Journal of Automated Reasoning (2020) 64:1553–1610
https://doi.org/10.1007/s10817-020-09541-4

Theorem Proving for Pointwise Metric Temporal Logic Over the Naturals via Translations

Ullrich Hustadt[1] · **Ana Ozaki**[2] · **Clare Dixon**[1]

Received: 31 May 2019 / Accepted: 6 January 2020 / Published online: 19 February 2020

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 (i) a strict monotonic function and by (ii) a non-strict monotonic function (which allows multiple states to be mapped to the same time point). We use this logic to model examples from robotics, traffic management, and scheduling, discussing the effects of different modelling choices. 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. We also define a branching-time version of the logic and provide translations into computation tree logic.

Keywords Metric temporal logic · Theorem proving · Modelling

1 Introduction

Linear and branching-time temporal logics have been used for a long time for the specification and verification of reactive systems. In linear-time temporal logic (LTL) [24,48] 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

C. Dixon was partially supported by the EPSRC funded RAI Hubs FAIR-SPACE (EP/R026092/1) and RAIN (EP/R026084/1), and the EPSRC funded programme Grant S4 (EP/N007565/1).

✉ Clare Dixon
cldixon@liverpool.ac.uk

Ullrich Hustadt
uhustadt@liverpool.ac.uk

Ana Ozaki
ana.ozaki@unibz.it

[1] Department of Computer Science, University of Liverpool, Liverpool, UK

[2] Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy

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 [6,7], considering different underlying models of time and operators allowed. MTL has been used to formalise vehicle routing problems [35], monitoring of algorithms [28,53] and cyber-physical systems [2], among others. A survey about MTL and its fragments can be found in [46]. It is known that MTL over the reals is undecidable, though, decidable fragments have been investigated [5,10,12].

Here we consider MTL with pointwise semantics over the natural numbers, following [6], 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, $\Diamond_{[3,7]}p$ 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 one, in MTL, time is allowed to irregularly 'jump' from one state to the next. The semantics uses a mapping from states to time points to keep track of the time difference between consecutive states. MTL formulae can then express constraints on such time differences. For example, using $\bigcirc_{[2,2]}p$ we can state that the time difference from the current state to the next state is two while $\bigcirc_{[3,\infty]}p$ states that the next state is at least three moments from now.

Following Alur and Henzinger [6], 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. 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 [6], 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: (i) a strict monotonic function and by (ii) 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 assuming a binary encoding of the constants). Since the satisfiability problem for LTL is PSpace-complete [51], our translations preserve the ExpSpace complexity of the MTL satisfiability problem over the natural numbers [6]. We provide a range of examples from robotics, traffic management, and scheduling. We present and discuss different modelling choices and their effect, including the choice between strict and non-strict semantics as the basis for the formalisation of a domain.

The translations allow us to use existing LTL solvers to determine the satisfiability of MTL formulae. To the best of our knowledge, there are no implementations of solvers for MTL with pointwise discrete semantics. We use several LTL solvers, covering a wide range of calculi and methods, to experimentally investigate the properties of the LTL formulae resulting from the translation of MTL formulae. We provide experimental results for both the modelled examples and we propose specific formulae to investigate and analyse the difference between the translations and provers.

Finally we consider a branching-time version of MTL and provide a translation into Computation Tree Logic (CTL) using the gap translation.

Our contributions are:

- translations from MTL to LTL which preserve the ExpSpace complexity of the MTL satisfiability problem and allow the use of LTL solvers to determine the satisfiability of MTL formulae;
- an experimental analysis of the behaviour of LTL solvers on the resulting formulae;
- encodings of problems from robotics, traffic management, and the classical multiprocessor job-shop scheduling problem [16,27] into MTL, as examples of the kind of problems that can be solved using MTL;
- discussions about the modelling of these examples that illuminate interesting aspects of MTL specifications; and
- a translation from a branching-time version of the logic into CTL that gives a tight 2EXPTIME upper bound to the satisfiability problem.

This paper is a revised and extended version of [33]. It includes a range of examples specified in MTL, full proofs, full details, updated and extended experimental results, a branching-time version of the logic with its translation into CTL and a more comprehensive discussion of related work.

In Sect. 2 we provide the syntax and semantics of LTL and MTL. We provide a range of examples in Sect. 3 encoding problems in MTL from robotics, traffic management, and job-shop scheduling. Our translations from MTL to LTL are provided in Sects. 4 and 5. Experimental results are provided in Sects. 6 and 7. A branching-time version of the logic and its translation into CTL is given in Sect. 8. We discuss related work in Sect. 9 and provide concluding remarks in Sect. 10.

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 $\Box\varphi \equiv \neg\Diamond\neg\varphi$. To simplify the presentation, we further use additional n-ary boolean operators. If $S = \{\varphi_1, \ldots, \varphi_n\}$ is a finite set of LTL formulae, then $\oplus_{\leq 1} S$, $\oplus_{\geq 1} S$, and $\oplus_{=1} S$ resp., are LTL formulae. The semantics of such formulae is as follows. Let σ' be a state sequence and $i \in \mathbb{N}$. Then, $(\sigma', i) \models \oplus_{\leq 1} S$ iff $(\sigma', i) \models \varphi_j$ for at most one $\varphi_j \in S$,

$1 \leq j \leq n$. Analogously, $(\sigma', i) \models \oplus_{\geq 1} S$ iff $(\sigma', i) \models \varphi_j$ for at least one $\varphi_j \in S, 1 \leq j \leq n$; and $(\sigma', i) \models \oplus_{=1} S$ iff $(\sigma', i) \models \oplus_{\geq 1} S \wedge \oplus_{\leq 1} S$.

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 $c_1, c_2 \in \mathbb{N}$. We sometimes write $k \in j + [c_1, c_2]$ with the meaning that $k \in [c_1 + j, c_2 + j]$. 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:

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

Further connectives are defined as usual: $\Diamond_I \varphi \equiv (\mathbf{true} \, \mathcal{U}_I \, \varphi)$ and $\Box_I \varphi \equiv \neg\Diamond_I \neg\varphi$. A timed state sequence $\rho = (\sigma, \tau)$ is a *model* of an MTL formula φ iff $(\rho, 0) \models \varphi$. An MTL formula φ is *satisfiable* iff there exists a model of φ and it is *valid* iff every timed state sequence is a model of it. To transform an MTL formula into Negation Normal Form, one uses the constrained dual until $\tilde{\mathcal{U}}_I$ operator [46], defined as $(\varphi \, \tilde{\mathcal{U}}_I \, \psi) \equiv \neg(\neg\varphi \, \mathcal{U}_I \, \neg\psi)$.

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 two. We can express this as follows:

$$\neg(\bigcirc_{[2,2]} p) \equiv \bigcirc_{[2,2]}\neg p \vee \bigcirc_{[0,1]} \mathbf{true} \vee \bigcirc_{[3,\infty)} \mathbf{true}.$$

Moreover, the usual LTL equivalence for negated until, namely

$$\neg(\varphi \, \mathcal{U} \, \psi) \equiv \Box\neg\psi \vee ((\neg\psi) \, \mathcal{U} \, (\neg\varphi \wedge \neg\psi))$$

does not hold in MTL, i.e.

$$\neg(\varphi \, \mathcal{U}_I \, \psi) \not\equiv \Box_I \neg\psi \vee ((\neg\psi) \, \mathcal{U}_I \, (\neg\varphi \wedge \neg\psi)).$$

To see this, consider the formula $\neg(p \, \mathcal{U}_{[2,2]} \, q)$. It is satisfied by a timed state sequence in which p never holds, state 0 is mapped to time point 0, state 1 is mapped to time point 1, and q only holds in state 2 which is mapped to time point 2. The formula is satisfied because p does not hold at state 0 or 1. However this timed state sequence does not satisfy $\Box_{[2,2]}\neg q \vee ((\neg q) \, \mathcal{U}_{[2,2]} \, (\neg p \wedge \neg q))$ as the eventuality of the second disjunct does not hold within the interval $[2, 2]$. Note that in LTL the equivalence holds. In our example, if we

remove the intervals from the equivalence and interpret the timed state sequence as an LTL model we would have the LTL formula $\Box\neg q \vee ((\neg q)\,\mathcal{U}\,(\neg p \wedge \neg q))$ holding at state 0 as the eventuality of the second disjunct holds at state 0 and 1. The intervals in MTL prevent this.

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 [23,55]. 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 Examples

We consider a range of examples that can be represented using MTL relating to robotics, traffic management, and job-shop scheduling.

First we consider a scenario relating to foraging robots based on a state transition system with timing constraints relating how long robots can spend in each state searching for food, moving home, etc. We first model a single robot using the strict semantics with timing constraints on the next operators to indicate the change from one state to another. We discuss the difficulties with this modelling approach for multiple robots and show how the scenario can be re-modelled to represent multiple robots of the same type. Next we consider an example related to traffic management, in particular, to traffic lights, again with the strict semantics. This has two traffic lights that are not identical, modelled using until operators with timing constraints that enforce the correct order and timings of the traffic light colours. The third example formulates multiprocessor job-shop scheduling. We have multiple processors and show how this can be modelled using both the strict and non-strict semantics.

3.1 Foraging Robots

This example is based on the models for foraging robot swarms [38]. Robots are located in an arena with a "home" for them, they leave the home, carry out a random walk to locate "food". If food is detected they move towards the food, attempt to grab it and return home. If food is not found or another robot grabs the detected food and the time for searching for food is over the robot will move home without food. The food is deposited at home and increases the overall energy of the swarm but searching for food decreases this energy. Here we focus on the timing constraints rather than energy. Initially we consider one robot and later we consider multiple robots. Robots can be in one of nine states:

resting	resting at home;
leavingHome	leaving home;
randomWalk	moving around looking for food;
moveToFood	moving towards food having located it;
scanArena	looking for food;

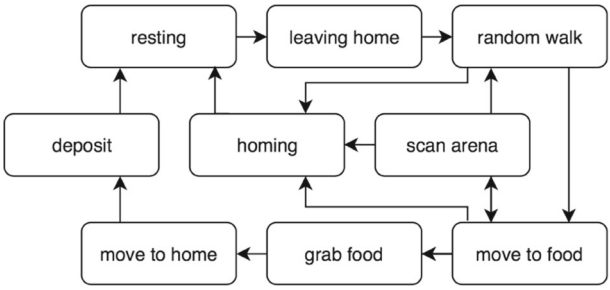

Fig. 1 State transition system for foraging robots

grabFood attempting to grab food that has been found;
moveToHome the robot moves home with food;
deposit the robot deposits the food at home, and
homing the robot moves home without food.

The state transition system representing their movement between states is shown in Fig. 1.

There are three parameters: the maximum time T_r that can be spent resting, the maximum time T_s that can be spent searching for food (*leavingHome*, *randomWalk*, *scanArena*, and *moveToFood*), and the maximum time T_d to get home with or without food (*moveToHome* and *homing*). The latter is based on an assumption that there is a maximum distance between the food and home. We assume that *grabFood* and *leavingHome* takes between two and three time units and *deposit* takes three to four time units. Initially the robot is resting (*resting*). Next we specify that the robot must be in exactly one of the states at any moment. Let

$$S = \{\mathit{resting}, \mathit{leavingHome}, \mathit{randomWalk}, \mathit{moveToFood}, \mathit{grabFood}, \mathit{moveToHome}, \mathit{deposit}, \mathit{homing}, \mathit{scanArena}\}$$

We require that each time point is 'marked' with exactly one of the states.

$$\Box_{[0,\infty)}(\oplus_{=1} S)$$

Next we define the state transition system representing the robot movement.

$$\begin{aligned}
\Box_{[0,\infty)}(\mathit{resting} &\rightarrow \bigcirc_{[1,T_r]}\mathit{leavingHome})\\
\Box_{[0,\infty)}(\mathit{leavingHome} &\rightarrow \bigcirc_{[2,3]}\mathit{randomWalk})\\
\Box_{[0,\infty)}(\mathit{randomWalk} &\rightarrow \bigcirc_{[1,\infty)}(\mathit{homing} \vee \mathit{moveToFood})\\
\Box_{[0,\infty)}(\mathit{moveToFood} &\rightarrow \bigcirc_{[1,\infty)}(\mathit{grabFood} \vee \mathit{homing} \vee \mathit{scanArena})\\
\Box_{[0,\infty)}(\mathit{scanArena} &\rightarrow \bigcirc_{[1,\infty)}(\mathit{randomWalk} \vee \mathit{moveToFood} \vee \mathit{homing})\\
\Box_{[0,\infty)}(\mathit{grabFood} &\rightarrow \bigcirc_{[2,3]}\mathit{moveToHome})\\
\Box_{[0,\infty)}(\mathit{moveToHome} &\rightarrow \bigcirc_{[1,T_d]}\mathit{deposit})\\
\Box_{[0,\infty)}(\mathit{deposit} &\rightarrow \bigcirc_{[3,4]}\mathit{resting})\\
\Box_{[0,\infty)}(\mathit{homing} &\rightarrow \bigcirc_{[1,T_d]}\mathit{resting})
\end{aligned}$$

The variable *searching* holds in all the states *leavingHome*, *randomWalk*, *scanArena*, *moveToFood* and nowhere else.

$$\Box_{[0,\infty)}(\mathit{searching} \leftrightarrow (\mathit{leavingHome} \vee \mathit{randomWalk} \vee \mathit{scanArena} \vee \mathit{moveToFood}))$$

If the robot is searching then there is a maximum amount of time it can stay searching.

$$\Box_{[0,\infty)}(searching \rightarrow \Diamond_{[1,T_s]}\neg searching) \tag{1}$$

There are a number of properties we could try to prove.

- Having left home the robot will eventually reach home again.

$$\Box_{[0,\infty)}(leavingHome \rightarrow \Diamond_{[0,\infty)}resting)$$

- The maximum time the robot needs to return home is x

$$\Box_{[0,\infty)}(leavingHome \rightarrow \Diamond_{[0,x]}resting)$$

  In particular, the lowest value x for which we can prove this property is the maximum time the robot can stay away from home.
- The minimum time the robot needs to return home is y.

$$\Box_{[0,\infty)}(leavingHome \rightarrow \Box_{[0,y-1]}\neg resting)$$

  Here the highest value y for which we can prove this property is the minimum time the robot needs to return home.

For a single robot, we believe that the most natural way to represent the foraging robot scenario is as above, with most timing constraints on the next operator. However, to represent multiple robots we would need a copy of each state and the formulae representing the transitions for each robot, i.e. propositions such as *resting* and *grabFood* would be parameterised by robot number (for example $resting_i$ and $grabFood_i$ for robot i). In this type of scenario and modelling, the next operator is working as a synchronisation point: all robots must move into another state together. In general, this does not accurately reflect the intended behaviour of robots as some robots may stay in the state they are currently in whilst others may change state. A potential solution to this would be to include the current state of the robot on the right hand side of the formulae under a next operator, as follows

$$\Box_{[0,\infty)}(resting_i \rightarrow \bigcirc_{[1,T_r]}(resting_i \vee leavingHome_i))$$

so that the robots do not have to change state at the same time. However this introduces other issues as the robot can now stay in the resting state for longer than T_r time units.

To model this correctly we would need to adopt an approach more like how we have dealt with the timings for searching, i.e. formulae for each transition in the state transition system, one denoting the possible next states and one denoting the required timing constraints to move out of the current state [like Eq. (1)]. We also need to introduce a new propositional variable for each robot, for each state, to denote the place where the proposition for that state changes from being false to becoming true. We need this so we can identify when we first entered some state so we can formalise the maximum time we can stay in this state. To illustrate this approach, we focus on the state $resting_i$ in the following. Here, we denote the new propositional variable that indicates a state change by $startResting_i$ and impose the following constraint on it.

$$\begin{aligned}&(resting_i \leftrightarrow startResting_i) \wedge \\ &\Box_{[0,\infty)}((\bigcirc_{[1,\infty)}startResting_i) \leftrightarrow (\neg resting_i \wedge \bigcirc_{[1,\infty)}resting_i))\end{aligned} \tag{2}$$

Next, for robot i, to represent the transition out of the state $resting_i$ we would have the following formulae, the first denoting the next possible states of robot i and the second enforcing the maximum time that robot i can be in the $resting$ state.

$$\Box_{[0,\infty)}(resting_i \rightarrow \bigcirc_{[1,\infty)}(resting_i \vee leavingHome_i))$$
$$\Box_{[0,\infty)}(startResting_i \rightarrow \Diamond_{[1,T_r]}\neg resting_i)$$

An alternative to the above two formulae would be using an until operator as follows

$$\Box_{[0,\infty)}(startResting_i \rightarrow (resting_i \,\mathcal{U}_{[1,T_r]}\, leavingHome_i))$$

Although we have introduced the proposition $startResting_i$ to ensure the relevant timed formulae only hold from where $resting_i$ first became true it is only essential here when the left hand side of the timing constraint is greater than one. Consider the following timed state sequence as an example. Let $\rho = (\sigma, \tau)$ be the timed state sequence where $\tau(0) = 0$, $\tau(1) = 3$, $\tau(2) = 4$, and $\tau(3) = 10$ where $(\rho, 0) \models resting_i$, $(\rho, 1) \models resting_i$, $(\rho, 2) \models \neg resting_i$ and $(\rho, 3) \models \neg resting_i$. We consider using

$$\Box_{[0,\infty)}(resting_i \rightarrow \Diamond_{[3,5]}\neg resting_i) \tag{3}$$

rather than

$$\Box_{[0,\infty)}(startResting_i \rightarrow \Diamond_{[3,5]}\neg resting_i) \tag{4}$$

In ρ, $resting_i$ changes from true to false within 3–5 time units from where $resting_i$ was first true, as $\tau(2) - \tau(0) \in [3, 5]$. So

$$(\rho, 0) \models \Diamond_{[3,5]}\neg resting_i$$

and therefore

$$(\rho, 0) \models (resting_i \rightarrow \Diamond_{[3,5]}\neg resting_i)$$

However

$$(\rho, 1) \not\models \Diamond_{[3,5]}\neg resting_i$$

as $\tau(2) - \tau(1) \notin [3, 5]$ and as $\tau(3) - \tau(1) \notin [3, 5]$ so

$$(\rho, 1) \not\models (resting_i \rightarrow \Diamond_{[3,5]}\neg resting_i)$$

so (3) does not hold. But if we introduce $startResting_i$ as defined in (2) then $(\rho, 0) \models startResting_i$, $(\rho, 1) \models \neg startResting_i$, $(\rho, 2) \models \neg startResting_i$ and $(\rho, 3) \models \neg startResting_i$, and Eq. (4) holds as desired.

We discuss something similar in the next example in Sect. 3.2 relating to traffic lights about the proposition $change_i$. Another solution to having multiple robots may be adopting the non-strict semantics in a similar way to what is presented for the multiprocessor job-shop scheduling problem in Sect. 3.3.

The timing aspects of other examples relating to robots could also be modelled. For example a personal robot assistant located in a house equipped with sensors that inform the robot about the state of the house and its participants is considered in [18,19,57]. The robot actions are controlled by a set of *behaviours*. These are of the form of "if …then …" where the "if" part depends on state of the robot, sensor data and internal variables and the "then" part contains a sequence of robot actions. These contain several timing aspects for example "wait ten seconds", "wait an hour" etc that we could model using this logic.

3.2 Traffic Lights

Consider a pair of traffic lights at a junction with two roads, R1 and R2, crossing each other. Traffic light one is for road R1 and traffic light two is for road R2. As road R1 is busier, there is a camera to detect when there are cars waiting for traffic light 1. This makes the lights have different behaviour so the problem is not symmetric.

We use the following propositional variables to denote the states that the traffic lights and the detection sensor can be in:

r_i	traffic light i is at red;
a_i^r	traffic light i is at amber following being at red;
a_i^g	traffic light i is at amber following being at green;
a_i	traffic light i is at amber (i.e. one of the above holds);
g_i	traffic light i is at green;
$detect$	traffic light one detects cars waiting;
$change_i$	traffic light i has just changed colour;

Traffic light $i \in \{1, 2\}$ is exactly red, amber or green at every moment in time.

$$\Box_{[0,\infty)}(\oplus_{=1}\{r_i, g_i, a_i\})$$

Traffic light $i \in \{1, 2\}$ is at amber if and only if either it is at amber after red or amber after green (at every moment in time). Note that while our modelling here does not explicitly allow the red and amber lights to both be on together, as is common in traffic light sequences in Europe, our use of two different propositional variables for amber after red and amber after green allows us to differentiate between these situations.

$$\Box_{[0,\infty)}(a_i \leftrightarrow (a_i^r \vee a_i^g))$$

Traffic light $i \in \{1, 2\}$ cannot be both amber after red and amber after green at the same moment in time.

$$\Box_{[0,\infty)}(\neg a_i^r \vee \neg a_i^g)$$

If one traffic light is green the other is not green.

$$\Box_{[0,\infty)}(\neg g_1 \vee \neg g_2)$$

The sequence of the traffic lights is red, then amber, then green, then amber, then red, and so on. Both traffic lights eventually change to amber after having been at green, stay at amber after green between one and two time units before changing to red, and stay at amber after red between one and three time units before changing to green.

The formalisation of these constraints requires some careful consideration. For instance, one might consider the formula $\Box_{[0,\infty)}(a_1^g \,\mathcal{U}_{[2,3]}\, r_1)$ as a possible way of expressing that red follows amber after green within two or three time units. However, together with the constraints above, it is unsatisfiable: suppose there is a timed state sequence $\rho = (\sigma, \tau)$ that satisfies our constraints. For any $i \in \mathbb{N}$, $(a_1^g \,\mathcal{U}_{[2,3]}\, r_1)$ is true at (ρ, i) and there exists $k \geq i$ with $\tau(k) - \tau(i) \in [2, 3]$ and r_1 is true at (ρ, k). But since $(a_1^g \,\mathcal{U}_{[2,3]}\, r_1)$ is always true, it is also true at (ρ, k) and since $0 \notin [2, 3]$, a_1^g must be true at (ρ, k). This contradicts the constraint that a_1^g and r_1 cannot both be true at any state. In the following we make use of the proposition $change_i$ together with a particular colour as a guard making sure that the timing constraints hold when a light changes to that colour.

The variable $change_i$ holds in the first moment in time and also if the traffic light was not red, (respectively amber, green) in the previous moment and is now red (respectively amber, green).

$$\begin{aligned} & change_i \\ & \wedge \Box_{[0,\infty)}((\bigcirc_{[1,\infty)} change_i) \leftrightarrow ((\neg r_i \wedge \bigcirc_{[1,\infty)} r_i) \vee (\neg a_i \wedge \bigcirc_{[1,\infty)} a_i) \\ & \qquad\qquad\qquad\qquad \vee (\neg g_i \wedge \bigcirc_{[1,\infty)} g_i))) \end{aligned}$$

Strictly, for a formula such as $\Box_{[0,\infty)}(g_1 \,\mathcal{U}_{[0,\infty)}\, a_1^g)$, expressing that traffic light 1 stays green until it eventually changes to amber after green, where the lower bound on the interval restricting the until operator is 0, we do not need a guard, but add one anyway in order to obtain the following more uniform formalisation.

$$\begin{aligned} & \Box_{[0,\infty)}((change_i \wedge g_i) \rightarrow (g_i \,\mathcal{U}_{[0,\infty)}\, a_i^g)) \\ & \Box_{[0,\infty)}((change_i \wedge a_i^g) \rightarrow (a_i^g \,\mathcal{U}_{[2,3]}\, r_i)) \\ & \Box_{[0,\infty)}((change_i \wedge a_i^r) \rightarrow (a_i^r \,\mathcal{U}_{[1,2]}\, g_i)) \end{aligned}$$

Regarding the change from red to amber, the two traffic lights behave differently. The following two formulae state that traffic light 1 will eventually change to amber after having been red, while traffic light 2 will do so after 0 to 4 time units.

$$\begin{aligned} & \Box_{[0,\infty)}((change_1 \wedge r_1) \rightarrow (r_1 \,\mathcal{U}_{[0,\infty)}\, a_1^r)) \\ & \Box_{[0,\infty)}((change_2 \wedge r_2) \rightarrow (r_2 \,\mathcal{U}_{[0,4]}\, a_2^r)) \end{aligned}$$

Finally, if the car detection sensor on road R1 detects a car and the traffic light is not green then it will be green within 3 time units.

$$\Box_{[0,\infty)}((detect \wedge \neg g_1) \rightarrow \Diamond_{[0,3]} g_1) \tag{5}$$

Properties we would like to check are as follows.

– Infinitely often each traffic light will be green.

$$\Box_{[0,\infty)} \Diamond_{[0,\infty)} g_1 \wedge \Box_{[0,\infty)} \Diamond_{[0,\infty)} g_2 \tag{6}$$

– If a car is detected on road R1 then the wait will be at most x time units.

$$\Box_{[0,\infty)}(detect \rightarrow \Diamond_{[0,x]} g_1) \tag{7}$$

It is worthwhile to note that the property $\Box_{[0,\infty)}(detect \rightarrow \Diamond_{[1,x']} g_1)$ is not provable for any value of x'. In a countermodel, g_1 always holds at exactly the same states as $detect$, i.e. it is true in states where $detect$ is true and false in states where $detect$ is false. This satisfies the constraint (5) trivially. Also (6) is still satisfied due to the until constraints forcing the correct light sequence thereby falsifying $\Diamond_{[1,x']} g_1$.

– If traffic light 2 is currently red then it will be green within 1 to y time units.

$$\Box_{[0,\infty)}(r_2 \rightarrow \Diamond_{[1,y]} g_2) \tag{8}$$

Here the property $\Box_{[0,\infty)}(r_2 \rightarrow \Diamond_{[0,y]} g_2)$ would hold for exactly the same values of y as property (8). This follows from the constraint that the traffic lights cannot be red and green at the same time.

– If a traffic light is currently amber or red, then a car has to wait at most z time units until the traffic light will be green. We could try to formalise this property as

$$\Box_{[0,\infty)}(((a_1 \vee r_1) \rightarrow \Diamond_{[0,z]} g_1) \wedge \Box_{[0,\infty)}((a_2 \vee r_2) \rightarrow \Diamond_{[0,z]} g_2) \tag{9}$$

However, the first conjunct of (9) does not follow from our specification as there is no upper bound on the time that traffic light 1 can stay red if no car is detected.
In order to correctly formalise the property we have to include that the presence of a car makes $detect$ true:

$$\Box_{[0,\infty)}(((a_1 \vee r_1) \wedge detect) \rightarrow \Diamond_{[0,z]} g_1) \wedge \Box_{[0,\infty)}((a_2 \vee r_2) \rightarrow \Diamond_{[0,z]} g_2) \tag{10}$$

3.3 Multiprocessor Job-Shop Scheduling

Our next example is a generalisation of the classical job-shop scheduling problem, called the Multiprocessor Job-shop Scheduling (MJS) problem [16,27]. The representation provided is based on that in [17]. 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 schedule such that after t time units all jobs will have been processed by the machines.

We use this example to illustrate encodings with both the strict and non-strict semantics. Under the strict semantics we use propositional variables $run_{j_i m_l}$ to encode that job j_i is running on machine m_l, whereas in the non-strict semantics we use separate propositional variables m_l for machines and run_{j_i} for (the execution of) jobs. So, the strict semantics uses quadratically more propositional variables than the non-strict semantics. This is due to the fact that in the non-strict semantics we can encode parallelism by associating multiple states with the same time point. Our encodings have the property that there is a schedule 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 schedule satisfying the constraints of an instance of the problem.

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.

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

– $startRun_{j_i}$, run_{j_i} and $hasRun_{j_i}$ denote the start, the execution and the end of the execution of job j_i on some machine, respectively;
– $startRun_{j_i m_l}$ and $run_{j_i m_l}$ 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 formulae state that (11) once a job starts running it must start running on one of the machines and that (12) once a job starts running on a machine it must run on that machine (where $\bigwedge_{1 \le i \le n}$ and $\bigwedge_{1 \le i \le n, 1 \le l \le k}$ in front of the formulas is omitted for brevity)

$$\Box_{[0,\infty)}(startRun_{j_i} \rightarrow \bigvee_{l=1}^{k} startRun_{j_i m_l}) \tag{11}$$

$$\Box_{[0,\infty)}(startRun_{j_i m_l} \rightarrow run_{j_i m_l}) \tag{12}$$

The parametric formula (13) 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 parametric formula (14), once a job has started it cannot be started again.

$$\Box_{[0,\infty)}(run_{j_i m_l} \rightarrow (\bigwedge_{p=1,p\neq l}^{k} \neg run_{j_i m_p} \wedge \bigwedge_{q=1,q\neq i}^{n} \neg run_{j_q m_l})) \quad (13)$$

$$\Box_{[0,\infty)}(startRun_{j_i} \rightarrow \bigcirc_{[0,\infty)}\Box_{[0,\infty)}\neg startRun_{j_i}) \quad (14)$$

We write $\neg run_{j_i}$ as a short hand for $\bigwedge_{l=1}^{k} \neg run_{j_i m_l}$. We can use (15) to denote that once job j_i is started to run on machine m_l it takes time $t_{j_i m_l}$ and (16) to denote that once job j_i has finished running on machine m_l it will not run again. Further, parametric formula (17) denotes that job j_i cannot be run until it has started.

$$\Box_{[0,\infty)}(startRun_{j_i m_l} \rightarrow \Box_{[0,t_{j_i m_l}-1]} run_{j_i m_l} \wedge \neg hasRun_{j_i}) \quad (15)$$

$$\Box_{[0,\infty)}(startRun_{j_i m_l} \rightarrow \Box_{[t_{j_i m_l},\infty)}(\neg run_{j_i} \wedge hasRun_{j_i})) \quad (16)$$

$$\Box_{[0,\infty)}(\neg run_{j_i} \,\mathcal{U}\, startRun_{j_i}) \quad (17)$$

We assume that initially no jobs have run, i.e., $\bigwedge_{i=1}^{n} \neg hasRun_{j_i}$; and that (18) if a job has not run and is currently not running then it will not have run in the next moment.

$$\Box_{[0,\infty)}((\neg hasRun_{j_i} \wedge \neg run_{j_i}) \rightarrow \bigcirc_{[0,\infty)}\neg hasRun\, j_i) \quad (18)$$

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

- $\Box_{[0,\infty)}(run_{j_i} \leftrightarrow run_{j_i,m_l})$: job j_i must run on machine m_l;
- $\Box_{[0,\infty)}(startRun_{j_i} \rightarrow \Diamond_{[1,\infty)} startRun_{j_m})$: job j_i must start before job j_m (note that this formalisation relies on the fact that each job is started exactly once);
- $\Diamond_{[c,d]} startRun_{j_i}$: 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

- $startRun_{j_i}$ and $hasRun_{j_i}$ denote the start and the end of job j_i on some machine, respectively;
- 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 Sect. 2):

$$\Box_{[0,\infty)}(\oplus_{=1}\Pi_m \wedge \oplus_{\leq 1}\Pi_j) \quad (19)$$

Parametric formula (20) specifies that if a job is running on one machine then it cannot be running on another. Parametric formula (21) 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).

$$\Box_{[0,\infty)}((m_l \wedge run_{j_i}) \rightarrow \bigwedge_{l'\neq l} \Box_{[0,\infty)}\neg(m_{l'} \wedge run_{j_i})) \quad (20)$$

$$\Box_{[0,\infty)}(startRun_{j_i} \rightarrow \bigcirc_{[0,\infty)}\Box_{[0,\infty)}\neg startRun_{j_i}) \quad (21)$$

We use the following

$$\Box_{[0,\infty)}((startRun_{j_i} \wedge m_l) \rightarrow (\Box_{[0,t_{j_i m_l}-1]}(\neg hasRun_{j_i} \wedge (m_l \rightarrow run_{j_i})) \wedge \Diamond_{[0,t_{j_i m_l}]} hasRun_{j_i})) \quad (22)$$

to denote that once job j_i started to run on machine m_l it takes time $t_{j_i m_l}$ and (23) to denote that once job j_i has finished running on machine m_l it will not run again. Further, we use $\Box_{[0,\infty)}(\neg run_{j_i} \,\mathcal{U}\, startRun_{j_i})$ to state a job j_i cannot be run until it is started and $\Box_{[0,\infty)}(\neg hasRun_{j_i} \,\mathcal{U}\, startRun_{j_i})$ to state that a job cannot have run before it starts (another rule above will make sure that $hasRun_{j_i}$ will hold after the run has finished).

$$\Box_{[0,\infty)}((startRun_{j_i} \wedge m_l) \rightarrow \Box_{[t_{j_i m_l}+1,\infty)}(\neg run_{j_i} \wedge hasRun_{j_i})) \quad (23)$$

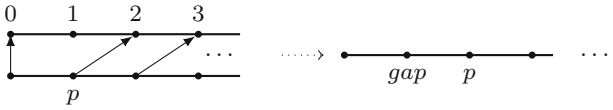

Fig. 2 Example illustrating Definition 1

We assume that initially 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 $\Diamond_{[0,t]} \bigwedge_{i=1}^{n} has Run_{j_i}$ and checking for satisfiability.

4 From MTL to LTL: Encoding 'Gaps'

Next we provide satisfiability preserving translations from MTL formulae for discrete time models into LTL using a new propositional variable gap, which is intended to hold at LTL states corresponding to unmapped time points in MTL models. Assume that our MTL formulae are in NNF and FNF. The main idea for our translation is to map each timed state sequence $\rho = (\sigma, \tau)$ into 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. Our LTL translations are exponential in the size of the input formula in MTL due to the binary encoding of the numbers in the intervals. This blow up is unavoidable and optimal, since reasoning in MTL is ExpSpace-hard and an exponential size LTL translation gives an ExpSpace upper bound.

Strict Semantics

For every model of a MTL formula there is going to be a model of the translated formula, such that $\neg gap$ 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 a state sequence $\sigma' = \sigma'_0\sigma'_1 \ldots$, 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 2 illustrates the mapping given by Definition 1. For instance, if $\rho = (\sigma, \tau)$ is the timed state sequence on the left side of Fig. 2a then $(\rho, 0) \models \bigcirc_{[2,3]} p$. Table 1 presents the translation of formulae of the form $\bigcirc_I \alpha$, $\alpha_1 \, \mathcal{U}_I \, \beta$ or $\alpha \, \tilde{\mathcal{U}}_I \, \beta$, where α, β are propositional formulae. Assume $(\varphi)^\sharp = \varphi$ if φ is a propositional formula. Observe that, in the definition of $\cdot^\sharp$, recursion is only used to deal with the case where 0 occurs on the left side of an interval. As shown in Table 1, we translate $\bigcirc_{[2,3]} p$ into:

$$\bigvee_{2 \le l \le 3} (\bigcirc^l (\neg gap \wedge p) \wedge \bigwedge_{1 \le k < l} \bigcirc^k gap).$$

Note that the state sequence represented on the right side of Fig. 2 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:

Table 1 Strict gap translation from MTL to LTL, where α, β are propositional formulae and $c_1, c_2 > 0$

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

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

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.

The following two propositions are useful for the proof of Theorem 1. Since gap is a propositional variable not occurring in σ, the time points mapped by the image of τ do not contain gap. Then, it is easy to see the following.

Proposition 1 *Given a timed state sequence $\rho = (\sigma, \tau)$, let σ' be as in Definition 1. Then, $(\sigma', 0) \models \Box(\Diamond\neg gap)$.*

The state sequence σ' in Definition 2 is such that $(\sigma', 0) \models \neg gap \wedge \Box(\Diamond\neg gap)$. Consequently, for the timed state sequence constructed in Definition 2 we have that for each $i \in \mathbb{N}$, $\tau(i) \in \mathbb{N}$. Also, for $i > 0$, $\tau(i) > \tau(i-1)$ and, so, $\tau : \mathbb{N} \to \mathbb{N}$ is well defined. The following proposition states this property.

Proposition 2 *Given a state sequence σ' such that $(\sigma', 0) \models \neg gap \wedge \Box(\Diamond\neg gap)$, let $\rho = (\sigma, \tau)$ be as in Definition 2. Then, $\tau : \mathbb{N} \to \mathbb{N}$ is a function such that $\tau(i+1) > \tau(i)$, for all $i \in \mathbb{N}$.*

We are ready for Theorem 1, which states the correctness of our translation from MTL to LTL using 'gap's. One can use similar arguments to show Theorem 2.

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 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 φ is satisfiable if, and only if, there is a timed state sequence $\rho = (\sigma, \tau)$ such that $(\rho, 0) \models \varphi$. Given a timed state sequence $\rho = (\sigma, \tau)$, let $\sigma' = \sigma'_0\sigma'_1 \ldots$ be as in Definition 1.

By Definition 1 we have that $(\sigma', \tau(0)) \models \neg gap$ and, by Proposition 1, $(\sigma', \tau(0)) \models \Box(\Diamond\neg gap)$. By definition of σ', for all propositional variables p_i occurring in σ and all $j \in \mathbb{N}$, we have $(\rho, j) \models p_i$ if, and only if, $(\sigma', \tau(j)) \models p_i$. With an inductive argument,

one can show that, for any propositional formula α not containing gap, $(\rho, j) \models \alpha$ if, and only if, $(\sigma', \tau(j)) \models \alpha$.

Also, $\varphi^\sharp \wedge \neg gap \wedge \Box(\Diamond \neg gap)$ is satisfiable if, and only if, there is a state sequence σ' such that $(\sigma', 0) \models \varphi^\sharp \wedge \neg gap \wedge \Box(\Diamond \neg gap)$. Given σ', let $\rho = (\sigma, \tau)$ be as in Definition 2. By Proposition 2, $\tau : \mathbb{N} \to \mathbb{N}$ is well defined. By definition of σ, for all propositional variables p_i occurring in σ'_n with $gap \notin \sigma'_n$ there is $j \in \mathbb{N}$ such that $\tau(j) = n$ and $(\rho, j) \models p_i$ iff $(\sigma', \tau(j)) \models p_i$. An inductive argument lifts this claim to propositional formulae.

Thus, following Table 1, we only need to show correspondences between ψ_i and $\neg gap \wedge \psi_i^\sharp$. This follows from Claims 1–3 below (other cases are similar), where $\rho = (\sigma, \tau)$ is a timed state sequence and $j \in \mathbb{N}$. Let $C - 1$ be the greatest number occurring in an interval in φ or 1, if none occur. We argue that, for all natural numbers c_1 with $0 < c_1 \leq c_2 < C$, the following claims hold.

Claim 1 $(\rho, j) \models \bigcirc_{[c_1,c_2]}\alpha$ *iff* $(\sigma', \tau(j)) \models \neg gap \wedge (\bigcirc_{[c_1,c_2]}\alpha)^\sharp$.

Proof $(\rho, j) \models \bigcirc_{[c_1,c_2]}\alpha$ iff $(\rho, j+1) \models \alpha$ and $\tau(j+1) \in \tau(j) + [c_1, c_2]$. By Definitions 1 and 2, this happens iff $(\sigma', \tau(j+1)) \models \neg gap \wedge \alpha$ and there is no $n \in \mathbb{N}$ such that $\tau(n) = m$, for $\tau(j) < m < \tau(j+1)$, meaning that for all such m, $(\sigma', m) \models gap$. As $\tau(j) + c_1 \leq \tau(j+1) \leq \tau(j) + c_2$, this happens iff $(\sigma', \tau(j)) \models \bigvee_{c_1 \leq l \leq c_2}(\bigcirc^l(\neg gap \wedge \alpha) \wedge \bigwedge_{1 \leq k < l} \bigcirc^k gap)$, and $(\sigma', \tau(j)) \models \neg gap$, by definition of σ'. So, $(\rho, j) \models \bigcirc_{[c_1,c_2]}\alpha$ iff $(\sigma', \tau(j)) \models \neg gap \wedge (\bigcirc_{[c_1,c_2]}\alpha)^\sharp$.

Claim 2 $(\rho, j) \models \alpha \, \mathcal{U}_{[c_1,c_2]} \, \beta$ *iff* $(\sigma', \tau(j)) \models \neg gap \wedge (\alpha \, \mathcal{U}_{[c_1,c_2]} \, \beta)^\sharp$.

Proof $(\rho, j) \models \alpha \mathcal{U}_{[c_1,c_2]} \beta$ iff there is $k \in \mathbb{N}$ such that $\tau(k) \in \tau(j) + [c_1, c_2]$ and $(\rho, k) \models \beta$ and for all $l \in \mathbb{N}$ with $j \leq l < k$ we have $(\rho, l) \models \alpha$. By Definitions 1 and 2, this happens iff (i) $(\sigma', \tau(k)) \models \neg gap \wedge \beta$, (ii) as $\tau(j) + c_1 \leq \tau(k) \leq \tau(j) + c_2$, $(\sigma', \tau(j)) \models \bigvee_{c_1 \leq m \leq c_2}(\bigcirc^m(\neg gap \wedge \beta))$, and (iii), for all $l \in \mathbb{N}$ with $j \leq l < k$ we have $(\sigma', \tau(l)) \models \alpha$. If $n \in \mathbb{N}$ is such that $\tau(j) \leq n < \tau(k)$ and there is no $l \in \mathbb{N}$ with $n = \tau(l)$ then, by definition of σ', $(\sigma', n) \models gap$.

Thus, by (i-iii), $(\rho, j) \models \alpha \, \mathcal{U}_{[c_1,c_2]} \, \beta$ iff
$(\sigma', \tau(j)) \models \bigvee_{c_1 \leq l \leq c_2}(\bigcirc^l(\neg gap \wedge \beta) \wedge \bigwedge_{0 \leq k < l} \bigcirc^k(gap \vee \alpha))$.

Claim 3 $(\rho, j) \models \alpha \, \tilde{\mathcal{U}}_{[c_1,c_2]} \, \beta$ *iff* $(\sigma', \tau(j)) \models \neg gap \wedge \neg(\neg(\alpha) \, \mathcal{U}_{[c_1,c_2]} \, \neg(\beta))^\sharp$.

Proof By the semantics of $\tilde{\mathcal{U}}$, we have that $(\rho, j) \models \alpha \, \tilde{\mathcal{U}}_{[c_1,c_2]} \, \beta$ iff $(\rho, j) \models \neg(\neg(\alpha) \, \mathcal{U}_{[c_1,c_2]} \, \neg(\beta))$. By Claim 2, $(\rho, j) \not\models \neg(\alpha) \, \mathcal{U}_{[c_1,c_2]} \, \neg(\beta)$ iff $(\sigma', \tau(j)) \not\models \neg gap \wedge (\neg(\alpha) \, \mathcal{U}_{[c_1,c_2]} \, \neg(\beta))^\sharp$. Since $(\sigma', \tau(j)) \models \neg gap$ (by definition of σ'), this happens iff $(\sigma', \tau(j)) \models \neg gap \wedge \neg(\neg(\alpha) \, \mathcal{U}_{[c_1,c_2]} \, \neg(\beta))^\sharp$. □

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 $\Diamond_I \psi \equiv \mathbf{true} \, \mathcal{U}_I \, \psi$):

$$\begin{aligned}\varphi^\sharp = p_0 \wedge \Box(p_0 \to (\neg gap \wedge (\textstyle\bigvee_{2 \leq l \leq 3}(\bigcirc^l(\neg gap \wedge p_1) \wedge \bigwedge_{1 \leq k < l} \bigcirc^k gap)) \\ \wedge \Box(p_1 \to (\neg gap \wedge (\textstyle\bigvee_{1 \leq l \leq 2}(\bigcirc^l(\neg gap \wedge \neg q))))).\end{aligned}$$

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.

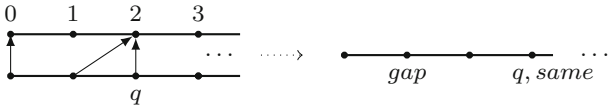

Fig. 3 Example illustrating Definition 3

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 3 illustrates the mapping given by Definition 3. For instance, if $\rho = (\sigma, \tau)$ is the non-strict timed state sequence on the left side then $(\rho, 0) \models \Diamond_{[2,2]}q$. Table 2 presents our translation of formulae of the form $\bigcirc_I\alpha$, $\alpha_1\, \mathcal{U}_I\, \beta$ or $\alpha\, \tilde{\mathcal{U}}_I\, \beta$, as in Table 1. 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 Fig. 3 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:

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

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:

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

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 $\varphi = p_0 \wedge \bigwedge_i \Box_{[0,\infty)}(p_i \rightarrow \psi_i)$ be an MTL formula in NNF and FNF. Let $\varphi^\sharp = p_0 \wedge \bigwedge_i \Box(p_i \rightarrow (\neg gap \wedge \psi_i^\sharp))$ 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, $\varphi^\sharp \wedge \neg gap \wedge \neg same \wedge \Box(\Diamond\neg gap) \wedge \Box(\neg same \vee \neg gap) \wedge \Box(gap \rightarrow \bigcirc\neg same)$ is satisfiable.*

Proof (Sketch) We use Definitions 3 and 4 to map models of φ into models of $\varphi^\sharp \wedge \neg gap \wedge \Box(\Diamond\neg gap)$ 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. □

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

MTL	Non-strict gap translation
$(\bigcirc_{[0,\infty)}\alpha)^\sharp$	$(\bigcirc_{[0,0]}\alpha)^\sharp \vee (\bigcirc_{[1,\infty)}\alpha)^\sharp$
$(\bigcirc_{[0,c_2]}\alpha)^\sharp$	$(\bigcirc_{[0,0]}\alpha)^\sharp \vee (\bigcirc_{[1,c_2]}\alpha)^\sharp$
$(\bigcirc_{[0,0]}\alpha)^\sharp$	$\bigcirc(\alpha \wedge same)$
$(\alpha\, \mathcal{U}_{[c_1,\infty)}\, \beta)^\sharp$	$\alpha \wedge \bigcirc((\alpha \wedge same)\, \mathcal{U}\, (\neg same \wedge (\alpha\, \mathcal{U}_{[c_1-1,\infty)}\, \beta)^\sharp))$
$(\alpha\, \mathcal{U}_{[0,\infty)}\, \beta)^\sharp$	$(gap \vee \alpha)\, \mathcal{U}\, (\neg gap \wedge \beta)$
$(\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta)^\sharp$	$\alpha \wedge \bigcirc((\alpha \wedge same)\, \mathcal{U}\, (\neg same \wedge (\alpha\, \mathcal{U}_{[c_1-1,c_2-1]}\, \beta)^\sharp))$
$(\alpha\, \mathcal{U}_{[0,0]}\, \beta)^\sharp$	$(\beta \wedge \neg gap) \vee (\alpha \wedge \bigcirc((\alpha \wedge same)\, \mathcal{U}\, (\beta \wedge same)))$
$(\alpha\, \mathcal{U}_{[0,c_2]}\, \beta)^\sharp$	$(\alpha\, \mathcal{U}_{[0,0]}\, \beta)^\sharp \vee (\alpha\, \mathcal{U}_{[1,c_2]}\, \beta)^\sharp$
$(\alpha\, \tilde{\mathcal{U}}_I\, \beta)^\sharp$	$\neg(\neg(\alpha)\, \mathcal{U}_I\, \neg(\beta))^\sharp$

5 From MTL to LTL: Encoding Time Differences

Next we provide satisfiability preserving translations from MTL formulae for discrete time models into LTL using new propositional variables to denote the time differences from the previous state. 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. Our LTL translations here are also exponential in the size of the input formula in MTL due to the binary encoding of the numbers in the intervals, and thus, yield tight ExpSpace upper bounds.

Strict Semantics

We denote by $C - 1$ 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 [7].

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 $(\rho_C, 0) \models \varphi$.*

Definition 5 Given a timed sequence $\rho = (\sigma, \tau)$ and $C \in \mathbb{N}$, we define a timed sequence $\rho_C = (\sigma_C, \tau_C)$ as follows:

- $\sigma_C = \sigma$;
- $\tau_C(0) = \min(\tau(0), C)$ and, for $i > 0$, we have that $\tau_C(i) = \tau_C(i-1) + \min(C, \tau(i) - \tau(i-1))$.

The following proposition states the main property of Definition 5.

Proposition 3 *Let $\rho = (\sigma, \tau)$ be a timed state sequence and let $\rho_C = (\sigma_C, \tau_C)$ be as in Definition 5. For all $i, j \in \mathbb{N}$ and all intervals of the form $I = [c_1, c_2]$ or $I = [c_1, \infty)$ with $c_1, c_2 < C$, the following holds:*

$$\tau(j) \in \tau(i) + I \Leftrightarrow \tau_C(j) \in \tau_C(i) + I$$

Proof First assume $I = [c_1, c_2]$. If $\tau(j) \in \tau(i) + [c_1, c_2]$ then $\tau(j) - \tau(i) \leq c_2 < C$ and, so, $\tau_C(j) - \tau_C(i) = \tau(j) - \tau(i)$. Thus, $\tau_C(j) \in \tau_C(i) + [c_1, c_2]$. Conversely, if $\tau_C(j) \in \tau_C(i) + [c_1, c_2]$ then $\tau_C(j) - \tau_C(i) \leq c_2 < C$ and, so, $\tau_C(j) - \tau_C(i) = \tau(j) - \tau(i)$. Thus, $\tau(j) \in \tau(i) + [c_1, c_2]$.

Now assume $I = [c_1, \infty)$. If $\tau(j) \in \tau(i) + [c_1, \infty)$ then $\tau(j) - \tau(i) \geq c_1$. If $\tau(j) - \tau(i) < C$ then $\tau_C(j) - \tau_C(i) = \tau(j) - \tau(i)$ and, so, $\tau_C(j) \in \tau_C(i) + [c_1, \infty)$. Otherwise $\tau(j) - \tau(i) \geq C$. Then, $\tau_C(j) - \tau_C(i) = C$. As $C > c_1$, we have that $\tau_C(j) \in \tau_C(i) + [c_1, \infty)$. Conversely, if $\tau_C(j) \in \tau_C(i) + [c_1, \infty)$ then $\tau_C(j) - \tau_C(i) \geq c_1$. If $\tau_C(j) - \tau_C(i) < C$ then $\tau_C(j) - \tau_C(i) = \tau(j) - \tau(i)$ and, so, $\tau(j) \in \tau(i) + [c_1, \infty)$. Otherwise $\tau_C(j) - \tau_C(i) = C$. Then, $\tau(j) - \tau(i) \geq C$. As $C > c_1$, we have that $\tau(j) \in \tau(i) + [c_1, \infty)$. □

We are now ready for Theorem 3.

Proof Let $\rho = (\sigma, \tau)$ be a timed state sequence and let $\rho_C = (\sigma_C, \tau_C)$ be as in Definition 5. By definition of ρ_C we have that ρ_C is C-bounded.

Assume w.l.o.g. that φ is in NNF. Let $sub(\varphi)$ be the set of all subformulae of φ. To prove this lemma, we argue by structural induction and show that for all $\varphi' \in sub(\varphi)$, if $(\rho, i) \models \varphi'$ then $(\rho_C, i) \models \varphi', i \in \mathbb{N}$.

In the base case, φ' is a propositional formula. Then, as $\sigma_C = \sigma$, we have that $(\rho, i) \models \varphi'$ implies $(\rho_C, i) \models \varphi', i \in \mathbb{N}$. Suppose that, for $\chi, \psi \in sub(\varphi)$ and $i \in \mathbb{N}$, $(\rho, i) \models \chi$ implies $(\rho_C, i) \models \chi$, and, $(\rho, i) \models \psi$ implies $(\rho_C, i) \models \psi$. We explain for $\bigcirc_I, \mathcal{U}_I$ and $\tilde{\mathcal{U}}_I$ (other cases are similar):

- φ' is of the form $\bigcirc_I \chi$: if $(\rho, i) \models \bigcirc_I \chi$ then $\tau(i+1) \in \tau(i) + I$ and $(\rho, i+1) \models \psi$. By the induction hypothesis, $(\rho_C, i+1) \models \psi$. By Proposition 3, $\tau_C(i+1) \in \tau_C(i) + I$. Then, $(\rho_C, i) \models \bigcirc_I \chi$.
- φ' is of the form $(\chi \, \mathcal{U}_I \, \psi)$: if $(\rho, i) \models (\chi \, \mathcal{U}_I \, \psi)$ then there is $k \in \mathbb{N}$ such that $k \geq i$, $\tau(k) \in \tau(i) + I$ and $(\rho, k) \models \psi$ and for all $j \in \mathbb{N}$, if $i \leq j < k$ then $(\rho, j) \models \chi$. By the induction hypothesis, $(\rho_C, k) \models \psi$ and $(\rho_C, j) \models \chi$, for all $j \in \mathbb{N}$ with $i \leq j < k$. By Proposition 3, $\tau_C(k) \in \tau_C(i) + I$. Then, $(\rho_C, i) \models (\chi \, \mathcal{U}_I \, \psi)$.
- φ' is of the form $(\chi \, \tilde{\mathcal{U}}_I \, \psi)$: if $(\rho, i) \models (\chi \, \tilde{\mathcal{U}}_I \, \psi)$ then either:

  1. for all $j \in \mathbb{N}$, if $\tau(j) \in \tau(i) + I$ then $(\rho, j) \models \psi$; or
  2. (if $c_1 > 0$) $(\rho, k) \models \chi$ for some $k \in \mathbb{N}$ such that $\tau(k) \in \tau(i) + [0, c_1 - 1]$, where c_1 is the left end-point of I; or
  3. $(\rho, l) \models \chi$ for some $l \in \mathbb{N}$ such that $\tau(l) \in \tau(i) + I$ and for all $l' \in \mathbb{N}$ such that $\tau(i) + c_1 \leq \tau(l') \leq \tau(l)$, we have $(\rho, l') \models \psi$.

  We make a case distinction. In Case (1), we have to establish that, for all $j \in \mathbb{N}$, if $\tau_C(j) \in \tau_C(i) + I$ then $(\rho_C, j) \models \psi$. By Proposition 3, if $\tau_C(j) \in \tau_C(i) + I$ then $\tau(j) \in \tau(i) + I$ and by induction hypothesis, $(\rho_C, j) \models \psi$, for all $j \in \mathbb{N}$ with $\tau(j) \in \tau(i) + I$. Therefore, $(\rho_C, i) \models (\chi \, \tilde{\mathcal{U}}_I \, \psi)$. Cases (2) and (3) can be proved with similar arguments. □

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 using propositional variables of the form s^n_m 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.[1]

[1] We write $min(l + k, 2 \cdot C)$ 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 $2 \cdot C$.

Definition 6 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 $i > 0$:

$$\sigma'_i = \sigma_i \cup \{s^1_k \mid \tau(i) - \tau(i-1) = k\} \cup$$
$$\{s^{j+1}_{min(l+k,2\cdot C)} \mid \tau(i) - \tau(i-1) = k \text{ and } s^j_l \in \sigma'_{i-1}\}$$

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

In Definition 6, if, for example, $\tau(2) - \tau(0) = 4$ then $(\sigma', 2) \models s^2_4$. Intuitively, the variable s^2_4 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 $\tau(1) - \tau(0) = 2$. By definition of σ', Lemma 1 is immediate (see definition of $\oplus_{\leq 1}$ and $\oplus_{\geq 1}$ in the Preliminaries).

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

1. $\bigcirc\Box \oplus_{=1} \Pi^1$, *for* $\Pi^1 = \{s^1_k \mid 1 \leq k \leq C\}$;
2. $\Box \oplus_{\leq 1} \Pi^i$, *for* $1 < i < C$ *and* $\Pi^i = \{s^i_j \mid i \leq j \leq 2 \cdot C\}$;
3. $\Box((\bigcirc s^1_k \wedge s^j_l) \rightarrow \bigcirc s^{j+1}_{min(l+k,2\cdot C)})$, *for* $\{s^1_k, s^j_l, s^{j+1}_{min(l+k,2\cdot C)}\} \subseteq \bigcup_{1\leq i<C} \Pi^i$.

Given a C-bounded timed state sequence $\rho = (\sigma, \tau)$, let $\sigma' = \sigma'_0\sigma'_1\ldots$ be as in Definition 6. Then, $(\sigma', 0) \models S_C$.

There is exactly one value k, $1 \leq k \leq C$, that is going to be equal to the sum of the time difference of the last state to the current. This is encoded by s^1_k (Point 1). Point 2 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 3 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 $l + k$.

Remark 1 To simplify the presentation, in this section we used variables of the form s^n_m, where n ranges from 1 to $C - 1$ and m ranges from 1 (or from 0 for the non-strict semantics) to $2 \cdot C$ (in fact $2 \cdot C - 1$ would be enough for the translation). As mentioned in Sect. 1, the exponential blow-up of the translations cannot be avoided. Though, we can have a slightly more succinct translation by encoding m in binary. So in our experiments, we encode variables of the form s^n_m with $n \cdot \log m$ variables, where for each n we represent m in binary. For example, s^2_4 is encoded with $s^2_2 \wedge \neg s^2_1 \wedge \neg s^2_0$.

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 7 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 $\sigma_i = \sigma'_i \setminus (\bigcup_{1\leq j<C} \Pi^j)$, for $i \in \mathbb{N}$, and:

$$\tau(i) = \begin{cases} 0 & \text{if } i = 0 \\ \tau(i-1) + k & \text{if } i > 0, (\sigma', i) \models s^1_k \end{cases}$$

Note that ρ, in particular, τ, in Definition 7 is well-defined because for every $i \in \mathbb{N}$ there is exactly one k such that $(\sigma', i) \models s^1_k$. Table 3 presents our translation of formulae of the form $\bigcirc_I \alpha$, $\alpha_1\, \mathcal{U}_I\, \beta$ or $\alpha\, \tilde{\mathcal{U}}_I\, \beta$, as in Tables 1 and 2. As shown in Table 3, we translate, for example, $\bigcirc_{[2,3]} p$ into $\bigcirc((s^1_2 \vee s^1_3) \wedge p)$. The following two propositions are useful for the proof of Theorem 4.

Table 3 Strict time difference translation from MTL to LTL where α, β are propositional logic formulae and $c_1, c_2 > 0$

MTL	Strict time difference translation
$(\bigcirc_{[c_1,\infty)}\alpha)^\sharp$	$\bigcirc((\bigvee_{c_1 \le i \le C} s_i^1) \wedge \alpha)$
$(\bigcirc_{[0,\infty)}\alpha)^\sharp$	$\bigcirc\alpha$
$(\bigcirc_{[c_1,c_2]}\alpha)^\sharp$	$\bigcirc((\bigvee_{c_1 \le i \le c_2} s_i^1) \wedge \alpha)$
$(\bigcirc_{[0,c_2]}\alpha)^\sharp$	$(\bigcirc_{[1,c_2]}\alpha)^\sharp$
$(\bigcirc_{[0,0]}\alpha)^\sharp$	**false**
$(\alpha\, \mathcal{U}_{[c_1,\infty)}\, \beta)^\sharp$	$\bigvee_{1 \le i \le c_1}(\bigcirc^i((\bigvee_{c_1 \le j \le c_1+C}\ s_j^i) \wedge (\alpha\, \mathcal{U}\, \beta)) \wedge (\bigwedge_{0 \le k < i} \bigcirc^k \alpha))$
$(\alpha\, \mathcal{U}_{[0,\infty)}\, \beta)^\sharp$	$(\alpha\, \mathcal{U}\, \beta)$
$(\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta)^\sharp$	$\bigvee_{1 \le i \le c_2}(\bigcirc^i((\bigvee_{c_1 \le j \le c_2}\ s_j^i) \wedge \beta) \wedge (\bigwedge_{0 \le k < i} \bigcirc^k \alpha))$
$(\alpha\, \mathcal{U}_{[0,c_2]}\, \beta)^\sharp$	$\beta \vee (\alpha\, \mathcal{U}_{[1,c_2]}\, \beta)^\sharp$
$(\alpha\, \mathcal{U}_{[0,0]}\, \beta)^\sharp$	β
$(\alpha\, \tilde{\mathcal{U}}_I\, \beta)^\sharp$	$\neg(\neg(\alpha)\, \mathcal{U}_I\, \neg(\beta))^\sharp$

Proposition 4 *Given a C-bounded timed state sequence $\rho = (\sigma, \tau)$, let σ' be as in Definition 6. For all $i < j \in \mathbb{N}$, if $\tau(j) - \tau(i) = m \le 2 \cdot C$ and $j - i = n < C$ then $(\sigma', j) \models s_m^n$.*

Proof In the base case $j = i + 1$, that is, $n = 1$. As ρ is C-bounded, $\tau(i+1) - \tau(i) = k$, for $1 \le k \le C$. Then, by Definition 6, $(\sigma', j) \models s_k^1$. Suppose that, for $j = i + n$, if $\tau(i+n) - \tau(i) = l$ then $(\sigma', i+n) \models s_l^n$ where $l \le 2 \cdot C$. In the induction step we consider $j = i + n + 1$. Let k' be such that $\tau(i+n+1) - \tau(i+n) = k'$. As $\tau(i+n) - \tau(i) = l$ and $\tau(i+n+1) - \tau(i+n) = k'$, we have that $\tau(i+n+1) - \tau(i) = l + k'$. By Definition 6, $(\sigma', i+n+1) \models s_{k'}^1$. By induction hypothesis, $(\sigma', j) \models s_l^n$. By Definition 6, if $l + k' \le 2 \cdot C$, then $(\sigma', i+n+1) \models s_{l+k'}^{n+1}$. □

Proposition 5 *Given a state sequence $\sigma' = \sigma'_0\sigma'_1 \ldots$ such that $(\sigma', 0) \models S_C$, let ρ be as in Definition 7. For all $i < j \in \mathbb{N}$, if $(\sigma', j) \models s_m^n$ with $j - i = n < C$ and $m \le 2 \cdot C$ then $\tau(j) - \tau(i) = m$.*

Proof In the base case $j = i + 1$, that is, $n = 1$. Assume that $(\sigma', i+1) \models s_k^1$. Then, by Definition 7, $\tau(i+1) - \tau(i) = k \le C$, $i \in \mathbb{N}$. Suppose that, for $j = i + n$, $(\sigma', i+n) \models s_m^n$, with $j - i = n < C$ and $m \le 2 \cdot C$, implies $\tau(i+n) - \tau(i) = m$. In the induction step we consider $j = i + n + 1$. Assume that $(\sigma', i+n+1) \models s_l^{n+1}$. By Points 1 and 2 of the definition of S_C, there is $1 \le k' \le C$ such that $(\sigma', i+n+1) \models s_{k'}^1$. As $(\sigma', i+n) \models s_m^n$ and $(\sigma', i+n+1) \models s_{k'}^1$, we have that, by Point 4 of the definition of S_C, $(\sigma', i+n+1) \models s_{min(m+k', 2\cdot C)}^{n+1}$. Also, by Point 3 of the definition, there is no $l \ne m + k'$ such that $(\sigma', i+n+1) \models s_l^{n+1}$. Then, we can assume that $l = min(m + k', 2 \cdot C)$. By induction hypothesis, for $m \le 2 \cdot C$, we have that $\tau(i+n) - \tau(i) = m$. By Definition 7, $\tau(i+n+1) - \tau(i+n) = k'$. Then, if $m + k' \le 2 \cdot C$ we have that $\tau(i+n+1) - \tau(i) = m + k'$. □

We now show Theorem 4, which states the correctness of our translation using time differences. One can use similar arguments to show Theorem 5.

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 φ is satisfiable if, and only if, there is a timed state sequence $\rho = (\sigma, \tau)$ such that $(\rho, 0) \models \varphi$. By Theorem 3, we can assume w.l.g. that ρ is C-bounded, where $C - 1$ is the largest constant in φ. Given a timed state sequence $\rho = (\sigma, \tau)$, let $\sigma' = \sigma'_0\sigma'_1 \ldots$ be as in Definition 6. By Lemma 1, $(\sigma', 0) \models S_C$. By definition of σ', for all propositional variables p_i occurring in σ and all $j \in \mathbb{N}$, we have $(\rho, j) \models p_i$ iff $(\sigma', j) \models p_i$. With an inductive argument one can show that, for any propositional formula α, $(\rho, j) \models \alpha$ iff $(\sigma', j) \models \alpha$.

Also, $\varphi^\sharp \wedge S_C$ is satisfiable if, and only if, there is a state sequence σ' such that $(\sigma', 0) \models \varphi^\sharp \wedge S_C$. Given σ', let $\rho = (\sigma, \tau)$ be a C-bounded timed state sequence as in Definition 7. By definition of σ, for all propositional variables p_i occurring in σ' but not in $\bigcup_{1 \leq i' < C} \Pi^{i'}$ and all $j \in \mathbb{N}$, we have $(\rho, j) \models p_i$ iff $(\sigma', j) \models p_i$. Clearly, for any propositional formula α, $(\rho, j) \models \alpha$ iff $(\sigma', j) \models \alpha$.

Thus, following Table 3, we only need to show correspondences between ψ_i and $\psi_i^\sharp$. This follows from Claims 4–6 below (other cases are similar), where $\rho = (\sigma, \tau)$ is a C-bounded timed state sequence. We argue that, for all natural numbers c_1 with $0 < c_1 \leq c_2 < C$, the following claims hold.

Claim 4 $(\rho, j) \models \bigcirc_{[c_1,c_2]}\alpha$ *iff* $(\sigma', j) \models (\bigcirc_{[c_1,c_2]}\alpha)^\sharp$.

Proof $(\rho, j) \models \bigcirc_{[c_1,c_2]}\alpha$, with $c_1 > 0$, iff $(\rho, j + 1) \models \alpha$ and $\tau(j + 1) \in \tau(j) + [c_1, c_2]$. By Definitions 6 and 7, this happens iff $(\sigma', j + 1) \models \alpha$ and $(\sigma', j + 1) \models \bigvee_{c_1 \leq i \leq c_2} s_i^1$, which is equivalent to $(\sigma', j) \models \bigcirc((\bigvee_{c_1 \leq i \leq c_2} s_i^1) \wedge \alpha)$.

Claim 5 $(\rho, j) \models (\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta)$ *iff* $(\sigma', j) \models (\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta)^\sharp$.

Proof $(\rho, j) \models \alpha \mathcal{U}_{[c_1,c_2]} \beta$, with $c_1 > 0$, iff there is $i \in \mathbb{N}$ such that $\tau(j+i) - \tau(j) \in [c_1, c_2]$, $(\rho, j + i) \models \beta$ and for all $n \in \mathbb{N}$, if $j \leq n < j + i$ then $(\rho, n) \models \alpha$. By Definitions 6 and 7, this happens iff $(\sigma', j + i) \models \beta$ and for all $n \in \mathbb{N}$, if $j \leq n < j + i$ then $(\sigma', n) \models \alpha$. By Propositions 4 and 5, for all $i \in \mathbb{N}$, with $0 < j - i = n < C$, and $m \leq 2 \cdot C$, we have that $(\sigma', j) \models s_m^n$ iff $\tau(j + i) - \tau(j) = m$. As $c_1 > 0$ and τ is strictly monotonically increasing, we have $1 \leq n \leq c_2$. Also, $c_1 \leq \tau(j + i) - \tau(j) = m \leq c_2 < C$,[2] so this happens iff $i \in \mathbb{N}$ is such that $(\sigma', j + i) \models \bigcirc^n(\bigvee_{c_1 \leq m \leq c_2} s_m^n) \wedge \beta$, with $n = j - i$, and $(\sigma', \ell) \models \alpha$ for all $\ell \in \mathbb{N}$ such that $j \leq \ell < j + i$. This is equivalent to $(\sigma', j) \models \bigvee_{1 \leq n \leq c_2}(\bigcirc^n((\bigvee_{c_1 \leq m \leq c_2} s_m^n) \wedge \beta) \wedge \bigwedge_{0 \leq k < n} \bigcirc^k \alpha)$.

Claim 6 $(\rho, j) \models \alpha\, \tilde{\mathcal{U}}_{[c_1,c_2]}\, \beta$ *iff* $(\sigma', j) \models \neg(\neg(\alpha)\, \mathcal{U}_{[c_1,c_2]}\, \neg(\beta))^\sharp$.

Proof $(\rho, j) \models \alpha\, \tilde{\mathcal{U}}_{[c_1,c_2]}\, \beta$ iff $(\rho, j) \models \neg(\neg(\alpha)\, \mathcal{U}_{[c_1,c_2]}\, \neg(\beta))$, by semantics of $\tilde{\mathcal{U}}$. By Claim 5, $(\rho, j) \not\models \neg(\alpha)\, \mathcal{U}_{[c_1,c_2]}\, \neg(\beta)$ iff $(\sigma', j) \not\models (\neg(\alpha)\, \mathcal{U}_{[c_1,c_2]}\, \neg(\beta))^\sharp$,
which is equivalent to $(\sigma', j) \models \neg(\neg(\alpha)\, \mathcal{U}_{[c_1,c_2]}\, \neg(\beta))^\sharp$. □

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:

$$\varphi^\sharp = p_0 \wedge \Box(p_0 \rightarrow (\bigcirc_{[2,3]} p_1)^\sharp) \wedge \Box(p_1 \rightarrow (\Diamond_{[1,2]} \neg q)^\sharp), \text{ where}$$

[2] The $2 \cdot C$ bound of Propositions 4 and 5 are useful for the proof of $\alpha\, \mathcal{U}_{[c,\infty)}\, \beta$.

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 $c_1, c_2 > 0$

MTL	Non-strict time difference translation
$(\bigcirc_{[k_1,\infty)}\alpha)^\sharp$	$\bigcirc((\bigvee_{k_1 \leq i \leq C} s_i^1) \wedge \alpha)$
$(\bigcirc_{[k_1,k_2]}\alpha)^\sharp$	$\bigcirc((\bigvee_{k_1 \leq i \leq k_2} s_i^1) \wedge \alpha)$
$(\alpha\, \mathcal{U}_{[c_1,\infty)}\, \beta)^\sharp$	$\alpha \wedge \bigcirc \bigvee_{1 \leq i \leq c_1}((\alpha \wedge s_0^1)\, \mathcal{U}^i\, (\neg s_0^1 \wedge \alpha),$ $(\neg s_0^1 \wedge (\bigvee_{c_1 \leq j \leq c_1 + C}\ s_j^i) \wedge (\alpha\, \mathcal{U}\, \beta)))$
$(\alpha\, \mathcal{U}_{[0,\infty)}\, \beta)^\sharp$	$(\alpha\, \mathcal{U}\, \beta)$
$(\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta)^\sharp$	$\alpha \wedge \bigcirc \bigvee_{1 \leq i \leq c_2}((\alpha \wedge s_0^1)\, \mathcal{U}^i\, (\neg s_0^1 \wedge \alpha),$ $(\neg s_0^1 \wedge (\bigvee_{c_1 \leq j \leq c_2}\ s_j^i) \wedge (\alpha\, \mathcal{U}_{[0,0]}\, \beta)^\sharp))$
$(\alpha\, \mathcal{U}_{[0,c_2]}\, \beta)^\sharp$	$(\alpha\, \mathcal{U}_{[0,0]}\, \beta)^\sharp \vee (\alpha\, \mathcal{U}_{[1,c_2]}\, \beta)^\sharp$
$(\alpha\, \mathcal{U}_{[0,0]}\, \beta)^\sharp$	$\beta \vee (\alpha \wedge \bigcirc((\alpha \wedge s_0^1)\, \mathcal{U}\, (\beta \wedge s_0^1)))$
$(\alpha\, \tilde{\mathcal{U}}_I\, \beta)^\sharp$	$\neg(\neg(\alpha)\, \mathcal{U}_I\, \neg(\beta))^\sharp$

$$(\bigcirc_{[2,3]} p_1)^\sharp = \bigcirc((\bigvee_{2 \leq i \leq 3} s_i^1) \wedge p_1) \text{ and}$$
$$(\Diamond_{[1,2]} \neg q)^\sharp = \bigvee_{1 \leq i \leq 2}(\bigcirc^i((\bigvee_{1 \leq j \leq 2}\ s_j^i) \wedge \neg q))$$

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

Non-Strict Semantics

We now show how we modify the Time Difference translation for non-strict timed state sequences. We extend the set $\Pi^1 = \{s_i^1 \mid 1 \leq i \leq C\}$ of propositional variables representing time differences with s_0^1, 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. For $m > 0$, the meaning of the variables of the form s_m^n 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 6, with the difference that, whenever $\tau(i) = \tau(i-1)$, we now make s_0^1 true in σ'_i and, for $m > 0$, we copy all variables of the form s_m^n in σ'_{i-1} to σ'_i. Let S'_C be the conjunction of the following:

1. $\bigcirc\Box \oplus_{=1} \Pi^1$, for $\Pi^1 = \{s_k^1 \mid 0 \leq k \leq C\}$;
2. $\Box \oplus_{\leq 1} \Pi^i$, for $1 < i < C$ and $\Pi^i = \{s_j^i \mid i \leq j \leq 2 \cdot C\}$;
3. $\Box((\bigcirc s_k^1 \wedge s_l^j) \rightarrow \bigcirc s_{min(l+k,2\cdot C)}^{j+1})$, for $\{s_k^1, s_l^j, s_{min(l+k,2\cdot C)}^{j+1}\} \subseteq \bigcup_{1 \leq i < C} \Pi^i$;
4. $\Box((\bigcirc s_0^1 \wedge s_l^j) \rightarrow \bigcirc s_l^j)$, for $s_l^j \in \bigcup_{1 \leq i < C} \Pi^i$ with $l > 0$.

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 4 which propagates the variables of the form s_m^n 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 7 (but now the time difference 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_m^n 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 $\varphi^\sharp = p_0 \wedge \bigwedge_i \Box(p_i \rightarrow \psi_i^\sharp)$ be the result of replacing each ψ_i in φ by $\psi_i^\sharp$ 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 6 and 7 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 s_0^1 to encode multiple states mapped to the same time point. □

6 Empirical Evaluation of the Translations

In order to empirically evaluate the translations, we have implemented them in SWI-Prolog and have used them together with the LTL satisfiability solvers `Aalta`, `Leviathan`, `LS4`, `LWB-MOD`, `LWB-SAT`, `NuSMV-BDD`, `NuSMV-SBMC`, `pltl (graph)`, `pltl (tree)`, `TRP++`, and `TSPASS`. The sources of the implementation of the translations and all auxiliary files necessary to replicate the evaluation can be found at [31].

`Aalta` [1] implements the obligation-based LTL satisfiability checking algorithm for finite and infinite traces devised by Li et al. [37]. We have performed experiments with both versions 1.2 and 2.0 of `Aalta`. As `Aalta` 2.0 was consistently several orders of magnitude faster than `Aalta` 1.2, we will only report the results for the latest version of `Aalta`.

`NuSMV` 2.6.0 [45] uses a reduction of the LTL satisfiability problem to the LTL model checking problem [15]. It is then possible to decide the latter problem either using a BDD-based algorithm (`NuSMV-BDD`) or a SAT-based algorithm (`NuSMV-SBMC`). We considered these to be two different solvers as they often exhibit different behaviours. We use `NuSMV-BDD` with the options `dynamic` for dynamic reordering and `elbwd` for backward image computation, for `NuSMV-SBMC` we have enabled the completeness check.

The Logics Workbench 1.1 (`LWB`) [39] contains implementations of two different LTL solvers. The first is a two-pass tableau-based decision procedure developed by Janssen [34] which underlies the `provable` and `satisfiable` functions of the `pltl` module of `LWB`. In the following we denote this procedure by `LWB-SAT`. The second is a one-pass tableau calculus by Schwendimann [50] which underlies the `model` function of the `pltl` module. In the following we denote this procedure by `LWB-MOD`. In contrast to `LWB-SAT`, `LWB-MOD` returns a model for satisfiable formulae. We again, consider these to be two different solvers as the underlying calculi are distinct.

`Leviathan` [36] is an LTL satisfiability checking and model building tool that implements a novel one-pass tableau calculus by Reynolds [11,49].

The `pltl` [47] system implements two tableau-based methods. The `tree` method is again based on Schwendimann's one-pass tableau calculus, the `graph` method is based on a one-pass and-or tree tableau calculus [26] resulting in a time complexity optimal decision

procedure for LTL. We denote the former by `pltl (tree)` and the latter by `pltl (graph)`. Neither of the two methods returns a model. In this section we will only report results for `pltl (graph)` as it always performs better than `pltl (tree)`.

`TRP++` 2.2 [54] is based on an ordered resolution calculus that operates on LTL formulae in a clausal normal form [32], while `TSPASS` [56] extends that calculus to monodic first-order linear time temporal logic over expanding domains [42]. We use `TSPASS` with the `ModelConstruction` option so that it returns a model for satisfiable formulae [41]. It should be noted that this model construction potentially requires significantly more resolution inferences to be performed than are necessary to just decide the satisfiability of a formula. Thus, the way we use `TSPASS` puts it as a distinct performance disadvantage over `TRP++` although both perform very similar inference steps when just deciding the satisfiability of an LTL formula.

`LS4` [40] is an LTL-prover based on labelled superposition with partial model guidance developed by Suda and Weidenbach [52]. It operates on LTL formulae in the same clausal normal form as `TRP++`. `LS4` optionally returns a model for satisfiable formulae.

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 $(\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta)$ or $\Diamond_{[c_1,c_2]}\beta$, 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. The hypothesis is that the Gap Translations lead to better behaviour of solvers on such formulae. 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 parametrised formulae $\theta^1_{b_1} := \Diamond_{[0,b_1]}p \wedge \Box_{[0,\infty)}\neg p$ for values of b_1 between 1 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 100 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 LTL solvers in the CPU time used to solve a particular formula. The experiments were conducted on a PC with Intel i7-2600 CPU @ 3.40 GHz and 16 GB main memory.

Recall that for the Time Difference Translations we encode variables of the form s^n_m with $n \cdot \log m$ variables, where for each n we represent m in binary, e.g., s^2_4 is encoded with $s^2_2 \wedge \neg s^2_1 \wedge \neg s^2_0$. In the following tables and graphs we denote this variant by 'bTD' to distinguish it from the variant used for related experiments in [33] and denoted by 'TD' there.

Tables 5 and 6 show the runtimes for the combination of our translations with the various LTL solvers on $\theta^1_{b_1}$, with values of b_1 between 1 and 10, and on $\theta^2_{b_2}$, with values of b_2 between 10 and 100 in steps of 10, respectively. An entry 'T/O' indicates that the timeout of 1000 CPU seconds was exceeded by an LTL solver while an entry 'Fail' indicates that the LTL solver encountered some other error condition and stopped before the timeout was reached. This is often a memory allocation error or some internal data structure running out of space.

Figures 4 and 5 show the same data in the form of 'heat maps' where different colours are used to represent different ranges of runtimes. This allows us to easily recognise significant differences in the performance of the various combinations of translations and LTL solver.

Table 5 Runtime, in CPU seconds, required to solve $\theta_{b_1}^1 = \Diamond_{[0,b_1]} p \wedge \Box_{[0,\infty)} \neg p$, $1 \leq b_1 \leq 10$, by a particular combination of a translation with respect to the strict/non-strict semantics and an LTL solver

b_1	TD Aalta	Gap Aalta	TD Leviathan	Gap Leviathan	TD LS4	Gap LS4	TD LWB MOD	Gap LWB MOD	TD LWB SAT	Gap LWB SAT
Strict semantics										
1	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**	0.01	**0.01**	0.01	**0.01**
2	0.01	**0.00**	T/O	**T/O**	0.00	**0.00**	0.01	**0.01**	9.78	**0.01**
3	0.02	**0.00**	T/O	**T/O**	0.00	**0.00**	0.31	**0.01**	Fail	**0.01**
4	0.07	**0.00**	T/O	**T/O**	0.00	**0.00**	5.29	**0.01**	Fail	**0.01**
5	2.17	**0.00**	T/O	**T/O**	0.00	**0.00**	58.61	**0.01**	Fail	**0.01**
6	2.35	**0.00**	T/O	**T/O**	0.01	**0.00**	T/O	**0.01**	Fail	**0.01**
7	Fail	**0.00**	T/O	**T/O**	0.03	**0.00**	T/O	**0.01**	Fail	**0.01**
8	Fail	**0.00**	T/O	**T/O**	0.05	**0.00**	T/O	**0.01**	Fail	**0.01**
9	Fail	**0.00**	T/O	**T/O**	0.09	**0.00**	T/O	**0.01**	Fail	**0.01**
10	Fail	**0.00**	T/O	**T/O**	0.15	**0.00**	T/O	**0.01**	T/O	**0.01**
Non-strict semantics										
1	0.02	**0.00**	T/O	**T/O**	0.00	**0.00**	0.06	**0.01**	6.42	**0.01**
2	0.09	**0.00**	T/O	**T/O**	0.00	**0.00**	36.95	**0.01**	Fail	**0.01**
3	1.56	**0.00**	T/O	**T/O**	0.02	**0.01**	T/O	**0.01**	Fail	**0.01**
4	4.58	**0.00**	T/O	**T/O**	0.09	**0.02**	T/O	**0.01**	Fail	**0.01**
5	4.09	**0.00**	T/O	**T/O**	0.33	**0.04**	T/O	**0.01**	Fail	**0.02**
6	40.48	**0.00**	T/O	**T/O**	0.99	**0.07**	T/O	**0.02**	Fail	**0.02**
9	Fail	**0.01**	T/O	**T/O**	15.38	**0.26**	T/O	**0.06**	Fail	**0.40**
10	Fail	**0.01**	T/O	**T/O**	60.83	**0.45**	T/O	**0.12**	Fail	**1.22**

Table 5 continued

b_1	TD NuSMV BDD	Gap NuSMV BDD	TD NuSMV SBMC	Gap NuSMV SBMC	TD pltl graph	Gap pltl graph	TD TRP++	Gap TRP++	TD TSPASS	Gap TSPASS
Strict semantics										
1	0.01	**0.00**	0.01	**0.00**	0.00	**0.00**	0.00	**0.00**	0.00	**0.00**
2	0.27	**0.00**	0.05	**0.01**	4.73	**0.00**	0.00	**0.00**	0.00	**0.00**
3	3.57	**0.00**	0.20	**0.01**	T/O	**0.00**	0.00	**0.00**	0.02	**0.00**
4	31.39	**0.00**	0.58	**0.02**	T/O	**0.00**	0.01	**0.00**	0.04	**0.00**
5	419.95	**0.00**	1.50	**0.02**	T/O	**0.00**	0.03	**0.00**	0.11	**0.00**
6	T/O	**0.00**	3.70	**0.02**	T/O	**0.00**	0.91	**0.00**	4.25	**0.00**
7	T/O	**0.01**	8.89	**0.03**	T/O	**0.00**	0.19	**0.00**	0.40	**0.00**
8	T/O	**0.01**	18.44	**0.04**	T/O	**0.00**	1.51	**0.00**	11.17	**0.00**
9	T/O	**0.01**	36.34	**0.05**	T/O	**0.00**	52.02	**0.00**	T/O	**0.00**
10	T/O	**0.01**	68.83	**0.07**	T/O	**0.00**	88.16	**0.00**	T/O	**0.00**
Non-strict semantics										
1	0.11	**0.00**	182.61	**0.05**	31.98	**0.00**	0.65	**0.00**	0.04	**0.00**
2	1.10	**0.00**	T/O	**0.11**	T/O	**0.00**	93.63	**0.02**	5.27	**0.02**
3	33.94	**0.00**	T/O	**0.23**	T/O	**0.00**	T/O	**0.05**	33.46	**0.03**
4	T/O	**0.01**	T/O	**0.48**	T/O	**0.00**	T/O	**0.10**	439.22	**0.04**
5	T/O	**0.01**	T/O	**1.01**	T/O	**0.00**	T/O	**0.18**	T/O	**0.07**
6	T/O	**0.01**	T/O	**1.82**	T/O	**0.01**	T/O	**0.28**	T/O	**0.10**
7	T/O	**0.01**	T/O	**4.13**	T/O	**0.04**	T/O	**0.43**	T/O	**0.14**
8	T/O	**0.01**	T/O	**5.06**	T/O	**0.10**	T/O	**0.59**	T/O	**0.19**
9	T/O	**0.01**	T/O	**12.16**	T/O	**0.26**	T/O	**0.84**	T/O	**0.25**
10	T/O	**0.02**	T/O	**17.11**	T/O	**0.75**	T/O	**1.11**	T/O	**0.33**

Table 6 Runtime, in CPU seconds, required to solve $\theta^2_{b_2} = \bigcirc_{[10,\infty)} p \wedge \bigcirc_{[b_2,\infty)} \neg p$, $10 \leq b_2 \leq 100$, by a particular combination of a translation with respect to the strict/non-strict semantics and an LTL solver

b_2	bTD+ Aalta	Gap+ Aalta	bTD+ Leviathan	Gap+ Leviathan	bTD+ LS4	Gap+ LS4	bTD+ LWB MOD	Gap+ LWB MOD	bTD+ LWB SAT	Gap+ LWB SAT
Strict semantics										
10	**0.00**	0.05	**0.00**	T/O	**0.00**	0.01	**0.01**	0.01	**0.01**	0.01
20	**0.00**	1.31	**T/O**	T/O	**0.00**	0.04	**0.01**	0.01	**0.01**	0.01
30	**0.00**	6.77	**T/O**	T/O	**0.00**	0.09	**0.01**	0.01	**0.01**	0.01
40	**0.00**	24.13	**T/O**	T/O	**0.00**	0.20	**0.01**	0.01	**0.01**	0.01
50	**0.00**	66.99	**T/O**	T/O	**0.00**	0.29	**0.01**	0.01	**0.01**	0.01
60	**0.00**	Fail	**T/O**	T/O	**0.00**	0.55	**0.01**	0.01	**0.01**	0.01
70	**0.00**	Fail	**T/O**	T/O	**0.00**	0.84	**0.01**	0.02	**0.01**	0.01
80	**0.00**	Fail	**T/O**	T/O	**0.00**	0.96	**0.02**	0.02	**0.01**	0.01
90	**0.00**	Fail	**T/O**	T/O	**0.00**	1.33	**0.01**	0.02	**0.01**	0.01
100	**0.00**	Fail	**T/O**	T/O	**0.00**	2.27	**0.01**	0.03	**0.01**	0.01
Non-strict semantics										
10	**0.00**	0.06	**0.00**	T/O	**0.00**	0.01	**0.01**	0.01	**0.01**	0.01
20	**0.00**	1.22	**Fail**	T/O	**0.00**	0.04	**0.01**	0.01	**0.01**	0.01
30	**0.00**	6.55	**Fail**	T/O	**0.00**	0.09	**0.01**	0.01	**0.01**	0.01
40	**0.00**	23.98	**Fail**	T/O	**0.00**	0.21	**0.01**	0.01	**0.01**	0.01
50	**0.00**	68.29	**Fail**	T/O	**0.00**	0.29	**0.01**	0.01	**0.01**	0.01
60	**0.00**	Fail	**Fail**	T/O	**0.00**	0.57	**0.01**	0.02	**0.01**	0.01
70	**0.00**	Fail	**Fail**	T/O	**0.00**	0.85	**0.01**	0.02	**0.01**	0.01
80	**0.00**	Fail	**Fail**	T/O	**0.00**	0.98	**0.01**	0.02	**0.01**	0.01
90	**0.00**	Fail	**Fail**	T/O	**0.00**	1.35	**0.01**	0.03	*0.02*	0.01
100	**0.00**	Fail	**Fail**	T/O	**0.00**	2.28	**0.01**	0.03	**0.01**	0.01

Table 6 continued

b_2	bTD+ NuSMV BDD	Gap+ NuSMV BDD	bTD+ NuSMV SBMC	Gap+ NuSMV SBMC	bTD+ pltl graph	Gap+ pltl graph	bTD+ TRP++	Gap+ TRP++	bTD+ TSPASS	Gap TSPASS
Strict semantics										
10	**0.00**	0.09	**0.00**	0.17	**0.00**	0.00	**0.00**	0.87	**0.00**	575.01
20	**0.01**	0.29	**0.01**	0.69	**0.00**	0.00	**0.00**	64.52	**0.00**	T/O
30	**0.00**	0.65	**0.01**	1.94	**0.00**	0.00	**0.00**	441.27	**0.01**	T/O
40	**0.02**	1.66	**0.02**	4.42	**0.00**	0.00	**0.00**	93.33	**0.02**	T/O
50	**0.02**	3.31	**0.02**	8.81	**0.00**	0.00	**0.00**	T/O	**0.04**	T/O
60	**0.01**	3.17	**0.01**	15.82	**0.00**	0.00	**0.00**	T/O	**0.08**	T/O
70	**0.30**	5.47	**0.07**	27.38	*0.01*	0.00	**0.00**	T/O	**0.18**	T/O
80	**0.21**	23.27	**0.06**	43.86	*0.01*	0.00	**0.00**	T/O	**0.27**	T/O
90	**0.20**	9.54	**0.05**	61.91	*0.01*	0.00	**0.00**	T/O	**0.40**	T/O
100	**0.09**	11.96	**0.04**	97.12	**0.00**	0.00	**0.00**	T/O	**0.59**	T/O
Non-strict semantics										
10	**0.00**	0.11	**0.01**	0.19	**0.00**	0.00	**0.00**	0.87	**0.00**	597.09
20	**0.01**	0.49	**0.01**	0.76	**0.00**	0.00	**0.00**	65.61	**0.00**	T/O
30	**0.01**	0.99	**0.00**	2.07	**0.00**	0.00	**0.00**	499.20	**0.01**	T/O
40	**0.07**	1.31	**0.02**	4.64	**0.00**	0.00	**0.00**	96.15	**0.02**	T/O
50	**0.02**	1.89	**0.02**	9.79	**0.00**	0.00	**0.00**	T/O	**0.04**	T/O
60	**0.01**	7.77	**0.01**	16.47	**0.00**	0.00	**0.00**	T/O	**0.07**	T/O
70	**0.32**	6.61	**0.06**	27.67	*0.01*	0.00	**0.00**	T/O	**0.17**	T/O
80	**0.21**	21.40	**0.06**	43.68	*0.01*	0.00	**0.00**	T/O	**0.27**	T/O
90	**0.10**	10.21	**0.05**	66.00	*0.02*	0.00	**0.00**	T/O	**0.40**	T/O
100	**0.09**	22.18	**0.04**	95.39	**0.00**	0.00	**0.00**	T/O	**0.59**	T/O

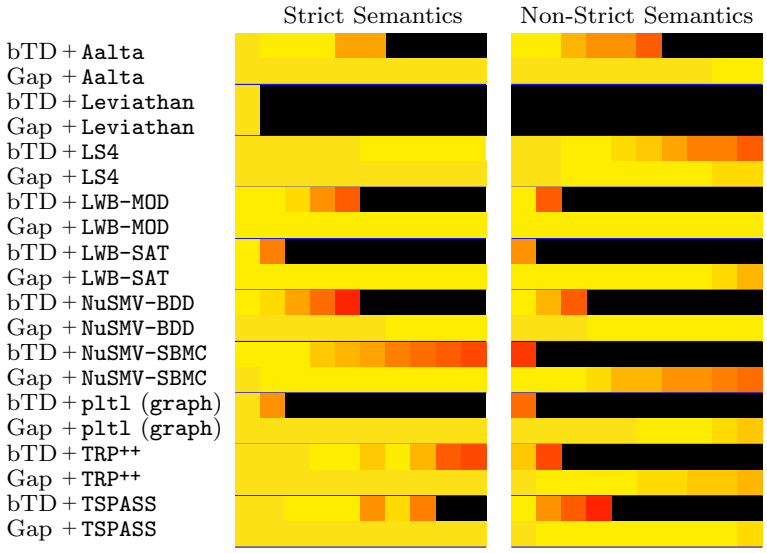

Fig. 4 Heat map of the runtimes on $\theta^1_{b_1} = \Diamond_{[0,b_1]} p \wedge \Box_{[0,\infty)} \neg p$. Each rectangle represents the runtime in CPU seconds of a particular combination of a translation with respect to the strict/non-strict semantics and an LTL solver on one instance of $\theta^1_{b_1}$, with runtimes given in colours as follows: < 0.01 sec; > 0.01 sec, ≤ 0.25 sec; > 0.25 sec, ≤ 0.50 sec; > 0.50 sec, ≤ 1 sec; > 1 sec, ≤ 2 sec; > 2 sec, ≤ 4 sec; > 4 sec, ≤ 8 sec; > 8 sec, ≤ 16 sec; > 16 sec, ≤ 32 sec; > 32 sec, ≤ 64 sec; > 64 sec, ≤ 125 sec; > 125 sec, ≤ 250 sec; > 250 sec, ≤ 500 sec; > 500 sec, ≤ 1000 sec; T/O, OoM or Fail

The results in Table 5 and Fig. 4 confirm our hypothesis that on $\theta^1_{b_1}$ the Gap translations will lead to better performance than the Time Difference translations. For both the strict and non-strict semantics, for all LTL solvers, and for all values of b_1 considered, the Gap translations result in better or equal performance than the Time Difference translations (highlighted in bold in the Table 5), with the vast majority of entries indicating a significantly worse performance for the Time Difference translations.

Regarding the strict versus the non-strict translations, we would expect that a non-strict translation results in worse performance compared to the corresponding strict translation as the search space for an LTL solver is larger. This is indeed the case for all provers.

The best performing combination is Gap+`Aalta`. The most 'robust' LTL solver, in the sense of producing the lowest total runtime across all translations and all instances of $\theta^1_{b_1}$ considered, is `LS4`.

The results in Table 6 and Fig. 5 also largely confirm our hypothesis that on $\theta^2_{b_2}$ the Time Difference translations will lead to better performance than the Gap translations. In total we have considered ten LTL solvers with two translations under two different semantics, giving us twenty points of comparison between a Time Difference translation and a Gap translation, each over ten instances of $\theta^2_{b_2}$. In Table 6 we have highlighted in bold all instances where a Time Difference translation leads to better or equal performance and in italics all instances where the opposite is true. We see that for only seven instances the latter is the case, six of them for `pltl (graph)`.

The reason is that most of background theories S_C and S'_C that form part of the Time Difference Translations turn out not to be relevant to the (un)satisfiability of $(\theta^2_{b_2})^\sharp$. Most

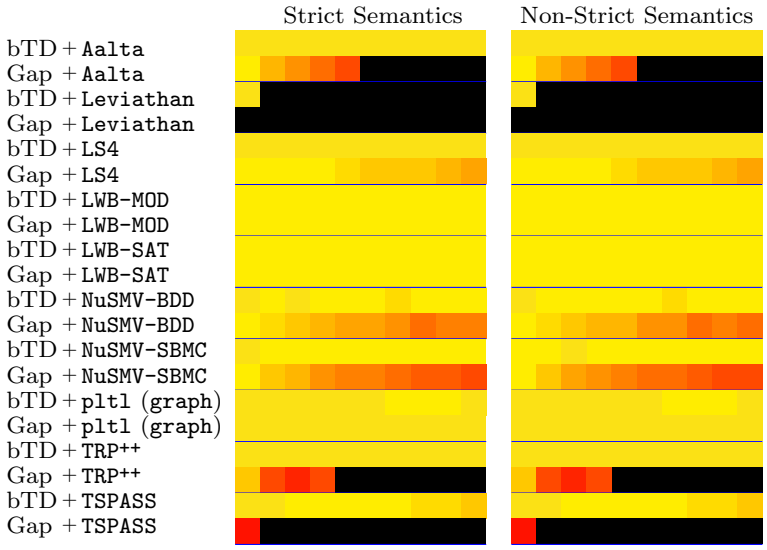

Fig. 5 Heat map of the runtimes on $\theta^2_{b_2} := \bigcirc_{[10,\infty)} p \wedge \bigcirc_{[c_1,\infty)} \neg p$. Each rectangle represents the runtime in CPU seconds of a particular combination of a translation with respect to the strict/non-strict semantics and an LTL solver on one instance of $\theta^2_{b_2}$, with runtimes given in colours as follows:

< 0.01 sec | > 0.01 sec, ≤ 0.25 sec | > 0.25 sec, ≤ 0.50 sec | > 0.50 sec, ≤ 1 sec
> 1 sec, ≤ 2 sec | > 2 sec, ≤ 4 sec | > 4 sec, ≤ 8 sec | > 8 sec, ≤ 16 sec
> 16 sec, ≤ 32 sec | > 32 sec, ≤ 64 sec | > 64 sec, ≤ 125 sec | > 125 sec, ≤ 250 sec
> 250 sec, ≤ 500 sec | > 500 sec, ≤ 1000 sec | T/O, OoM or Fail

provers appear to be able to derive a contradiction without too many inferences involving S_C or S'_C, while `pltl (graph)` does 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 `pltl (graph)` also performs better with the Time Difference Translations than with the Gap Translations on all instances.

`Aalta`, `LS4` and `TRP++` with a Time Difference translation as well as `pltl (graph)` with a Gap translation can solve all instances in negligible time. The most 'robust' LTL solver on $\theta^2_{b_2}$ over all translations is `pltl (graph)`.

Overall the evaluation presented in this section confirms that no translation is always strictly 'better' than the other. It will depend on the characteristics of the formula and sometimes on the LTL solver whether a Time Difference translation or a Gap translation results in better performance and therefore in a greater likelihood of deciding a formula in reasonable time. However, the experiments also show the significant performance improvements that can be achieved by choosing the 'right' translation and, arguably, the simplicity of the Gap translations means that more often than not it is the translation to use. The experimental results presented in the next section provide additional support for this.

7 Examples Revisited

In the following we revisit the examples given in Sect. 3. For the foraging robots, traffic lights and multiprocessor job-shop scheduling scenarios, we have stated a range of properties that

we will now attempt to prove using our translations together with the LTL solvers `Aalta`, `Leviathan`, `LS4`, `LWB-MOD`, `LWB-SAT`, `pltl (graph)`, `NuSMV-BDD`, `NuSMV-SBMC`, `TRP++`, and `TSPASS`. The collection of MTL formulae for the specifications and properties can again be found at [31].

7.1 Foraging Robots

For the foraging robots example we use the specification given in Sect. 3.1 with the maximum time that the robot can be spent resting $T_r = \infty$, the maximum time that the robot be spent searching for food $T_s = 5$, and the maximum time that the robot needs to get home with our without food $T_d = 2$. Let Σ_{FR} denote the conjunction of the formulae in the specification. We want to prove that the following properties follow from Σ_{FR}:

- Having left home the robot will eventually reach home again.
$$\Box_{[0,\infty)}(\mathit{leavingHome} \rightarrow \Diamond_{[0,\infty)}\mathit{resting}) \qquad (F_1)$$
- The maximum time the robot needs to return home is x
$$\Box_{[0,\infty)}(\mathit{leavingHome} \rightarrow \Diamond_{[0,x]}\mathit{resting}) \qquad (F_2^x)$$
The lowest value x for which we can prove that (F_2^x) follows from Σ_{FR} is the maximum time the robot needs to return home. For our choice of T_d, T_r and T_s that lowest value of x is 9 and in our experiments we consider F_2^8 and F_2^9, i.e., $x = 8$ and $x = 9$, respectively.
- The minimum time the robot needs to return home is y.
$$\Box_{[0,\infty)}(\mathit{leavingHome} \rightarrow \Box_{[0,y-1]}\neg\mathit{resting}) \qquad (F_3^y)$$
The highest value y for which we can prove that (F_3^y) follows from Σ_{FR} can be proved is the minimum time the robot needs to return home. This value is independent of the choice of T_d, T_r and T_s, instead it depends on the lower bounds in relevant intervals. For Σ_{FR} that highest value of y is 3 and in our experiments we consider F_3^3 and F_3^4, i.e., $y = 3$ and $y = 4$, respectively.

To prove that $\Sigma_{FR} \rightarrow \varphi$ holds, we check the unsatisfiability of $\Sigma_{FR} \wedge \neg\varphi$. In Table 7 we denote $\Sigma_{FR} \wedge \neg\varphi$ by $\bar{\varphi}$ in the first column. The second column indicates whether $\bar{\varphi}$ is satisfiable (S) or unsatisfiable (U). The remaining columns of the table show the runtime for each translation and each prover. We again have used a timeout of 1000 CPU seconds and as before 'T/O' in the table indicates that a prover was not able to determine the satisfiability of the translated formula within that time while 'Fail' indicates that an LTL solver encountered some other error condition and stopped before the timeout was reached.

On all formulae each LTL solver, except `TSPASS`, performs better with the Gap Translation than with the Time Difference Translation. Only `LS4` is able to decide the satisfiability of all formulae with both translations. `LWB-MOD` and `NuSMV-BDD` are able to decide the satisfiability of all formulae with the Gap Translation but fail to decide any of the formulae with the Time Difference Translation. The latter is also true for `LWB-SAT`, `pltl`, and `TRP++`. `TSPASS` is the only system showing better performance with the Time Difference Translation, it can solve two unsatisfiable formulae with that translation but none with the Gap Translation.

7.2 Traffic Lights

For the traffic lights example we use the specification given in Sect. 3.2, denoted by Σ_{TR}. We want to prove that the following properties follow from Σ_{TR}:

Table 7 Runtime, in CPU seconds, required to solve Foraging Robot properties by a particular combination of a translation and an LTL solver

	S	bTD+ Aalta	Gap+ Aalta	bTD+ Leviathan	Gap+ Leviathan	bTD+ LS4	Gap+ LS4	bTD+ LWB MOD	Gap+ LWB MOD	bTD+ LWB SAT	Gap+ LWB SAT
$\bar{F}_1$	U	549.99	Fail	T/O	T/O	0.04	0.15	T/O	275.55	T/O	23.33
$\bar{F}_2^8$	S	Fail	6.50	Fail	Fail	0.31	0.01	T/O	6.15	T/O	8.67
$\bar{F}_2^9$	U	Fail	22.17	Fail	Fail	3.84	0.01	T/O	5.21	T/O	32.23
$\bar{F}_3^3$	U	0.92	0.60	T/O	T/O	0.01	0.00	T/O	2.93	T/O	21.49
$\bar{F}_3^4$	S	T/O	1.24	T/O	T/O	0.02	0.00	T/O	4.83	T/O	T/O
	S	bTD+ NuSMV BDD	Gap+ NuSMV BDD	bTD+ NuSMV SBMC	Gap+ NuSMV SBMC	bTD+ pltl graph	Gap+ pltl graph	bTD+ TRP++	Gap+ TRP++	bTD+ TSPASS	Gap+ TSPASS
$\bar{F}_1$	U	T/O	1.29	Fail	Fail	T/O	T/O	Fail	T/O	96.95	T/O
$\bar{F}_2^8$	S	T/O	2.84	19.70	0.81	T/O	T/O	T/O	T/O	T/O	T/O
$\bar{F}_2^9$	U	T/O	2.66	Fail	T/O	T/O	T/O	T/O	T/O	T/O	T/O
$\bar{F}_3^3$	U	T/O	1.32	Fail	Fail	T/O	T/O	Fail	T/O	6.73	T/O
$\bar{F}_3^4$	S	T/O	1.81	1.90	0.20	T/O	T/O	Fail	T/O	T/O	T/O

– Infinitely often each traffic light will be green.

$$\Box_{[0,\infty)}\Diamond_{[0,\infty)}g_1 \wedge \Box_{[0,\infty)}\Diamond_{[0,\infty)}g_2 \qquad (T_1)$$

– If a car is detected on road R1 then the wait will be at most x_1 time units.

$$\Box_{[0,\infty)}(detect \rightarrow \Diamond_{[0,x]}g_1) \qquad (T_2^x)$$

This property holds for $x \geq 3$ but not $0 \leq x < 3$. In our experiments we consider T_2^2 and T_2^3, i.e., $x = 2$ and $x = 3$, respectively.

– If traffic light 2 is currently red then it will be green within 1 to y time units.

$$\Box_{[0,\infty)}(r_2 \rightarrow \Diamond_{[1,y]}g_2) \qquad (T_3^y)$$

This property holds for $y \geq 7$ but not $0 \leq y < 7$. Consequently, we will consider T_3^6 and T_3^7, i.e., $y = 6$ and $y = 7$.

– If a traffic light is currently amber or red, then a car has to wait at most z time units until the traffic light will be green.

$$\Box_{[0,\infty)}(((a_1 \vee r_1) \wedge detect) \rightarrow \Diamond_{[0,z]}g_1) \wedge \Box_{[0,\infty)}((a_2 \vee r_2) \rightarrow \Diamond_{[0,z]}g_2) \qquad (T_4^z)$$

This property should hold for $z \geq 9$ but not $0 \leq z < 9$ and we focus on T_4^8 and T_4^9.

As for the Foraging Robots example, in order to prove $\Sigma_{TR} \rightarrow \varphi$ for an MTL formula φ, we check the unsatisfiability of $\Sigma_{TR} \wedge \neg\varphi$. In Table 8 we denote $\Sigma_{TR} \wedge \neg\varphi$ by $\bar{\varphi}$ in the first column. The second column indicates whether $\bar{\varphi}$ is satisfiable (S) or not (U). The remaining columns show the runtime for each translation and each prover. We have for the first time included results for `pltl (tree)`, as there are instances on which it performs better than `pltl (graph)`.

As we can see, these formulae are much more challenging than those for the Foraging Robots example. This is not surprising if, for instance, one considers the state space of potential models for our specifications. A robot could only be in one of nine states. In contrast, each of the two traffic lights can be in one of four states plus we have a sensor that can be in one of two states. That already gives us 32 global states, only two of which are straightforwardly excluded by the constraint that not both traffic lights can be green at the same time. Furthermore, for the Foraging Robots example we only had one 'defined' propositional variable *searching* that had a rather straightforward definition while in the Traffic Lights example we have $change_1$ and $change_2$ that are subject to much more complicated temporal constraints.

As a consequence, only `NuSMV-BDD` with the Gap Translation and `LS4` with the Time Difference Translation can solve all formulae. For `LS4` the Gap Translation still results in better performance on all formulae that can be solved by both translations within the time limit. The same is true for `NuSMV-SBMC` and for `Aalta`. `LWB-SAT`, `pltl`, `TRP++` and `TSPASS` fail to solve a single formula with either translation within the timeout.

7.3 Multiprocessor Job-Shop Scheduling

Regarding the Multiprocessor Job-shop Scheduling problems (MJS problems for short) 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 an 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 Eqs. 15, 16, 22 and 23, 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.

Table 8 Runtime, in CPU seconds, required to solve traffic lights properties by a particular combination of a translation and an LTL solver

	S	bTD+ Aalta	Gap+ Aalta	bTD+ Leviathan	Gap+ Leviathan	bTD+ LS4	Gap+ LS4	bTD+ LWB MOD	Gap+ LWB MOD	bTD+ LWB SAT	Gap+ LWB SAT
T_1	S	T/O	Fail	Fail	T/O	3.05	T/O	T/O	T/O	T/O	T/O
T_2^2	S	T/O	Fail	T/O	T/O	0.18	0.18	T/O	T/O	T/O	T/O
T_2^3	U	4.74	3.14	Fail	Fail	0.01	0.00	T/O	T/O	T/O	T/O
T_3^6	S	Fail	T/O	Fail	Fail	0.10	0.23	T/O	37.43	T/O	T/O
T_3^7	U	Fail	Fail	Fail	Fail	4.39	0.82	T/O	T/O	T/O	T/O
T_4^8	S	Fail	T/O	Fail	Fail	1.08	0.60	T/O	T/O	T/O	T/O
T_4^9	U	Fail	Fail	Fail	Fail	132.07	0.70	T/O	T/O	T/O	T/O
	S	bTD+ NuSMV BDD	Gap+ NuSMV BDD	bTD+ NuSMV SBMC	Gap+ NuSMV SBMC	bTD+ pltl graph	Gap+ pltl graph	bTD+ TRP++	Gap+ TRP++	bTD+ TSPASS	Gap+ TSPASS
T_1	S	T/O	25.09	Fail	T/O	Fail	T/O	T/O	T/O	T/O	T/O
T_2^2	S	T/O	48.35	4.42	0.27	Fail	T/O	T/O	T/O	T/O	T/O
T_2^3	U	T/O	9.31	Fail	Fail	Fail	T/O	T/O	T/O	T/O	T/O
T_3^6	S	T/O	29.93	3.94	1.31	T/O	T/O	T/O	T/O	T/O	T/O
T_3^7	U	T/O	26.17	Fail	Fail	T/O	T/O	T/O	T/O	T/O	T/O
T_4^8	S	T/O	203.83	13.71	3.43	T/O	T/O	T/O	T/O	T/O	T/O
T_4^9	U	T/O	83.85	Fail	Fail	T/O	T/O	T/O	T/O	T/O	T/O

It is an interesting characteristic of the MTL formulae for MJS instances that unsatisfiable MTL formulae are smaller than satisfiable MTL formulae. A time bound t that is too small leads to an MJS problem for which no schedule exists that can complete all jobs within the given time bound while a sufficiently large time bound guarantees that such a schedule can be found. By our encodings, a small time bound results in a smaller LTL formula than a large time bound. The difference in size is more pronounced for the Time Difference Translations than for the Gap Translations. This is in contrast to, say, random 3CNF formulae or random modal K3CNF formulae that are often used for benchmarking SAT solvers or modal logic provers, where unsatisfiable formulae tend to be larger than satisfiable formulae.

For our experiments we created 36 MJS problems with a 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 5. We then constructed MTL formulae for the strict semantics and the non-strict semantics according to the formalisations in Sect. 3.3. Finally, we used a combination of one the encodings with an LTL solver to determine the satisfiability of each MTL formula, or if the LTL solver has that capability, attempt to find a model for it, with a timeout of 1000 CPU seconds.

If an LTL solver reports that the formula resulting from an MJS problem with n jobs, k machines and time bound t is satisfiable, then a schedule exists that completes all n jobs on k machines within t time points. However, this provides us with no information about what that schedule might be. If for a satisfiable formula the LTL solver also returns a model, then the information at which time point $startRun_{j_i m_l}$ (or $startRun j_i \wedge m_l$ for the non-strict semantics) becomes true indicates when a particular job has to be started on a particular machine in order to complete all jobs within the time bound, thus, the model provides us with a schedule.

Among the systems included in our experiments, only `Leviathan`, `LS4`, `LWB-MOD`, `NuSMV-BDD`, `NuSMV-SBMC`, and `TSPASS` can produce models and have been used with the command line options that require them to do so.

Tables 9, 10, 11, 12, 13, 14 show the results for the formalisation of MJS problems in MTL with strict semantics and non-strict semantics, respectively. The first three columns in each table show the job list J, the number k of machines and the time bound t. The fourth column indicates whether the corresponding MTL formula is satisfiable (S) or unsatisfiable (U). The remaining columns of the table show the runtime for each translation and each prover ('T/O' indicates timeout, 'Fail' indicates that the solver encountered some error condition before reaching the timeout). Figure 6 shows the same data in the form of a heat map.

Regarding the formalisation of MJS problems in the strict semantics, we see in Fig. 6 and Tables 9, 10, 11 for the 296 MJS problem for which an LTL solver can determine its satisfiability with either the Gap or the Time Difference Translation, for 183 problems the Gap Translation results in better performance, for 32 problems the opposite is true, while for 81 problems we get the same performance. The two LTL solvers for which the Time Difference Translation regularly results in better performance are `TRP++` and `TSPASS`. The Gap Translation together with `LS4` offers the best performance for every instance.

Regarding the formalisation of MJS problems in the non-strict semantics, the most striking observation we can make from Fig. 6 and from the data in Tables 12, 13, 14 is how much more challenging the corresponding LTL satisfiability problems are for all the provers. In total there are 36 MJS problems on which we benchmarked 11 LTL solvers / settings with 2 different translations producing a total of 792 results. Of these 449 were timeouts, compared to 206 timeouts for MJS problems in the strict semantics. In a further 130 instances an LTL solver failed, that is, it encountered a memory allocation error, exhausted the limited size of some internal data structure, or produced some other internal failure before reaching the timeout.

Table 9 Runtime, in CPU seconds, required to solve MJS problems by a particular combination of a translation with respect to the strict semantics and an LTL solver (`LWB-MOD`, `LWB-SAT`, `pltl (graph)`, `pltl (tree)`)

J	k	t	S	bTD+ LWB MOD	Gap+ LWB MOD	bTD+ LWB SAT	Gap+ LWB SAT	bTD+ pltl graph	Gap+ pltl graph	bTD+ pltl tree	Gap+ pltl tree
1	1	0	U	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00
1	1	1	S	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00
1	1	2	S	0.05	0.01	89.47	0.01	6.71	0.00	6.68	0.00
1	1	3	S	1.83	0.01	Fail	0.01	T/O	0.00	T/O	0.00
1,2	1	0	U	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00
1,2	1	1	U	0.01	0.01	0.34	0.01	964.39	0.11	281.09	0.06
1,2	1	2	U	0.42	0.01	Fail	0.02	T/O	1.65	T/O	0.67
1,2	1	3	S	92.83	0.02	Fail	0.03	T/O	3.29	T/O	1.27
1,2	2	0	U	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00
1,2	2	1	U	0.25	0.01	7.45	0.05	T/O	0.84	T/O	1.44
1,2	2	2	S	0.17	0.02	Fail	0.08	T/O	11.80	T/O	18.04
1,2	2	3	S	2.34	0.03	Fail	0.11	T/O	11.79	T/O	18.17
1,1,2	2	0	U	0.01	0.01	0.02	0.01	0.00	0.00	0.00	0.00
1,1,2	2	1	U	0.02	0.02	639.29	9.82	T/O	370.68	T/O	521.65
1,1,2	2	2	S	0.36	0.12	Fail	4.15	T/O	T/O	T/O	T/O
1,1,2	2	3	S	4.04	0.11	Fail	4.07	T/O	T/O	T/O	T/O
1,1,2	3	0	U	0.01	0.01	0.02	0.02	0.00	0.00	0.00	0.00
1,1,2	3	1	U	1.42	0.03	T/O	133.82	T/O	T/O	T/O	T/O
1,1,2	3	2	S	0.82	0.41	Fail	243.63	T/O	T/O	T/O	T/O
1,1,2	3	3	S	3.39	0.18	Fail	247.49	T/O	T/O	T/O	T/O
1,1,2,2	2	0	U	0.01	0.01	0.07	0.06	0.00	0.00	0.00	0.00
1,1,2,2	2	1	U	0.02	0.02	T/O	55.56	T/O	T/O	T/O	T/O
1,1,2,2	2	2	U	6.34	0.73	Fail	207.58	T/O	T/O	T/O	T/O
1,1,2,2	2	3	S	T/O	1.83	Fail	286.03	T/O	T/O	T/O	T/O

Table 9 continued

J	k	t	S	bTD+ LWB MOD	Gap+ LWB MOD	bTD+ LWB SAT	Gap+ LWB SAT	bTD+ pltl graph	Gap+ pltl graph	bTD+ pltl tree	Gap+ pltl tree
1,1,2,2	3	0	U	0.01	0.01	0.03	0.10	0.00	0.00	0.00	0.00
1,1,2,2	3	1	U	0.06	0.05	T/O	T/O	T/O	T/O	T/O	T/O
1,1,2,2	3	2	S	3.75	3.85	Fail	T/O	T/O	T/O	T/O	T/O
1,1,2,2	3	3	S	T/O	2.72	Fail	T/O	T/O	T/O	T/O	T/O
1,2,2,3	2	1	U	0.03	0.02	T/O	80.00	T/O	T/O	T/O	T/O
1,2,2,3	2	2	U	532.16	0.73	Fail	279.51	T/O	T/O	T/O	T/O
1,2,2,3	2	3	U	T/O	4.06	Fail	580.51	T/O	T/O	T/O	T/O
1,2,2,3	2	4	S	T/O	7.71	T/O	T/O	T/O	T/O	T/O	T/O
1,2,3,4	2	2	U	T/O	0.68	T/O	176.96	T/O	T/O	T/O	T/O
1,2,3,4	2	3	U	T/O	4.28	T/O	544.73	T/O	T/O	T/O	T/O
1,2,3,4	2	4	U	T/O	12.98	T/O	T/O	T/O	T/O	T/O	T/O
1,2,3,4	2	5	S	T/O	28.03	T/O	T/O	T/O	T/O	T/O	T/O

Table 10 Runtime, in CPU seconds, required to solve MJS problems by a particular combination of a translation with respect to the strict semantics and an LTL solver (`Aalta`, `Leviathan`, `TRP++`, `TSPASS`)

J	k	t	S	bTD+ `Aalta`	Gap+ `Aalta`	bTD+ Leviathan	Gap+ Leviathan	bTD+ `TRP++`	Gap+ `TRP++`	bTD+ `TSPASS`	Gap+ `TSPASS`
1	1	0	U	0.00	0.00	0.62	20.03	0.00	0.00	0.00	0.00
1	1	1	S	0.00	0.00	2.59	T/O	0.00	0.22	0.02	1.11
1	1	2	S	0.02	0.00	Fail	T/O	0.06	0.39	0.13	2.00
1	1	3	S	0.21	0.00	Fail	T/O	T/O	0.52	1.01	7.35
1,2	1	0	U	0.00	0.00	Fail	T/O	0.00	0.00	0.01	0.01
1,2	1	1	U	0.00	0.00	Fail	T/O	0.00	0.00	0.01	0.03
1,2	1	2	U	0.05	0.01	Fail	T/O	1.85	0.16	15.10	0.08
1,2	1	3	S	1.77	0.01	Fail	T/O	T/O	T/O	0.02	259.68
1,2	2	0	U	0.00	0.00	Fail	T/O	0.00	0.00	0.01	0.01
1,2	2	1	U	0.00	0.00	Fail	T/O	0.01	0.04	0.04	0.08
1,2	2	2	S	0.07	0.01	Fail	T/O	739.38	T/O	46.35	34.80
1,2	2	3	S	0.14	0.01	Fail	T/O	T/O	T/O	477.05	118.83
1,1,2	2	0	U	0.00	0.00	Fail	T/O	0.00	0.00	0.01	0.02
1,1,2	2	1	U	0.01	0.01	Fail	T/O	0.00	0.08	0.02	0.20
1,1,2	2	2	S	0.08	0.03	Fail	T/O	T/O	T/O	129.76	943.98
1,1,2	2	3	S	0.49	0.02	Fail	T/O	T/O	T/O	T/O	T/O
1,1,2	3	0	U	0.00	0.00	Fail	T/O	0.00	0.00	0.01	0.02
1,1,2	3	1	U	0.01	0.02	Fail	T/O	0.07	0.33	0.10	0.28
1,1,2	3	2	S	0.07	0.04	Fail	T/O	T/O	T/O	157.11	150.70
1,1,2	3	3	S	0.76	0.03	Fail	T/O	T/O	T/O	T/O	640.67
1,1,2,2	2	0	U	0.00	0.00	Fail	T/O	0.00	0.01	0.02	0.03
1,1,2,2	2	1	U	0.01	0.01	Fail	T/O	0.06	0.20	0.03	0.53
1,1,2,2	2	2	U	0.11	0.07	Fail	T/O	T/O	38.11	T/O	1.83
1,1,2,2	2	3	S	1.01	0.06	Fail	T/O	T/O	T/O	T/O	T/O

Table 10 continued

J	k	t	S	bTD+ Aalta	Gap+ Aalta	bTD+ Leviathan	Gap+ Leviathan	bTD+ TRP++	Gap+ TRP++	bTD+ TSPASS	Gap+ TSPASS
1,1,2,2	3	0	U	0.00	0.00	Fail	T/O	0.00	0.02	0.02	0.04
1,1,2,2	3	1	U	0.02	0.02	Fail	T/O	0.31	0.96	0.09	0.76
1,1,2,2	3	2	S	0.28	0.07	Fail	T/O	T/O	T/O	T/O	T/O
1,1,2,2	3	3	S	0.34	0.05	Fail	T/O	T/O	T/O	T/O	T/O
1,2,2,3	2	1	U	0.04	0.01	Fail	T/O	0.13	0.29	0.06	0.73
1,2,2,3	2	2	U	0.52	0.03	Fail	T/O	T/O	382.45	T/O	1.31
1,2,2,3	2	3	U	7.53	0.24	Fail	T/O	T/O	T/O	T/O	591.02
1,2,2,3	2	4	S	13.00	0.28	Fail	T/O	T/O	T/O	T/O	T/O
1,2,3,4	2	2	U	0.94	0.06	Fail	Fail	T/O	T/O	T/O	2.26
1,2,3,4	2	3	U	11.19	0.27	Fail	Fail	T/O	T/O	T/O	15.00
1,2,3,4	2	4	U	T/O	0.66	Fail	Fail	T/O	T/O	T/O	T/O
1,2,3,4	2	5	S	116.90	0.99	Fail	Fail	T/O	T/O	T/O	T/O

Table 11 Runtime, in CPU seconds, required to solve MJS problems by a particular combination of a translation with respect to the strict semantics and an LTL solver (`LS4`, `NuSMV-BDD`, `NuSMV-SBMC`)

J	k	t	S	bTD+ LS4	Gap+ LS4	bTD+ NuSMV BDD	Gap+ NuSMV BDD	bTD+ NuSMV SBMC	Gap+ NuSMV SBMC
1	1	0	U	0.00	0.00	0.10	0.01	0.01	0.01
1	1	1	S	0.00	0.00	0.10	0.04	0.04	0.02
1	1	2	S	0.00	0.00	0.61	0.04	0.08	0.02
1	1	3	S	0.00	0.00	3.85	0.05	0.21	0.03
1,2	1	0	U	0.00	0.00	1.40	0.15	0.05	0.02
1,2	1	1	U	0.00	0.00	1.59	0.26	0.08	0.04
1,2	1	2	U	0.00	0.00	2.70	0.26	0.11	0.06
1,2	1	3	S	0.01	0.00	11.08	0.46	0.43	0.11
1,2	2	0	U	0.00	0.00	1.92	0.39	0.07	0.04
1,2	2	1	U	0.00	0.00	1.92	0.59	0.10	0.06
1,2	2	2	S	0.00	0.00	3.66	0.59	0.17	0.12
1,2	2	3	S	0.01	0.00	62.05	0.72	0.32	0.12
1,1,2	2	0	U	0.00	0.00	3.49	1.24	0.10	0.06
1,1,2	2	1	U	0.00	0.00	4.74	1.90	0.14	0.10
1,1,2	2	2	S	0.00	0.00	15.07	1.66	0.29	0.19
1,1,2	2	3	S	0.01	0.00	156.74	2.91	0.50	0.19
1,1,2	3	0	U	0.00	0.00	6.00	3.24	0.12	0.09
1,1,2	3	1	U	0.00	0.00	11.06	4.14	0.18	0.13
1,1,2	3	2	S	0.00	0.00	17.95	8.98	0.28	0.26
1,1,2	3	3	S	0.00	0.00	T/O	13.21	0.47	0.26
1,1,2,2	2	0	U	0.00	0.00	8.47	3.39	0.14	0.10
1,1,2,2	2	1	U	0.00	0.00	16.15	5.63	0.21	0.14
1,1,2,2	2	2	U	0.00	0.00	34.44	15.34	0.29	0.20
1,1,2,2	2	3	S	0.02	0.00	T/O	34.70	0.81	0.37

Table 11 continued

J	k	t	S	bTD+ LS4	Gap+ LS4	bTD+ NuSMV BDD	Gap+ NuSMV BDD	bTD+ NuSMV SBMC	Gap+ NuSMV SBMC
1,1,2,2	3	0	U	0.00	0.00	14.54	8.59	0.18	0.14
1,1,2,2	3	1	U	0.00	0.00	52.47	14.63	0.27	0.21
1,1,2,2	3	2	S	0.01	0.01	79.05	26.54	0.52	0.39
1,1,2,2	3	3	S	0.01	0.01	T/O	34.65	0.78	0.40
1,2,2,3	2	1	U	0.00	0.00	83.95	7.89	0.37	0.16
1,2,2,3	2	2	U	0.00	0.00	131.15	12.01	0.52	0.22
1,2,2,3	2	3	U	0.01	0.00	197.42	16.62	0.67	0.29
1,2,2,3	2	4	S	0.03	0.01	T/O	21.11	1.78	0.48
1,2,3,4	2	2	U	0.01	0.00	T/O	14.41	0.93	0.23
1,2,3,4	2	3	U	0.04	0.00	888.57	15.89	1.20	0.30
1,2,3,4	2	4	U	0.17	0.02	T/O	48.37	1.50	0.39
1,2,3,4	2	5	S	0.15	0.02	T/O	21.88	3.94	0.61

Table 12 Runtime, in CPU seconds, required to solve MJS problems by a particular combination of a translation with respect to the non-strict semantics and an LTL solver (`LWB-MOD`, `LWB-SAT`, `pltl (graph)`, `pltl (tree)`)

J	k	t	S	bTD+ LWB MOD	Gap+ LWB MOD	bTD+ LWB SAT	Gap+ LWB SAT	bTD+ pltl graph	Gap+ pltl graph	bTD+ pltl tree	Gap+ pltl tree
1	1	0	U	12.28	0.01	Fail	0.01	T/O	0.00	T/O	0.00
1	1	1	S	9.87	0.01	Fail	0.01	T/O	0.01	T/O	0.00
1	1	2	S	11.46	0.01	Fail	0.01	T/O	0.02	T/O	0.01
1	1	3	S	T/O	0.02	Fail	0.01	T/O	0.02	T/O	0.01
1,2	1	0	U	T/O	0.02	Fail	0.08	T/O	1.10	T/O	0.63
1,2	1	1	U	T/O	0.11	Fail	1.68	T/O	15.98	T/O	7.74
1,2	1	2	U	T/O	0.52	Fail	19.16	T/O	120.69	T/O	35.48
1,2	1	3	S	T/O	0.32	Fail	46.44	T/O	586.95	T/O	83.87
1,2	2	0	U	T/O	1.03	Fail	20.39	T/O	T/O	T/O	T/O
1,2	2	1	U	T/O	10.38	Fail	763.74	T/O	T/O	T/O	T/O
1,2	2	2	S	T/O	3.89	Fail	396.55	T/O	T/O	T/O	T/O
1,2	2	3	S	T/O	13.55	Fail	738.24	T/O	T/O	T/O	T/O
1,1,2	2	0	U	T/O	12.50	Fail	T/O	T/O	T/O	T/O	T/O
1,1,2	2	1	U	T/O	782.01	Fail	T/O	T/O	T/O	T/O	T/O
1,1,2	2	2	S	T/O	162.42	Fail	T/O	T/O	T/O	T/O	T/O
1,1,2	2	3	S	T/O	982.67	Fail	T/O	T/O	T/O	T/O	T/O
1,1,2	3	0	U	T/O	T/O	Fail	Fail	T/O	T/O	T/O	T/O
1,1,2	3	1	U	T/O	T/O	Fail	Fail	T/O	T/O	T/O	T/O
1,1,2	3	2	S	T/O	T/O	Fail	Fail	T/O	T/O	T/O	T/O
1,1,2	3	3	S	T/O	T/O	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2,2	2	0	U	T/O	51.83	Fail	T/O	T/O	T/O	T/O	T/O
1,1,2,2	2	1	U	T/O	T/O	Fail	T/O	T/O	T/O	T/O	T/O
1,1,2,2	2	2	U	T/O	T/O	Fail	T/O	T/O	T/O	T/O	T/O
1,1,2,2	2	3	S	T/O	T/O	Fail	T/O	T/O	T/O	T/O	T/O

Table 12 continued

J	k	t	S	bTD+ LWB MOD	Gap+ LWB MOD	bTD+ LWB SAT	Gap+ LWB SAT	bTD+ pltl graph	Gap+ pltl graph	bTD+ pltl tree	Gap+ pltl tree
1,1,2,2	3	0	U	T/O	T/O	Fail	Fail	T/O	T/O	T/O	T/O
1,1,2,2	3	1	U	T/O	T/O	Fail	Fail	T/O	T/O	T/O	T/O
1,1,2,2	3	2	S	T/O	T/O	Fail	Fail	T/O	T/O	T/O	T/O
1,1,2,2	3	3	S	T/O	T/O	Fail	Fail	T/O	T/O	T/O	T/O
1,2,2,3	2	1	U	T/O	T/O	Fail	T/O	T/O	T/O	T/O	T/O
1,2,2,3	2	2	U	T/O	T/O	Fail	T/O	T/O	T/O	T/O	T/O
1,2,2,3	2	3	U	T/O	T/O	Fail	T/O	T/O	T/O	T/O	T/O
1,2,2,3	2	4	S	T/O	T/O	Fail	T/O	T/O	T/O	T/O	T/O
1,2,3,4	2	2	U	T/O	T/O	Fail	T/O	T/O	T/O	T/O	T/O
1,2,3,4	2	3	U	T/O	T/O	Fail	T/O	T/O	T/O	T/O	T/O
1,2,3,4	2	4	U	T/O	T/O	Fail	T/O	T/O	T/O	T/O	T/O
1,2,3,4	2	5	S	T/O	T/O	Fail	T/O	T/O	T/O	T/O	T/O

Table 13 Runtime, in CPU seconds, required to solve MJS problems by a particular combination of a translation with respect to the non-strict semantics and an LTL solver (`Aalta`, `Leviathan`, `TRP++`, `TSPASS`)

J	k	t	S	bTD+ `Aalta`	Gap+ `Aalta`	bTD+ Leviathan	Gap+ Leviathan	bTD+ `TRP++`	Gap+ `TRP++`	bTD+ `TSPASS`	Gap+ `TSPASS`
1	1	0	U	0.08	0.01	Fail	T/O	T/O	143.99	448.09	80.30
1	1	1	S	0.01	0.02	T/O	T/O	T/O	T/O	485.57	93.04
1	1	2	S	0.24	0.00	T/O	T/O	T/O	T/O	T/O	461.46
1	1	3	S	0.90	0.01	T/O	T/O	T/O	T/O	T/O	T/O
1,2	1	0	U	2.41	0.06	T/O	Fail	T/O	T/O	T/O	T/O
1,2	1	1	U	9.77	0.20	T/O	Fail	T/O	T/O	T/O	T/O
1,2	1	2	U	13.57	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,2	1	3	S	10.62	0.10	T/O	Fail	T/O	T/O	T/O	T/O
1,2	2	0	U	3.54	0.29	T/O	Fail	T/O	T/O	T/O	T/O
1,2	2	1	U	3.87	3.92	T/O	Fail	T/O	T/O	T/O	T/O
1,2	2	2	S	Fail	0.72	T/O	Fail	T/O	T/O	T/O	T/O
1,2	2	3	S	3.95	0.18	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2	2	0	U	62.73	2.69	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2	2	1	U	326.11	11.76	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2	2	2	S	Fail	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2	2	3	S	316.40	1.14	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2	3	0	U	16.59	6.19	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2	3	1	U	Fail	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2	3	2	S	42.01	1.24	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2	3	3	S	9.95	T/O	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2,2	2	0	U	309.85	3.43	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2,2	2	1	U	408.69	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2,2	2	2	U	Fail	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2,2	2	3	S	Fail	Fail	T/O	Fail	T/O	T/O	T/O	T/O

Table 13 continued

J	k	t	S	bTD+ Aalta	Gap+ Aalta	bTD+ Leviathan	Gap+ Leviathan	bTD+ TRP++	Gap+ TRP++	bTD+ TSPASS	Gap+ TSPASS
1,1,2,2	3	0	U	487.85	223.50	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2,2	3	1	U	Fail	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2,2	3	2	S	Fail	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,1,2,2	3	3	S	Fail	5.38	T/O	Fail	T/O	T/O	T/O	T/O
1,2,2,3	2	1	U	T/O	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,2,2,3	2	2	U	Fail	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,2,2,3	2	3	U	Fail	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,2,2,3	2	4	S	Fail	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,2,3,4	2	2	U	T/O	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,2,3,4	2	3	U	Fail	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,2,3,4	2	4	U	Fail	Fail	T/O	Fail	T/O	T/O	T/O	T/O
1,2,3,4	2	5	S	Fail	Fail	T/O	Fail	T/O	T/O	T/O	T/O

Table 14 Runtime, in CPU seconds, required to solve MJS problems by a particular combination of a translation with respect to the non-strict semantics and an LTL solver (`LS4`, `NuSMV-BDD`, `NuSMV-SBMC`)

J	k	t	S	bTD+ `LS4`	Gap+ `LS4`	bTD+ `NuSMV BDD`	Gap+ `NuSMV BDD`	bTD+ `NuSMV SBMC`	Gap+ `NuSMV SBMC`
1	1	0	U	0.00	0.00	2.66	0.01	Fail	3.05
1	1	1	S	0.00	0.00	3.01	0.05	0.12	0.03
1	1	2	S	0.00	0.00	2.51	0.05	0.13	0.03
1	1	3	S	0.01	0.00	52.36	0.05	0.35	0.03
1,2	1	0	U	0.03	0.00	61.51	0.26	Fail	T/O
1,2	1	1	U	0.05	0.02	421.65	0.28	Fail	T/O
1,2	1	2	U	0.10	34.42	33.54	0.33	Fail	T/O
1,2	1	3	S	0.05	0.01	T/O	0.61	0.70	0.12
1,2	2	0	U	0.07	0.08	34.48	0.79	Fail	Fail
1,2	2	1	U	0.46	1.71	44.12	1.59	Fail	T/O
1,2	2	2	S	0.06	0.03	324.58	4.15	0.62	0.13
1,2	2	3	S	0.04	0.03	614.42	5.95	0.63	0.14
1,1,2	2	0	U	116.57	0.94	T/O	2.97	Fail	T/O
1,1,2	2	1	U	T/O	T/O	T/O	6.03	Fail	T/O
1,1,2	2	2	S	0.10	0.06	217.56	31.27	0.94	0.25
1,1,2	2	3	S	0.08	0.06	T/O	76.94	0.96	0.27
1,1,2	3	0	U	T/O	T/O	87.65	6.48	Fail	T/O
1,1,2	3	1	U	T/O	T/O	T/O	25.01	Fail	T/O
1,1,2	3	2	S	0.10	0.12	T/O	277.16	1.11	0.36
1,1,2	3	3	S	0.11	0.09	T/O	628.20	1.13	0.37
1,1,2,2	2	0	U	T/O	3.21	245.82	5.35	Fail	T/O
1,1,2,2	2	1	U	T/O	T/O	581.67	22.54	Fail	T/O
1,1,2,2	2	2	U	T/O	T/O	T/O	78.75	Fail	T/O
1,1,2,2	2	3	S	0.22	0.16	T/O	409.10	1.51	0.52

Table 14 continued

J	k	t	S	bTD+ LS4	Gap+ LS4	bTD+ NuSMV BDD	Gap+ NuSMV BDD	bTD+ NuSMV SBMC	Gap+ NuSMV SBMC
1,1,2,2	3	0	U	T/O	T/O	T/O	70.22	Fail	Fail
1,1,2,2	3	1	U	T/O	T/O	T/O	253.77	Fail	T/O
1,1,2,2	3	2	S	0.33	0.28	T/O	T/O	1.75	0.72
1,1,2,2	3	3	S	0.22	0.40	T/O	T/O	1.78	0.70
1,2,2,3	2	1	U	T/O	T/O	T/O	17.76	Fail	Fail
1,2,2,3	2	2	U	T/O	T/O	T/O	103.68	Fail	T/O
1,2,2,3	2	3	U	T/O	T/O	T/O	T/O	Fail	T/O
1,2,2,3	2	4	S	0.76	0.47	T/O	T/O	3.73	0.78
1,2,3,4	2	2	U	T/O	T/O	T/O	127.44	Fail	T/O
1,2,3,4	2	3	U	T/O	T/O	T/O	T/O	Fail	T/O
1,2,3,4	2	4	U	T/O	T/O	T/O	T/O	Fail	T/O
1,2,3,4	2	5	S	2.37	2.90	T/O	T/O	9.71	1.02

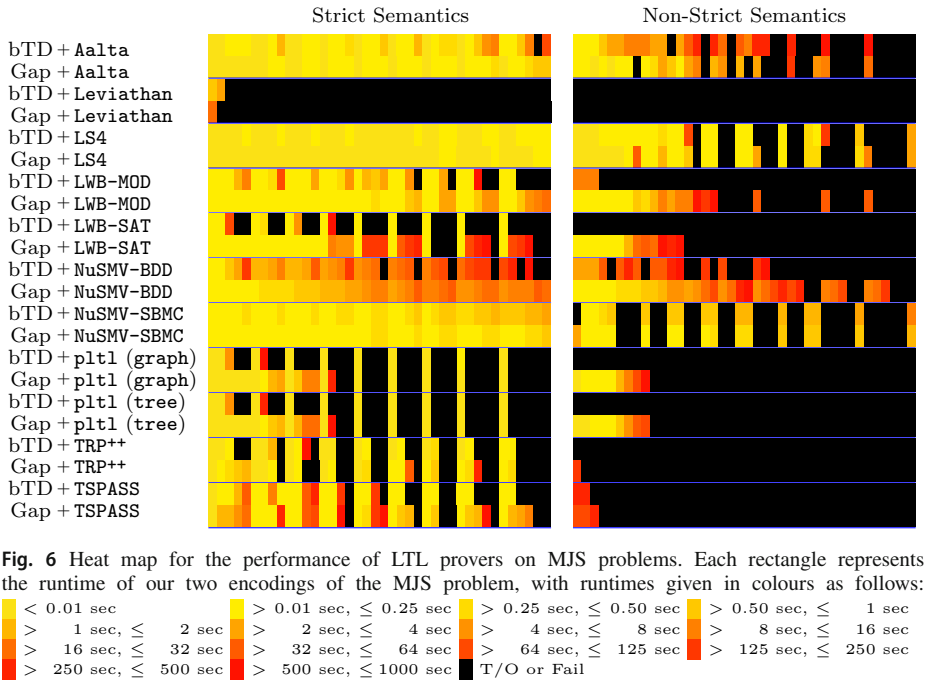

Fig. 6 Heat map for the performance of LTL provers on MJS problems. Each rectangle represents the runtime of our two encodings of the MJS problem, with runtimes given in colours as follows: < 0.01 sec; > 0.01 sec, ≤ 0.25 sec; > 0.25 sec, ≤ 0.50 sec; > 0.50 sec, ≤ 1 sec; > 1 sec, ≤ 2 sec; > 2 sec, ≤ 4 sec; > 4 sec, ≤ 8 sec; > 8 sec, ≤ 16 sec; > 16 sec, ≤ 32 sec; > 32 sec, ≤ 64 sec; > 64 sec, ≤ 125 sec; > 125 sec, ≤ 250 sec; > 250 sec, ≤ 500 sec; > 500 sec, ≤ 1000 sec; T/O or Fail

This compares to 53 failures for the strict semantics. So, in both categories the numbers more than double, resulting only in 213 instances where a combination of a translation with an LTL solver succeeded within the time given compared to 533 instances with the strict semantics.

Overall, the Non-Strict Gap Translation results in better performance than the Non-Strict Time Difference Translation: For the 139 MJS problem where an LTL solver can decide its satisfiability with either the Gap or Time Difference Translation, for 125 of those problems the Non-Strict Gap Translations results in better performance, for 11 problems the opposite is true while for just 3 problems the two encodings result in the same performance. The combination of the Non-Strict Gap Translation and `NuSMV-BDD` solves the most problems within the time given, 29 out of 36. Of the seven MJS problems that `NuSMV-BDD` with the Non-Strict Gap Translation cannot solve, four can be solved by both `LS4` and `NuSMV-SBMC`, interestingly, with both the Non-Strict Gap Translation and Non-Strict Time Difference Translation. The best results can be obtained by a combination of the Non-Strict Gap Translation with a portfolio consisting of `LS4` and `NuSMV-BDD`.

It is worth pointing out that MJS is not our intended domain of application for the translations we have presented. There is a lot of symmetry in these problems that a more specialised solver can take advantage of while LTL solvers are oblivious to it, e.g., the order in which jobs are executed on a particular machine does not affect the overall time to completion nor does the choice of machine. However it provides us with a parameterised problem set on which we can experiment for both semantics. For problems containing less symmetry that can naturally be formalised in MTL, and for which scheduling or planning is just one part of it, the approach used here can be beneficial.

In summary, the experimental results presented in this section provide further insights into the differing behaviour of the Gap Translations and Time Difference Translations on a range of examples.

8 Branching Metric Temporal Logic

Combinations of branching temporal logic with quantitative time have been proposed with continuous semantics [4,8,44] and with discrete semantics [20]. The logic we propose, which we call CTL_M, is closely related to the logic Complete Real-Time[3] Computation Tree Logic (CRTCTL), presented in [20]. The authors show that the complexity of the satisfiability problem for CRTCTL is 2EXPTIME-complete. In the following, we show that the satisfiability problem for CTL_M is in 2EXPTIME via a translation to CTL. We then explain how to translate CRTCTL to CTL_M, which implies that our upper bound is tight. In this section, our main goal is to introduce CTL_M and show complexity results for this logic. For this purpose we only need to provide one type of translation. We show that the 'gap' translation can be adapted to the branching time setting, under the strict semantics. One could similarly adapt the time difference translation and the non-strict semantics from the linear to the branching case.

We introduce the syntax and semantics of CTL and CTL_M. Let $\mathcal{P}$ be a (countably infinite) set of propositional symbols. Well formed formulae in CTL are formed according to the rule:

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

where $p \in \mathcal{P}$ and $\mathbf{Q} \in \{\mathbf{E}, \mathbf{A}\}$.

As usual, the structure of a CTL model has the shape of an infinite tree. For the purposes of this work, a *tree* is a pair $T = (W, <)$ such that (i) '$<$' is a binary irreflexive and transitive relation over a non-empty set of nodes W and (ii) for all $w \in W$ the set $\{w' \in W \mid w' < w\}$ is linearly ordered by '$<$'. It is assumed that T has only one root (i.e. minimal element) denoted by ε. A path π in a tree $T = (W, <)$ starting at a node $w \in W$ is a maximal sequence $w_0 w_1 \ldots$ with $w_0 = w$ and $w_i < w_{i+1}$, for all $i \geq 0$. We use $\pi[i]$ to denote w_i. In the following we assume that our trees have a (countably) infinite number of nodes and that our paths are infinite sequences of nodes. The semantics of CTL is defined as follows (we omit propositional cases).

Definition 8 A *state tree* $\varrho = (T, \sigma)$ is a pair consisting of a tree $T = (W, <)$ and a function $\sigma : W \to 2^{\mathcal{P}}$ that maps every $w \in W$ to a set of propositional symbols in $\mathcal{P}$.

$$\begin{aligned}
(\varrho, w) &\models \mathbf{E}\bigcirc\varphi && \text{iff } \exists\pi \text{ such that } \pi[0] = w, (\varrho, \pi[1]) \models \varphi \\
(\varrho, w) &\models \mathbf{E}(\varphi \,\mathcal{U}\, \psi) && \text{iff } \exists\pi \text{ such that } \pi[0] = w, \exists k \in \mathbb{N} : (\varrho, \pi[k]) \models \psi; \\
& && \quad \text{and } \forall j \in \mathbb{N}, (0 \leq j < k) \text{ implies } (\varrho, \pi[j]) \models \varphi \\
(\varrho, w) &\models \mathbf{A}\bigcirc\varphi && \text{iff } \forall\pi \text{ with } \pi[0] = w, (\varrho, \pi[1]) \models \varphi \\
(\varrho, w) &\models \mathbf{A}(\varphi \,\mathcal{U}\, \psi) && \text{iff } \forall\pi \text{ with } \pi[0] = w, \exists k \in \mathbb{N} : (\varrho, \pi[k]) \models \psi; \\
& && \quad \text{and } \forall j \in \mathbb{N}, (0 \leq j < k) \text{ implies } (\varrho, \pi[j]) \models \varphi
\end{aligned}$$

Further connectives are defined as: $\mathbf{true} \equiv \neg(\mathbf{false})$, $\mathbf{Q}(\mathbf{true}\,\mathcal{U}\,\varphi) \equiv \mathbf{Q}(\Diamond\varphi)$, $\mathbf{A}(\Diamond\varphi) \equiv \neg\mathbf{E}(\Box\neg\varphi)$, $\mathbf{E}(\Diamond\varphi) \equiv \neg\mathbf{A}(\Box\neg\varphi)$, where $\mathbf{Q} \in \{\mathbf{E}, \mathbf{A}\}$. A state tree $\varrho = (T, \sigma)$ is a *model* of a CTL formula φ iff $(\varrho, \varepsilon) \models \varphi$. A CTL formula φ is *satisfiable* iff there exists a model of φ and it is *valid* iff every state tree is a model of it. CTL_M formulae are constructed in a way similar to CTL with the difference that path specific operators are constrained by intervals, as in MTL. Well formed formulae in CTL_M are formed according to the rule:

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

where $p \in \mathcal{P}$ and $\mathbf{Q} \in \{\mathbf{E}, \mathbf{A}\}$. The semantics of CTL_M is defined as follows (we omit propositional cases).

Definition 9 A *timed state tree* $\varrho = (T, \sigma, \tau)$ over $(\mathbb{N}, <)$ is a triple consisting of a tree $T = (W, <)$, a function $\sigma : W \to 2^{\mathcal{P}}$ that maps every $w \in W$ to a set of propositional

[3] Here *real-time* refers to real time systems, not to the real numbers.

symbols in $\mathcal{P}$; and a function $\tau : W \to \mathbb{N}$ that maps every $w \in W$ to a time point $\tau(w)$ such that, for all paths π in T, $\tau(\pi[0]) < \tau(\pi[1])$.

The semantics of temporal connectives is then defined as follows.

$(\varrho, w) \models \mathbf{E}\bigcirc_I \varphi$ iff $\exists \pi$ such that $\pi[0] = w$, $(\varrho, \pi[1]) \models \varphi$ and $\tau(\pi[1]) \in \tau(w) + I$

$(\varrho, w) \models \mathbf{E}(\varphi\, \mathcal{U}_I\, \psi)$ iff $\exists \pi$ with $\pi[0] = w$, $\exists k \in \mathbb{N} : \tau(\pi[k]) \in \tau(w) + I$ and $(\varrho, \pi[k]) \models \psi$; and $\forall j \in \mathbb{N}$, $(0 \leq j < k)$ implies $(\varrho, \pi[j]) \models \varphi$

$(\varrho, w) \models \mathbf{A}\bigcirc_I \varphi$ iff $\forall \pi$ such that $\pi[0] = w$, we have $(\varrho, \pi[1]) \models \varphi$ and $\tau(\pi[1]) \in \tau(w) + I$

$(\varrho, w) \models \mathbf{A}(\varphi\, \mathcal{U}_I\, \psi)$ iff $\forall \pi$ with $\pi[0] = w$, $\exists k \in \mathbb{N} : \tau(\pi[k]) \in \tau(w) + I$ and $(\varrho, \pi[k]) \models \psi$; and $\forall j \in \mathbb{N}$, $(0 \leq j < k)$ implies $(\varrho, \pi[j]) \models \varphi$

Further connectives are defined as: $\mathbf{true} \equiv \neg(\mathbf{false})$, $\mathbf{Q}(\mathbf{true}\, \mathcal{U}_I\, \varphi) \equiv \mathbf{Q}(\Diamond_I \varphi)$, $\mathbf{A}(\Diamond_I \varphi) \equiv \neg\mathbf{E}(\Box_I \neg\varphi)$, $\mathbf{E}(\Diamond_I \varphi) \equiv \neg\mathbf{A}(\Box_I \neg\varphi)$, where $\mathbf{Q} \in \{\mathbf{E}, \mathbf{A}\}$. A timed state tree $\varrho = (T, \sigma, \tau)$ is a *model* of a $\mathrm{CTL}_{\mathsf{M}}$ formula φ iff $(\varrho, \varepsilon) \models \varphi$. A $\mathrm{CTL}_{\mathsf{M}}$ formula φ is *satisfiable* iff there exists a model of φ. A $\mathrm{CTL}_{\mathsf{M}}$ formula φ is in *Negation Normal Form (NNF)* iff the negation operator ($\neg$) occurs only in front of propositional symbols. To transform a $\mathrm{CTL}_{\mathsf{M}}$ formula into Negation Normal Form, we use $\mathbf{E}\tilde{\mathcal{U}}_I$ and $\mathbf{A}\tilde{\mathcal{U}}_I$, defined as $\mathbf{E}(\varphi\, \tilde{\mathcal{U}}_I\, \psi) \equiv \neg\mathbf{A}(\neg\varphi\, \mathcal{U}_I\, \neg\psi)$ and $\mathbf{A}(\varphi\, \tilde{\mathcal{U}}_I\, \psi) \equiv \neg\mathbf{E}(\neg\varphi\, \mathcal{U}_I\, \neg\psi)$. A $\mathrm{CTL}_{\mathsf{M}}$ formula φ is in *Flat Normal Form (FNF)* iff it is of the form $p_0 \wedge \bigwedge_i \mathbf{A}(\Box_{[0,\infty)}(p_i \to \psi_i))$ 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 $\mathbf{Q}\bigcirc_I \psi_1$, $\mathbf{Q}(\psi_1\, \mathcal{U}_I\, \psi_2)$ or $\mathbf{Q}(\psi_1\, \tilde{\mathcal{U}}_I\, \psi_2)$ where ψ_1, ψ_2 are formulae of propositional logic and $\mathbf{Q} \in \{\mathbf{E}, \mathbf{A}\}$.

8.1 $\mathrm{CTL}_{\mathsf{M}}$ to CTL Translation: Encoding 'Gaps'

Assume that our $\mathrm{CTL}_{\mathsf{M}}$ formulae are in NNF and FNF. We translate $\mathrm{CTL}_{\mathsf{M}}$ formulae over timed state trees into CTL using gap, a fresh propositional symbol, similar to what we did with our first translation from MTL to LTL. For this, we need to map each timed state tree $\varrho = (T, \sigma, \tau)$ to a state tree ϱ' such that ϱ is a model of a $\mathrm{CTL}_{\mathsf{M}}$ formula if, and only if, ϱ' is a model of our CTL translation. We now define our mappings between $\mathrm{CTL}_{\mathsf{M}}$ and CTL models.

Let $\varrho = (T, \sigma, \tau)$ with $T = (W, <)$ be a timed state tree. With every $w, w' \in W$ and $k \in \mathbb{N}$ we uniquely associate a new node $w''_{(w,w',k)}$. To simplify notation, in the following we denote $w''_{(w,w',k)}$ by $wg^k w'$.

Definition 10 Given a timed state tree $\varrho = (T, \sigma, \tau)$ with $T = (W, <)$, we define a state tree (T', σ') with $T' = (W', <')$, as follows:

$$W' = W \cup \{wg^k w' \mid w, w' \in W, w < w', 1 \leq k < \tau(w') - \tau(w)\}.$$

We call nodes of the form $wg^k w'$ 'gap' nodes.[4] We now define $<'$ as the transitive closure of all of the following:

1. $w <' wgw'$, if $wgw' \in W'$;
2. $wg^k w' <' wg^{k+1} w'$;
3. $wg^k w' <' w'$, if $wg^{k+1} w' \notin W'$;
4. $w <' w'$ if $wgw' \notin W'$ and $w < w'$;

[4] We assume that g is a symbol not among those used to represent $w \in W$.
g^k denotes a sequence of gs of length k, i.e., $g^1 = g$ and $g^{k+1} = gg^k$, for every $k > 0$.

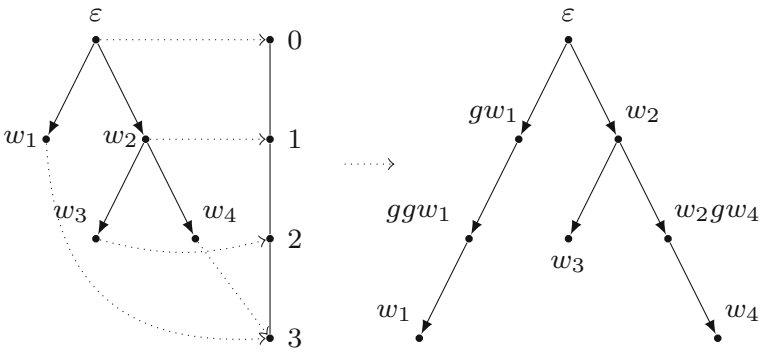

Fig. 7 Example mapping as in Definition 10

Table 15 CTL_{M} to CTL translation using 'gap' where $\mathbf{Q} \in \{\mathbf{E}, \mathbf{A}\}$ and α, β are propositional logic formulae and $c_1, c_2 > 0$

CTL_{M}	CTL Gap Translation
$(\mathbf{Q}\bigcirc_{[0,\infty)}\alpha)^\sharp$	$(\mathbf{Q}\bigcirc_{[1,\infty)}\alpha)^\sharp$
$(\mathbf{Q}\bigcirc_{[c_1,\infty)}\alpha)^\sharp$	$\underbrace{\mathbf{Q}\bigcirc(gap\wedge}_{c_1-1 \text{ times}} \ldots \mathbf{Q}\bigcirc(\mathbf{Q}(gap\,\mathcal{U}\,(\neg gap \wedge \alpha)))\ldots)$
$(\mathbf{Q}\bigcirc_{[c_1,c_2]}\alpha)^\sharp$	$\underbrace{\mathbf{Q}\bigcirc(gap\wedge}_{c_1-1 \text{ times}} \ldots \underbrace{\mathbf{Q}\bigcirc((\neg gap \wedge \alpha) \vee (gap\wedge}_{c_2-c_1 \text{ times}} \ldots \mathbf{Q}\bigcirc(\neg gap \wedge \alpha)\ldots))\ldots)$
$(\mathbf{Q}\bigcirc_{[0,0]}\alpha)^\sharp$	**false**
$(\mathbf{Q}\bigcirc_{[0,c_2]}\alpha)^\sharp$	$(\mathbf{Q}\bigcirc_{[1,c_2]}\alpha)^\sharp$
$(\mathbf{Q}(\alpha\,\mathcal{U}_{[0,\infty)}\,\beta))^\sharp$	$(\neg gap \wedge \beta) \vee (\mathbf{Q}(\alpha\,\mathcal{U}_{[1,\infty)}\,\beta))^\sharp$
$(\mathbf{Q}(\alpha\,\mathcal{U}_{[c_1,\infty)}\,\beta))^\sharp$	$\alpha \wedge \underbrace{\mathbf{Q}\bigcirc((gap \vee \alpha)\wedge}_{c_1-1 \text{ times}} \ldots \mathbf{Q}\bigcirc(\mathbf{Q}(gap \vee \alpha)\,\mathcal{U}\,(\neg gap \wedge \beta))\ldots)$
$(\mathbf{Q}(\alpha\,\mathcal{U}_{[c_1,c_2]}\,\beta))^\sharp$	$\alpha \wedge \underbrace{\mathbf{Q}\bigcirc((gap \vee \alpha)\wedge}_{c_1-1 \text{ times}} \ldots \underbrace{\mathbf{Q}\bigcirc((\neg gap \wedge \beta) \vee ((gap \vee \alpha)\wedge}_{c_2-c_1 \text{ times}} \ldots \mathbf{Q}\bigcirc(\neg gap \wedge \beta)\ldots))\ldots)$
$(\mathbf{Q}(\alpha\,\mathcal{U}_{[0,0]}\,\beta))^\sharp$	$\neg gap \wedge \beta$
$(\mathbf{Q}(\alpha\,\mathcal{U}_{[0,c_2]}\,\beta))^\sharp$	$(\neg gap \wedge \beta) \wedge (\mathbf{Q}(\alpha\,\mathcal{U}_{[1,c_2]}\,\beta))^\sharp$
$(\mathbf{E}(\alpha\,\tilde{\mathcal{U}}_I\,\beta))^\sharp$	$\neg(\mathbf{A}(\neg(\alpha)\,\mathcal{U}_I\,\neg(\beta)))^\sharp$
$(\mathbf{A}(\alpha\,\tilde{\mathcal{U}}_I\,\beta))^\sharp$	$\neg(\mathbf{E}(\neg(\alpha)\,\mathcal{U}_I\,\neg(\beta)))^\sharp$

For a 'gap' node $wg^k w'$, we define $\sigma'(wg^k w') = \{gap\}$ and, for $w \in W$, we define $\sigma'(w) = \sigma(w)$.

Intuitively, in Point (1) we relate a node in W' to a 'gap' node; in Point (2) we relate two consecutive 'gap' nodes; in Point (3) we relate the last of a sequence of 'gap' nodes to a node in W'; finally in Point (4) we relate two non-'gap' nodes in W' if there is no gap between them, i.e, when $\tau(w') - \tau(w) = 1$.

Figure 7 illustrates the mapping given by Definition 10. For instance, if $\varrho = (T, \sigma, \tau)$ is the timed state tree on the left side of Fig. 7 and $\sigma(w_1) = \{p\}$ then $(\varrho, \varepsilon) \models \mathbf{E}\bigcirc_{[2,3]}p$.

Table 15 presents our translation. As shown in Table 15, we translate $\mathbf{E}\bigcirc_{[2,3]}p$ into:

$$\mathbf{E}\bigcirc(gap \wedge \mathbf{E}\bigcirc((\neg gap \wedge p) \vee (gap \wedge \mathbf{E}\bigcirc(\neg gap \wedge p)))).$$

Note that the state tree represented on the right side of Fig. 7 is a model of this CTL formula. Since gap is a propositional symbol not occurring in ϱ, the states from $W \subseteq W'$ do not contain gap.

The next proposition follows straightforwardly from Definition 10.

Proposition 6 *Given a timed state tree $\varrho = (T, \sigma, \tau)$, let ϱ' be as in Definition 10. Then, $(\varrho', \varepsilon) \models \neg gap \wedge \mathbf{A}(\Box(\mathbf{A}\Diamond\neg gap))$.*

Definition 11 Given a state tree $\varrho' = (T', \sigma')$ with $T' = (W', <')$ such that $(\varrho', \varepsilon) \models \neg gap \wedge \mathbf{A}(\Box(\mathbf{A}\Diamond\neg gap))$, we define $\varrho = (T, \sigma, \tau)$, with $T = (W, <)$, as follows. We start by defining W and σ.

$$W = \{w \mid w \in W', gap \notin \sigma'(w)\} \quad \text{and} \quad \sigma(w) = \sigma'(w) \text{ for all } w \in W.$$

It remains to define τ and $<$. For any distinct $w, w' \in W$, we define $<$ as the transitive closure of:

- $w < w'$ iff there is a path π in T' such that $\pi[0] = w$ and there is $k \in \mathbb{N}$ with $\pi[k] = w'$ and $\forall j, 0 < j < k, gap \in \sigma'(\pi[j])$.

To define τ, we first set $\tau(\varepsilon) = 0$. Now, for any distinct $w < w' \in W$, there is a path π in T' such that $\pi[0] = w$ and there is $k \in \mathbb{N}$ with $\pi[k] = w'$ and $\forall j, 0 < j < k, gap \in \sigma'(\pi[j])$. Observe that such k is unique and we set $\tau(w') = \tau(w) + k$.

As ϱ' is such that $(\varrho', \varepsilon) \models \neg gap \wedge \mathbf{A}(\Box(\mathbf{A}\Diamond\neg gap))$, for each $w \in W$, we define $\tau(w) \in \mathbb{N}$. Also, for all π in T, $\tau(\pi[1]) > \tau(\pi[0])$, and so, $\tau : W \to \mathbb{N}$ is well defined. The following proposition states this property.

Proposition 7 *Given a state tree ϱ' such that $(\varrho', \varepsilon) \models \neg gap \wedge \mathbf{A}\Box(\mathbf{A}\Diamond\neg gap)$, let $\varrho = (T, \sigma, \tau)$ be as in Definition 11. Then, $\tau : W \to \mathbb{N}$ is a function such that $\tau(\pi[0]) < \tau(\pi[1])$, for all paths π in T.*

We are ready for Theorem 6, which shows correctness of our translation from $\mathrm{CTL}_{\mathsf{M}}$ to CTL using 'gap's.

Theorem 6 *Let $\varphi = p_0 \wedge \bigwedge_i \mathbf{A}(\Box_{[0,\infty)}(p_i \to \psi_i))$ be an $\mathrm{CTL}_{\mathsf{M}}$ formula in NNF and FNF. Let $\varphi^\sharp = p_0 \wedge \bigwedge_i \mathbf{A}(\Box(p_i \to (\neg gap \wedge \psi_i^\sharp)))$ be the result of replacing each ψ_i in φ by $\psi_i^\sharp$ as in Table 15. Then, φ is satisfiable if, and only if, $\varphi^\sharp \wedge \neg gap \wedge \mathbf{A}(\Box(\mathbf{A}\Diamond\neg gap))$ is satisfiable.*

Proof φ is satisfiable if, and only if, there is a timed state tree $\varrho = (T, \sigma, \tau)$ such that $(\varrho, \varepsilon) \models \varphi$. Given a timed state tree $\varrho = (T, \sigma, \tau)$, let $\varrho' = (T', \sigma')$ be as in Definition 10. By Proposition 6, $(\varrho', \varepsilon) \models \neg gap \wedge \mathbf{A}(\Box(\mathbf{A}\Diamond\neg gap))$. By definition of ϱ', for all $w \in W$ and all propositional variables p_i occurring in $\sigma(w)$, we have $(\varrho, w) \models p_i$ iff $(\varrho', w) \models p_i$. With an inductive argument one can show that, for any propositional formula α, $(\varrho, w) \models \alpha$ iff $(\varrho', w) \models \alpha$.

Also, $\varphi^\sharp \wedge \neg gap \wedge \mathbf{A}(\Box(\mathbf{A}\Diamond\neg gap))$ is satisfiable if, and only if, there is a state tree $\varrho' = (T, \sigma')$ such that $(\varrho', \varepsilon) \models \varphi^\sharp \wedge \neg gap \wedge \mathbf{A}(\Box(\mathbf{A}\Diamond\neg gap))$. Given ϱ', let $\varrho = (T, \sigma, \tau)$ be a timed state tree as in Definition 11. By Proposition 7, $\tau : W \to \mathbb{N}$ is well defined. By definition of ϱ, for all $w \in W'$ with $gap \notin \sigma'(w)$, we have that for p_i a propositional variable occurring in $\sigma'(w)$, $(\varrho, w) \models p_i$ iff $(\varrho', w) \models p_i$. As before, an inductive argument lifts this claim to propositional formulae. That is, for any propositional formula α, $(\varrho, w) \models \alpha$ iff $(\varrho', w) \models \alpha$.

Thus, following our translation in Table 15, we only need to show correspondences between ψ_i and $\neg gap \wedge \psi_i^\sharp$. This follows from Claims 7–9 below (other cases are similar).

In our claims, we use the following bijection between (infinite) paths π in T and π' in T', which is implied by Definitions 10 and 11: $\pi \sim \pi'$ if, and only if, $\pi[0] = \pi'[0]$ and $\forall j \in \mathbb{N}$, $\pi[j] = \pi'[k]$, where $\tau(\pi[j]) - \tau(\pi[0]) = k$. We now argue that, for all natural numbers c_1, c_2 with $0 < c_1 \leq c_2 < C$, the following claims hold.

Claim 7 $(\varrho, w) \models \mathbf{A}\bigcirc_{[c_1,c_2]}\alpha$ *iff* $(\varrho', w) \models \neg gap \wedge (\mathbf{A}\bigcirc_{[c_1,c_2]}\alpha)^\sharp$.

Proof $(\varrho, w) \models \mathbf{A}\bigcirc_{[c_1,c_2]}\alpha$ iff for each π in T with $\pi[0] = w$, we have $\tau(\pi[1]) \in \tau(w) + [c_1, c_2]$ and $(\varrho, \pi[1]) \models \alpha$. By Definitions 10 and 11, this happens iff there is a π' in T' such that $\pi \sim \pi'$ and, for $k = \tau(\pi[1]) - \tau(\pi[0])$, $(\varrho', \pi'[k]) \models \neg gap \wedge \alpha$ and $\forall j \in \mathbb{N}$ with $0 < j < k$, $(\varrho', \pi'[j]) \models gap$. Since π was arbitrary, this holds for all such π. As $\sim$ is a bijection, we obtain this for all such π' in T'. Then, $(\varrho, w) \models \mathbf{A}\bigcirc_{[c_1,c_2]}\alpha$ iff $(\varrho', w) \models \neg gap \wedge (\mathbf{A}\bigcirc_{[c_1,c_2]}\alpha)^\sharp$.

Claim 8 $(\varrho, w) \models \mathbf{A}(\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta)$ *iff* $(\varrho', w) \models \neg gap \wedge (\mathbf{A}(\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta))^\sharp$.

Proof $(\varrho, w) \models \mathbf{A}(\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta)$ iff for each π in T with $\pi[0] = w$, there is $k \in \mathbb{N}$ such that $\tau(\pi[k]) \in \tau(w) + [c_1, c_2]$ and $(\varrho, \pi[k]) \models \beta$; and $\forall j \in \mathbb{N}$, $0 \leq j < k$ implies $(\varrho, \pi[j]) \models \alpha$. By Definitions 10 and 11, this happens iff there is a π' in T' such that $\pi \sim \pi'$ and, for $k' = \tau(\pi[k]) - \tau(\pi[0])$, $(\varrho', \pi'[k']) \models \neg gap \wedge \beta$ and $\forall j \in \mathbb{N}$ with $0 \leq j < k'$, $(\varrho', \pi'[j]) \models gap \vee \alpha$. Since π was arbitrary, this holds for all such π. As $\sim$ is a bijection, we obtain this for all such π' in T'. Thus, $(\varrho, w) \models \mathbf{A}(\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta)$ iff $(\varrho', w) \models \neg gap \wedge (\mathbf{A}(\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta))^\sharp$.

Claim 9 $(\varrho, w) \models \mathbf{E}(\alpha\, \tilde{\mathcal{U}}_{[c_1,c_2]}\, \beta)$ *iff* $(\varrho', w) \models \neg gap \wedge \neg(\mathbf{A}(\neg(\alpha)\, \mathcal{U}_{[c_1,c_2]}\, \neg(\beta)))^\sharp$.

Proof By semantics of $\mathbf{E}\tilde{\mathcal{U}}_I$, we have that $(\varrho, w) \models \mathbf{E}(\alpha\, \tilde{\mathcal{U}}_{[c_1,c_2]}\, \beta)$ iff $(\varrho, w) \models \neg(\mathbf{A}(\neg(\alpha)\, \mathcal{U}_{[c_1,c_2]}\, \neg(\beta)))$. By Claim 8, $(\varrho, w) \not\models \mathbf{A}(\neg(\alpha)\, \mathcal{U}_{[c_1,c_2]}\, \neg(\beta))$ iff $(\varrho', w) \not\models \neg gap \wedge (\mathbf{A}(\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta))^\sharp$. As $(\varrho', w) \models \neg gap$ (by definition of (ϱ', w)), we have that $(\varrho', w) \not\models (\mathbf{A}(\alpha\, \mathcal{U}_{[c_1,c_2]}\, \beta))^\sharp$. Then, $(\varrho, w) \models \mathbf{E}(\alpha\, \tilde{\mathcal{U}}_{[c_1,c_2]}\, \beta)$ iff $(\varrho', w) \models \neg gap \wedge \neg(\mathbf{A}(\neg(\alpha)\, \mathcal{U}_{[c_1,c_2]}\, \neg(\beta)))^\sharp$. □

Due to the binary encoding of constants, our translation is exponential in the size of the input. Then, since satisfiability of CTL can be decided in EXPTIME [21], we obtain 2EXPTIME as upper bound of this problem for $\mathrm{CTL}_{\mathsf{M}}$. Our lower bound follows from the fact that we can translate CRTCTL (which is 2EXPTIME-complete), presented in [20], to $\mathrm{CTL}_{\mathsf{M}}$. In CRTCTL, the models are the same as in CTL (infinite trees with a labelling function that maps nodes to propositional symbols) and the syntax extends CTL with formulae of the form $\mathbf{Q}(\phi \mathcal{U}^{\leq k}\, \psi)$ and $\mathbf{Q}(\phi \mathcal{U}^{=k}\, \psi)$, where $\mathbf{Q} \in \{\mathbf{A}, \mathbf{E}\}$ and $k \in \mathbb{N}$. For example, $\mathbf{E}(\mathbf{true}\mathcal{U}^{\leq 5}\, p)$ means that there is a path in which p occurs within 5 states from now. In fact, we can see CRTCTL models as a special case of $\mathrm{CTL}_{\mathsf{M}}$ models where time differences from one state to another can only be 1. Then, one can easily translate CRTCTL formulae into $\mathrm{CTL}_{\mathsf{M}}$ by replacing temporal operators $\mathcal{U}^{\leq k}$ with $\mathcal{U}_{[0,k]}$ and $\mathcal{U}^{=k}$ with $\mathcal{U}_{[k,k]}$. For the CTL temporal operators we add the interval $[0, \infty)$. Finally we include $\mathbf{A}\Box_{[0,\infty)}(\bigcirc_{[1,1]}\mathbf{true})$ to ensure that the time difference between states is always 1. For example, we translate $\mathbf{E}\bigcirc p \wedge \mathbf{A}(p\, \mathcal{U}^{\leq 3}\, q)$ into $\mathbf{E}\bigcirc_{[0,\infty)} p \wedge \mathbf{A}(p\, \mathcal{U}_{[0,3]}\, q) \wedge \mathbf{A}\Box_{[0,\infty)}(\bigcirc_{[1,1]}\mathbf{true})$.

Theorem 7 *Satisfiability of $\mathrm{CTL}_{\mathsf{M}}$ is 2EXPTIME-complete.*

9 Related Work

As already mentioned in the introduction, propositional MTL has been proposed in the 90's by Alur and Henzinger [6,7] as a formalism for specifying quantitative temporal constraints. More recently, combinations of MTL with Description Logic [9,29] and Datalog [13,14] have also been investigated.

Since its proposal, several authors have used MTL to specify problems from a range of domains (security, robotics, among others). Karaman and Frazzoli use a fragment of MTL to specify some aspects of Vehicle Routing Problems (VRP) [35]. VRPs are a generalisation of the traveling salesman problem with a fleet of vehicles leaving from and returning to one or more departure locations, to delivery locations via a network of communication links e.g. roads or rail tracks, that also must satisfy certain constraints and minimizes transportation costs. In the mentioned work, their focus is on Unmanned Aerial Vehicles (UAVs) with target, launch and landing sites, times spent at the locations, as well as between them and parameters relating to relative risk for using a particular UAV and the time a UAV can be used without re-fueling [35]. This is solved by developing a Mixed-Integer Linear Programming (MILP) algorithm. The style of these problems is very similar to the job-shop scheduling examples we provided in Sect. 3.3, but an important difference is that problem specifications are not fully given in a logical formalism. A similar approach is taken by Zhou et al. [59], they applied a variant of MTL to formalise some constraints in motion planning problems in a dynamic environment for a UAV and for a car. Again, problems are translated into MILP.

Luo et. al define SMTL, an augmented version of a fragment of MTL, suitable for Scheduling Algorithms [43]. This language differs from ours as it has no next operator and the semantics is not pointwise. Fragments of SMTL are considered by restricting aspects of the language so each event can only occur a fixed number of times. They provide examples discussing how these fragments can be used to specify both Job Shop Scheduling and Temporal Networks. To find a schedule, a translation is provided into first order temporal logic and then to Satisfiability Modulo Theory (SMT).

Hoxha et. al describe a tool (Visual Specification Tool, VISPEC) that allows users that are not expert in logics or MTL to specify problems using a graphical formalism [30]. This is automatically translated into MTL. A usability study of the tool is provided and two applications relating to robot surgery and quadcopters are presented. The focus is on specifications rather than satisfiability checking that we consider here.

The robustness of signals modelling or controlling physical processes is considered by Fainekos and Pappas [22]. Robustness relates to how much a trajectory can be perturbed without changing the truth value of a specification expressed in the LTL or MTL. This work is used by Alqahtani et. al to consider safety in relation to path planning applied to an autonomous drone in an adversarial environment where specifications are given in MTL [3]. Robustness is further considered by Abbas et. al in the context of cyber physical systems [2]. They provide a testing framework that returns counterexamples using an MTL specification that minimises a robustness metric.

Thati and Roşu propose a runtime verification and monitoring approach for checking MTL formulae over execution traces of algorithms [53]. An algorithm is presented where both past and future-time MTL formulae are checked against timestamped execution traces. As we do, the authors also restrict the intervals on temporal operators to be natural numbers (or infinity) but also allow past-time operators. Models are timed state sequences allowing real-valued time points. Two subsets of MTL are also considered limiting the specifications to be checked to past-time MTL formulae and LTL. We note that this is a runtime verification approach rather than related to theorem proving.

A runtime verification and monitoring approach using MTL specifications applied to the security domain is presented by Gunadi and Tiu [28]. In particular, the authors consider the privilege escalation attack in the Android operating system. A privilege escalation attack occurs when access services or resources that the attacker doesn't have permissions for is obtained. The past-time fragment of MTL is considered and models are based on the natural numbers and the intervals on temporal operators are natural numbers (where the left hand side is always 0). Similarly to the work by Thati and Roşu [53], the focus is on presenting and implementing an algorithm to check formulae over traces along with its application to the privilege escalation attack.

10 Conclusions

This work considers MTL, a temporal logic that allows timing constraints on the temporal operators, with a pointwise semantics over the natural numbers. We presented four satisfiability preserving translations from MTL to LTL. The translations using time difference are based on the MTL decision procedure presented in [6] 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. We specified examples from robotics, traffic management, and scheduling using MTL. Modelling these examples showed that using the next operator to specify the timing constraints on the next state a robot can move into acts a synchronisation point for all robots in multi-robot systems. Additional care has to be taken using until or eventually operators with intervals where the left hand side is greater than zero (for example $\mathcal{U}_{[1,8]}$) in the scope of an always operator $\Box_{[0,\infty)}$ as this can inadvertently lead to unsatisfiable timing constraints.

For the scheduling example we gave specifications based both on the strict semantics and on the non-strict semantics of MTL. We then used ten different LTL solvers to explore the extent these can solve satisfiability problems based on these specifications. Our experiments seem to indicate that (i) a formalisation based on the non-strict semantics results in poor performance compared to a formalisation based on the strict semantics, (ii) for the vast majority of problems a Gap translation results in better performance than a Time Difference translation, and (iii) LTL solvers that consistently perform well on translated formulae meaning a combination of translation and use of an LTL solver provides a viable means for MTL satisfiability checking and MTL theorem proving.

We have explored the question on what classes of formulae the use of each translation could be advantageous. We have defined two sets of parametrised formulae and again used a range of LTL solvers to verify that for one a Gap translation results in better performance while for the other the Time translation leads to better performance.

We also introduced an branching-time variant of MTL and provided a translation using 'gaps' into the branching-time temporal logic CTL. This opens up the opportunity to use CTL solvers such as [58] in a similar way as we have done for LTL solvers.

As future work, we intend to investigate whether we can translate PDDL3.0 statements [25] into MTL and apply our translations to the planning domain.

References

1. Aalta. http://lab205.org/aalta/
2. Abbas, H., Fainekos, G., Sankaranarayanan, S., Ivančić, F., Gupta, A.: Probabilistic temporal logic falsification of cyber-physical systems. ACM Trans. Embedded Comput. Syst. (TECS) **12**(2s), 95:1–95:30 (2013). https://doi.org/10.1145/2465787.2465797
3. Alqahtani, S., Riley, I., Taylor, S., Gamble, R., Mailler, R.: MTL robustness for path planning with A*. In: André, E., Koenig, S., Dastani, M., Sukthankar, G. (eds.) International Foundation for Autonomous Agents and Multiagent Systems, AAMAS 2018, pp. 247–255 (2018)
4. Alur, R., Courcoubetis, C., Dill, D.L.: Model-checking in dense real-time. Inf. Comput. **104**(1), 2–34 (1993). https://doi.org/10.1006/inco.1993.1024
5. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. J. ACM **43**(1), 116–146 (1996). https://doi.org/10.1145/227595.227602
6. Alur, R., Henzinger, T.A.: Real-time logics: complexity and expressiveness. Inf. Comput. **104**(1), 35–77 (1993). https://doi.org/10.1006/inco.1993.1025
7. Alur, R., Henzinger, T.A.: A really temporal logic. J. ACM **41**(1), 181–204 (1994). https://doi.org/10.1145/174644.174651
8. Alur, R., Henzinger, T.A., Ho, P.: Automatic symbolic verification of embedded systems. IEEE Trans. Softw. Eng. **22**(3), 181–201 (1996). https://doi.org/10.1109/32.489079
9. Baader, F., Borgwardt, S., Koopmann, P., Ozaki, A., Thost, V.: Metric temporal description logics with interval-rigid names. In: Dixon, C., Finger, M. (eds.) FroCoS 2017, LNCS, vol. 10483, pp. 60–76 (2017). https://doi.org/10.1007/978-3-319-66167-4_4
10. Bersani, M.M., Rossi, M., San Pietro, P.: A tool for deciding the satisfiability of continuous-time metric temporal logic. Acta Inform. **53**(2), 171–206 (2016). https://doi.org/10.1109/TIME.2013.20
11. Bertello, M., Gigante, N., Montanari, A., Reynolds, M.: Leviathan: a new LTL satisfiability checking tool based on a one-pass tree-shaped tableau. In: Kambhampati, S. (ed.) IJCAI 2016, pp. 950–956. IJCAI/AAAI Press, Menlo Park (2016)
12. Bouyer, P., Markey, N., Ouaknine, J., Worrell, J.: The cost of punctuality. In: LICS 2007, pp. 109–120. IEEE (2007). https://doi.org/10.1109/LICS.2007.49
13. Brandt, S., Kalaycı, E.G., Kontchakov, R., Ryzhikov, V., Xiao, G., Zakharyaschev, M.: Ontology-based data access with a Horn fragment of metric temporal logic. In: Singh, S., Markovitch, S. (eds.) AAAI 2017, pp. 1070–1076. AAAI Press, Menlo Park (2017)
14. Brandt, S., Kalaycı, E.G., Ryzhikov, V., Xiao, G., Zakharyaschev, M.: Querying log data with metric temporal logic. J. Artif. Intell. Res. **62**, 829–877 (2018). https://doi.org/10.1613/jair.1.11229
15. 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: Brinksma, E., Larsen, K.G. (eds.) CAV 2002, LNCS, vol. 2404, pp. 359–364. Springer, Berlin (2002). https://doi.org/10.1007/3-540-45657-0_29
16. Dauzère-Pérès, S., Paulli, J.: An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Ann. Oper. Res. **70**, 281–306 (1997). https://doi.org/10.1023/A:1018930406487
17. Dixon, C., Fisher, M., Konev, B.: Temporal logic with capacity constraints. In: Konev, B., Wolter, F. (eds.) FroCoS 2007, LNCS, vol. 4720, pp. 163–177. Springer, Berlin (2007). https://doi.org/10.1007/978-3-540-74621-8_11
18. Dixon, C., Webster, M., Saunders, J., Fisher, M., Dautenhahn, K.: "The Fridge Door is Open"–temporal verification of a robotic assistant's behaviours. In: Mistry, M., Leonardis, A., Witkowski, M., Melhuish, C. (eds.) TAROS 2014, LNCS, vol. 8717, pp. 97–108. Springer, Berlin (2014). https://doi.org/10.1007/978-3-319-10401-0_9
19. Duque, I., Dautenhahn, K., Koay, K.L., Willcock, L., Christianson, B.: Knowledge-driven user activity recognition for a smart house - development and validation of a generic and low-cost, resource-efficient system. In: Miller, L. (ed.) ACHI 2013, pp. 141–146. IARIA XPS Press, Wilmington (2013)
20. Emerson, E., Mok, A., Sistla, A., Srinivasan, J.: Quantitative temporal reasoning. Real-Time Syst. **4**(4), 331–352 (1992). https://doi.org/10.1007/BF00355298

21. Emerson, E.A., Halpern, J.Y.: Decision procedures and expressiveness in the temporal logic of branching time. In: Lewis, H.R., Simons, B.B., Burkhard, W.A., Landweber, L.H. (eds.) STOC 1982, pp. 169–180. ACM, New York (1982)
22. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comput. Sci. **410**(42), 4262–4291 (2009). https://doi.org/10.1016/j.tcs.2009.06.021
23. Fisher, M.: A normal form for temporal logics and its applications in theorem-proving and execution. J. Logic Comput. **7**(4), 429–456 (1997)
24. Gabbay, D., Pnueli, A., Shelah, S., Stavi, J.: On the temporal analysis of fairness. In: Abrahams, P.W., Lipton, R.J., Bourne, S.R. (eds.) POPL 1980, pp. 163–173. ACM, New York (1980). https://doi.org/10.1145/567446.567462
25. 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. Artif. Intell. **173**(5–6), 619–668 (2009). https://doi.org/10.1016/j.artint.2008.10.012
26. Goré, R.: And-or tableaux for fixpoint logics with converse: LTL, CTL, PDL and CPDL. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014, LNCS, vol. 8562, pp. 26–45. Springer, Berlin (2014). https://doi.org/10.1007/978-3-319-08587-6_3
27. Graham, R.L.: Bounds for certain multiprocessing anomalies. Bell Labs Tech. J. **45**(9), 1563–1581 (1966)
28. Gunadi, H., Tiu, A.: Efficient runtime monitoring with metric temporal logic: a case study in the Android operating system. In: Jones, C., Pihlajasaari, P., Sun, J. (eds.) FM 2014, LNCS, vol. 8442, pp. 296–311. Springer, Berlin (2014). https://doi.org/10.1007/978-3-319-06410-9_21
29. Gutiérrez-Basulto, V., Jung, J.C., Ozaki, A.: On metric temporal description logics. In: Kaminka, G.A., Fox, M. (eds.) ECAI 2016, Frontiers in Artificial Intelligence and Applications, vol. 285, pp. 837–845. IOS Press, Amsterdam (2016). https://doi.org/10.3233/978-1-61499-672-9-837
30. Hoxha, B., Mavridis, N., Fainekos, G.: VISPEC: A graphical tool for elicitation of MTL requirements. In: IROS 2015, vol. 2015-December, pp. 3486–3492. IEEE, New York (2015). https://doi.org/10.1109/IROS.2015.7353863
31. Hustadt, U., Dixon, C., Ozaki, A.: Mtl: Tools and experiments. University of Liverpool, Liverpool http://cgi.csc.liv.ac.uk/~ullrich/MTL (2019)
32. Hustadt, U., Konev, B.: TRP^{++} 2.0: a temporal resolution prover. In: Baader, F. (ed.) Proceedings of the CADE-19, LNCS, vol. 2741, pp. 274–278. Springer, Berlin (2003). https://doi.org/10.1007/978-3-540-45085-6_21
33. Hustadt, U., Ozaki, A., Dixon, C.: Theorem proving for metric temporal logic over the naturals. In: de Moura, L. (ed.) CADE-26, LNCS, vol. 10395, pp. 326–343. Springer, Berlin (2017). https://doi.org/10.1007/978-3-319-63046-5_20
34. Janssen, G.: Logics for digital circuit verification: Theory, algorithms, and applications. Ph.D. thesis, Eindhoven University of Technology, Eindhoven (1999)
35. Karaman, S., Frazzoli, E.: Vehicle routing problem with metric temporal logic specifications. In: CDC 2008, pp. 3953–3958. IEEE (2008). https://doi.org/10.1109/CDC.2008.4739366
36. Leviathan. https://github.com/Corralx/leviathan
37. Li, J., Yao, Y., Pu, G., Zhang, L., He, J.: Aalta: an LTL satisfiability checker over infinite/finite traces. In: Cheung, S.C., Orso, A., Storey, M.A.D. (eds.) SIGSOFT FSE 2014, pp. 731–734. ACM, New York (2014). https://doi.org/10.1145/2635868.2661669
38. Liu, W., Winfield, A.: Modelling and optimisation of adaptive foraging in swarm robotic systems. Int. J. Robot. Res. **29**(14), 1743–1760 (2010)
39. Logics Workbench. http://www.lwb.unibe.ch/index.html
40. LS4. https://github.com/quickbeam123/ls4
41. Ludwig, M., Hustadt, U.: Resolution-based model construction for PLTL. In: Lutz, C., Raskin, J. (eds.) TIME 2009, pp. 73–80. IEEE, New York (2009). https://doi.org/10.1109/TIME.2009.11
42. Ludwig, M., Hustadt, U.: Implementing a fair monodic temporal logic prover. AI Commun. **23**(2–3), 69–96 (2010)
43. Luo, R., Valenzano, R.A., Li, Y., Beck, J.C., McIlraith, S.A.: Using metric temporal logic to specify scheduling problems. In: Baral, C., Delgrande, J.P., Wolter, F. (eds.) KR 2016, pp. 581–584. AAAI Press, Menlo Park (2016)
44. Mohammed, A., Furbach, U.: MAS: qualitative and quantitative reasoning. In: Dennis, L., Boissier, O., Bordini, R.H. (eds.) ProMAS 2011, LNCS, vol. 7217, pp. 114–132. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-31915-0_7
45. NuSMV. http://nusmv.fbk.eu/
46. Ouaknine, J., Worrell, J.: Some recent results in metric temporal logic. In: Cassez, F., Jard, C. (eds.) FORMATS 2008, LNCS, vol. 5215, pp. 1–13. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-85778-51

47. pltl. http://users.cecs.anu.edu.au/~rpg/PLTLProvers/
48. Pnueli, A.: The temporal logic of programs. In: Proceedings of the SFCS '77, pp. 46–57. IEEE (1977). https://doi.org/10.1109/SFCS.1977.32
49. Reynolds, M.: A new rule for LTL tableaux. In: D. Cantone, G. Delzanno (eds.) GandALF 2016, EPTCS, vol. 226, pp. 287–301 (2016). https://doi.org/10.4204/EPTCS.226.20
50. Schwendimann, S.: A new one-pass tableau calculus for PLTL. In: de Swart, H.C.M. (ed.) TABLEAUX 1998, LNCS, vol. 1397, pp. 277–292. Springer, Berlin (1998)
51. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. J. ACM **32**(3), 733–749 (1985). https://doi.org/10.1145/3828.3837
52. Suda, M., Weidenbach, C.: A PLTL-prover based on labelled superposition with partial model guidance. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012, LNCS, vol. 7364, pp. 537–543. Springer, Berlin (2012). https://doi.org/10.1007/978-3-642-31365-3_42
53. Thati, P., Roşu, G.: Monitoring algorithms for metric temporal logic specifications. Electron. Notes Theor. Comput. Sci. **113**, 145–162 (2005)
54. TRP^{++}. http://cgi.csc.liv.ac.uk/~konev/software/trp++/
55. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In: Siekmann, J.H., Wrightson, G. (eds.) Automation of Reasoning, vol 2: Classical Papers on Computational Logic 1967–1970, pp. 466–483. Springer, Berlin (1983)
56. TSPASS. http://cgi.csc.liv.ac.uk/~michael/TLBook/TSPASS-System/
57. Webster, M., Dixon, C., Fisher, M., Salem, M., Saunders, J., Koay, K.L., Dautenhahn, K., Saez-Pons, J.: Toward reliable autonomous robotic assistants through formal verification: a case study. IEEE Trans. Hum. Mach. Syst. **46**(2), 186–196 (2016). https://doi.org/10.1109/THMS.2015.2425139
58. Zhang, L., Hustadt, U., Dixon, C.: A resolution calculus for the branching-time temporal logic CTL. ACM Trans. Comput. Log **15**(1), 10:1–10:38 (2014). https://doi.org/10.1145/2529993
59. Zhou, Y., Maity, D., Baras, J.S.: Optimal mission planner with timed temporal logic constraints. In: ECC 2015, pp. 759–764. IEEE, New York (2015). https://doi.org/10.1109/ECC.2015.7330634