Theorem Proving for Metric Temporal Logic over the Naturals

Ullrich Hustadt[1], Ana Ozaki[2], and Clare Dixon[1]

[1] Department of Computer Science, University of Liverpool, UK
{cldixon,uhustadt}@liverpool.ac.uk
[2] Center for Advancing Electronics Dresden (cfaed), TU Dresden
Ana.Ozaki@tu-dresden.de

Abstract We study translations from Metric Temporal Logic (MTL) over the natural numbers to Linear Temporal Logic (LTL). In particular, we present two approaches for translating from MTL to LTL which preserve the ExpSpace complexity of the satisfiability problem for MTL. In each of these approaches we consider the case where the mapping between states and time points is given by (1) a strict monotonic function and by (2) a non-strict monotonic function (which allows multiple states to be mapped to the same time point). Our translations allow us to utilise LTL solvers to solve satisfiability and we empirically compare the translations, showing in which cases one performs better than the other.

1 Introduction

Linear and branching-time temporal logics have been used for the specification and verification of reactive systems. In linear-time temporal logic [22,11] we can, for example, express that a formula ψ holds now or at some point in the future using the formula $\Diamond\psi$ (ψ holds eventually). However, some applications require not just that a formula ψ will hold eventually but that it holds within a particular time-frame, for example, between 3 and 7 moments from now.

To express such constraints, a range of Metric Temporal Logics (MTL) have been proposed [3,4], considering different underlying models of time and operators allowed. MTL has been used to formalise vehicle routing problems [17], monitoring of algorithms [27] and cyber-physical systems [1], among others [15]. A survey about MTL and its fragments can be found in [20]. It is known that MTL over the reals is undecidable, though, decidable fragments have been investigated [6,2,5].

Here we consider MTL with pointwise semantics over the natural numbers, following [3], where each state in the sequence is mapped to a time point on a time line isomorphic to the natural numbers. In this instance of MTL, temporal operators are annotated with intervals, which can be finite or infinite. For example, $\Diamond_{[3,7]}$ means that p should hold in a state that occurs in the interval $[3,7]$ of time, while $\Box_{[2,\infty)}p$ means that p should hold in all states that occur at least 2 moments from now. In contrast to LTL, where the time difference from one state to the next is always 1, in MTL, time is allowed to irregularly 'jump' from one state to the next. For example, using $\bigcirc_{[2,2]}p$ we can state that the time difference from the current state to the next state is 2.

Furthermore, following Alur and Henzinger [3], the mapping between states and time points is given by a (weakly) monotonic function, which allows multiple states to be mapped to the same time point. Underlying this semantics is the so-called *digital-clock assumption*: Different states that are associated with the same discrete clock record events happening between successive clock ticks. Similarly, if no events occur over one or more successive clock ticks, no state will be associated with those clock ticks. In this work, we also consider the semantics where the mapping between states and time points is given by a strictly monotonic function, which forces time to progress from one state to another.

We provide two approaches for translating from MTL to LTL: in the first approach we introduce a fresh propositional variable that we call 'gap', which is used to encode the 'jumps' between states, as mentioned above; the second approach is inspired by [3], where fresh propositional variables encode time differences between states. In each approach we consider the case where the mapping between states and time points is given by

1. a strict monotonic function and by
2. a non-strict monotonic function (which allows multiple states to be mapped to the same time point).

All translations are polynomial w.r.t. the largest constant occurring in an interval (although exponential in the size of the MTL formula due to the binary encoding of the constants). Since the satisfiability problem for LTL is PSpace-complete [25], our translations preserve the ExpSpace complexity of the MTL satisfiability problem over the natural numbers [3].

Using these translations from MTL to LTL, we apply four temporal solvers, one resolution based [16], one tableau based [13], one based on model checking [7], and the other based on labelled superposition with partial model guidance [18]; to investigate the properties of the resulting formulae experimentally. To the best of our knowledge, there are no implementations of solvers for MTL with pointwise discrete semantics. In particular, our contributions are:

- translations from MTL to LTL which preserve the ExpSpace complexity of the MTL satisfiability problem;
- an experimental analysis of the behaviour of LTL solvers on the resulting formulae;
- to exemplify which kind of problems can be solved using MTL we also provide encodings of the classical Multiprocessor Job-Shop Scheduling problem [14,8] into MTL.

In the following we provide the syntax and semantics of LTL and MTL (Section 2), show our translations from MTL to LTL (Sections 3 and 4) and experimental results (Section 5). We then show how one can encode the Multiprocessor Job-Shop Scheduling problem into MTL with strict and non-strict semantics (Section 6) and present experimental results (Section 7).

2 Preliminaries

We briefly state the syntax and semantics of LTL and MTL. Let $\mathcal{P}$ be a (countably infinite) set of propositional variables. Well formed formulae in LTL are formed according to the rule:

$$\varphi, \psi \quad := \quad p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid \bigcirc\varphi \mid (\varphi \mathcal{U} \psi)$$

where $p \in \mathcal{P}$. We often omit parentheses if there is no ambiguity. We denote by $\bigcirc^c$ a sequence of c next operators, i.e., $\bigcirc^0\varphi = \varphi$ and $\bigcirc^{n+1}\varphi = \bigcirc\bigcirc^n\varphi$, for every $n \in \mathbb{N}$.

An *LTL model* or *state sequence* σ over $(\mathbb{N}, <)$ is an infinite sequence of states $\sigma_i \subseteq \mathcal{P}$, $i \in \mathbb{N}$. The semantics of LTL is defined as follows.

$$\begin{array}{ll}
(\sigma, i) \models p & \text{iff } p \in \sigma_i \\
(\sigma, i) \models (\varphi \wedge \psi) & \text{iff } (\sigma, i) \models \varphi \text{ and } (\sigma, i) \models \psi \\
(\sigma, i) \models \neg\varphi & \text{iff } (\sigma, i) \not\models \varphi \\
(\sigma, i) \models \bigcirc\varphi & \text{iff } (\sigma, i+1) \models \varphi \\
(\sigma, i) \models (\varphi \mathcal{U} \psi) & \text{iff } \exists k \geq i : (\sigma, k) \models \psi \text{ and } \ \forall j,\ i \leq j < k : (\sigma, j) \models \varphi
\end{array}$$

Further connectives can be defined as usual: $\mathbf{true} \equiv p \vee \neg p$, $\mathbf{false} \equiv \neg(\mathbf{true})$, $\Diamond\varphi \equiv \mathbf{true}\mathcal{U}\varphi$ and $\Box\varphi \equiv \neg\Diamond\neg\varphi$. MTL formulae are constructed in a way similar to LTL, with the difference that temporal operators are now bounded by an interval I with natural numbers as end-points or ∞ on the right side. Note that since we work with natural numbers as end-points we can assume w.l.o.g that all our intervals are of the form $[c_1, c_2]$ or $[c_1, \infty)$, where $c_1, c_2 \in \mathbb{N}$. Well formed formulae in MTL are formed according to the rule:

$$\varphi, \psi \quad := \quad p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid \bigcirc_I\varphi \mid (\varphi \mathcal{U}_I \psi)$$

where $p \in \mathcal{P}$. A *timed state sequence* $\rho = (\sigma, \tau)$ over $(\mathbb{N}, <)$ is a pair consisting of an infinite sequence σ of states $\sigma_i \subseteq \mathcal{P}$, $i \in \mathbb{N}$, and a function $\tau : \mathbb{N} \to \mathbb{N}$ that maps every i corresponding to the i-th state to a time point $\tau(i)$ such that $\tau(i) < \tau(i+1)$. A *non-strict* timed state sequence $\rho = (\sigma, \tau)$ over $(\mathbb{N}, <)$ is a pair consisting of an infinite sequence σ of states $\sigma_i \subseteq \mathcal{P}$, $i \in \mathbb{N}$, and a function $\tau : \mathbb{N} \to \mathbb{N}$ that maps every i corresponding to the i-th state to a time point $\tau(i)$ such that $\tau(i) \leq \tau(i+1)$. We assume w.l.o.g. that $\tau(0) = 0$. The semantics of MTL is defined as follows (we omit propositional cases, which are as in LTL).

$$\begin{array}{ll}
(\rho, i) \models p & \text{iff } p \in \sigma_i \\
(\rho, i) \models (\varphi \wedge \psi) & \text{iff } (\rho, i) \models \varphi \text{ and } (\rho, i) \models \psi \\
(\rho, i) \models \neg\varphi & \text{iff } (\rho, i) \not\models \varphi \\
(\rho, i) \models \bigcirc_I\varphi & \text{iff } (\rho, i+1) \models \varphi \text{ and } \tau(i+1) - \tau(i) \in I \\
(\rho, i) \models (\varphi \mathcal{U}_I \psi) & \text{iff } \exists k \geq i : \tau(k) - \tau(i) \in I \text{ and } (\rho, k) \models \psi \\
 & \quad \text{and } \forall j,\ i \leq j < k : (\rho, j) \models \varphi
\end{array}$$

Further connectives can be defined as usual: $\Diamond_I\varphi \equiv \mathbf{true}\mathcal{U}_I\varphi$ and $\Box_I\varphi \equiv \neg\Diamond_I\neg\varphi$. To transform an MTL formula into Negation Normal Form, one uses the constrained dual until $\tilde{\mathcal{U}}_I$ operator [20], defined as $(\varphi\tilde{\mathcal{U}}_I\psi) \equiv \neg(\neg\varphi\mathcal{U}_I\neg\psi)$.

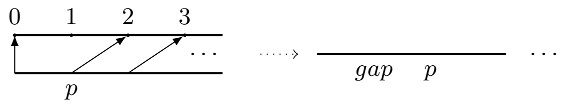

Figure 1: Example illustrating Definition 1

An MTL formula φ is in *Negation Normal Form (NNF)* iff the negation operator ($\neg$) occurs only in front of propositional variables. One of the differences between MTL and LTL is that in LTL we have the equivalence $\neg(\bigcirc p) \equiv \bigcirc \neg p$, whereas in MTL $\neg(\bigcirc_{[2,2]} p) \not\equiv \bigcirc_{[2,2]} \neg p$. If $\neg(\bigcirc_{[2,2]} p)$ then either p does not occur in the next state or the next state does not occur with time difference 2. We can express this as follows: $\neg(\bigcirc_{[2,2]} p) \equiv \bigcirc_{[2,2]} \neg p \vee \bigcirc_{[0,1]} \mathbf{true} \vee \bigcirc_{[3,\infty)} \mathbf{true}$.

An MTL formula φ is in *Flat Normal Form (FNF)* iff it is of the form $p_0 \wedge \bigwedge_i \Box_{[0,\infty)}(p_i \rightarrow \psi_i)$ where p_0, p_i are propositional variables or **true** and ψ_i is either a formula of propositional logic or it is of the form $\bigcirc_I \psi_1$, $\psi_1 \mathcal{U}_I \psi_2$ or $\psi_1 \tilde{\mathcal{U}}_I \psi_2$ where ψ_1, ψ_2 are formulae of propositional logic.

One can transform an MTL formula into FNF by renaming subformulae with nested operators, as in [10,29]. For example, assume that we are given the following MTL formula: $\bigcirc_{[2,3]}(\neg \Box_{[1,2]} q)$. We first transform our formula into NNF and obtain: $\bigcirc_{[2,3]}(\Diamond_{[1,2]} \neg q)$. We then transform it into FNF: $p_0 \wedge \Box_{[0,\infty)}(p_0 \rightarrow \bigcirc_{[2,3]} p_1) \wedge \Box_{[0,\infty)}(p_1 \rightarrow \Diamond_{[1,2]} \neg q)$. The transformations into NNF and FNF are satisfiability preserving and can be performed in polynomial time.

3 From MTL to LTL: encoding 'gaps'

Assume that our MTL formulae are in NNF and FNF. The main idea for our proof is to map each timed state sequence $\rho = (\sigma, \tau)$ to a state sequence σ' such that $\rho = (\sigma, \tau)$ is a model of an MTL formula if, and only if, σ' is a model of our LTL translation. We first present our translation using the strict semantics and then show how to adapt it for the non-strict semantics, where multiple states are allowed to be mapped to the same time point.

Strict Semantics We translate MTL formulae for discrete time models into LTL using a new propositional variable *gap*. $\neg gap$ is true in those states σ'_j of σ' such that there is $i \in \mathbb{N}$ with $\tau(i) = j$ and *gap* is true in all other states of σ'. We now define our mappings between MTL and LTL models.

Definition 1. *Given a timed state sequence $\rho = (\sigma, \tau)$, we define $\sigma' = \sigma'_0 \sigma'_1 \ldots$, where σ'_j is as follows:*

$$\sigma'_j = \begin{cases} \sigma_i & \text{if there is } i \in \mathbb{N} \text{ such that } \tau(i) = j; \\ \{gap\} & \text{otherwise.} \end{cases}$$

Figure 1 illustrates the mapping given by Definition 1. For instance, if $\rho = (\sigma, \tau)$ is the timed state sequence on the left side of Figure 1(a) then $(\rho, 0) \models$

MTL	Strict Gap Translation
$(\bigcirc_{[0,\infty)}\alpha)^\sharp$	$(\bigcirc_{[1,\infty)}\alpha)^\sharp$
$(\bigcirc_{[c_1,\infty)}\alpha)^\sharp$	$(\bigwedge_{1\leq k<c_1}\bigcirc^k gap)\wedge\bigcirc^{c_1}(gap\mathcal{U}(\alpha\wedge\neg gap))$
$(\bigcirc_{[c_1,c_2]}\alpha)^\sharp$	$\bigvee_{c_1\leq l\leq c_2}(\bigcirc^l(\neg gap\wedge\alpha)\wedge\bigwedge_{1\leq k<l}\bigcirc^k gap)$
$(\bigcirc_{[0,0]}\alpha)^\sharp$	**false**
$(\bigcirc_{[0,c_2]}\alpha)^\sharp$	$(\bigcirc_{[1,c_2]}\alpha)^\sharp$
$(\alpha\mathcal{U}_{[0,\infty)}\beta)^\sharp$	$(gap\vee\alpha)\mathcal{U}(\neg gap\wedge\beta)$
$(\alpha\mathcal{U}_{[c_1,\infty)}\beta)^\sharp$	$(\bigwedge_{0\leq k<c_1}\bigcirc^k(gap\vee\alpha))\wedge\bigcirc^{c_1}((gap\vee\alpha)\mathcal{U}(\neg gap\wedge\beta))$
$(\alpha\mathcal{U}_{[c_1,c_2]}\beta)^\sharp$	$\bigvee_{c_1\leq l\leq c_2}(\bigcirc^l(\neg gap\wedge\beta)\wedge\bigwedge_{0\leq k<l}\bigcirc^k(gap\vee\alpha))$
$(\alpha\mathcal{U}_{[0,0]}\beta)^\sharp$	$\neg gap\wedge\beta$
$(\alpha\mathcal{U}_{[0,c_2]}\beta)^\sharp$	$(\neg gap\wedge\beta)\vee(\alpha\mathcal{U}_{[1,c_2]}\beta)^\sharp$

Table 1: Strict Gap Translation from MTL to LTL, where α,β are propositional formulae and $c_1,c_2>0$.

$\bigcirc_{[2,3]}p$. As shown in Table 1, we translate $\bigcirc_{[2,3]}p$ into: $\bigvee_{2\leq l\leq 3}(\bigcirc^l(\neg gap\wedge p)\wedge\bigwedge_{1\leq k<l}\bigcirc^k gap)$.

Note that the state sequence represented on the right side of Figure 1 is a model of the translation. Since *gap* is a propositional variable not occurring in σ, the time points mapped by the image of τ do not contain *gap*.

Definition 2. *Given a state sequence σ' such that $(\sigma',0)\models\neg gap\wedge\Box(\Diamond\neg gap)$, we inductively define $\rho=(\sigma_0,\tau(0))(\sigma_1,\tau(1))\ldots$, where $(\sigma_0,\tau(0))=(\sigma'_0,0)$ and, for $i,j,k\in\mathbb{N}$ and $i>0$, $(\sigma_i,\tau(i))$ is as follows:*

$$\sigma_i=\sigma'_j \text{ and } \tau(i)=j \quad \text{if } j>\tau(i-1),\ gap\notin\sigma'_j \text{ and for all } k,\ \tau(i-1)<k<j,\ gap\in\sigma'_k.$$

As σ' is such that $(\sigma',0)\models\neg gap\wedge\Box(\Diamond\neg gap)$, for each $i\in\mathbb{N}$ we have $\tau(i)\in\mathbb{N}$. Also, for $i>0$, $\tau(i)>\tau(i-1)$ and, so, $\tau:\mathbb{N}\to\mathbb{N}$ is well defined.

Example Assume that we are given the following MTL formula in NNF and FNF: $\varphi=p_0\wedge\Box_{[0,\infty)}(p_0\to\bigcirc_{[2,3]}p_1)\wedge\ \Box_{[0,\infty)}(p_1\to\Diamond_{[1,2]}\neg q)$. Using Table 1, we translate φ into LTL as follows (recall that $\Diamond_I\psi\equiv\mathbf{true}\mathcal{U}_I\psi$):

$$\varphi^\sharp=p_0\wedge\Box_{[0,\infty)}(p_0\to(\neg gap\wedge(\bigvee_{2\leq l\leq 3}(\bigcirc^l(\neg gap\wedge p_1)\wedge\bigwedge_{1\leq k<l}\bigcirc^k gap)) \\ \wedge\Box_{[0,\infty)}(p_1\to(\neg gap\wedge(\bigvee_{1\leq l\leq 2}(\bigcirc^l(\neg gap\wedge\neg q))))$$

We are ready for Theorem 1, which states the correctness of our translation from MTL to LTL using 'gap's.

Theorem 1 *Let $\varphi=p_0\wedge\bigwedge_i\Box_{[0,\infty)}(p_i\to\psi_i)$ be an MTL formula in NNF and FNF. Let $\varphi^\sharp=p_0\wedge\bigwedge_i\Box(p_i\to(\neg gap\wedge\psi_i^\sharp))$ be the result of replacing each ψ_i in*

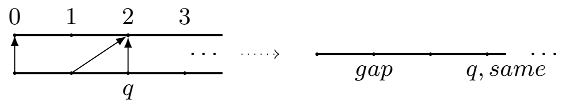

Figure 2: Example illustrating Definition 3

φ by $\psi_i^\sharp$ as in Table 1. Then, φ is satisfiable if, and only if, $\varphi^\sharp \wedge \neg gap \wedge \Box(\Diamond\neg gap)$ is satisfiable.

Proof (Sketch). Assume φ is satisfied by a timed state sequence $\rho = (\sigma, \tau)$. We then use Definition 1 to define a state sequence σ' and show with a structural inductive argument that σ' is a model of $\varphi^\sharp \wedge \neg gap \wedge \Box(\Diamond\neg gap)$. For the other direction, we assume that $\varphi^\sharp \wedge \neg gap \wedge \Box(\Diamond\neg gap)$ is satisfied by a state sequence σ' and use Definition 2 to define a timed state sequence ρ. We again use a structural inductive argument to show that ρ is a model of φ. ❑

Non-Strict Semantics We now show how we modify the Gap translation for non-strict timed state sequences. We introduce a fresh propositional variable called 'same'. *same* is true exactly in those states σ'_j of σ' such that there is $i \in \mathbb{N}$ with $\tau(i) = j$ and, for $i > 0$, $\tau(i) = \tau(i-1)$. Note that *same* and *gap* cannot both be true in any state. We say that a state s is a *gap state* if $gap \in s$. We now define our mappings between MTL and LTL models.

Definition 3. *Let $\rho = (\sigma, \tau)$ be a non-strict timed state sequence. We define $\sigma' = \sigma'_0\sigma'_1\ldots$ by initially setting $\sigma' = \sigma$ and then modifying σ' with the two following steps:*
1. *For $i > 0$, if $\tau(i) - \tau(i-1) = 0$ then set $\sigma'_i := \sigma_i \cup \{same\}$;*
2. *For $i, j \geq 0$, if σ'_j is the i-th non-gap state in σ', σ'_{j+1} is a non-gap state and $\tau(i+1) - \tau(i) = k > 1$ then add $k-1$ states of the form $\{gap\}$ between σ'_j and σ'_{j+1}.*

Figure 2 illustrates the mapping given by Definition 3. For instance, if $\rho = (\sigma, \tau)$ is the non-strict timed state sequence on the left side of Figure 1 then $(\rho, 0) \models \Diamond_{[2,2]}q$. As shown in Table 2, we translate $\Diamond_{[2,2]}q$ into: $same\mathcal{U}(\neg same \wedge \bigcirc(same\mathcal{U}(\neg same \wedge \bigcirc((q \wedge \neg gap) \vee \bigcirc(same\mathcal{U}(q \wedge same))))))$. The main distinction from the translation presented in Table 1 is that here we use nested until operators to make progress in our encoding of the time line whenever we find a state with $\neg same$. Note that the state sequence represented on the right side of Figure 1 is a model of the translation (recall that $\Diamond_{[2,2]}q \equiv \mathbf{true}\mathcal{U}_{[2,2]q}$).

Definition 4. *Let σ' be a state sequence such that $(\sigma', 0) \models \neg gap \wedge \neg same \wedge \Box(\Diamond\neg gap) \wedge \Box(\neg same \vee \neg gap) \wedge \Box(gap \rightarrow \bigcirc\neg same)$. We first define $\tau : \mathbb{N} \rightarrow \mathbb{N}$ by setting $\tau(0) = 0$ and, for $i > 0$, $\tau(i)$ is as follows:*

$$\tau(i) = \begin{cases} \tau(i-1) & \text{if } \sigma'_j \text{ is the } i+1\text{-th non-gap state and } same \in \sigma'_j \\ \tau(i-1)+k+1 & \text{otherwise,} \end{cases}$$

MTL	Non-Strict Gap Translation
$(\bigcirc_{[0,\infty)}\alpha)^\sharp$	$(\bigcirc_{[0,0]}\alpha)^\sharp \vee (\bigcirc_{[1,\infty)}\alpha)^\sharp$
$(\bigcirc_{[0,c_2]}\alpha)^\sharp$	$(\bigcirc_{[0,0]}\alpha)^\sharp \vee (\bigcirc_{[1,c_2]}\alpha)^\sharp$
$(\bigcirc_{[0,0]}\alpha)^\sharp$	$\bigcirc(\alpha \wedge \mathit{same})$
$(\alpha\mathcal{U}_{[c_1,\infty)}\beta)^\sharp$	$\alpha \wedge \bigcirc((\alpha \wedge \mathit{same})\mathcal{U}(\neg \mathit{same} \wedge (\alpha\mathcal{U}_{[c_1-1,\infty)}\beta)^\sharp))$
$(\alpha\mathcal{U}_{[0,\infty)}\beta)^\sharp$	$(\mathit{gap} \vee \alpha)\mathcal{U}(\neg \mathit{gap} \wedge \beta)$
$(\alpha\mathcal{U}_{[c_1,c_2]}\beta)^\sharp$	$\alpha \wedge \bigcirc((\alpha \wedge \mathit{same})\mathcal{U}(\neg \mathit{same} \wedge (\alpha\mathcal{U}_{[c_1-1,c_2-1]}\beta)^\sharp))$
$(\alpha\mathcal{U}_{[0,0]}\beta)^\sharp$	$(\beta \wedge \neg \mathit{gap}) \vee (\alpha \wedge \bigcirc((\alpha \wedge \mathit{same})\mathcal{U}(\beta \wedge \mathit{same})))$
$(\alpha\mathcal{U}_{[0,c_2]}\beta)^\sharp$	$(\alpha\mathcal{U}_{[0,0]}\beta)^\sharp \vee (\alpha\mathcal{U}_{[1,c_2]}\beta)^\sharp$

Table 2: Non-Strict Gap Translation from MTL to LTL, using *gap* and *same*, where α, β are propositional logic formulae, $c_1, c_2 > 0$ and $(\bigcirc_{[c_1,\infty)}\alpha)^\sharp$ and $(\bigcirc_{[c_1,c_2]}\alpha)^\sharp$ are as in Table 1.

where $k \geq 0$ is the number of gap states between the i-th and $i+1$-th non-gap states. We now define σ as follows:

$$\sigma_i = \sigma'_j \setminus \{\mathit{same}\}, \textit{ where } \sigma'_j \textit{ is the } i+1\textit{-th non-gap state.}$$

We are ready for Theorem 2, which states the correctness of our translation from MTL to LTL using the variables 'gap' and 'same'.

Theorem 2 *Let $\varphi = p_0 \wedge \bigwedge_i \Box_{[0,\infty)}(p_i \to \psi_i)$ be an MTL formula in NNF and FNF. Let $\varphi^\sharp = p_0 \wedge \bigwedge_i \Box(p_i \to (\neg \mathit{gap} \wedge \psi_i^\sharp))$ be the result of replacing each ψ_i in φ by $\psi_i^\sharp$ as in Table 2. Then, φ is satisfiable if, and only if, $\varphi^\sharp \wedge \neg \mathit{gap} \wedge \neg \mathit{same} \wedge \Box(\Diamond \neg \mathit{gap}) \wedge \Box(\neg \mathit{same} \vee \neg \mathit{gap}) \wedge \Box(\mathit{gap} \to \bigcirc \neg \mathit{same})$ is satisfiable.*

Proof (Sketch). We use Definitions 3 and 4 to map models of φ into models of $\varphi^\sharp \wedge \neg \mathit{gap} \wedge \Box(\Diamond \neg \mathit{gap})$ and vice versa. The correctness of our translation is again given by a structural inductive argument. As mentioned, the main difference w.r.t. to Theorem 1 is that here we use the propositional variable *same* to encode multiple states mapped to the same time point. ❏

4 From MTL to LTL: encoding time differences

Assume that our MTL formulae are in NNF and FNF. Similar to the previous section our proof strategy relies on mapping each timed state sequence $\rho = (\sigma, \tau)$ to a state sequence σ' such that $\rho = (\sigma, \tau)$ is a model of an MTL formula if, and only if, σ' is a model of our LTL translation. We first show a translation under the strict semantics and then we show how to adapt it for the non-strict semantics.

Strict Semantics Let $C-1$ be the greatest number occurring in an interval in an MTL formula φ or 1, if none occur. We say that a timed state sequence $\rho = (\sigma, \tau)$ is *C-bounded*, for a constant $C \in \mathbb{N}$, if $\tau(0) \leq C$ and, for all $i \in \mathbb{N}$, $\tau(i+1) - \tau(i) \leq C$. To map a timed state sequence $\rho = (\sigma, \tau)$ to a state sequence σ' we employ the following result adapted from [4].

Theorem 3 *Let φ be an MTL formula. If there is a timed state sequence $\rho = (\sigma, \tau)$ such that $(\rho, 0) \models \varphi$ then there is a C-bounded timed state sequence ρ_C such that $(\rho_C, 0) \models \varphi$.*

By Theorem 3, w.l.o.g., we can consider only timed state sequences where the time difference from a state to its previous state is bounded by C. Then, we can encode time differences with a set $\Pi_\delta = \{\delta_i^- \mid 1 \leq i \leq C\}$ of propositional variables where each δ_i^- represents a time difference of i w.r.t. the previous state (one could also encode the time difference to the next state instead of the difference from the previous state). We also use propositional variables of the form s_m^n with the meaning that 'the sum of the time differences from the last n states to the current state is m'. For our translation, we only need to define these variables up to sums bounded by $2 \cdot C$. We can now define our mapping from an MTL model to an LTL model[3].

Definition 5. *Given a C-bounded timed state sequence $\rho = (\sigma, \tau)$, we define $\sigma' = \sigma'_0 \sigma'_1 \ldots$ by setting $\sigma'_0 = \sigma_0$ and, for $i > 0$:*

$$\begin{aligned}\sigma'_i = \sigma_i &\cup \{\delta_k^-, s_k^1 \mid \tau(i) - \tau(i-1) = k, 1 \leq k \leq C\} \\ &\cup \ \{s^{j+1}_{\mathsf{min}(l+k, 2\cdot C)} \mid s_k^1 \in \sigma'_i \text{ and } s_l^j \in \sigma'_{i-1}\}\end{aligned}$$

where $1 \leq j < 2 \cdot C$, $1 \leq l \leq 2 \cdot C$ and $1 \leq k \leq C$ (assume variables of the form s_m^n and δ_n^- do not occur in σ).

In Definition 5, if, for example, $\tau(2) - \tau(0) = 4$ then $(\sigma', 2) \models s_4^2$. Intuitively, the variable s_4^2 allow us to group together all the cases where the sum of the time differences from the last 2 states to the current state is 4. This happens when: $\tau(2) - \tau(1) = 3$ and $\tau(1) - \tau(0) = 1$; or $\tau(2) - \tau(1) = 1$ and $\tau(1) - \tau(0) = 3$; or $\tau(2) - \tau(1) = 2$ and $\tau(1) - \tau(0) = 2$.

The next lemma gives the main properties of σ'. First, we need some notation. We use two additional n-ary boolean operators $\oplus_{=1}$ and $\oplus_{\leq 1}$. If $S = \{\varphi_1, \ldots, \varphi_n\}$ is a finite set of LTL formulae, then $\oplus_{=1}(\varphi_1, \ldots, \varphi_n)$, also written $\oplus_{=1}S$, is an LTL formula. Let σ' be a state sequence and $i \in \mathbb{N}$. Then $(\sigma', i) \models \oplus_{=1}S$ iff $(\sigma', i) \models \varphi_j$ for exactly one $\varphi_j \in S$, $1 \leq j \leq n$. Similarly, $(\sigma', i) \models \oplus_{\leq 1}S$ iff $(\sigma', i) \models \varphi_j$ for at most one $\varphi_j \in S$, $1 \leq j \leq n$. By definition of σ' the following lemma is immediate.

Lemma 1. *Let S_C be the conjunction of the following:*

[3] We write $\mathsf{min}(l+k, 2\cdot C)$ for the minimum between $l+k$ and $2 \cdot C$. If the minimum is $2 \cdot C$ then $s^{j+1}_{2 \cdot C}$ means that the sum of the last $j+1$ variables is greater or equal to $2 \cdot C$.

MTL	Strict Time Difference Translation
$(\bigcirc_{[c_1,\infty)}\alpha)^\sharp$	$\bigcirc((\bigvee_{c_1\leq i\leq C}\delta_i^-)\wedge\alpha)$
$(\bigcirc_{[0,\infty)}\alpha)^\sharp$	$\bigcirc\alpha$
$(\bigcirc_{[c_1,c_2]}\alpha)^\sharp$	$\bigcirc((\bigvee_{c_1\leq i\leq c_2}\delta_i^-)\wedge\alpha)$
$(\bigcirc_{[0,c_2]}\alpha)^\sharp$	$(\bigcirc_{[1,c_2]}\alpha)^\sharp$
$(\bigcirc_{[0,0]}\alpha)^\sharp$	**false**
$(\alpha\mathcal{U}_{[c_1,\infty)}\beta)^\sharp$	$\bigvee_{1\leq i\leq c_1}(\bigcirc^i((\bigvee_{c_1\leq j\leq c_1+C}\ s_j^i)\wedge\alpha\mathcal{U}\beta)\wedge(\bigwedge_{0\leq k<i}\bigcirc^k\alpha))$
$(\alpha\mathcal{U}_{[0,\infty)}\beta)^\sharp$	$\alpha\mathcal{U}\beta$
$(\alpha\mathcal{U}_{[c_1,c_2]}\beta)^\sharp$	$\bigvee_{1\leq i\leq c_2}(\bigcirc^i((\bigvee_{c_1\leq j\leq c_2}\ s_j^i)\wedge\beta)\wedge(\bigwedge_{0\leq k<i}\bigcirc^k\alpha))$
$(\alpha\mathcal{U}_{[0,c_2]}\beta)^\sharp$	$\beta\vee(\alpha\mathcal{U}_{[1,c_2]}\beta)^\sharp$
$(\alpha\mathcal{U}_{[0,0]}\beta)^\sharp$	β

Table 3: Strict Time Difference Translation from MTL to LTL where α, β are propositional logic formulae and $c_1, c_2 > 0$.

1. $\bigcirc\Box\oplus_{=1}\Pi_\delta$, *for* $\Pi_\delta = \{\delta_k^- \mid 1\leq k\leq C\}$*;*
2. $\Box(\delta_k^-\leftrightarrow s_k^1)$, *for* $1\leq k\leq C$*;*
3. $\Box\oplus_{\leq 1}\Pi^i$, *for* $1\leq i\leq 2\cdot C$ *and* $\Pi^i=\{s_j^i \mid i\leq j\leq 2\cdot C\}$*;*
4. $\Box((\bigcirc s_k^1\wedge s_l^j)\rightarrow\bigcirc s^{j+1}_{\mathsf{min}(l+k,2\cdot C)})$, *for* $\{s_k^1, s_l^j, s^{j+1}_{\mathsf{min}(l+k,2\cdot C)}\}\subseteq\bigcup_{1\leq i\leq 2\cdot C}\Pi^i$*.*

Given a C*-bounded timed state sequence* $\rho=(\sigma,\tau)$*, let* $\sigma'=\sigma_0'\sigma_1'\ldots$ *be as in Definition 5. Then,* $(\sigma',0)\models S_C$*.*

Point 1 ensures that at all times, the time difference k from the current state to the previous (if it exists) is uniquely encoded by the variable δ_k^-. In Point 2 we have that the sum of the difference of the last state to the current, encoded by s_k^1, is exactly δ_k^-. Point 3 ensures that at all times we cannot have more than one value for the sum of the time differences of the last i states. Finally, Point 4 has the propagation of sum variables: if the sum of the last j states is l and the time difference to the next is k then the next state should have that the sum of the last $j+1$ states is $l+k$. We now define our mapping from an LTL model of S_C to an MTL model (for this mapping, we actually only need Point 1).

Definition 6. *Given a state sequence* $\sigma'=\sigma_0'\sigma_1'\ldots$ *such that* $(\sigma',0)\models S_C$*, we define a* C*-bounded timed state sequence* $\rho=(\sigma,\tau)$ *by setting* $\sigma_i=\sigma_i'\setminus(\Pi_\delta\cup\bigcup_{1\leq j\leq 2C}\Pi^j)$*, for* $i\in\mathbb{N}$*, and:*

$$\tau(i)=\begin{cases}0 & \text{if } i=0\\ \tau(i-1)+k & \text{if } i>0,\ \delta_k^-\in\sigma_i'\end{cases}$$

Note that ρ, in particular, τ, in Definition 6 is well-defined because for every $i\in\mathbb{N}$ there is exactly one k such that $\delta_k^-\in\sigma_i'$. As shown in Table 3, we translate, for example, $\bigcirc_{[2,3]}p$ into $\bigcirc((\delta_2^-\vee\delta_3^-)\wedge p)$. We are ready for Theorem 4, which states the correctness of our translation using time differences.

MTL	Non-Strict Time Difference Translation
$(\bigcirc_{[k_1,\infty)}\alpha)^\sharp$	$\bigcirc((\bigvee_{k_1\leq i\leq C}\delta_i^-)\wedge\alpha)$
$(\bigcirc_{[k_1,k_2]}\alpha)^\sharp$	$\bigcirc((\bigvee_{k_1\leq i\leq k_2}\delta_i^-)\wedge\alpha)$
$(\alpha\mathcal{U}_{[c_1,\infty)}\beta)^\sharp$	$\alpha\wedge\bigcirc\bigvee_{1\leq i\leq c_1}((\alpha\wedge\delta_0^-)\mathcal{U}^i(\neg\delta_0^-\wedge\alpha),(\neg\delta_0^-\wedge(\bigvee_{c_1\leq j\leq c_1+C}\ s_j^i)\wedge\alpha\mathcal{U}\beta))$
$(\alpha\mathcal{U}_{[0,\infty)}\beta)^\sharp$	$\alpha\mathcal{U}\beta$
$(\alpha\mathcal{U}_{[c_1,c_2]}\beta)^\sharp$	$\alpha\wedge\bigcirc\bigvee_{1\leq i\leq c_2}((\alpha\wedge\delta_0^-)\mathcal{U}^i(\neg\delta_0^-\wedge\alpha),(\neg\delta_0^-\wedge(\bigvee_{c_1\leq j\leq c_2}\ s_j^i)\wedge(\alpha\mathcal{U}_{[0,0]}\beta)^\sharp))$
$(\alpha\mathcal{U}_{[0,c_2]}\beta)^\sharp$	$(\alpha\mathcal{U}_{[0,0]}\beta)^\sharp\vee(\alpha\mathcal{U}_{[1,c_2]}\beta)^\sharp$
$(\alpha\mathcal{U}_{[0,0]}\beta)^\sharp$	$\beta\vee(\alpha\wedge\bigcirc((\alpha\wedge\delta_0^-)\mathcal{U}(\beta\wedge\delta_0^-)))$

Table 4: Non-Strict Time Difference Translation from MTL to LTL where α,β are propositional logic formulae, $k_1,k_2\geq 0$ and $c_1,c_2>0$.

Theorem 4 *Let $\varphi = p_0\wedge\bigwedge_i\Box_{[0,\infty)}(p_i\rightarrow\psi_i)$ be an MTL formula in NNF and FNF. Let $\varphi^\sharp = p_0\wedge\bigwedge_i\Box(p_i\rightarrow\psi_i^\sharp)$ be the result of replacing each ψ_i in φ by $\psi_i^\sharp$ as in Table 3. Then, φ is satisfiable if, and only if, $\varphi^\sharp\wedge S_C$ is satisfiable.*

Proof (Sketch). Assume φ is satisfied by a timed state sequence $\rho=(\sigma,\tau)$. We then use Definition 5 to define a state sequence σ' and show with a structural inductive argument that σ' is a model of $\varphi^\sharp\wedge S_C$. For the other direction, we assume that $\varphi^\sharp\wedge S_C$ is satisfied by a state sequence σ' and use Definition 6 to define a timed state sequence ρ. We again use a structural inductive argument to show that ρ is a model of φ. ❑

Example Assume that we are given the following MTL formula in NNF and FNF: $\varphi = p_0\wedge\Box_{[0,\infty)}(p_0\rightarrow\bigcirc_{[2,3]}p_1)\wedge\Box_{[0,\infty)}(p_1\rightarrow\Diamond_{[1,2]}\neg q)$. Using Table 3, we translate φ into LTL as follows:

$$\begin{aligned}\varphi^\sharp = p_0 &\wedge \Box_{[0,\infty)}(p_0\rightarrow(\neg gap\wedge(\bigcirc_{[2,3]}p_1)^\sharp))\\ &\wedge\Box_{[0,\infty)}(p_1\rightarrow(\neg gap\wedge(\Diamond_{[1,2]}\neg q)^\sharp)),\end{aligned}$$

where

$$\begin{aligned}(\bigcirc_{[2,3]}p_1)^\sharp &= \bigcirc((\bigvee_{2\leq i\leq 3}\delta_i^-)\wedge p_1)\\ (\Diamond_{[1,2]}\neg q)^\sharp &= \bigvee_{1\leq i\leq 2}(\bigcirc^i((\bigvee_{1\leq j\leq 2}\ s_j^i)\wedge\neg q))\end{aligned}$$

(recall that $\Diamond_I\psi\equiv\mathbf{true}\mathcal{U}_I\psi$). By Theorem 4, φ is satisfiable iff $\varphi^\sharp\wedge S_4$ is satisfiable, where S_4 is the conjunction of the following:

1. $\bigcirc\Box\oplus_{=1}\mathit{\Pi}_\delta$, for $\mathit{\Pi}_\delta=\{\delta_k^-\mid 1\leq k\leq 4\}$;
2. $\Box(\delta_k^-\leftrightarrow s_k^1)$, for $1\leq k\leq 4$;
3. $\Box\oplus_{\leq 1}\mathit{\Pi}^i$, for $1\leq i\leq 8$ and $\mathit{\Pi}^i=\{s_j^i\mid i\leq j\leq 8\}$;
4. $\Box(\bigcirc s_k^1\wedge s_l^j\rightarrow\bigcirc s_{\mathsf{min}(l+k,8)}^{j+1})$, for $\{s_k^1,s_l^j,s_{\mathsf{min}(l+k,8)}^{j+1}\}\subseteq\bigcup_{1\leq i\leq 8}\mathit{\Pi}^i$.

Non-Strict Semantics We now show how we modify the Time Difference translation for non-strict timed state sequences. We extend the set $\mathit{\Pi}_\delta = \{\delta^-_i \mid 1 \leq i \leq C\}$ of propositional variables representing time differences with δ^-_0, which holds whenever the time difference to the previous state is 0. We say that a state is *non-zero* if the time difference to the previous state is non-zero. The meaning of the variables of the form s^n_m also needs to change, it now indicates that 'the sum of the time differences from the last n *non-zero* states to the current state is m'. As before, for our translation, we only need to define these variables up to sums bounded by $2 \cdot C$. We can now define our mapping from an MTL model to an LTL model.

Given a C-bounded non-strict timed state sequence (σ, τ), we define a state sequence σ' as in Definition 5, with the difference that, whenever $\tau(i) = \tau(i-1)$, we now make δ^-_0 true in σ'_i and copy all variables of the form s^n_m in σ'_{i-1} to σ'_i. Let S'_C be the conjunction of the following:

1. $\bigcirc\Box \oplus_{=1} \mathit{\Pi}_\delta$, for $\mathit{\Pi}_\delta = \{\delta^-_k \mid 0 \leq k \leq C\}$;
2. $\Box(\delta^-_k \leftrightarrow s^1_k)$, for $1 \leq k \leq C$;
3. $\Box \oplus_{\leq 1} \mathit{\Pi}^i$, for $1 \leq i \leq 2 \cdot C$ and $\mathit{\Pi}^i = \{s^i_j \mid i \leq j \leq 2 \cdot C\}$;
4. $\Box((\bigcirc s^1_k \wedge s^j_l) \rightarrow \bigcirc s^{j+1}_{\mathsf{min}(l+k, 2\cdot C)})$, for $\{s^1_k, s^j_l, s^{j+1}_{\mathsf{min}(l+k,2\cdot C)}\} \subseteq \bigcup_{1\leq i\leq 2\cdot C} \mathit{\Pi}^i$;
5. $\Box((\bigcirc\delta^-_0 \wedge s^j_l) \rightarrow \bigcirc s^j_l)$, for $s^j_l \in \bigcup_{1\leq i\leq 2\cdot C} \mathit{\Pi}^i$.

It is easy to see that $(\sigma', 0) \models S'_C$. Note that the only difference from S'_C to S_C, defined in Lemma 1, is Point 5 which propagates the variables of the form s^n_m to the next state if the time difference is zero. The mapping from an LTL model of S'_C to an MTL model is defined in the same way as in Definition 6 (but now k in δ^-_k can be zero). To simplify the notation, in Table 4 we write $\phi\mathcal{U}^n\gamma, \chi$ as a shorthand for $\phi\mathcal{U}(\gamma \wedge \bigcirc(\phi\mathcal{U}^{n-1}\gamma, \chi))$, where $\phi\mathcal{U}^1\gamma, \chi = \phi\mathcal{U}\chi$. Theorem 5 states the correctness of our translation (Table 4) using non-strict time differences. It can be proved with ideas similar to those used in the proof of Theorem 4. The main distinction appears in the translation of the 'until' formulas, where we nest until operators so that we can count n non-zero states and then check whether a variable of the form s^n_m holds (in the strict case all states are non-zero, so in Table 3 we can count these states with next operators).

Theorem 5 *Let $\varphi = p_0 \wedge \bigwedge_i \Box_{[0,\infty)}(p_i \rightarrow \psi_i)$ be an MTL formula in NNF and FNF. Let $\varphi^\sharp = p_0 \wedge \bigwedge_i \Box(p_i \rightarrow \psi^\sharp_i)$ be the result of replacing each ψ_i in φ by $\psi^\sharp_i$ as in Table 4. Then, φ is satisfiable if, and only if, $\varphi^\sharp \wedge S'_C$ is satisfiable.*

Proof (Sketch). We use our modified versions of Definitions 5 and 6 for the non-strict semantics to map models of φ into models of $\varphi^\sharp \wedge S'_C$ and vice versa. The correctness of our translation is again given by a structural inductive argument. As mentioned, the main difference w.r.t. to Theorem 4 is that here we use the propositional variable δ^-_0 to encode multiple states mapped to the same time point. ❑

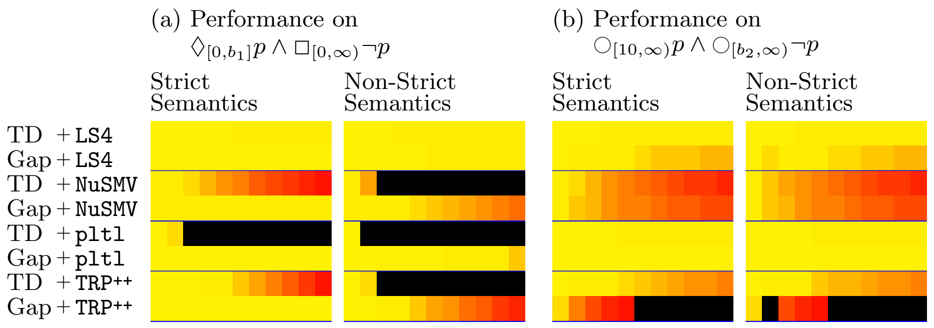

Figure 3: Heat map for the performance of LTL provers on $\theta^1_{b_1}$ and $\theta^2_{b_2}$. Each rectangle represents the runtime of a prover on an encoding of a formula, with runtimes given in colours as follows:

🟨 $<$ 0.01 sec	🟨 $>$ 0.01 sec, $\leq$ 0.25 sec	🟨 $>$ 0.25 sec, $\leq$ 0.50 sec	🟧 $>$ 0.50 sec, $\leq$ 1 sec
🟧 $>$ 1 sec, $\leq$ 2 sec	🟧 $>$ 2 sec, $\leq$ 4 sec	🟧 $>$ 4 sec, $\leq$ 8 sec	🟧 $>$ 8 sec, $\leq$ 16 sec
🟧 $>$ 16 sec, $\leq$ 32 sec	🟥 $>$ 32 sec, $\leq$ 64 sec	🟥 $>$ 64 sec, $\leq$ 125 sec	🟥 $>$ 125 sec, $\leq$ 250 sec
🟥 $>$ 250 sec, $\leq$ 500 sec	🟥 $>$ 500 sec, $\leq$1000 sec	⬛ Timeout exceeded	

5 Empirical Evaluation of the Translations

In order to empirically evaluate the translations, we have used them together with four LTL satisfiability solvers, LS4, NuSMV, pltl and TRP++. The last three performed well in the LTL solver comparison by Schuppan and Darmawan [23] while LS4 has been included because of its excellent performance in our experiments.

NuSMV 2.6.0 [19] uses a reduction of the LTL satisfiability problem to the LTL model checking problem [7]. It is then possible to decide the latter problem either using a BDD-based algorithm or a SAT-based algorithm. Here, we use the latter with completeness check enabled which turns NuSMV into a decision procedure for the LTL satisfiability problem. With the pltl [21] system we have used the **graph** method which is based on a one-pass and-or tree tableau calculus [13] and is time complexity optimal for LTL. TRP++ 2.2 [28] is based on an ordered resolution calculus that operates on LTL formulae in a clausal normal form [16]. LS4 [18] is an LTL prover based on labelled superposition with partial model guidance developed by Suda and Weidenbach [26]. It operates on LTL formulae in the same clausal normal form as TRP++.

We focus on formulae where differences between the two translations could lead to differences in the behaviour of solvers on these formulae. In particular, for $(\alpha\mathcal{U}_{[c_1,c_2]}\beta)$ the Strict and Non-Strict Time Difference Translations contain disjunctive subformulae of the form $\bigvee_{c_1\leq j\leq c_2} s^i_j$ that have no equivalence in the Strict and Non-Strict Gap Translations of that formula. Each sum variable s^i_j is also subject to the constraints expressed by S_C. It is a reasonable hypothesis that this will have a detrimental effect on the performance of a solver. On the other hand, for $\bigcirc_{[c_1,\infty)}\alpha$ both Gap Translations contain an eventuality formula $gap\mathcal{U}(\alpha \wedge \neg gap)$ that is not present in the Time Difference Translations of this formula. Here, the hypothesis is that the Time Difference Translations lead to better behaviour of solvers.

To test our two hypotheses, we consider the unsatisfiable parameterised formulae $\theta^1_{b_1} := \Diamond_{[0,b_1]}p \wedge \Box_{[0,\infty)}\neg p$ for values of b_1 between 0 and 10, and $\theta^2_{b_2} := \bigcirc_{[10,\infty)}p \wedge \bigcirc_{[b_2,\infty)}\neg p$ for values of b_2 between 10 and 110 in steps of 10. After transformation to Flat Normal Form, we apply one of the four translations, and run a solver five times on the resulting LTL formula (with a timeout of 1000 CPU seconds), and then determine the median CPU time over those five runs. We refer to that median CPU time as the runtime. The repeated runs are necessary to moderate the fluctuations shown by all provers in the CPU time used to solve a particular formula. The experiments were conducted on a PC with Intel i7-2600 CPU @ 3.40GHz and 16GB main memory.

Figure 3 shows the runtimes in the form of a heat map. Figure 3(a) confirms our hypothesis that for $(\alpha\mathcal{U}_{[c_1,c_2]}\beta)$ the Gap Translations, independent of the semantics, lead to better performance than the Time Difference Translations. Figure 3(b) confirms that the Time Difference Translations lead to better performance on $\bigcirc_{[c_1,\infty)}\alpha$ for `LS4` and `TRP++`, but not for `NuSMV` and `pltl`. The reason are the background theories S_C and S'_C that form part of the Time Difference Translations, most of which turn out not to be relevant to the (un)satisfiability of $(\theta^2_{b_2})^\sharp$. `LS4` and `TRP++` appear to be able to derive a contradiction without too many inferences involving S_C or S'_C, while `NuSMV` and `pltl` do not. If one restricts S_C and S'_C by hand to smaller sets strictly necessary to establish the (un)satisfiability of $(\theta^2_{b_2})^\sharp$, then `NuSMV` and `pltl` also perform better with the Time Difference Translations than with the Gap Translations.

6 An Example: Multiprocessor Job-Shop Scheduling

We consider a generalisation of the classic job-shop scheduling problem, called the Multiprocessor Job-shop Scheduling (MJS) problem [14,8]. The representation provided is based on that in [9]. Here a set of jobs have to be processed on a set of machines running in parallel. Each job requires a number of processor steps to complete (this number may also depend on the machine, i.e., job i may run faster in machine j than in machine l). The question is whether there is a scheduling such that after t time units all jobs will have been processed by the machines.

We first show how one can encode the problem in MTL with the strict semantics and then we show the encoding with the non-strict semantics. Our encodings have the property that: there is a scheduling if and only if there is a model for the resulting MTL formulae. One can use any model of the MTL formulae to create a scheduling satisfying the constraints of the problem.

Strict Semantics Assume we have n jobs $j_1, j_2, \ldots, j_n$ and k machines $m_1, m_2, \ldots, m_k$. Let

- $\mathit{start\_run}_{j_i}$, run_{j_i} and $\mathit{has\_run}_{j_i}$ denote the start, the execution and the end of the execution of job j_i on some machine, respectively;
- $\mathit{start\_run}_{j_i m_l}$ and $\mathit{run}_{j_i m_l}$ denote the start and the execution of job j_i on machine m_l, respectively; and
- $t_{j_i m_l}$ to denote the time taken to run job j_i on machine m_l.

The following equations state that (1) once a job starts running it must start running on one of the machines and that (2) once a job starts running on a machine it must run on that machine (where $\bigwedge_{1\leq i\leq n}$ and $\bigwedge_{1\leq i\leq n, 1\leq l\leq k}$ in front of the formulas is omitted for brevity)

$$\Box(start\_run_{j_i} \rightarrow \textstyle\bigvee_{l=1}^{k} start\_run_{j_i m_l}) \tag{1}$$

$$\Box(start\_run_{j_i m_l} \rightarrow run_{j_i m_l}) \tag{2}$$

Equation (3) states that: if a job is running on one machine then it cannot be running on another (integrity of jobs); and another job cannot be running on the same machine (integrity of machines). By Equation (4), once a job has started it cannot be started again.

$$\Box(run_{j_i m_l} \rightarrow (\textstyle\bigwedge_{p=1,p\neq l}^{k} \neg run_{j_i m_p} \wedge \bigwedge_{q=1,q\neq i}^{n} \neg run_{j_q m_l})) \tag{3}$$

$$\Box(start\_run_{j_i} \rightarrow \bigcirc\Box\neg start\_run_{j_i}) \tag{4}$$

We write $\neg run_{j_i}$ as a short hand for $\bigwedge_{l=1}^{k} \neg run_{j_i m_l}$. We can use (5) to denote that once job j_i is started to run on machine m_l it takes time $t_{j_i m_l}$ and (6) to denote that once job j_i has finished running on machine m_l it will not run again. Further, Equation (7) denotes that job j_i cannot be run until it has started.

$$\Box(start\_run_{j_i m_l} \rightarrow \Box_{[0,t_{j_i m_l}-1]} run_{j_i m_l} \wedge \neg has\_run_{j_i}) \tag{5}$$

$$\Box(start\_run_{j_i m_l} \rightarrow \Box_{[t_{j_i m_l},\infty)}(\neg run_{j_i} \wedge has\_run_{j_i})) \tag{6}$$

$$\Box(\neg run_{j_i} \mathcal{U} start\_run_{j_i}) \tag{7}$$

We assume initially that no jobs have run, i.e., $\bigwedge_{i=1}^{n} \neg has\_run_{j_i}$; and that (8) if a job has not run and is currently not running then it has not run in the next moment.

$$\Box((\neg has\_run_{j_i} \wedge \neg run_{j_i}) \rightarrow \bigcirc\neg has\_run j_i) \tag{8}$$

We can now check whether we can achieve a schedule after at most t time points by adding $\Diamond_{[0,t]} \bigwedge_{i=1}^{n} has\_run_{j_i}$. We can also specify constraints on jobs such as

- $\Box(run_{j_i} \leftrightarrow run_{j_i,m_l})$: job j_i must run on machine m_l;
- $\Diamond(start\_run_{j_i} \rightarrow \Diamond_{[1,\infty)} start\_run_{j_m})$: job j_i must start before job j_m;
- $\Diamond_{[c,d]} start\_run_{j_i}$: job j_i must start at a point within the interval $[c, d]$.

Non-Strict Semantics We again assume we have n jobs $j_1, j_2, \ldots, j_n$ and k machines $m_1, m_2, \ldots, m_k$. Let

- $start\_run_{j_i}$ and $has\_run_{j_i}$ denote the start and the end of job j_i on some machine, respectively;
- m_l denote a state of machine m_l;
- run_{j_i} denote that job j_i is running on some machine; and
- $t_{j_i m_l}$ denote the time taken to run job j_i on machine m_l.

In each state exactly one of the variables of the form m_l is true. Also, in each state at most one job is running, but now we may have multiple states at the same time. Let $\Pi_m = \{m_1, \ldots, m_k\}$ and $\Pi_j = \{run_{j_1}, \ldots, run_{j_n}\}$. The following states the constraints mentioned above (the meaning of $\oplus_{=1}$ and $\oplus_{\leq 1}$ is as described in Section 3):

$$\Box(\oplus_{=1}\Pi_m \wedge \oplus_{\leq 1}\Pi_j) \tag{9}$$

Equation (10) specifies that if a job is running on one machine then it cannot be running on another. Equation (11) states that once a job is started it cannot be started again (where $\bigwedge_{1\leq i\leq n, 1\leq l\leq k}$ and $\bigwedge_{1\leq i\leq n}$ is again omitted).

$$\Box((m_l \wedge run_{j_i}) \rightarrow \textstyle\bigwedge_{l'\neq l} \Box\neg(m_{l'} \wedge run_{j_i})) \tag{10}$$

$$\Box(start\_run_{j_i} \rightarrow \bigcirc\Box\neg start\_run_{j_i}) \tag{11}$$

We use the following

$$\begin{aligned}\Box((start\_run_{j_i} \wedge m_l) \rightarrow (&\Box_{[0,t_{j_i m_l}-1]}(\neg has\_run_{j_i} \wedge (m_l \rightarrow run_{j_i}))\\ &\wedge \Diamond_{[0,t_{j_i m_l}]} has\_run_{j_i}))\end{aligned} \tag{12}$$

to denote that once job j_i started to run on machine m_l it takes time $t_{j_i m_l}$ and (13) to denote that once job j_i has finished running on machine m_l it will not run again. Further, we use $\Box(\neg run_{j_i} \mathcal{U} start\_run_{j_i})$ to state a job j_i cannot be run until it is started and $\Box(\neg has\_run_{j_i} \mathcal{U} start\_run_{j_i})$ to state that a job cannot have run before it starts (another rule above will make sure that $has\_run_{j_i}$ will hold after the run has finished).

$$\Box((start\_run_{j_i} \wedge m_l) \rightarrow \Box_{[t_{j_i m_l}+1,\infty)}(\neg run_{j_i} \wedge has\_run_{j_i})) \tag{13}$$

We assume initially that no jobs have run, i.e., $\bigwedge_{i=1}^{n} \neg has\_run_{j_i}$. We can now check whether we can achieve a schedule after at most t time points by adding $\Diamond_{[0,t]} \bigwedge_{i=1}^{n} has\_run_{j_i}$.

7 Experiments with MJS Problems

We have performed an experimental evaluation of the combination of our translations with `LS4`, `NuSMV`, `pltl` and `TRP++`. Regarding the MJS problems used in the evaluation we made the simplifying assumption that a job j_i, for each i, $1 \leq i \leq n$, takes the same amount of time t_i on whichever machine it is processed on. We can then characterise a MJS problem by stating (i) a *job list* J consisting of a list of durations $(t'_1, \ldots, t'_n)$, (ii) the number k of machines available, and (iii) the time bound t. In equations 5, 6, 12 and 13, for every i, $1 \leq i \leq n$, and every l, $1 \leq l \leq k$, $t_{j_i m_l}$ will be given by t'_{j_i}. The time bound t is used in the formula $\Diamond_{[0,t]} \bigwedge_{i=1}^{n} has\_run_{j_i}$ that expresses the requirement for a schedule that completes all n jobs on k machines in at most t time points.

For our experiments we created 35 MJS problems with number n of jobs between 1 and 4, a duration t'_i of a job between 1 and 4, a number k of machines between 1 and 3 and finally a time bound t between 0 and 4. We then constructed corresponding MTL formulae for both the strict and the non-strict semantics. Each formula was transformed to FNF, translated to LTL using one of the encodings, and each solver run five times on the resulting LTL formula (with a timeout of 1000 CPU seconds), and the median CPU time over those five runs determined. We refer to that median CPU time as the runtime. Figure 4 shows the runtimes in the form of a heat map.

Regarding the formalisation of MJS problems in the strict semantics, we see in Figure 4 that for every prover the Gap Translation results in equal or better performance than the Time Difference Translation on every single problem. The

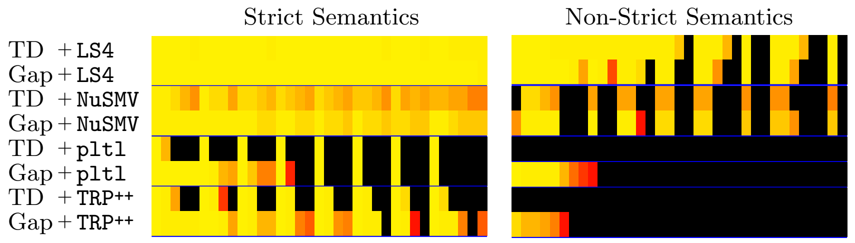

Figure 4: Heat map for the performance of LTL provers on MJS problems. Each rectangle represents the runtime of a prover on an encoding of the MJS problem, with runtimes given in colours as follows:

🟨 $<$ 0.01 sec	🟨 $>$ 0.01 sec, $\leq$ 0.25 sec	🟨 $>$ 0.25 sec, $\leq$ 0.50 sec	🟨 $>$ 0.50 sec, $\leq$ 1 sec
🟧 $>$ 1 sec, $\leq$ 2 sec	🟧 $>$ 2 sec, $\leq$ 4 sec	🟧 $>$ 4 sec, $\leq$ 8 sec	🟧 $>$ 8 sec, $\leq$ 16 sec
🟧 $>$ 16 sec, $\leq$ 32 sec	🟧 $>$ 32 sec, $\leq$ 64 sec	🟧 $>$ 64 sec, $\leq$ 125 sec	🟥 $>$ 125 sec, $\leq$ 250 sec
🟥 $>$ 250 sec, $\leq$ 500 sec	🟥 $>$ 500 sec, $\leq$ 1000 sec	⬛ Timout exceeded	

Gap Translation together with LS4 offers the best performance for every instance but does not provide models for satisfiable problems. NuSMV is the only prover that returns models of satisfiable problems and its combination with the Gap Translation provides the second best performance overall.

Regarding the formalisation of MJS problems in the non-strict semantics, the most striking observation we can make from Figure 4 is how much more challenging the corresponding LTL satisfiability problems are for all the provers, as indicated by the very high number of timeouts. Overall, the Non-Strict Gap Translation still results in better performance than the Non-Strict Time Difference Translation. The combination of the Non-Strict Gap Translation and LS4 is again the best performing single approach, but exceeds the timeout for most of the unsatisfiable MJS problems. NuSMV is again the second best prover. It is able to solve and return a model for all satisfiable problems. With the Non-Strict Gap Translation it typically does so an order of magnitude faster than with the Non-Strict Time Difference Translation. On unsatisfiable problems, NuSMV with the Non-Strict Time Difference Translation exceeds the timeout on all unsatisfiable problems and with the Non-Strict Gap Translation it does so on 18 out of 20 unsatisfiable problems. In summary, the experimental results presented in this section provide further evidence of the significant performance improvements that can be gained from the use of the Gap over Time Difference Translations.

8 Conclusion

We presented and evaluated experimentally four translations from MTL to LTL. The translations using time difference are based on the MTL decision procedure presented in [3] and use the bounded model property. Note that the translations using 'gap' are proved independently of this property. Our translations provide a route to practical reasoning about MTL over the naturals via LTL solvers. As future work, we intend to investigate whether we can translate PDDL3.0 statements [12] into MTL and apply our translations to the planning domain.

References

1. Abbas, H., Fainekos, G., Sankaranarayanan, S., Ivančić, F., Gupta, A.: Probabilistic temporal logic falsification of cyber-physical systems. ACM Transactions on Embedded Computing Systems (TECS) 12(2s), 95:1–95:30 (2013)
2. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. J. ACM 43(1), 116–146 (1996)
3. Alur, R., Henzinger, T.A.: Real-time logics: Complexity and expressiveness. Inf. Comput. 104(1), 35–77 (1993)
4. Alur, R., Henzinger, T.A.: A really temporal logic. J. ACM 41(1), 181–204 (1994)
5. Bersani, M.M., Rossi, M., San Pietro, P.: A tool for deciding the satisfiability of continuous-time metric temporal logic. Acta Informatica 53(2), 171–206 (2016)
6. Bouyer, P., Markey, N., Ouaknine, J., Worrell, J.: The cost of punctuality. In: Proc. LICS 2007. pp. 109–120. IEEE (2007)
7. Cimatti, A., Clarke, E.M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: An OpenSource tool for symbolic model checking. In: Proc. CAV 2002. LNCS, vol. 2404, pp. 359–364. Springer (2002)
8. Dauzère-Pérès, S., Paulli, J.: An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Annals of Operations Research 70, 281–306 (1997)
9. Dixon, C., Fisher, M., Konev, B.: Temporal Logic with Capacity Constraints. In: Proc. FroCoS 2007. LNCS, vol. 4720, pp. 163–177. Springer (2007)
10. Fisher, M.: A normal form for temporal logics and its applications in theorem-proving and execution. Journal of Logic and Computation 7(4), 429–456 (1997)
11. Gabbay, D., Pnueli, A., Shelah, S., Stavi, J.: On the temporal analysis of fairness. In: Proc. POPL '80. pp. 163–173. ACM (1980)
12. Gerevini, A., Haslum, P., Long, D., Saetti, A., Dimopoulos, Y.: Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. Artificial Intelligence 173(5-6) (2009)
13. Goré, R.: And-or tableaux for fixpoint logics with converse: LTL, CTL, PDL and CPDL. In: Proc. IJCAR 2014. LNCS, vol. 8562, pp. 26–45. Springer (2014)
14. Graham, R.L.: Bounds for certain multiprocessing anomalies. Bell Labs Technical Journal 45(9), 1563–1581 (1966)
15. Gunadi, H., Tiu, A.: Efficient runtime monitoring with metric temporal logic: A case study in the Android operating system. In: Proc. FM 2014. LNCS, vol. 8442, pp. 296–311. Springer (2014)
16. Hustadt, U., Konev, B.: TRP++2.0: A temporal resolution prover. In: Proc. CADE-19. LNCS, vol. 2741, pp. 274–278. Springer (2003)
17. Karaman, S., Frazzoli, E.: Vehicle routing problem with metric temporal logic specifications. In: Proc. CDC 2008. pp. 3953–3958. IEEE (2008)
18. `LS4`, `https://github.com/quickbeam123/ls4`
19. `NuSMV`, `http://nusmv.fbk.eu/`
20. Ouaknine, J., Worrell, J.: Some recent results in metric temporal logic. In: Proc. FORMATS 2008. LNCS, vol. 5215, pp. 1–13. Springer (2008)
21. `pltl`, `http://users.cecs.anu.edu.au/~rpg/PLTLProvers/`
22. Pnueli, A.: The temporal logic of programs. In: Proc. SFCS '77. pp. 46–57. IEEE (1977)
23. Schuppan, V., Darmawan, L.: Evaluating LTL satisfiability solvers. In: Proc. ATVA 2011. LNCS, vol. 6996, pp. 397–413. Springer (2011)

24. Schwendimann, S.: A new one-pass tableau calculus for PLTL. In: Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX '98. Lecture Notes in Computer Science, vol. 1397, pp. 277–292. Springer (1998)
25. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. J. ACM 32(3), 733–749 (1985)
26. Suda, M., Weidenbach, C.: A PLTL-prover based on labelled superposition with partial model guidance. In: Proc. IJCAR. LNCS, vol. 7364, pp. 537–543. Springer (2012)
27. Thati, P., Roşu, G.: Monitoring algorithms for metric temporal logic specifications. Electronic Notes in Theoretical Computer Science 113, 145–162 (2005)
28. `TRP++`, `http://cgi.csc.liv.ac.uk/~konev/software/trp++/`
29. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In: Automation of reasoning, pp. 466–483. Springer (1983)