf{U}_{\leq d_1} \, \psi_2 \in \alpha_i \Rightarrow \exists (d_2 \leq d_1)(\psi_1 \, \mathbf{U}_{\leq d_2} \, \psi_2 \in \alpha_j)$,
   - and $\psi_1 \, \mathbf{R}_{\leq d_1} \, \psi_2 \in \alpha_i \Rightarrow \exists (d_2 \geq d_1)(\psi_1 \, \mathbf{R}_{\leq d_1} \, \psi_2 \in \alpha_j)$

   then $\varphi \equiv \varphi \setminus \{\alpha_j\}$
4. if exists j such that $\psi_1 \, \mathbf{U}_{\leq d_1} \, \psi_2 \in \alpha_j$, $\psi_1 \, \mathbf{U}_{\leq d_2} \, \psi_2 \in \alpha_j$ and $d_1 < d_2$ then $\alpha_j \equiv \alpha_j \setminus \{\psi_1 \, \mathbf{U}_{\leq d_2} \, \psi_2\}$
5. if exists j such that $\psi_1 \, \mathbf{R}_{\leq d_1} \, \psi_2 \in \alpha_j$, $\psi_1 \, \mathbf{R}_{\leq d_2} \, \psi_2 \in \alpha_j$ and $d_1 < d_2$ then $\alpha_j \equiv \alpha_j \setminus \{\psi_1 \, \mathbf{R}_{\leq d_1} \, \psi_2\}$
6. duplicate clauses and duplicate literals in a clause can be removed

Let B be the set of all basic subformulas of φ and n be the cardinality of B. First, we prove that the size of formula φ is at most exponential in n. This is indeed truth as each clause, except for special clauses true and false, is a subset of B and the number of literals in each clause is at most equal to n. Therefore $|\varphi| \in \mathcal{O}(2^n)$.

Next, from the definition of the function succ it follows that new basic subformulas can be introduced through its application. For instance, when succ is applied on a subformula $\psi_1 \, \mathbf{U}_{\leq d_1} \, \psi_2$, new basic subformula $\psi_1 \, \mathbf{U}_{\leq d_2} \, \psi_2$ may be introduced. However, application of the function succ can change the bound of the topmost temporal operator only. Therefore, though the set of potential basic subformulas is arbitrarily large, any clause can be reduced to n elements using reduction rules 4 and 5.

Furthermore thanks to the reduction the rule 3, it is possible to keep the number of clauses below 2^n. This can be proved as follows. Let there is more than 2^n pairwise distinct clauses such that none can be eliminated by the rule 3. Then there exist distinct clauses α_i, α_j, reals d_1, d_2, e_1, e_2 and MITL$_{\leq}$ formulas $\varphi_1, \varphi_2, \psi_1$, and ψ_2 such that one of the following is true:

- $\varphi_1 \mathbf{U}_{\leq d_1} \varphi_2 \in \alpha_i$, $\psi_1 \mathbf{U}_{\leq e_1} \psi_2 \in \alpha_i$, $\varphi_1 \mathbf{U}_{\leq d_2} \varphi_2 \in \alpha_j$, $\psi_1 \mathbf{U}_{\leq e_2} \psi_2 \in \alpha_j$, $d_1 < d_2$, $e_1 > e_2$, $\varphi_1 \mathbf{U}_{\leq d_1} \varphi_2$ and $\varphi_1 \mathbf{U}_{\leq d_2} \varphi_2$ are results of different number of applications of function $succ$ on some formula $\varphi_1 \mathbf{U}_{\leq d} \varphi_2$, and similarly $\psi_1 \mathbf{U}_{\leq e_1} \psi_2$ and $\psi_1 \mathbf{U}_{\leq e_2} \psi_2$ are results of different number of applications of function $succ$ on some formula $\psi_1 \mathbf{U}_{\leq e} \psi_2$.

- $\varphi_1 \mathbf{U}_{\leq d_1} \varphi_2 \in \alpha_i$, $\psi_1 \mathbf{R}_{\leq e_1} \psi_2 \in \alpha_i$, $\varphi_1 \mathbf{U}_{\leq d_2} \varphi_2 \in \alpha_j$, $\psi_1 \mathbf{R}_{\leq e_2} \psi_2 \in \alpha_j$, $d_1 < d_2$, $e_1 < e_2$, $\varphi_1 \mathbf{U}_{\leq d_1} \varphi_2$ and $\varphi_1 \mathbf{U}_{\leq d_2} \varphi_2$ are results of different number of applications of function $succ$ on some formula $\varphi_1 \mathbf{U}_{\leq d} \varphi_2$, and similarly $\psi_1 \mathbf{R}_{\leq e_1} \psi_2$ and $\psi_1 \mathbf{R}_{\leq e_2} \psi_2$ are results of different number of applications of function $succ$ on some formula $\psi_1 \mathbf{R}_{\leq e} \psi_2$.

- $\varphi_1 \mathbf{U}_{\leq d_1} \varphi_2 \in \alpha_i$, $\psi_1 \mathbf{R}_{\leq e_1} \psi_2 \in \alpha_i$, $\varphi_1 \mathbf{U}_{\leq d_2} \varphi_2 \in \alpha_j$, $\psi_1 \mathbf{R}_{\leq e_2} \psi_2 \in \alpha_j$, $d_1 > d_2$, $e_1 > e_2$, $\varphi_1 \mathbf{U}_{\leq d_1} \varphi_2$ and $\varphi_1 \mathbf{U}_{\leq d_2} \varphi_2$ are results of different number of applications of function $succ$ on some formula $\varphi_1 \mathbf{U}_{\leq d} \varphi_2$, and similarly $\psi_1 \mathbf{R}_{\leq e_1} \psi_2$ and $\psi_1 \mathbf{R}_{\leq e_2} \psi_2$ are results of different number of applications of function $succ$ on some formula $\psi_1 \mathbf{R}_{\leq e} \psi_2$.

- $\varphi_1 \mathbf{R}_{\leq d_1} \varphi_2 \in \alpha_i$, $\psi_1 \mathbf{R}_{\leq e_1} \psi_2 \in \alpha_i$, $\varphi_1 \mathbf{R}_{\leq d_2} \varphi_2 \in \alpha_j$, $\psi_1 \mathbf{R}_{\leq e_2} \psi_2 \in \alpha_j$, $d_1 < d_2$, $e_1 > e_2$, $\varphi_1 \mathbf{R}_{\leq d_1} \varphi_2$ and $\varphi_1 \mathbf{R}_{\leq d_2} \varphi_2$ are results of different number of applications of function $succ$ on some formula $\varphi_1 \mathbf{R}_{\leq d} \varphi_2$, and similarly $\psi_1 \mathbf{R}_{\leq e_1} \psi_2$ and $\psi_1 \mathbf{R}_{\leq e_2} \psi_2$ are results of different number of applications of function $succ$ on some formula $\psi_1 \mathbf{R}_{\leq e} \psi_2$.

In the first case, existence of such $\alpha_i, \alpha_j, \ldots$ implies existence of clause α_k and α_l such that $\alpha_k \equiv \alpha_i \cup \{\varphi_1 \mathbf{U}_{\leq d_2} \varphi_2\}$ and $\alpha_l \equiv \alpha_j \cup \{\varphi_1 \mathbf{U}_{\leq e_1} \varphi_2\}$. However, this contradicts our assumption as both α_i and α_j can be eliminated by the rule 3 because of α_k and α_l respectively. Other cases are handled analogously. Consequently, $|succ^i(\varphi, w)| \in \mathcal{O}(2^n)$ for every $i \in \mathbb{N}$ and as $n \leq |\varphi|$ the proof is complete.

■

We have shown how to use the function $succ$ for offline monitoring. As opposed to algorithms based on the automata-theoretic approach, the algorithm based on the function $succ$ targets the task of deciding MITL$_{\leq}$ directly, without employing the (timed) automata framework. Although theoretical complexity of both approaches is the same, the latter is more efficient in practice.

Unfortunately, timed observation sequences resulting from online monitoring are not always formula-fine. This issue can be dealt with in several ways. An approach suitable for discrete-time systems is to compute the value of the function $succ$ each unit of time. However, this approach is of

practical use only when a computation of the function $succ$ can be done within the unit of time. Moreover, real-time systems are out of reach of this approach.

The following lemma indicates another possibility of how to cope with timed observation sequences that are not formula-fine. In parcticular, the lemma identifies fragments of MITL$_{\leq}$ for which the function $succ$ is of some use even when the timed observation sequence being verified is not formula-fine.

Lemma 4.10 *Let* $\rho_1 = ((a_0, a_1, \ldots), (I_0, I_1, \ldots))$ *and* $\rho_2 = ((a_0, a_0, a_1, \ldots), (I'_0, I''_0, I_1, \ldots))$ *are timed observation sequences such that* $I_0 = I'_0 \cup I''_0$*, and* φ *be an MITL*$_{\leq}$ *formula. Then for any timed observation sequence* π*,*

1. *If* $\varphi \in MITL_{\leq}(\mathbf{U}_{\leq d}, \mathbf{G})$*, then* $\exists(\chi \in succ(\varphi, \rho_1))(\pi \models \chi)$ *implies* $\exists(\psi \in succ(\varphi, \rho_2))(\exists(\chi \in succ(\psi, \rho_2^{|I'_0|}))(\pi \models \chi))$*.*

2. *If* $\varphi \in MITL_{\leq}(\mathbf{R}_{\leq d}, \mathbf{F})$*, then* $\exists(\psi \in succ(\varphi, \rho_2))(\exists(\chi \in succ(\psi, \rho_2^{|I'_0|}))$ $(\pi \models \chi))$ *implies* $\exists \chi \in succ(\varphi, \rho_1)(\pi \models \chi)$*.*

Proof By induction on the structure of formula φ. For the base step of both lemmas we prove that $succ(\varphi, \rho_1) = \bigcup_{\psi \in succ(\varphi, \rho_2)} succ(\psi, \rho_2^{|I'_0|})$.

- $\varphi \equiv true$

  Clearly, $succ(true, \rho_1)$ as well as $succ(true, \rho_2)$ equals $true$. Therefore $\bigcup_{\psi \in succ(true, \rho_2)} succ(\psi, \rho_2^{|I'_0|}) = \bigcup_{\psi \in true} succ(\psi, \rho_2^{|I'_0|}) = true$.

- $\varphi \equiv false$

  Clearly, $succ(false, \rho_1)$ as well as $succ(false, \rho_2)$ equals $false$. Therefore $\bigcup_{\psi \in succ(false, \rho_2)} succ(\psi, \rho_2^{|I'_0|}) = \bigcup_{\psi \in false} succ(\psi, \rho_2^{|I'_0|}) = false$.

- $\varphi \equiv p$

  First, let $p \in s_0$. Clearly, $succ(p, \rho_1)$ as well as $succ(p, \rho_2)$ equals $true$. Therefore $\bigcup_{\psi \in succ(p, \rho_2)} succ(\psi, \rho_2^{|I'_0|}) = \bigcup_{\psi \in true} succ(\psi, \rho_2^{|I'_0|}) = true$.

  Next, let $p \notin s_0$. Clearly, $succ(p, \rho_1)$ as well as $succ(p, \rho_2)$ equals $false$. Therefore $\bigcup_{\psi \in succ(p, \rho_2)} succ(\psi, \rho_2^{|I'_0|}) = \bigcup_{\psi \in false} succ(\psi, \rho_2^{|I'_0|}) = false$.

- $\varphi \equiv \neg p$

  Proof of this case is an analogy to the previous one.

Inductive step for the first lemma:

- $\varphi \equiv \varphi_1 \wedge \varphi_2$

  Clearly, $\exists(\chi \in succ(\varphi_1 \wedge \varphi_2, \rho_1))(\pi \models \chi)$ implies $\exists(\chi_1 \in succ(\varphi_1, \rho_1))$ $(\pi \models \chi_1)$ and $\exists(\chi_2 \in succ(\varphi_2, \rho_1))(\pi \models \chi_2)$. Consequently, $\exists(\psi_1 \in succ(\varphi_1, \rho_2))(\exists(\chi_1 \in succ(\psi_1, \rho_2^{|I_0'|}))(\pi \models \chi_1))$ and $\exists(\psi_2 \in succ(\varphi_2, \rho_2))$ $(\exists(\chi_2 \in succ(\psi_2, \rho_2^{|I_0'|}))(\pi \models \chi_2))$ using induction hypotesis.

  Therefore exists $\psi_1 \in succ(\varphi_1, \rho_2)$ and $\psi_2 \in succ(\varphi_2, \rho_2)$ such that $\exists(\chi \in succ(\psi_1 \wedge \psi_2, \rho_2^{|I_0'|}))(\pi \models \chi))$. Finally, $\exists(\psi \in succ(\varphi_1 \wedge \varphi_2, \rho_2))$ $(\exists(\chi \in succ(\psi, \rho_2^{|I_0'|}))(\pi \models \chi))$

- $\varphi \equiv \varphi_1 \vee \varphi_2$

  Clearly, $\exists(\chi \in succ(\varphi_1 \vee \varphi_2, \rho_1))(\pi \models \chi)$ implies $\exists(\chi_1 \in succ(\varphi_1, \rho_1))$ $(\pi \models \chi_1)$ or $\exists(\chi_2 \in succ(\varphi_2, \rho_1))(\pi \models \chi_2)$. Consequently, $\exists(\psi_1 \in succ(\varphi_1, \rho_2))(\exists(\chi_1 \in succ(\psi_1, \rho_2^{|I_0'|}))(\pi \models \chi_1))$ or $\exists(\psi_2 \in succ(\varphi_2, \rho_2))$ $(\exists(\chi_2 \in succ(\psi_2, \rho_2^{|I_0'|}))(\pi \models \chi_2))$ using induction hypothesis.

  Therefore exists ψ_1 from $succ(\varphi_1, \rho_2)$ or ψ_2 from $succ(\varphi_2, \rho_2)$ such that $\exists(\chi \in succ(\psi_1 \vee \psi_2, \rho_2^{|I_0'|}))(\pi \models \chi))$. Finally, $\exists(\psi \in succ(\varphi_1 \vee \varphi_2, \rho_2))$ $(\exists(\chi \in succ(\psi, \rho_2^{|I_0'|}))(\pi \models \chi))$.

- $\varphi \equiv \varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2$

  First, let $|I_0| > d$. Then $\exists(\chi \in succ(\varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2, \rho_1))(\pi \models \chi)$ implies $\exists(\chi_2 \in succ(\varphi_2, \rho_1))(\pi \models \chi_2)$. Consequently, $\exists(\psi_2 \in succ(\varphi_2, \rho_2))$ $(\exists(\chi_2 \in succ(\psi_2, \rho_2^{|I_0'|}))(\pi \models \chi_2))$ using induction hypothesis. Finally, $\exists(\psi \in succ(\varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2, \rho_2))(\exists(\chi \in succ(\psi, \rho_2^{|I_0'|}))(\pi \models \chi))$.

  Next, let $|I_0| \leq d$. Then $\exists(\chi \in succ(\varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2, \rho_1))(\pi \models \chi)$ implies $\exists(\chi_2 \in succ(\varphi_2, \rho_1))(\pi \models \chi_2)$ or $\exists(\chi_1 \in succ(\varphi_1, \rho_1))(\pi \models \chi_1)$ and $\pi \models \varphi_1 \, \mathbf{U}_{\leq d-|I_0|} \, \varphi_2)$.

  If $\exists(\chi_2 \in succ(\varphi_2, \rho_1))(\pi \models \chi_2)$ we use the same argument as before.

  Otherwise, $\exists(\chi_1 \in succ(\varphi_1, \rho_1))(\pi \models \chi_1)$ and $\pi \models \varphi_1 \mathbf{U}_{\leq d-|I_0|} \varphi_2$. Then $\exists(\psi_1 \in succ(\varphi_1, \rho_2))(\exists(\chi_1 \in succ(\psi_1, \rho_2^{|I_0'|}))(\pi \models \chi_1))$ using induction hypothesis. Further, as $\varphi_1 \in \text{MITL}_{\leq}(\mathbf{U}_{\leq d}, \mathbf{G})$, $\exists(\chi_1 \in succ(\varphi_1, \rho_1))$ $(\pi \models \chi_1)$ implies $\exists(\chi_1 \in succ(\varphi_1, \rho_2^{|I_0'|}))(\pi \models \chi_1)$ using a simple argument. Finally, $\exists(\chi_1 \in succ(\varphi_1, \rho_2^{|I_0'|}))(\pi \models \chi_1)$ together with $\pi \models \varphi_1 \, \mathbf{U}_{\leq d-|I_0|} \, \varphi_2$ implies $\exists(\chi \in succ(\varphi_1 \, \mathbf{U}_{\leq d-|I_0'|} \, \varphi_2, \rho_2^{|I_0'|}))(\pi \models \chi)$.

In summary, $\exists(\chi_1 \in succ(\varphi_1, \rho_1))(\pi \models \chi_1)$ and $\pi \models \varphi_1 \mathbf{U}_{\leq d-|I_0|} \varphi_2$ implies $\exists(\psi_1 \in succ(\varphi_1, \rho_2))(\exists(\chi_1 \in succ(\psi_1, \rho_2^{|I_0'|}))(\pi \models \chi_1))$ and $\exists(\chi \in succ(\varphi_1 \mathbf{U}_{\leq d-|I_0'|} \varphi_2, \rho_2^{|I_0'|}))(\pi \models \chi)$. These two together imply $\exists(\psi \in succ(\varphi_1 \mathbf{U}_{\leq d} \varphi_2, \rho_2))\ (\exists\chi \in succ(\psi, \rho_2^{|I_0'|}))(\pi \models \chi))$.

- $\varphi \equiv \mathbf{G}\varphi_1$

  Clearly, $\exists(\chi \in succ(\mathbf{G}\varphi_1, \rho_1))(\pi \models \chi)$ implies $\exists(\chi_1 \in succ(\varphi_1, \rho_1))$ $(\pi \models \chi_1)$ and $\pi \models \mathbf{G}\varphi_1$. By induction hypothesis, $\exists(\psi_1 \in succ(\varphi_1, \rho_2))$ $(\exists(\chi_1 \in succ(\psi_1, \rho_2^{|I_0'|}))(\pi \models \chi_1))$. Further, as $\varphi_1 \in \text{MITL}_{\leq}(\mathbf{U}_{\leq d}, \mathbf{G})$, $\exists(\chi_1 \in succ(\varphi_1, \rho_1))(\pi \models \chi_1)$ implies $\exists(\chi_1 \in succ(\varphi_1, \rho_2^{|I_0|}))(\pi \models \chi_1)$ using a simple argument. Finally, $\pi \models \mathbf{G}\varphi_1$, $\exists(\psi_1 \in succ(\varphi_1, \rho_2))$ $(\exists(\chi_1 \in succ(\psi_1, \rho_2^{|I_0'|}))\ (\pi \models \chi_1))$, and $\exists(\chi_1 \in succ(\varphi_1, \rho_2^{I_0}))(\pi \models \chi_1)$ implies $\exists(\psi \in succ(\mathbf{G}\varphi_1, \rho_2))\ (\exists(\chi \in succ(\psi, \rho_2^{|I_0'|}))(\pi \models \chi))$.

Inductive step for the second lemma:

- $\varphi \equiv \varphi_1 \wedge \varphi_2$ and $\varphi \equiv \varphi_1 \vee \varphi_2$

  Proof for each of these cases is an inversion of arguments used for proving the same case in the first lemma.

- $\varphi \equiv \varphi_1 \mathbf{R}_{\leq d} \varphi_2$

  First, let $|I_0| > d$. Then $\exists(\psi \in succ(\varphi_1 \mathbf{R}_{\leq d} \varphi_2, \rho_2))(\exists(\chi \in succ(\psi, \rho_2^{|I_0'|}))$ $(\pi \models \chi))$ clearly implies $\exists(\psi_2 \in succ(\varphi_2, \rho_2))(\exists(\chi_2 \in succ(\psi_2, \rho_2^{|I_0'|}))$ $(\pi \models \chi_2))$. Further $\exists(\chi_2 \in succ(\varphi_2, \rho_1))(\pi \models \chi_2)$ using induction hypothesis. Consequently, $\exists(\chi \in succ(\varphi_1 \mathbf{R}_{\leq d} \varphi_2, \rho_1))(\pi \models \chi)$.

  Next, let $|I_0| \leq d$. Then $\exists(\psi \in succ(\varphi_1 \mathbf{R}_{\leq d} \varphi_2, \rho_2))(\exists(\chi \in succ(\psi, \rho_2^{|I_0'|}))$ $(\pi \models \chi))$ implies $\exists(\psi_2 \in succ(\varphi_2, \rho_2))(\exists(\chi_2 \in succ(\psi_2, \rho_2^{|I_0'|}))(\pi \models \chi_2))$ and either $\exists(\psi_1 \in succ(\varphi_1, \rho_2))(\exists(\chi_1 \in succ(\psi_1, \rho_2^{|I_0'|}))(\pi \models \chi_1))$ or $\exists(\chi \in succ(\varphi_1 \mathbf{R}_{\leq d-|I_0'|} \varphi_2, \rho_2))(\pi \models \chi)$. Clearly, $\exists(\chi_2 \in succ(\varphi_2, \rho_1))$ $(\pi \models \chi_2)$ using induction hypothesis.

  First, let $\exists(\psi_1 \in succ(\varphi_1, \rho_2))(\exists(\chi_1 \in succ(\psi_1, \rho_2^{|I_0'|}))(\pi \models \chi_1))$. Then $\exists(\chi_1 \in succ(\varphi_1, \rho_1)(\pi \models \chi_1)$, again using induction hypothesis. Consequently, $\exists(\chi \in succ(\varphi_1 \mathbf{R}_{\leq d} \varphi_2, \rho_1))(\pi \models \chi)$.

  Next, let $\exists(\chi \in succ(\varphi_1 \mathbf{R}_{\leq d-|I_0'|} \varphi_2, \rho_2^{|I_0'|}))(\pi \models \chi)$. Consequently, $\exists(\chi_2 \in succ(\varphi_2, \rho_2^{|I_0'|}))(\pi \models \chi_2)$ and $\exists(\chi_1 \in succ(\varphi_1, \rho_2^{|I_0'|}))\ (\pi \models \chi_1)$ or $\pi \models \varphi_1 \mathbf{R}_{\leq d-|I_0|} \varphi_2$.

First, let $\exists(\chi_1 \in succ(\varphi_1, \rho_2^{|I'_0|}))(\pi \models \chi_1)$. Then using a simple argument $\exists(\chi_1 \in succ(\varphi_1, \rho_1))(\pi \models \chi_1)$ as $\varphi_1 \in \text{MITL}_{\leq}(\mathbf{R}_{\leq d}, \mathbf{F})$. Consequently, $\exists(\chi \in succ(\varphi_1 \, \mathbf{R}_{\leq d} \, \varphi_2, \rho_1))(\pi \models \chi)$.

Finally, let $\pi \models \varphi_1 \, \mathbf{R}_{\leq d-|I_0|} \, \varphi_2$. Then again $\exists(\chi \in succ(\varphi_1 \, \mathbf{R}_{\leq d} \, \varphi_2, \rho_1))$ $(\pi \models \chi)$.

- $\varphi \equiv \mathbf{F} \, \varphi_1$

  Clearly, $\exists(\psi \in succ(\mathbf{F} \, \varphi_1, \rho_2))(\exists(\chi \in succ(\psi, \rho_2^{|I'_0|}))(\pi \models \chi))$ implies either $\exists(\psi \in succ(\varphi_1, \rho_2))(\exists(\chi \in succ(\psi, \rho_2^{|I'_0|}))(\pi \models \chi))$ or $\exists(\chi \in succ(\mathbf{F} \, \varphi_1, \rho_2^{|I'_0|}))(\pi \models \chi)$.

  First, let $\exists(\psi \in succ(\varphi_1, \rho_2))(\exists(\chi \in succ(\psi, \rho_2^{|I'_0|}))(\pi \models \chi))$. Then $\exists(\chi \in succ(\varphi_1, \rho_1))(\pi \models \chi)$ using induction hypothesis. Consequently, $\exists(\chi \in succ(\mathbf{F} \, \varphi_1, \rho_1))(\pi \models \chi)$.

  Next, let $\exists(\chi \in succ(\mathbf{F} \, \varphi_1, \rho_2^{|I'_0|}))(\pi \models \chi)$. Then $\exists(\chi \in succ(\mathbf{F} \, \varphi_1, \rho_1))$ $(\pi \models \chi)$ using a simple argument as $\varphi_1 \in \text{MITL}_{\leq}(\mathbf{R}_{\leq d}, \mathbf{F})$.

■

Example
We show that opposite implications to those of Lemma 4.10 do not hold:
1. First, let us consider an MITL$_{\leq}$ formula $\varphi = (\, \mathbf{F}_{\leq 10} \, p) \, \mathbf{U}_{\leq 5} \, (\, \mathbf{F}_{\leq 10} \, q)$, a timed observation sequence $\rho_1 = ((\emptyset, \ldots), ([0, 6), \ldots))$, and its refinement $\rho_2 = ((\emptyset, \emptyset, \ldots), ([0, 3), [3, 6), \ldots))$. Then $succ(\varphi, \rho_1) = \{\, \mathbf{F}_{\leq 4} \, q\}$, $succ(\varphi, \rho_2) = \{((\mathbf{F}_{\leq 10} p) \, \mathbf{U}_{\leq 2} \, (\mathbf{F}_{\leq 10} q)) \wedge (\mathbf{F}_{\leq 7} p), \mathbf{F}_{\leq 7} q\}$, $succ(\mathbf{F}_{\leq 7} q, \rho_2^3) = \{\mathbf{F}_{\leq 4} q\}$ and $succ((\mathbf{F}_{\leq 7} p) \wedge ((\mathbf{F}_{\leq 10} p) \mathbf{U}_{\leq 2} (\mathbf{F}_{\leq 10} q)), \rho_2^3) = \{(\mathbf{F}_{\leq 4} p) \wedge (\mathbf{F}_{\leq 7} q)\}$. Clearly, $\{\, \mathbf{F}_{\leq 4} \, q, (\, \mathbf{F}_{\leq 4} \, p) \wedge (\, \mathbf{F}_{\leq 7} \, q)\}$ does not imply $\mathbf{F}_{\leq 4} \, q$.
2. Next, let us consider an MITL$_{\leq}$ formula $\varphi = (\, \mathbf{G}_{\leq 10} \, p) \, \mathbf{R}_{\leq 5} \, (\, \mathbf{G}_{\leq 10} \, q)$, a timed observation sequence $\rho_1 = ((\{p, q\}, \ldots), ([0, 6), \ldots))$, and its refinement $\rho_2 = ((\{p, q\}, \{p, q\}, \ldots), ([0, 3), [3, 6), \ldots))$. Then $succ(\varphi, \rho_1) = \{\, \mathbf{G}_{\leq 4} \, q\}$, $succ(\varphi, \rho_2) = \{(\, (\, \mathbf{G}_{\leq 7} \, q) \wedge ((\, \mathbf{G}_{\leq 10} \, p) \, \mathbf{R}_{\leq 2} \, (\, \mathbf{G}_{\leq 10} \, q)), \, \mathbf{G}_{\leq 7} \, p) \wedge (\, \mathbf{G}_{\leq 7} \, q)\}$, $succ((\, \mathbf{G}_{\leq 7} \, p) \wedge (\, \mathbf{G}_{\leq 7} \, q), \rho_2^3) = \{(\, \mathbf{G}_{\leq 4} \, p) \wedge (\, \mathbf{G}_{\leq 4} \, q)\}$, and $succ((\, \mathbf{G}_{\leq 7} \, q) \wedge ((\, \mathbf{G}_{\leq 10} \, p) \, \mathbf{R}_{\leq 2} \, (\, \mathbf{G}_{\leq 10} \, q)), \rho_2^3) = \{\, \mathbf{G}_{\leq 7} \, q\}$ after a simplification. Clearly, $\mathbf{G}_{\leq 4} \, q$ does not imply $\{(\, \mathbf{G}_{\leq 4} \, p) \wedge (\, \mathbf{G}_{\leq 4} \, q), \, \mathbf{G}_{\leq 7} \, q\}$.

The Lemma 4.10 estabilishes a connection between the validity of a formula from a certain fragment of MITL$_{\leq}$ on a timed observation sequence ρ and its possible *refinement*. A timed observation sequence can be refined by dividing an arbitrary interval in two and duplicating respective state in the underlying observation sequence. For instance in the Lemma 4.10 the

timed observation sequence ρ_2 is a refinement of the timed observation sequence ρ_1. If ρ satisfies a formula from MITL$_{\leq}(\mathbf{U}_{\leq d}, \mathbf{G})$, so does any of its refinements and if it does not satisfy a formula from MITL$_{\leq}(\mathbf{R}_{\leq d}, \mathbf{F})$, neither does any of its refinements.

When evaluating a formula over a timed observation sequence, the function *succ* intuitively takes into account the validity of formulas on borders of the respective intervals only. However, as the timed observation sequence being examined might not be formula-fine, this may result into an incorrect answer.

Consequently, we present an alternative definition of the function *succ*. On the one hand, it removes the requirement of formula-fineness, but on the other hand, it introduces obstacles to its practical utilization.

Definition 4.11 *Given an MITL$_{\leq}$ formula φ and a timed observation sequence $\rho = ((a_0, a_1, \ldots), (I_0, I_1, \ldots))$ we define the function $succ^*(\varphi, \rho)$ inductively on the structure of φ as follows.*

- $succ^*(\mathit{true}, \rho) = \mathit{true}$
- $succ^*(\mathit{false}, \rho) = \mathit{false}$
- $succ^*(p, \rho) = \begin{cases} \mathit{true} & p \in s_0 \\ \mathit{false} & p \notin s_0 \end{cases}$
- $succ^*(\neg p, \rho) = \begin{cases} \mathit{true} & p \notin s_0 \\ \mathit{false} & p \in s_0 \end{cases}$
- $succ^*(\varphi_1 \wedge \varphi_2, \rho) = \{\psi_1 \wedge \psi_2 \mid \psi_1 \in succ^*(\varphi_1, \rho), \psi_2 \in succ^*(\varphi_2, \rho)\}$
- $succ^*(\varphi_1 \vee \varphi_2, \rho) = succ^*(\varphi_1, \rho) \cup succ^*(\varphi_2, \rho)$
- $succ^*(\varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2, \rho) =$

$$\begin{array}{ll} \bigcup_{0 \leq t_1 \leq d} \left(succ^*(\varphi_2, \rho^{t_1}) \wedge \bigwedge_{0 \leq t_2 < t_1} succ^*(\varphi_1, \rho^{t_2}) \right) & \text{if } |I_0| > d \\ \bigcup_{0 \leq t_1 \leq |I_0|} \left(succ^*(\varphi_2, \rho^{t_1}) \wedge \bigwedge_{0 \leq t_2 < t_1} succ^*(\varphi_1, \rho^{t_2}) \right) \cup & \\ \left(\left(\bigwedge_{0 \leq t_1 < |I_0|} succ^*(\varphi_1, \rho^{t_1}) \right) \wedge \varphi_1 \, \mathbf{U}_{\leq d - |I_0|} \, \varphi_2 \right) & \text{otherwise} \end{array}$$

- $succ(\varphi_1 \, \mathbf{R}_{\leq d} \, \varphi_2, \rho) =$

$$\begin{array}{ll} \left(\bigwedge_{0 \leq t_1 \leq d} succ^*(\varphi_2, \rho^{t_1}) \right) \cup & \\ \bigcup_{0 \leq t_1 \leq d} \left(succ^*(\varphi_1, \rho^{t_1}) \wedge \bigwedge_{0 \leq t_2 \leq t_1} succ^*(\varphi_2, \rho^{t_2}) \right) & \text{if } |I_0| > d \\ \bigcup_{0 \leq t_1 < |I_0|} \left(succ^*(\varphi_1, \rho^{t_1}) \wedge \bigwedge_{0 \leq t_2 \leq t_1} succ^*(\varphi_2, \rho^{t_2}) \right) \cup & \\ \left(\left(\bigwedge_{0 \leq t_1 < |I_0|} succ^*(\varphi_2, \rho^{t_1}) \right) \wedge \varphi_1 \, \mathbf{R}_{\leq d - |I_0|} \, \varphi_2 \right) & \text{otherwise} \end{array}$$

Note that for a formula-fine timed observation sequence this definition collapses into the previous one.

Lemma 4.12 *Let φ is an MITL$_{\leq}$ formula and $\rho = ((a_0, a_1, \ldots), (I_0, I_1, \ldots))$ a timed observation sequence. Then $\rho \models \varphi \Leftrightarrow \exists \psi \in succ^*(\varphi, \rho)(\rho^{|I_0|} \models \psi)$.*

Proof The proof follows the same structure as before. Arguments based on the fact that ρ is φ-fine are replaced by induction hypothesis using the definition of $succ^*(\varphi_1 \, \mathbf{U}_{\leq d} \, \varphi_2, \rho)$ and $succ^*(\varphi_1 \, \mathbf{R}_{\leq d} \, \varphi_2, \rho)$. Despite some technicalities, the proof is quite simple and we leave it as an exercise for an interested reader.

■

Although function $succ^*$ removes the requirement of formula-fineness, its practical use is hampered by the potential presence of infinite conjunctions and disjunctions in the resulting formula. A formula with such operators is, at least syntactically, no longer a member of MITL$_{\leq}$. However, for certain fragments of MITL$_{\leq}$ the presence of such operators can be eliminated using the semantics of MITL$_{\leq}$. We actually conjecture that any formula resulting from computation of function $succ^*$ has an equivalent formula in MITL$_{\leq}$. However, a resolution of our conjecture is left as an open problem.

Alternatively, the problem of online monitoring of an MITL$_{\leq}$ formula over a timed observation sequence can be solved using Geilen's algorithm for deciding MITL$_{\leq}$ (see [17], Chapter 8). Given a formula φ, the algorithm builds a timed automata that accepts precisely those timed observation sequences which violate φ. Online monitoring is then realized as a computation of this automata.

In summary, the approach based on formula rewriting can be percieved as a framework for solving some instances of the monitoring problem. Its advantage is that it avoids a potentially costly creation of timed automata. Nevertheless, some instances of the monitoring problem are out of reach of this approach. To handle these instances algorithms based on the automata-theoretic approach are the most efficient method known so far.

4.3 Partial observability and measurement inaccuracies

In this section we address two issues. One is related to monitoring of systems in general and one is exclusively related to monitoring of real-time systems. In the remainder we assume that $\mathcal{A}_\varphi$ is a *tableau automata* (see [17], page 169) accepting precisely those observation sequences which satisfy φ.

Partial observability

The *partial observability* issue occurs when a system specification, in our case an MITL$_{\leq}$ formula, describes a behaviour of unobservable parts of the system being verified. States of the tableau automata are labeled by propositions. To enable partial observability, we simply abstract away from all unobservable proposition, that is we create a projection of the labeling function. Then, given a timed observation sequence ρ and an MITL$_{\leq}$ formula φ, we create tableau automata $\mathcal{A}_{\varphi}$ and $\mathcal{A}_{\neg\varphi}$ and modify their labeling function accordingly. Based on results of computations of $\mathcal{A}_{\varphi}$ and $\mathcal{A}_{\neg\varphi}$ over ρ we state that the following holds.

- $\mathcal{A}_{\varphi}$ accepts $\rho \implies \rho$ can be refined and its observation sequence can be augmented with unobservable propositions such that the resulting timed observation sequence satisfies φ.
- $\mathcal{A}_{\varphi}$ does not accept $\rho \implies$ no matter how ρ is refined and its observation sequence is augmented with unobservable propositions the resulting timed observation sequence always violate φ.
- $\mathcal{A}_{\neg\varphi}$ accepts $\rho \implies \rho$ can be refined and its observation sequence can be augmented with unobservable propositions such that the resulting timed observation sequence violates φ.
- $\mathcal{A}_{\neg\varphi}$ does not accept $\rho \implies$ no matter how ρ is refined and its observation sequence is augmented with unobservable propositions the resulting timed observation sequence always violate φ.

Measurement inaccuracies

Measurement inaccuracies are inherent for any measurement of real-time system. In the context of the MITL$_{\leq}$, both issues can be dealt with using the automata-theoretic approach. A specification of a real-time system may contain timing constraints. To verify such a specification it is essential to measure the actual timing of changes of system state with the highest possible precision. However, a higher precision introduces a further complexity and it is infeasible to measure the timing precisely for real-time systems. To avoid this problem, one can abstract away from the precise timing and use lower and upper bounds approximating the actual timing. Therefore a change occuring at time close to $\sqrt{2}$ could be described as a change occuring sometime between time 1 and 2. In other words, we create an over-approximation of the actual run.

The only obstacle to the application of this simple idea is that the description of the behaviour being monitored is no longer a single timed observation sequence but an infite set of timed observation sequences. The key to make use of the timing abstraction is to represent the infite set of timed observation sequences as a timed automata. This is easy and can be done even on-the fly.

Then given a timed automata $\mathcal{A}_\rho$ describing the behaviour being monitored and an MITL$_{\leq}$ formula φ, we test for the emptiness both the intersection $\mathcal{A}_\rho \cap \mathcal{A}_\varphi$ and the intersection $\mathcal{A}_\rho \cap \mathcal{A}_{\neg\varphi}$. Based on results of the test we state that the following holds.

- $\mathcal{A}_\varphi \cap \mathcal{A}_\rho = \emptyset \implies$ any potential system behaviour violates φ
- $\mathcal{A}_\varphi \cap \mathcal{A}_\rho \neq \emptyset \implies$ there is a potential system behaviour satisfying φ
- $\mathcal{A}_{\neg\varphi} \cap \mathcal{A}_\rho = \emptyset \implies$ every potential system behaviour satisfies φ
- $\mathcal{A}_{\neg\varphi} \cap \mathcal{A}_\rho \neq \emptyset \implies$ there is a potential system behaviour violating φ

Chapter 5

Conclusions

"As for the future, your task is not to foresee it, but to enable it."
- Antoine de Saint-Exupéry

In this thesis, we have studied usage of monitoring for the purpose of the verification. In particular, we have investigated techniques for the formal verification of observed behaviours. As our specification language we have chosen two temporal logics, *Linear Temporal Logic* (LTL) and a fragment of *Metric Interval Temporal Logic* (MITL$_{\leq}$). Although the latter can be viewed as an extension of the former, the techniques for deciding each of them differ. Therefore we make conclusions for each of them separetely.

As far as monitoring techniques for LTL are concerned, we have presented an asymptotically optimal algorithm based on the formula rewriting approach that can be used both for offline and online monitoring. Further we have estabilished a link to automata-theoretic approach. Lastly, we have extended the algorithm to provide for partial observability, that is monitoring of a system with unobservable parts.

In the context of MITL$_{\leq}$, we have—analogously to LTL—presented an asymptotically optimal algorithm based on the formula rewriting approach. Although the algorithm provides for offline monitoring, its applicability to online monitoring is limited. Consequently, we have pointed out situations for which the algorithm works. On top of that, we have referred to an algorithm, based on the automata-theoretic approach, which provides for online monitoring. Finally, we have described extensions of the latter algorithm that cope with partial observability and measurement inaccuracies.

Future work

This work indicates several directions for future research. First and foremost, it is very important to resolve our conjecture about the function $succ^*$ from the end of Chapter 4. We believe that its resolution could provide a deeper insight into MITL$_{\leq}$. Furthermore, its positive resolution would

extend the applicability of our algorithm. Next, it would be interesting to estabilish a link between our algorithm for $\text{MITL}_{\leq}$ and the algorithm based on automata-theoretic approach. Lastly, we plan to thoroughly evaluate performance of our algorithms.

Bibliography

[1] Rajeev Alur, Tomas Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. In *Symposium on Principles of Distributed Computing*, pages 139–152, 1991.

[2] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. In *Proc. 5th Logic in Computer Science (LICS)*, pages 390–401, 1990.

[3] Rajeev Alur and Thomas A. Henzinger. Logics and Models of Real-Time: A Survey. In *Real Time: Theory in Practice*, volume 600, pages 74–106. Springer-Verlag, 1991.

[4] Jiří Barnat, Luboš Brim, Ivana Černá, and Pavel Šimeček. DiVinE the distributed verification environment. In *4th International Workshop on Parallel and Distributed Methods in verifiCation (PDMC)*, Lisbon, Portuga, July 2005.

[5] Jiří Barnat, Luboš Brim, and Jitka Stříbrná. Distributed LTL model-checking in SPIN. In Matthew B. Dwyer, editor, *8th International SPIN Workshop*, number 2057 in Lecture Notes in Computer Science, pages 200–216. Springer, 2001.

[6] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, number 3185 in LNCS, pages 200–236. Springer–Verlag, September 2004.

[7] Boris Beizer. *Software Testing Techniques*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[8] Luboš Brim, Ivana Černá, Pavel Krčál, and Radek Pelánek. Distributed LTL model checking based on negative cycle detection. In *Proc. Foundations of Software Technology and Theoretical Computer Science*, volume 2245 of *LNCS*, pages 96–107. Springer, 2001.

[9] Luboš Brim, Ivana Černá, Pavel Moravec, and Jiří Šimša. Accepting predecessors are better than back edges in distributed ltl model-checking. In *Formal Methods in Computer-Aided Design (FMCAD)*, volume 3312 of *LNCS*, pages 352–366. Springer, 2004.

[10] Julius Richard Büchi. On a decision method in restricted second order arithmetic. In *Proc. International Congress on Logic, Methodology and Philosophy Science*, pages 1–11. Stanford university Press, 1960.

[11] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, 1986.

[12] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 1999.

[13] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms*. MIT Press, 2001.

[14] David A. Duffy. *Principles of automated theorem proving*. John Wiley & Sons, Inc., New York, NY, USA, 1991.

[15] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Property specification patterns for finite-state verification. In Mark Ardis, editor, *Proceedings 2nd Workshop on Formal Methods in Software Practice (FMSP-98)*, pages 7–15, New York, 1998. ACM Press.

[16] Marc Geilen. On the construction of monitors for temporal logic properties. In *Workshop on Runtime Verification (RV'2001)*, volume 55 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2001.

[17] Marc Geilen. *Formal Techniques for Verification of Complex Real-time Systems*. PhD thesis, Eindhoven University of Technology, 2002.

[18] Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification Testing and Verification*, pages 3–18, Warsaw, Poland, 1995. Chapman & Hall.

[19] Gerard J. Holzmann. *The SPIN model checker: Primer and reference manual*. Addison Wesley, 2004.

[20] Saul A. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[21] Orna Kupferman and Moshe Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, 2001.

[22] Orna Kupferman, Moshe Y. Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[23] Leslie Lamport. Proving correctness of multiprocess programs. *IEEE Transactions Software Engineering*, 3(2):125–143, 1977.

[24] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, Berlin, January 1992.

[25] Nicolas Markey and Philippe Schnoebelen. Model checking a path (preliminary report). In *Proc. Concurrency Theory (CONCUR'2003), Marseille, France*, volume 2761 of *Lecture Notes in Computer Science*, pages 251–265. Springer, August 2003.

[26] A. Prasad Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6(5):495–512, 1994.

[27] Emanuel Sperner. Ein satz über untermengen einer endlichen menge. *Math. Z.*, 27:544–548, 1928.

[28] Prasanna Thati and Grigore Rosu. Monitoring algorithms for metric temporal logic specifications. *Electronic Notes in Theoretical Computer Science*, 113:145–162, 2005.

[29] Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. *Lecture Notes in Computer Science*, 1043:238–266, 1996.

[30] Moshe Y. Vardi and Pierre Wolper. Automata theoretic techniques for modal logics of programs: (extended abstract). In *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 446–456, New York, NY, USA, 1984. ACM Press.

Appendix A

Description of the appended CD

The appended CD contains source files of this thesis as well as a prototype implementation of algorithms from Section 3 and 4.

In the directory `/tex` there are files `chap01.tex` through `chap05.tex`, and `main.tex` that contain LaTeX source of this thesis. The bibliography can be found in a separate file `thesis.bib` and the style and extra fonts from this thesis are in the subdirectory `/misc`.

The directory `/prototype` contains a prototype implementation of the following algorithms. The algorithm which uses the function $succ$ and given an LTL formula φ creates a Buchi automata accepting the language $\{w \; \varphi \models w\}$ and Geilen's algorithm which given an MITL$_{\leq}$ formula creates a timed automata accepting precisely those timed observation sequence which satisfy the formula. The implementation of the latter algorithm is a work of Gerd Behrman. Both algorithms are implemented in an object-oriented language Ruby. Ruby interpreter for Windows and Linux can be found in the directory `/install`.