Active Learning of Mealy Machines with Timers[*]

Véronique Bruyère[1] , Bharat Garhewal[2] , Guillermo A. Pérez[3] , Gaëtan Staquet[1,3] , and Frits W. Vaandrager[2]

[1] University of Mons, Belgium
{veronique.bruyere,gaetan.staquet}@umons.ac.be
[2] Radboud University, The Netherlands
{b.garhewal,f.vaandrager}@cs.ru.nl
[3] University of Antwerp – Flanders Make, Belgium
guillermo.perez@uantwerpen.be

Abstract. We present the first algorithm for query learning of a class of Mealy machines with timers in a black-box context. Our algorithm is an extension of the $L^{\#}$ algorithm of Vaandrager et al. [32] to a timed setting. We rely on symbolic queries which empower us to reason on untimed executions while learning. Similarly to the algorithm for learning timed automata of Waga [33], these symbolic queries can be implemented using finitely many concrete queries. Experiments with a prototype implementation, written in Rust, show that our algorithm is able to efficiently learn realistic benchmarks.

Keywords: Timed systems, model learning, active automata learning

1 Introduction

To understand and verify complex systems, we need accurate models that are understandable for humans or can be analyzed fully automatically. Such models, are typically not available for legacy software and for AI systems constructed from training data. Model learning is a technology that potentially may fill this gap. In this work, we consider a specific kind of model learning: active automata learning, which is a black-box technique for constructing state machine models of software and hardware from information obtained through testing (i.e., providing inputs and observing the resulting outputs). It has been successfully used in many applications, e.g., for spotting bugs in implementations of major network protocols [12–16, 27]. We refer to [23, 30] for surveys and further references.

Timing plays a crucial role in many applications. However, extending model learning algorithms to a setting that incorporates quantitative timing information turns out to be challenging. Twenty years ago, the first papers on this subject were published [19, 26], but we still do not have scalable algorithms for

---

a general class of timed models. Consequently, in applications of model learning technology, timing issues still need to be artificially suppressed.

Several authors have proposed active learning algorithms for the popular framework of timed automata (TA) [3], which extends DFAs with clock variables. Some of these proposals, for instance [20–22] have never been implemented. In recent years, however, several algorithms have been proposed and implemented that successfully learned realistic benchmark models. A first line of work restricts to subclasses of TAs such as deterministic one-clock timed automata (DOTAs) [5, 34]. A second line of work explores synergies between active and passive learning algorithms. Aichernig et al. [1, 29], for instance, employ a passive learning algorithm based on genetic programming to generate hypothesis models, which are subsequently refined using equivalence queries. A major result was obtained recently by Waga [33], who presents an algorithm for active learning of (general) deterministic TAs and shows the effectiveness of the algorithm on various benchmarks. Waga's algorithm is inspired by ideas of Maler & Pnueli [26]. In particular, based on the notion of elementary languages of [26], Waga uses *symbolic queries* which are then implemented using finitely many *concrete queries*. A challenge for learning algorithms for timed automata is the inference of the guards and resets that label transitions. As a result, the algorithm of Waga [33] requires an exponential number of concrete membership queries to implement a single symbolic query. Notably, symbolic queries are also used for learning other families of automata, such as register automata [18].

Given the difficulties of inferring the guards and resets of TAs, Vaandrager et al. [31] propose to consider learning algorithms for models using timers instead of clocks, e.g., the class of models defined by Dill [11]. The value of a timer decreases when time advances, whereas the value of a clock increases. Vaandrager et al. [31] present a slight generalization of the notion of a Mealy machine by adding a single timer (MM1T). In an MM1T, the timer can be set to integer values on transitions and may be stopped or time out in later transitions. The timer expires when its value becomes 0 and at this point a timeout event occurs. Each timeout triggers an observable output, allowing a learner to observe the occurrence of timeouts. The absence of guards and invariants in MM1Ts simplifies learning. A learner still has to determine which transitions (re)start timers, but this no longer creates a combinatorial blow-up. If a transition sets a timer, then slight changes in the timing of this transition will trigger corresponding changes in the timing of the resulting timeout, allowing a learner to identify the exact cause of each timeout. Even though many realistic systems can be modeled as MM1Ts (e.g., the benchmarks in [31] and the brick sorter and traffic controller examples in [10]), the restriction to a single timer is a serious limitation. Therefore, Kogel et al. [25] propose *Mealy machines with local timers (MMLTs)*, an extension of MM1T with multiple timers subject to carefully chosen constraints to enable efficient learning. Although quite interesting, the constraints of MMLTs are too restrictive for many applications (e.g., the FDDI protocol described in Appendix F). Also, any MMLT can be converted to an equivalent MM1T. We explored in [7] a general extension of MM1Ts with *multiple timers (MMTs)*, and

show that for MMTs that are *race-avoiding*, the cause of a timeout event can be efficiently determined by "wiggling" the timing of input events. Finally, MMTs form a subclass of TAs, and, thus, existing model checking algorithms and tools for TAs (such as UPPAAL[4]) can be used.

The main result of this paper is a learning algorithm for the MMT models of [7], that is obtained by extending the $L^{\#}$ learning algorithm for Mealy machines of Vaandrager et al. [32] and using ideas of Maler & Pnueli [26]. Experiments with a prototype implementation, written in Rust, show that our algorithm is able to efficiently learn realistic benchmarks.

2 Mealy Machines with Timers

A *Mealy machine* is a variant of the classical finite automaton that associates an output with each transition instead of associating a boolean with each state. A Mealy machine can be seen as a relation between input words and output words. Mealy machines with timers [31] can then be used to enforce timing constraints over the behavior of the relation, e.g., if we send a message and do not receive the acknowledgment after d units of time, we resend the message.

We fix a non-empty, finite set I of *inputs* and a non-empty set O of *outputs*. A *Mealy machine with timers* uses a finite set X of *timers*, from which we define the set $TO[X] = \{to[x] \mid x \in X\}$ of *timeouts of X*, and write $A(\mathcal{M})$ for the set $I \cup TO[X]$ of *actions of $\mathcal{M}$*: reading an input (an *input action*), or processing a timeout (a *timeout action*). Finally, $U(\mathcal{M}) = (X \times \mathbb{N}^{>0}) \cup \{\bot\}$ is the set of *updates of $\mathcal{M}$*, where (x, c) means that timer x is started with value c, and $\bot$ stands for no timer update. We impose certain constraints on the shape of MMTs, e.g., a timer x can time out only when it is active, allowing us to define hereafter the timed semantics in a straightforward manner.

Definition 1 (Mealy machine with timers). *A Mealy machine with timers (MMT, for short) is a tuple $\mathcal{M} = (X, Q, q_0, \chi, \delta)$ where:*

- *X is a finite set of timers (we assume $X \cap \mathbb{N}^{>0} = \emptyset$),*
- *Q is a finite set of states, with $q_0 \in Q$ the initial state,*
- *$\chi : Q \to \mathcal{P}(X)$ assigns a set of active timers to each state, and*
- *$\delta : Q \times A(\mathcal{M}) \rightharpoonup Q \times O \times U(\mathcal{M})$ is a partial transition function.*

We write $q \xrightarrow{i/o}_{u} q'$ if $\delta(q, i) = (q', o, u)$. We require the following:

1. *In the initial state, no timer is active, i.e., $\chi(q_0) = \emptyset$.*
2. *All active timers of the target state of a transition come from the source state, except at most one timer that may be started on the transition. That is, if $q \xrightarrow{i/o}_{\bot} q'$, $\chi(q') \subseteq \chi(q)$, and, if $q \xrightarrow{i/o}_{(x,c)} q'$ with $c \in \mathbb{N}^{>0}$, $\chi(q') \setminus \{x\} \subseteq \chi(q)$.*
3. *If timer x times out then x was active in the source state, i.e., if $q \xrightarrow{to[x]/o}_{u} q'$ then $x \in \chi(q)$. Moreover, if $u \neq \bot$, then u must be (x, c) for some $c \in \mathbb{N}^{>0}$.*
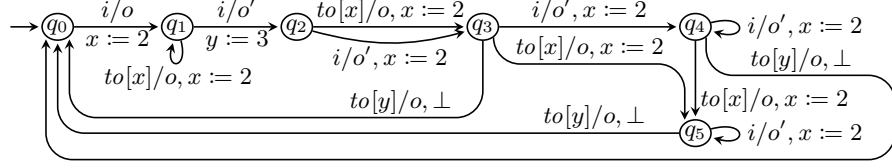
---
[4] https://uppaal.org/

Fig. 1: An MMT with $\chi(q_0) = \emptyset$, $\chi(q_1) = \{x\}$, and $\chi(q) = \{x, y\}$ for all other q.

When needed, we add a superscript to indicate which MMT is considered, e.g., $Q^{\mathcal{M}}, q_0^{\mathcal{M}}$, etc. Missing symbols in $q \xrightarrow{i/o}_{u} q'$ are quantified existentially. We say a transition $q \xrightarrow{}_{u} q'$ *starts* (resp. *restarts*) the timer x if $u = (x, c)$ and x is inactive (resp. active) in q. We say that a transition $q \xrightarrow{i} q'$ with $i \neq to[x]$ *stops* the timer x if x is inactive in q'. The requirement that a $to[x]$-transition can only restart the timer x is without loss of generality and a choice made to simplify the presentation (see Appendix E.2). An example of an MMT is given in Figure 1.

A *run* π of $\mathcal{M}$ either consists of a single state p_0 or of a nonempty sequence of transitions $\pi = p_0 \xrightarrow{i_1/o_1}_{u_1} p_1 \xrightarrow{i_2/o_2}_{u_2} \cdots \xrightarrow{i_n/o_n}_{u_n} p_n$. We denote by $runs(\mathcal{M})$ the set of runs of $\mathcal{M}$. We often write $q \xrightarrow{i} \; \in runs(\mathcal{M})$ to highlight that $\delta(q, i)$ is defined. We lift the notation to words $i_1 \cdots i_n$ as usual: $p_0 \xrightarrow{i_1 \cdots i_n} p_n \in runs(\mathcal{M})$ if there exists a run $p_0 \xrightarrow{i_1} \cdots \xrightarrow{i_n} p_n \in runs(\mathcal{M})$. Note that any run π is uniquely determined by its first state p_0 and word, as $\mathcal{M}$ is deterministic. A run $p_0 \xrightarrow{i_1 \cdots i_n}$ is *x-spanning* (with $x \in X$) if it begins with a transition (re)starting x, ends with a $to[x]$-transition, and no intermediate transition restarts or stops x.

2.1 Timed Semantics

The semantics of an MMT $\mathcal{M}$ is defined via an infinite-state labeled transition system describing all possible configurations and transitions between them. A *valuation* is a partial function $\kappa \colon X \rightharpoonup \mathbb{R}^{\geq 0}$ that assigns nonnegative real numbers to timers. For $Y \subseteq X$, we write $\mathsf{Val}(Y)$ for the set of all valuations κ with $\mathsf{dom}(\kappa) = Y$. A *configuration* of $\mathcal{M}$ is a pair (q, κ) where $q \in Q$ and $\kappa \in \mathsf{Val}(\chi(q))$. The *initial configuration* is the pair (q_0, κ_0) where $\kappa_0 = \emptyset$ since $\chi(q_0) = \emptyset$. If $\kappa \in \mathsf{Val}(Y)$ is a valuation in which all timers from Y have a value of at least $d \in \mathbb{R}^{\geq 0}$, then d units of time may *elapse*. We write $\kappa - d \in \mathsf{Val}(Y)$ for the resulting valuation that satisfies $(\kappa - d)(x) = \kappa(x) - d$, for all $x \in Y$. If the valuation κ contains a value $\kappa(x) = 0$ for some timer x, then x may *time out*. We define the transitions between configurations $(q, \kappa), (q', \kappa')$ as follows:

delay transition $(q, \kappa) \xrightarrow{d} (q, \kappa - d)$, with $\kappa(x) \geq d$ for every $x \in \chi(q)$,

discrete transition $(q, \kappa) \xrightarrow{i/o}_{(x,c)} (q', \kappa')$, with $q \xrightarrow{i/o}_{(x,c)} q'$ a transition, $\kappa'(x) = c$ and $\kappa'(y) = \kappa(y)$ for all $y \in \chi(q')$ such that $y \neq x$. Moreover, if $i = to[x]$, $\kappa(x) = 0$ and it is a *timeout transition*; otherwise, it is an *input transition*.

A *timed run* of $\mathcal{M}$ is a sequence of configuration transitions such that delay and discrete transitions alternate, beginning and ending with a delay transition. We say a configuration (q, κ) is *reachable* if there is a timed run ρ that starts with the initial configuration and ends with (q, κ). The *untimed projection* of ρ, noted $untime(\rho)$, is the run obtained by omitting the valuations and delay transitions of ρ. A run π is said *feasible* if there is a timed run ρ such that $untime(\rho) = \pi$.

A *timed word* over a set Σ is an alternating sequence of delays from $\mathbb{R}^{\geq 0}$ and symbols from Σ, such that it starts and ends with a delay. The length of a timed word w, noted $|w|$, is the number of symbols of Σ in w. Note that, when $\Sigma = A(\mathcal{M})$, a timed run reading a timed word w is uniquely determined by its first configuration and w. We thus write $(p, \kappa) \xrightarrow{w}$ for a timed run. A timed run ρ is called *x-spanning* (with $x \in X$) if $untime(\rho)$ is x-spanning.

Example 1. Let $\mathcal{M}$ be the MMT of Figure 1. The timed run reading the timed word $0.5 \cdot i \cdot 1 \cdot i \cdot 1 \cdot to[x] \cdot 2 \cdot to[y] \cdot 0$ and its untimed projection are:

$$(q_0, \emptyset) \xrightarrow{0.5} (q_0, \emptyset) \xrightarrow[(x,2)]{i/o} (q_1, x = 2) \xrightarrow{1} (q_1, x = 1) \xrightarrow[(y,3)]{i/o'} (q_2, x = 1, y = 3)$$

$$\xrightarrow{1} (q_2, x = 0, y = 2) \xrightarrow[(x,2)]{to[x]/o} (q_3, x = 2, y = 2) \xrightarrow{2} (q_3, x = 0, y = 0)$$

$$\xrightarrow[\perp]{to[y]/o} (q_0, \emptyset) \xrightarrow{0} (q_0, \emptyset)$$

$$\pi = q_0 \xrightarrow[(x,2)]{i/o} q_1 \xrightarrow[(y,3)]{i/o'} q_2 \xrightarrow[(x,2)]{to[x]/o} q_3 \xrightarrow[\perp]{to[y]/o} q_0.$$

Hence, π is feasible, unlike the run $q_0 \xrightarrow{i \cdot i \cdot to[y]}$, as the value of y in q_2 is always greater than the value of x, no matter the chosen delays. Observe that $q_1 \xrightarrow{i \cdot to[x] \cdot to[y]}$ is y-spanning, while $q_0 \xrightarrow{i \cdot i \cdot to[x]}$ is x-spanning.

As $(q_2, x = 0, y = 2)$ is reachable, we say that x is *enabled* in q_2, i.e., it is possible to observe the timeout of x in q_2 along some timed run. However, y is not enabled in q_2, as it is impossible to reach a configuration $(q_2, x = \cdot, y = 0)$. We write $\chi_0(q)$ for the set of all enabled timers of q, i.e., $\chi_0(q) = \{x \in \chi(q) \mid \exists (q_0, \emptyset) \xrightarrow{w} (q, \kappa) : \kappa(x) = 0\}$. If q has at least one enabled timer then, just by waiting in q for long enough, we can force one timer to reach the value zero. A desirable property is for all such behaviors to have a corresponding timeout transition, which is the case for $\mathcal{M}$. We thus say that an MMT $\mathcal{N}$ is *complete* if for all $q \in Q$ and all $i \in I \cup TO[\chi_0(q)]$ we have $q \xrightarrow{i} \in runs(\mathcal{N})$.

2.2 Symbolic equivalence of MMTs

Later in this work, we need to decide whether two complete MMTs, that may use different timers, describe the same timed behaviors, up to timer renaming. To do so, we introduce a way to *symbolically* describe runs to abstract time and timer names. We define *symbolic words* $\mathtt{w}$ over the alphabet $\mathtt{A} = I \cup TO[\mathbb{N}^{>0}]$ to describe the x-spanning sub-runs of a run in the following way. Along a run π of

a complete MMT, for any $to[x]$-transition there must exist an earlier transition (re)starting x. The part of the run between the last such transition and the $to[x]$ is x-spanning. Hence, for a given $to[x]$-transition of π, there is a unique transition that is the source of this timeout transition. Let $w = i_1 \cdots i_n$ be a word over $A(\mathcal{M})$ that is the label of a run $\pi = p_0 \xrightarrow[u_1]{i_1} p_1 \xrightarrow[u_2]{i_2} \cdots \xrightarrow{i_n} p_n \in runs(\mathcal{M})$. The *symbolic word* (sw, in short) *of* w is the word $\overline{w} = \mathtt{i_1} \cdots \mathtt{i_n}$ over $\mathtt{A}$ such that, for every $k \in \{1, \ldots, n\}$, $\mathtt{i_k} = i_k$ if $i_k \in I$ and $\mathtt{i_k} = to[j]$ where $j < k$ is the index of the last transition (re)starting x if $i_k = to[x]$. Conversely, given a symbolic word $\mathtt{w} = \mathtt{i_1} \ldots \mathtt{i_n}$ over $\mathtt{A}$, one can convert it into a run $q_0 \xrightarrow{w}$ using concrete timeout symbols such that $\overline{w} = \mathtt{w}$ if such a run exists in $\mathcal{M}$. In an abuse of notation, we write $q_0 \xrightarrow{\mathtt{w}}$ to denote the run $q_0 \xrightarrow{w}$ such that $\overline{w} = \mathtt{w}$.

Example 2. Let $\mathcal{M}$ be the MMT of Figure 1 and $\pi = q_0 \xrightarrow[(x,2)]{i} q_1 \xrightarrow[(x,2)]{to[x]} q_1 \xrightarrow[(x,2)]{to[x]} q_1$ be a run. Let us construct the symbolic word $\mathtt{w} = \mathtt{i_1} \cdot \mathtt{i_2} \cdot \mathtt{i_3}$ such that $\overline{i \cdot to[x] \cdot to[x]} = \mathtt{w}$. As the first action of π is the input i, we get $\mathtt{i_1} = i$. The second action of π is $to[x]$ and the last transition to (re)start x is the first transition of π. So, $\mathtt{i_2} = to[1]$. Likewise, the last symbol $\mathtt{i_3}$ of $\mathtt{w}$ must be $to[2]$, as the second transition of π restarts x. Hence, $\mathtt{w} = i \cdot to[1] \cdot to[2]$. In the opposite direction, it is not hard to see that the symbolic word $\mathtt{w} = i \cdot i \cdot to[1] \cdot to[2]$ induces the run $q_0 \xrightarrow[(x,2)]{i} q_1 \xrightarrow[(y,3)]{i} q_2 \xrightarrow{to[x]} q_2 \xrightarrow{to[y]} q_0$ in $\mathcal{M}$ such that $\overline{i \cdot i \cdot to[x] \cdot to[y]} = \mathtt{w}$.

Definition 2 (Symbolic equivalence). *Two complete MMTs $\mathcal{M}$ and $\mathcal{N}$ are symbolically equivalent, noted $\mathcal{M} \overset{sym}{\approx} \mathcal{N}$, if for any sw $\mathtt{w} = \mathtt{i_1} \cdots \mathtt{i_n}$ over $\mathtt{A}$:*

- $q_0^{\mathcal{M}} = q_0 \xrightarrow[u_1]{\mathtt{i_1}/o_1} \cdots \xrightarrow[u_n]{\mathtt{i_n}/o_n} q_n$ *is feasible in $\mathcal{M}$ iff $q_0^{\mathcal{N}} = q_0' \xrightarrow[u_1']{\mathtt{i_1}/o_1'} \cdots \xrightarrow[u_n']{\mathtt{i_n}/o_n'} q_n'$ is feasible in $\mathcal{N}$.*
- *Moreover, for all $j \in \{1, \ldots, n\}$, $o_j = o_j'$ and, if $q_{j-1} \xrightarrow{\mathtt{i_j} \cdots \mathtt{i_k}} q_k$ is spanning, then $u_j = (x, c)$ and $u_j' = (x', c')$ with $c = c'$.*

Notice that $q_{j-1} \xrightarrow{\mathtt{i_j} \cdots \mathtt{i_k}} q_k$ is spanning in $\mathcal{M}$ if and only if $q_{j-1}' \xrightarrow{\mathtt{i_j} \cdots \mathtt{i_k}} q_k'$ is spanning in $\mathcal{N}$ as both machines read the same symbolic word. Notice also that no condition is imposed on the updates u_j, u_j' appearing outside the start of spanning runs. As outputs and updates at the start of spanning runs are the same, symbolic equivalence implies the classical *timed equivalence* (see Appendix A).

2.3 Learning framework

As usual for learning algorithms, we rely on Angluin's framework [6]: we assume we have a teacher who knows $\mathcal{M}$, and a learner who does not know $\mathcal{M}$ but can query the teacher to obtain knowledge about $\mathcal{M}$. Let us first characterize the set of MMTs that we consider for our learning algorithm. We say that an MMT $\mathcal{M}$ is *s-learnable* (the s stands for symbolically) if it is complete and every run of $\mathcal{M}$ is feasible. The MMT of Figure 1 is s-learnable. From any complete MMT

$\mathcal{M}$, one can construct an s-learnable MMT $\mathcal{N}$ that is symbolically equivalent, by using *zones* (akin to the homonymous concept for timed automata, see [8] for an introduction) to represent sets of valuations. See Appendix B for details.

Lemma 1. *For any complete MMT $\mathcal{M}$, there is an s-learnable MMT $\mathcal{N} \overset{sym}{\approx} \mathcal{M}$.*

We now define the *queries* the learner uses to gather knowledge about $\mathcal{M}$, the MMT of the teacher. For classical Mealy machines [28, 32], there are two queries: *output queries* providing the sequence of outputs for a given input word, and *equivalence queries* asking whether a hypothesis $\mathcal{H}$ is correct. If it is not, a counterexample is returned, i.e., a word w inducing different outputs in $\mathcal{H}$ and in $\mathcal{M}$. In this work, we need to adapt those queries to encode the timed behavior induced by the timers of $\mathcal{M}$. As two MMTs do not use the same timers in general, we rely on sws, and adapt our queries for the same. In order to deal with timed behavior, we also need a new type of query, called a *wait query*. In the definition, it is required that the word induces a run in $\mathcal{M}$, which can be ensured by stepwise increasing the length of the considered words (see Example 3).

Definition 3 (Symbolic queries).   *The learner uses three symbolic queries:*

OQ^s(w) *with w a sw such that $q_0^{\mathcal{M}} \overset{w}{\to} \in runs(\mathcal{M})$, returns the outputs of $q_0^{\mathcal{M}} \overset{w}{\to}$.*

WQ^s(w) *with w a sw such that $q_0^{\mathcal{M}} \overset{i_1}{\to} \cdots \overset{i_n}{\to} q_n \in runs(\mathcal{M})$ with $\overline{i_1 \cdots i_n} = w$, returns all pairs (j, c) such that $q_{j-1} \xrightarrow[(x,c)]{i_j} q_j \xrightarrow{i_{j+1}\cdots i_n \cdot to[x]}$ is x-spanning.*

EQ^s($\mathcal{H}$) *with $\mathcal{H}$ a complete MMT, returns **yes** if $\mathcal{H} \overset{sym}{\approx} \mathcal{M}$, or a sw witnessing the non-equivalence.*

OQ^s and **EQ^s** are analogous to regular output and equivalence queries for Mealy machines, while **WQ^s** provides, for each timer x enabled at the end of the run induced by the symbolic word, the transition which last (re)started x and the constant c with which x was (re)started. We claim these symbolic queries can be performed via concrete output and equivalence queries, i.e., queries using tiws instead of sws, under the assumption that $\mathcal{M}$ is *race-avoiding* [7]. In short, a race-avoiding MMT allows runs to be observed deterministically, in the sense that any feasible run is the untimed projection of a run ρ where all delays are non-zero and there are no two timers that time out at the same time along ρ. Not all MMTs are race-avoiding and a 3EXP algorithm to decide whether one is race-avoiding is given in [7]. See Appendix C for a proof.

Lemma 2. *For race-avoiding MMTs, the three symbolic queries can be implemented via a polynomial number of concrete output and equivalence queries.*

3   Learning algorithm

We now describe our learning algorithm for MMTs, called $L_{\mathrm{MMT}}^{\#}$. Let $\mathcal{M}$ be the hidden s-learnable MMT we want to learn. We first define the data structure of the learner called an *observation tree*, before explaining how to use the queries
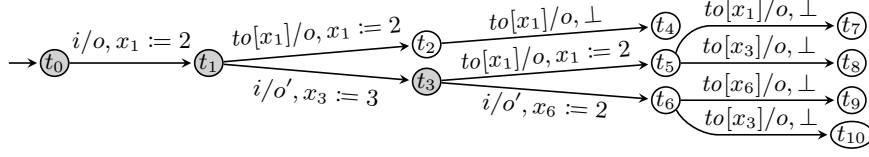
Fig. 2: Sample observation tree (we write x_i instead of x_{t_i} for all states t_i) with $\chi(t_1) = \chi(t_2) = \{x_1\}$, $\chi(t_3) = \chi(t_5) = \{x_1, x_3\}$, $\chi(t_6) = \{x_3, x_6\}$, and $\chi(t) = \emptyset$ for the other states t.

to extend it. We then give a way to construct a hypothesis from the tree, and the main loop of $L^{\#}_{\mathrm{MMT}}$. We now state the complexity of $L^{\#}_{\mathrm{MMT}}$ (see Appendix E.3 for a proof). Observe that the complexity is polynomial, if $|X^{\mathcal{M}}|$ is fixed.

Theorem 1. *The $L^{\#}_{MMT}$ algorithm terminates and returns an MMT $\mathcal{N} \overset{sym}{\approx} \mathcal{M}$ of size polynomial in $|Q^{\mathcal{M}}|$ and factorial in $|X^{\mathcal{M}}|$, in time and number of symbolic queries polynomial in $|Q^{\mathcal{M}}|, |I|$ and the length of the longest counterexample returned by the teacher, and factorial in $|X^{\mathcal{M}}|$.*

3.1   Observation tree

We describe the main data structure of our learning algorithm: a modification of the observation tree used for $L^{\#}$ [32]. Such a tree, noted $\mathcal{T}$, is an MMT that stores the observations obtained via symbolic queries. We impose that $\mathcal{T}$ is tree-shaped, and that every run is feasible. Each state q of $\mathcal{T}$ has its own timer x_q that can only be started by the incoming transition of q, and may be restarted only by a $to[x_q]$-transition. Due to its tree-shape nature, we can impose strict constraints on the sets of active and enabled timers: a timer x is active in q if and only if there is an x-spanning run traversing q, and is enabled if and only if the $to[x]$-transition is defined from q. We focus here on providing the main ideas (see Appendix D for details).

Definition 4 (Observation tree). *An* observation tree *is a tree-shaped MMT $\mathcal{T} = (X, Q, q_0, \chi, \delta)$ such that $X = \{x_q \mid q \in Q \setminus \{q_0\}\}$, every run of $\mathcal{T}$ is feasible,*

- *for all $p \xrightarrow[(x,c)]{i} q$ with $i \in I$, we have $x = x_q$,*
- *for all $q \in Q, x \in X$, we have $x \in \chi(q)$ if and only if there is an x-spanning run traversing q, and*
- *for all $q \in Q, x \in X$, we have $x \in \chi_0(q)$ if and only if $q \xrightarrow{to[x]} \in runs(\mathcal{T})$.*

We explain how to use $\mathbf{OQ^s}$ and $\mathbf{WQ^s}$ to gradually grow $\mathcal{T}$ on an example.

Example 3. Let $\mathcal{M}$ be the MMT of Figure 1 and $\mathcal{T}$ be the observation tree of Figure 2, except that $t_3 \xrightarrow{i} \notin runs(\mathcal{T})$, i.e., the subtree rooted at t_6 is not present in the tree. We construct that subtree, via $\mathbf{OQ^s}$ and $\mathbf{WQ^s}$.

First, we create the $t_3 \xrightarrow{i}$ transition. Let $w = i \cdot i$ be the unique word such that $t_0 \xrightarrow{w} t_3$ and $\mathtt{w} = \overline{w} = i \cdot i$ be the corresponding sw. As $\mathcal{M}$ is s-learnable (and, thus, complete), if follows that $q_0^{\mathcal{M}} \xrightarrow{\mathtt{w} \cdot i} \in runs(\mathcal{M})$. So, we can call $\mathbf{OQ^s}(\mathtt{w} \cdot i)$, which returns $o \cdot o' \cdot o'$. The last symbol o' must then be outputted by the new transition, i.e., we create $t_3 \xrightarrow[\perp]{i/o'} t_6$. Observe that we initially use a $\perp$ update, as, for now, we do not have any information about the potential update of the corresponding transition (here, $q_2 \xrightarrow[(x,2)]{i} q_3$) in $\mathcal{M}$. Later on, we may discover that the transition must start a timer, in which case $\perp$ will be replaced by an actual update. That is, $\perp$ is used as a sort of wildcard while learning.

We then perform a symbolic wait query in t_6, i.e., call $\mathbf{WQ^s}(\mathtt{w} \cdot i)$, which returns the set $\{(2, 3), (3, 2)\}$ meaning that the second transition of the run $q_0^{\mathcal{T}} \xrightarrow{\mathtt{w} \cdot i}$ must (re)start a timer with the constant 3, and the third transition must also (re)start a timer but with the constant 2. So, the $\perp$ of the newly created transition is replaced by $(x_6, 2)$ (as the label of the transition is an input). It remains to create the $to[x_3]$- and $to[x_6]$-transitions from t_6 by performing two symbolic output queries. We call $\mathbf{OQ^s}(\mathtt{w} \cdot i \cdot to[2])$ and $\mathbf{OQ^s}(\mathtt{w} \cdot i \cdot to[3])$ (we know that both words label runs of $\mathcal{M}$ by the symbolic wait query), and create the transitions. We thus obtain the tree of Figure 2.

Functional simulation. One can show that each state of an observation tree can be *mapped* to a state of $\mathcal{M}$, by adapting the notion of *functional simulation* of $L^{\#}$ [32]. Here, we outline what this means and refer to Appendix D.1 for more details. We have a function $f : Q^{\mathcal{T}} \to Q^{\mathcal{M}}$ such that $f(q_0^{\mathcal{T}}) = q_0^{\mathcal{M}}$ and every outgoing transition from a state $q \in Q^{\mathcal{T}}$ can be reproduced from $f(q)$ while producing the same output. In addition, since $\mathcal{T}$ and $\mathcal{M}$ may use *different* timers, we need a function $g : X^{\mathcal{T}} \to X^{\mathcal{M}}$ that maps active timers of $\mathcal{T}$ to timers of $\mathcal{M}$. We lift g to actions such that $g(i) = i$ for every $i \in I$ and $g(to[x]) = to[g(x)]$ for every $x \in \mathsf{dom}(g)$. We require that for all $q, q' \in Q^{\mathcal{T}}, x, y \in X^{\mathcal{T}}$ and $i \in I$:

1. any timer that is active in q must have a corresponding active timer in $f(q)$, i.e., if $x \in \chi^{\mathcal{T}}(q)$, then $g(x) \in \chi^{\mathcal{M}}(f(q))$;
2. two distinct timers that are active in q must be mapped to two distinct timers in $\mathcal{M}$, i.e., if $x \neq y$ and both are active in q, then $g(x) \neq g(y)$;
3. if there exists a transition $q \xrightarrow[u]{i/o} q'$ in $\mathcal{T}$, there must exist a transition reading $g(i)$ from $f(q)$ that also outputs o, i.e., $f(q) \xrightarrow[u']{g(i)/o} f(q')$ in $\mathcal{M}$. Furthermore, if $u = (x, c)$, the transition in $\mathcal{M}$ must (re)start the timer $g(x)$ at value c, i.e., $u' = (g(x), c)$. However, if $u = \perp$, we do not impose anything on u'.

One can show the following properties from the above constraints.

Lemma 3. *Let $\mathcal{M}$ be an s-learnable MMT, $\mathcal{T}$ be an observation tree, and f and g be the functions described above. Then, for any state $q \in Q^{\mathcal{T}}$, we have $|\chi^{\mathcal{T}}(q)| \leq |\chi^{\mathcal{M}}(f(q))|$ and for all $x \in \chi_0^{\mathcal{T}}(q)$, it holds that $g(x) \in \chi_0^{\mathcal{M}}(f(q))$.*

We say that a state q of $\mathcal{T}$ is *explored* once $\mathbf{WQ^s}(q)$ has been called. By definition of wait queries, it means that every enabled timer of $f(q)$ is identified in q, i.e., $|\chi_0(q)| = |\chi_0(f(q))|$. We thus obtain a *one-to-one correspondence* between $\chi_0^{\mathcal{T}}(q)$ and $\chi_0^{\mathcal{M}}(f(q))$. Define $\mathcal{E}^{\mathcal{T}}$ as the maximal set of explored states of $\mathcal{T}$. During the learning algorithm, $\mathcal{E}^{\mathcal{T}}$ induces a subtree containing $q_0^{\mathcal{T}}$. Explored states only make sense when $\mathcal{M}$ is s-learnable, as, otherwise, multiple states with different numbers of enabled timers could be mapped, via f, to a single state in $\mathcal{M}$, i.e., we do not have a one-to-one correspondence. See Appendix D.1.

Example 4. In the tree of Figure 2, the explored states are t_0, t_1, t_2, t_3, t_5, and t_6. In Example 3, t_0 to t_3, and t_5 were already explored and formed a subtree. The wait query over t_6 made it an explored state, i.e., $\mathcal{E}^{\mathcal{T}}$ is still a subtree. During the learning process, we will ensure that a wait query is only performed on a state that is an immediate successor of an explored state.

Apartness. A key aspect of $L^{\#}$ is the notion of *apartness* indicating which states have different behaviors. In the setting of regular Mealy machines, states p, p' are *apart* if they have different output responses to the same input word. In our case, we need to handle the fact that different timers in $\mathcal{T}$ can represent the same timer in $\mathcal{M}$. An example showcasing the various notions is given below.

First, recall that if two distinct timers x and y are both active in the same state of $\mathcal{T}$, it must hold that $g(x) \neq g(y)$, i.e., they correspond to different timers in $\mathcal{M}$. Hence, we say that x, y are *apart*, noted $x \mathbin{\#} y$ if there exists $q \in Q^{\mathcal{T}}$ such that $x, y \in \chi^{\mathcal{T}}(q)$. Then, defining apartness for states requires us to quantify possible equivalences of timers. For this, we rely on the concept of *matching*. Given two finite sets S and T, a relation $m \subseteq S \times T$ is a *matching* from S to T if it is an injective partial function. We write $m : S \leftrightarrow T$ if m is a matching from S to T. A matching m is *maximal* if it is total or surjective. Given two states p and p' of an observation tree $\mathcal{T}$, we consider a matching $m : \chi^{\mathcal{T}}(p) \leftrightarrow \chi^{\mathcal{T}}(p')$, denoted by abuse of notation as $m : p \leftrightarrow p'$. A matching is meant to indicate that the timers x and $m(x)$ could represent the same timer in $\mathcal{M}$. Hence, we say that m is *valid* if for all $x \in \mathsf{dom}(m)$, $\neg(x \mathbin{\#} m(x))$. We lift m to actions: $m(i) = i$ for all $i \in I$, and $m(to[x]) = to[m(x)]$ for every $x \in \mathsf{dom}(m)$.

We now move towards defining the apartness of two states p_0 and p_0', which requires to be able to "mimic" a run $p_0 \xrightarrow{w}$ from p_0', in the sense that every $to[x]$-symbol appearing in w must be replaced by a $to[y]$ with y given by some matching. Let $\pi = p_0 \xrightarrow{i_1} p_1 \xrightarrow{i_2} \cdots \xrightarrow{i_n} p_n$ and $\pi' = p_0' \xrightarrow{i_1'} p_1' \xrightarrow{i_2'} \cdots \xrightarrow{i_n'} p_n'$. We lift a matching $m : p_0 \leftrightarrow p_0'$ to runs π, π' as follows. For π' to match π, we require that for all $j \in \{1, \ldots, n\}$: *(i)* If $i_j \in I$, then $i_j' = i_j$. *(ii)* If $i_j = to[x]$ for some $x \in X^{\mathcal{T}}$ then $x \in \chi^{\mathcal{T}}(p_0)$ or $x = x_{p_k}$ for some k. Then, i_j' is either $to[m(x)]$, or $to[x_{p_k'}]$ with the same k. When they match, we write $m_{\pi'}^{\pi} : \pi \leftrightarrow \pi'$ with $m_{\pi'}^{\pi} = m \cup \{(x_{p_k}, x_{p_k'}) \mid k \leq n\}$ and $i_j' = m_{\pi'}^{\pi}(i_j)$ for every j. For a fixed $\pi \in runs(\mathcal{T})$ and m, there is at most one run $\pi' \in runs(\mathcal{T})$ such that $m_{\pi'}^{\pi} : \pi \leftrightarrow \pi'$. We denote by $read_{\pi}^{m}(p_0')$ this unique run π' if it exists.

Definition 5 (Apartness). *Two states p_0, p_0' are m-apart with $m : p_0 \leftrightarrow p_0'$, written $p_0 \#^m p_0'$, if there are $\pi = p_0 \xrightarrow{i_1} \cdots \xrightarrow[u]{i_n/o} p_n$ and $\pi' = p_0' \xrightarrow{i_1'} \cdots \xrightarrow[u']{i_n'/o'} p_n'$ with $m_{\pi'}^\pi : \pi \leftrightarrow \pi'$, and*

Structural apartness *there exists $x \in dom(m_{\pi'}^\pi)$ such that $x \not\Vdash m_{\pi'}^\pi(x)$, or*
Behavioral apartness *one of the following holds:*

$$o \neq o' \qquad\qquad \text{(outputs)}$$
$$u = (x, c) \wedge u' = (x', c') \wedge c \neq c' \qquad\qquad \text{(constants)}$$
$$p_n, p_n' \in \mathcal{E}^\mathcal{T} \wedge |\chi_0(p_n)| \neq |\chi_0(p_n')| \qquad\qquad \text{(sizes)}$$
$$p_n, p_n' \in \mathcal{E}^\mathcal{T} \wedge \exists x \in dom(m_{\pi'}^\pi) : (x \in \chi_0(p_n) \Leftrightarrow m_{\pi'}^\pi(x) \notin \chi_0(p_n')) \quad \text{(enabled)}$$

The word $w = i_1 \ldots i_n$ is called a witness of $p_0 \#^m p_0'$, noted $w \vdash p_0 \#^m p_0'$.

Example 5. In the examples, we write $x \mapsto x', y \mapsto y'$ for the matching m such that $m(x) = x'$ and $m(y) = y'$. Let $\mathcal{T}$ be the observation tree of Figure 2 and $\pi = t_0 \xrightarrow[(x_1,2)]{i/o} t_1 \xrightarrow{to[x_1]/o} t_2 \in runs(\mathcal{T})$. We compute $read_\pi^\emptyset(t_3)$ (where $\emptyset$ denotes the empty matching). The first symbol in π is i, i.e., we take the transition $t_3 \xrightarrow[(x_6,2)]{i/o'} t_6$. The second symbol in π is $to[x_1]$. Since x_1 was a fresh timer started along π, we retrieve the corresponding fresh timer in the new run, which is x_6. Hence, $\pi' = read_\pi^\emptyset(t_3) = t_3 \xrightarrow[(x_6,2)]{i/o'} t_6 \xrightarrow{to[x_6]/o} t_9$. As the first transition of π outputs o but the first transition of π' outputs $o' \neq o$, $i \vdash t_0 \#^\emptyset t_3$ by (outputs). Since $t_1, t_6 \in \mathcal{E}^\mathcal{T}$ and $|\chi_0(t_1)| = 1 \neq |\chi_0(t_6)| = 2$, $\varepsilon \vdash t_1 \#^\emptyset t_6$ and $i \vdash t_0 \#^\emptyset t_3$ by (sizes). Now, let $\sigma = t_1 \xrightarrow[(x_3,3)]{i/o'} t_3 \xrightarrow{to[x_1]} t_5$ and $\sigma' = read_{\pi'}^{x_1 \mapsto x_3}(t_3) = t_3 \xrightarrow[(x_6,2)]{i/o'} t_6 \xrightarrow{to[x_3]} t_{10}$. Since $x_1 \mapsto x_3$ is invalid, $t_1 \#^{x_1 \mapsto x_3} t_3$ is structural. We also have $i \vdash t_1 \#^{x_1 \mapsto x_3} t_3$ due to (constants).

Observe that whenever $w \vdash p \#^m p'$ and $read_{p \xrightarrow{w}}^m(p') = p' \xrightarrow{w'}$, it holds that $w' \vdash p' \#^{m^{-1}} p$. Moreover, any extension m' of m is such that $w \vdash p \#^{m'} p'$, i.e., taking a larger matching does not break the apartness, as $read_{p \xrightarrow{w}}^m(p') = read_{p \xrightarrow{w}}^{m'}(p') = p' \xrightarrow{w'}$. Finally, we claim that the definition of apartness is reasonable: when $p \#^m p'$, then $f(p) \neq f(p')$ (the two states are really distinct) or $g(x) \neq g(m(x))$ for some x (m and g do not agree). Appendix D.2 gives a proof.

Theorem 2. *Let $\mathcal{T}$ be an observation tree for an s-learnable MMT with functional simulation $\langle f, g \rangle$, $p, p' \in Q^\mathcal{T}$, and $m, m' : p \leftrightarrow p'$ matchings. Then,*

- *$w \vdash p \#^m p' \wedge m \subseteq m' \Rightarrow w \vdash p \#^{m'} p'$, and*
- *$p \#^m p' \Rightarrow f(p) \neq f(p') \vee \exists x \in dom(m) : g(x) \neq g(m(x))$.*

3.2   Hypothesis construction

In this section, we provide the construction of a hypothesis MMT $\mathcal{H}$ from $\mathcal{T}$. In short, we extend the tree such that some conditions are satisfied and we define a subset of $Q^{\mathcal{T}}$, called the *basis*, that forms the set of states of $\mathcal{H}$. Similar to $L^{\#}$ [32], we then "fold" the tree; that is, some transitions $q \to r$ must be redirected to some state p of the basis. For MMTs, we also need to map *every* timer active in r to an active timer of p. Formally, we define:

- The *basis* $\mathcal{B}^{\mathcal{T}}$ is a subtree of $Q^{\mathcal{T}}$ such that $q_0^{\mathcal{T}} \in \mathcal{B}^{\mathcal{T}}$ and $p \#^m p'$ for any $p \neq p' \in \mathcal{B}^{\mathcal{T}}$ and maximal matching $m : p \leftrightarrow p'$. By Theorem 2, we thus know that $f(p) \neq f(p')$ or $g(x) \neq g(m(x))$ for some $x \in \mathsf{dom}(m)$. As we have this for every maximal m, we *conjecture* that $f(p) \neq f(p')$. We may be wrong, i.e., $f(p) = f(p')$ but we need a matching that is currently unavailable, due to unknown active timers which will be discovered later.
- The *frontier* $\mathcal{F}^{\mathcal{T}} \subseteq Q^{\mathcal{T}}$ is the set of immediate non-basis successors of basis states. We say $p \in \mathcal{B}^{\mathcal{T}}$ and $r \in \mathcal{F}^{\mathcal{T}}$ are *compatible* under a maximal matching m if $\neg(p \#^m r)$. We write $compat^{\mathcal{T}}(r)$ for the set of all such pairs (p, m).

During the learning algorithm, the tree will be extended such that a complete MMT can be constructed from $\mathcal{T}$. We will ensure that *(A)* each basis and frontier state is explored, i.e., $\mathcal{B}^{\mathcal{T}} \cup \mathcal{F}^{\mathcal{T}} \subseteq \mathcal{E}^{\mathcal{T}}$, in order to discover timers as fast as possible, *(B)* the basis is *complete*, in the sense that $p \xrightarrow{i}$ is defined for every $i \in I \cup TO[\chi_0^{\mathcal{T}}(p)]$, and *(C)* for every $r \in \mathcal{F}^{\mathcal{T}}$, $compat^{\mathcal{T}}(r) \neq \emptyset$ and $|\chi^{\mathcal{T}}(p)| = |\chi^{\mathcal{T}}(r)|$ for every $(p, m) \in compat^{\mathcal{T}}(r)$. All of these constraints can be obtained via $\mathbf{OQ^s}$ and $\mathbf{WQ^s}$, as we illustrate in the next example.

Example 6. In order to simplify the explanations, we assume from now on that a call to $\mathbf{OQ^s}(\mathtt{w} \cdot \mathtt{i})$ with $\mathtt{i} \in I$ automatically adds the corresponding transition to $\mathcal{T}$, and that a call to $\mathbf{WQ^s}(\mathtt{w})$ automatically calls $\mathbf{OQ^s}(\mathtt{w} \cdot to[j])$, for every $to[j]$ deduced from the wait query, modifies updates accordingly, and adds the new explored states to $\mathcal{E}^{\mathcal{T}}$. Recall that $\mathcal{E}^{\mathcal{T}}$ is exactly the set of states in which we performed a $\mathbf{WQ^s}$. Moreover, we let $\mathbf{OQ^s}(q, i)$ denote $\mathbf{OQ^s}(\mathtt{w} \cdot \mathtt{i})$ and $\mathbf{WQ^s}(q)$ denote $\mathbf{WQ^s}(\mathtt{w})$ with $\mathtt{w}$ such that $q_0^{\mathcal{T}} \xrightarrow{\mathtt{w}} q \in runs(\mathcal{T})$.

  Let the MMT of the teacher be the MMT $\mathcal{M}$ of Figure 1 and $\mathcal{T}$ be the observation tree of Figure 2. One can check that t_0, t_1, and t_3 are all pairwise apart under any maximal matching. So, $\mathcal{B}^{\mathcal{T}} = \{t_0, t_1, t_3\}$ and $\mathcal{F}^{\mathcal{T}} = \{t_2, t_5, t_6\}$. The basis states are highlighted in gray in the figure. Moreover, we have $compat^{\mathcal{T}}(t_2) = \{(t_1, x_1 \mapsto x_1), (t_3, x_1 \mapsto x_1)\}$, and $compat^{\mathcal{T}}(t_5) = compat^{\mathcal{T}}(t_6) = \emptyset$. We thus *promote* t_6 by moving it from the frontier to the basis, i.e., $\mathcal{B}^{\mathcal{T}}$ is now $\{t_0, t_1, t_3, t_6\}$. In order to satisfy *(B)*, we call $\mathbf{OQ^s}(t_6, i)$ and add a new transition $t_6 \xrightarrow[\perp]{i/o'} t_{11}$. We call $\mathbf{WQ^s}(t_9), \mathbf{WQ^s}(t_{10})$, and $\mathbf{WQ^s}(t_{11})$, which yield $\chi(t_9) = \chi_0(t_9) = \{x_3\}$, $\chi(t_{10}) = \emptyset$, and $\chi(t_{11}) = \chi_0(t_{11}) = \{x_3, x_{11}\}$. Hence, we get *(A)* with $\mathcal{F}^{\mathcal{T}} = \{t_2, t_5, t_9, t_{10}, t_{11}\}$, $compat^{\mathcal{T}}(t_2) = \{(t_1, x_1 \mapsto x_1), (t_3, x_1 \mapsto x_1)\}$, $compat^{\mathcal{T}}(t_5) = \{(t_6, x_6 \mapsto x_1, x_3 \mapsto x_3)\}$, $compat^{\mathcal{T}}(t_9) = \emptyset$, $compat^{\mathcal{T}}(t_{10}) = \{(t_0, \emptyset)\}$, and $compat^{\mathcal{T}}(t_{11}) = \{(t_6, x_6 \mapsto x_{11}, x_3 \mapsto x_3)\}$.
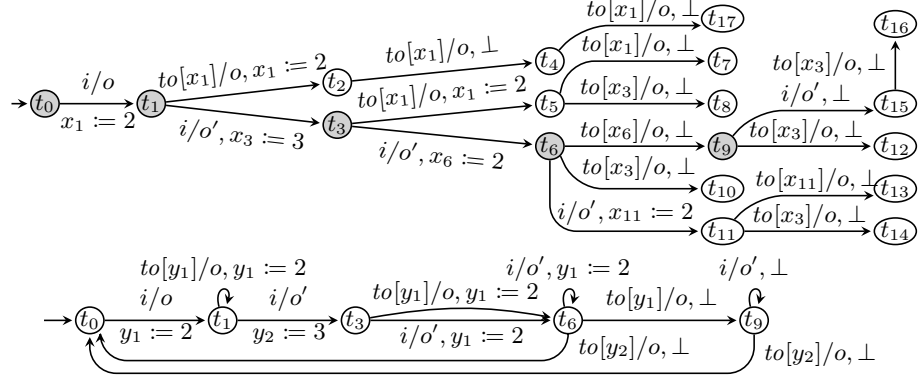
Fig. 3: On top, an observation tree from which the hypothesis MMT at the bottom is constructed, with $y_1 = [\![x_1]\!]_{\equiv}$ and $y_2 = [\![x_3]\!]_{\equiv}$. Basis states are highlighted with a gray background.

Observe that *(C)* is not satisfied, due to $(t_3, m) \in compat^{\mathcal{T}}(t_2)$ but $|\chi(t_3)| = 2$ while $|\chi(t_2)| = 1$. In order to resolve this issue, we *replay* the run $t_3 \xrightarrow{to[x_1]to[x_3]} t_8$ (witnessing that x_3 is active in t_3 and eventually times out) from the state t_2. As we did when defining apartness, we rely on a mapping that we update after each transition. The starting mapping is given by the compatible function. Here, consider $m = (x_1 \mapsto x_1)$. Since $t_2 \xrightarrow{to[x_1]} t_4$ is already defined (and the transition does not change the mapping), let us focus on replaying $t_5 \xrightarrow{to[x_3]}$ from t_4. We first call $\mathbf{WQ^s}(t_4)$ to learn $\chi_0(t_4) = \chi(t_4) = \{x_1\}$. As m dictates $x_1 \mapsto x_1$, it is impossible to replay $t_5 \xrightarrow{to[x_3]}$ from t_4. Hence, $t_3 \#^m t_2$ and the set of compatible states of t_2 is reduced. In general, replaying a run may have two results: finding a new active timer in a state, or a new apartness pair (either due to a fail, as we illustrated, or due to newly defined transitions). In any case, the set of compatible states of a state gets reduced. See Appendix E.1 for more details.

We now explain, via an example, how to construct a hypothesis $\mathcal{H}$ such that $Q^{\mathcal{H}} = \mathcal{B}^{\mathcal{T}}$. The idea is to pick a $(p, m) \in compat^{\mathcal{T}}(r)$ for each frontier state r. Then, the unique transition $q \xrightarrow{i} r$ in $\mathcal{T}$ becomes $q \xrightarrow{i} p$ in $\mathcal{H}$. We also globally rename the timers according to m.

Example 7. Let $\mathcal{M}$ be the MMT of Figure 1 and $\mathcal{T}$ be the observation tree of Figure 3, with $\mathcal{B}^{\mathcal{T}} = \{t_0, t_1, t_3, t_6, t_9\}$ and $\mathcal{F}^{\mathcal{T}} = \{t_2, t_5, t_{10}, t_{11}, t_{12}, t_{15}\}$. Moreover, $compat^{\mathcal{T}}(t_2) = \{(t_1, x_1 \mapsto x_1)\}$, $compat^{\mathcal{T}}(t_{10}) = compat^{\mathcal{T}}(t_{12}) = \{(t_0, \emptyset)\}$, $compat^{\mathcal{T}}(t_5) = \{(t_6, x_6 \mapsto x_1, x_3 \mapsto x_3)\}$, $compat^{\mathcal{T}}(t_{15}) = \{(t_9, x_3 \mapsto x_3)\}$, and $compat^{\mathcal{T}}(t_{11}) = \{(t_6, x_6 \mapsto x_{11}, x_3 \mapsto x_3)\}$.

We construct $\mathcal{H}$ with $Q^{\mathcal{H}} = \mathcal{B}^{\mathcal{T}}$. While defining the transitions $q \to q'$ is easy when $q, q' \in \mathcal{B}^{\mathcal{T}}$, we have to redirect the transition to some basis state when $q' \in \mathcal{F}^{\mathcal{T}}$. To do so, we first define a map $\mathbf{h} : \mathcal{F}^{\mathcal{T}} \to \mathcal{B}^{\mathcal{T}}$, and an equivalence

relation $\equiv$ over the set of active timers of the basis and the frontier. For each $r \in \mathcal{F}^{\mathcal{T}}$, we pick $(p, m) \in compat^{\mathcal{T}}(r)$, define $\mathbf{h}(r) = p$, and add $x \equiv m(x)$ for every $x \in \mathsf{dom}(m)$ (and compute the symmetric and transitive closure of $\equiv$). Here, we obtain $\mathbf{h}(t_2) = t_1, \mathbf{h}(t_5) = \mathbf{h}(t_{11}) = t_6, \mathbf{h}(t_{10}) = \mathbf{h}(t_{12}) = t_0, \mathbf{h}(t_{15}) = t_9, x_1 \equiv x_6 \equiv x_{11}$, and $x_3 \equiv x_3$. We check whether we have $x \equiv y$ and $x \mathbin{\char"6F} y$, in which case, we restart again by picking some different (p, m). Here, this does not hold and we construct $\mathcal{H}$ by copying the transitions starting from a basis state (while folding the tree when required), except that a timer x is replaced by its equivalence class $[\![x]\!]_{\equiv}$. Figure 3 gives the resulting $\mathcal{H}$.

We highlight that it is *not* always possible to construct $\equiv$ such that $\neg(x \mathbin{\char"6F} y)$ for every $x \equiv y$. In that case, we instead construct a *generalized* MMT, in which every transition can arbitrarily rename the active timers. The size of that generalized MMT is also $|\mathcal{B}^{\mathcal{T}}|$. From the generalized MMT, a classical MMT can be constructed of size $n! \cdot |\mathcal{B}^{\mathcal{T}}|$, with $n = \max_{p \in \mathcal{B}^{\mathcal{T}}} |\chi^{\mathcal{T}}(p)|$. See Appendix E.2. We observed on practical examples that a valid $\equiv$ can be constructed.

3.3   Main loop

We now give the main loop of $L^{\#}_{\mathrm{MMT}}$. We initialize $\mathcal{T}$ to only contain $q_0^{\mathcal{T}}$, $\mathcal{B}^{\mathcal{T}} = \mathcal{E}^{\mathcal{T}} = \{q_0^{\mathcal{T}}\}$, and $\mathcal{F}^{\mathcal{T}} = \emptyset$. The main loop is split into two parts:

Refinement loop The *refinement loop* extends the tree to obtain the conditions *(A)* to *(C)* (see page 12), by performing the following operations, in this order, until no more changes are possible:

  Seismic If we discover a new active timer in a basis state, then it may be that $\neg(q \mathbin{\#^m} q')$ for some $q, q' \in \mathcal{B}^{\mathcal{T}}$ and maximal m, due to the new timer. To avoid this, we reset the basis back to $\{q_0^{\mathcal{T}}\}$, as soon as a new timer is found, without removing states from $\mathcal{T}$.

  Promotion If $compat^{\mathcal{T}}(r)$ is empty for some frontier state r, then we know that $q \mathbin{\#^m} r$ for every $q \in \mathcal{B}^{\mathcal{T}}$ and maximal matching $m : q \leftrightarrow r$. Hence, we promote r to the basis.

  Completion If an i-transition is missing from some basis state p, we complete the basis with that transition. Recall that it is sufficient to only check for i that are inputs.

  Active timers We ensure $|\chi^{\mathcal{T}}(p)| = |\chi^{\mathcal{T}}(r)|$ for every $(p, \cdot) \in compat^{\mathcal{T}}(r)$.

Hypothesis and equivalence We call $\mathbf{EQ^s}(\mathcal{H})$ with $\mathcal{H}$ a hypothesis. If the answer is **yes**, we return $\mathcal{H}$. Otherwise, we process the counterexample, as now explain in an example.

Example 8. Let $\mathcal{T}$ be the observation tree of Figure 4 with $\mathcal{B}^{\mathcal{T}} = \{t_0, t_1\}$, $\mathcal{F}^{\mathcal{T}} = \{t_2, t_3\}$, and $compat^{\mathcal{T}}(t_2) = compat^{\mathcal{T}}(t_3) = \{(t_1, x_1 \mapsto x_1)\}$. Figure 4 also gives the MMT constructed from $\mathcal{T}$, which is not symbolically equivalent to the MMT of Figure 1, with $\mathbf{w} = i \cdot i \cdot to[1] \cdot to[2]$ as a counterexample. We extend $\mathcal{T}$ such that $\mathbf{w}$ can be read in $\mathcal{T}$. That is, we call $\mathbf{WQ^s}(t_5)$ and discover that the transition from t_1 to t_3 must start the timer x_3. Hence, t_3 has no compatible state anymore
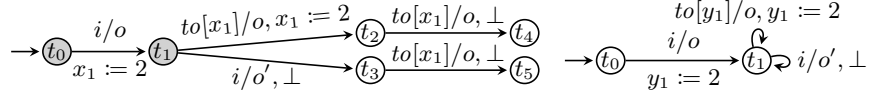
Fig. 4: On the left, an observation tree from which the hypothesis MMT on the right is constructed, with $y_1 = [\![x_1]\!]_\equiv$. Basis states have a gray background.

(i.e., we found a new apartness pair) and gets promoted. After completing the tree and performing $\mathbf{WQ^s}$ on the frontier states, we get the tree from Figure 2.

In our example, simply adding the symbolic word provided by the teacher was enough to discover a new apartness pair (meaning that the hypothesis can no longer be constructed). However, there may be cases where we need to *replay* (see Example 6) a part of the newly added run $t_0 \xrightarrow{\mathbf{w}}$: split the run into $t_0 \xrightarrow{\mathbf{u}} t \xrightarrow{\mathbf{v}}$ with $t \in \mathcal{F}^{\mathcal{T}}$ and replay $t \xrightarrow{\mathbf{v}}$ from the state compatible with t that was selected to build the hypothesis. Repeat this principle until a compatible set is reduced.

4   Implementation and Experiments

We have implemented the $L_{\mathrm{MMT}}^{\#}$ algorithm as an open-source tool.[5] As we do not yet have a timed conformance testing algorithm for checking equivalence between a hypothesis and the teacher's MMT, we utilize a BFS algorithm to check for equivalence between the two MMTs.[6] We have evaluated the performance of our tool on a selection of both real and synthetic benchmarks.

We use the AKM, TCP and Train benchmarks from [31], and the CAS, Light and PC benchmarks from [1]. These have also been used for experimental evaluation by [25,31,33] and can be described as Mealy machines with a single timer. We introduce two additional benchmarks with two timers: a model of an FDDI station, and the MMT of Figure 1. We refer to Appendix F for details on our FDDI benchmark. We did not include the FDDI 2 process benchmark from [33], as our implementation cannot (yet) handle the corresponding gMMTs. Finally, we learned instances of the Oven and WSN MMLTs benchmarks from [25]. We modified the timing parameters to generate smaller MMTs with a single timer (MM1Ts). For each experiment, we record the number of $\mathbf{OQ^s}$, $\mathbf{WQ^s}$, $\mathbf{EQ^s}$, and the time taken to finish the experiment. Note, in practice, a $\mathbf{WQ^s}$, in addition to returning the list of timeouts and their constants, also provides the outputs of the timeout transitions. This is straightforward, as a $\mathbf{WQ^s}$ must necessarily trigger the timeouts in order to observe them. Thus, we do not count the $\mathbf{OQ^s}$ associated with a $\mathbf{WQ^s}$.[7]

---

[5] See: `https://gitlab.science.ru.nl/bharat/mmt_lsharp` and Zenodo [17].

[6] That is, seeking a difference in behavior in the product of the hypothesis and the SUL zone automaton.

[7] An upper bound on the number of $\mathbf{OQ^s}$ can be obtained by adding $|X||\mathbf{WQ^s}|$ to the values for $|\mathbf{OQ^s}|$.

Model	$\lvert Q \rvert$	$\lvert I \rvert$	$\lvert X \rvert$	$\lvert \mathbf{WQ^s} \rvert$	$\lvert \mathbf{OQ^s} \rvert$	$\lvert \mathbf{EQ^s} \rvert$	Time[ms]	$\lvert \mathbf{MQ} \rvert$ [33]	$\lvert \mathbf{EQ} \rvert$ [33]
AKM	4	5	1	22	35	2	684	12263	11
CAS	8	4	1	60	89	3	1344	66067	17
Light	4	2	1	10	13	2	302	3057	7
PC	8	9	1	75	183	4	2696	245134	23
TCP	11	8	1	123	366	8	3182	11300	15
Train	6	3	1	32	28	3	1559		
MMT of Fig. 1	3	1	2	11	5	2	1039	-	-
FDDI 1-station	9	2	2	32	20	1	1105	118193	8
Oven	12	5	1	907	317	3	9452	-	-
WSN	9	4	1	175	108	4	3291	-	-

Table 1: Experimental Results.

Table 1 lists the results of our experiments, and also the number of concrete membership and equivalence queries used by Waga's [33]. Comparison of learning algorithms for timed systems is complicated. First of all, we need to convert the numbers of symbolic $L^{\#}_{\text{MMT}}$ queries to concrete queries. This can be done using the polynomial bounds given in Appendix C.[8] However, for MM1Ts each symbolic query can be implemented using a single concrete query (see Lemma 3 in [31]). Several algorithms presented in the literature learn TAs [1, 5, 33, 34]. Typically, a TA model of some system will have different numbers of states and transitions than an MMT model: Mealy machines tend to be more compact than TAs, but the use of timers may lead to more states than a TA encoding. Therefore we cannot just compare numbers of queries. As a final complication, observe that equivalence queries can be implemented in different ways, which may affect the total number of queries required for learning. MMLTs [25] can be converted to equivalent MM1Ts [31], but this may blow up of the number of states. Since $L^{\#}_{\text{MMT}}$ learns the MM1Ts, it is less efficient than the MMLT learner of [25] which learns the more compact MMLT representations. However, $L^{\#}_{\text{MMT}}$ can handle a larger class of models.

5   Future work

A major challenge in $L^{\#}_{\text{MMT}}$ is to infer the update on transitions. This would become easier if we know in advance that timers can only be started by specific inputs, akin to what is done in *event recording automata* (ERA) [4] for timed automata. Although, as discussed in [31], the restrictions of ERAs make it hard to capture the timing behavior of standard network protocols, it would be interesting to study the theoretical complexity of learning a system that can be modelled by ERAs as well as MMTs. A more interesting trail would be to allow transitions to start multiple timers, instead of a single one, which would permit

---

[8] For the FDDI protocol, we can show that we need at most 1282 concrete queries to do our symbolic wait and output queries in total.

more complex models (such as those resulting of parallel composition of simpler models) to be learned.

References

1. Aichernig, B.K., Pferscher, A., Tappler, M.: From Passive to Active: Learning Timed Automata Efficiently. In: Lee, R., Jha, S., Mavridou, A. (eds.) Proceedings of the 12th International Symposium NASA Formal Methods, NFM 2020. Lecture Notes in Computer Science, vol. 12229, pp. 1–19. Springer (2020). https://doi.org/10.1007/978-3-030-55754-6_1
2. Alur, R.: Timed Automata. In: Halbwachs, N., Peled, D.A. (eds.) Proceedings of the 11th International Conference Computer Aided Verification, CAV 1999. Lecture Notes in Computer Science, vol. 1633, pp. 8–22. Springer (1999). https://doi.org/10.1007/3-540-48683-6_3
3. Alur, R., Dill, D.L.: A Theory of Timed Automata. Theoretical Computer Science **126**(2), 183–235 (1994). https://doi.org/10.1016/0304-3975(94)90010-8
4. Alur, R., Fix, L., Henzinger, T.A.: Event-Clock Automata: A Determinizable Class of Timed Automata. Theoretical Computer Science **211**(1-2), 253–273 (1999). https://doi.org/10.1016/S0304-3975(97)00173-4
5. An, J., Chen, M., Zhan, B., Zhan, N., Zhang, M.: Learning One-Clock Timed Automata. In: Biere, A., Parker, D. (eds.) Proceedings of the 26th International Conference Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2020. Lecture Notes in Computer Science, vol. 12078, pp. 444–462. Springer (2020). https://doi.org/10.1007/978-3-030-45190-5_25
6. Angluin, D.: Learning Regular Sets from Queries and Counterexamples. Information and Computation **75**(2), 87–106 (1987). https://doi.org/10.1016/0890-5401(87)90052-6
7. Bruyère, V., Pérez, G.A., Staquet, G., Vaandrager, F.W.: Automata with Timers. In: Petrucci, L., Sproston, J. (eds.) Proceedings of the 21st International Conference Formal Modeling and Analysis of Timed Systems, FORMATS 2023. Lecture Notes in Computer Science, vol. 14138, pp. 33–49. Springer (2023). https://doi.org/10.1007/978-3-031-42626-1_3
8. Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R. (eds.): Handbook of Model Checking. Springer (2018). https://doi.org/10.1007/978-3-319-10575-8
9. Daws, C., Olivero, A., Tripakis, S., Yovine, S.: The Tool KRONOS. In: Alur, R., Henzinger, T.A., Sontag, E.D. (eds.) Hybrid Systems III: Verification and Control, Proceedings of the DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems. Lecture Notes in Computer Science, vol. 1066, pp. 208–219. Springer (1995). https://doi.org/10.1007/BFB0020947
10. Dierl, S., Howar, F.M., Kauffman, S., Kristjansen, M., Larsen, K.G., Lorber, F., Mauritz, M.: Learning Symbolic Timed Models from Concrete Timed Data. In: Rozier, K.Y., Chaudhuri, S. (eds.) Proceedings of the 15th International Symposium NASA Formal Methods, NFM 2023. Lecture Notes in Computer Science, vol. 13903, pp. 104–121. Springer (2023). https://doi.org/10.1007/978-3-031-33170-1_7
11. Dill, D.L.: Timing Assumptions and Verification of Finite-State Concurrent Systems. In: Sifakis, J. (ed.) Proceedings of the International Workshop Automatic Verification Methods for Finite State Systems. Lecture Notes in Computer Science, vol. 407, pp. 197–212. Springer (1989). https://doi.org/10.1007/3-540-52148-8_17

12. Ferreira, T., Brewton, H., D'Antoni, L., Silva, A.: Prognosis: closed-box analysis of network protocol implementations. In: Kuipers, F.A., Caesar, M.C. (eds.) Proceedings of the ACM SIGCOMM 2021 Conference. pp. 762–774. ACM (2021). `https://doi.org/10.1145/3452296.3472938`

13. Fiterau-Brostean, P., Howar, F.: Learning-Based Testing the Sliding Window Behavior of TCP Implementations. In: Petrucci, L., Seceleanu, C., Cavalcanti, A. (eds.) Proceedings of the Critical Systems: Formal Methods and Automated Verification - Joint 22nd International Workshop on Formal Methods for Industrial Critical Systems FMICS and 17th International Workshop on Automated Verification of Critical Systems AVoCS 2017. Lecture Notes in Computer Science, vol. 10471, pp. 185–200. Springer (2017). `https://doi.org/10.1007/978-3-319-67113-0_12`

14. Fiterau-Brostean, P., Janssen, R., Vaandrager, F.W.: Combining Model Learning and Model Checking to Analyze TCP Implementations. In: Chaudhuri, S., Farzan, A. (eds.) Proceedings of the 28th International Conference Computer Aided Verification, CAV 2016. Lecture Notes in Computer Science, vol. 9780, pp. 454–471. Springer (2016). `https://doi.org/10.1007/978-3-319-41540-6_25`

15. Fiterau-Brostean, P., Jonsson, B., Merget, R., de Ruiter, J., Sagonas, K., Somorovsky, J.: Analysis of DTLS Implementations Using Protocol State Fuzzing. In: Capkun, S., Roesner, F. (eds.) Proceedings of the 29th USENIX Security Symposium, USENIX Security 2020. pp. 2523–2540. USENIX Association (2020), `https://www.usenix.org/conference/usenixsecurity20/presentation/fiterau-brostean`

16. Fiterau-Brostean, P., Lenaerts, T., Poll, E., de Ruiter, J., Vaandrager, F.W., Verleg, P.: Model learning and model checking of SSH implementations. In: Erdogmus, H., Havelund, K. (eds.) Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software. pp. 142–151. ACM (2017). `https://doi.org/10.1145/3092282.3092289`

17. Garhewal, B.: L# MMT artifact (2024). `https://doi.org/10.5281/ZENODO.10647627`, `https://zenodo.org/doi/10.5281/zenodo.10647627`

18. Garhewal, B., Vaandrager, F.W., Howar, F., Schrijvers, T., Lenaerts, T., Smits, R.: Grey-Box Learning of Register Automata. In: Dongol, B., Troubitsyna, E. (eds.) Integrated Formal Methods - 16th International Conference, IFM 2020, Lugano, Switzerland, November 16-20, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12546, pp. 22–40. Springer (2020). `https://doi.org/10.1007/978-3-030-63461-2_2`

19. Grinchtein, O., Jonsson, B., Leucker, M.: Learning of Event-Recording Automata. In: Lakhnech, Y., Yovine, S. (eds.) Proceedings of the Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems - Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004. Lecture Notes in Computer Science, vol. 3253, pp. 379–396. Springer (2004). `https://doi.org/10.1007/978-3-540-30206-3_26`

20. Grinchtein, O., Jonsson, B., Leucker, M.: Learning of event-recording automata. Theoretical Computer Science **411**(47), 4029–4054 (2010). `https://doi.org/10.1016/J.TCS.2010.07.008`

21. Grinchtein, O., Jonsson, B., Pettersson, P.: Inference of Event-Recording Automata Using Timed Decision Trees. In: Baier, C., Hermanns, H. (eds.) Proceedings of the 17th International Conference Concurrency Theory, CONCUR 2006. Lecture Notes in Computer Science, vol. 4137, pp. 435–449. Springer (2006). `https://doi.org/10.1007/11817949_29`

22. Henry, L., Jéron, T., Markey, N.: Active Learning of Timed Automata with Unobservable Resets. In: Bertrand, N., Jansen, N. (eds.) Proceedings of the 18th International Conference Formal Modeling and Analysis of Timed Systems, FORMATS 2020. Lecture Notes in Computer Science, vol. 12288, pp. 144–160. Springer (2020)

23. Howar, F., Steffen, B.: Active Automata Learning in Practice - An Annotated Bibliography of the years 2011 to 2016. In: Bennaceur, A., Hähnle, R., Meinke, K. (eds.) Machine Learning for Dynamic Software Analysis: Potentials and Limits - International Dagstuhl Seminar 16172, Dagstuhl Castle, Germany, April 24-27, 2016, Revised Papers. Lecture Notes in Computer Science, vol. 11026, pp. 123–148. Springer (2018). https://doi.org/10.1007/978-3-319-96562-8_5

24. Johnson, M.J.: Proof that Timing Requirements of the FDDI Token Ring Protocol are Satisfied. IEEE Transactions on Computers **35**(6), 620–625 (1987). https://doi.org/10.1109/TCOM.1987.1096832

25. Kogel, P., Klös, V., Glesner, S.: Learning Mealy Machines with Local Timers. In: Li, Y., Tahar, S. (eds.) Proceedings of the 24th International Conference on Formal Engineering Methods Formal Methods and Software Engineering, ICFEM 2023. Lecture Notes in Computer Science, vol. 14308, pp. 47–64. Springer (2023). https://doi.org/10.1007/978-981-99-7584-6_4

26. Maler, O., Pnueli, A.: On Recognizable Timed Languages. In: Walukiewicz, I. (ed.) Proceedings of the 7th International Conference Foundations of Software Science and Computation Structures, FOSSACS 2004. Lecture Notes in Computer Science, vol. 2987, pp. 348–362. Springer (2004). https://doi.org/10.1007/978-3-540-24727-2_25

27. de Ruiter, J., Poll, E.: Protocol State Fuzzing of TLS Implementations. In: Jung, J., Holz, T. (eds.) Proceedings of the 24th USENIX Security Symposium, USENIX Security 15. pp. 193–206. USENIX Association (2015), https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/de-ruiter

28. Shahbaz, M., Groz, R.: Inferring Mealy Machines. In: Cavalcanti, A., Dams, D. (eds.) Proceedings of the 16th Formal Methods, FM 2009. Lecture Notes in Computer Science, vol. 5850, pp. 207–222. Springer (2009). https://doi.org/10.1007/978-3-642-05089-3_14

29. Tappler, M., Aichernig, B.K., Larsen, K.G., Lorber, F.: Time to Learn - Learning Timed Automata from Tests. In: André, É., Stoelinga, M. (eds.) Proceedings of the 17th International Conference Formal Modeling and Analysis of Timed Systems, FORMATS 2019. Lecture Notes in Computer Science, vol. 11750, pp. 216–235. Springer (2019). https://doi.org/10.1007/978-3-030-29662-9_13

30. Vaandrager, F.W.: Model learning. Communications of the ACM **60**(2), 86–95 (2017). https://doi.org/10.1145/2967606

31. Vaandrager, F.W., Bloem, R., Ebrahimi, M.: Learning Mealy Machines with One Timer. In: Leporati, A., Martín-Vide, C., Shapira, D., Zandron, C. (eds.) Proceedings of the 15th International Conference Language and Automata Theory and Applications, LATA 2021. Lecture Notes in Computer Science, vol. 12638, pp. 157–170. Springer (2021). https://doi.org/10.1007/978-3-030-68195-1_13

32. Vaandrager, F.W., Garhewal, B., Rot, J., Wißmann, T.: A New Approach for Active Automata Learning Based on Apartness. In: Fisman, D., Rosu, G. (eds.) Proceedings of the 28th International Conference Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2022. Lecture Notes in Computer Science, vol. 13243, pp. 223–243. Springer (2022). https://doi.org/10.1007/978-3-030-99524-9_12

33. Waga, M.: Active Learning of Deterministic Timed Automata with Myhill-Nerode Style Characterization. In: Enea, C., Lal, A. (eds.) Proceedings of the 35th International Conference Computer Aided Verification, CAV 2023. Lecture Notes in Computer Science, vol. 13964, pp. 3–26. Springer (2023). https://doi.org/10.1007/978-3-031-37706-8_1
34. Xu, R., An, J., Zhan, B.: Active Learning of One-Clock Timed Automata Using Constraint Solving. In: Bouajjani, A., Holík, L., Wu, Z. (eds.) Proceedings of the 20th International Symposium Automated Technology for Verification and Analysis, ATVA 2022. Lecture Notes in Computer Science, vol. 13505, pp. 249–265. Springer (2022). https://doi.org/10.1007/978-3-031-19992-9_16

A   More details on equivalence of two MMTs

In this section, we give more details about equivalence of two MMTs. We first define the classical notion of *timed* equivalence, before showing that the symbolic equivalence introduced in Section 2.2 refines this timed equivalence. Moreover, we give a counterexample for the other direction. That is, we prove that timed equivalence does not imply symbolic equivalence.

A.1   Timed equivalence

In order to define timed equivalence between two MMTs that do not use the same timers, we first introduce a way to "hide" the timeouts behind the delays. As it is not possible to let time elapse when a timer times out, a complete MMT $\mathcal{M}$ can be assumed to automatically process timeouts when they occur. Let $w = d_1 i_1 \cdots i_n d_{n+1}$ be a timed word over I (and not over $A(\mathcal{M})$), called a *timed input word* (tiw, in short). When a timer times out while reading w, $\mathcal{M}$ automatically triggers the corresponding timeout-transition. As more than one timer may time out simultaneously (or some timers time out at the same time an input must be processed), a single tiw w can induce many timed runs. Let $\circledcirc runs(w)$ be the set of all timed runs induced by w. Each $\rho \in \circledcirc runs(w)$ yields a *timed output word* (tow, in short), which is a timed word over O, obtained by collecting the outputs of every transition and the delays between them. Let $toutputs(w)$ be the set of all tows produced by the runs in $\circledcirc runs(w)$. Then, we say that two complete MMTs $\mathcal{M}$ and $\mathcal{N}$ are *equivalent*, noted $\mathcal{M} \overset{\text{time}}{\approx} \mathcal{N}$, whenever, for all tiws w, $toutputs^{\mathcal{M}}(w) = toutputs^{\mathcal{N}}(w)$.

A.2   Symbolic equivalence refines timed equivalence

Lemma 4. *Let $\mathcal{M}$ and $\mathcal{N}$ be two complete MMTs. If $\mathcal{M} \overset{sym}{\approx} \mathcal{N}$, then $\mathcal{M} \overset{time}{\approx} \mathcal{N}$.*

Proof. Towards a contradiction, assume $\mathcal{M} \overset{\text{sym}}{\approx} \mathcal{N}$ but $\mathcal{M} \overset{\text{time}}{\not\approx} \mathcal{N}$. Then, there must exist a tiw w such that $toutputs^{\mathcal{M}}(w) \neq toutputs^{\mathcal{N}}(w)$. Without loss of generality, there is a timed run

$$\rho = (q_0^{\mathcal{M}}, \emptyset) \xrightarrow{d_1} (q_0^{\mathcal{M}}, \emptyset) \xrightarrow[u_1]{i_1/o_1} \cdots \xrightarrow[u_n]{i_n/o_n} (q_n, \kappa_n) \xrightarrow{d_{n+1}} (q_n, \kappa_n - d_{n+1})$$

in $\circledcirc runs^{\mathcal{M}}(w)$ such that $tow(\rho) = d_1 \cdot o_1 \cdots d_n \cdot o_n \cdot d_{n+1}$ and $tow(\rho) \notin toutputs^{\mathcal{N}}(w)$.[9] Let $\mathtt{w} = \mathtt{i_1} \cdots \mathtt{i_n}$ be the symbolic word of $i_1 \cdots i_n$.

Let us consider the longest possible timed run of $\mathcal{N}$

$$\rho' = (q_0^{\mathcal{N}}, \emptyset) \xrightarrow{d_1} (q_0^{\mathcal{N}}, \emptyset) \xrightarrow[u_1']{i_1'/o_1} \cdots \xrightarrow[u_j']{i_j'/o_j} (q_j', \kappa_j') \xrightarrow{d_{j+1}} (q_j', \kappa_j' - d_{j+1})$$

such that

---

[9] Recall that $tow(\rho)$ denotes the tow produced by the timed run ρ.

- it reads a prefix of $i_1 \cdots i_n$ (up to the names of the timers) with

$$i'_k = \begin{cases} i_k & \text{if } i_k \in I \\ to[x'] & \text{for some } x' \in \chi^{\mathcal{N}}(q'_{k-1}) \text{ if } i_k = to[x] \text{ for some } x \in \chi^{\mathcal{M}}(q_{k-1}) \end{cases}$$

  for all $k \in \{1, \ldots, j\}$,
- the delays d_k, $k \in \{1, \ldots, j+1\}$, and the outputs o_k, $k \in \{1, \ldots, j\}$, are the same as in the timed run ρ, and
- the symbolic word of $i'_1 \cdots i'_j$ is equal to $\mathtt{i_1} \cdots \mathtt{i_j}$.

Such a timed run ρ' exists with $j \geq 0$, and we have $j < n$. Towards a contradiction, let us show that we can extend it.

First, we argue that for any $d \in \mathbb{R}^{\geq 0}$

$$\exists x \in \chi^{\mathcal{M}}(q_j) : (\kappa_j - d)(x) = 0 \quad \Leftrightarrow \quad \exists x' \in \chi^{\mathcal{N}}(q'_j) : (\kappa'_j - d)(x') = 0. \quad (1)$$

We show the $\Rightarrow$ direction. The other direction can be obtained with similar arguments. Since $(\kappa_j - d)(x) = 0$, we have that $x \in \chi_0^{\mathcal{M}}(q_j)$. As $\mathcal{M}$ is complete, it follows that $q_j \xrightarrow{to[x]} \in runs(\mathcal{M})$ and we can take the transition $(q_j, \kappa_j - d) \xrightarrow{to[x]}$. Thus, for some $k \in \{1, \ldots, j-1\}$, the sub-run

$$q_{k-1} \xrightarrow[(x,c)]{i_k} \cdots \xrightarrow{i_j} q_j \xrightarrow{to[x]}$$

is x-spanning. Hence, $\overline{i_1 \cdots i_j \cdot to[x]} = \mathtt{i_1} \cdots \mathtt{i_j} \cdot to[k]$. As $\mathcal{M} \overset{\mathrm{sym}}{\approx} \mathcal{N}$, we deduce that there exists some timer x' such that the run $q_0^{\mathcal{N}} \xrightarrow{i'_1 \cdots i'_j \cdot to[x']}$ is feasible with $\overline{i'_1 \cdots i'_j \cdot to[x']} = \overline{i_1 \cdots i_j \cdot to[x]}$ (i.e., $\mathcal{M}$ and $\mathcal{N}$ read the same symbolic word), x' is enabled in q'_j, and the sub-run

$$q'_{k-1} \xrightarrow[(x',c')]{i'_k} \cdots \xrightarrow{i'_j} q'_j \xrightarrow{to[x']}$$

is x'-spanning with $c' = c$. Since the delays in the timed run ρ' are the same as in ρ and x, x' are both started at the same constant c along the j-th transition, it naturally follows that $(\kappa'_j - d)(x') = 0$.

Consider now the action i_{j+1}. We have two cases:

- If $i_{j+1} \in I$, the transition $(q'_j, \kappa'_j - d_{j+1}) \xrightarrow{i'_{j+1}} (q'_{j+1}, \kappa'_{j+1})$ with $i'_{j+1} = i_{j+1}$ is defined as $\mathcal{N}$ is complete by hypothesis.
- If $i_{j+1} = to[x]$ for some $x \in \chi_0^{\mathcal{M}}(q_j)$, we have by (1) that there exists $x' \in \chi_0^{\mathcal{N}}(q'_j)$ such that $(\kappa'_j - d_{j+1})(x') = 0$. As $\mathcal{N}$ is complete, we can thus take the transition

$$(q'_j, \kappa'_j - d_{j+1}) \xrightarrow{to[x']} (q'_{j+1}, \kappa'_{j+1}).$$
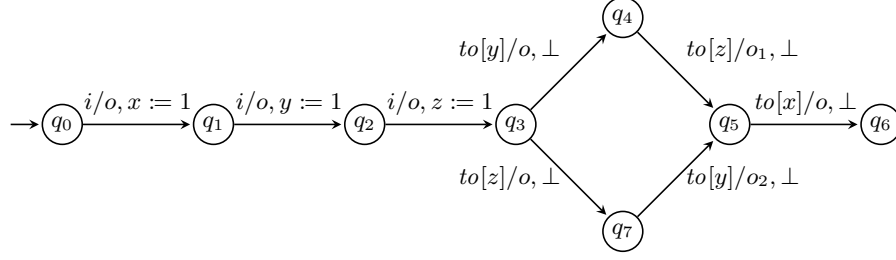
Fig. 5: An MMT with $\chi(q_0) = \chi(q_6) = \emptyset$, $\chi(q_1) = \chi(q_5) = \{x\}$, $\chi(q_2) = \chi(q_7) = \{x, y\}$, $\chi(q_3) = \{x, y, z\}$, $\chi(q_4) = \{x, z\}$. Every missing transition $q \xrightarrow[u]{i/\omega} p$ to obtain a complete MMT is such that $p = q_6, \omega = o$, and $u = \bot$.

By the previous arguments establishing (1), $\overline{i'_1 \cdots i'_{j+1}} = \mathbf{i}_1 \cdots \mathbf{i}_{j+1}$ follows. As $\mathcal{M} \overset{\text{sym}}{\approx} \mathcal{N}$, we get that the output o' of $q'_j \xrightarrow{i'_{j+1}/o'} q'_{j+1}$ is equal to o_{j+1}.

It remains to prove that the delay transition $(q'_{j+1}, \kappa'_{j+1}) \xrightarrow{d_{j+2}}$ is possible. Assume the contrary, i.e., there exists a timer $x' \in \chi^{\mathcal{N}}(q'_{j+1})$ such that $\kappa'_{j+1}(x') < d_{j+2}$. Let $d' = \kappa'_{j+1}(x')$. We thus have that $(\kappa'_{j+1} - d')(x') = 0$. By (1) applied to q_{j+1} and q'_{j+1}, there must exist a timer x such that $(\kappa_{j+1} - d')(x) = 0$, i.e., it is not possible to wait d_{j+2} units of time in (q_{j+1}, κ_{j+1}) and we have a contradiction.

We are thus able to extend the timed run ρ' which leads to the contradiction. We conclude that the symbolic equivalence implies the timed equivalence. □

A.3 Timed equivalence does not refine symbolic equivalence

Let $\mathcal{M}$ be the MMT of Figure 5. We make $\mathcal{M}$ complete by adding $q \xrightarrow[\bot]{i/o} q_6$ for every missing transition. Observe that $O = \{o, o_1, o_2\}$ and that $q_4 \xrightarrow{to[z]}$ outputs o_1 while $q_7 \xrightarrow{to[y]}$ outputs o_2. Moreover, let $\mathcal{N}$ be a copy of $\mathcal{M}$ such that o_1 and o_2 are swapped. Let us argue that $\mathcal{M} \overset{\text{time}}{\approx} \mathcal{N}$ but $\mathcal{M} \overset{\text{sym}}{\not\approx} \mathcal{N}$, starting with the latter. We write $q_j^{\mathcal{M}}$ and $q_j^{\mathcal{N}}$ to distinguish the states of $\mathcal{M}$ and $\mathcal{N}$.

Let $\mathbf{w} = i \cdot i \cdot i \cdot to[2] \cdot to[3]$ be a symbolic word, inducing the following runs:

$$q_0^{\mathcal{M}} \xrightarrow[(x,1)]{i/o} q_1^{\mathcal{M}} \xrightarrow[(y,1)]{i/o} q_2^{\mathcal{M}} \xrightarrow[(z,1)]{i/o} q_3^{\mathcal{M}} \xrightarrow[\bot]{to[y]/o} q_4^{\mathcal{M}} \xrightarrow[\bot]{to[z]/o_1} q_5^{\mathcal{M}}$$

$$q_0^{\mathcal{N}} \xrightarrow[(x,1)]{i/o} q_1^{\mathcal{N}} \xrightarrow[(y,1)]{i/o} q_2^{\mathcal{N}} \xrightarrow[(z,1)]{i/o} q_3^{\mathcal{N}} \xrightarrow[\bot]{to[y]/o} q_4^{\mathcal{N}} \xrightarrow[\bot]{to[z]/o_2} q_5^{\mathcal{N}}.$$

Hence, $\mathcal{M} \overset{\text{sym}}{\not\approx} \mathcal{N}$ as the last pair of transitions has different outputs.

So, it remains to show that $\mathcal{M} \overset{\text{time}}{\approx} \mathcal{N}$. Clearly, any tiw induces the same run in both $\mathcal{M}$ and $\mathcal{N}$ (up to the outputs), as they have exactly the same transitions. That is, any tiw w is such that $q_0^{\mathcal{M}} \xrightarrow{w} q_j^{\mathcal{M}}$ if and only if $q_0^{\mathcal{N}} \xrightarrow{w} q_j^{\mathcal{N}}$. Moreover,

outputs are the same in $\mathcal{M}$ and $\mathcal{N}$, except that o_1 and o_2 are swapped. So, let us focus on the transitions $q_4^{\mathcal{M}} \xrightarrow{to[z]/o_1}$ and $q_7^{\mathcal{M}} \xrightarrow{to[y]/o_2}$. It is not hard to see that the only way these transitions are triggered is to start all timers x, y, and z without any delay in between, to go through in either $(q_4^{\mathcal{M}}, x = 0, z = 0)$ or $(q_7^{\mathcal{M}}, x = 0, y = 0)$, and to trigger the $to[z]$ and $to[y]$ transitions, respectively, i.e., we must take the following two timed runs in $\mathcal{M}$:

$$(q_0^{\mathcal{M}}, \emptyset) \xrightarrow{0} (q_0^{\mathcal{M}}, \emptyset) \xrightarrow[(x,1)]{i/o} (q_1^{\mathcal{M}}, x = 1) \xrightarrow{0} (q_1^{\mathcal{M}}, x = 1) \xrightarrow[(y,1)]{i/o} (q_2^{\mathcal{M}}, x = 1, y = 1)$$

$$\xrightarrow{0} (q_2^{\mathcal{M}}, x = 1, y = 1) \xrightarrow[(z,1)]{i/o} (q_3^{\mathcal{M}}, x = 1, y = 1, z = 1)$$

$$\xrightarrow{1} (q_3^{\mathcal{M}}, x = 0, y = 0, z = 0) \xrightarrow[\perp]{to[y]/o} (q_4^{\mathcal{M}}, x = 0, z = 0)$$

$$\xrightarrow{0} (q_4^{\mathcal{M}}, x = 0, z = 0) \xrightarrow[\perp]{to[z]/o_1} (q_5^{\mathcal{M}}, x = 0) \xrightarrow{0} (q_5^{\mathcal{M}}, x = 0)$$

and

$$(q_0^{\mathcal{M}}, \emptyset) \xrightarrow{0} (q_0^{\mathcal{M}}, \emptyset) \xrightarrow[(x,1)]{i/o} (q_1^{\mathcal{M}}, x = 1) \xrightarrow{0} (q_1^{\mathcal{M}}, x = 1) \xrightarrow[(y,1)]{i/o} (q_2^{\mathcal{M}}, x = 1, y = 1)$$

$$\xrightarrow{0} (q_2^{\mathcal{M}}, x = 1, y = 1) \xrightarrow[(z,1)]{i/o} (q_3^{\mathcal{M}}, x = 1, y = 1, z = 1)$$

$$\xrightarrow{1} (q_3^{\mathcal{M}}, x = 0, y = 0, z = 0) \xrightarrow[\perp]{to[z]/o} (q_7^{\mathcal{M}}, x = 0, y = 0)$$

$$\xrightarrow{0} (q_7^{\mathcal{M}}, x = 0, y = 0) \xrightarrow[\perp]{to[y]/o_2} (q_5^{\mathcal{M}}, x = 0) \xrightarrow{0} (q_5^{\mathcal{M}}, x = 0).$$

We can then obtain the same runs in $\mathcal{N}$, up to a swap of o_1 and o_2:

$$(q_0^{\mathcal{N}}, \emptyset) \xrightarrow{0} (q_0^{\mathcal{N}}, \emptyset) \xrightarrow[(x,1)]{i/o} (q_1^{\mathcal{N}}, x = 1) \xrightarrow{0} (q_1^{\mathcal{N}}, x = 1) \xrightarrow[(y,1)]{i/o} (q_2^{\mathcal{N}}, x = 1, y = 1)$$

$$\xrightarrow{0} (q_2^{\mathcal{N}}, x = 1, y = 1) \xrightarrow[(z,1)]{i/o} (q_3^{\mathcal{N}}, x = 1, y = 1, z = 1)$$

$$\xrightarrow{1} (q_3^{\mathcal{N}}, x = 0, y = 0, z = 0) \xrightarrow[\perp]{to[y]/o} (q_4^{\mathcal{N}}, x = 0, z = 0)$$

$$\xrightarrow{0} (q_4^{\mathcal{N}}, x = 0, z = 0) \xrightarrow[\perp]{to[z]/o_2} (q_5^{\mathcal{N}}, x = 0) \xrightarrow{0} (q_5^{\mathcal{N}}, x = 0)$$

and

$$(q_0^{\mathcal{N}}, \emptyset) \xrightarrow{0} (q_0^{\mathcal{N}}, \emptyset) \xrightarrow[(x,1)]{i/o} (q_1^{\mathcal{N}}, x = 1) \xrightarrow{0} (q_1^{\mathcal{N}}, x = 1) \xrightarrow[(y,1)]{i/o} (q_2^{\mathcal{N}}, x = 1, y = 1)$$

$$\xrightarrow{0} (q_2^{\mathcal{N}}, x = 1, y = 1) \xrightarrow[(z,1)]{i/o} (q_3^{\mathcal{N}}, x = 1, y = 1, z = 1)$$

$$\xrightarrow{1} (q_3^{\mathcal{N}}, x = 0, y = 0, z = 0) \xrightarrow[\perp]{to[z]/o} (q_7^{\mathcal{N}}, x = 0, y = 0)$$

$$\xrightarrow{0} (q_7^{\mathcal{N}}, x = 0, y = 0) \xrightarrow[\perp]{to[y]/o_1} (q_5^{\mathcal{N}}, x = 0) \xrightarrow{0} (q_5^{\mathcal{N}}, x = 0).$$

Hence, any tiw w inducing the first run in $\mathcal{M}$ necessarily induces the second run, too (and the runs triggering $to[x]$). Moreover, w also induces the two runs

in $\mathcal{N}$ (and the runs triggering $to[x]$). We thus conclude that $toutputs^{\mathcal{M}}(w) = toutputs^{\mathcal{N}}(w)$ for every tiw w. That is, $\mathcal{M} \overset{\text{time}}{\approx} \mathcal{N}$ and $\mathcal{M} \overset{\text{sym}}{\napprox} \mathcal{N}$.

B   Proof of Lemma 1

Lemma 1. *For any complete MMT $\mathcal{M}$, there is an s-learnable MMT $\mathcal{N} \overset{sym}{\approx} \mathcal{M}$.*

In order to prove this lemma, we first properly adapt the notion of *zones* from timed automata [2, 8] to MMTs.[10] Given a complete MMT $\mathcal{M}$, we show that the MMT constructed from the reachable zones of $\mathcal{M}$ is s-learnable.

B.1   Zones

Let X be a set of timers. A *zone Z over X* is a set of valuations over X, i.e., $Z \subseteq \mathsf{Val}(X)$, described by the following grammar:

$$\phi = x < c \mid x \le c \mid c < x \mid c \le x \mid x - y < c \mid x - y \le c \mid \phi_1 \wedge \phi_2$$

with $x, y \in X$ and $c \in \mathbb{N}$. It may be that a zone is empty. For the particular case $X = \emptyset$, we have that $\mathsf{Val}(X) = \{\emptyset\}$, meaning that a zone Z is either the zone $\{\emptyset\}$ or the empty zone.

Given a zone Z over X, a set $Y \subseteq X$, a timer x (that does not necessarily belong to X), and a constant $c \in \mathbb{N}^{>0}$, we define the following operations:

– The *downward closure* of Z where we let some time elapsed in all valuations of Z. That is, we obtain all valuations that can be reached from Z by waiting (we take delays that do not exceed the smallest value to avoid going below zero):
$$Z{\downarrow} = \{\kappa - d \mid \kappa \in Z, d \le \min_{y \in \mathsf{dom}(\kappa)} \kappa(y)\}.$$

– The *restriction* of Z to $Y \ne \emptyset$ where, for every valuation of Z, we discard values associated with timers in $X \setminus Y$, i.e., we only keep the timers that are in Y:
$$Z{\lceil}Y = \{\kappa' \in \mathsf{Val}(Y) \mid \exists \kappa \in Z, \forall y \in Y : \kappa'(y) = \kappa(y)\}.$$

If Y is empty, then we define the restriction as:
$$Z{\lceil}\emptyset = \begin{cases} \emptyset & \text{if } Z = \emptyset \\ \{\emptyset\} & \text{otherwise.} \end{cases}$$

– The *assignment* of x to c in the zone Z over X. Either x is already in X in which case we simply overwrite the value of x by c, or x is not in X in which case we "extend" the valuations of Z by adding x:
$$Z[x = c] = \{\kappa' \in \mathsf{Val}(X \cup \{x\}) \mid \kappa'(x) = c \wedge$$
$$\exists \kappa \in Z, \forall y \in X \setminus \{x\} : \kappa'(y) = \kappa(y)\}.$$

---

[10] The concept of *region* from timed automata was adapted to automata with timers in [7].

– The *timeout of x* in Z where we keep valuations of Z s.t. $x \in X$ times out:

$$to[Z, x] = \{\kappa \in Z \mid \kappa(x) = 0\}.$$

If Z is a zone, then $Z{\downarrow}$, $Z{\lceil}Y$, $Z[x = c]$, and $to[Z, x]$ are again zones [8]. Observe that $Z{\downarrow}$ and $to[Z, x]$ are zones over X (when $x \in X$), $Z{\lceil}Y$ is a zone over Y, and $Z[x = c]$ is a zone over $X \cup \{x\}$.

B.2   Zone MMT

Given a complete MMT $\mathcal{M}$, we explain how to construct its *zone MMT* that we denote $zone(\mathcal{M})$. The states of $zone(\mathcal{M})$ are pairs (q, Z) where q is a state of $\mathcal{M}$ and Z is a zone included in $\mathsf{Val}(\chi(q))$. The idea to construct $zone(\mathcal{M})$ is to start from the pair $(q_0^{\mathcal{M}}, \{\emptyset\})$ and explore every outgoing transition of $q_0^{\mathcal{M}}$. In general, we want to define the outgoing transitions of the current pair (q, Z). To do so, we consider the outgoing transitions of q in the complete machine $\mathcal{M}$. For every $q \xrightarrow[u]{i} q'$ with $i \in I$, we reproduce the same transition in $zone(\mathcal{M})$ (as it is always possible to trigger an input transition). That is, we define $(q, Z) \xrightarrow[u]{i} (q', Z')$ with a zone Z' that depends on the update: if $u = \bot$, then Z' is obtained by the restriction of Z to the active timers of q'; if $u = (x, c)$, then we first assign x to c. In both cases, we also let time elapse, i.e., we always compute the downward closure. Finally, we perform the same idea with every $q \xrightarrow{to[x]}$ such that $x \in \chi(q)$, except that we first only consider the valuations of Z where x is zero. If $to[Z, x]$ is empty, we do not define the transition. Hence, in this way, we construct the states (q, Z) of $zone(\mathcal{M})$ such that $Z \neq \emptyset$ and that are reachable from its initial state $(q_0^{\mathcal{M}}, \{\emptyset\})$.

More formally, let $\mathcal{M} = (X^{\mathcal{M}}, Q^{\mathcal{M}}, q_0^{\mathcal{M}}, \chi^{\mathcal{M}}, \delta^{\mathcal{M}})$ be a complete MMT. We first define the tuple $\mathcal{Z} = (X^{\mathcal{Z}}, Q^{\mathcal{Z}}, q_0^{\mathcal{Z}}, \chi^{\mathcal{Z}}, \delta^{\mathcal{Z}})$ with:

– $X^{\mathcal{Z}} = X^{\mathcal{M}}$,
– $Q^{\mathcal{Z}} = \{(q, Z) \mid q \in Q^{\mathcal{M}}, Z \subseteq \mathsf{Val}(\chi^{\mathcal{M}}(q)) \wedge Z \neq \emptyset\}$,
– $q_0^{\mathcal{Z}} = (q_0^{\mathcal{M}}, \{\emptyset\})$,
– For any $(q, Z) \in Q^{\mathcal{Z}}$, we define $\chi^{\mathcal{Z}}((q, Z)) = \chi^{\mathcal{M}}(q)$, i.e., we simply copy the active timers of q,

– Let $(q, Z) \in Q^{\mathcal{Z}}$ and $q \xrightarrow[u]{i/o} q'$ be a transition of $\mathcal{M}$. We define

$$Z' = \begin{cases} Z & \text{if } u = \bot \text{ and } i \in I \\ Z[x = c] & \text{if } u = (x, c) \text{ and } i \in I \\ to[Z, x] & \text{if } u = \bot \text{ and } i = to[x] \\ (to[Z, x])[x = c] & \text{if } u = (x, c) \text{ and } i = to[x]. \end{cases}$$

Then, if $Z' \neq \emptyset$, we restrict Z' to $\chi^{\mathcal{M}}(q')$ and let time elapse. That is, we define

$$\delta^{\mathcal{Z}}((q, Z), i) = \left(\left(q', \left(Z'{\lceil}\chi^{\mathcal{M}}(q') \right){\downarrow} \right), o, u \right).$$

The MMT $zone(\mathcal{M})$ is then the MMT $\mathcal{Z}$ restricted to its reachable states. Observe that the set of actions of $zone(\mathcal{M})$ is the set of actions of $\mathcal{M}$, i.e., $A(zone(\mathcal{M})) = A(\mathcal{M})$.

Let us now argue that $zone(\mathcal{M})$ has finitely many states and it is well-formed. We will prove later that it is also complete.

Lemma 5. *Let $\mathcal{M}$ be a complete MMT. Then, $zone(\mathcal{M})$ has finitely many states and is well-formed.*

Proof. The zone MMT is clearly well-formed because its transitions mimics the transitions of $\mathcal{M}$ and for any $(q, Z) \in Q^{\mathcal{Z}}$, we have $\chi^{\mathcal{Z}}((q, Z)) = \chi^{\mathcal{M}}(q)$.

By construction, the states (q, Z) of $zone(\mathcal{M})$ are such that Z is a zone over $\chi^{\mathcal{M}}(q)$, described as a finite conjunction of constraints of the shape $x \bowtie c$ or $x - y \bowtie c$, with $\bowtie \in \{<, \leq, \geq, >\}$ and $c \in \mathbb{N}$. For each timer x, let c_x be the maximal constant appearing on an update (re)starting x. Since the value of a timer can only decrease, it is clear that we will never reach a zone where $x > c_x$. Moreover, as the value of a timer must remain at least zero at any time, we also have a lower bound. In other words, we know that each timer x will always be confined between zero and c_x. From the shape of the constraints and these bounds, it follows immediately that there are finitely many zones. Hence, $zone(\mathcal{M})$ has finitely many states. $\qquad\square$

We now move towards proving the required properties to show Lemma 1 where the announced MMT $\mathcal{N}$ is the zone MMT of $\mathcal{M}$:

- Both MMTs $\mathcal{M}$ and $zone(\mathcal{M})$ have the same timed runs. That is, for any state $q \in Q^{\mathcal{M}}$, it holds that $(q_0^{\mathcal{M}}, \emptyset) \xrightarrow{w} (q, \kappa)$ if and only if $((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset) \xrightarrow{w} ((q, Z), \kappa)$ for some zone Z. See Proposition 1.
- $zone(\mathcal{M})$ is complete, by Corollary 1.
- $\mathcal{M}$ and $zone(\mathcal{M})$ have the same feasible runs, by Corollary 2.
- $\mathcal{M}$ and $zone(\mathcal{M})$ are symbolically equivalent, by Proposition 2.
- Any run of $zone(\mathcal{M})$ is feasible, by Proposition 3.

Proposition 1. *Let $zone(\mathcal{M})$ be the zone MMT of some complete MMT $\mathcal{M}$. Then, for every state $q \in Q^{\mathcal{M}}$, valuation $\kappa \in \mathsf{Val}(\chi^{\mathcal{M}}(q))$, and timed word w,*

$$(q_0^{\mathcal{M}}, \emptyset) \xrightarrow{w} (q, \kappa) \text{ in } \mathcal{M} \quad \Leftrightarrow \quad ((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset) \xrightarrow{w} ((q, Z), \kappa) \text{ in } zone(\mathcal{M})$$

for some zone Z over $\chi^{\mathcal{M}}(q)$ such that $\kappa \in Z$.

Proof. We focus on the $\Rightarrow$ direction. The other direction can be obtained with similar arguments. Let $q \in Q^{\mathcal{M}}, \kappa \in \mathsf{Val}(\chi^{\mathcal{M}}(q))$, and w be a timed word such that $(q_0^{\mathcal{M}}, \emptyset) \xrightarrow{w} (q, \kappa)$. We show that there exists a zone Z over $\chi^{\mathcal{M}}(q)$ such that $\kappa \in Z$ and $((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset) \xrightarrow{w} ((q, Z), \kappa)$. We proceed by induction over the length of w.

Base case: $|w| = 0$, i.e., $w = d$ with $d \in \mathbb{R}^{\geq 0}$. Since no timer is active, it is clear that we have the runs

$$(q_0^{\mathcal{M}}, \emptyset) \xrightarrow{d} (q_0^{\mathcal{M}}, \emptyset) \qquad \text{and} \qquad ((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset) \xrightarrow{d} ((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset),$$

and $\emptyset \in \{\emptyset\}$.

Induction step: let $k \in \mathbb{N}$ and assume the implication is true for every timed word of length k. Let $w = w' \cdot i \cdot d$ of length $k+1$, i.e., $|w'| = k$, $i \in A(\mathcal{M})$, and $d \in \mathbb{R}^{\geq 0}$. Then, we have

$$(q_0^{\mathcal{M}}, \emptyset) \xrightarrow{w'} (p, \lambda) \xrightarrow[u]{i} (q, \kappa) \xrightarrow{d} (q, \kappa - d).$$

This implies that $d \leq \min_{y \in \chi^{\mathcal{M}}(q)} \kappa(y)$.[11] By induction hypothesis, we have that

$$((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset) \xrightarrow{w'} ((p, Z_p), \lambda)$$

such that $\lambda \in Z_p$. It is then sufficient to show that we have

$$((p, Z_p), \lambda) \xrightarrow[u]{i} ((q, Z), \kappa) \xrightarrow{d} ((q, Z), \kappa - d)$$

with $\kappa - d \in Z$.

By construction of the zone MMT and as $p \xrightarrow{i} q$ is defined in $\mathcal{M}$, the i-transition from (p, Z_p) to (q, Z) is defined if and only if Z is not empty and

$$Z = \begin{cases} (Z_p \lceil \chi^{\mathcal{M}}(q)) \downarrow & \text{if } i \in I \text{ and } u = \bot \\ ((Z_p[x = c]) \lceil \chi^{\mathcal{M}}(q)) \downarrow & \text{if } i \in I \text{ and } u = (x, c) \\ ((to[Z_p, x]) \lceil \chi^{\mathcal{M}}(q)) \downarrow & \text{if } i = to[x] \text{ and } u = \bot \\ (((to[Z_p, x])[x = c]) \lceil \chi^{\mathcal{M}}(q)) \downarrow & \text{if } i = to[x] \text{ and } u = (x, c). \end{cases}$$

Since Z_p is not empty (as $\lambda \in Z_p$) and $(p, \lambda) \xrightarrow{i}$ can be triggered (meaning that $\lambda(x) = 0$ if $i = to[x]$), we have that Z is also not empty. Hence, $((p, Z_p), \lambda) \xrightarrow{i} ((q, Z), \kappa)$ is well-defined and can be triggered.

Let us show that $\kappa \in Z$. By definition of a timed run, $\kappa \in \mathsf{Val}(\chi^{\mathcal{M}}(q))$. Moreover, Z is a zone over $\chi^{\mathcal{M}}(q)$. We know that $\lambda \in Z_p$ and κ is constructed from λ by discarding the values for timers that are stopped by the discrete transition and, maybe, (re)starting a timer. Since Z is constructed using the same operations, it follows that $\kappa \in Z$.

Finally, we process the delay d. We already know that $d \leq \min_{y \in \chi^{\mathcal{M}}(q)} \kappa(y)$. Hence, it is feasible to wait d units of time from $((q, Z), \kappa)$. Moreover, as Z is already its downward closure, we still have that $\kappa - d \in Z$.

We thus obtain the implication. Again, one can show the other direction using similar arguments, by definition of $zone(\mathcal{M})$. □

From the (proof of the) previous proposition, we can easily obtain that $zone(\mathcal{M})$ is complete, if $\mathcal{M}$ is complete, Indeed, the construction of $zone(\mathcal{M})$ immediately copies input-transitions, while timeout-transitions are reproduced when the timer is enabled in the state, meaning that $zone(\mathcal{M})$ is complete.

---

[11] We recall that $\min_{y \in \chi^{\mathcal{M}}(q)} \kappa(y) = +\infty$ when $\chi^{\mathcal{M}}(q) = \emptyset$.

Corollary 1. *Let $\mathcal{M}$ be a complete MMT and $zone(\mathcal{M})$ be its zone MMT. Then, $zone(\mathcal{M})$ is complete.*

From the previous proposition, we also conclude that any feasible run of $\mathcal{M}$ can be reproduced in $zone(\mathcal{M})$ and vice-versa. Recall that $A(\mathcal{M}) = A(zone(\mathcal{M}))$.

Corollary 2. *Let $\mathcal{M}$ be a complete MMT and $zone(\mathcal{M})$ be its zone MMT. Then, for all words $w \in A(\mathcal{M})^*$*

$$q_0^{\mathcal{M}} \xrightarrow{w} q \text{ is in } runs(\mathcal{M}) \text{ and is feasible}$$

$$\Leftrightarrow (q_0^{\mathcal{M}}, \{\emptyset\}) \xrightarrow{w} (q, Z) \text{ is in } runs(zone(\mathcal{M})) \text{ and is feasible for some zone } Z.$$

Proof. Let $q_0^{\mathcal{M}} \xrightarrow{w} q$ be a feasible run of $\mathcal{M}$. Then, there exists a timed run $(q_0^{\mathcal{M}}, \emptyset) \xrightarrow{v} (q, \kappa)$ of $\mathcal{M}$. By Proposition 1, it follows that $((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset) \xrightarrow{v} ((q, Z), \kappa)$ is a timed run of $zone(\mathcal{M})$ for some zone Z such that $\kappa \in Z$. As v and w use the same actions, the run $(q_0^{\mathcal{M}}, \{\emptyset\}) \xrightarrow{w} (q, Z)$ is a feasible run of $zone(\mathcal{M})$. The other direction holds with similar arguments. $\square$

Let us now move towards proving that $\mathcal{M} \overset{\text{sym}}{\approx} zone(\mathcal{M})$.

Proposition 2. *Let $\mathcal{M}$ be a complete MMT. Then, $\mathcal{M} \overset{sym}{\approx} zone(\mathcal{M})$.*

Proof. We have to show that for every symbolic word $\mathtt{i_1} \cdots \mathtt{i_n}$ over $I \cup TO[\mathbb{N}^{>0}]$:

- $q_0^{\mathcal{M}} \xrightarrow[u_1]{\mathtt{i_1}/o_1} q_1 \cdots \xrightarrow[u_n]{\mathtt{i_n}/o_n} q_n$ is a feasible run in $\mathcal{M}$ if and only if $q_0^{zone(\mathcal{M})} \xrightarrow[u'_1]{\mathtt{i_1}/o'_1}$ $q'_1 \cdots \xrightarrow[u'_n]{\mathtt{i_n}/o'_n} q'_n$ is a feasible run in $zone(\mathcal{M})$.
- Moreover,
    - $o_j = o'_j$ for all $j \in \{1, \ldots, n\}$, and
    - $q_j \xrightarrow{\mathtt{i_j}} \cdots \xrightarrow{\mathtt{i_k}} q_k$ is spanning $\Rightarrow u_j = (x, c) \wedge u'_j = (x', c') \wedge c = c'$.

Let $\mathtt{w} = \mathtt{i_1} \cdots \mathtt{i_n}$ be a symbolic word such that $q_0^{\mathcal{M}} \xrightarrow{\mathtt{w}} q_n$ is a feasible run of $\mathcal{M}$. Hence, there exists $w = i_1 \cdots i_n$ such that $\overline{w} = \mathtt{w}$ and $q_0^{\mathcal{M}} \xrightarrow[u_1]{i_1/o_1} q_1 \cdots \xrightarrow[u_n]{i_n/o_n} q_n$ is a feasible run of $\mathcal{M}$. By Corollary 2, it follows that $(q_0^{\mathcal{M}}, \{\emptyset\}) \xrightarrow[u'_1]{i_1/o'_1}$ $(q_1, Z_1) \cdots \xrightarrow[u'_n]{i_n/o'_n} (q_n, Z_n)$ is a feasible run of $zone(\mathcal{M})$. By construction of $zone(\mathcal{M})$, we immediately have that $o_j = o'_j$ and $u_j = u'_j$ for every j. Therefore, $\overline{i'_1 \cdots i'_n} = \mathtt{w}$ and $(q_0^{\mathcal{M}}, \{\emptyset\}) \xrightarrow{\mathtt{w}} (q_n, Z_n)$ is a feasible run of $\mathcal{M}$. Hence, the direction from $\mathcal{M}$ to $zone(\mathcal{M})$ holds. The other direction follows with the same arguments. We thus conclude that $\mathcal{M} \overset{\text{sym}}{\approx} zone(\mathcal{M})$. $\square$

Finally, we show that any run of $zone(\mathcal{M})$ is feasible.

Proposition 3. *Let $\mathcal{M}$ be a complete MMT. Then, any run of $zone(\mathcal{M})$ is feasible.*

Proof. As all states of $zone(\mathcal{M})$ are reachable, we can restrict the proof to runs starting at the initial state of $zone(\mathcal{M})$. To get Proposition 3, let us prove that for any run $\pi = (q_0^{\mathcal{M}}, \{\emptyset\}) \xrightarrow{w} (q, Z)$ of $zone(\mathcal{M})$, for any $\kappa \in Z$, there exists a timed run

$$\rho = ((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset) \xrightarrow{v} ((q, Z), \kappa)$$

such that $untime(\rho) = \pi$. We prove this property by induction over $n = |w|$.

Base case: $n = 0$, i.e., $w = \varepsilon$. Let $\pi = (q_0^{\mathcal{M}}, \{\emptyset\}) \xrightarrow{\varepsilon} (q_0^{\mathcal{M}}, \{\emptyset\})$. It is clear that there exists $\rho = ((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset) \xrightarrow{d} ((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset)$ and $\emptyset \in \{\emptyset\}$ for any $d \in \mathbb{R}^{\geq 0}$. And we have $untime(\rho) = \pi$.

Induction step: Let $k \in \mathbb{N}$ and assume the proposition holds for every word of length k. Let w of length $k + 1$, i.e., we can decompose $w = w' \cdot i$ with $i \in A(\mathcal{M})$ and $|w'| = k$. We show that, if $\pi = (q_0^{\mathcal{M}}, \{\emptyset\}) \xrightarrow{w} (q, Z)$ is a run and κ is a valuation in Z, there exists a timed run $\rho = ((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset) \xrightarrow{v} ((q, Z), \kappa)$ such that $untime(\rho) = \pi$.

Assume that π is a run of $zone(\mathcal{M})$ and let $\pi' = (q_0^{\mathcal{M}}, \{\emptyset\}) \xrightarrow{w'} (p, Z_p)$. Observe that π' is a sub-run of π. By the induction hypothesis, we know that for any $\lambda \in Z_p$, there exists a timed run $\rho' = ((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset) \xrightarrow{v'} ((p, Z_p), \lambda)$ such that $untime(\rho') = \pi'$. Hence, let us focus on the last transition $(p, Z_p) \xrightarrow{i} (q, Z)$ of π. By construction of $zone(\mathcal{M})$, given (q, Z) and $\kappa \in Z$, we deduce that there exists $d \in \mathbb{R}^{\geq 0}$ and $\lambda \in Z_p$ such that

$$((p, Z_p), \lambda) \xrightarrow{i} ((q, Z), \kappa + d) \xrightarrow{d} ((q, Z), \kappa).$$

Thus, by the induction hypothesis with this λ, it follows that we have the timed run

$$\rho = ((q_0^{\mathcal{M}}, \{\emptyset\}), \emptyset) \xrightarrow{v'} ((p, Z_p), \lambda) \xrightarrow{i} ((q, Z), \kappa + d) \xrightarrow{d} ((q, Z), \kappa).$$

such that $untime(\rho) = \pi$. $\qquad\square$

B.3   Proof of Lemma 1

We are now ready to prove Lemma 1 which we repeat one more time.

Lemma 1. *For any complete MMT $\mathcal{M}$, there is an s-learnable MMT $\mathcal{N} \overset{sym}{\approx} \mathcal{M}$.*

Proof. Let $\mathcal{M}$ be a complete MMT and $zone(\mathcal{M})$ be its zone MMT. By Corollary 1 and Propositions 2 and 3, $zone(\mathcal{M})$ is complete, any run of $zone(\mathcal{M})$ is feasible, and $\mathcal{M} \overset{sym}{\approx} zone(\mathcal{M})$. Hence, $zone(\mathcal{M})$ is s-learnable and satisfies the lemma. $\qquad\square$

C   Proof of Lemma 2

Lemma 2. *For race-avoiding MMTs, the three symbolic queries can be implemented via a polynomial number of concrete output and equivalence queries.*

In order to prove this lemma, we first properly define what is a *race-avoiding* MMT. We then explain how to construct a tiw ending in a state q of the MMT in Appendix C.2 Finally, in Appendix C.4, we define the concrete output and equivalence queries, and show how to use the tiws to obtain the lemma.

C.1   Race-avoiding MMT

Let $\mathcal{M}$ be an MMT. We say that $\mathcal{M}$ is *race-avoiding* [7] if every feasible run π is the untimed projection of a timed run in which all delays are non-zero and at most one timer times out in any configuration of the timed run. This implies that there always exists a tiw such that there exists a unique timed run in $\mathcal{M}$ (as we never have any choice to make while reading the tiw) and the untimed projection of the run is π. Formally, any feasible run $\pi = p_0 \xrightarrow{i_1} p_1 \xrightarrow{i_2} \cdots \xrightarrow{i_n} p_n$ with $p_0 = q_0$ is the untimed projection of a timed run $\rho = (p_0, \emptyset) \xrightarrow{d_1} (p_0, \emptyset) \xrightarrow{i_1} (p_1, \kappa_1) \xrightarrow{d_2} \cdots \xrightarrow{i_n} (p_n, \kappa_n) \xrightarrow{d_{n+1}} (p_n, \kappa_n - d_{n+1})$ such that:

- all delays are non-zero: $d_j > 0$ for any $j \in \{1, \ldots, n+1\}$,
- a timer times out precisely when we want to process its timeout: in any $(\kappa_j - d_{j+1})$ and $x \in \chi(p_j)$ with $j \in \{1, \ldots, n-1\}$, we have $(\kappa_j - d_{j+1})(x) = 0$ if and only if $i_{j+1} = to[x]$, and
- no timer times out in $\kappa_n - d_{n+1}$: $(\kappa_n - d_{n+1})(x) \neq 0$ for all $x \in \chi(p_n)$.

The notion of race-avoiding[12] machine is introduced in [7] with a 3EXP algorithm to decide whether a machine is race-avoiding.

The next lemma holds by taking a timed run satisfying the above constraints. The next section is devoted to constructing such a timed run.

Lemma 6. *Let $\mathcal{M}$ be a race-avoiding MMT in which every run is feasible. Then, for any run π of $\mathcal{M}$, there exists a tiw w such that*

- *there exists a unique timed run $\rho = (q_0, \emptyset) \xrightarrow{w}$, and*
- *untime$(\rho) = \pi$.*

C.2   Construction of a timed run reaching a state

Assume that $\mathcal{M}$ is a race-avoiding MMT such that every run of $\mathcal{M}$ is feasible. Let

$$\pi = p_0 \xrightarrow[u_1]{i_1/o_1} p_1 \xrightarrow[u_2]{i_2/o_2} \cdots \xrightarrow[u_n]{i_n/o_n} p_n \in runs(\mathcal{M})$$

with $p_0 = q_0$ (i.e., we start from the initial state). We explain how to construct a tiw w that satisfies Lemma 6, i.e., such that $(q_0, \emptyset) \xrightarrow{w}$ is the unique timed run reading w and whose untimed projection is π. We do this in two steps: we first construct a timed word over $A(\mathcal{M})$ and then transform it to remove the timeout symbols.

---

[12] Whenever we have a zero-delay between two actions of a timed run, we say that we have a *race*.

Since π is feasible, there exists a timed run

$$\rho = (p_0, \emptyset) \xrightarrow{d_1} (p_0, \emptyset) \xrightarrow[u_1]{i_1/o_1} (p_1, \kappa_1) \xrightarrow{d_2} \cdots \xrightarrow[u_n]{i_n/o_n} (p_n, \kappa_n) \xrightarrow{d_{n+1}} (p_n, \kappa_n - d_{n+1})$$

such that $untime(\rho) = \pi$. Moreover, as $\mathcal{M}$ is race-avoiding, we can assume that

- $d_j > 0$ for any $j \in \{1, \ldots, n+1\}$,
- $(\kappa_j - d_{j+1})(x) = 0$ if and only if $i_{j+1} = to[x]$ for all $j \in \{1, \ldots, n-1\}$ and some $x \in \chi(p_j)$, and
- $(\kappa_n - d_{n+1})(x) \neq 0$ for all $x \in \chi(p_n)$.

Let $w = d_1 i_1 \cdots d_n i_n d_{n+1}$ be the timed word over $A(\mathcal{M})$ composed of the delays and actions seen along ρ. Recall that any timeout can only occur if the timer was started on a previous transition and enough time elapsed, e.g., if $u_j = (x, c)$, $i_k = to[x]$, and x is not restarted between i_{j+1} and i_k (i.e., the run from is p_{j-1} to p_k is x-spanning), then the sum of the delays d_{j+1} to d_k must be equal to c. In a similar manner, if $u_j = (x, c)$ but there are no k such that $i_k = to[x]$, then either x was restarted, stopped or the run ended before x could reach zero. Hence, w must satisfy the following set of constraints:

- For all $j \in \{1, \ldots, n+1\}$, $d_j \in \mathbb{R}^{>0}$.
- For any j and k such that $p_{j-1} \xrightarrow[(x,c)]{i_j} p_j \xrightarrow{i_{j+1}} \cdots \xrightarrow{i_k = to[x]} p_k$ is an x-spanning run, the sum of the delays d_{j+1} to d_k must be equal to c, i.e., $\sum_{\ell=j+1}^{k} d_\ell = c$.
- For any j such that $u_j = (x, c)$ and there is no $k > j$ such that $i_k = to[x]$, then either x is restarted or stopped by some transition, or the last action i_n is read before c units of time elapse.
  - In the first case, let $k > j$ such that $i_k \neq to[x]$ and $p_{k-1} \xrightarrow{i_k}$ restarts or stops x. Then, the sum of the delays d_{j+1} to d_k must be strictly less than c, i.e., $\sum_{\ell=j+1}^{k} d_\ell < c$.
  - In the second case (so, $x \in \chi(p_n)$ and x does not time out after waiting d_{n+1}), the sum of the delays d_{j+1} to d_{n+1} must be strictly less than c, i.e., $\sum_{\ell=j+1}^{n+1} d_\ell < c$.

Observe that these constraints are all linear. Moreover, if we consider the delays d_j as *variables*, one can still gather the constraints and use them to find a value for each d_j. We denote by $\mathrm{cnstr}(\pi)$ the set of constraints for π over the variables representing the delays. Notice that there may be multiple different solutions. Importantly, from the arguments given above, a solution always exists.

Lemma 7. *Let $\mathcal{M}$ be a race-avoiding MMT such that every run of $\mathcal{M}$ is feasible, and π be a run of $\mathcal{M}$ starting from the initial state of $\mathcal{M}$. Then, $\mathrm{cnstr}(\pi)$ has a solution.*

It remains to explain how to construct a tiw w' from the timed word $w = d_1 i_1 \ldots d_n i_n d_{n+1}$ over $A(\mathcal{M})$, i.e., how to drop the timeouts while still inducing the same timed run. If $i_j = to[x]$ for some timer x, we remove i_j and replace

the delay d_j by $d_j + d_{j+1}$. We repeat this until all timeouts are removed from w. Observe that w' contains at most as many symbols as w and the sum of the delays of w' is equal to the sum of the delays of w. To simplify the rest of this section, we assume from now on that $\mathrm{cnstr}(\pi)$ provides a tiw satisfying the constraints. Furthermore, for a state q, we write $\mathrm{cnstr}(q)$ to denote a tiw returned by $\mathrm{cnstr}(\pi)$ with π a feasible run from q_0 to q (if one exists).

C.3   Now, with partial knowledge

The above construction explains how to construct such a run when $\mathcal{M}$ is known. However, during the learning process, $\mathcal{M}$ is unknown. In this section, we explain how to construct a timed run from the data structure introduced in Section 3.1, i.e., from an observation tree $\mathcal{T}$. In other words, the constructed tiws must come from $\mathrm{cnstr}^{\mathcal{T}}(q)$ with $q \in Q^{\mathcal{T}}$. Observe that $\mathcal{T}$ is race-avoiding when $\mathcal{M}$ is race-avoiding. Moreover, due to the partial knowledge stored in $\mathcal{T}$, we may have

$$q \xrightarrow[\perp]{i} \in runs(\mathcal{T}) \qquad \text{and} \qquad f(q) \xrightarrow[(x,c)]{g(i)} \in runs(\mathcal{M}).$$

Hence, the constructed tiw from $\mathcal{T}$ may still induce multiple runs in $\mathcal{M}$. By relying on the race-avoiding aspect of both $\mathcal{T}$ and $\mathcal{M}$, we still have a way to force determinism. In short, we can change the delays to ensure that the fractional part of the sum of the delays up to an *input* is unique (see the notion of *block wiggling* in [7, Sec. 4]).[13] Then, the sum of the delays up to any timeout must have a fractional part that is equal to the fractional part of the delays up to the input that started the timer for the first time. In other words, all $to[x]$-transitions that are induced by an input starting x share the same fractional part.

Formally, let $w = d_1 i_1 \cdots i_n d_{n+1}$ be a tiw that is constructed by $\mathrm{cnstr}^{\mathcal{T}}(q)$, and $\rho \in \circledcirc runs^{\mathcal{M}}(w)$ be a timed run of $\mathcal{M}$ reading w, i.e.,

$$\rho = (q_0, \kappa_0) \xrightarrow{d'_1} (q_0, \kappa_0 - d'_1) \xrightarrow{i'_1} \cdots \xrightarrow{i'_m} (q_m, \kappa_m) \xrightarrow{d'_{m+1}} (q_m, \kappa_m - d'_{m+1})$$

with $q_0 = q_0^{\mathcal{M}}, \kappa_0 = \emptyset$ such that timeouts are inserted when needed and delays d_i are split accordingly. Moreover, let us denote by D_j the sum of all delays from d'_1 to d'_j, and by $\mathrm{frac}(c)$ the fractional part $c - \lfloor c \rfloor$ of $c \in \mathbb{R}^{\geq 0}$. Notice that if $q_{j-1} \xrightarrow[(x,c)]{i'_j} \cdots \xrightarrow{i'_k = to[x]} q_k$ with $i'_j \in I$ is x-spanning, then $\mathrm{frac}(D_j) = \mathrm{frac}(D_k)$ (as $c \in \mathbb{N}^{>0}$ units of time must have elapsed between the two actions i'_j and i'_k). Moreover, if $q_{k-1} \xrightarrow[(x,c')]{to[x]} \cdots \xrightarrow{i'_\ell = to[x]} q_\ell$ is again x-spanning, then $\mathrm{frac}(D_j) = \mathrm{frac}(D_k) = \mathrm{frac}(D_\ell)$. So, if we carefully select the delays such that every *input* i'_j induces a unique fractional part for D_j, we ensure that two actions will never happen with a zero-delay between then. Indeed, if $i_\alpha = to[x]$ and $i_\beta = to[y]$

---

[13] In practice, rational numbers are sufficient for these fractional parts and, thus, can be perfectly encoded in a computer.

with $x \neq y$, then $\mathrm{frac}(D_\alpha) \neq \mathrm{frac}(D_\beta)$ and some time must elapse between the two actions. Hence, w can be constructed from $\mathcal{T}$ such that $|toutputs^{\mathcal{M}}(w)| = 1$. From now on, we assume that $\mathrm{cnstr}^{\mathcal{T}}(q)$ provides such a word.

C.4   Concrete queries and proof of Lemma 2

Let us now introduce the *concrete* queries that work over the timed semantics of the MMT model. They are a direction adaptation of the queries used for Mealy machines [28,32]: one requests the output of a timed word, and one asks whether a hypothesis is correct.

Definition 6 (Concrete Queries). *Let $\mathcal{M}$ be the s-learnable and race-avoiding MMT of the teacher. The* concrete queries *the learner can use are:*

- **OQ**(w) *with w a tiw such that $toutputs^{\mathcal{M}}(w) \neq \emptyset$: the teacher outputs a tow in $toutputs^{\mathcal{M}}(w)$.*
- **EQ**$(\mathcal{H})$ *with $\mathcal{H}$ a complete MMT $\mathcal{H}$: the teacher replies* **yes** *if $\mathcal{M} \overset{time}{\approx} \mathcal{H}$, or a tiw w such that $toutputs^{\mathcal{M}}(w) \neq toutputs^{\mathcal{H}}(w)$.*

To ease the explanation, let us assume that the returned counterexample w is such that $|toutputs^{\mathcal{M}}(w)| = 1$, as $\mathcal{M}$ is race-avoiding. We now describe how to obtain each symbolic query from these concrete queries, i.e., we prove Lemma 2.

Symbolic output query. Let us start with symbolic output queries. We recall the definition. For a sw (symbolic word) $\mathtt{w}$ such that $\pi = q_0^{\mathcal{M}} \overset{\mathtt{w}}{\to} \ \in runs(\mathcal{M})$, **OQ**$^{\mathbf{s}}(\mathtt{w})$ returns the sequence of outputs seen along the run π.

So, let $\mathtt{w}$ be a sw for which we want to ask **OQ**$^{\mathbf{s}}(\mathtt{w})$. During the learning process, such a query is always used to define a new transition reading $i \in A(\mathcal{T})$ from a state q that is already present in $\mathcal{T}$. Hence, we focus on this case. We can assume that $q_0^{\mathcal{M}} \overset{\mathtt{w}}{\to} \ \in runs(\mathcal{M})$.[14] Let

$$\pi^{\mathcal{T}} = p_0 \overset{i_1}{\longrightarrow} p_1 \overset{i_2}{\longrightarrow} \cdots \overset{i_n}{\longrightarrow} p_n \in runs(\mathcal{T})$$

such that $p_0 = q_0^{\mathcal{T}}$, $p_n = q$, and $\overline{i_1 \cdots i_n \cdot i} = \mathtt{w}$. That is, we retrieve the unique run going from $q_0^{\mathcal{T}}$ to q, convert the actions into a symbolic word, alongside the action i (whose transition is not necessarily already in the tree). Since $q_0^{\mathcal{M}} \overset{\mathtt{w}}{\to} \ \in runs(\mathcal{M})$, $\overline{i_1 \cdots i_n \cdot i}$ is well-defined (the last symbol is either an input, or $to[j]$ for some appropriate $j \in \{1, \ldots, n\}$). We now construct a tiw that corresponds to the sw $\mathtt{w}$.

Let $v = \mathrm{cnstr}^{\mathcal{T}}(p_n)$. We can then ask **OQ**$(v \cdot d \cdot i \cdot 0)$ such that the sum of the last delay of v plus $d > 0$ leads to a very small delay (sufficiently small to ensure that no timer can time out just before i). So, we have the unique timed run

$$\rho^{\mathcal{T}} = (q_0^{\mathcal{T}}, \emptyset) \overset{d_1}{\longrightarrow} (q_0^{\mathcal{T}}, \emptyset) \overset{i_1}{\longrightarrow} (p_1, \kappa_1) \overset{d_2}{\longrightarrow} \cdots \overset{i_n}{\longrightarrow} (p_n, \kappa_n)$$

---

[14] We explain in Section 3.1 how to ensure this.

$$\xrightarrow{d_{n+1}+d} (p_n, \kappa_n - (d_{n+1} + d))$$

of $\mathcal{T}$ reading v. Moreover, let

$$\rho^{\mathcal{M}} = (q_0^{\mathcal{M}}, \emptyset) \xrightarrow{d_1'} (q_0^{\mathcal{M}}, \emptyset) \xrightarrow{i_1'} (p_1', \kappa_1') \xrightarrow{d_2'} \cdots \xrightarrow{i_m'} (p_m', \kappa_m')$$

$$\xrightarrow{d_{m+1}'} (p_m', \kappa_m' - d_{m+1}')$$

be the unique timed run of $\mathcal{M}$ reading $v \cdot d \cdot i \cdot 0$ (see Appendix C.2 for the uniqueness of the run). Since $\mathcal{T}$ only holds partial knowledge about $\mathcal{M}$, it is possible that the constraints to build v are not enough, in the sense that we wait for too long in some configuration of the run in $\mathcal{M}$. Then, $\mathcal{M}$ has to process a timeout-transition that is *unexpected*, i.e., there is no timeout-transition at that specific moment in the run of $\mathcal{T}$. (An input-transition can never be unexpected). Let $i_k' = to[x']$ be the first unexpected timeout-transition. Since any $to[x']$-transition implies that x' was started before, there must exist an index $j \in \{1, \ldots, k-1\}$ such that

$$(p_{j-1}', \kappa_{j-1}') \xrightarrow{d_j'} (p_{j-1}', \kappa_{j-1}' - d_j') \xrightarrow[(x',c)]{i_j'} \cdots \xrightarrow{i_k' = to[x']}$$

is x'-spanning. Moreover, as k is the first such unexpected timeout, it must be that $d_\ell' = d_\ell$ for all $\ell \in \{1, \ldots, k-1\}$. Finally, by construction of v, we deduce that we have

$$(p_{j-1}, \kappa_{j-1}) \xrightarrow{d_j} (p_{j-1}, \kappa_{j-1} - d_j) \xrightarrow[\perp]{i_j}$$

in $\mathcal{T}$. Indeed, otherwise, we would have an update (x, c) (with the same c by the fact that $\mathcal{T}$ is an observation tree for $\mathcal{M}$), i.e., i_k' would not be unexpected, as both timed runs use the same delays up to that point.

Then, it means that we discovered a new enabled timer in p_{k-1}. Moreover, we know, by the fact that each timeout is associated with a unique fractional part, that it must be the transition reading i_j that (re)starts a timer. Let y be x_{p_j} if $i_j \in I$, and be x if $i_j = to[x]$. Then, we can create the transition $p_{k-1} \xrightarrow{to[y]}$ and change the update of the transition reading i_j to replace $\perp$ by an update (y, c) (the constant can be inferred from the delays in $\rho^{\mathcal{T}}$).

Hence, if an unexpected timeout occurs, we can add more information to the tree and the constraints of $\mathrm{cnstr}^{\mathcal{T}}(p_n)$ are refined. After expanding the tree, we create a new $v = \mathrm{cnstr}^{\mathcal{T}}(p_n)$ until there is no unexpected timeout. Observe there can only be at most n unexpected timeouts. That is:

Proposition 4. *We need at most n concrete output queries to perform one symbolic output query on a symbolic word of length n.*

Symbolic wait query. Let us proceed with symbolic output queries. We recall the definition. For a sw (symbolic word) $\mathtt{w}$ inducing a concrete run $\pi = q_0^{\mathcal{M}} \xrightarrow{i_1}$

$\cdots \xrightarrow{i_n} q_n \in runs(\mathcal{M})$ such that $\overline{i_1 \cdots i_n} = \mathtt{w}$, $\mathbf{WQ^s}(\mathtt{w})$ returns the set of all pairs (j, c) such that $q_{j-1} \xrightarrow[(x,c)]{i_j} \cdots \xrightarrow{i_n} q_n \xrightarrow{to[x]}$ is x-spanning.

Let $\mathtt{w}$ be a sw for which we want to call $\mathbf{WQ^s}(\mathtt{w})$. As for symbolic output queries, such a query is performed to know the set of enabled timers of a state q that is already present in $\mathcal{T}$. Let $v = \mathrm{cnstr}^{\mathcal{T}}(q)$ be a tiw. Let us assume in the following that there is no unexpected timeout when performing a concrete output query (otherwise, we can proceed as explained above). So, we have that $(q_0^{\mathcal{M}}, \emptyset) \xrightarrow{v} (f(q), \kappa)$. Recall that each input in v is such that the fractional part of the delays up to the input is unique. Hence, it is sufficient to wait "long enough" in $(f(q), \kappa)$ to identify one potential enabled timer. For now, assume the learner knows a constant Δ that is at least as large as the largest constant appearing on any update of $\mathcal{M}$. That is, if we wait Δ units of time in a configuration and no timeout occurs, then we are sure that $\chi_0^{\mathcal{M}}(f(q)) = \emptyset$ (i.e., we add Δ to the last delay of v). We discuss below how to deduce Δ during the learning process. Moreover, by the uniqueness of the fractional parts, it is easy to identify which transition (re)started the timer that times out.

So, we have to explain how to ensure that we eventually observe every enabled timer of $f(q)$. Recall that a timer must be started on a transition before q to be potentially active in q. Thus, we define a set $Potential(q) = \{x_p \mid p \text{ is an ancestor of } q\}$ that contains every timer that *may* be enabled in q. Our idea is to check each timer one by one to determine whether it is enabled.

We select any timer x from $Potential(q)$ and refine the constraints of $\mathrm{cnstr}^{\mathcal{T}}(q)$ to enforce that the last delay is equal to Δ, and the input that initially starts x is triggered as soon as possible while still satisfying the other constraints of $\mathrm{cnstr}^{\mathcal{T}}(q)$. It may be that the resulting constraints for that x are not satisfiable, meaning by Lemma 7 that the run ending with $to[x]$ is not feasible and x can not be enabled in q. If there exists a solution, i.e., a tiw v, we can ask $\mathbf{OQ}(w)$ to obtain a tow ω. By the uniqueness of the fractional parts, it is thus easy to check whether x times out by waiting in q. So, we can easily deduce which transition restarts x. Moreover, from the delays in ω, the constant of the update restarting x can be computed.

We repeat this procedure for every timer in $Potential(q)$. Once this is done, we know the immediate timers of q. We can thus add q to $\mathcal{E}^{\mathcal{T}}$. Let n be the number of states in the path from q_0 to q. The size of $Potential(q)$ is at most n, and, for every timer x in this set, we need at most n concrete output queries.

Proposition 5. *We need at most n^2 concrete output queries to perform one symbolic wait query (correct up to our guess of Δ) on a symbolic word of length n.*

Guessing Δ. Let us quickly explain how the learner can infer Δ during the learning process. At first, Δ can be assumed to be any integer (preferably small when interacting with real-world systems). At some point, an update (x, c) may be learned by performing a wait query (or processing the counterexample of an equivalence query) with $c > \Delta$. That is, we now know that Δ is not the

largest constant appearing in $\mathcal{M}$. We thus set Δ to be c. This implies that a new wait query must be performed in every explored state, in order to discover potentially missing enabled timers. That is, throughout the learning algorithm, the set of enabled timers in $\mathcal{T}$ may be an under-approximation of the set of enabled timers of the corresponding state in $\mathcal{M}$ (cf. seismic events introduced in Section 3.3 which require rebuilding the tree from the root). However, we will eventually learn the correct value of Δ (cf. Appendix E.3 for a bound relative to the unknown $\mathcal{M}$ on how many times seismic events and, more generally, the discovery of new timers, can occur).

Symbolic equivalence query. Finally, let us explain how to do a symbolic equivalence query using a concrete equivalence query and knowledge of the MMT. Let $\mathcal{H}$ be the complete hypothesis provided to $\mathbf{EQ^s}(\mathcal{H})$. Notice that we don't even need to use a concrete equivalence query. Indeed, we can implement a symbolic equivalence algorithm (similar to the reachability algorithm [7] for automata with timers) which will satisfy the required specification. As an alternative, we could use the contrapositive of Lemma 4 and make use of a concrete equivalence query (in case one is already available) to construct a witness of symbolic nonequivalence. Importantly, when using this alternative solution, the implemented symbolic equivalence query will return **yes** if the hypothesis is timed equivalent to the hidden MMT (as opposed to when they are symbolically equivalent, cf. Lemma 4). If they are not timed equivalent, it will construct a counterexample of symbolic equivalence from the counterexample of timed equivalence. This is not exactly the specification of the symbolic equivalence query as in the rest of the paper, but it suffices for our learning algorithm to terminate and return an MMT that is timed equivalent to the hidden one we are trying to learn.

Note that we only need at most one concrete equivalence query to implement the symbolic version of the query.

Proposition 6. *We need at most one concrete equivalence query to perform one symbolic equivalence query.*

In conclusion, each symbolic query can be done via a polynomial number of concrete queries. Hence, we proved Lemma 2.

D   More details on observation trees

Let $\mathcal{M}$ be the s-learnable MMT of the teacher. In this section, we give more details on observation trees and their properties. We first formalize how one can map each state (resp. timer) of $\mathcal{T}$ to a state (resp. timer) of $\mathcal{M}$. Then, in Appendix D.2, we prove Theorem 2. In Appendix D.3, we show that if w is a witness of $p_0 \#^m p_0'$ and w can be read from a state r (i.e., $read^m_{\substack{w \\ p_0 \to}}(r)$ is defined), then r must be apart from p_0 or p_0' under some matchings derived from m. This property is used in the learning algorithm to reduce the number of possible hypotheses that can be constructed. See Appendix E.3.

D.1   Functional simulation

In order to link (the finitely many) runs of $\mathcal{T}$ to runs of $\mathcal{M}$, we need a function $f : Q^{\mathcal{T}} \to Q^{\mathcal{M}}$ that maps states of $\mathcal{T}$ to states of $\mathcal{M}$ such that every outgoing transition from a state $q \in Q^{\mathcal{T}}$ can be reproduced from $f(q)$ while producing the same output. In addition, since $\mathcal{T}$ and $\mathcal{M}$ may use different timers, we need a function $g : X^{\mathcal{T}} \to X^{\mathcal{M}}$ that maps active timers of $\mathcal{T}$ to timers of $\mathcal{M}$. We require that when a timer x is active in q, the corresponding timer $g(x)$ is active in $f(q)$. When x and y are distinct timers that are both active in some state q, we do not allow g to map x and y to the same timer of $\mathcal{M}$. These conditions imply that the number of timers that are active in $f(q)$ is at least as large as the number of timers active in q. Furthermore, for any transition $q \xrightarrow[u]{i/o} q'$ in $\mathcal{T}$, there exists a transition $f(q) \xrightarrow[u']{i'/o} f(q')$ in $\mathcal{M}$ such that $i' = i$ if $i \in I$, or $i' = to[g(x)]$ if $i = to[x]$, i.e., we read a corresponding action, output the same symbol and reach the state corresponding to q'. Moreover, if $u = (x, c)$, then $u' = (g(x), c)$, i.e., we do the same update. However, if $u = \bot$, we may not have found the actual update yet, so we do not impose anything on u' (it can be any update in $U(\mathcal{M})$). Then, for a run π of $\mathcal{T}$, we can consider its corresponding run in $\mathcal{M}$ via f and g, noted $\langle f, g \rangle(\pi)$, which must preserve the spanning sub-runs.

Definition 7 (Functional simulation). *Let $\mathcal{T}$ be an observation tree and $\mathcal{M}$ be a s-learnable MMT. A functional simulation $\langle f, g \rangle : \mathcal{T} \to \mathcal{M}$ is a pair of a map $f : Q^{\mathcal{T}} \to Q^{\mathcal{M}}$ and a map $g : \cup_{q \in Q^{\mathcal{T}}} \chi^{\mathcal{T}}(q) \to X^{\mathcal{M}}$. Let g be lifted to actions such that $g(i) = i$ for every $i \in I$, and $g(to[x]) = to[g(x)]$ for every $x \in \mathbf{dom}(g)$. We require that $\langle f, g \rangle$ preserves initial states, active timers, and transitions:*

$$f(q_0^{\mathcal{T}}) = q_0^{\mathcal{M}} \tag{FS0}$$

$$\forall q \in Q^{\mathcal{T}}, \forall x \in \chi^{\mathcal{T}}(q) : g(x) \in \chi^{\mathcal{M}}(f(q)) \tag{FS1}$$

$$\forall q \in Q^{\mathcal{T}}, \forall x, y \in \chi^{\mathcal{T}}(q) : x \neq y \ \Rightarrow \ g(x) \neq g(y) \tag{FS2}$$

$$\forall q \xrightarrow[(x,c)]{i/o} q' : f(q) \xrightarrow[(g(x),c)]{g(i)/o} f(q') \tag{FS3}$$

$$\forall q \xrightarrow[\bot]{i/o} q' : f(q) \xrightarrow{g(i)/o} f(q') \tag{FS4}$$

Thanks to (FS3) and (FS4), we lift $\langle f, g \rangle$ to runs in a straightforward manner. We require:

$$\forall \pi \in \mathit{runs}(\mathcal{T}) : \ \langle f, g \rangle(\pi) \text{ is } y\text{-spanning} \ \Rightarrow \ \exists x : \pi \text{ is } x\text{-spanning} \wedge g(x) = y \tag{FS5}$$

We say that $\mathcal{T}$ is an observation tree for $\mathcal{M}$ *if there exists $\langle f, g \rangle : \mathcal{T} \to \mathcal{M}$.*

Example 9. Let $\mathcal{M}$ be the MMT of Figure 1. Then, the observation tree $\mathcal{T}$ of Figure 2 is an observation tree for $\mathcal{M}$ with the functional simulation $\langle f, g \rangle$ such that

$$f(t_0) = f(t_8) = f(t_{10}) = q_0 \qquad f(t_1) = f(t_2) = f(t_4) = q_1 \qquad f(t_3) = q_2$$
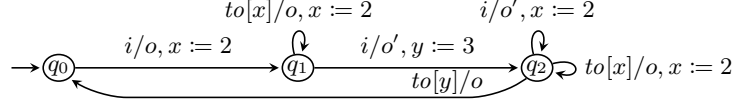
Fig. 6: An MMT with $\chi(q_0) = \emptyset$, $\chi(q_1) = \{x\}$, and $\chi(q_2) = \{x, y\}$.

$$f(t_5) = f(t_6) = q_3 \qquad\qquad f(t_7) = f(t_9) = q_5,$$

and

$$g(x_1) = g(x_6) = x \qquad\qquad g(x_3) = y.$$

Let $\pi = t_1 \xrightarrow{i \cdot to[x_1] \cdot to[x_3]}$. So, $\langle f, g \rangle(\pi) = q_1 \xrightarrow{i \cdot to[x] \cdot to[y]}$. Observe that π is x_3-spanning and $\langle f, g \rangle(\pi)$ is y-spanning. As $g(x_3) = y$, (FS5) is satisfied.

Observe that for fixed $\mathcal{T}$ and $\mathcal{M}$ there exists at most one functional simulation. Further properties can be deduced from the definition of $\langle f, g \rangle$.

Corollary 3. *Let $\mathcal{T}$ be an observation tree for a s-learnable $\mathcal{M}$ with $\langle f, g \rangle$. Then, for all states $q \in Q^{\mathcal{T}}$ we have:*

$$|\chi^{\mathcal{T}}(q)| \leq |\chi^{\mathcal{M}}(f(q))| \qquad and \qquad \forall x \in \chi_0^{\mathcal{T}}(q) : g(x) \in \chi_0^{\mathcal{M}}(f(q)).$$

Proof. We start with the first part, i.e., $|\chi^{\mathcal{T}}(q)| \leq |\chi^{\mathcal{M}}(f(q))|$. By (FS1), we have that any timer x that is active in q is such that $g(x)$ is active in $f(x)$. Moreover, by (FS2), $g(x) \neq g(y)$ for any $x \neq y \in \chi^{\mathcal{T}}(q)$. So, it is not possible for q to have more active timers than $f(q)$.

Now, the second part, i.e., $\forall x \in \chi_0^{\mathcal{T}}(q) : g(x) \in \chi_0^{\mathcal{M}}(f(q))$. Let $x \in \chi_0^{\mathcal{T}}(q)$. By definition of $\mathcal{T}$, it follows that $q \xrightarrow{to[x]}$ is defined. So, by (FS3) and (FS4), we have $f(q) \xrightarrow{to[g(x)]}$, meaning that $g(x) \in \chi_0^{\mathcal{M}}(f(q))$, as $\mathcal{M}$ is complete. $\qquad\square$

Finally, let us highlight that the notion of explored states only makes sense when $\mathcal{M}$ is s-learnable. Let $\mathcal{T}$ be the observation tree of Figure 2 and $\mathcal{N}$ be the not-s-learnable MMT of Figure 6. We can still define the maps $f : Q^{\mathcal{T}} \to Q^{\mathcal{N}}$ and $g : X^{\mathcal{T}} \to X^{\mathcal{N}}$:

$$f(t_0) = f(t_8) = f(t_{10}) = q_0 \quad f(t_1) = f(t_2) = f(t_4) = q_1$$
$$f(t_3) = f(t_5) = f(t_6) = f(t_7) = f(t_9) = q_2$$
$$g(x_1) = g(x_6) = x \qquad\qquad g(x_3) = y.$$

We have $|\chi_0^{\mathcal{T}}(t_3)| = 1$ but $|\chi_0^{\mathcal{N}}(f(t_3))| = |\chi_0^{\mathcal{N}}(q_2)| = 2$. However, as every run of $\mathcal{T}$ must be feasible and x_3 cannot time out in t_3, it is impossible to get the equality. Therefore, in order to define explored states, we must require that $\mathcal{M}$ is s-learnable.

D.2   Proof of Theorem 2

We now show Theorem 2, which we repeat for convenience.

Theorem 2. *Let $\mathcal{T}$ be an observation tree for an s-learnable MMT with functional simulation $\langle f, g \rangle$, $p, p' \in Q^{\mathcal{T}}$, and $m, m' : p \leftrightarrow p'$ matchings. Then,*

- $w \vdash p \;\#^m\; p' \wedge m \subseteq m' \Rightarrow w \vdash p \;\#^{m'}\; p'$, and
- $p \;\#^m\; p' \Rightarrow f(p) \neq f(p') \vee \exists x \in \text{dom}(m) : g(x) \neq g(m(x))$.

Let us show each part separately.

Lemma 8. $w \vdash p \;\#^m\; p' \wedge m \subseteq m' \Rightarrow w \vdash p \;\#^{m'}\; p'$.

Proof. Let $w \vdash p \;\#^m\; p'$ and $m \subseteq m'$. Moreover, let $p_0 = p$, $p'_0 = p'$,

$$\pi = p_0 \xrightarrow{i_1} p_1 \xrightarrow{i_2} \cdots \xrightarrow[u]{i_n/o} p_n, \text{ and}$$

$$\pi' = read^{m}_{p \xrightarrow{w}}(p') = p'_0 \xrightarrow{i'_1} p'_1 \xrightarrow{i'_2} \cdots \xrightarrow[u']{i'_n/o'} p'_n$$

with $m^{\pi}_{\pi'} : \pi \leftrightarrow \pi'$. By definition, each i_j is either an input, or $to[x]$ with $x \in$ $\text{dom}(m^{\pi}_{\pi'})$. Thus, since $m \subseteq m'$, it follows that $read^{m'}_{p \xrightarrow{w}}(p')$ uses exactly the same actions and takes the same transitions as $read^{m}_{p \xrightarrow{w}}(p')$. That is, $read^{m'}_{p \xrightarrow{w}}(p') = read^{m}_{p \xrightarrow{w}}(p')$. There are five cases:

- There exists $x \in \text{dom}(m^{\pi}_{\pi'})$ such that $x \not{\#} m^{\pi}_{\pi'}(x)$. If $x \not{\#} m(x)$, then $x \not{\#} m'(x)$ since $m \subseteq m'$. If there exists $k \in \{1, \ldots, n\}$ such that $x_{p_k} \not{\#} x_{p'_k}$, this does not change when extending m. Hence, $x \in \text{dom}(m'^{\pi}_{\pi'})$ and $x \not{\#} m'^{\pi}_{\pi'}(x)$, i.e., we have $w \vdash p \;\#^{m'}\; p'$.
- $o \neq o'$, which, clearly, does not depend on m. So, $w \vdash p \;\#^{m'}\; p'$.
- Likewise if $u = (x, c)$ and $u' = (x', c')$ with $c \neq c'$.
- $p_n, p'_n \in \mathcal{E}^{\mathcal{T}}$ and $|\chi_0(p_n)| \neq |\chi_0(p'_n)|$, which, again, does not depend on m. So, $w \vdash p \;\#^{m'}\; p'$.
- $p_n, p'_n \in \mathcal{E}^{\mathcal{T}}$ and there is $x \in \text{dom}(m^{\pi}_{\pi'})$ such that $x \in \chi_0(p_n) \Leftrightarrow m^{\pi}_{\pi'}(x) \notin \chi_0(p'_n)$. If $x \in \text{dom}(m)$ and as $m \subseteq m'$, we still have $x \in \chi_0(p_n) \Leftrightarrow m'(x) \notin \chi_0(p'_n)$. Likewise if there is a $k \in \{1, \ldots, n\}$ such that $x_{p_k} \in \chi_0(p_n) \Leftrightarrow x_{p'_k} \notin \chi_0(p'_n)$. Therefore, we have again $w \vdash p \;\#^{m'}\; p'$.

In every case, we obtain that $w \vdash p \;\#^{m'}\; p'$. ☐

Hence, the first part of Theorem 2 holds. We now focus on the second part, which requires an intermediate result. Recall that, given two runs $\pi = p_0 \xrightarrow{i_1} \cdots \xrightarrow{i_n} p_n$ and $\pi' = p'_0 \xrightarrow{i'_1} \cdots \xrightarrow{i'_n} p'_n$, $m^{\pi}_{\pi'} : \pi \leftrightarrow \pi'$ denotes the matching such that $m^{\pi}_{\pi'} = m \cup \{(x_{p_j}, x_{p'_j}) \mid j \in \{1, \ldots, n\}\}$ and $i'_j = m^{\pi}_{\pi'}(i_j)$ for all $j \in \{1, \ldots, n\}$. Given such a matching $m^{\pi}_{\pi'} : \pi \leftrightarrow \pi'$, the next lemma states that if $f(p_0) = f(p'_0)$ and m agrees with g, then $\langle f, g \rangle(\pi) = \langle f, g \rangle(\pi')$ and $m^{\pi}_{\pi'}$ (restricted to the started timers, ensuring that g is defined over those timers) also agrees with g.

Lemma 9. *Let $p_0, p_0' \in Q^{\mathcal{T}}$ and a matching $m : p_0 \leftrightarrow p_0'$ such that $f(p_0) = f(p_0')$ and $g(x) = g(m(x))$ for all $x \in \mathsf{dom}(m)$. Moreover, let $w = i_1 \cdots i_n$ be a word such that*

$$\pi = p_0 \xrightarrow{i_1} p_1 \xrightarrow{i_2} \cdots \xrightarrow{i_n} p_n \in runs(\mathcal{T}), \ and$$

$$\pi' = read_\pi^m(p_0') = p_0' \xrightarrow{i_1'} p_1' \xrightarrow{i_2'} \cdots \xrightarrow{i_n'} p_n' \in runs(\mathcal{T}).$$

Then, $\langle f, g \rangle(\pi) = \langle f, g \rangle(\pi')$ and $g(x) = g(m_{\pi'}^\pi(x))$ for all $x \in \mathsf{dom}(m_{\pi'}^\pi)$ with $x \in \mathsf{dom}(m)$ or $x = x_{p_k}$, $k \in \{1, \cdots, n\}$, such that x_{p_k} is started along π and x_{p_k}' is started along π'.

Proof. We prove the lemma by induction over n, the length of w.

Base case: $|w| = 0$, i.e., $w = \varepsilon$. We thus have

$$\pi = p_0 \xrightarrow{\varepsilon} p_0 \qquad\qquad \text{and} \qquad\qquad \pi' = p_0' \xrightarrow{\varepsilon} p_0'$$

which means that we have the following runs in $\mathcal{M}$

$$f(p_0) \xrightarrow{\varepsilon} f(p_0) \qquad\qquad \text{and} \qquad\qquad f(p_0') \xrightarrow{\varepsilon} f(p_0').$$

As $f(p_0) = f(p_0')$, these runs of $\mathcal{M}$ are equal. Moreover, as $m_{\pi'}^\pi = m$, the second part of the lemma holds.

Induction step: Let $\ell \in \mathbb{N}$ and assume the lemma holds for length ℓ. Let $v = i_1 \cdots i_{\ell+1} = w \cdot i_{\ell+1}$ be a word of length $\ell + 1$ such that

$$p_0 \xrightarrow{i_1} \cdots \xrightarrow{i_\ell} p_\ell \xrightarrow{i_{\ell+1}} p_{\ell+1} \in runs(\mathcal{T}) \text{ and}$$

$$read_{p_0 \xrightarrow{w \cdot i_{\ell+1}} p_{\ell+1}}^m (p_0') = p_0' \xrightarrow{i_1'} \cdots \xrightarrow{i_\ell'} p_n' \xrightarrow{i_{\ell+1}'} p_{\ell+1}' \in runs(\mathcal{T}).$$

Let $\pi = p_0 \xrightarrow{w} p_\ell$ and $\pi' = read_\pi^m(p_0') = p_0' \xrightarrow{w'} p_\ell'$. By the induction hypothesis with w, it holds that

- the runs $\langle f, g \rangle(\pi)$ and $\langle f, g \rangle(\pi')$ are equal, and
- $g(x) = g(m_{\pi'}^\pi(x))$ for all started timers $x \in \mathsf{dom}(m_{\pi'}^\pi)$.

It is thus sufficient to show that

- $\langle f, g \rangle(p_\ell \xrightarrow{i} p_{\ell+1}) = \langle f, g \rangle(p_\ell' \xrightarrow{i'} p_{\ell+1}')$, and
- $g(x_{p_{\ell+1}}) = g(x_{p_{\ell+1}'})$ if both $x_{p_{\ell+1}}$ and $x_{p_{\ell+1}'}$ are started, i.e., if $x_{p_{\ell+1}} \in \chi(p_{\ell+1})$ and $x_{p_{\ell+1}'} \in \chi(p_{\ell+1}')$.

By definition of $read_{p_0 \xrightarrow{w \cdot i_{\ell+1}} p_{\ell+1}}^m (p_0')$, we have

$$i_{\ell+1}' = \begin{cases} i_{\ell+1} & \text{if } i_{\ell+1} \in I \\ to[m(x)] & \text{if } i_{\ell+1} = to[x] \text{ with } x \in \mathsf{dom}(m) \\ to[x_{p_k'}] & \text{if } i_{\ell+1} = to[x_{p_k}] \text{ with } k \in \{1, \ldots, \ell\} \end{cases}$$

We can be more precise for the last case, i.e., when $i_{\ell+1} = to[x_{p_k}]$ with $k \in \{1, \ldots, \ell\}$. As $p_\ell \xrightarrow{to[x_{p_k}]} \in runs(\mathcal{T})$, it must be that $x_{p_k} \in \chi(p_\ell)$. Hence, by definition of an observation tree, $x_{p_k} \in \chi(p_k)$. Likewise, as $p'_\ell \xrightarrow{to[x_{p'_k}]} \in runs(\mathcal{T})$, it follows that $x_{p'_k} \in \chi(x_{p'_k})$. Hence, $g(x_{p_k})$ and $g(x_{p'_k})$ are both defined when the third case holds.

By definition of g, it holds that

$$g(i_{\ell+1}) = \begin{cases} i_{\ell+1} & \text{if } i_{\ell+1} \in I \\ to[g(x)] & \text{if } i_{\ell+1} = to[x] \text{ with } x \in \mathsf{dom}(m) \\ to[g(x_{p_k})] & \text{if } i_{\ell+1} = to[x_{p_k}] \text{ with } k \in \{1, \ldots, \ell\} \end{cases}$$

and

$$g(i'_{\ell+1}) = \begin{cases} i'_{\ell+1} & \text{if } i'_{\ell+1} \in I \\ to[g(m(x))] & \text{if } i'_{\ell+1} = to[m(x)] \text{ with } x \in \mathsf{dom}(m) \\ to[g(x_{p'_k})] & \text{if } i'_{\ell+1} = to[x_{p'_k}] \text{ with } k \in \{1, \ldots, \ell\}. \end{cases}$$

We have

- $i'_{\ell+1} = i_{\ell+1}$ if $i_{\ell+1} \in I$,
- $g(m(x)) = g(x)$ for all $x \in \mathsf{dom}(m)$ by the lemma statement, and
- by the induction hypothesis, $g(x_{p'_k}) = g(x_{p_k})$ for all $k \in \{1, \ldots, \ell\}$ such that $x_{p_k} \in \chi(p_k)$ and $x_{p'_k} \in \chi(p'_k)$.

It follows that $g(i'_{\ell+1}) = g(i_{\ell+1})$.

As $f(p_\ell) = f(p'_\ell)$ by induction hypothesis and $g(i_{\ell+1}) = g(i'_{\ell+1})$, it holds by determinism of $\mathcal{M}$ that $\langle f, g \rangle (p_\ell \xrightarrow{i} p_{\ell+1}) = \langle f, g \rangle (p'_\ell \xrightarrow{i'} p'_{\ell+1})$.

To complete the proof of the lemma, it remains to prove that if $x_{p_{\ell+1}} \in \chi(p_{\ell+1})$ and $x_{p'_{\ell+1}} \in \chi(p'_{\ell+1})$, then $g(x_{p_{\ell+1}}) = g(x_{p'_{\ell+1}})$. We have $x_{p_{\ell+1}} \in \chi(p_{\ell+1})$ (resp. $x_{p'_{\ell+1}} \in \chi(p'_{\ell+1})$) if the update of the transition $p_\ell \xrightarrow{i} p_{\ell+1}$ (resp. $p'_\ell \xrightarrow{i'} p'_{\ell+1}$) is equal to $(x_{p_{\ell+1}}, c)$ for some c (resp. $(x_{p'_{\ell+1}}, c')$ for some c'). As $\langle f, g \rangle (p_\ell \xrightarrow{i} p_{\ell+1}) = \langle f, g \rangle (p'_\ell \xrightarrow{i'} p'_{\ell+1})$ and $\langle f, g \rangle$ is a functional simulation, by (FS3), we get that $g(x_{p_{\ell+1}}) = g(x_{p'_{\ell+1}})$ and $c = c'$. $\qquad\square$

We are now ready the second part of Theorem 2.

Theorem 3 (Soundness). *Let $\mathcal{T}$ be an observation tree for a s-learnable MMT $\mathcal{M}$ with the functional simulation $\langle f, g \rangle$, $p, p' \in Q^{\mathcal{T}}$, and $m : p \leftrightarrow p'$ be a matching. Then, $p \mathrel{\#^m} p' \Rightarrow f(p) \neq f(p') \vee \exists x \in \mathsf{dom}(m) : g(x) \neq g(m(x))$.*

Proof. Towards a contradiction, assume $w = i_1 \cdots i_n \vdash p \mathrel{\#^m} p'$, $f(p) = f(p')$ and $\forall x \in \mathsf{dom}(m) : g(x) = g(m(x))$. Let

$$\pi = p_0 \xrightarrow{i_1} \cdots \xrightarrow{i_n} p_n \qquad \text{and} \qquad \pi' = read_\pi^m(p'_0) = p'_0 \xrightarrow{i'_1} \cdots \xrightarrow{i'_n} p'_n$$

with $p_0 = p$ and $p'_0 = p'$. Both runs exist in $\mathcal{T}$ as $w \vdash p_0 \#^m p'_0$. By Lemma 9, we thus have $\langle f, g \rangle(\pi) = \langle f, g \rangle(\pi')$ and $g(x) = g(m^\pi_{\pi'}(x))$ for all started timers $x \in \mathsf{dom}(m^\pi_{\pi'})$. In particular, the equality of the runs holds for the last transition:

$$f(p_{n-1}) \xrightarrow[u]{g(i_n)/o} f(p_n) \qquad = \qquad f(p'_{n-1}) \xrightarrow[u]{g(i'_n)/o} f(p'_n). \qquad (2)$$

Notice the same output and update, by determinism of $\mathcal{M}$.

First, if $w \vdash p_0 \#^m p'_0$ is structural, then there must exist a timer $x \in \mathsf{dom}(m^\pi_{\pi'})$ such that $x \,{\not\mathrel{\#}}\, m^\pi_{\pi'}(x)$. By definition of the timer apartness, $x \neq m^\pi_{\pi'}(x)$ and there must exist a state q such that $x, m^\pi_{\pi'}(x) \in \chi(q)$. By (FS2), $g(x) \neq g(m^\pi_{\pi'}(x))$, which is a contradiction.

Hence, assume $w \vdash p_0 \#^m p'_0$ is behavioral. Let us study the different cases.

- Assume (outputs) holds, i.e.,

$$p_{n-1} \xrightarrow{i_n/o_n} p_n \qquad \text{and} \qquad p'_{n-1} \xrightarrow{i'_n/o'_n} p'_n$$

with $o_n \neq o'_n$. By (FS3), (FS4), and (2), we have $o_n = o$ and $o'_n = o$, which is a contradiction.

- Assume (constants) holds, i.e.,

$$p_{n-1} \xrightarrow[(x,c)]{i_n} p_n \qquad \text{and} \qquad p'_{n-1} \xrightarrow[(x',c')]{i'_n} p'_n$$

with $c \neq c'$. By (FS3),

$$f(p_{n-1}) \xrightarrow[(g(x),c)]{g(i_n)} f(p_n) \qquad \text{and} \qquad f(p'_{n-1}) \xrightarrow[(g(x'),c')]{g(i'_n)} f(p'_n).$$

By (2), we get $(g(x), c) = (g(x'), c')$, which is a contradiction with $c \neq c'$.

- Assume (sizes) holds, i.e., $p_n, p'_n \in \mathcal{E}^\mathcal{T}$ and $|\chi^\mathcal{T}_0(p_n)| \neq |\chi^\mathcal{T}_0(p'_n)|$. By the definition of explored states (see Section 3.1), we have

$$|\chi^\mathcal{T}_0(p_n)| = |\chi^\mathcal{M}_0(f(p_n))| \qquad \text{and} \qquad |\chi^\mathcal{T}_0(p'_n)| = |\chi^\mathcal{M}_0(f(p'_n))|.$$

It follows that $|\chi^\mathcal{M}_0(f(p_n))| \neq |\chi^\mathcal{M}_0(f(p'_n))|$, which is in contradiction with $f(p_n) = f(p'_n)$.

- Assume (enabled) holds, i.e., $p_n, p'_n \in \mathcal{E}^\mathcal{T}$ and there exists $x \in \mathsf{dom}(m^\pi_{\pi'})$ such that

$$x \in \chi^\mathcal{T}_0(p_n) \Leftrightarrow m^\pi_{\pi'}(x) \notin \chi^\mathcal{T}_0(p'_n).$$

Without loss of generality, suppose $x \in \chi^\mathcal{T}_0(p_n)$ and $m^\pi_{\pi'}(x) \notin \chi^\mathcal{T}_0(p'_n)$. Recall that, by (FS2) and the second part of Corollary 3, we have $y \in \chi^\mathcal{T}_0(q) \Leftrightarrow g(y) \in \chi^\mathcal{M}_0(f(q))$ for all $q \in \mathcal{E}^\mathcal{T}$ and $y \in \mathsf{dom}(g)$ (see Section 3.1), In order to leverage this, we thus need to argue that $m^\pi_{\pi'}(x) \in \mathsf{dom}(g)$, i.e., the timer is started at some point. There are two cases:

- If $x \in \mathsf{dom}(m)$, then, by definition, $x \in \chi^{\mathcal{T}}(p_0)$ and $m^\pi_{\pi'} = m(x) \in \chi^{\mathcal{T}}(p'_0)$. Hence,

$$g(x) \in \chi_0^{\mathcal{M}}(f(p_n)) \qquad \text{and} \qquad g(m^\pi_{\pi'}(x)) \notin \chi_0^{\mathcal{M}}(f(p'_n)).$$

  As $g(m^\pi_{\pi'}(x)) = g(x)$ and $f(p'_n) = f(p_n)$, we have $g(x) \notin \chi_0^{\mathcal{M}}(f(p_n))$, which is a contradiction.

- If $x \notin \mathsf{dom}(m)$, then it must be that $x = x_{p_k}$ for some $k \in \{1, \dots, n\}$. Let k be the smallest such index. We can assume that $y \in \chi_0^{\mathcal{T}}(p_n) \Leftrightarrow m(y) \in \chi_0^{\mathcal{T}}(p'_n)$ for each $y \in \mathsf{dom}(m)$. That is, we have

$$x_{p_k} \in \chi_0^{\mathcal{T}}(p_n) \Leftrightarrow x_{p'_k} \notin \chi_0^{\mathcal{T}}(p'_n).$$

This means that $x_{p_j} \in \chi_0^{\mathcal{T}}(p_n) \Leftrightarrow x_{p'_j} \in \chi_0^{\mathcal{T}}(p'_n)$ for every $j \in \{1, \dots, k-1\}$. Recall that we assumed $x_{p_k} \in \chi_0^{\mathcal{T}}(p_n)$ and $x_{p'_k} \notin \chi_0^{\mathcal{T}}(p'_n)$.

By definition of an observation tree, this means that $p_n \xrightarrow{to[x_{p_k}]} p_{n+1} \in runs(\mathcal{T})$ for some state p_{n+1}. Moreover, as $f(p_n) = f(p'_n)$, $p_n, p'_n \in \mathcal{E}^{\mathcal{T}}$ and $|\chi_0^{\mathcal{T}}(p_n)| = |\chi_0^{\mathcal{T}}(p'_n)|$, there exist $x' \in \chi_0^{\mathcal{T}}(p'_n)$ and $y \in \chi_0^{\mathcal{M}}(f(p_n))$ such that

$$g(x') = g(x_{p_k}) = y, \qquad p'_n \xrightarrow{to[x']} p'_{n+1}, \text{ and}$$

$$\langle f, g \rangle (p_n \xrightarrow{to[x_{p_k}]} p_{n+1}) = \langle f, g \rangle (p'_n \xrightarrow{to[x']} p'_{n+1}).$$

That is,

$$(f(p_n) = f(p'_n)) \xrightarrow{to[y]} (f(p_{n+1}) = f(p'_{n+1})).$$

Let $\ell \in \{k, \dots, n\}$ be the largest index such that $f(p_{\ell-1}) \xrightarrow[\;(y,c)\;]{g(i_\ell)} f(p_\ell)$, i.e., ℓ is the index of the last transition before $f(p_n)$ that (re)starts y. In other words, $\langle f, g \rangle (p_{\ell-1} \xrightarrow{i_\ell \cdots i_n \cdot to[x_{p_k}]} p_{n+1})$ is y-spanning. (Observe that we may have $\ell = k$.) As

$$\langle f, g \rangle (p_{\ell-1} \xrightarrow{i_\ell \cdots i_n \cdot to[x_{p_k}]} p_{n+1}) \quad = \quad \langle f, g \rangle (p'_{\ell-1} \xrightarrow{i'_\ell \cdots i'_n \cdot to[x']} p'_{n+1}),$$

it follows by (FS5) and (FS2) that $p'_{\ell-1} \xrightarrow{i'_\ell \cdots i'_n \cdot to[x']} p'_{n+1}$ is x'-spanning, and thus $p'_{\ell-1} \xrightarrow[\;(x',c)\;]{i'_\ell} p'_\ell$.

In order to obtain a contradiction, let us argue that $x' = x_{p'_k}$. Once we have this equality, we can deduce that $x_{p'_k} \in \chi_0^{\mathcal{T}}(p'_n)$ (as $x' \in \chi_0^{\mathcal{T}}(p'_n)$), which is a contradiction with our assumption that $x_{p'_k} \notin \chi_0^{\mathcal{T}}(p'_n)$. To do so, we start from the $g(i_\ell)$-transition of the run $f(p_{k-1}) \xrightarrow{g(i_k \cdots i_n) \cdot to[y]} f(p_{n+1})$ and backtrack until we identify the transition that initially starts y.

When we consider $g(i_\ell)$, we have two cases:

* $g(i_\ell) \in I$, meaning that $i_\ell = i'_\ell = g(i_\ell) \in I$ and the corresponding transitions in $\mathcal{T}$ start a fresh timer. Hence, by definition of $\mathcal{T}$, it must be that $p_{\ell-1} \xrightarrow[(x_{p_k}, c)]{i_\ell} p_\ell$ for some $c \in \mathbb{N}^{>0}$. That is, $\ell = k$. Moreover, $p'_{\ell-1} \xrightarrow[(x', c)]{i'_\ell} p'_\ell$ as the sub-run starting with that transition must be x'-spanning. Hence, $x' = x_{p'_k}$.

* $g(i_\ell) \notin I$, i.e., $g(i_\ell) = to[y]$ meaning that $i_\ell = to[z]$ with $g(z) = y = g(x_{p_k})$. Since x_{p_k} and z are both active in p_ℓ and $g(x_{p_k}) = g(z)$, it must be that $x_{p_k} = z$ by the contrapositive of (FS2). Likewise, $i'_\ell = to[x']$. We can thus seek a new y-spanning run that ends by the transition $\xrightarrow{to[y]} f(p_\ell)$. Let $j \in \{1, \ldots, \ell - 1\}$ be the index of the first transition of this y-spanning run. Observe that $j \geq k$. Indeed, if $j < k$, then it is not possible for x_{p_k} to be enabled in p_ℓ (by definition of an observation tree). That is, $j \in \{k, \ldots, \ell - 1\}$.

  When considering the transitions at indices ℓ and n, and those at indices j and $\ell - 1$, we have similar situations:

  · $p_{\ell-1} \xrightarrow{to[x_{p_k}]}$ and $p_n \xrightarrow{to[x_{p_k}]}$,

  · $p'_{\ell-1} \xrightarrow{to[x']}$ and $p'_n \xrightarrow{to[x']}$,

  · $(f(p_{\ell-1}) = f(p'_{\ell-1})) \xrightarrow{to[y]}$ and $(f(p_n) = f(p'_n)) \xrightarrow{to[y]}$, and

  · none of the updates between $f(p_\ell)$ and $f(p_n)$ restarts y, and likewise between $f(p_j)$ and $f(p_{\ell-1})$.

  Hence, we can repeat the same arguments using j and $\ell - 1$, instead of ℓ and n.

  That is, we can keep backtracking in $\mathcal{T}$ until we find a transition reading a symbol in I. As we argued, this transition must necessarily read i_k (so, $\ell = k$) and we conclude that $x' = x_{p_k}$.

In every case, we obtain that $x' = x_{p_k}$. As said above, this is enough to deduce a contradiction.

In every case, we obtain a contradiction. So, $f(p) \neq f(p')$ or $g(x) \neq g(m(x))$ for some $x \in \mathsf{dom}(m)$.   □

D.3   Weak co-transitivity

In $L^\#$ [32], the learner can reduce the number of hypotheses that can be constructed from a tree by exploiting the *weak co-transitivity* lemma, stating that, if we can read a witness w of the apartness (defined for classical Mealy machines) $p \mathrel{\#} p'$ from some state r, then $p \mathrel{\#} r$ or $p' \mathrel{\#} r$ (or both). Hence, before folding the tree to obtain a hypothesis, it is possible to ensure that each frontier state can be mapped to a single basis state.

For MMTs, this is trickier, as we have to take into account the timers and the mappings. That is, our version of the weak co-transitivity lemma states that if we can read a witness w of the *behavioral* apartness $p_0 \mathrel{\#^m} p'_0$ from a third state r_0 via some matching $\mu : p_0 \leftrightarrow r_0$, then we can conclude that p_0 and r_0

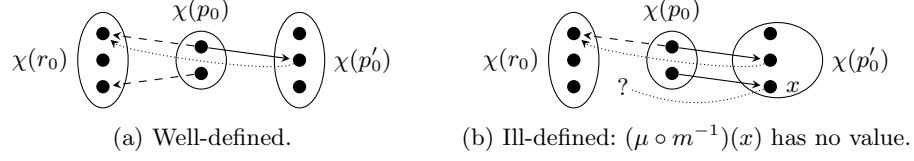(a) Well-defined.   (b) Ill-defined: $(\mu \circ m^{-1})(x)$ has no value.

Fig. 7: Visualizations of compositions $\mu \circ m^{-1}$ where m is drawn with solid lines, μ with dashed lines, and $\mu \circ m^{-1}$ with dotted lines.

are μ-apart or that p_0' and r_0 are $(\mu \circ m^{-1})$-apart. However, when $p_0 \mathrel{\#}^m p_0'$ is due to (constants), we need to extend the witness. In this case, since x is active in p_n (as $u = (x,c)$), there must exist an x-spanning run $p_{n-1} \xrightarrow[u]{i_n/o} p_n \xrightarrow{w^x}$ (by definition of an observation tree, see Definition 4). Hence, we actually "read" $w \cdot w^x$ from r_0, in order to ensure that an update (x',c') is present on the last transition of $read^\mu_{p_0 \xrightarrow{w}}(r_0)$.

Notice that the lemma requires that $\mathsf{dom}(m) \subseteq \mathsf{dom}(\mu)$ for the matching $\mu \circ m^{-1}$. See Figure 7 for illustrations of a well- and an ill-defined $\mu \circ m^{-1}$.

Lemma 10 (Weak co-transitivity). *Let $p_0, p_0', r_0 \in Q^{\mathcal{T}}$, $m : p_0 \leftrightarrow p_0'$ and $\mu : p_0 \leftrightarrow r_0$ be two matchings such that $\mathsf{dom}(m) \subseteq \mathsf{dom}(\mu)$. Let $w = i_1 \dots i_n$ be a witness of the behavioral apartness $p_0 \mathrel{\#}^m p_0'$ and $read^m_{p_0 \xrightarrow{w} p_n}(p_0') = p_0' \xrightarrow{w'} p_n'$. Let w^x be defined as follows:*

- *if $p_0 \mathrel{\#}^m p_0'$ due to (constants), w^x is a word such that $p_{n-1} \xrightarrow{i_n} p_n \xrightarrow{w^x}$ is x-spanning,*
- *otherwise, $w^x = \varepsilon$.*

If $read^\mu_{p_0 \xrightarrow{w \cdot w^x}}(r_0) \in runs(\mathcal{T})$ with $r_n \in \mathcal{E}^{\mathcal{T}}$, then $p_0 \mathrel{\#}^\mu r_0$ or $p_0' \mathrel{\#}^{\mu \circ m^{-1}} r_0$.

Proof. Let $p_0, p_0', r_0 \in Q^{\mathcal{T}}$, and $m : p_0 \leftrightarrow p_0'$ and $\mu : p_0 \leftrightarrow r_0$ be two matchings such that $\mathsf{dom}(m) \subseteq \mathsf{dom}(\mu)$. Let $w \cdot w^x$, w', and the runs as described in the statement, $n = |w|$, and $\ell = |w \cdot w^x|$. Moreover, let v be the word labeling the run from p_0, i.e., such that $read^\mu_{p_0 \xrightarrow{w \cdot w^x}}(r_0) = r_0 \xrightarrow{v} r_\ell$. We write w_j (resp. w_j', v_j) for a symbol of $w \cdot w^x$ (resp. w', v). (So, $n = \ell$ whenever $w \vdash p_0 \mathrel{\#}^m p_0'$ due to a condition that is not (constants), and $n < \ell$ otherwise.) We then have

$$p_0 \xrightarrow{w_1} p_1 \xrightarrow{w_2} \cdots \xrightarrow{w_n} p_n \xrightarrow{w_{n+1}} \cdots \xrightarrow{w_\ell} p_\ell,$$

$$read^m_{p_0 \xrightarrow{w} p_n}(p_0') = p_0' \xrightarrow{w_1'} p_1' \xrightarrow{w_2'} \cdots \xrightarrow{w_n'} p_n',$$

$$read^\mu_{p_0 \xrightarrow{w \cdot w^x} p_\ell}(r_0) = r_0 \xrightarrow{v_1} r_1 \xrightarrow{v_2} \cdots \xrightarrow{v_n} r_n \xrightarrow{v_{n+1}} \cdots \xrightarrow{v_\ell} r_\ell,$$

$$read^{\mu \circ m^{-1}}_{p_0' \xrightarrow{w'} p_n'}(r_0) = read^\mu_{p_0 \xrightarrow{w} p_n}(r_0) = r_0 \xrightarrow{v_1} r_1 \xrightarrow{v_2} \cdots \xrightarrow{v_n} r_n$$

with $r_n \in \mathcal{E}^{\mathcal{T}}$, by hypothesis. The run from p_0' does not read w^x after p_n'.

A first possibility is that $p_0 \#^\mu r_0$ or $p_0' \#^{\mu \circ m^{-1}} r_0$ due to structural apartness (with $w \cdot w^x$ or w' as witness). If this does not happen, from w being a witness of the behavioral apartness $p_0 \#^m p_0'$, we have to show that $p_0 \#^\mu r_0$ or $p_0' \#^{\mu \circ m^{-1}} r_0$ for one case among (outputs), (constants), (sizes), or (enabled). We do it by a case analysis. Let $o, o', \omega \in O$ such that $p_{n-1} \xrightarrow{w_n/o} p_n, p_{n-1}' \xrightarrow{w_n'/o'} p_n'$, and $r_{n-1} \xrightarrow{v_n/\omega} r_n$.

- If $o \neq o'$, then, necessarily, $\omega \neq o$ or $\omega \neq o'$ and we can apply (outputs) to obtain $p_0 \#^\mu r_0$ or $p_0' \#^{\mu \circ m^{-1}} r_0$.
- If $|\chi_0(p_n)| \neq |\chi_0(p_n')|$, then, necessarily, $|\chi_0(r_n)| \neq |\chi_0(p_n)|$ or $|\chi_0(r_n)| \neq |\chi_0(p_n')|$. As $p_n, p_n', r_n \in \mathcal{E}^{\mathcal{T}}$, we can apply (sizes) and get $p_0 \#^\mu r_0$ or $p_0' \#^{\mu \circ m^{-1}} r_0$.
- Suppose now that $p_0 \#^m p_0'$ is due to (enabled).
  - If $x \in \mathsf{dom}(m)$ and $x \in \chi_0(p_n) \Leftrightarrow m(x) \notin \chi_0(p_n')$, then $x \in \mathsf{dom}(\mu)$ and, necessarily, depending on whether $\mu(x) \in \chi_0(r_n)$ or $\mu(x) \notin \chi_0(r_n)$, we have either $x \in \chi_0(p_n) \Leftrightarrow \mu(x) \notin \chi_0(r_n)$ or $m(x) \in \chi_0(p_n') \Leftrightarrow \mu(m^{-1}(m(x))) = \mu(x) \notin \chi_0(r_n)$. Hence, (enabled) applies (as $p_n, p_n', r_n \in \mathcal{E}^{\mathcal{T}}$).
  - If $x_{p_k} \in \chi_0(p_n) \Leftrightarrow x_{p_k'} \notin \chi_0(p_n')$ for some $k \in \{1, \ldots, n\}$, we conclude with arguments similar to the previous case that (enabled) is also satisfied.
- Finally, if none of the above holds, $p_0 \#^m p_0'$ is due to (constants). We thus have

$$p_{n-1} \xrightarrow[(x,c)]{w_n} p_n \xrightarrow{w_{n+1}} \cdots \xrightarrow{w_\ell} p_\ell,$$

$$p_{n-1}' \xrightarrow[(x',c')]{w_n'} p_n', \text{ and}$$

$$r_{n-1} \xrightarrow[u]{v_n} r_n \xrightarrow{v_{n+1}} \cdots \xrightarrow{v_\ell} r_\ell$$

with $c \neq c'$ and $w_\ell = to[x]$. Finally, let

$$y = \begin{cases} \mu(x) & \text{if } x \in \mathsf{dom}(m) \\ x_{r_k} & \text{if } x = x_{p_k} \text{ for some } k \in \{1, \ldots, n\} \end{cases}$$

which means that

$$v_\ell = to[y] = \begin{cases} to[\mu(x)] & \text{if } x \in \mathsf{dom}(m) \\ to[x_{r_k}] & \text{if } x = x_{p_k} \text{ for some } k \in \{1, \ldots, n\}. \end{cases}$$

We argue that $u = (y, d)$ for some constant d that is distinct from either c or c'. Once we have this, (constants) applies and we obtain the desired result. We have three cases:

- If $w_n \in I$, it must be that $x = x_{p_n}$ as an input transition can only start a fresh timer in $\mathcal{T}$. Then, $v_n = w_n$ as $w_n \in I$, $w_\ell = to[x_{p_n}]$, and $y = x_{r_n}$. So, $v_\ell = to[x_{r_n}]$. Moreover, as the only transition that can start x_{r_n} for the first time is $r_{n-1} \xrightarrow{v_n} r_n$, we conclude that $u = (y, d) = (x_{r_n}, d)$.

- If $w_n = to[x]$ with $x \in \mathsf{dom}(m)$, then $x \in \mathsf{dom}(\mu)$, $w_n = w_\ell = to[x]$, and $v_n = v_\ell = to[y] = to[\mu(x)]$. Assume $u = \bot$, i.e., we do not restart y from r_{n-1} to r_n. In other words, y is not active in r_n. Recall that, in an observation tree, it is impossible to start again a timer that was previously active (as, for every timer z, there is a unique transition that can start z for the first time). So, y can not be active in $r_{\ell-1}$. But, then, v_ℓ can not be $to[y]$, which is a contradiction. Hence, $u = (y, d) = (\mu(x), d)$.

- If $w_n = to[x_{p_k}]$ with $k \in \{1, \ldots, n-1\}$, then $w_n = w_\ell = to[x_{p_k}]$ and $v_n = v_\ell = to[x_{r_k}]$. With arguments similar to the previous case, we conclude that $u = (y, d) = (x_{r_k}, d)$.

<div align="right">□</div>

E More details on the learning algorithm

In this section, we give further details on the ideas introduced in Sections 3.2 and 3.3. We first clarify the notion of *replaying* a run from Example 6 and state a useful property. We then introduce generalized MMTs and explain how to construct one from $\mathcal{T}$ in Appendix E.2. Finally, we prove Theorem 1 in Appendix E.3.

E.1 Replaying a run

Recall that *(C)* requires that, for every $r \in \mathcal{F}^{\mathcal{T}}$ and $(p, m) \in compat^{\mathcal{T}}(r)$, $|\chi^{\mathcal{T}}(p)| = |\chi^{\mathcal{T}}(r)|$ holds. In Example 6, we introduced the idea of replaying a run to ensure that *(C)* is satisfied: if p and r do not have the same number of active timers, we extend the tree by mimicking the run showing that some timer is active in p (i.e., a run starting from p and ending in the timeout of that timer) from r, using and extending some matching.

Let us first formalize this algorithm. Let $p_0, p_0' \in Q^{\mathcal{T}}$, $m : p_0 \leftrightarrow p_0'$ be a matching, and $w = i_1 \cdots i_n$ be a word such that $p_0 \xrightarrow{i_1} p_1 \xrightarrow{i_2} \cdots \xrightarrow{i_n} p_n \in runs(\mathcal{T})$. We provide a function $replay^m_{p_0 \xrightarrow{w} p_n}(p_0')$ that extends the tree by replaying the run $p_0 \xrightarrow{w} p_n$ from p_0' as much as possible, or we discover a new apartness pair $p_0 \#^m p_0'$, or we discover a new active timer. Intuitively, we replay the run transition by transition while performing symbolic wait queries in every reached state in order to determine the enabled timers (which extends $\mathcal{E}^{\mathcal{T}}$). This may modify the number of active timers of p_0', meaning that m may become non-maximal. As we are only interested in maximal matchings, we stop early. This may also induce a new apartness pair $p_0 \#^m p_0'$, and we also stop early (notice that this may already hold without adding any state in $\mathcal{T}$). If the number of active timers of p_0' remains unchanged and no new apartness pair is discovered, we consider

the next symbol i of w and try to replay it. Determining the next symbol i' to use in the run from p'_0 follows the same idea as for $read^m_{p_0 \xrightarrow{w} p_n}(p'_0)$. If $i \in I$, then $i' = i$ (recall that it is always possible to replay i as $\mathcal{M}$ is complete, since it is s-learnable). If $i = to[x]$, we have three cases:

- $x \in \mathsf{dom}(m)$, in which case $i' = to[m(x)]$;
- $x = x_{p_k}$ is a fresh timer, i.e., p_k appears on the run from p_0, in which case we consider the timer started on the corresponding transition from p'_0: $i' = to[x_{p'_k}]$;
- none of the previous case holds: $x \in \chi^{\mathcal{T}}(p_0) \setminus \mathsf{dom}(m)$ and we cannot replay i.

To avoid this last case, we consider the longest prefix v of w where each action i of v is an input or is such that $m(i)$ is defined or $i = to[x_{p_k}]$ for some state p_k.

Formally, assume that we already replayed $p_0 \xrightarrow{i_1} p_1 \xrightarrow{i_2} \cdots \xrightarrow{i_{j-1}} p_{j-1}$ and obtained the run $p'_0 \xrightarrow{i'_1} p'_1 \xrightarrow{i'_2} \cdots \xrightarrow{i'_{j-1}} p'_{j-1}$, and we try to replay i_j from p'_{j-1}. We extend the tree with a symbolic output query when $i_j \in I$ and a symbolic wait query in every case. If the wait query leads to a discovery of new active timers of p'_0, we stop and return ACTIVE. If we can already deduce $p_0 \#^m p'_0$ from the replayed part, we also stop and return APART. Since $\neg(p_0 \#^m p'_0)$ and by the output and wait queries, there must exist $p_{j-1} \xrightarrow{i'_j}$ such that

- $i'_j = i_j$ if $i_j \in I$,
- $i'_j = to[m(x)]$ if $i_j = to[x]$ ($m(x)$ is well-defined by the considered prefix v of w), or
- $i'_j = to[x_{p'_k}]$.

Indeed, if the timeout-transition is not defined, then $p_0 \#^m p'_0$ by (enabled). Hence, we continue the procedure with the next symbol of w. If we completely replayed w and did not discover any new timer or apartness pair, we return DONE. Otherwise, we perform one last wait query and check whether we obtain apartness (by the following lemma, we return ACTIVE otherwise).

We now give a lemma stating some properties of the replay function. Namely, when $replay^m_\pi(p_0)$ successfully replays the complete run, it follows that $read^m_\pi(p_0)$ is well-defined and yields a run in $\mathcal{T}$. From that, we can deduce that, when π ends with a transition reading the timeout of x and cannot be completely reproduced from p'_0 (that has less active timers than p_0), we either have a new apartness pair or a new timer in p'_0. These properties are enough to conclude that one can obtain *(C)* by successively replaying some runs, as the set of compatible states of any frontier state r will eventually only contain basis states with the same number of active timers as r.

Lemma 11. *Let $\pi = p_0 \xrightarrow{w} \; \in runs(\mathcal{T})$, $p'_0 \in Q^{\mathcal{T}}$, and $m : p_0 \leftrightarrow p'_0$ be a maximal matching:*

- *$replay^m_\pi(p'_0) = \mathrm{DONE}$ implies that $read^m_\pi(p'_0)$ is now a run of $\mathcal{T}$.*

– $replay_\pi^m(p_0')$ *returns* APART *or* ACTIVE *when* $|\chi^{\mathcal{T}}(p_0)| > |\chi^{\mathcal{T}}(p_0')|$ *and* w
ends with $to[x]$ *for some* $x \in \chi^{\mathcal{T}}(p_0) \setminus dom(m)$.

Proof. Observe that the second item follows immediately from the first, given
the fact that we process a proper prefix of w in that case. That is, it is sufficient
to show the first item.

Let $w = i_1 \cdots i_n$ and $\pi = p_0 \xrightarrow{i_1} p_1 \xrightarrow{i_2} \cdots \xrightarrow{i_n} p_n$. Towards a contradiction,
assume that $replay_\pi^m(p_0') = \text{DONE}$ but $read_\pi^m(p_0')$ is not a run of $\mathcal{T}$. Then, let
$\ell \in \{1, \ldots, n-1\}$ be the largest index such that

$$read^m_{p_0 \xrightarrow{i_1 \cdots i_\ell}}(p_0') = p_0' \xrightarrow{i_1'} p_1' \xrightarrow{i_2'} \cdots \xrightarrow{i_\ell'} p_\ell' \in runs(\mathcal{T}).$$

Hence, $p_0 \xrightarrow{i_1} p_1 \xrightarrow{i_2} \cdots \xrightarrow{i_\ell} p_\ell \xrightarrow{i_{\ell+1}} \in runs(\mathcal{T})$ and $read^m_{p_0 \xrightarrow{i_1 \cdots i_\ell \cdot i_{\ell+1}}}(p_0') =$
$p_0' \xrightarrow{i_1'} p_1' \xrightarrow{i_2'} \cdots \xrightarrow{i_\ell'} p_\ell' \xrightarrow{i_{\ell+1}'} \notin runs(\mathcal{T})$. First, if $i_{\ell+1} \in I$, then we must have
performed a symbolic output query in p_ℓ', i.e., $p_\ell' \xrightarrow{i_{\ell+1}'} \in runs(\mathcal{T})$. Second, if
$i_{\ell+1} = to[x_{p_k}]$ for some $k \in \{1, \ldots, \ell\}$, then we have that $p_0 \#^m p_0'$ by (enabled).
Likewise when $i_{\ell+1} = to[x]$ with $x \in dom(m)$.

Therefore, assume $i_{\ell+1}$ is the timeout of some timer in $\chi^{\mathcal{T}}(p_0) \notin dom(m)$.
Since $replay_\pi^m(p_0') = \text{DONE}$, we have that $\neg(p_0 \#^m p_0')$ and we did not discover
a new active timer in p_0'. Hence,

$$|\chi_0^{\mathcal{T}}(p_\ell)| = |\chi_0^{\mathcal{T}}(p_\ell')| \tag{3}$$

$$\forall y \in dom(m) : y \in \chi_0^{\mathcal{T}}(p_\ell) \Leftrightarrow m(y) \in \chi_0^{\mathcal{T}}(p_\ell'), \tag{4}$$

$$\forall k \in \{1, \ldots, \ell\} : x_{p_k} \in \chi_0^{\mathcal{T}}(p_\ell) \Leftrightarrow x_{p_k'} \in \chi_0^{\mathcal{T}}(p_\ell'), \tag{5}$$

As m is maximal, we deduce from (4) and (5) that all enabled timers in p_ℓ' have
their corresponding enabled timer in p_ℓ. However, x is an enabled timer in p_ℓ
that does not appear among those corresponding timers as $x \notin dom(m)$. This is
in contradiction with (3). We thus conclude that $replay_\pi^m(p_0') \neq \text{DONE}$. $\square$

E.2   Generalized MMTs and hypothesis construction

In this section, we introduce generalized MMTs (that allow timer renamings
alongside the transitions), and show that a symbolically equivalent MMT al-
ways exists. This MMT suffers a factorial blowup, in general. We then give the
construction of a generalized MMT from $\mathcal{T}$, and give an example where the con-
struction of an MMT as explained in Example 7 fails, as the equivalence relation
groups together timers that are known to be apart.

Generalized MMTs. In short, a generalized MMT is similar to an MMT,
except that the update of a transition $q \xrightarrow{i} q'$ is now a function instead of a
value in $(X \times \mathbb{N}^{>0}) \cup \{\bot\}$. For the gMMT to be well-formed, we request that the

domain of such a function is exactly the set of active timers of q'. Moreover, its range must be the set of active timers of q or a natural constant. That is, each timer x' of q' must either come from an active timer x of q (we rename x into x'), or be (re)started with a constant. We also require that at most one timer is started per transition, as in MMTs. Finally, if $i = to[x]$, we forbid to rename x into x', i.e., x' cannot be obtained from x: it must be the renaming of some other timer or be explicitly started by the transition. An example is given below.

Definition 8 (gMMT). *A generalized Mealy machine with timers (gMMT, for short) is a tuple $\mathcal{M} = (X, Q, q_0, \chi, \delta)$ where:*

- *X is a finite set of timers (we assume $X \cap \mathbb{N}^{>0} = \emptyset$),*
- *Q is a finite set of states, with $q_0 \in Q$ the initial state,*
- *$\chi : Q \to \mathcal{P}(X)$ is a total function that assigns a finite set of active timers to each state, and*
- *$\delta : Q \times A(\mathcal{M}) \rightharpoonup Q \times O \times (X \to (X \cup \mathbb{N}^{>0}))$ is a partial transition function that assigns a state-output-update triple to a state-action pair.*

We write $q \xrightarrow[\mathfrak{r}]{i/o} q'$ if $\delta(q, i) = (q', o, \mathfrak{r})$. We require the following:

- *In the initial state, no timer is active, i.e., $\chi(q_0) = \emptyset$.*
- *For any transition $q \xrightarrow[\mathfrak{r}]{} q'$, $\mathfrak{r}$ must be an injective function whose domain is exactly the set of active timers of q' (i.e., $\mathsf{dom}(\mathfrak{r}) = \chi$), and whose range is composed of timers that were active in q or constants from $\mathbb{N}^{>0}$ (i.e., $\mathsf{ran}(\mathfrak{r}) \subset \chi(q) \cup \mathbb{N}^{>0}$). Finally, there is at most one (re)started timer, i.e., there is at most one $x \in \mathsf{dom}(\mathfrak{r})$ such that $\mathfrak{r}(x) \in \mathbb{N}^{>0}$.*
- *For any transition $q \xrightarrow[\mathfrak{r}]{to[x]} q'$, it must be that x was active in q and x cannot be used as a value of $\mathfrak{r}$, i.e., $x \in \chi(q)$ and $x \notin \mathsf{ran}(\mathfrak{r})$.*

Observe that an MMT is in fact a gMMT where all renaming maps on transitions coincide with the identity function (except for those mapping to an integer, which are regular updates).

We now adapt the timed semantics of the model via the following rules. Again, they are similar to the rules for MMTs, except that we use $\mathfrak{r}$ to rename and start timers. Let $(q, \kappa), (q', \kappa')$ be two configurations of a gMMT:

$$\frac{\forall x \colon \kappa(x) \geq d}{(q, \kappa) \xrightarrow{d} (q, \kappa - d)}$$

$$\frac{q \xrightarrow[\mathfrak{r}]{i/o} q', \quad i = to[x] \Rightarrow \kappa(x) = 0, \quad \forall x \in \chi(q') \colon \kappa'(x) = \begin{cases} \mathfrak{r}(x) & \text{if } \mathfrak{r}(x) \in \mathbb{N}^{>0} \\ \kappa(\mathfrak{r}(x)) & \text{otherwise} \end{cases}}{(q, \kappa) \xrightarrow[\mathfrak{r}]{i/o} (q', \kappa')}$$

We immediately obtain the definitions of enabled timers and complete gMMT. Moreover, it is clear that the notion of timed equivalence from Section 2.2 can be applied to two complete gMMTs, or a gMMT and an MMT, both complete.
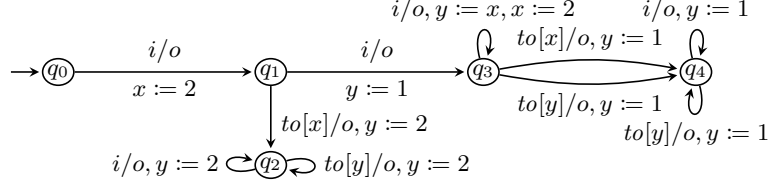
Fig. 8: A generalized MMT with $\chi(q_0) = \emptyset$, $\chi(q_1) = \{x\}$, $\chi(q_2) = \chi(q_4) = \{y\}$, and $\chi(q_3) = \{x, y\}$.

Example 10. Let $\mathcal{M}$ be the gMMT of Figure 8 with timers $X = \{x, y\}$. Update functions are shown along each transition. For instance, x is started to 2 by the transition from q_0 to q_1, while the self-loop over q_3 renames x into y (i.e., y copies the current value of x) and then restarts x to 2. Let us illustrate this with the following timed run:

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow{i} (q_1, x = 2) \xrightarrow{0.5} (q_1, x = 1.5) \xrightarrow{i} (q_3, x = 1.5, y = 1)$$
$$\xrightarrow{1} (q_3, x = 0.5, y = 0) \xrightarrow{i} (q_3, x = 2, y = 0.5) \xrightarrow{0.5} (q_4, y = 1) \xrightarrow{0} (q_4, y = 1).$$

Observe that y takes the value of x when taking the i-loop of q_3.

We also highlight that some of the timeout transitions restart a timer that is not the one timing out, as illustrated by the timed run

$$(q_0, \emptyset) \xrightarrow{1} (q_0, \emptyset) \xrightarrow{i} (q_1, x = 2) \xrightarrow{2} (q_1, x = 0) \xrightarrow{to[x]} (q_2, y = 2) \xrightarrow{0} (q_2, y = 2).$$

It is not hard to see that we have the following enabled timers per state: $\chi_0(q_0) = \emptyset$, $\chi_0(q_1) = \{x\}$, $\chi_0(q_2) = \chi_0(q_4) = \{y\}$, and $\chi_0(q_3) = \{x, y\}$. From there, we conclude that $\mathcal{M}$ is complete.

Let us move towards providing a definition of symbolic equivalence (Definition 2) between a gMMT and an MMT. First, we adapt the notion of x-spanning runs. Recall that a run π of an MMT is said x-spanning if $\pi = p_0 \xrightarrow[u_1]{i_1} p_1 \xrightarrow[u_2]{i_2} \cdots \xrightarrow{i_n} p_n$ with

- $u_1 = (x, c)$ for some $c \in \mathbb{N}^{>0}$,
- $u_j \neq (x, c')$ for every $j \in \{2, \ldots, n-1\}$ and $c' \in \mathbb{N}^{>0}$,
- $x \in \chi(p_j)$ for all $j \in \{2, \ldots, n-1\}$, and
- $i_n = to[x]$.

The adaptation to gMMTs is straightforward: the first transition must start a timer x, each update function renames the timer (in a way, x remains active but under a different name), and the final transition reads the timeout of the timer corresponding to x. We say that a run π of a gMMT is *spanning* if

$$\pi = p_0 \xrightarrow[\mathfrak{r}_1]{i_1} p_1 \xrightarrow[\mathfrak{r}_2]{i_2} \cdots \xrightarrow{i_n} p_n$$

and there exist timers $x_1, \ldots, x_n$ such that

- $\mathfrak{r}_1(x_1) = c$ for some $c \in \mathbb{N}^{>0}$,
- $\mathfrak{r}_j(x_j) = x_{j-1}$ for every $j \in \{2, \ldots, n-1\}$ (this implies that $x_j \in \chi(p_j)$), and
- $i_n = to[x_{n-1}]$.

Observe that the notion of symbolic words (Section 2.2) still holds using this definition of spanning runs.

We can thus obtain a definition of symbolic equivalence between a complete gMMT and a complete MMT. In short, we impose the same constraints as in Definition 2:

- a run reading a symbolic word exists in the gMMT if and only if one exists in the MMT,
- if they both exist, we must see the same outputs and for transitions starting a timer that eventually times out during the run (i.e., the sub-run is spanning), we must have the same constants.

Definition 9 (Symbolic equivalence between gMMT and MMT). *Let $\mathcal{M}$ be a complete gMMT and $\mathcal{N}$ be a complete MMT. We say that $\mathcal{M}$ and $\mathcal{N}$ are* symbolically equivalent, *also noted $\mathcal{M} \stackrel{sym}{\approx} \mathcal{N}$, if for every symbolic word $\mathtt{w} = \mathtt{i_1} \cdots \mathtt{i_n}$ over $I \cup TO[\mathbb{N}^{>0}]$:*

- $q_0^{\mathcal{M}} \xrightarrow[\mathfrak{r}_1]{\mathtt{i_1}/o_1} q_1 \cdots \xrightarrow[\mathfrak{r}_n]{\mathtt{i_n}/o_n} q_n$ *is a feasible run in $\mathcal{M}$ if and only if $q_0^{\mathcal{N}} \xrightarrow[u'_1]{\mathtt{i_1}/o'_1}$
  $q'_1 \cdots \xrightarrow[u'_n]{\mathtt{i_n}/o'_n} q'_n$ *is a feasible run in $\mathcal{N}$.*
- *Moreover,*
    - *$o_j = o'_j$ for all $j \in \{1, \ldots, n\}$, and*
    - *$q_{j-1} \xrightarrow{\mathtt{i_j} \cdots \mathtt{i_k}} q_k$ is spanning implies that there is timer x such that $\mathfrak{r}_j(x) = c$, $u'_j = (x', c')$, and $c = c'$.*

We then obtain that $\mathcal{M} \stackrel{sym}{\approx} \mathcal{N}$ implies that $\mathcal{M} \stackrel{time}{\approx} \mathcal{N}$, with arguments similar to those presented in Appendix A.

Existence of a symbolically equivalent MMT. Let $\mathcal{M}$ be a complete gMMT. We give a construction of a complete MMT $\mathcal{N}$ such that $\mathcal{M} \stackrel{sym}{\approx} \mathcal{N}$. Intuitively, we rename the timers of $\mathcal{M}$, on the fly, into the timers of $\mathcal{N}$ and keep track in the states of $\mathcal{N}$ of the current renaming. Due to the fact that each transition of $\mathcal{M}$ can freely rename timers, it is possible that x is mapped to x_j in a state of $\mathcal{N}$, but mapped to x_k (with $k \neq j$) in some other state. That is, we sometimes need to split states of $\mathcal{M}$ into multiple states in $\mathcal{N}$, accordingly to the update functions. An example is given below.

Formally, we define $\mathcal{N} = (X^{\mathcal{N}}, Q^{\mathcal{N}}, q_0^{\mathcal{N}}, \chi^{\mathcal{N}}, \delta^{\mathcal{N}})$ with

- $X^{\mathcal{N}} = \{x_1, \ldots, x_n\}$ with $n = \max_{q \in Q^{\mathcal{M}}} |\chi^{\mathcal{M}}(q)|$.
- $Q^{\mathcal{N}} = \{(q, \mu) \in Q^{\mathcal{M}} \times (\chi^{\mathcal{M}}(q) \leftrightarrow X^{\mathcal{N}}) \mid |\mathsf{ran}(\mu)| = |\chi^{\mathcal{M}}(q)|\}$. The idea is that μ dictates how to rename a timer from $\mathcal{M}$ into a timer of $\mathcal{N}$, for this specific state. As said above, the renaming may change transition by transition.

- $q_0^{\mathcal{N}} = (q_0^{\mathcal{M}}, \emptyset)$.
- $\chi^{\mathcal{M}}((q, \mu)) = \mathsf{ran}(\mu)$ for all $(q, \mu) \in Q^{\mathcal{N}}$. Then, $|\chi^{\mathcal{N}}((q, \mu))| = |\chi^{\mathcal{M}}(q)|$.
- The function $\delta^{\mathcal{N}} : Q^{\mathcal{N}} \times A(\mathcal{N}) \to Q^{\mathcal{N}} \times O \times U(\mathcal{N})$ is defined as follows. Let $q \xrightarrow[\mathfrak{r}]{i/o} q'$ be a run of $\mathcal{M}$ and $(q, \mu) \in Q^{\mathcal{N}}$.
  - If $i \in I$, we have two different cases depending on whether $\mathfrak{r}$ (re)starts a fresh timer or does not (re)start anything. That is, let $\delta^{\mathcal{N}}((q, \mu), i) = ((q', \mu'), o, u)$ with u and μ' defined as follows.
    * If $\mathfrak{r}(x) = c \in \mathbb{N}^{>0}$ for a timer x, i.e., the transition of $\mathcal{M}$ (re)starts a fresh timer, then, in $\mathcal{N}$, we want to start a timer that is not already tied to some timer. Let $\nu = \mu \circ (\mathfrak{r} \setminus \{(x, c)\})$, i.e., the matching telling us how to rename every timer, except x, after taking the transition. As $|X^{\mathcal{N}}| = \max_{p \in Q^{\mathcal{M}}} |\chi^{\mathcal{M}}(q)| = \max_{(p, \nu) \in Q^{\mathcal{N}}} |\chi^{\mathcal{N}}((q, \nu))|$, it follows that $|X^{\mathcal{N}}| > |\mathsf{ran}(\nu)|$. Hence, there exists a timer $x_j \in X^{\mathcal{N}}$ such that $x_j \notin \mathsf{ran}(\nu)$. We then say that x is mapped to x_j and follow ν for the other timers. That is,

      $$u = (x_j, c) \qquad \text{and} \qquad \mu' = \nu \cup \{(x, x_j)\}.$$

    * If $\mathfrak{r}(x) \notin \mathbb{N}^{>0}$ for any timer x, i.e., the transition does not (re)start anything, then, in $\mathcal{N}$, we also do not restart anything. Hence,

      $$u = \bot \qquad \text{and} \qquad \mu' = \mu \circ \mathfrak{r}.$$

  - If $i = to[x]$, we have two cases depending on whether the transition start a timer, or not. That is, we define $\delta^{\mathcal{N}}((q, \mu), to[\mu(x)]) = ((q', \mu'), o, u)$ with u and μ' defined as follows.
    * If $\mathfrak{r}(y) = c \in \mathbb{N}^{>0}$ for some timer y, then, in $\mathcal{N}$, we want to restart x. That is, we restart the timer that times out. Again, the remaining timers simply follow $\mathfrak{r}$. Hence,

      $$u = (\mu(x), c) \quad \text{and} \quad \mu' = (\mu \circ (\mathfrak{r} \setminus \{(y, c)\})) \cup \{(y, \mu(x))\}.$$

    * If $\mathfrak{r}(y) \notin \mathbb{N}^{>0}$ for any timer y, then, in $\mathcal{N}$, we do not restart anything. Hence,

      $$u = \bot \qquad \text{and} \qquad \mu' = \mu \circ \mathfrak{r}.$$

In order to obtain a deterministic procedure, let us assume that the fresh timer x_j is picked with the smallest possible j. Figure 9 gives the MMT constructed from the gMMT of Figure 8.

It should be clear that $\mathcal{M}$ is complete since $\mathcal{N}$ is complete (and one can obtain $\mathcal{M}$ back from $\mathcal{N}$ as a sort of homomorphic image of $\mathcal{N}$).

Lemma 12. *Let $\mathcal{M}$ be a complete gMMT and $\mathcal{N}$ be the MMT constructed as explained above. Then, $\mathcal{N}$ is complete and its number of states is in $\mathcal{O}\left(n! \cdot |Q^{\mathcal{M}}|\right)$ with $n = \max_{q \in Q^{\mathcal{M}}} |\chi^{\mathcal{M}}(q)|$.*
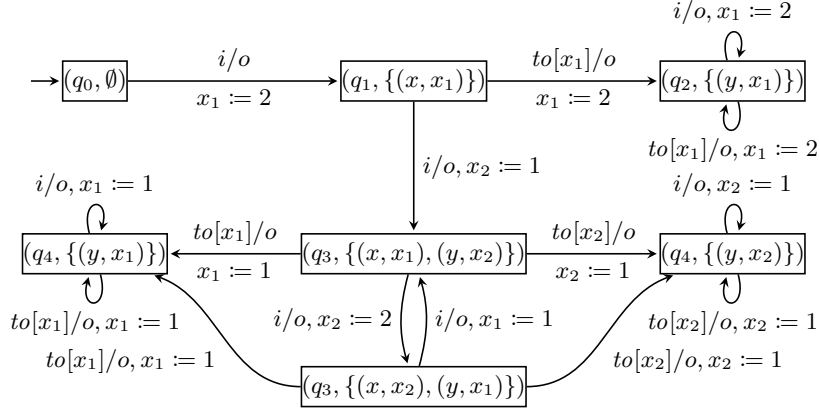
Fig. 9: The MMT obtained from the gMMT of Figure 8.

The following lemma highlights the relation between a transition of $\mathcal{M}$ and a corresponding transition in $\mathcal{N}$. It holds by construction of $\mathcal{N}$.

Lemma 13. *Let $(q, \mu) \in Q^{\mathcal{N}}$ and $q \xrightarrow{i/o}_{\mathfrak{r}} q' \in runs(\mathcal{M})$. Then, we have the transition $(q, \mu) \xrightarrow{i'/o'}_{u} (q', \mu') \in runs(\mathcal{N})$ with $o = o'$, and i' and u as follows.*

  - *If $i \in I$, then $i' = i$ and*

$$u = \begin{cases} (x, c) & \text{for some } x \notin ran(\mu \circ \mathfrak{r}), \text{ if } \mathfrak{r}(y) = c \in \mathbb{N}^{>0} \text{ for some } y \\ \bot & \text{if for all } x, \mathfrak{r}(x) \notin \mathbb{N}^{>0}. \end{cases}$$

  - *If $\mathtt{i_j} = to[x]$, then $i' = to[\mu(x)]$ and*

$$u_j = \begin{cases} (\mu(x), c) & \text{if there exists a timer } y \text{ such that } \mathfrak{r}(y) = c \\ \bot & \text{if for all } x, \mathfrak{r}_j(x) \notin \mathbb{N}^{>0}. \end{cases}$$

Then, by applying this lemma over and over on each transition along a run, we obtain that we always see the same outputs and the same updates in both machines.

Corollary 4. *For every symbolic word $\mathtt{w} = \mathtt{i_1} \cdots \mathtt{i_n}$, we have*

$$q_0^{\mathcal{M}} \xrightarrow{\mathtt{i_1}/o_1}_{\mathfrak{r}_1} q_1 \xrightarrow{\mathtt{i_2}/o_2}_{\mathfrak{r}_2} \cdots \xrightarrow{\mathtt{i_n}/o_n}_{\mathfrak{r}_n} q_n \in runs(\mathcal{M})$$

$$\Leftrightarrow (q_0^{\mathcal{M}}, \emptyset) \xrightarrow{\mathtt{i_1}/o_1}_{u_1} (q_1, \mu_1) \xrightarrow{\mathtt{i_2}/o_2}_{u_2} \cdots \xrightarrow{\mathtt{i_n}/o_n}_{u_n} (q_n, \mu_n) \in runs(\mathcal{N})$$

with u_j defined as follows for every j:

– *If* $\mathtt{i_j} \in I$, *then*

$$u_j = \begin{cases} (x_k, c) & \text{for some } x_k \notin \mathsf{ran}(\mu_{j-1} \circ \mathfrak{r}_j), \text{ if there exists } x \text{ such that} \\ & \mathfrak{r}_j(x) = c \in \mathbb{N}^{>0} \\ \bot & \text{if for all } x, \ \mathfrak{r}_j(x) \notin \mathbb{N}^{>0}. \end{cases}$$

– *If* $\mathtt{i_j} = to[k]$, *then*

$$u_j = \begin{cases} (x, c) & \text{if there exists } x \text{ such that } \mathfrak{r}_j(x) = c \in \mathbb{N}^{>0} \text{ and} \\ & (q_{k-1}, \mu_{k-1}) \xrightarrow[(x,c')]{i_k} \\ \bot & \text{if for all } x, \ \mathfrak{r}_j(x) \notin \mathbb{N}^{>0}. \end{cases}$$

This directly implies that if a run is feasible in $\mathcal{M}$, the corresponding run is also feasible in $\mathcal{N}$, and vice-versa.

Corollary 5. *For any symbolic word* $\mathtt{w}$, $q_0^{\mathcal{M}} \xrightarrow{\mathtt{w}} \ \in runs(\mathcal{M})$ *is feasible if and only if* $q_0^{\mathcal{N}} \xrightarrow{\mathtt{w}} \ \in runs(\mathcal{N})$ *is feasible.*

This is enough to obtain the desired result: $\mathcal{M} \overset{\mathrm{sym}}{\approx} \mathcal{N}$, as any run in one can be reproduced in the other, and we see the same outputs and updates (for spanning sub-runs) along these runs.

Corollary 6. $\mathcal{M} \overset{sym}{\approx} \mathcal{N}$.

Construction of a gMMT hypothesis. Let us now describe how a complete gMMT $\mathcal{H}$ is constructed from $\mathcal{T}$. We assume that *(C)* holds, i.e.,

– $compat^{\mathcal{T}}(r) \neq \emptyset$ for every frontier state r, and
– $|\chi^{\mathcal{T}}(r)| = |\chi^{\mathcal{T}}(p)|$ for each $r \in \mathcal{F}^{\mathcal{T}}$ and $(p, m) \in compat^{\mathcal{T}}(r)$.

Hence, m is bijective. The idea is to use the set of basis states as the states of $\mathcal{H}$, with exactly the same active timers per state. Then, for any transition $q \xrightarrow[u]{i/o} q' \in runs(\mathcal{T})$ with $q, q' \in \mathcal{B}^{\mathcal{T}}$, we do not rename anything in $\mathcal{H}$. If $u = (x, c)$, then the update function of $\mathcal{H}$ also (re)starts x to c. Finally, for a transition $q \xrightarrow[u]{i/o} r \in runs(\mathcal{T})$ with $r \in \mathcal{B}^{\mathcal{T}}$, we arbitrarily select a pair $(p, m) \in compat^{\mathcal{T}}(r)$ and define a transition $q \xrightarrow[\mathfrak{r}]{i/o} p$ where $\mathfrak{r}$ renames every timer according to m. In other words, the only functions that actually rename timers come from folding the tree, i.e., when we exit the basis in $\mathcal{T}$.

Definition 10 (Generalized MMT hypothesis). *We define the gMMT* $\mathcal{H} = (X^{\mathcal{H}}, Q^{\mathcal{H}}, q_0^{\mathcal{H}}, \chi^{\mathcal{H}}, \delta^{\mathcal{H}})$ *where:*

– $X^{\mathcal{H}} = \bigcup_{q \in \mathcal{B}^{\mathcal{T}}} \chi^{\mathcal{T}}(q)$,
– $Q^{\mathcal{H}} = \mathcal{B}^{\mathcal{T}}$, *with* $q_0^{\mathcal{H}} = q_0^{\mathcal{T}}$,
– $\chi^{\mathcal{H}}(q) = \chi^{\mathcal{T}}(q)$ *for each* $q \in \mathcal{B}^{\mathcal{T}}$, *and*

– $\delta^{\mathcal{H}}$ *is constructed as follows. Let* $q \xrightarrow[u]{i/o} q'$ *be a transition in* $\mathcal{T}$ *with* $q \in \mathcal{B}^{\mathcal{T}}$.
  We have four cases:
  - *If* $q' \in \mathcal{B}^{\mathcal{T}}$ *(i.e., the transition remains within the basis) and* $u = \bot$, *then we define* $\delta^{\mathcal{H}}(q, i) = (q', o, \mathfrak{r})$ *with, for all* $x \in \chi^{\mathcal{T}}(q')$, $\mathfrak{r}(x) = x$.
  - *If* $q' \in \mathcal{B}^{\mathcal{T}}$ *and* $u = (y, c)$, *then we define* $\delta^{\mathcal{H}}(q, i) = (q', o, \mathfrak{r})$ *with* $\mathfrak{r}(y) = c$ *and, for all* $x \in \chi^{\mathcal{T}}(q') \setminus \{y\}$, $\mathfrak{r}(x) = x$.
  - *If* $q' \in \mathcal{F}^{\mathcal{T}}$ *(i.e., the transition leaves the basis) and* $u = \bot$, *then we select an arbitrary* $(p, m) \in compat^{\mathcal{T}}(r)$ *and define* $\delta^{\mathcal{H}}(q, i) = (p, o, \mathfrak{r})$ *with, for all* $x \in \chi^{\mathcal{T}}(q')$, $\mathfrak{r}(m^{-1}(x)) = x$.
  - *If* $q' \in \mathcal{F}^{\mathcal{T}}$ *and* $u = (y, c)$, *then we select an arbitrary* $(p, m) \in compat^{\mathcal{T}}(r)$ *and define* $\delta^{\mathcal{H}}(q, i) = (p, o, \mathfrak{r})$ *with* $\mathfrak{r}(m^{-1}(y)) = c$ *and, for all* $x \in \chi^{\mathcal{T}}(q') \setminus \{y\}$, $\mathfrak{r}(m^{-1}(x)) = x$

It is not hard to see that $\mathcal{H}$ is well-formed and complete, as it is constructed from $\mathcal{T}$. Indeed, recall that $q \xrightarrow{i} \in runs(\mathcal{T})$ is defined for each $q \in \mathcal{B}^{\mathcal{T}}$ if and only if $i \in I \cup TO[\chi_0^{\mathcal{T}}(q)]$, i.e., the basis is complete.

Example 11. Let $\mathcal{T}$ be the observation tree of Figure 3. We can observe that

$$compat^{\mathcal{T}_6}(t_2) = \{(t_1, x_1 \mapsto x_1)\} \qquad compat^{\mathcal{T}_6}(t_{10}) = \{(t_0, \emptyset)\}$$
$$compat^{\mathcal{T}_6}(t_5) = \{(t_6, x_6 \mapsto x_1, x_3 \mapsto x_3)\} \qquad compat^{\mathcal{T}_6}(t_{12}) = \{(t_0, \emptyset)\}$$
$$compat^{\mathcal{T}_6}(t_{11}) = \{(t_6, x_6 \mapsto x_{11}, x_3 \mapsto x_3)\} \quad compat^{\mathcal{T}_6}(t_{15}) = \{(t_9, x_3 \mapsto x_3)\}.$$

We thus construct a gMMT $\mathcal{H}$ as follows.

– The states of $\mathcal{H}$ are t_0, t_1, t_3, t_6, and t_9.
– The i-transition from t_0 to $t_1 \in \mathcal{B}^{\mathcal{T}}$ starts the timer x_1 to 2.
– The i-transition from t_1 to $t_3 \in \mathcal{B}^{\mathcal{T}}$ keeps x_1 as-is and starts a new timer x_3.
– The i-transition from t_3 to $t_6 \in \mathcal{B}^{\mathcal{T}}$ stops x_1 and starts x_6. The timer x_3 is unchanged. Observe that, so far, we followed exactly the transitions defined within the basis of $\mathcal{T}$.
– We consider the $to[x_1]$-transition from t_3 to t_5 in $\mathcal{T}$. As $t_5 \in \mathcal{F}^{\mathcal{T}}$, we select a pair (p, m) in $compat^{\mathcal{T}}(t_5)$. Here, the only possibility is $(t_6, x_6 \mapsto x_1, x_3 \mapsto x_3)$. So, we define the renaming function $\mathfrak{r}_{5 \mapsto 6}$ such that $\mathfrak{r}_{5 \mapsto 6}(x_6) = 2$ and $\mathfrak{r}_{5 \mapsto 6}(x_3) = x_3$. Hence, we have the transition $t_3 \xrightarrow[\mathfrak{r}_{5 \mapsto 6}]{to[x_1]/o} t_6$.

And so on for the remaining transitions. The resulting gMMT is given in Figure 10. A more complex example is provided in the next section.

Finally, while one can then convert the gMMT hypothesis $\mathcal{H}$ into an MMT hypothesis, it is not required. Indeed, in order to avoid the factorial blowup, one can simply give $\mathcal{H}$ to the teacher who then has to check whether $\mathcal{H}$ and its hidden MMT $\mathcal{M}$ are symbolically equivalent. For instance, the teacher may construct the zone gMMT of $\mathcal{H}$ and the zone MMT of $\mathcal{M}$ (see Appendix B) and then check the equivalence between those models. This does not necessitate to construct an MMT.
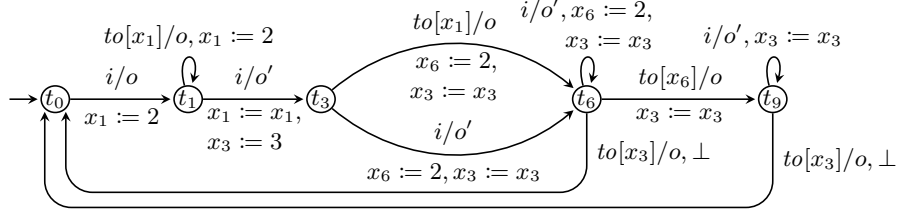
Fig. 10: A gMMT constructed from the observation tree of Figure 3.

Example of a case where gMMTs are required. Finally, we give an example of an observation tree from which the construction of $\equiv$ (as explained earlier) fails. That is, we obtain $x \equiv y$ but $x \not\Vdash y$. Let $\mathcal{M}$ be the MMT of Figure 11. For simplicity, we omit all outputs in this section.

Observe that replacing the transition $q_0 \xrightarrow[(y,1)]{j} q_4$ by $q_0 \xrightarrow[(x,1)]{j} q_3$ would yield an MMT symbolically equivalent to $\mathcal{M}$. Indeed, both q_3 and q_4 have the same behavior, up to a renaming of the timer. So, the j-transition from q_0 can freely go to q_3 or q_4, under the condition that it starts respectively x or y. Here, we fix that it goes to q_4 and starts y. However, the learning algorithm may construct a hypothesis where it instead goes to q_3. In short, this uncertainty will lead us to an invalid $\equiv$.

Let $\mathcal{T}$ be the observation tree of Figure 11. One can check that $\mathcal{T}$ is an observation tree for $\mathcal{M}$, i.e., there exists a functional simulation $\langle f, g \rangle : \mathcal{T} \to \mathcal{M}$. We have the following compatible sets:

$$compat^{\mathcal{T}}(t_2) = compat^{\mathcal{T}}(t_{14}) = \{(t_1, x_1 \mapsto x_1)\}$$
$$compat^{\mathcal{T}}(t_4) = compat^{\mathcal{T}}(t_5) = compat^{\mathcal{T}}(t_6) = \{(t_3, x_2 \mapsto x_2)\}$$
$$compat^{\mathcal{T}}(t_{12}) = \{(t_3, x_2 \mapsto x_1)\}$$
$$compat^{\mathcal{T}}(t_{13}) = \{(t_3, x_2 \mapsto x_{11})\}$$
$$compat^{\mathcal{T}}(t_{18}) = \{(t_{11}, x_1 \mapsto x_1, x_{11} \mapsto x_{11})\}$$

By constructing the equivalence relation $\equiv \ \subseteq \{x_1, x_2, x_{11}\} \times \{x_1, x_2, x_{11}\}$, we obtain that

$$x_1 \equiv x_2 \qquad \text{due to } (t_3, x_2 \mapsto x_1) \in compat^{\mathcal{T}}(t_{12}), \text{ and}$$
$$x_2 \equiv x_{11} \qquad \text{due to } (t_3, x_2 \mapsto x_{11}) \in compat^{\mathcal{T}}(t_{13}).$$

So, $x_1 \equiv x_{11}$. However, notice that $x_1 \not\Vdash x_{11}$, as both timers are active in t_{11}. Since that relation is the only possibility, we conclude that it is not always possible to construct a relation that does not put together two apart timers.

E.3   Proof of Theorem 1

Let us now show Theorem 1, which gives the termination and complexity of $L_{\text{MMT}}^{\#}$. Before that, we introduce an optimization of our learning algorithm
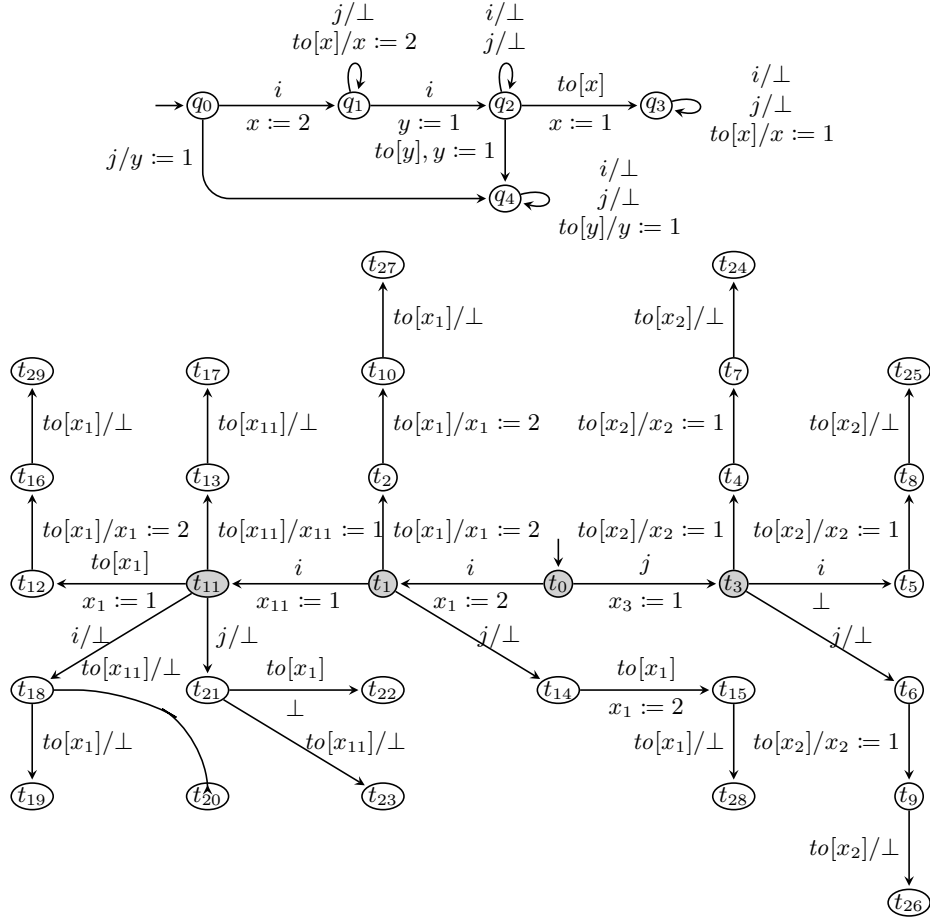
Fig. 11: An MMT with $\chi(q_0) = \emptyset, \chi(q_1) = \chi(q_3) = \{x\}, \chi(q_2) = \{x, y\}, \chi(q_4) = \{y\}$, and an observation tree in which basis states are highlighted in gray. For simplicity, the output o of each transition is omitted.
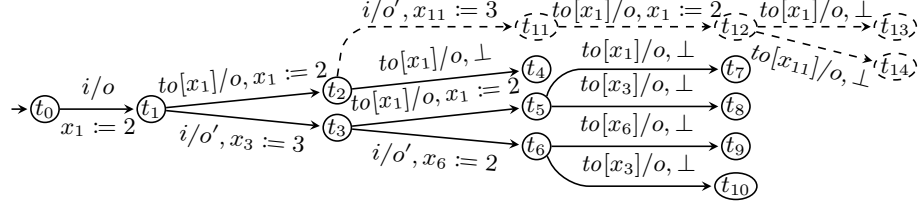
Fig. 12: Extension of the observation tree of Figure 2 obtained by calling $replay_\pi^{x_1 \mapsto x_1}(t_2)$, where $\pi = t_1 \xrightarrow{i \cdot to[x_1] \cdot to[x_3]}$. New states and transitions are highlighted with dashed lines.

that reduces the size of each compatible set, by leveraging weak co-transitivity (Lemma 10). In turn, this diminishes the number of hypotheses that can be constructed and, thus, helps the learner to converge towards the target MMT while reducing the number of equivalence queries needed.

Let $r \in \mathcal{F}^\mathcal{T}$ and assume we have $(p, \mu) \in compat^\mathcal{T}(r)$ and $(p', \mu') \in compat^\mathcal{T}(r)$ with $p \neq p'$ and maximal matchings $\mu : p \leftrightarrow r$ and $\mu' : p' \leftrightarrow r$. These matchings are necessarily valid by definition of $compat^\mathcal{T}$. We also assume that $|\chi^\mathcal{T}(p)| = |\chi^\mathcal{T}(r)| = |\chi^\mathcal{T}(p')|$ (by applying the above idea). As $p, p' \in \mathcal{B}^\mathcal{T}$, it must be that $p \#^m p'$ for any maximal matching $m : p \leftrightarrow p'$. In particular, take $m : p \leftrightarrow p'$ such that $m = \mu'^{-1} \circ \mu$. Notice that $\mathsf{dom}(m) \subseteq \mathsf{dom}(\mu)$ and $\mu' = \mu \circ m^{-1}$ (see Figure 7 for a visualization). It always exists and is unique as the three states have the same number of active timers. There are two cases: either any witness $w \vdash p \#^m p'$ is such that the apartness is structural, in which case we cannot apply Lemma 10, or there is a witness $w \vdash p \#^m p'$ where the apartness is behavioral. In that case, let also w^x be as described in Lemma 10. We then replay the run $p \xrightarrow{w \cdot w^x}$ from r using μ. We have three cases:

- $replay_{p \xrightarrow{w \cdot w^x}}^\mu (r) = \text{APART}$, meaning that $p \#^\mu r$. Then, (p, μ) is no longer in $compat^\mathcal{T}(r)$.
- $replay_{p \xrightarrow{w \cdot w^x}}^\mu (r) = \text{ACTIVE}$, in which case we discovered a new active timer in r. Hence, we now have that $|\chi^\mathcal{T}(p)| \neq |\chi^\mathcal{T}(r)|$ and we can reapply the idea of replaying runs showcasing timeouts (see Example 6) to obtain the equality again, or that p and r are not compatible anymore.
- $replay_{p \xrightarrow{w \cdot w^x}}^\mu (r) = \text{DONE}$, meaning that we could fully replay $p \xrightarrow{w \cdot w^x}$ and thus did not obtain $p \#^\mu r$. By Lemma 10, it follows that $p' \#^{\mu'} r$.

Hence, it is sufficient to call $replay_{p \xrightarrow{w \cdot w^x}}^\mu (r)$ when $w \vdash p \#^{\mu'^{-1} \circ \mu} p'$ is behavioral.

Unlike in [32], we cannot always obtain $|compat^\mathcal{T}(r)| = 1$, as Lemma 10 cannot be applied when the considered apartness pairs are structural.

Example 12. Let the MMT of Figure 1 be the MMT of the teacher and $\mathcal{T}$ be the observation tree of Figure 2. As explained in Example 6, we have $compat^\mathcal{T}(t_2) =$

$\{(t_1, x_1 \mapsto x_1), (t_3, x_1 \mapsto x_1)\}$. Let us extend the tree in order to apply weak co-transitivity to deduce that $t_1 \#^{x_1 \mapsto x_1} t_2$ or $t_1 \#^{x_1 \mapsto x_1} t_3$. We have that $i \vdash t_1 \#^{x_1 \mapsto x_1} t_3$ due to (constants). Hence, we replay the run $\pi = t_1 \xrightarrow{i \cdot to[x_1] \cdot to[x_3]}$ from t_2 using the matching $x_1 \mapsto x_1$ (i.e., we have $w^x = to[x_1] \cdot to[x_3]$). That is, we call $replay_\pi^{x_1 \mapsto x_1}(t_2)$. The resulting tree is given in Figure 12. Recall that the function returned DONE. So, $\neg(t_1 \#^{m_1 \mapsto x_1} t_2)$. By Lemma 10, it must be that $t_3 \#^{x_1 \mapsto x_1} t_2$. It is indeed the case as $i \vdash t_3 \#^{x_1 \mapsto x_1} t_2$ by (constants). Hence, we now have $compat^\mathcal{T}(t_2) = \{(t_1, x_1 \mapsto x_1)\}$.

Using this idea, we then add a new step inside the refinement loop, taking place after **Active timers**:

WCT   where **WCT** stands for Weak Co-Transitivity. As explained above, we minimize each compatible set by extending the tree to leverage Lemma 10 as much as possible.

We require several intermediate results. First, we argue that the refinement loop of our algorithm eventually terminates, i.e., a hypothesis is eventually constructed. Under the assumption that the basis is finite, observe that the frontier is then finite (as $I \cup TO[\cup_{p \in \mathcal{B}^\mathcal{T}} \chi^\mathcal{T}(p)]$ is finite). Hence, we can apply **Completion**, **Active timers**, **WCT** only a finite number of times. It is thus sufficient to show that the basis cannot grow forever, i.e., that **Promotion** is applied a finite number of times, which implies that **Seismic** is also applied a finite number of times. The next lemma states an upper bound over the number of basis states. In short, if this does not hold, one can construct a matching $m_g : p \leftrightarrow p'$ such that $g(x) = g(m(x))$ for all $x \in \mathsf{dom}(m_g)$ for two states p, p' such that $f(p) = f(p')$. By the contrapositive of Theorem 3, we conclude that either p or p' cannot be in the basis.

Lemma 14. $|\mathcal{B}^\mathcal{T}| \leq |Q^\mathcal{M}| \cdot 2^{|X^\mathcal{M}|}$.

Proof. In order to distinguish the theoretical definition of $\mathcal{B}^\mathcal{T}$ and its computation, let us denote by B the basis as computed in the refinement loop of $L_{\mathrm{MMT}}^\#$. We highlight that B may not always satisfy the definition of $\mathcal{B}^\mathcal{T}$ *during* the refinement loop. Indeed, as explained in Section 3.3, when a new active timer is found in a basis state, we may have $\neg(p \#^m p')$ for some $p \neq p' \in \mathcal{B}^\mathcal{T}$ and maximal matching $m : p \leftrightarrow p'$. We thus need to perform **Seismic** and recompute B. Below, we will establish that $B \leq |Q^\mathcal{M}| \cdot 2^{|X^\mathcal{M}|}$. The same (or even simpler) arguments yield the bound for $\mathcal{B}^\mathcal{T}$.

Let us assume we already treated the pending **Seismic** (if there is one), i.e.,

$$\forall p, p' \in B : p \neq p' \Rightarrow p \#^m p' \text{ for all maximal matchings } m : p \leftrightarrow p' \quad (6)$$

Towards a contradiction, assume $|B| > |Q^\mathcal{M}| \cdot 2^{|X^\mathcal{M}|}$. By Pigeonhole principle, there must exist two states $p \neq p' \in B$ such that $f(p) = f(p')$ (as $|B| > |Q^\mathcal{M}|$) and which furthermore satisfy $g(\chi^\mathcal{T}(p)) = g(\chi^\mathcal{T}(p'))$. This implies that $|\chi^\mathcal{T}(p)| = |\chi^\mathcal{T}(p')|$. Let $m_g : p \leftrightarrow p'$ be the matching such that $g(x) = g(m_g(x))$

for all $x \in \mathsf{dom}(m_g)$. It necessarily exists as $g(\chi^{\mathcal{T}}(p)) = g(\chi^{\mathcal{T}}(p'))$. Moreover, m_g is maximal. Hence, we have $p \#^{m_g} p'$ by (6). However, as $f(p) = f(p')$ and $g(x) = g(m_g(x))$ for all $x \in \mathsf{dom}(m_g)$, and by the contrapositive of Theorem 3, it then follows that $\neg(p \#^{m_g} p')$. We thus obtain a contradiction, meaning that both p and p' cannot be in B at the same time. Hence, $|B| \leq |Q^{\mathcal{M}}| \cdot 2^{|X^{\mathcal{M}}|}$. $\quad\square$

We immediately get bounds on the size of the frontier and the compatible sets. For the former, note that the number of immediate successors of each state is bounded by $|I| + |X^{\mathcal{M}}| = |A(\mathcal{M})|$; for the latter, the number of maximal matchings is at most $|X^{\mathcal{M}}|!$ since the number of active timers in any state from the observation tree is bounded by the same value from the hidden MMT $\mathcal{M}$.

Corollary 7.

$$|\mathcal{F}^{\mathcal{T}}| \leq |Q^{\mathcal{M}}| \cdot 2^{|X^{\mathcal{M}}|} \cdot (|A(\mathcal{M})|)$$

$$\max_{r \in \mathcal{F}^{\mathcal{T}}} |compat^{\mathcal{T}}(r)| \leq |Q^{\mathcal{M}}| \cdot 2^{|X^{\mathcal{M}}|} \cdot |X^{\mathcal{M}}|! \ .$$

Finally, we give an upper bound over the length of the minimal words ending in $to[x]$ for any $q \in Q^{\mathcal{T}}$ and $x \in \chi^{\mathcal{T}}(q)$. That is, when applying **WCT**, one can seek a short run to be replayed (typically, via a BFS).

Lemma 15. $\max_{q \in Q^{\mathcal{T}}} \max_{x \in \chi^{\mathcal{T}}(q)} \min_{q \xrightarrow{w \cdot to[x]} \in runs(\mathcal{T})} |w \cdot to[x]| \leq |Q^{\mathcal{M}}|.$

Proof. Towards a contradiction, assume that

$$\max_{q \in Q^{\mathcal{T}}} \max_{x \in \chi^{\mathcal{T}}(q)} \min_{q \xrightarrow{w \cdot to[x]} \in runs(\mathcal{T})} |w \cdot to[x]| > |Q^{\mathcal{M}}|.$$

Then, there must exist a state p_0 and a timer $x \in \chi^{\mathcal{T}}(p_0)$ such that the length of $w \cdot to[x]$ is strictly greater than the number of states of $\mathcal{M}$, i.e., we have a run

$$\pi = p_0 \xrightarrow{i_1} \cdots \xrightarrow{i_\ell} p_\ell \xrightarrow{to[x]} \in runs(\mathcal{T})$$

with $\ell > |Q^{\mathcal{M}}|$. Observe that $p_\ell \in \mathcal{E}^{\mathcal{T}}$, as $p_\ell \xrightarrow{to[x]}$ is defined. By Pigeonhole principle, we thus have $f(p_j) = f(p_\ell)$ for some $j \in \{1, \ldots, \ell - 1\}$. As $\mathcal{E}^{\mathcal{T}}$ is tree-shaped, it follows that $p_j \in \mathcal{E}^{\mathcal{T}}$ (since $j < \ell$).

We thus need to argue that $p_j \xrightarrow{to[x]} \in runs(\mathcal{T})$ to obtain our contradiction. Since $f(p_j) = f(p_\ell)$, it naturally follows that $\chi_0^{\mathcal{M}}(f(p_j)) = \chi_0^{\mathcal{M}}(f(p_\ell))$. Moreover, $x \in \chi^{\mathcal{T}}(p_j)$ as x is active in both p_0 and p_ℓ. Hence, $g(x) \in \chi_0^{\mathcal{M}}(f(p_j))$ as $g(x) \in \chi_0^{\mathcal{M}}(f(p_\ell))$. So, it must be that $p_j \xrightarrow{to[x]} \in runs(\mathcal{T})$ since $p_j \in \mathcal{E}^{\mathcal{T}}$. We thus have a contradiction as $p_0 \xrightarrow{i_1 \cdots i_j \cdot to[x]} \in runs(\mathcal{T})$ and $j < \ell$. $\quad\square$

Let us now prove Theorem 1, which we repeat.

Theorem 1. *The $L^{\#}_{MMT}$ algorithm terminates and returns an MMT $\mathcal{N} \overset{sym}{\approx} \mathcal{M}$ of size polynomial in $|Q^{\mathcal{M}}|$ and factorial in $|X^{\mathcal{M}}|$, in time and number of symbolic queries polynomial in $|Q^{\mathcal{M}}|, |I|$ and the length of the longest counterexample returned by the teacher, and factorial in $|X^{\mathcal{M}}|$.*

Proof. Let us start with showing that the algorithm eventually terminates. First, we formally prove that the refinement loop always finishes, i.e., that the basis and the frontier always stabilize. By Lemma 14 and Corollary 7, we have

$$|\mathcal{B}^{\mathcal{T}}| \leq |Q^{\mathcal{M}}| \cdot 2^{|X^{\mathcal{M}}|} \tag{7}$$

$$|\mathcal{F}^{\mathcal{T}}| \leq |Q^{\mathcal{M}}| \cdot 2^{|X^{\mathcal{M}}|} \cdot (|A(\mathcal{M})|) \tag{8}$$

$$\max_{r \in \mathcal{F}^{\mathcal{T}}} |compat^{\mathcal{T}}(r)| \leq |Q^{\mathcal{M}}| \cdot 2^{|X^{\mathcal{M}}|} \cdot |X^{\mathcal{M}}|! \tag{9}$$

Furthermore, by the first part of Corollary 3

$$\max_{q \in Q^{\mathcal{T}}} |\chi^{\mathcal{T}}(q)| \leq |\chi^{\mathcal{T}}(f(q))| \leq |X^{\mathcal{M}}|. \tag{10}$$

Let us argue that each part of the refinement loop is applied finitely many times. We write $|\textbf{Seismic}|$ for the number of times **Seismic** (see Section 3.3) is applied.

- The maximal number of applied **Seismic** *per basis state* is bounded by $|X^{\mathcal{M}}|$, by (10). Indeed, each **Seismic** event is due to the discovery of a new active timer in a basis state. Hence, by (7),

$$|\textbf{Seismic}| \leq |\mathcal{B}^{\mathcal{T}}| \cdot |X^{\mathcal{M}}| \leq |Q^{\mathcal{M}}| \cdot |X^{\mathcal{M}}| \cdot 2^{|X^{\mathcal{M}}|}. \tag{11}$$

- By (7), the number of times **Promotion** is applied between two instances of **Seismic** is bounded by $|Q^{\mathcal{M}}| \cdot 2^{|X^{\mathcal{M}}|}$. So,

$$|\textbf{Promotion}| \leq |\mathcal{B}^{\mathcal{T}}| \cdot |\textbf{Seismic}| \leq |Q^{\mathcal{M}}|^2 \cdot |X^{\mathcal{M}}| \cdot 2^{2|X^{\mathcal{M}}|}. \tag{12}$$

- Between two cases of **Seismic**, the number of **Completion** is bounded by $|I| \cdot |\mathcal{B}^{\mathcal{T}}|$, as, in the worst case, each input-transition is missing from each basis state. In general, we may have multiple frontier states $r_1, \dots, r_n$ such that $f(r_1) = \cdots = f(r_n)$. Thus, $compat^{\mathcal{T}}(r_1) = \cdots = compat^{\mathcal{T}}(r_n)$, after minimizing each set. Due to **Seismic**, $L^{\#}_{\text{MMT}}$ potentially has to choose multiple times one of those states. So, in the worst case, we select a different r_j each time. As each new basis state may not have all of its outgoing transitions,

$$|\textbf{Completion}| \leq |I| \cdot |\mathcal{B}^{\mathcal{T}}| \cdot |\textbf{Seismic}| = |I| \cdot |\textbf{Promotion}|$$
$$\leq |I| \cdot |Q^{\mathcal{M}}|^2 \cdot |X^{\mathcal{M}}| \cdot 2^{2|X^M|}. \tag{13}$$

- Between two instances of **Seismic**, the number of pairs $(p, m) \in compat^{\mathcal{T}}(r)$ such that $|\chi^{\mathcal{T}}(p)| \neq |\chi^{\mathcal{T}}(r)|$ is directly given by (9) for each frontier state r:

$$|\textbf{Active timers}| \leq |\mathcal{F}^{\mathcal{T}}| \cdot \max_{r \in \mathcal{F}^{\mathcal{T}}} |compat^{\mathcal{T}}(r)| \cdot |\textbf{Seismic}|$$
$$\leq |A(\mathcal{M})| \cdot |Q^{\mathcal{M}}|^3 \cdot |X^{\mathcal{M}}| \cdot 2^{3|X^{\mathcal{M}}|} \cdot |X^{\mathcal{M}}|!. \tag{14}$$

– With similar arguments,

$$|\mathbf{WCT}| \leq |\mathcal{F}^{\mathcal{T}}| \cdot \left(\max_{r \in \mathcal{F}^{\mathcal{T}}} |compat^{\mathcal{T}}(r)| \right)^2 \cdot |\mathbf{Seismic}|$$
$$\leq |A(\mathcal{M})| \cdot |Q^{\mathcal{M}}|^4 \cdot |X^{\mathcal{M}}|^2 \cdot 2^{4|X^{\mathcal{M}}|} \cdot \left(|X^{\mathcal{M}}|! \right)^2. \tag{15}$$

Since each part of the refinement loop can only be applied a finite number of times, it follows that the loop always terminates. Thus, it remains to prove that $L_{\mathrm{MMT}}^{\#}$ constructs finitely many hypotheses. Given how a counterexample is processed, it is now hard to see that any counterexample results in a new timer being discovered for a state in the basis (leading to an occurrence of **Seismic**), or a compatibility set decreasing in size (potentially leading to a **Promotion**). We already know that $|\mathbf{Seismic}|$ and $|\mathbf{Promotion}|$ are bounded by a finite constant. Moreover, by (9), each compatible set contains finitely many pairs. So, there can only be finitely many counterexamples, and, thus, hypotheses. More precisely, the number of hypotheses is bounded by

$$|\mathbf{Seismic}| \cdot |\mathbf{Promotion}| \cdot \max_{r \in \mathcal{F}^{\mathcal{T}}} |compat^{\mathcal{T}}(r)| \leq |Q^{\mathcal{M}}|^4 \cdot |X^{\mathcal{M}}|^2 \cdot 2^{4|X^{\mathcal{M}}|} \cdot |X^{\mathcal{M}}|! . \tag{16}$$

To establish that $\mathcal{N}$ is equivalent to $\mathcal{M}$, we observe that the last equivalence query to the teacher confirmed they are symbolically equivalent. Hence, by Lemma 4, they are also timed equivalent, i.e., $\mathcal{N} \overset{\mathrm{time}}{\approx} \mathcal{M}$. From our construction of an MMT hypothesis based on an gMMT (see Appendix E.2), we get that the intermediate gMMT has at most $|Q^{\mathcal{M}}| \cdot 2^{|X^{\mathcal{M}}|}$ states (by (7)). Hence, the final MMT has at most $|Q^{\mathcal{M}}| \cdot 2^{|X^{\mathcal{M}}|} \cdot |X^{\mathcal{M}}|!$ states by Lemma 12, i.e., a number that is polynomial in $|Q^{\mathcal{M}}|$ and factorial in $|X^{\mathcal{M}}|$, as announced.

We now prove the claimed number of queries. We start with the number of queries per step of the refinement loop and to process a counterexample.

Seismic   Applying **Seismic** does not require any symbolic queries.

Promotion   Let r be the state newly added to $\mathcal{B}^{\mathcal{T}}$. We thus need to do a wait query in each r' such that $r \overset{i}{\rightarrow} r'$ for some $i \in A(\mathcal{T})$. By (10), there are at most $|A(\mathcal{M})|$ wait queries.

Completion   A single application of **Completion** requires a single symbolic output query and a single wait query.

Active timers   Each occurrence of **Active timers** necessitates to replay a run π from state q. Let n be the number of transitions in π. In the worst case, we have to perform n symbolic output queries and n symbolic wait queries. By Lemma 15, $n \leq |Q^{\mathcal{M}}|$, if we always select a minimal run ending in the timeout of the desired timer.

WCT   Likewise, applying **WCT** requires to replay a run of length n, i.e., we do n symbolic output queries and n wait queries. This time, let us argue that we can always select a run such that: $n \leq |\mathcal{B}^{\mathcal{T}}| + 1 + |Q^{\mathcal{M}}| + \ell + (|\mathcal{B}^{\mathcal{T}}| + 1) \cdot |\mathbf{Seismic}|$. (Recall that ℓ is the length of the longest counterexample.) Let us decompose the summands appearing on the right piece by piece:

- $|\mathcal{B}^{\mathcal{T}}| + 1$ denotes the worst possible depth for a frontier state. Indeed, it may be that all basis states are on a single branch. So, the frontier states of the last basis state of that branch are at depth $|\mathcal{B}^{\mathcal{T}}| + 1$.
- $|Q^{\mathcal{M}}|$ comes from Lemma 15, as (enabled) requires to see the timeout of some timer.
- ℓ comes from the counterexample processing (see next item).
- When we previously replayed a witness of apartness due to some occurrences of **WCT**, we had to copy runs from a frontier state. Since the worst possible depth of a frontier state is $|\mathcal{B}^{\mathcal{T}}| + 1$, this means we added (at most) that length of the copied run when counted from the root of the observation tree. Since these replays may have triggered some instances of **Seismic**, the basis must have been recomputed each time. As explained above, we may not obtain the same exact basis, but the bound over the number of states is still (7). So, in the worst case, we add $|\mathbf{Seismic}|$ many times $(|\mathcal{B}^{\mathcal{T}}| + 1)$ to the longest branch of the tree.

Processing a counterexample First, in the worst case, we have to add the complete counterexample to observe what is needed, creating a new run in the tree, whose length is thus ℓ. Recall that each iteration of the counterexample processing splits a run $p \xrightarrow{v}$ with $p \in \mathcal{B}^{\mathcal{T}}$ into $p \xrightarrow{v'} r \xrightarrow{v''}$ such that $v = v' \cdot v''$ and $r \in \mathcal{F}^{\mathcal{T}}$. It may be that every v' is of length 1, meaning that we replay runs of lengths $\ell, \ell - 1, \ldots, 1$. So, we do $\ell + \ell - 1 + \cdots + 1 = \frac{\ell^2 + \ell}{2}$ symbolic output queries and the same number of wait queries.

By combining with the bounds of (11) to (15), we obtain the following bounds.

- The number of symbolic output queries is bounded by

$$1 \cdot |\mathbf{Completion}| + |Q^{\mathcal{M}}| \cdot |\mathbf{Active\ timers}|$$
$$+ \left(|\mathcal{B}^{\mathcal{T}}| + 1 + |Q^{\mathcal{M}}| + \ell + (|\mathcal{B}^{\mathcal{T}}| + 1) \cdot |\mathbf{Seismic}|\right) \cdot |\mathbf{WCT}|$$

  Clearly, $(\ell + |\mathcal{B}^{\mathcal{T}}| \cdot |\mathbf{Seismic}|) \cdot |\mathbf{WCT}|$ is bigger than the other operands (observe that ℓ is independent from $|Q^{\mathcal{M}}|, |I|$, and $|X^{\mathcal{M}}|$). Hence, the number of symbolic output queries is in

$$\mathcal{O}\left((\ell + |Q^{\mathcal{M}}|^2 \cdot |X^{\mathcal{M}}| \cdot 2^{2|X^{\mathcal{M}}|}) \cdot |A(\mathcal{M})| \cdot |Q^{\mathcal{M}}|^4 \cdot |X^{\mathcal{M}}|^2 \cdot 2^{4|X^{\mathcal{M}}|} \cdot (|X^{\mathcal{M}}|!)^2 \right).$$

- The number of symbolic wait queries is bounded by

$$|A(\mathcal{M})| \cdot |\mathbf{Promotion}| + 1 \cdot |\mathbf{Completion}| + |Q^{\mathcal{M}}| \cdot |\mathbf{Active\ timers}|$$
$$+ \left(|\mathcal{B}^{\mathcal{T}}| + 1 + |Q^{\mathcal{M}}| + \ell + (|\mathcal{B}^{\mathcal{T}}| + 1) \cdot |\mathbf{Seismic}|\right) \cdot |\mathbf{WCT}|.$$

  Again, $(\ell + |\mathcal{B}^{\mathcal{T}}| \cdot |\mathbf{Seismic}|) \cdot |\mathbf{WCT}|$ is bigger than the other operands. That is, we obtain the same complexity results as for symbolic output queries.
- The number of symbolic equivalence queries is exactly the number of constructed hypothesis. It is thus bounded by (16).
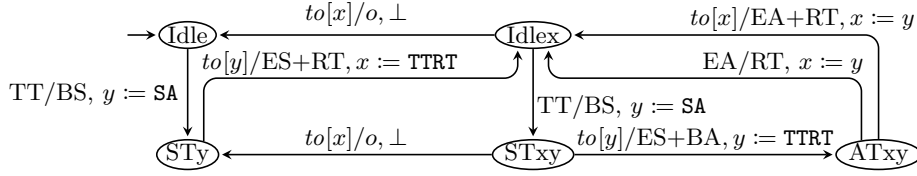
We thus obtain the announced complexity results. $\qquad\square$

Fig. 13: A gMMT model of one FDDI station.

F   FDDI protocol

By far the largest benchmark that is learned by Waga [33] is an fragment of the FDDI communication protocol [24], based on a timed automata model described in [9]. FDDI (Fiber Distributed Data Interface) is a protocol for a token ring that is composed of N identical stations. Figure 13 shows a gMMT-translation of the timed automata model for a single station from [9]. (See Definition 8 for the definition of gMMTs.) In the initial state `Idle`, the station is waiting for the token. When the token arrives (`TT`), the station begins with transmission of synchronous messages (`BS`). A timer y ensures that synchronous transmission ends (`ES`) after exactly `SA` time units, for some constant `SA` (= Synchronous Allocation). The station also maintains a timer x, that expires exactly `TTRT+SA` time units after the previous receipt of the token, for some constant `TTRT` (= Target Token Rotation Timer). When synchronous transmission ends and timer x has not expired yet, the station has the possibility to begin transmission of asynchronous messages (`BA`). Asynchronous transmission must end (`EA`) and the token must be returned (`RT`) at the latest when x expires.[15] Upon entering location `Idlex`, we ensure that timer x will expire exactly `TTRT+SA` time units after the previous `TT` event. In a FDDI token ring of size N, an `RT` event of station i will instantly trigger a `TT` event of station $(i + 1) \mod N$. Waga [33] considered the instance with 2 stations, `SA` = 20, and `TTRT` = 100.

---

[15] In location `ATxy`, timer x will expire before timer y (we may formally prove this by computing the zone graph): (1) The value of y in location `ATxy` is at most `TTRT`. (2) Hence, the value of x in location `Idlex` is at most `TTRT`. (3) So x is at most `TTRT` upon arrival in location `STxy`, and at most `TTRT−SA` upon arrival in location `ATxy`. (4) Thus x is smaller than y in location `ATxy` and will expire first.