Master thesis
Computing Science

Radboud University

k-*A*-Complete Conformance Testing of Mealy Machines with Timers

Author:
Bram Pellen
s1047349

First supervisor/assessor:
prof. dr. Frits Vaandrager
f.vaandrager@cs.ru.nl

Second assessor:
dr. Jurriaan Rot
j.rot@cs.ru.nl

April 23, 2025

This document is my final report for the master thesis that I completed as a part of the requirements for the degree of Master of Science.

Abstract

We present the first formal black-box conformance testing method for Mealy machines with timers (MMTs). We first develop our testing method for Mealy machines with a single timer (MM1Ts), as a stepping stone towards our method for MMTs. We prove that both of our testing procedures are k-A-complete. The use of our MM1T method strengthens the correctness guarantee provided by the active MM1T learning method of Vaandrager et al. [2023]. Our conformance testing procedure for MMTs supports specifications that are provided as generalized MMTs (gMMTs), as defined by Bruyère et al. [2024], rather than as MMTs. This makes it more flexible in use, since MMTs can easily be converted into gMMTs. We create an additional algorithm, due to which our MMT testing procedure can nearly be used to strengthen the correctness guarantee provided by Bruyère et al. [2024]'s learning procedure for MMTs.

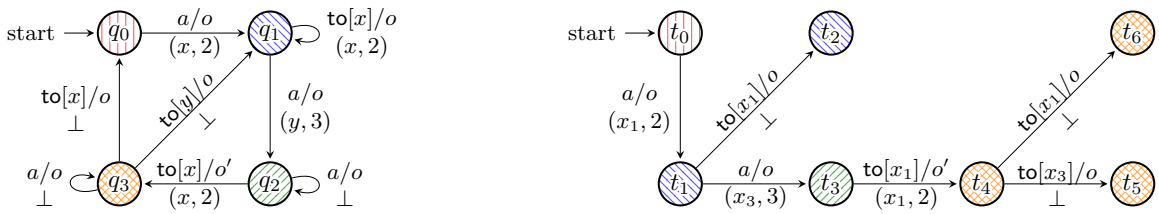
Contents

Chapter 1

Introduction

In this thesis, we address the problem of black-box conformance testing for timed systems. Black box conformance testing is the activity of determining whether a black-box system under test (SUT) conforms to a given specification. We start from a simple setting in which both the specification and the behavior of the SUT can be described by state-based systems known as Mealy machines [Mealy, 1955]. Mealy machines yield observable outputs in response to inputs from their environment. For every input, a Mealy machine also performs a state transition that determines how it responds to subsequent inputs. In this setting, we use finite-length input sequences to try to determine whether the Mealy machines $\mathcal{M}$ that describe the behavior of the SUT behave equivalently to a specification given by a Mealy machine $\mathcal{S}$. We call such an input sequence σ a *test*. Let $\mathcal{M}$ be a Mealy machine that describes the behavior of the SUT. We say that the SUT *passes* test σ iff $\mathcal{M}$ provides the same sequence of outputs in response to σ as $\mathcal{S}$. The SUT is said to *fail* σ otherwise. We call a finite set of tests a *test suite*. We say that the SUT *passes* a test suite $TS^{\mathcal{S}}$ iff the SUT passes every test of $TS^{\mathcal{S}}$. Otherwise, we say that the SUT fails $TS^{\mathcal{S}}$. We would want to generate test suites $TS^{\mathcal{S}}$ that are *complete*, in the sense that the SUT passes $TS^{\mathcal{S}}$ iff it is equivalent to specification $\mathcal{S}$. Unfortunately, complete test suites do not exist for arbitrary SUTs, since a finite number of finite-length tests could never show that the specification and the SUT exhibit the same behavior for all possible input sequences. An alternative to this completeness requirement is that of k-completeness. For a specification $\mathcal{S}$ and a natural number k, a test suite $TS_k^{\mathcal{S}}$ is k-complete if any SUT with at most k more states than $\mathcal{S}$ passes $TS_k^{\mathcal{S}}$ iff it is equivalent to $\mathcal{S}$. The k-completeness requirement is used by various Mealy machine conformance testing methods, such as the W-method [Chow, 1978, Vasilevskii, 1973], the H-method [Dorofeeva et al., 2005], the Wp-method [Fujiwara et al., 1991] and the HSI-method [Luo et al., 1995], among others [Dorofeeva et al., 2010a]. In Vaandrager et al. [2024], Vaandrager et al. introduced the notion of k-A-complete test suites. They proved that for Mealy machines, k-A-completeness subsumes k-completeness, and they provide a sufficient condition for k-A-complete test suites, under certain reasonable assumptions for A. *Fault domains* provide a different way to characterize completeness criteria for test suites and for conformance testing methods. A fault domain is a set $\mathcal{U}$ of Mealy machines. A test suite $TS^{\mathcal{S}}$ is $\mathcal{U}$-complete if for each $\mathcal{M} \in \mathcal{U}$, $\mathcal{U}$ only passes the test suite if it is equivalent to specification $\mathcal{S}$. For example, the fault domain for the k-completeness criterium is the set $\mathcal{U}_{n+k}$, where n is the number of states of the specification.

Advances in conformance testing are relevant to the area of active model learning. In active model learning, the goal is to learn a model of a system under learning (SUL) from information that is acquired through interaction with the system. Many approaches to active model learning are based on the *minimally adequate teacher* (MAT) framework that Angluin introduced in 1987. This framework describes the learning process as an interaction between a learner that wishes to learn a model of the SUL, and a teacher that knows the SUL's inner workings. The learner cannot directly ask the teacher for a description of the SUL's possible behavior. Instead, the learner asks the teacher queries to try and learn about this behavior. The learner can ask two types of queries. With membership queries (MQs), the learner asks the teacher for the output sequence that the SUT returns in response to a given input sequence. With equivalence queries (EQs), the learner asks whether a Mealy machine that it formulated from the data that it acquired up to this point is equivalent to the SUL. To answer these equivalence queries, the teacher needs to have an *equivalence oracle*



(a) An MMT

(b) An observation tree for the MMT on the left

Figure 1.1: An MMT, along with an observation tree for that MMT. The state colors indicate the correspondence between the states of the two models

that it can ask whether a given hypothesis model is equivalent to the black-box SUL.
Black-box conformance testing methods provide a means to create approximations of equivalence oracles with proven correctness guarantees. To construct an approximate equivalence oracle from a conformance testing method, we can treat hypotheses models as specifications, and SULs as SUTs. If we can assume that the SUL has at most k more states than the hypotheses produced by the learning method, then the use of a k-complete conformance testing method guarantees that the equivalence oracle judges any hypothesis to be equivalent to the SUL iff it is truly equivalent to the SUL. More generally, a conformance testing method that is proven to be correct under a specific fault domain $\mathcal{U}$ is also a valid equivalence oracle under the assumption that the SUL is in $\mathcal{U}$. Conformance testing methods can only provide approximations of equivalence oracles, because they can never be complete. These approximate equivalence oracles are approximations in the sense that they can never be guaranteed to work for arbitrary black-box SUTs. We already discussed the existence of conformance testing methods for Mealy machines that can thus be used to approximate equivalence oracles with proven correctness guarantees. However, not all model learning methods aim to learn Mealy machines.

The information that can be learned about a system's behavior by analyzing learned models is ultimately limited by the expressiveness of the models supported by the utilized model learning method. For instance, standard Mealy machines lack the ability to encode the real-time behavior that is exhibited by many real-world systems and protocols. Vaandrager et al. [2023] provided a method for learning Mealy machines with a single timer (MM1Ts), and Bruyère et al. [2024] introduced a method for learning Mealy machines with any finite number of timers (MMTs). MM1Ts generalize Mealy machines by extending the system with a single timer that can trigger state transitions when it runs out of time. MMTs further generalize these MM1Ts by having multiple independent timers that can be active simultaneously. Figure 1.1(a) shows an example of an MMT. This example model has q_0 for its initial state. This means that if the input a is the very first input that this model receives, then it transitions from the state q_0 to the state q_1 in a move that also yields output o, and that starts timer x with value 2. A subsequent delay of 2 time units would automatically trigger the timeout-transition with the special input $\mathsf{to}[x]$ that state q_1 has to itself.

The MM1T learning approach is based on Angluin's MAT framework. It requires an MM1T teacher that it can ask timed versions of the MAT membership and equivalence queries. It also introduces mappings between Mealy machines and MM1Ts, which it uses to construct an MM1T learner from a Mealy machine learning method, such as the L^*_M [Shahbaz and Groz, 2009] method, the TTT [Isberner et al., 2014] method, or the Suffix1by1 [Irfan et al., 2010] method. The MM1T learner uses an adapter that captures the information that it learns on the SUL in an *observation tree*. Observation trees are tree-shaped models that capture information that is observed about the behavior of other models.
The MMT learning method is based on $L^\#$, which is a method for learning Mealy machines that heavily relies on the use of observation trees [Vaandrager et al., 2022]. As such, the MMT learning method relies on observation trees as well. Figure 1.1(b) shows an observation tree $\mathcal{T}$ that the MMT learning method might create for the MMT $\mathcal{M}$ of Fig. 1.1(a). The inputs, outputs and timer updates for the transition sequences found in $\mathcal{T}$ aren't exactly equal to those supported by $\mathcal{M}$. Still, these sequences exhibit the same observable behavior, because the names of timers and the presence of timer updates that do not lead to timeouts cannot be ourwardly observed. Like the MM1T learning method, the MMT method also relies on the ability to evaluate timed versions of the MAT membership and equivalence queries.

No equivalence oracles with proven correctness guarantees have so far been developed for the Mealy machines with timers supported by the MM1T and MMT learning procedures. It could very well be possible to approximate such oracles with random testing, but there are as of yet no publications that explore this direction, and it would not come with any correctness guarantees. These two learning papers use distinct techniques to circumvent this lack of a black-box equivalence oracle in their experimental evaluations. The MM1T procedure does so through the use of conventional k-complete Mealy machine conformance testing methods. They use these testing methods to determine whether the hypotheses are equivalent to the SUL MM1Ts from their case studies when both are converted to Mealy machines. This approach cannot be used for black-box SULs, since the conversion method from MM1Ts to Mealy machines cannot be used on black-box MM1Ts. The MM1T method circumvents the lack of a black-box equivalence oracle for MMTs with a breadth-first-search-based algorithm that compares the learned hypotheses with modified versions of the SUL MMTs from the case studies. This approach doesn't work for black-box SULs, since this modification of the SULs cannot be used on black-box MMTs.

In this thesis, we develop k-A-complete conformance testing methods for MM1Ts and MMTs, in order to provide the learning methods from Vaandrager et al. [2023] and Bruyère et al. [2024] with approximations of equivalence oracles that have proven correctness guarantees. We first develop our testing method for MM1Ts as a stepping stone towards our method for MMTs. Vaandrager [2024] introduced a sufficient condition for k-complete test suites for Mealy machines. They proved that if an observation tree that is valid for both the specification and the SUT can satisfy certain conditions, then the fact that such a tree exists proves that the specification and the SUT are equivalent. Our testing methods attempt to construct such an observation tree for our timed settings. We prove that both of our testing methods are k-A-complete.
The main concern of both of our testing methods is to construct an observation tree that is valid for both the specification and the SUT. We keep expanding this observation tree until it either meets criteria for which we prove that they are sufficient to conclude that the specification and the SUT are equivalent, or until we find that one cannot construct an observation tree that is valid for both the specification and the SUT. In the latter case, we may conclude that the specification and the SUT are inequivalent, and we return a counterexample sequence for which the specification and the SUT exhibit different behavior. We also provide a way to minimize the specifications used by the MM1T conformance testing method, in order to lift the requirement for minimal specifications that this testing method inherits from Vaandrager [2024].

Our testing methods interact directly with the SUT. They terminate as soon as they can conclude that the SUT does not conform to the specification. These properties set them apart from the conformance testing methods that we discussed before, all of which generate test suites that then still need to be run on the SUT.

The rest of this thesis is structured as follows. In Chapter 2, we explain the notation we use and the preliminary notions that our conformance testing method for MM1Ts relies on. We introduce our MM1T testing method in Chapter 3. In Chapter 4, we discuss MMTs, generalized MMTs and other notions from Bruyère et al. [2024] that our conformance testing method for MMTs relies on. We introduce our conformance testing method for MMTs in Chapter 5. Chapter 6 contains our conclusions about our methods and our suggestions for future work. The appendices contain proofs and other material that we omitted from the main text to increase the document's readability.

Chapter 2

MM1T Testing Preliminaries

In this chapter, we introduce the notation, definitions and additional notions that our testing method for MM1Ts relies on. We start by introducing the notation that we use in this report. Next, we define Mealy machines, and we explain what it means to do conformance testing for Mealy machines. We discuss the notions of k-complete and k-A-complete test suites for Mealy machines. We end our discussion with an explanation of MM1Ts.

2.1 Notation

The **cardinality** of a set X is denoted by $|X|$. We use $\mathcal{P}(X)$ to denote the **power set** of X.

2.1.1 Functions

We write $f\colon X \rightharpoonup Y$ to denote that f is a **partial function** from domain X to codomain Y. Partial function f is **defined** for x, denoted $f(x)\downarrow$, if $\exists y\colon f(x) = y$. We use $f(x)\uparrow$ to indicate that f is **undefined** for x. We often treat partial functions $f\colon X \rightharpoonup Y$ as sets of pairs $\{(x, y) \in X \times Y \mid f(x) = y\}$.
Let $f\colon X \rightharpoonup Y$ and $g\colon X \rightharpoonup Y$ be partial functions. Then:

- $f(x) = g(x)$ iff either $f(x)\uparrow$ and $g(x)\uparrow$, or if $f(x)\downarrow$, $g(x)\downarrow$ and $f(x)$ and $g(x)$ yield the same value,
- $f = g$ iff $f(x) = g(x)$ for all $x \in X$, and
- $f \subseteq g$ iff, for all $x \in X$ for which $f(x)\downarrow$, $g(x)\downarrow$ and $g(x) = f(x)$.

A **total function** is a partial function for which, for all $x \in X\colon f(x)\downarrow$. We write $f\colon X \to Y$ to denote that f is a total function $f\colon X \rightharpoonup Y$. We usually refer to a function as partial to specify that it may or may not be total. Every function that we discuss in this thesis is total when we don't specify otherwise.

Let f be a partial or total function with a domain X. We write $f(A)$ to denote f's **image** for the subset $A \subseteq X$ of X, defined as:

$$f(A) = \{f(x) \mid x \in A\colon f(x)\downarrow\}$$

The function $\pi_1\colon X \times Y \to X$ is the **first projection** of pairs $(x, y) \in X \times Y$, defined as: $\pi_1((x, y)) = x$. The **second projection**, $\pi_2\colon X \times Y \to Y$, is similarly defined as: $\pi_2((x, y)) = y$.

2.1.2 Sequences

We consider **sequence**s of elements of sets. The symbol ϵ denotes the empty sequence. We fix the set $X = \{a, b\}$ for the examples of this subsection. We use $\sigma \cdot \rho$ to denote the **concatenation** of the sequences σ and ρ. We usually omit the concatenation operator when we append or prepend a single element to a sequence. The concatenation of all sequences of two sets of sequences X and Y, denoted $X \cdot Y$, is defined as:

$$X \cdot Y = \{x\ y \mid x \in X \wedge y \in Y\}$$

When concatenating sets of sequences, we sometimes omit the brackets around sets that consist of a single sequence of length 1. The notation X^n indicates the set of sequences over a set X that are n elements long, i.e. the set:

$$\begin{aligned} X^0 &= \{\epsilon\} \\ X^{n+1} &= X \cdot X^n \end{aligned}$$

We can see that $|X^n| = (|X|)^n$. We define the set $X^{\leq n}$ of sequences over X with length $\leq n$ as:

$$X^{\leq n} = \bigcup_{0 \leq j \leq n} X^j.$$

We can similarly define the set $X^{\geq n}$ of sequences over X with length $\geq n$ as:

$$X^{\geq n} = \bigcup_{j \geq n} X^j.$$

The set of all sequences over X of *any* length is given by:

$$X^* = \bigcup_{n \in \mathbb{N}} X^n.$$

We use $|\sigma| \in \mathbb{N}$ to indicate the length of sequence σ. We can define $|\sigma|$ as:

$$\begin{aligned} |\epsilon| &= 0 \\ |x\ \sigma| &= 1 + |\sigma|. \end{aligned}$$

Let $\sigma \in X^*$ be a sequence over X. If $|\sigma| \geq 1$, then:

- $\mathsf{head}(\sigma)$ yields the first element of σ:

$$\mathsf{head}(\sigma) = \begin{cases} a & \text{if } \sigma = a\ \rho \wedge a \in X \wedge \rho \in X^* \\ \text{undefined} & \text{otherwise,} \end{cases}$$

and

- $\mathsf{tail}(\sigma)$ yields the elements that follow after $\mathsf{head}(\sigma)$ in σ:

$$\mathsf{tail}(\sigma) = \begin{cases} \rho & \text{if } \sigma = a\ \rho \wedge a \in X \wedge \rho \in X^* \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We can see that for all sequences σ of length at least one, $\sigma = \mathsf{head}(\sigma)\ \mathsf{tail}(\sigma)$. Let $\sigma = \rho \cdot \rho'$ be a sequence. Then ρ is a **prefix** of σ, and ρ' is a **suffix** of σ.

2.2 Mealy Machines

Mealy machines are a type of state-based system. A Mealy machine yields observable outputs in response to the inputs that it receives from its environment. The state transitions that the Mealy machine performs in response to these inputs determine how the machine responds to subsequent inputs [Mealy, 1955].

Definition 2.2.1 (Partial Mealy machine). A **partial Mealy machine** is a tuple $\mathcal{M} = (Q, q_{\mathcal{I}}, I, O, \delta, \lambda)$, where:

- Q is a finite set of **states**; $q_{\mathcal{I}} \in Q$ is the **initial state**,

- I is a finite set of **inputs**,
- O is a set of **outputs**,
- $\delta\colon Q \times I \rightharpoonup Q$ is a partial **transition function**, and
- $\lambda\colon Q \times I \rightharpoonup O$ is a partial **output function**, satisfying:

  $$\lambda(q,i)\downarrow \quad \iff \quad \delta(q,i)\downarrow,$$

  (every transition has both an input and an output).

Mealy machine $\mathcal{M}$ is said to be **input complete** when its output and transition functions are total, since this would imply that each of $\mathcal{M}$'s state-input pairs has an associated transition.

Every Mealy machine that we discuss in this thesis is input complete, unless we specify otherwise.

Let $\mathcal{M}$ be an arbitrary (partial) Mealy machine with a set of states Q. We generalize the transition function to sequences of inputs, i.e. to elements of I^*. We get, for all $q \in Q$, all $i \in I$ and all $\sigma \in I^*$:

$$\delta^*(q,\epsilon) = q$$

$$\delta^*(q, i\ \sigma) = \begin{cases} \delta^*(\delta(q,i),\sigma) & \text{if } \delta(q,i)\downarrow \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We similarly generalize the output function to sequences of inputs. We get, for all $q \in Q$, all $i \in I$ and all $\sigma \in I^*$:

$$\lambda^*(q,\epsilon) = \epsilon$$

$$\lambda^*(q, i\ \sigma) = \begin{cases} \lambda(q,i)\ \lambda^*(\delta(q,i),\sigma) & \text{if } \delta(q,i)\downarrow \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We sometimes use a superscript to specify which model we consider, e.g. $Q^{\mathcal{M}}$, $q_{\mathcal{I}}^{\mathcal{M}}$.

Definition 2.2.2 (Trace equivalence). Let $\mathcal{M}$ and $\mathcal{N}$ be two Mealy machines with the same set of inputs, I. States $q^{\mathcal{M}}$ and $q^{\mathcal{N}}$ are **trace equivalent (equivalent)**, denoted $q^{\mathcal{M}} \approx_{trace} q^{\mathcal{N}}$ iff, for all input sequences $\sigma \in I^*$: $\lambda^{\mathcal{M}^*}(q^{\mathcal{M}},\sigma) = \lambda^{\mathcal{N}^*}(q^{\mathcal{N}},\sigma)$.

Mealy machines $\mathcal{M}$ and $\mathcal{N}$ are considered to be **trace equivalent (equivalent)**, denoted $\mathcal{M} \approx_{trace} \mathcal{N}$, iff $q_{\mathcal{I}}^{\mathcal{M}} \approx_{trace} q_{\mathcal{I}}^{\mathcal{N}}$.

Note that two states of the same Mealy machine can also be trace (in)equivalent, as the above definition does not require for $\mathcal{M}$ and $\mathcal{N}$ to be different Mealy machines.

Definition 2.2.3 (Connected Mealy machine). A Mealy machine $\mathcal{M}$ with a set of states Q and an initial state $q_{\mathcal{I}}$ is **connected** when, for all states $q \in Q$ there exists an input sequence $\sigma \in I^*$ such that $\delta^*(q_{\mathcal{I}},\sigma) = q$.

Definition 2.2.4 (Minimal Mealy machine). A Mealy machine is said to be **minimal** if no two of its states are equivalent.

We rely on the notion of **apartness** to prove the validity of our method. Apartness is a form of inequality that is constructive, in the sense that one can never simply assert that two values are apart from one another. Instead, one always needs to provide an example that shows that the values are unequal [Troelstra and Schwichtenberg, 2000, Geuvers and Jacobs, 2021].

Example 2.2.1. Vaandrager et al. [2022] applies the notion of apartness to Mealy machines. For a Mealy machine $\mathcal{M}$, they consider two states $q, q' \in Q$ to be apart, denoted $q \# q'$, iff there is some $\sigma \in I^*$ such that $\lambda^*(q,\sigma)\downarrow$, $\lambda^*(q',\sigma)\downarrow$, and $\lambda^*(q,\sigma) \neq \lambda^*(q',\sigma)$. Input sequence σ is then called a **witness** of $q \# q'$.

Note that it is possible for two Mealy machines states q and q' to be neither trace equivalent nor apart, since the apartness $q \# q'$ could only be established if a witness of this apartness has been identified.

The term "apartness" is often applied to relations that are irreflexive, symmetric, and co-transitive [Geuvers and Jacobs, 2021]. Not all notions of apartness satisfy all three of these properties. The notion of apartness for Mealy machines discussed in Example 2.2.1 satisfies all three of these properties for input-complete Mealy machines.

2.3 Conformance Testing for Mealy Machines

Conformance testing for Mealy machines is the activity of determining whether a **system under test (SUT)** whose behavior can be described by an (unknown) Mealy machine $\mathcal{M}$ behaves equivalently to a given **specification** Mealy machine $\mathcal{S}$. We say that an SUT whose behavior can be described by a Mealy machine $\mathcal{M}$ **conforms** to specification Mealy machine $\mathcal{S}$ iff $\mathcal{M} \approx_{trace} \mathcal{S}$.

In general, conformance testing methods generate a finite collection of tests that can reveal whether supposed implementations conform to the specification. Such a collection of tests is called a **test suite**. For Mealy machines, a test suite $TS^{\mathcal{S}}$ for a specification Mealy machine $\mathcal{S}$ is a set of finite-length sequences over the set of inputs $I^{\mathcal{S}}$ of $\mathcal{S}$, each of which is referred to as an individual **test**. The SUT **passes** test $\sigma \in TS^{\mathcal{S}}$ iff its behavior is described by a Mealy machine $\mathcal{M}$, for which $\lambda^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma) = \lambda^{\mathcal{S}^*}(s_{\mathcal{I}}, \sigma)$. The SUT passes a test suite iff it passes all of its tests. The SUT can then be called an **implementation** of the specification.

2.4 k-Complete Test Suites for Mealy Machines

Let $\mathcal{S}$ be a specification Mealy machine. We would ideally want to compute **complete** test suites for $\mathcal{S}$, which we could then use to determine for any Mealy machine $\mathcal{M}$ that describes the SUT whether $\mathcal{M} \approx_{trace} \mathcal{S}$. Mealy machine $\mathcal{M}$ would pass such a test suite iff $\mathcal{M} \approx_{trace} \mathcal{S}$.

Unfortunately, complete test suites do not exist. The problem is that any test case is always finite in length. Let k be the length of the longest test case of a test suite $TS^{\mathcal{S}}$ for a specification $\mathcal{S}$. Let $\mathcal{M}$ be an SUT for which all states that can be reached from the initial state within k state transitions exhibit the same output behavior as their corresponding states in $\mathcal{S}$, and for which the states beyond that point exhibit different behavior from their counterparts in $\mathcal{S}$. SUT $\mathcal{M}$ would pass test suite $TS^{\mathcal{S}}$, even though $\mathcal{M} \not\approx_{trace} \mathcal{S}$.

A typical approach to circumvent this issue is to put a limit on the number of states that the SUTs are allowed to have. This approach leads to the notion of k-complete test suites:

Definition 2.4.1 (k-Complete test suites for Mealy machines). Let $\mathcal{S}$ be a Mealy machine, and let $k \in \mathbb{N}$. Then test suite $TS^{\mathcal{S}}_k$ is **k-complete** for $\mathcal{S}$ if, for any SUT Mealy machine $\mathcal{M}$ with at most k extra states with respect to $\mathcal{S}$:

$$\mathcal{M} \text{ passes } TS^{\mathcal{S}}_k \iff \mathcal{M} \approx_{trace} \mathcal{S}.$$

Fault domains offer an alternative way to characterize completeness measures for test suites.

Definition 2.4.2 (Fault domains and $\mathcal{U}$-completeness). Let $\mathcal{S}$ be a Mealy machine. A **fault domain** is a set $\mathcal{U}$ of Mealy machines. A test suite $TTS^{\mathcal{S}}$ for $\mathcal{S}$ is **$\mathcal{U}$-complete** if, for each $\mathcal{M} \in \mathcal{U}$, $\mathcal{M}$ passes $TTS^{\mathcal{S}}$ implies $\mathcal{M} \approx_{trace} \mathcal{S}$.

Consider for example the following fault domain:

Definition 2.4.3. Let $m \in \mathbb{N}^{>0}$. Then $\mathcal{U}_m$ is the set of all Mealy machines with at most m states.

We can use fault domain $\mathcal{U}_m$ to express the k-completeness property:

Definition 2.4.4 (k-Complete test suites for Mealy machines in terms of fault domains). Let $\mathcal{S}$ be a Mealy machine, and let $k \in \mathbb{N}$. Let $m = |Q^{\mathcal{S}}| + k$. Then $TS^{\mathcal{S}}_k$ is **k-complete** for $\mathcal{S}$ if, for any SUT Mealy machine $\mathcal{M} \in \mathcal{U}_m$:

$$\mathcal{M} \text{ passes } TS^{\mathcal{S}}_k \iff \mathcal{M} \approx_{trace} \mathcal{S}.$$

There are various methods for deriving k-complete test suites for Mealy machines. This survey Dorofeeva et al. [2010b] covers multiple well-known approaches, including the W-method [Chow, 1978, Vasilevskii, 1973], the H-method [Dorofeeva et al., 2005], and several other approaches that improve upon the W-method.

We discuss the W- and the H-methods in the next two subsections. Both of these methods rely on the notion of state covers:

Definition 2.4.5. Let $\mathcal{M}$ be a Mealy machine with a set of inputs I. A **state cover** [Dorofeeva et al., 2010b] for $\mathcal{M}$ is a set $C \subseteq I^*$ that contains for every state $q \in Q$ a sequence $\sigma \in C$ for which $\delta^*(q_\mathcal{I}, \sigma) = q$. State covers are also required to contain the empty sequence, ϵ. A state cover is **prefix closed** if, for all $\sigma \in C$, all prefixes ρ of σ are also in C. A state cover is **minimal** if, for all $\sigma, \rho \in C$ with $\sigma \neq \rho$, $\delta^*(q_\mathcal{I}, \sigma) \neq \delta^*(q_\mathcal{I}, \rho)$.

Informally, a state cover C is a set of input sequences of a Mealy machine $\mathcal{M}$ that contains for every state $q \in Q$ a finite-length input sequence that reaches q from $\mathcal{M}$'s initial state, $q_\mathcal{I}$.

2.4.1 The W-Method

The W-method [Chow, 1978, Vasilevskii, 1973] is a method for deriving k-complete test suites for input-complete Mealy machines. Let $\mathcal{M}$ be a minimal connected Mealy machine, and let $k \in \mathbb{N}$. Use of the W-method requires the computation of the following three sets of input sequences [Chow, 1978, Vasilevskii, 1973]:

- the state cover $C \subseteq I^*$,
- the set $I^{\leq k+1} \subseteq I^*$, and
- the **characterization set** $W \subseteq I^*$ that contains for every pair of states $q, q' \in Q$ for which $q \neq q'$ an input sequence σ such that $\lambda^*(q, \sigma) \neq \lambda^*(q', \sigma)$. Such an input sequence shows that $q \not\approx_{trace} q'$.

The full k-complete test suite $TS_k^\mathcal{M} \subseteq I^*$ for $\mathcal{M}$ and k is given by Chow [1978] as:

$$TS_k^\mathcal{M} = C \cdot I^{\leq k+1} \cdot W.$$

2.4.2 The H-method

The H-method [Dorofeeva et al., 2005] is a method for deriving k-complete test suites for Mealy machines. Let $\mathcal{M}$ be a minimal connected Mealy machine, let C be a prefix-closed state cover for $\mathcal{M}$, and let $k \in \mathbb{N}$. The H-method constructs a k-complete test suite for $\mathcal{M}$ in four steps:

1. $TS_k^\mathcal{M} = C \cdot I^{\leq k+1}$.
2. For all $\sigma, \sigma' \in C$. Let $q = \delta^*(q_\mathcal{I}, \sigma)$, and let $q' = \delta^*(q_\mathcal{I}, \sigma')$. If $q \neq q'$ and there are no input sequences $\sigma \cdot w, \sigma' \cdot w \in TS_k^\mathcal{M}$ for which $\lambda^*(q, w) \neq \lambda^*(q', w)$, then find an input sequence w such that $\lambda^*(q, w) \neq \lambda^*(q', w)$ and add the input sequences $\sigma \cdot w$ and $\sigma' \cdot w$ to $TS_k^\mathcal{M}$.
3. For all $\sigma \in C$ and $\rho \in (C \cdot I^{\leq k+1}) \setminus C$ for which $q = \delta^*(q_\mathcal{I}, \sigma)$, $t = \delta^*(q_\mathcal{I}, \rho)$, and $q \neq t$; if there are no input sequences $\sigma \cdot w, \rho \cdot w \in TS_k^\mathcal{M}$ for which $\lambda^*(q, w) \neq \lambda^*(t, w)$, then find an input sequence w such that $\lambda^*(q, w) \neq \lambda^*(t, w)$ and add the input sequences $\sigma \cdot w$ and $\rho \cdot w$ to $TS_k^\mathcal{M}$.
4. If $k \geq 1$, then for all $\sigma \in (C \cdot I^{\leq k+1}) \setminus C$ and $\rho \in I \cdot I^{\leq k-1}$ such that $\sigma \cdot \rho \in C \cdot I^{\leq k+1}$, $t = \delta^*(q_\mathcal{I}, \sigma)$, $t' = \delta^*(t, \rho)$, and $t \neq t'$; if there are no input sequences $\sigma \cdot w, \sigma \cdot \rho \cdot w \in TS_k^\mathcal{M}$ for which $\lambda^*(t, w) \neq \lambda^*(t', w)$, then find an input sequence w such that $\lambda^*(t, w) \neq \lambda^*(t', w)$ and add the input sequences $\sigma \cdot w$ and $\sigma \cdot \rho \cdot w$ to $TS_k^\mathcal{M}$.

2.5 k-A-Complete Test Suites for Mealy Machines

Vaandrager et al. [2024] introduced a notion of test suite completeness that subsumes k-completeness. To this end, they introduced the following fault domain:

Definition 2.5.1. Let $k \in \mathbb{N}$, and let $\mathsf{A} \subseteq I^*$. Then $\mathcal{U}_k^\mathsf{A}$ is the set of all Mealy machines $\mathcal{M}$ for which every state of $\mathcal{M}$ can be reached by an input sequence $\sigma \cdot \rho$, for some $\sigma \in \mathsf{A}$ and $\rho \in I^{\leq k}$.

The idea is that a test suite computed for a Mealy machine $\mathcal{M}$ often consists of tests of which the prefixes are input sequences from some set of input sequences A. This set is typically given by a state cover for $\mathcal{M}$, as is the case in the W- and H-methods. The tests generated by such testing methods often proceed with between 0 and k inputs that, together with the prefixes from A, are meant to reach all states of the SUT [Vaandrager et al., 2024]. The W- and H-methods both follow this principle.

Remember that the W- and H-methods are both k-complete, and that for k-completeness we have the fault domain $\mathcal{U}_m$, where $m = |Q^{\mathcal{S}}| + k$. Let k be a natural number, and let $\mathcal{M}$ be a Mealy machine in $\mathcal{U}_m$. Let $\mathsf{A} \subseteq I^*$. It is possible that even though $\mathcal{M} \in \mathcal{U}_m$, $\mathcal{M} \notin \mathcal{U}_k^{\mathsf{A}}$. In particular, if $\mathcal{M}$ is minimal and A is a minimal state cover for $\mathcal{M}$, then it could *not* be the case that there exist $\sigma, \rho \in \mathsf{A} : \sigma \neq \rho \wedge \delta^*(q_{\mathcal{I}}, \sigma) \approx_{trace} \delta^*(q_{\mathcal{I}}, \rho)$. This motivates the use of the additional fault domain:

Definition 2.5.2. Let $\mathsf{A} \subseteq I^*$. Then $\mathcal{U}^{\mathsf{A}}$ is the set of all Mealy machines $\mathcal{M}$ for which there are $\sigma, \rho \in \mathsf{A}$ with $\sigma \neq \rho$ and $\delta^*(q_{\mathcal{I}}, \sigma) \approx_{trace} \delta^*(q_{\mathcal{I}}, \rho)$.

Let $\mathsf{A} \subseteq I^*$ be a finite set of input sequences with $\epsilon \in \mathsf{A}$, and let k and m be natural numbers with $m = |\mathsf{A}| + k$. Then $\mathcal{U}_m \subseteq \mathcal{U}_k^{\mathsf{A}} \cup \mathcal{U}^{\mathsf{A}}$. The fault domain for k-A-completeness is given by $\mathcal{U}_k^{\mathsf{A}} \cup \mathcal{U}^{\mathsf{A}}$, to ensure that k-A-completeness subsumes k-completeness.

Let $\mathcal{S}$ be a Mealy machine with a set of inputs I, and let $k \in \mathbb{N}$. Then:

Definition 2.5.3 (k-A-complete test suites for Mealy machines). Let $\mathcal{S}$ be a Mealy machine with a set of inputs I, let $k \in \mathbb{N}$, and let $\mathsf{A} \subseteq I^*$. Then test suite $TS^{\mathcal{S}}$ is ***k*-A-complete** for $\mathcal{S}$ if, for any SUT Mealy machine $\mathcal{M} \in \mathcal{U}_k^{\mathsf{A}} \cup \mathcal{U}^{\mathsf{A}}$:

$$\mathcal{M} \text{ passes } TS^{\mathcal{S}} \iff \mathcal{M} \approx_{trace} \mathcal{S}.$$

2.6 Mealy Machines With a Single Timer (MM1Ts)

The Mealy machines with a single timer (MM1Ts) that we consider in this work were first introduced by Vaandrager et al. as a generalization of Mealy machines [Vaandrager et al., 2023]. This section provides an overview of their most important definitions, explanations and conclusions.

MM1Ts are Mealy machines that are extended with a single timer that can trigger a state transition when it runs out of time. When that happens, a transition with a special timeout input is taken.

Definition 2.6.1 (Mealy machine with a single timer). A **Mealy machine with a single timer (MM1T)** is a tuple $\mathcal{M} = (Q, q_{\mathcal{I}}, I, O, \delta, \lambda, \tau)$, where:

- $Q = Q_{off} \cup Q_{on}$ is a finite set of **states**, partitioned into subsets where the timer is off and on, respectively,
- $q_{\mathcal{I}} \in Q_{off}$ is the **initial state**,
- I is a finite set of **inputs** that contains a special element timeout,
- O is a set of **outputs**,
- $\delta \colon Q \times I \rightharpoonup Q$ is a **transition function**, satisfying

$$\delta(q, i)\uparrow \iff q \in Q_{off} \wedge i = \mathsf{timeout} \tag{2.1}$$

  (inputs are always defined, except for timeout in states where the timer is off),

- $\lambda \colon Q \times I \rightharpoonup O$ is an **output function**, satisfying

$$\lambda(q, i)\downarrow \iff \delta(q, i)\downarrow \tag{2.2}$$

  (every transition has both an input and an output), and

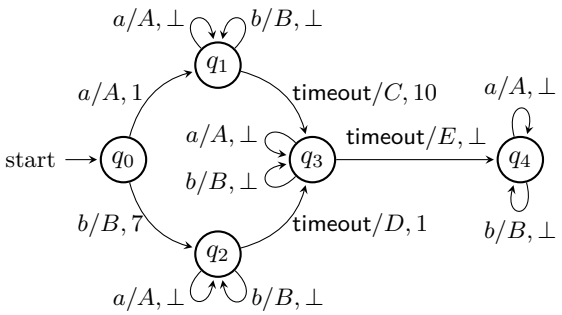


Figure 2.1: An MM1T with $Q_{on} = \{q_1, q_2, q_3\}$ and $Q_{off} = \{q_0, q_4\}$

- $\tau \colon Q \times I \rightharpoonup \mathbb{N}^{>0}$ is a **reset function**, satisfying

$$\tau(q,i)\downarrow \quad \implies \quad \delta(q,i) \in Q_{on} \tag{2.3}$$

$$q \in Q_{off} \quad \wedge \quad \delta(q,i) \in Q_{on} \quad \implies \quad \tau(q,i)\downarrow \tag{2.4}$$

$$\delta(q, \mathsf{timeout}) \in Q_{on} \quad \implies \quad \tau(q, \mathsf{timeout})\downarrow \tag{2.5}$$

  (when a transition (re)sets the timer, the timer is on in the target state; when it moves from a state where the timer is off to a state where the timer is on, it sets the timer; if the timer stays on after a timeout, it is reset).

We sometimes use this input-complete version of the reset function:

$$\tau_\bot(q,i) = \begin{cases} \tau(q,i) & \text{if } \tau(q,i)\downarrow \\ \bot & \text{otherwise.} \end{cases}$$

Let $\delta(q,i) = q'$ and $\lambda(q,i) = o$. We write $q \xrightarrow{i/o,n} q'$ if $\tau(q,i) = n \in \mathbb{N}^{>0}$, and $q \xrightarrow{i/o,\bot} q'$ or just $q \xrightarrow{i/o} q'$ if $\tau(q,i)\uparrow$.

Example 2.6.1. Figure 2.1 show an example of an MM1T. The MM1T has five states, with $Q_{on} = \{q_1, q_2, q_3\}$ and $Q_{off} = \{q_0, q_4\}$. The numerical constants displayed on some of the transitions update the timer to the integer value that they specify. A timeout transition is traversed whenever the timer runs out of time. The only way to transition from any of the states q_1, q_2 and q_3 to a different state is by waiting for a timeout to occur.

Definition 2.6.2 (Partial MM1T). A **partial MM1T** is an MM1T for which the transition function doesn't have to satisfy the condition of Rule 2.1. It instead has to satisfy a weaker version of this rule, with the implication in only one direction:

$$q \in Q_{off} \quad \wedge \quad i = \mathsf{timeout} \quad \implies \quad \delta(q,i)\uparrow.$$

We sometimes refer to MM1Ts for which the transition function does satisfy the condition of Rule 2.1 as **complete MM1Ts**.

We use the same method to generalize a (partial) MM1T's transition and output functions to sequences of inputs as we used to generalize those for (partial) Mealy machines. We get, for all $q \in Q$, all $i \in I$ and all $\sigma \in I^*$:

$$\delta^*(q, \epsilon) = q$$

$$\delta^*(q, i\ \sigma) = \begin{cases} \delta^*(\delta(q,i), \sigma) & \text{if } \delta(q,i)\downarrow \\ \text{undefined} & \text{otherwise} \end{cases}$$

and:

$$\lambda^*(q,\epsilon) = \epsilon$$
$$\lambda^*(q, i\ \sigma) = \begin{cases} \lambda(q,i)\ \lambda^*(\delta(q,i),\sigma) & \text{if } \delta(q,i)\downarrow \wedge \lambda(q,i)\downarrow \\ \text{undefined} & \text{otherwise.} \end{cases}$$

There are two semantics for MM1Ts, an untimed and a timed one.

2.6.1 Untimed Semantics

The untimed semantics is defined in terms of **untimed words**. An untimed word gathers the inputs, the outputs, and the values to which the timer is set in every transition. Untimed words over inputs I and outputs O are defined as sequences:

$$(i_0, o_0, n_0)\ (i_1, o_1, n_1)\ \ldots\ (i_k, o_k, n_k),$$

where, for each index $0 \leq j \leq k$: $i_j \in I$, $o_j \in O$, and $n_j \in (\mathbb{N} \cup \{\bot\})$. Untimed words can start in any state of an MM1T.

Example 2.6.2. An example of an untimed word that starts in the initial state of the MM1T of Figure 2.1 is:

$$(a, A, 1)\ (b, B, \bot)\ (\mathsf{timeout}, C, 10)\ (a, A, \bot)\ (\mathsf{timeout}, E, \bot)\ (a, A, \bot).$$

An **untimed run** of an MM1T $\mathcal{M}$ is a sequence:

$$q_0 \xrightarrow{i_0/o_0,n_0} q_1 \xrightarrow{i_1/o_1,n_1} \ldots \xrightarrow{i_k/o_k,n_k} q_{k+1}$$

such that, for every $j \leq k$, $q_j \xrightarrow{i_j/o_j,n_j} q_{j+1}$ is a transition of $\mathcal{M}$. Untimed runs can also start in any state of an MM1T.

Example 2.6.3. Let w be the untimed word of Example 2.6.2, and let $\mathcal{M}$ be the MM1T of Figure 2.1. The untimed run from $\mathcal{M}$'s initial state **over** w is given by:

$$q_0 \xrightarrow{a/A,1} q_1 \xrightarrow{b/B,\bot} q_1 \xrightarrow{\mathsf{timeout}/C,10} q_3 \xrightarrow{a/A,\bot} q_3 \xrightarrow{\mathsf{timeout}/E,\bot} q_4 \xrightarrow{a/A,\bot} q_4.$$

Note that an MM1T state always has at most one untimed run for every untimed word. We say that w is an **untimed word of** $\mathcal{M}$'s state q iff q has an untimed run over w. We say that $\mathcal{M}$ **has an untimed word** w iff w is an untimed word of $\mathcal{M}$'s initial state.

We use the notion of untimed words to define the notion of untimed equivalence:

Definition 2.6.3 (Untimed equivalence). Let $\mathcal{M}$ and $\mathcal{N}$ be two MM1Ts with the same inputs. States $q^{\mathcal{M}}, q^{\mathcal{N}}$ are **untimed equivalent**, denoted $q^{\mathcal{M}} \approx_{untimed} q^{\mathcal{N}}$, iff they have the same sets of untimed words.

MM1Ts $\mathcal{M}$ and $\mathcal{N}$ are **untimed equivalent**, denoted $\mathcal{M} \approx_{untimed} \mathcal{N}$, iff $q_{\mathcal{I}}^{\mathcal{M}} \approx_{untimed} q_{\mathcal{I}}^{\mathcal{N}}$.

Note that two states of the same MM1T can also be untimed (in)equivalent, as the above definition does not require for $\mathcal{M}$ and $\mathcal{N}$ to be different MM1Ts.

2.6.2 Timed Semantics

The timed semantics of an MM1T $\mathcal{M}$ describes its real-time behavior. It associates an infinite-state transition system $\mathsf{tsem}(\mathcal{M})$ to $\mathcal{M}$. Every state of $\mathsf{tsem}(\mathcal{M})$ combines an MM1T state with a timer value. We call these states **configurations**. A configuration is thus a pair (q, t), where $q \in Q$ is a state and $t \in (\mathbb{R}^{\geq 0} \cup \{\infty\})$ is a timer value. We require $t = \infty$ iff $q \in Q_{off}$. The **initial configuration** is given by $(q_{\mathcal{I}}, \infty)$. The transition system describes all possible configurations and all transitions between them. We use four rules to define the

transition relation that describes how one configuration may evolve into another. For all $q \in Q$, $r \in Q_{off}$, $s, s' \in Q_{on}$, $i \in I$, $o \in O$, $t \in \mathbb{R}^{\geq 0} \cup \{\infty\}$, $d \in \mathbb{R}^{\geq 0}$ and $n \in \mathbb{N}^{>0}$:

$$\frac{d \leq t}{(q,t) \xrightarrow{d} (q, t-d)} \tag{2.6}$$

$$\frac{q \xrightarrow{i/o,n} s, \quad i = \mathsf{timeout} \Rightarrow t = 0}{(q,t) \xrightarrow{i/o} (s,n)} \tag{2.7}$$

$$\frac{q \xrightarrow{i/o} r, \quad i = \mathsf{timeout} \Rightarrow t = 0}{(q,t) \xrightarrow{i/o} (r,\infty)} \tag{2.8}$$

$$\frac{s \xrightarrow{i/o} s', \quad i \neq \mathsf{timeout}}{(s,t) \xrightarrow{i/o} (s',t)} \tag{2.9}$$

Rule 2.6 states that when the time advances by d time units, the timer decreases by d. The timer may not go below 0. We use the convention that, for any $d \in \mathbb{R}^{\geq 0}$, $\infty - d = \infty$. The time may thus advance indefinitely when the timer is off. Rule 2.7 describes transitions that (re)set the timer; a timeout may only occur if the timer has expired in the source state. Rule 2.8 describes transitions for which the timer is off in the target state; again, a timeout may only occur if the timer has expired in the source state. Finally, Rule 2.9 describes transitions for which the timer remains on and is not reset.

A **timed word** over inputs I and outputs O is a sequence:

$$w = (t_0, i_0, o_0)\ (t_1, i_1, o_1)\ \ldots\ (t_k, i_k, o_k),$$

where, for each index $0 \leq j \leq k$: $t_j \in \mathbb{R}^{\geq 0}$, $i_j \in I$, and $o_j \in O$. A timed word w describes a possible behavior that may be observed when interacting with an MM1T: after a delay of t_j time units, input i_j is provided and output o_j is obtained in response. This process then repeats for index $j+1$. For a given timed word w, a **timed run** of the MM1T $\mathcal{M}$ over w is a sequence:

$$\alpha = C_0 \xrightarrow{t_0} C_0' \xrightarrow{i_0/o_0} C_1 \xrightarrow{t_1} C_1' \xrightarrow{i_1/o_1} C_2 \ldots \xrightarrow{t_k} C_k' \xrightarrow{i_k/o_k} C_{k+1}$$

that begins with $\mathsf{tsem}(\mathcal{M})$'s initial configuration C_0 and where, for each $j \leq k$: $C_j \xrightarrow{t_j} C_j'$ and $C_j' \xrightarrow{i_j/o_j} C_{j+1}$ are transitions of $\mathsf{tsem}(\mathcal{M})$. Since MM1Ts are deterministic, $\mathcal{M}$ has at most one timed run over each timed word w. We say that w is a timed word of $\mathcal{M}$ iff $\mathcal{M}$ has a timed run over w.

Example 2.6.4. An example of a timed word for the MM1T of Figure 2.1 is:

$$(12, a, A)\ (0.2, b, B)\ (0.8, \mathsf{timeout}, C)\ (1.1, a, A)\ (8.9, \mathsf{timeout}, E)\ (17, a, A).$$

This timed word has the corresponding timed run:

$$\begin{aligned}(q_0,\infty) &\xrightarrow{12} (q_0,\infty) \xrightarrow{a/A} (q_1,1) \xrightarrow{0.2} (q_1,0.8) \xrightarrow{b/B} (q_1,0.8) \xrightarrow{0.8} (q_1,0) \xrightarrow{\mathsf{timeout}/C} (q_3,10) \\ &\xrightarrow{1.1} (q_3,8.9) \xrightarrow{a/A} (q_3,8.9) \xrightarrow{8.9} (q_3,0) \xrightarrow{\mathsf{timeout}/E} (q_4,\infty) \xrightarrow{17} (q_4,\infty) \xrightarrow{a/A} (q_4,\infty).\end{aligned}$$

We use the notion of timed words to define timed equivalence:

Definition 2.6.4 (Timed equivalence). Let $\mathcal{M}$ and $\mathcal{N}$ be MM1Ts. Then, $\mathcal{M}$ and $\mathcal{N}$ are **timed equivalent**, denoted $\mathcal{M} \approx_{timed} \mathcal{N}$, if and only if $\mathcal{M}$ and $\mathcal{N}$ have the same sets of timed words.

For any two MM1Ts $\mathcal{M}$ and $\mathcal{N}$, if $\mathcal{M}$ and $\mathcal{N}$ are untimed equivalent, then they are also timed equivalent. The converse also holds. We thus know that:

Lemma 2.6.1. Let $\mathcal{M}$ and $\mathcal{N}$ be MM1Ts. Then, $\mathcal{M} \approx_{timed} \mathcal{N}$ if and only if $\mathcal{M} \approx_{untimed} \mathcal{N}$.

A **timed input word** is an alternating sequence of delays from $\mathbb{R}^{\geq 0}$ and inputs from $I \setminus \{\mathsf{timeout}\}$, such that the first and last elements are delays. Timed input words are thus elements of $\mathbb{R}^{\geq 0}\ ((I \setminus \{\mathsf{timeout}\})\ \mathbb{R}^{\geq 0})^*$. The $\bullet$ operation for concatenating two timed input words puts the timed input words in sequence, adding the first delay of the second timed input word to the final delay of the first timed input word:

$$(u\ d) \bullet (d'\ u') = u\ (d + d')\ u'.$$

Timed words w can be reduced to timed input words $tiw(w)$ by removing the outputs and the occurrences of $\mathsf{timeout}$, by replacing consecutive times with their sum, and by appending 0 to the end of the sequence in certain cases:

$$tiw(\epsilon) = 0$$

$$tiw((t, i, o)\ w) = \begin{cases} t \bullet tiw(w) & \text{if } i = \mathsf{timeout} \\ (t\ i\ 0) \bullet tiw(w) & \text{otherwise.} \end{cases}$$

Example 2.6.5. Let w be the timed word:

$$w = (12, a, A)\ (0.2, b, B)\ (0.8, \mathsf{timeout}, C)\ (1.1, a, A)\ (8.9, \mathsf{timeout}, E)\ (17, a, A).$$

Then, $tiw(w) = 12\ a\ 0.2\ b\ 1.9\ a\ 25.9\ a\ 0$.

There are two possibilities whenever a timed input word performs a delay that precisely matches the timer's current value, followed by an input i from $I \setminus \{\mathsf{timeout}\}$. There is the possibility that the $\mathsf{timeout}$ is processed first, followed by input i. There is also the possibility that the $\mathsf{timeout}$ is skipped entirely. This difference can cause **race conditions**.

Example 2.6.6. Let $u = 12\ a\ 1\ b$ be a timed input word. Then, there are two possible timed words for the MM1T of Figure 2.1:

$$(12, a, A)\ (1, \mathsf{timeout}, C)\ (0, b, B)$$

and:

$$(12, a, A)\ (1, b, B).$$

Chapter 3

k-A-Complete MM1T Conformance Testing

In this chapter, we introduce our k-A-complete conformance testing method for MM1Ts. We start by specifying the requirements that our method imposes on the specification. Next, we specify what data our test procedure captures. The final section introduces our conformance testing method for MM1Ts.

3.1 Requirements for the Specification

The W-method for Mealy machines requires that the specification Mealy machine is connected and minimal. We do make the same assumptions for our specification MM1Ts. We proceed by defining these notions:

Definition 3.1.1 (Connected MM1T). An MM1T $\mathcal{M}$ is **connected** iff, for each state $q \in Q$ there exists an input sequence $\sigma \in I^*$ such that $\delta^*(q_{\mathcal{I}}, \sigma) = q$.

The definition of untimed equivalence of MM1T states gives rise to a notion of minimal MM1Ts:

Definition 3.1.2 (Minimal MM1T). An MM1T $\mathcal{M}$ is **minimal** iff, for all pairs of states $q_1, q_2 \in Q$, $q_1 \approx_{untimed} q_2$ iff $q_1 = q_2$.

Vaandrager et al. [2023] introduced both a mapping that expresses MM1Ts in terms of Mealy machines, as well as a mapping that can express certain Mealy machines in terms of MM1Ts. We include these two mappings in Appendix A.1. Using these mappings, we can prove that methods for minimizing Mealy machines can be used to minimize MM1Ts:

Lemma 3.1.1. Let $\mathcal{M}$ be an MM1T. Let M be a Mealy machine obtained by minimizing Mealy machine $\mathsf{Mealy}(\mathcal{M})$. Then, $\mathsf{MM1T}(M)$ is a valid MM1T, minimal and untimed equivalent to $\mathcal{M}$.

The proof of Lemma 3.1.1 can be found in Appendix B.2. From this point onwards, we will assume that the specification MM1T is minimal.

We also make the assumption that the specification and the SUT have a known shared upper bound Δ on the value that the timer may be (re)set to in any transition. This assumption implies that waiting for at least Δ time units in any given state would reveal whether the timer is on or off in that state, since a timeout would occur within Δ time units if, and only if, the timer is on.

3.2 The Test Data Captured by our Procedure

Before we can introduce our MM1T testing procedure, we first need to introduce some concepts that our approach heavily relies on. These concepts are based on work from Vaandrager [2024], a paper that introduced a certain sufficient condition for a Mealy machine test suite to be k-complete. In this paper, Vaandrager captures information about the behavior that the specification $\mathcal{S}$ and the SUT $\mathcal{M}$ exhibit in response to the

test suite in an observation tree that represents both $\mathcal{S}$ and $\mathcal{M}$. The sufficient condition requires that the observation tree contains access sequences for all states of the specification, that it contains successors for these states for all inputs up to a depth of $k+1$, and that certain apartness relations hold between the states of the observation tree.

Our testing procedure is inspired by this approach. We introduce this procedure in the next section, in which it will become clear how we use a version of the sufficient condition to prove the validity of our testing method. The method itself is based on versions of the dependencies of the condition that we adapt to an MM1T setting.

We start by defining transition-preserving maps between the states of two (partial) MM1Ts:

Definition 3.2.1 (Functional simulation). Let $\mathcal{T}$ and $\mathcal{M}$ be two (partial) MM1Ts with the same set of inputs I. A **functional simulation** $f\colon \mathcal{T} \to \mathcal{M}$ is a function $f\colon Q^{\mathcal{T}} \to Q^{\mathcal{M}}$ for which:

- $f(q_{\mathcal{I}}^{\mathcal{T}}) = q_{\mathcal{I}}^{\mathcal{M}}$,
- $f(q) \in Q_{on}^{\mathcal{M}} \Leftrightarrow q \in Q_{on}^{\mathcal{T}}$, and
- $q \xrightarrow{i/o,n} q' \Rightarrow f(q) \xrightarrow{i/o,n} f(q')$.

The intuition is that functional simulations preserve the transition structure with its outputs and timer updates, the initial state, and the status of whether the timer is on or off.

Definition 3.2.2. Let $\mathcal{M}$ be an (partial) MM1T. The partial function $uWord_q$ yields the untimed word that starts from state $q \in Q$ for given input sequences $\sigma \in I^*$. We define this function as, for all $i \in I$ and all $\sigma \in I^*$:

$$uWord_q(\epsilon) = \epsilon$$

$$uWord_q(i\ \sigma) = \begin{cases} (i, \lambda(q,i), \tau_\perp(q,i))\ uWord_{\delta(q,i)}(\sigma) & \text{if } \delta(q,i)\downarrow \wedge\ uWord_{\delta(q,i)}(\sigma)\downarrow \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We can see from this definition that $uWord_q(\sigma)\downarrow$ iff $\delta^*(q_{\mathcal{I}}, \sigma)\downarrow$.

Lemma 3.2.1. Let $\mathcal{T}$ and $\mathcal{M}$ be MM1Ts with the same inputs, I. Let $f\colon \mathcal{T} \to \mathcal{M}$ be a functional simulation. Then, for all $q \in Q^{\mathcal{T}}$, $uWord_q^{\mathcal{T}} \subseteq uWord_{f(q)}^{\mathcal{M}}$.

Proof. We prove the property by showing that for all $\sigma \in I^*$ for which $uWord_q^{\mathcal{T}}(\sigma)\downarrow$: $uWord_{f(q)}^{\mathcal{M}}(\sigma)\downarrow$ and $uWord_q^{\mathcal{T}}(\sigma) = uWord_{f(q)}^{\mathcal{M}}(\sigma)$. We use an induction on σ:

- Base case: $\sigma = \epsilon$. Then:

  $$uWord_q^{\mathcal{T}}(\epsilon) = \epsilon = uWord_{f(q)}^{\mathcal{M}}(\epsilon).$$

- Inductive step case: Let $\sigma = \rho\ i$ for some $\rho \in I^*$ and $i \in I$. We use the induction hypothesis: $uWord_q^{\mathcal{T}}(\rho)\downarrow \implies \Big(uWord_{f(q)}^{\mathcal{M}}(\rho)\downarrow \wedge\ uWord_q^{\mathcal{T}}(\rho) = uWord_{f(q)}^{\mathcal{M}}(\rho)\Big)$. Let $q'_\rho = \delta^{\mathcal{T}^*}(q, \rho)$. We will assume that $uWord_q^{\mathcal{T}}(\rho\ i)\downarrow$, because there are no restrictions on $uWord_{f(q)}^{\mathcal{M}}(\rho\ i)$ when $uWord_q^{\mathcal{T}}(\rho\ i)\uparrow$. This assumption tells us that $\delta^{\mathcal{T}^*}(q, \rho\ i)\downarrow$, per the definition of the $uWord$ function. The fact that f is a functional simulation tells us that the existence of the transition:

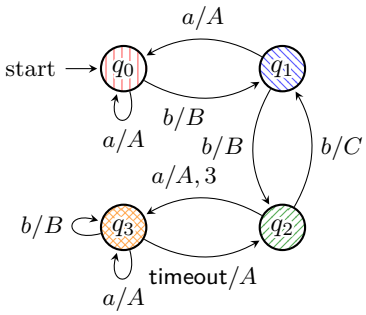
  $$\delta^{\mathcal{T}^*}(q,\rho) \xrightarrow{i/o,n} \delta^{\mathcal{T}^*}(q, \rho\ i)$$

  for some $o \in O^{\mathcal{T}}$ and $n \in (\mathbb{N}^{>0} \cup \{\perp\})$ implies the existence of the transition:

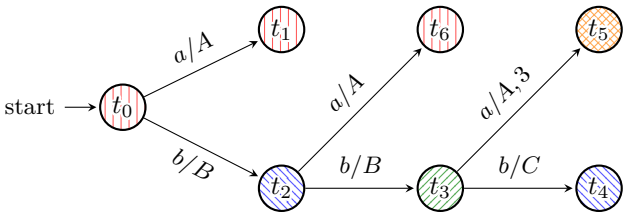
  $$f(\delta^{\mathcal{T}^*}(q,\rho)) \xrightarrow{i/o,n} f(\delta^{\mathcal{T}^*}(q, \rho\ i)).$$

  This tells us that:

  $$\begin{aligned} uWord_q^{\mathcal{T}}(\rho\ i) &= uWord_q^{\mathcal{T}}(\rho)\ (i,o,n) \\ &= uWord_{f(q)}^{\mathcal{M}}(\rho)\ (i,o,n) \qquad (IH) \\ &= uWord_{f(q)}^{\mathcal{M}}(\rho\ i). \end{aligned}$$



(a) An MMIT



(b) An observation tree for the MM1T on the left with $Q_{on} = \{t_5\}$ and $Q_{off} = \{t_0, t_1, t_2, t_3, t_4, t_6\}$

Figure 3.1: An MM1T, along with an observation tree for that MM1T

Hence we have shown that, for all $q \in Q^{\mathcal{T}}$, $uWord_q^{\mathcal{T}} \subseteq uWord_{f(q)}^{\mathcal{M}}$. □

During testing, we capture information about the inputs, outputs, timer updates and the timer's on/off status in a tree-shaped, partial MM1T that we refer to as an observation tree.

Definition 3.2.3 ((Observation) tree MM1T). A partial MM1T $\mathcal{T}$ is a **tree** iff, for every state $q \in Q$ there is a unique input sequence $\mathsf{access}(q) \in I^*$ such that $\delta^*(q_I, \mathsf{access}(q)) = q$. A tree MM1T $\mathcal{T}$ is an **observation tree** for an MM1T $\mathcal{M}$ iff there exists a functional simulation $f : \mathcal{T} \to \mathcal{M}$.

Example 3.2.1. Figure 3.1(b) shows an observation tree for the MM1T of Figure 3.1(a). We colored the states to indicate the functional simulation.

Our testing procedure will try to construct a partial MM1T $\mathcal{T}$ for which there are functional simulations between $\mathcal{T}$ and specification $\mathcal{S}$, and between $\mathcal{T}$ and SUT $\mathcal{M}$. If such a $\mathcal{T}$ cannot be constructed, then we know that $\mathcal{S}$ and $\mathcal{M}$ do not describe the same behavior, which would imply that $\mathcal{M} \not\approx_{untimed} \mathcal{S}$.

Definition 3.2.4 (Apartness for (partial) MM1Ts). Let $\mathcal{T}$ be an (partial) MM1T. States $q_1, q_2 \in Q$ are **apart**, denoted $q_1 \# q_2$, iff there exists an input sequence $\sigma \in I^*$ for which:

- $\delta^*(q_1, \sigma)\downarrow$, $\delta^*(q_2, \sigma)\downarrow$, and $uWord_{q_1}(\sigma) \neq uWord_{q_2}(\sigma)$; or
- $\delta^*(q_1, \sigma)\downarrow$, $\delta^*(q_2, \sigma)\downarrow$, and $\delta^*(q_1, \sigma) \in Q_{on} \Leftrightarrow \delta^*(q_2, \sigma) \in Q_{off}$.

We then call σ a **witness** of $q_1 \# q_2$, which we denote by $\sigma \vdash q_1 \# q_2$.

The first condition is used to tells states q_1 and q_2 apart based on whether they have different outputs or timer updates at the same index along their untimed runs for any input sequence σ for which $\delta^*(q_1, \sigma)\downarrow$ and $\delta^*(q_2, \sigma)\downarrow$. This condition cannot be used to determine whether there is an input sequence σ for which $\delta^*(q_1, \sigma) \in Q_{on} \Leftrightarrow \delta^*(q_1, \sigma) \in Q_{off}$, since if $\delta^*(q_1, \sigma) \in Q_{on} \wedge \delta^*(q_1, \sigma) \in Q_{off}$, then $\delta^*(q_1, \sigma\ \mathsf{timeout})\downarrow$ and $\delta^*(q_2, \sigma\ \mathsf{timeout})\uparrow$. Furthermore, $\delta^*(q_1, \sigma) \in Q_{on} \Leftrightarrow \delta^*(q_1, \sigma) \in Q_{off}$ can hold even if $\delta^*(q_1, \sigma)\downarrow$, $\delta^*(q_2, \sigma)\downarrow$, and $uWord_{q_1}(\sigma) \neq uWord_{q_2}(\sigma)$. We therefore include the second condition, which allows us to directly conclude that $q_1 \# q_2$ when $\delta^*(q_1, \sigma)\downarrow$, $\delta^*(q_2, \sigma)\downarrow$, and $\delta^*(q_1, \sigma) \in Q_{on} \Leftrightarrow \delta^*(q_2, \sigma) \in Q_{off}$.

Example 3.2.2. Let $\mathcal{M}$ be the MM1T of Figure 3.1(a), and let $\mathcal{T}$ be the observation tree MM1T for $\mathcal{M}$ from Figure 3.1(b). These are some of the apartness relations between states of $\mathcal{T}$:

$$b \vdash t_2 \# t_3 \qquad a \vdash t_2 \# t_3 \qquad \epsilon \vdash t_3 \# t_5 \qquad b\ a \vdash t_0 \# t_2 \qquad b\ b \vdash t_0 \# t_2.$$

States t_2 and t_3 are apart both because they have different output symbols for input b, and because they have different timer resets for input a. States t_3 and t_5 are apart because the timer is off in the former, but not in the latter. Finally, states t_0 and t_2 are apart because for their b-transition, they reach the states t_2 and t_3, which are themselves apart for the two reasons we discussed.

These apartness relations for states of $\mathcal{T}$ are matched by the following apartness relations between their corresponding states of $\mathcal{M}$:

$$b \vdash q_1 \# q_2 \qquad a \vdash q_1 \# q_2 \qquad \epsilon \vdash q_2 \# q_3 \qquad b\ a \vdash q_0 \# q_1 \qquad b\ b \vdash q_0 \# q_1.$$

These apartness relations hold for the same reasons for which their corresponding apartness relations in $\mathcal{T}$ hold.

Example 3.2.2 alludes to a second way to characterize apartness for (partial) MM1Ts:

Definition 3.2.5 (Deduction-based definition of apartness for (partial) MM1Ts). Let $\mathcal{M}$ be a (partial) MM1T. States $q_1, q_2 \in Q$ are **apart**, denoted $q_1 \# q_2$, iff any of the following conditions holds:

$$\frac{q_1 \in Q_{on} \quad q_2 \in Q_{off}}{\epsilon \vdash q_1 \# q_2} \qquad \frac{q_1 \xrightarrow{i/o_1,n_1} q_1' \quad q_2 \xrightarrow{i/o_2,n_2} q_2' \quad (o_1 \neq o_2 \vee n_1 \neq n_2)}{i \vdash q_1 \# q_2}$$

$$\frac{\delta(q_1,i)\downarrow \quad \delta(q_2,i)\downarrow \quad \sigma \vdash \delta(q_1,i) \# \delta(q_2,i)}{i\ \sigma \vdash q_1 \# q_2}$$

where $i \in I$, $\sigma \in I^*$, $o_1, o_2 \in O$, and $n_1, n_2 \in \mathbb{N}^{>0} \cup \{\bot\}$.

Note that all three of the definitions of apartness that we have introduced so far in this chapter are irreflexive and symmetric.

Lemma 3.2.2. Definition 3.2.4 and Definition 3.2.5 induce the same notion of apartness for (partial) MM1Ts.

Proof. Let $\#^e$ denote the apartness relation of Definition 3.2.4, and let $\#^d$ denote the apartness relation of Definition 3.2.5. Let $\mathcal{M}$ be a (partial) MM1T, and let $q_1, q_2 \in Q$ be two states of $\mathcal{M}$.

We prove the that for all $\sigma \in I^*$: $\sigma \vdash q_1 \#^e q_2 \Leftrightarrow \sigma \vdash q_1 \#^d q_2$:

- We show that for all $\sigma \in I^*$: $\sigma \vdash q_1 \#^e q_2 \Rightarrow \sigma \vdash q_1 \#^e q_2$. We perform a case distinction on the two conditions under which $\sigma \vdash q_1 \#^e q_2$:
  - If $\delta^*(q_1,\sigma)\downarrow$, $\delta^*(q_2,\sigma)\downarrow$, and $uWord_{q_1}(\sigma) \neq uWord_{q_2}(\sigma)$, then there exists a maximum-length prefix ρ of σ such that $\delta^*(q_1,\rho)\downarrow$, $\delta^*(q_2,\rho)\downarrow$, and $uWord_{q_1}(\rho) = uWord_{q_2}(\rho)$. Let ρ be this prefix of σ, and let $i \in I$ be the input such that $\rho\ i$ is a prefix of σ. Let $q_1' = \delta^*(q_1,\rho)$ and $q_2' = \delta^*(q_2,\rho)$. We then know that:

    $$(\lambda(q_1',i) = o_1 \neq o_2 = \lambda(q_2',i)) \qquad \vee \qquad (\tau_\bot(q_1',i) = n_1 \neq n_2 = \tau_\bot(q_2',i)).$$

    Let $q_1'' = \delta(q_1',i)$ and $q_2'' = \delta(q_2',i)$. Definition 3.2.5 tells us that:

    $$\frac{q_1' \xrightarrow{i/o_1,n_1} q_1'' \quad q_2' \xrightarrow{i/o_2,n_2} q_2'' \quad (o_1 \neq o_2 \vee n_1 \neq n_2)}{i \vdash q_1' \#^d q_2'},$$

    which tells us that $i \vdash q_1' \#^d q_2'$. Repeated use of Definition 3.2.5's final rule would now reveal that $\rho\ i \vdash q_1 \#^d q_2$. This implies that $\sigma \vdash q_1 \#^d q_2$, as required.
  - If $\delta^*(q_1,\sigma)\downarrow$, $\delta^*(q_2,\sigma)\downarrow$, and $\delta^*(q_1,\sigma) \in Q_{on} \Leftrightarrow \delta^*(q_2,\sigma) \in Q_{off}$. Let $q_1' = \delta^*(q_1,\sigma)$ and $q_2' = \delta^*(q_2,\sigma)$. Definition 3.2.5 tells us that:

    $$\frac{q_1' \in Q_{on} \quad q_2' \in Q_{off}}{\epsilon \vdash q_1' \#^d q_2'} \qquad \vee \qquad \frac{q_2' \in Q_{on} \quad q_1' \in Q_{off}}{\epsilon \vdash q_2' \#^d q_1'}.$$

    The fact that the apartness defined in Definition 3.2.5 is symmetric implies that in both cases, $\epsilon \vdash q_1' \#^d q_2'$. Repeated use of Definition 3.2.5's final rule would now reveal that $\sigma \vdash q_1 \#^d q_2$, as required.

We have thus shown that for all $\sigma \in I^*$: $\sigma \vdash q_1 \#^e q_2 \Rightarrow \sigma \vdash q_1 \#^d q_2$.

- We show by induction on the length of input sequence σ that for all $\sigma \in I^*$: $\sigma \vdash q_1 \#^d q_2 \Rightarrow \sigma \vdash q_1 \#^e q_2$:
  1. Base case 1: $\epsilon \vdash q_1 \#^d q_2$. Then Definition 3.2.5 tells us that $q_1 \in Q_{on}$ and $q_2 \in Q_{off}$. Since $\delta^*(q_1, \epsilon)\downarrow$, $\delta^*(q_2, \epsilon)\downarrow$, $\delta^*(q_1, \epsilon) \in Q_{on}$ and $\delta^*(q_2, \epsilon) \in Q_{off}$, the second condition of Definition 3.2.4 tells us that $\epsilon \vdash q_1 \#^e q_2$, as required.
  2. Base case 2: $i \vdash q_1 \#^d q_2$ with $i \in I$. Then Definition 3.2.5 tells us that:

     $$q_1 \xrightarrow{i/o_1,n_1} q_1' \quad \wedge \quad q_2 \xrightarrow{i/o_2,n_2} q_2' \quad \wedge \quad (o_1 \neq o_2 \vee n_1 \neq n_2).$$

     Since $\delta^*(q_1, i)\downarrow$, $\delta^*(q_2, i)\downarrow$, and $uWord_{q_1}(i) \neq uWord_{q_2}(i)$, the second condition of Definition 3.2.4 tells us that $i \vdash q_1 \#^e q_2$, as required.
  3. Inductive step case: $i\ \rho \vdash q_1 \#^d q_2$ with $i \in I$ and $\rho \in I^*$. Then Definition 3.2.5 tells us that $\delta(q_1, i)\downarrow$, $\delta(q_2, i)\downarrow$ and $\rho \vdash \delta(q_1, i) \#^d \delta(q_2, i)$. We use the induction hypothesis:

     $$\rho \vdash p_1 \#^d p_2 \implies \rho \vdash p_1 \#^e p_2.$$

     Applying the induction hypothesis to $\rho \vdash \delta(q_1, i) \#^d \delta(q_2, i)$ yields $\rho \vdash \delta(q_1, i) \#^e \delta(q_2, i)$. Definition 3.2.4 tells us that there are two conditions under which this apartness could hold:
     - $\delta^*(\delta(q_1, i), \rho)\downarrow$, $\delta^*(\delta(q_2, i), \rho)\downarrow$, and $uWord_{\delta(q_1,i)}(\rho) \neq uWord_{\delta(q_2,i)}(\rho)$. Then $\delta^*(q_1, i\ \rho)\downarrow$, $\delta^*(q_2, i\ \rho)\downarrow$, and $uWord_{q_1}(i\ \rho) \neq uWord_{q_2}(i\ \rho)$. This tells us that $i\ \rho \vdash q_1 \#^e q_2$, as required.
     - $\delta^*(\delta(q_1, i), \rho)\downarrow$, $\delta^*(\delta(q_2, i), \rho)\downarrow$, and $\delta^*(\delta(q_1, i), \rho) \in Q_{on} \Leftrightarrow \delta^*(\delta(q_2, i), \rho) \in Q_{off}$. We then know that $\delta^*(q_1, i\ \rho)\downarrow$, $\delta^*(q_2, i\ \rho)\downarrow$, and $\delta^*(q_1, i\ \rho) \in Q_{on} \Leftrightarrow \delta^*(q_2, i\ \rho) \in Q_{off}$. This tells us that $i\ \rho \vdash q_1 \#^e q_2$, as required.

     We thus know that in all cases, $i\ \sigma \vdash q_1 \#^d q_2$ with $i \in I$ and $\sigma \in I^*$ implies that $i\ \sigma \vdash q_1 \#^e q_2$.

  We have thus shown that for all $\sigma \in I^*$: $\sigma \vdash q_1 \#^d q_2 \Rightarrow \sigma \vdash q_1 \#^e q_2$.

We have shown that for all $\sigma \in I^*$: $\sigma \vdash q_1 \#^e q_2 \Leftrightarrow \sigma \vdash q_1 \#^d q_2$. □

Lemma 3.2.2 tells us that both definitions of apartness for (partial) MM1T describe the same notion of apartness. We can show that this notion of apartness satisfies a property known as weak co-transitivity:

Lemma 3.2.3 (Weak co-transitivity for observation tree MM1T apartness). Let $\mathcal{T}$ be an observation tree MM1T. Then, for all $r, r', q \in Q$, and all $\sigma \in I^*$:

$$\sigma \vdash r \# r' \quad \wedge \quad \delta^*(q, \sigma)\downarrow \quad \implies \quad \sigma \vdash q \# r \quad \vee \quad \sigma \vdash q \# r'$$

Proof. The fact that $\delta^*(q, \sigma)\downarrow$ tells us that $uWord_q(\sigma)\downarrow$. We perform a case distinction on the two possible conditions under which $\sigma \vdash r \# r'$:

- in case apartness follows from $uWord_r(\sigma) \neq uWord_{r'}(\sigma)$, we get:

  $$uWord_q(\sigma) \neq uWord_r(\sigma) \quad \vee \quad uWord_q(\sigma) \neq uWord_{r'}(\sigma)$$

  which tells us that:

  $$\sigma \vdash q \# r \quad \vee \quad \sigma \vdash q \# r'.$$

- otherwise, the timer is either on in $\delta^*(r, \sigma)$ and off in $\delta^*(r', \sigma)$, or off in $\delta^*(r, \sigma)$ and on in $\delta^*(r', \sigma)$. We thus know that whether the timer is on or off in $\delta^{s*}(q, \sigma)$, the opposite is true for either $\delta^*(r, \sigma)$ or $\delta^*(r', \sigma)$. Therefore, $\sigma \vdash q \# r$ or $\sigma \vdash q \# r'$.

We thus know that in any case, $\sigma \vdash r \# r'$ and $\delta^*(q, \sigma)\downarrow$ imply that $\sigma \vdash q \# r \vee \sigma \vdash q \# r'$, for all $r, r', q \in Q$, and all $\sigma \in I^*$. □

We can show that functional simulations can never map apart states to states that are untimed equivalent:

Lemma 3.2.4. Let $\mathcal{T}$ be an observation tree MM1T for MM1T $\mathcal{M}$, and let $f\colon \mathcal{T} \to \mathcal{M}$ be a functional simulation. Then, for all $q_1, q_2 \in Q^{\mathcal{T}}$:

$$q_1 \# q_2 \text{ in } \mathcal{T} \quad \Longrightarrow \quad f(q_1) \not\approx_{untimed} f(q_2) \text{ in } \mathcal{M}.$$

Proof. The existence of the apartness $q_1 \# q_2$ implies that there exists an input sequence $\sigma \in I^*$ such that $\sigma \vdash q_1 \# q_2$. The definition of apartness for (partial) MM1Ts tells us that there could be two conditions under which $q_1 \# q_2$. We perform a case distinction on these two conditions:

- if $\delta^{\mathcal{T}^*}(q_1, \sigma)\downarrow$, $\delta^{\mathcal{T}^*}(q_2, \sigma)\downarrow$ and $uWord^{\mathcal{T}}_{q_1}(\sigma) \neq uWord^{\mathcal{T}}_{q_2}(\sigma)$, we know that $uWord^{\mathcal{T}}_{q_1}(\sigma)\downarrow$ and $uWord^{\mathcal{T}}_{q_2}(\sigma)\downarrow$. Lemma 3.2.1 now tells us that:

  $$uWord^{\mathcal{T}}_{q_1}(\sigma)\downarrow \quad \Longrightarrow \quad uWord^{\mathcal{M}}_{f(q_1)}(\sigma)\downarrow \qquad \wedge \qquad uWord^{\mathcal{T}}_{q_2}(\sigma)\downarrow \quad \Longrightarrow \quad uWord^{\mathcal{M}}_{f(q_2)}(\sigma)\downarrow.$$

  Lemma 3.2.1 also tells us that $uWord^{\mathcal{M}}_{f(q_1)}(\sigma) = uWord^{\mathcal{T}}_{q_1}(\sigma)$, and that $uWord^{\mathcal{M}}_{f(q_2)}(\sigma) = uWord^{\mathcal{T}}_{q_2}(\sigma)$. Therefore:

  $$uWord^{\mathcal{M}}_{f(q_1)}(\sigma) = uWord^{\mathcal{T}}_{q_1}(\sigma) \neq uWord^{\mathcal{T}}_{q_2}(\sigma) = uWord^{\mathcal{M}}_{f(q_2)}(\sigma).$$

  Thus, $uWord^{\mathcal{M}}_{f(q_1)} \neq uWord^{\mathcal{M}}_{f(q_2)}$;

- otherwise, we know that either $\delta^*(q_1, \sigma) \in Q^{\mathcal{T}}_{on}$ and $\delta^*(q_2, \sigma) \in Q^{\mathcal{T}}_{off}$, or $\delta^*(q_1, \sigma) \in Q^{\mathcal{T}}_{off}$ and $\delta^*(q_2, \sigma) \in Q^{\mathcal{T}}_{on}$. In the first case, we can conclude that since $\delta^*(q_1, \sigma) \in Q^{\mathcal{T}}_{on}$, $uWord_{q_1}(\sigma\ \mathsf{timeout})\downarrow$. Lemma 3.2.1 then tells us that $uWord^{\mathcal{M}}_{f(q_1)}(\sigma\ \mathsf{timeout})\downarrow$. On the other hand, the fact that $\delta^*(q_2, \sigma) \in Q^{\mathcal{T}}_{off}$ tells us that $uWord^{\mathcal{T}}_{q_2}(\sigma\ \mathsf{timeout})\uparrow$. We could show by an inductive argument that $\delta^*(q_2, \sigma) \in Q^{\mathcal{T}}_{off}$ implies that $\delta^*(f(q_2), \sigma) \in Q^{\mathcal{M}}_{off}$. Thus, $uWord^{\mathcal{M}}_{f(q_2)}(\sigma\ \mathsf{timeout})\uparrow$. Therefore, in case $\delta^*(q_1, \sigma) \in Q^{\mathcal{T}}_{on}$ and $\delta^*(q_2, \sigma) \in Q^{\mathcal{T}}_{off}$, $uWord^{\mathcal{M}}_{f(q_1)} \neq uWord^{\mathcal{M}}_{f(q_2)}$.
  We can make a similar argument to show that in case $\delta^*(q_1, \sigma) \in Q^{\mathcal{T}}_{off}$ and $\delta^*(q_2, \sigma) \in Q^{\mathcal{T}}_{on}$, $uWord^{\mathcal{M}}_{f(q_1)} \neq uWord^{\mathcal{M}}_{f(q_2)}$.

We thus know that since $q_1 \# q_2$, $uWord^{\mathcal{M}}_{f(q_1)} \neq uWord^{\mathcal{M}}_{f(q_2)}$. Our notion of untimed MM1T state equivalence tells us that therefore, $f(q_1) \not\approx_{untimed} f(q_2)$. □

We also need the following auxiliary notion:

Definition 3.2.6 (Stratification). Let $\mathcal{M}$ be an MM1T with a set of inputs I, and let $\mathcal{T}$ be an observation tree for $\mathcal{M}$. Then $I^{\mathcal{T}} = I^{\mathcal{M}}$. Let $A \subseteq I^*$ be a nonempty, finite, prefix closed set of input sequences. Then A induces a **stratification** of $Q^{\mathcal{T}}$ as follows:

1. A state q of $\mathcal{T}$ is called a **basis state** iff $\mathsf{access}(q) \in A$. We write B to denote the set of basis states: $B \coloneqq \{q \in Q^{\mathcal{T}} \mid \mathsf{access}(q) \in A\}$. Note that, since A is nonempty and prefix closed, initial state $q^{\mathcal{T}}_{\mathcal{I}}$ is in the basis, and all states on the path leading to a basis state are basis states as well.

2. We write F^0 for the set of immediate successors of basis states that are not basis states themselves: $F^0 \coloneqq \{q' \in Q^{\mathcal{T}} \setminus B \mid \exists q \in B, i \in I\colon q' = \delta^{\mathcal{T}}(q, i)\}$. We refer to F^0 as the **0-level frontier**.

3. For $k > 0$, the ***k*-level frontier** F^k is the set of immediate successors of $k-1$-level frontier states: $F^k \coloneqq \{q' \in Q^{\mathcal{T}} \mid \exists q \in F^{k-1}, i \in I\colon q' = \delta^{\mathcal{T}}(q, i)\}$.

We often use $F^{<k}$ to denote the set $F^0 \cup \cdots \cup F^{k-1}$ of the states in the first k frontiers, and $F^{\leq k}$ to denote the set $F^0 \cup \cdots \cup F^k$ of all states in the first $k+1$ frontiers.
We say that basis B is **complete** if:

- for each $\sigma \in A$ there is a state $q \in B$ with $\delta^{\mathcal{T}^*}(q^{\mathcal{T}}_{\mathcal{I}}, \sigma) = q$, and if
- for each $q \in B$ and each $i \in (I \setminus \{\mathsf{timeout}\}) \cup \{\mathsf{timeout} \mid q \in Q^{\mathcal{T}}_{on}\}$, $\delta^{\mathcal{T}}(q, i)\downarrow$.

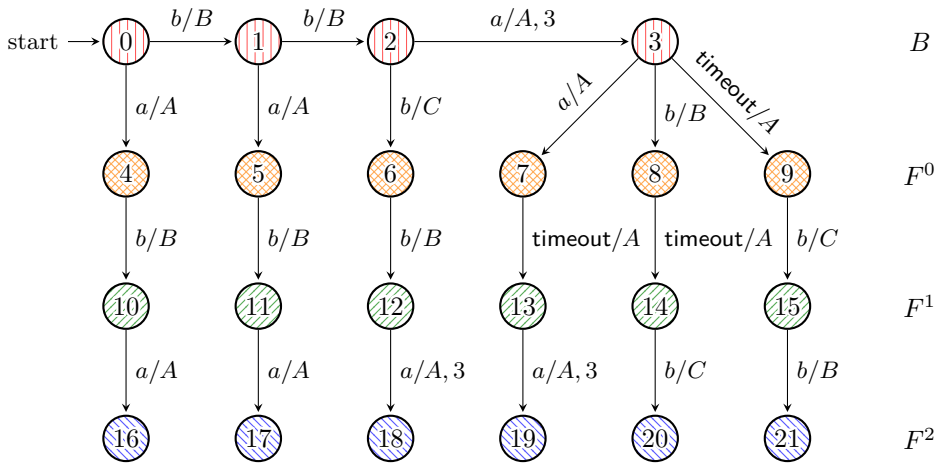


Figure 3.2: A stratification of an observation tree for the MM1T of Figure 3.1(a), induced by $A = \{\epsilon, b, b\ b, b\ b\ a\}$

For $k \in \mathbb{N}$, the k-level frontier is **complete** if for each $q \in F^k$ and each $i \in (I \backslash \{\mathsf{timeout}\}) \cup \{\mathsf{timeout} \mid q \in Q^{\mathcal{T}}_{on}\}$, $\delta^{\mathcal{T}}(q, i)\downarrow$.

For each state $q \in Q^{\mathcal{T}}$ we define the **candidate set** $\mathcal{C}(q)$ as the set of basis states that are not apart from q: $\mathcal{C}(q) \coloneqq \{q' \in B \mid \neg(q \# q')\}$. A state $q \in Q^{\mathcal{T}}$ is **identified** if its candidate set is singleton.

Example 3.2.3. Figure 3.2 shows a stratification of an observation tree for the MM1T of Figure 3.1(a). We colored the states to indicate the basis and the first three frontiers. The stratification's basis is complete, but its 0, 1 and 2-level frontiers are not.

We now have everything we need to be able to define our MM1T testing procedure.

3.3 The Testing Procedure

Our testing procedure is inspired by the apartness-based method for learning Mealy machines introduced in Vaandrager et al. [2022], as well as by the apartness-based perspective on conformance testing of Mealy machines that is provided in Vaandrager [2024]. The procedure also strongly resembles the H-method [Dorofeeva et al., 2005] for computing k-complete test suites for Mealy machines. We discuss the similarities and differences between the H-method and our testing procedure in Section 3.3.7. In our method, we nondeterministically expand a tree MM1T that functions as an observation tree for both the specification $\mathcal{S}$ and the SUT $\mathcal{M}$, until we can either conclude that $\mathcal{M} \approx_{untimed} \mathcal{S}$, or we discover that $\mathcal{M} \not\approx_{untimed} \mathcal{S}$.

Our testing procedure returns a counterexample input sequence in case $\mathcal{M} \not\approx_{untimed} \mathcal{S}$. We assume that the specification MM1T $\mathcal{S}$ is minimal, connected and complete.

Algorithm 1 shows the main testing procedure.

Algorithm 1: Procedure for testing MM1Ts

$\mathcal{T} \leftarrow$ a fresh, partial MM1T with an initial state $q_{\mathcal{I}}^{\mathcal{T}}$;
$B \leftarrow \{q_{\mathcal{I}}^{\mathcal{T}}\}$;
for $\sigma \in C$ **do**
  $c \leftarrow addTransitions_{\mathcal{M}}^{\mathcal{S}}(q_{\mathcal{I}}^{\mathcal{T}}, \sigma)$;
  if $c \in I^*$ **then return** c;
  $B \leftarrow B \cup \{\delta^{\mathcal{T}^*}(q_{\mathcal{I}}^{\mathcal{T}}, \sigma)\}$;
end for
while *any of the rules can still be applied* **do**
  [] $\neg(r \# r')$, *for some* $r, r' \in B$ *for which* $r \neq r' \rightarrow$ ▷ `Rule (IdentifyBasisStates)`
    $c \leftarrow makeObsTreeStatesApart^{\mathcal{S}}(r, r')$;
    if $c \in I^*$ **then return** c;
  [] $\delta^{\mathcal{T}}(q, i)\uparrow$ *and* $\delta^{\mathcal{S}^*}(s_{\mathcal{I}}^{\mathcal{S}}, \mathsf{access}(q)\ i)\downarrow$, *for some* $q \in B \cup F^{<k}$ *and* $i \in I \rightarrow$ ▷ `Rule (ExtendFrontiers)`
    $c \leftarrow addTransition_{\mathcal{M}}^{\mathcal{S}}(q, i)$;
    if $c \in I^*$ **then return** c;
  [] $r \# r' \wedge \neg(r \# t) \wedge \neg(r' \# t)$, *for some* $r, r' \in B$ *and* $t \in F^k \rightarrow$ ▷ `Rule (IdentifyFrontiers)`
    $\sigma \leftarrow$ a witness of $r \# r'$;
    $c \leftarrow addTransitions_{\mathcal{M}}^{\mathcal{S}}(t, \sigma)$;
    if $c \in I^*$ **then return** c;
  [] $r \# t' \wedge \neg(r \# t'') \wedge \neg(t' \# t'')$, *for some* $r \in B$, $t' \in F^k$ *and* $t'' \in F^{<k} \rightarrow$ ▷ `Rule (ExtendCoTransitivity)`
    $\sigma \leftarrow$ a witness of $r \# t'$;
    $c \leftarrow addTransitions_{\mathcal{M}}^{\mathcal{S}}(t'', \sigma)$;
    if $c \in I^*$ **then return** c;
end
return `yes`;

Algorithm 2: Sub-procedure for making two observation tree states apart when their specification counterparts are apart

Procedure $makeObsTreeStatesApart^{\mathcal{S}}(q, q' \in Q^{\mathcal{T}})$**:**
  if $q \# q'$ **then**
    return;
  end
  $s \leftarrow \delta^{\mathcal{S}^*}(s_{\mathcal{I}}^{\mathcal{S}}, \mathsf{access}(q))$;
  $s' \leftarrow \delta^{\mathcal{S}^*}(s_{\mathcal{I}}^{\mathcal{S}}, \mathsf{access}(q'))$;
  $\sigma \leftarrow$ a witness of $s \# s'$;
  $c \leftarrow addTransitions_{\mathcal{M}}^{\mathcal{S}}(q, \sigma)$;
  if $c \in I^*$ **then return** c;
  $c' \leftarrow addTransitions_{\mathcal{M}}^{\mathcal{S}}(q', \sigma)$;
  if $c' \in I^*$ **then return** c';

We start the procedure by computing a prefix-closed state cover C for the specification. Such a state cover could be computed via a simple breadth-first search [Zuse, 1972]. The procedure maintains an observation tree $\mathcal{T}$ that is valid for both the specification, and the SUT. It tracks a stratification induced by C. The observation tree initially contains only an initial state. This state is indeed in the basis, since C being a state cover implies that $\epsilon \in C$. The next step is to complete the basis by adding transitions for all input sequences of C to $\mathcal{T}$.
The procedure then proceeds to non-deterministically extend $\mathcal{T}$. It keeps adding transitions to $\mathcal{T}$, until the basis and the first $k+1$ frontiers of the stratification induced by C are complete, certain states are identified and a certain apartness relations hold between its states. The testing procedure terminates and returns a counterexample if it discovers a conflict between the specification and the SUT during its operation.

Whenever we extend the observation tree with one or more transitions, there is always the possibility that we discover a conflict in the outputs, timer updates or the on/off status of the successor states between the specification and the SUT's counterparts what would be the new observation tree transition. We terminate the testing procedure and yield a counterexample if we discover such a conflict. We explain the procedure for adding one or more transitions to the observation tree in subsections 3.3.2 and 3.3.3, respectively.
The algorithm non-deterministically chooses between the following four rules, and it terminates once none of these rules can be applied any longer:

Rule (IdentifyBasisStates) Let $r, r' \in B$ be two distinct basis states that are not pairwise apart. We know that since state cover C is prefix closed, r and r' represent different states of the specification. Let s be the specification state that corresponds to r, and let s' be the specification state that corresponds to r'. Then, since $s \neq s'$, we know from the fact that $\mathcal{S}$ is minimal that $uWord_s(\rho) \neq uWord_{s'}(\rho)$, and thus that $\rho \vdash s \# s'$ for some input sequence $\rho \in I^*$. We use the procedure from Algorithm 2 to make r and r' apart in $\mathcal{T}$. This procedure first finds an input sequence σ such that $\sigma \vdash s \# s'$, $\delta^{\mathcal{S}}(s, \sigma)\downarrow$ and $\delta^{\mathcal{S}^*}(s', \sigma)\downarrow$. It does so by looking for an input sequence for which s and s' either exhibit different output or timer update behavior, or that terminate in specification states that would be immediately apart from one another. The procedure then adds σ from both r and r' to make $\sigma \vdash r \# r'$.

Rule (ExtendFrontiers) Let $q \in B \cup F^{<k}$ be a state that can reach one of the first $k+1$ frontiers in a single transition step. If q has no outgoing transition for an input $i \in I^{\mathcal{S}}$ for which $\delta^{\mathcal{S}^*}(s_{\mathcal{I}}, \mathsf{access}(q)\ i)\downarrow$, i.e. the corresponding state of the specification does have an outgoing transition for i, then we add a transition for i to $\mathcal{T}$. In doing so, we extend the frontier that q's new i-transition transitions into.

Rule (IdentifyFrontiers) When a state $t \in F^k$ from the $k+1$-level frontier is not identified, then there are at least two distinct basis states $r, r' \in B$ from which t is not apart. If $\sigma \vdash r \# r'$, then we use weak co-transitivity to make t apart from r, from r', or from both r and r', i.e. $\sigma \vdash t \# r \ \vee \ \sigma \vdash t \# r'$.

Rule (ExtendCoTransitivity) When a basis state $r \in B$ and a state $t' \in F^k$ from the $k+1$-level frontier are apart, then we will make it so that any state $t'' \in F^{<k}$ from the first k frontiers is apart from r, from t', or from both r and t'.
If either $r \# t''$ or $t' \# t''$ already holds, then there is nothing to be done.
If $r \# t'$, $\neg(r \# t'')$ and $\neg(t' \# t'')$, then we simply use co-transitivity to make it so that $r \# t'' \vee t' \# t''$. We do so by adding the transitions for a witness σ of $r \# t'$ from t''.

We will now discuss the various sub-procedures that we use in Algorithm 1.

3.3.1 Determining Whether a Transition has Conflicts Between the Specification and the SUT

Vaandrager et al. [2023] introduced a method for extending an observation tree for a black-box MM1T with a single transition. This method works by composing a timed input word that it then submits to an MM1T teacher following the MAT framework for active model learning. A careful choice of the final delay causes the timed input word to reveal whether the timer is updated, with what constant it is then updated, and whether the timer is on or off in the new state. We use the same approach not to extend observation tree $\mathcal{T}$ directly, but to obtain for a given observation tree state $q \in Q^{\mathcal{T}}$ and input $i \in I^{\mathcal{T}}$ the triple:

$$\mathit{obtainTheSUTOutputTimerUpdateAndTargetStateOnOffStatus}^{\mathcal{M}}_{\mathcal{T}}(q, i) = (o, n, b),$$

where $o = \delta^{\mathcal{M}}(q^{\mathcal{M}}, i)$, $n = \tau^{\mathcal{M}}_{\perp}(q^{\mathcal{M}}, i)$, and $b = \texttt{yes} \Leftrightarrow \delta^{\mathcal{M}}(q^{\mathcal{M}}, i) \in Q^{\mathcal{M}}_{on}$ with $q^{\mathcal{M}} = \delta^{\mathcal{M}^*}(q^{\mathcal{M}}_{\mathcal{I}}, \mathsf{access}(q))$.

Sub-procedure Algorithm 3 describes the way we use this triple to determine whether the specification and SUT transitions that (would) correspond to the observation tree transition (q, i) have a conflict in their outputs, timer updates and/or the on/off status of their respective successor states.

Algorithm 3: Procedure that determines whether the specification and SUT transitions that (would) correspond to this observation tree transition have a conflict in their outputs, timer updates and/or the on/off status of their respective successor states

```
1  Procedure transitionHasConflicts(q ∈ Q^T, i ∈ I^S):
2      s ← δ^{S*}(s^S_I, access(q));
3      s' ← δ^S(s, i);
       // Obtain the output, timer update and timer status of the SUT's corresponding
          transition
4      (o, n, b) ← obtainTheSUTOutputTimerUpdateAndTargetStateOnOffStatus^M_T(q, i);
       // Determine whether this information conforms to that of the specification's
          corresponding transition
5      if o ≠ λ^S(s, i) ∨ n ≠ τ^S_⊥(s, i) then
6          return yes;
7      end
8      if (b = yes ⇔ s' ∉ Q^S_on) then
9          return yes;
10     end
11     return no;
```

The computation of these triples is the only place where our procedure touches on the timed MM1T semantics. The remainder of the procedure only deals with the untimed semantics.

3.3.2 Extending the Observation Tree With a Single Transition

Algorithm 4 describes our method for extending the observation tree with a new transition. The procedure begins by checking whether the counterpart of the requested transition exists in the specification. If not, then the procedure simply terminates, returning the input sequence $\mathsf{access}(q)\ i$ as the counterexample. In fact, this input sequence always serves as the counterexample, since we yield a counterexample iff we encounter a

conflict in the outputs, timer updates or the on/off status of the successor state for the i-transition.
If the specification does have a counterpart for the i-transition, then we check whether there is a conflict between the transition's counterparts in the specification and the SUT. We add the new transition to the observation tree if there is no such error. If there is, then we return the counterexample.

Algorithm 4: Sub-procedure for extending the observation tree with a single transition

Procedure $addTransition^{\mathcal{S}}_{\mathcal{M}}(q \in Q^{\mathcal{T}}, i \in I^{\mathcal{S}})$**:**
  $s \leftarrow \delta^{\mathcal{S}^*}(s^{\mathcal{S}}_{\mathcal{I}}, \mathsf{access}(q))$;
  // Don't add the transition from q if the specification doesn't have a transition for i from its corresponding state s
  if $\delta^{\mathcal{S}}(s,i)\uparrow$ **then**
    return $\mathsf{access}(q)$ i;
  end
  // Don't add the transition if doing so would result in $\mathcal{T}$ no longer being an observation tree for both the specification and the SUT
  if $transitionHasConflicts(q,i) =$ yes **then**
    return $\mathsf{access}(q)$ i;
  end
  /* Since there is no conflict between the transitions for the specification and the SUT, we can simply read the output, timer update and timer status information from the specification's transition and then add that to $\mathcal{T}$. */
  // Create a fresh observation tree state
  $q' \leftarrow$ a fresh MM1T state;
  // Mark the new state as on iff $\delta^{\mathcal{S}}(s,i)$ is on
  if $\delta^{\mathcal{S}}(s,i) \in Q^{\mathcal{S}}_{on}$ **then**
    $Q^{\mathcal{T}}_{on} \leftarrow Q^{\mathcal{T}}_{on} \cup \{q'\}$;
  end
  else
    $Q^{\mathcal{T}}_{off} \leftarrow Q^{\mathcal{T}}_{off} \cup \{q'\}$;
  end
  // Extend the input and output sets with those use in the specification's corresponding transition
  $I^{\mathcal{T}} \leftarrow I^{\mathcal{T}} \cup \{i\}$;
  $O^{\mathcal{T}} \leftarrow O^{\mathcal{T}} \cup \{\lambda^{\mathcal{S}}(s,i)\}$;
  // Record the new state transition
  $\delta^{\mathcal{T}}(q,i) \leftarrow q'$;
  // Use the output from the specification's corresponding transition
  $\lambda^{\mathcal{T}}(q,i) \leftarrow \lambda^{\mathcal{S}}(s,i)$;
  // Use the same timer update as the specification's corresponding transition
  if $\tau^{\mathcal{S}}(s,i)\downarrow$ **then**
    $\tau^{\mathcal{T}}(q,i) \leftarrow \tau^{\mathcal{S}}(s,i)$;
  end
  return yes;

3.3.3 Extending the Observation Tree With a Sequence of Transitions

Algorithm 5 shows a sub-procedure that extends the observation tree with all transitions induced by a given input sequence. The procedure stops in case it discovers a conflict in the outputs, timer updates or the on/off status of the successor state. The procedure then terminates, returning a counterexample.

Algorithm 5: Sub-procedure for extending the observation tree with multiple transitions

```
1  Procedure addTransitions^S_M(q ∈ Q^T, σ ∈ I*):
2      while σ ≠ ϵ do
3          i ← head(σ);
4          if δ^T(q, i)↑ then
5              c ← addTransition^S_M(q, i);
6              if c ∈ I* then return c;
7          end
8          q ← δ(q, i);
9          σ ← tail(σ);
10     end
11     return yes;
```

3.3.4 Termination

We can prove that the procedure of Algorithm 1 always terminates within a finite number of rule applications:

Lemma 3.3.1. The procedure of Algorithm 1 always terminates within a finite number of rule applications.

Proof. The IdentifyBasisStates rule can only be applied as many times are there are elements in state cover C. Therefore, since $\mathcal{S}$ has a finite number of elements, C is finite as well, and the IdentifyBasisStates rule can only be applied a finite number of times. The size of the basis is therefore also finite.
The finite size of the basis also imposes a limit on the size that the 0-frontier can reach through repeated application of the ExtendFrontiers rule. The fact that the maximum size of the 0-frontier is finite in turn implies that the maximum size of the 1-level frontier is finite, and so on. The ExtendFrontiers rule can only be applied a finite number of times, since the maximum sizes of the first $k+1$ frontiers are all finite.
Every application of the IdentifyFrontiers rule makes a state from the $k+1$-level frontier apart from at least one basis state. This rule can only be applied a finite number of times, since the maximum size of any frontier is always finite, which implies that the maximum sizes of the first $k+1$ frontiers are always finite.
Every application of the ExtendCoTransitivity rule makes a state from the first k frontiers apart from a state from the $k+1$-level frontier. This rule can only be applied a finite number of times, since the number of states in any frontier is always finite.
We may thus conclude that all four of Algorithm 1's rules can only be applied a finite number of times. The loop-condition of Line 8 will thus always be met after a finite number of loop iterations. The algorithm will therefore always terminate within a finite number of rule applications. □

3.3.5 k-A-Complete Test Suites for MM1Ts

Fault domains are the same for MM1Ts as for Mealy machines, apart from the use of untimed equivalence instead of trace equivalence. We thus get:

Definition 3.3.1 (Fault domains and $\mathcal{U}$-completeness for MM1Ts). Let $\mathcal{S}$ be an MM1T. A **fault domain** is a set $\mathcal{U}$ of MM1Ts. A test suite $TTS^{\mathcal{S}}$ for $\mathcal{S}$ is **$\mathcal{U}$-complete** if, for each $\mathcal{M} \in \mathcal{U}$, $\mathcal{M}$ passes $TTS^{\mathcal{S}}$ implies $\mathcal{M} \approx_{untimed} \mathcal{S}$.

We can define the relevant fault domains:

Definition 3.3.2. Let $k \in \mathbb{N}$, and let $\mathsf{A} \subseteq I^*$. Then $\mathcal{U}_k^{\mathsf{A}}$ is the set of all MM1Ts $\mathcal{M}$ for which, for each state $q \in Q$ there are $\sigma \in \mathsf{A}$ and $\rho \in A^{\leq k}$ such that $\delta^*(q_{\mathcal{I}}, \sigma \cdot \rho) = q$.

Definition 3.3.3. Let $\mathsf{A} \subseteq I^*$. Then $\mathcal{U}^{\mathsf{A}}$ is the set of all MM1Ts $\mathcal{M}$ for which there are $\sigma, \rho \in \mathsf{A}$ with $\sigma \neq \rho$ and $\delta^*(q_{\mathcal{I}}, \sigma) \approx_{untimed} \delta^*(q_{\mathcal{I}}, \rho)$.

We can now define the relevant notion of k-A-completeness:

Definition 3.3.4 (*k*-A-complete test suites for MM1Ts). Let $\mathcal{S}$ be an MM1T with a set of inputs I, let $k \in \mathbb{N}$, and let $\mathsf{A} \subseteq I^*$. Then test suite $TS^{\mathcal{S}}$ is ***k*-A-complete** for $\mathcal{S}$ if, for any SUT MM1T $\mathcal{M} \in \mathcal{U}_k^{\mathsf{A}} \cup \mathcal{U}^{\mathsf{A}}$:

$$\mathcal{M} \text{ passes } TS^{\mathcal{S}} \iff \mathcal{M} \approx_{untimed} \mathcal{S}.$$

3.3.6 *k*-A-Completeness of the Procedure

Our approach to proving the k-A-completeness of our procedure is inspired by Vaandrager et al. [2024]'s sufficient condition for the k-A-completeness of test suites for Mealy machines. We will use the following theorem to prove that for any natural number k and any minimal and prefix-closed state cover C of the specification, Algorithm 1 is a valid and k-C-complete conformance testing procedure for MM1Ts:

Theorem 3.3.1. Let $k \in \mathbb{N}^{>0}$. Let $\mathcal{S}$ be a minimal complete MM1T, and let $C \subseteq I^*$ be a minimal and prefix-closed state cover for $\mathcal{S}$. Let $\mathcal{M}$ be a complete MM1T from $\mathcal{U}_k^C \cup \mathcal{U}^C$ that has the same set of inputs I as $\mathcal{S}$. Let $\mathcal{T}$ be an observation tree for both $\mathcal{M}$ and $\mathcal{S}$, and let $B, F^0, F^1, \ldots$ be the stratification of $Q^{\mathcal{T}}$ induced by C. Suppose that B and $F^{<k}$ are all complete, all states in B and F^k are identified, and the following condition holds:

$$\forall t' \in F^k, t'' \in F^{<k}\colon \qquad \mathcal{C}(t') = \mathcal{C}(t'') \quad \vee \quad t' \# t''. \tag{3.1}$$

Then $\mathcal{M} \approx_{untimed} \mathcal{S}$.

The proof of Theorem 3.3.1 can be found in Appendix B.1.1.

Algorithm 1 does not directly guarantee that the condition of Equation (3.1) will hold. Its ExtendCoTransitivity rule instead guarantees that once the procedure is done, the following condition will hold:

$$\forall r \in B, t' \in F^k, t'' \in F^{<k}\colon \qquad r \# t' \implies r \# t'' \vee t' \# t''. \tag{3.2}$$

We use the following property to prove that Equation (3.2) implies Equation (3.1):

Lemma 3.3.2. Let $\mathcal{S}$ be an MM1T, and let $\mathcal{T}$ be an observation tree for $\mathcal{S}$. Let B be the basis of a stratification of $Q^{\mathcal{T}}$. Suppose that $q, q' \in Q^{\mathcal{T}}$ and q is identified. Then:

$$\mathcal{C}(q) = \mathcal{C}(q') \vee q \# q' \iff (\forall r \in B\colon r \# q \implies r \# q' \vee q \# q').$$

The proof of Lemma 3.3.2 can be found in Appendix B.1.2.

Let k be a natural number, and let A be a prefix-closed state cover of the specification. The validity of Algorithm 1 as a k-A-complete testing procedure for MM1Ts is a corollary of Theorem 3.3.1:

Corollary 3.3.1. Let $\mathcal{S}$ be a complete, minimal MM1T. Let C be a minimal and prefix-closed state cover for $\mathcal{S}$. Let k be a natural number, and let $\mathcal{M}$ be an MM1T from $\mathcal{U}_k^C \cup \mathcal{U}^C$. The procedure of Algorithm 1 returns `yes` iff $\mathcal{M} \approx_{untimed} \mathcal{S}$, and it returns a counterexample in the form of an input sequence iff $\mathcal{M} \not\approx_{untimed} \mathcal{S}$.

Proof. We know from Lemma 3.3.1 that Algorithm 1 always terminates within a finite number of rule applications. We see on Line 24 that the algorithm returns `yes` once none of the rules can be applied anymore. When that happens, we know from the IdentifyBasisStates rule that all basis states are identified, from the ExtendFrontiers rule that B and $F^{<k}$ are all complete, from the IdentifyFrontiers rule that all states in F^k are identified, and from the ExtendCoTransitivity rule that Equation (3.2) holds. Therefore, by Lemma 3.3.2:

$$\forall t' \in F^k, t'' \in F^{<k}\colon \qquad \mathcal{C}(t') = \mathcal{C}(t'') \quad \vee \quad t' \# t''.$$

Theorem 3.3.1 thus tells us that if Algorithm 1 terminates because none of its rules can be applied anymore, then $\mathcal{M} \approx_{untimed} \mathcal{S}$.
The only circumstance under which Algorithm 1 terminates before all four rules are exhausted is if it finds a conflict between the outputs, timer updates or the on/off status of the timer in the successor state between corresponding transitions of $\mathcal{M}$ and $\mathcal{S}$, in wich case the algorithm returns a counterexample input sequence. The presence of such a conflict would then indeed imply that $\mathcal{M} \not\approx_{untimed} \mathcal{S}$. □

We can now prove that Algorithm 1 is k-C-complete, where k is an arbitrary natural number and C is a minimal and prefix-closed state cover for the specification:

Corollary 3.3.2. Let $\mathcal{S}$ be a complete, minimal MM1T, and let C be a minimal and prefix-closed state cover for $\mathcal{S}$. Let k be a natural number. Then Algorithm 1 is k-C-complete.

Proof. Let $\mathcal{M}$ be an MM1T in $\mathcal{U}_k^C \cup \mathcal{U}^C$. Then Corollary 3.3.1 tells us that Algorithm 1 returns `yes` iff $\mathcal{M} \approx_{untimed} \mathcal{S}$, as required. □

3.3.7 Comparison With the H-Method

The rules used by the testing procedure of Algorithm 1 closely resemble the four steps of the H-method [Dorofeeva et al., 2005] for computing k-complete test suites for Mealy machines. Both methods use a prefix-closed state cover C for the minimal specification model to cover all specification states in what we call the basis. They also both explore the first $k+1$ frontiers from all basis states, and they both ensure that certain states can be told apart based on their behavior. The H-method ensures that certain states are trace inequivalent, while Algorithm 1 ensures that states are apart according to the notion of apartness for observation tree MM1Ts. The correspondence between the H-method and Algorithm 1 is as follows:

1. Step 1 of the H-method is to create an initial test suite $TS_k^{\mathcal{S}} = C \cdot I^{\leq k+1}$. Algorithm 1 similarly starts by adding every sequence of C to observation tree $\mathcal{T}$. Repeatedly applying the ExtendFrontiers rule until it can no longer be used would then complete the first k frontiers. Afterwards, the set of the access sequences for all states of $\mathcal{T}$ is equal to $C \cdot I^{\leq k+1}$.

2. Step 2 of the H-method ensures that all distinct basis states r, r' are distinguishable by ensuring that $TS_k^{\mathcal{M}}$ contains input sequences $\mathsf{access}(r) \cdot w$ and $\mathsf{access}(r') \cdot w$, where $\lambda^{\mathcal{S}^*}(r, w) \neq \lambda^{\mathcal{S}^*}(r', w)$. The IdentifyBasisStates rule achieves the same purpose for all MM1T basis states by doing essentially the same thing: finding the specification states s and s' that correspond to r and r', finding an input sequence σ for which s and s' behave differently, and then testing the input sequences $\mathsf{access}(r \cdot \sigma)$ and $\mathsf{access}(r' \cdot \sigma)$. Step 2 is thus akin to repeatedly applying the IdentifyBasisStates rule until it can no longer be used.

3. Step 3 of the H-method makes all basis states r distinguishable from all frontier states t that do not correspond to the same state of the Mealy machine as r. This is akin to making all frontier states identified, which is what the IdentifyFrontiers rule does in Algorithm 1 for all states from the $k+1$-level frontier. Step 3 thus resembles the act of repeatedly applying the IdentifyFrontiers rule until it can no longer be used.
   The two procedures do differ in their approach, since step 3 directly compares r and t, while the IdentifyFrontiers rule takes two distinct basis states r and r', and then uses weak co-transitivity to find an input sequence σ with which it can make at least one of the two apart from t.
   This alternative to the IdentifyFrontiers rule more closely resembles step 3 of the H-method than the version that we use in Algorithm 1:

Algorithm 6: Alternative to Rule (IdentifyFrontiers)

```
[] ¬(t # r) ∧ r ≠ r_t, for some t ∈ F^k, r, r_t ∈ B, for which r_t = basisStateFor^S_(T,C)(t) →   ▷ Rule (IdentifyFrontiers)
    c ← makeObsTreeStatesApart^S(t, r);
    if c ∈ I* then return c;
```

This version relies on a the sub-procedure $basisStateFor^{\mathcal{S}}_{(\mathcal{T},C)}$ that obtains for a given observation tree state the basis state that corresponds to the same specification state:

We don't use this version of the IdentifyFrontiers rule by default, since it is more complicated than the version from Algorithm 1.

Algorithm 7: Sub-procedure for finding the basis state that corresponds to a given observation tree state

Procedure $basisStateFor^{\mathcal{S}}_{(\mathcal{T},C)}(q \in Q^{\mathcal{T}})$**:**
  $s \leftarrow \delta^{\mathcal{S}^*}(s^{\mathcal{S}}_{\mathcal{I}}, \mathsf{access}(q))$;
  $\sigma \leftarrow \rho \in C\colon \delta^{\mathcal{S}^*}(s^{\mathcal{S}}_{\mathcal{I}}, \rho) = s$;
  $r \leftarrow \delta^{\mathcal{T}^*}(q^{\mathcal{T}}_{\mathcal{I}}, \sigma)$;
  return r;

4. Step 4 of the H-method ensures that all distinct states t, t' from the first $k+1$ frontiers are distinguishable, granted that there is an input sequence σ such that $t' = \delta^*(t, \sigma)$. Algorithm 1's final rule ExtendCoTransitivity doesn't resemble this procedure too closely, as it instead pursues the co-transitivity property of Equation (3.2). Still, the two procedures do resemble each other in that they make certain frontier states distinguishable from one another when this is required by their respective testing methods. Step 4 is thus somewhat analogous to repeatedly applying the ExtendCoTransitivity rule until it can no longer be used.

Another similarity between the H-method and Algorithm 1 is that while both methods make use of distinguishing sequences between states, neither method specifies how exactly these sequences are to be found. Algorithm 1 makes heavy use of input sequences that witness the apartness between two states, but the precise approach for finding these witnesses is left as an implementation detail. The H-method similarly leaves the approach for finding the distinguishing sequences between distinct specification states as an implementation detail.

Notice that in this comparison, we never once mentioned the fact that while Algorithm 1 performs its rules, it always checks for conflicts between the outputs, timer updates, and the on/off status of the timer in the successor state for all transitions of every sequence that it processes. Whereas the H-method only computes a test suite that then still needs to be evaluated on the specification and the SUT, Algorithm 1 effectively evaluates every test as soon as it generates it. This means unlike the H-method, our method will terminate as soon as it generates a test that can show a conflict between the behavior of the specification and the SUT.

3.3.8 The Order in Which the Rules are Applied

Algorithm 1 applies its four rules non-deterministically. There are thus various possible orders in which the rules can be applied. This non-determinism makes for an additional challenge when it comes to implementing the procedure. We therefore identify fixed orders in which the rules can be applied.

There are rule application ordenings in which each rule can be applied repeatedly until it can never be used again. Such ordenings are ideal, since they allow for the rules to be applied as four sequential steps. These ordenings thus resemble the stepwise approach taken by the H-method. Any such ordening has to account for the following restrictions:

- The IdentifyBasisStates and ExtendFrontiers rules have no restrictions on when they can be applied.
- The IdentifyFrontiers rule relies on basis states being identified (IdentifyBasisStates). It also relies on the existence of states from the first $k+1$ frontiers (ExtendFrontiers).
- The ExtendCoTransitivity rule relies on the existence of states from the first $k+1$ frontiers (ExtendFrontiers). It also relies on these states being identified (IdentifyFrontiers).

Any ordening in which the rules are exhausted one-by-one may thus start with either the IdentifyBasisStates rule or the ExtendFrontiers rule, but has to end with the IdentifyFrontiers rule, followed by the ExtendCoTransitivity rule.

Note that the ordening in which the IdentifyBasisStates rule is exhausted after the ExtendFrontiers rule, but before the IdentifyFrontiers rule closely resembles the H-method. We refer to Section 3.3.7 for more details.

Chapter 4

MMT Testing Preliminaries

In this chapter, we introduce the reader to the concept of the Mealy Machines with Multiple Timers (MMTs), which were first introduced in Bruyère et al. [2024]. Almost everything in this chapter is taken directly from Bruyère et al. [2024]. We include this material in this report for the reader's convenience.

4.1 Mealy Machines With Multiple Timers

MMTs function as a generalization of MM1Ts. Instead of one timer, an MMT can have any finite number of timers for which timeouts can occur. The set of timers associated with an MMT $\mathcal{M}$ is captured in the set X. The timeout inputs that we use for MM1Ts are replaced with inputs $\mathsf{to}[x]$, where $x \in X$. We collectively refer to inputs and timeouts as **actions**. We sometimes call inputs **input actions**, and timeouts **timeout actions**. The set of all actions of $\mathcal{M}$ is given by $A = I \cup TO(X)$, with $TO(X) \coloneqq \{\mathsf{to}[x] \mid x \in X\}$. The MM1T's reset function is replaced with an **update function** that assigns an update from $U \coloneqq (X \times \mathbb{N}^{>0}) \cup \{\bot\}$ to each transition. We define MMTs as follows:

Definition 4.1.1 (Mealy machine with timers). A **Mealy machine with timers (MMT)** is a tuple $\mathcal{M} = (Q, q_{\mathcal{I}}, X, I, O, \mathcal{X}, \delta, \lambda, \tau)$, where:

- Q is a finite set of **states**,
- $q_{\mathcal{I}} \in Q$ is the **initial state**,
- X is a finite set of **timers**,
- I is a finite set of **inputs**,
- O is a set of **outputs**,
- $\mathcal{X} \colon Q \to \mathcal{P}(X)$ is a function that assigns a finite set of **active timers** to each state,
- $\delta \colon Q \times A \rightharpoonup Q$ is a **transition function**,
- $\lambda \colon Q \times A \rightharpoonup O$ is an **output function**, and
- $\tau \colon Q \times A \to U$ is an **update function**.

Let $\delta(q, i) = q'$ and $\lambda(q, i) = o$. We write $q \xrightarrow[\bot]{i/o} q'$ if $\tau(q, i) = \bot$, and $q \xrightarrow[(x,c)]{i/o} q'$ when $\tau(q, i) = (x, c)$. An MMT is **valid** if and only if its active timer, transition, output and update functions satisfy the following

rules, for all $q, q' \in Q$, $i \in A$, $o \in O$, $x, y \in X$, and $c \in \mathbb{N}^{>0}$:

$$\mathcal{X}(q_{\mathcal{I}}) = \emptyset \tag{4.1}$$
$$\lambda(q,i)\downarrow \iff \delta(q,i)\downarrow \tag{4.2}$$
$$q \xrightarrow[\bot]{i/o} q' \implies \mathcal{X}(q') \subseteq \mathcal{X}(q) \tag{4.3}$$
$$q \xrightarrow[(x,c)]{i/o} q' \implies x \in \mathcal{X}(q') \quad \wedge \quad \mathcal{X}(q') \setminus \{x\} \subseteq \mathcal{X}(q) \tag{4.4}$$
$$q \xrightarrow[\bot]{\mathsf{to}[x]/o} q' \implies x \in \mathcal{X}(q) \quad \wedge \quad x \notin \mathcal{X}(q') \tag{4.5}$$
$$q \xrightarrow[(y,c)]{\mathsf{to}[x]/o} q' \implies x \in \mathcal{X}(q) \quad \wedge \quad x = y. \tag{4.6}$$

In this report, we always assume MMTs to be valid, unless we specify otherwise.

Missing symbols in $q \xrightarrow[u]{i/o} q'$ are quantified existentially.

Example 4.1.1. $q \xrightarrow[u]{i/o}$ means that there exists a state q' such that $q \xrightarrow[u]{i/o} q'$.

Example 4.1.2. $q \xrightarrow{i}$ indicates that there exist an output o and an update u such that $q \xrightarrow[u]{i/o}$.

Bruyère et al. [2024] also introduced *generalized* MMTs (gMMTs), which can **rename** timers along their state transitions. This is the only difference between the two model types:

Definition 4.1.2 (Generalized Mealy machines with timers). A **generalized Mealy machine with timers (gMMT)** is a tuple $\mathcal{M} = (Q, q_{\mathcal{I}}, X, I, O, \mathcal{X}, \delta, \lambda, \tau)$, where $Q, q_{\mathcal{I}}, X, I, O, \mathcal{X}, \delta$ and λ are the same as for MMTs. The only difference is in the update function, which now has the signature: $\tau \colon Q \times A \to (X \to (X \cup \mathbb{N}^{>0}))$ that allows for individual transitions to assign values to multiple timers. Let $\delta(q,i) = q'$, $\lambda(q,i) = o$ and $\mathfrak{r} = \tau(q,i)$. We write $q \xrightarrow[\mathfrak{r}]{i/o} q'$. A gMMT is **valid** if and only if its set of timers, active timer function, transition function and update function satisfy the following rules, for all $q, q' \in Q$, $i \in A$, $x \in X$, and $\mathfrak{r} \colon X \to (X \cup \mathbb{N}^{>0})$:

$$X \cap \mathbb{N}^{>0} = \emptyset \tag{4.7}$$
$$\mathcal{X}(q_{\mathcal{I}}) = \emptyset \tag{4.8}$$
$$\lambda(q,i)\downarrow \iff \delta(q,i)\downarrow \tag{4.9}$$
$$q \underset{\mathfrak{r}}{\to} q' \implies \mathfrak{r} \text{ is injective} \quad \wedge \quad \mathsf{dom}(\mathfrak{r}) = \mathcal{X}(q') \quad \wedge \quad \mathsf{ran}(\mathfrak{r}) \subset (\mathcal{X}(q) \cup \mathbb{N}^{>0}) \quad \wedge \quad \text{there is at most one } x \in \mathsf{dom}(\mathfrak{r}) \text{ with } \mathfrak{r}(x) \in \mathbb{N}^{>0} \tag{4.10}$$
$$q \xrightarrow[\mathfrak{r}]{\mathsf{to}[x]} q' \implies x \in \mathcal{X}(q) \quad \wedge \quad x \notin \mathsf{ran}(\mathfrak{r}). \tag{4.11}$$

As for MMTs, we always assume gMMTs to be valid, unless we specify otherwise. For MMTs, we say that a transition $q \underset{u}{\to} q'$ **starts** (resp. **restarts**) timer x if $u = (x, c)$ and x is inactive (resp. active) in q. We similarly say for gMMTs that a transition $q \underset{\mathfrak{r}}{\to} q'$ **starts** (resp. **restarts**) timer x if $\mathfrak{r}(x) \in \mathbb{N}^{>0}$ and x is inactive (resp. active) in q. We say that a transition $q \xrightarrow{i} q'$ with $i \neq \mathsf{to}[x]$ **stops** timer x if x is inactive in q'.

MMTs and gMMTs have in common that they can (re)start at most one timer in any given state transition. The *difference* between the two is that while MMTs can only (re)start timers in their transitions, gMMTs can also swap the values of active timers in their state transitions. This ability, which is called timer **renaming**, enables gMMTs to express some models more succinctly than is possible with MMTs.

Let $\mathcal{M}$ be an arbitrary (g)MMT with a set of states Q. We generalize the transition function to sequences of actions, i.e. to elements of A^*. We get, for all $q \in Q$, all $i \in A$ and all $\sigma \in A^*$:

$$\delta^*(q, \epsilon) = q$$
$$\delta^*(q, i\ \sigma) = \begin{cases} \delta^*(\delta(q,i), \sigma) & \text{if } \delta(q,i)\downarrow \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We similarly generalize the output function to sequences of actions. We get, for all $q \in Q$, all $i \in A$ and all $\sigma \in A^*$:

$$\lambda^*(q, \epsilon) = \epsilon$$
$$\lambda^*(q, i\ \sigma) = \begin{cases} \lambda(q,i)\ \lambda^*(\delta(q,i), \sigma) & \text{if } \delta(q,i)\downarrow \wedge \lambda(q,i)\downarrow \\ \text{undefined} & \text{otherwise.} \end{cases}$$

4.2 Untimed Semantics

Definition 4.2.1 (Run). A **run** of MMT $\mathcal{M}$ consists of either a single state q, or of a nonempty sequence of transitions:

$$q_0 \xrightarrow[u_1]{i_1/o_1} q_1 \xrightarrow[u_2]{i_2/o_2} \dots \xrightarrow[u_n]{i_n/o_n} q_n.$$

We use $runs(\mathcal{M})$ to denote the set of all runs of $\mathcal{M}$. Note that any run π is uniquely determined by its first state q_0 and input sequence, as (g)MMTs are deterministic.

Definition 4.2.2 (Spanning run for MMTs). A run $q_0 \xrightarrow[u_1]{i_1} \dots \xrightarrow[u_n]{i_n} q_n$ is said to be ***x*-spanning** (with $x \in X$) if it begins with a transition (re)starting x, ends with a $\mathsf{to}[x]$ transition, and no intermediate transition restarts or stops x. That is, $u_1 = (x, c)$, $i_n = \mathsf{to}[x]$, $u_j \neq (x, d)$ for all $j \in \{2, \dots, n-1\}$ and $d \in \mathbb{N}^{>}0$, and $x \in \mathcal{X}(q_j)$ for all $j \in \{2, \dots, n-1\}$.

Definition 4.2.3 (Spanning run for gMMTs). A run:

$$q_0 \xrightarrow[\mathfrak{r}_1]{i_1} q_0 \xrightarrow[\mathfrak{r}_2]{i_2} \dots \xrightarrow{i_n} q_n$$

and there exist timers $x_1, \dots, x_n$ 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 $j \in \{2, \dots, n-1\}$ (this implies that $x_j \in \mathcal{X}(q_j)$), and
- $i_n = \mathsf{to}[x_{n-1}]$.

4.3 Timed Semantics

Let $\mathcal{M}$ be an (g)MMT. A **valuation** is a partial function $\kappa \colon X \rightharpoonup \mathbb{R}^{>0}$ that assigns nonnegative real numbers to timers. For a set of timers $Y \subseteq X$, $\mathsf{Val}(Y)$ denotes the set of all valuations κ with $\mathsf{dom}(\kappa) = Y$. A **configuration** of $\mathcal{M}$ is a pair $(q, \kappa) \in Q \times \mathsf{Val}(Y)$. The **initial configuration** of $\mathcal{M}$ is the pair $(q_\mathcal{I}, \kappa)$, where $\kappa = \emptyset$ since $\mathcal{X}(q_\mathcal{I}) = \emptyset$.
If $\kappa \in \mathsf{Val}(Y)$ is a valuation in which $\forall x \in Y \colon \kappa(x) \geq d \in \mathbb{R}^{>0}$, then d units of timer may **elapse**. We write $\kappa - d \in \mathsf{Val}(Y)$ for the resulting valuation, which satisfies $\forall x \in Y \colon (\kappa - d)(x) = \kappa(x) - d$. If there exists a timer x such that $\kappa(x) = 0$, then x may **time out**. The transitions between configurations $(q, \kappa), (q', \kappa')$ are defined as follows:

- $(q, \kappa) \xrightarrow{d} (q, \kappa - d)$, with $\kappa(x) \geq d$ for every $x \in \mathcal{X}(q)$ is a **delay transition**, and

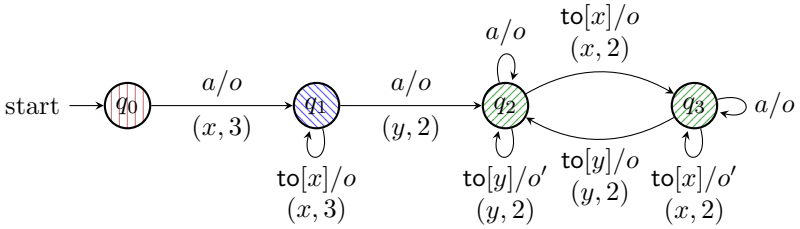


Figure 4.1: An MMT, with $\mathcal{X}(q_0) = \mathcal{X}_0(q_0) = \emptyset$, $\mathcal{X}(q_1) = \mathcal{X}_0(q_1) = \{x\}$, and $\mathcal{X}(q_2) = \mathcal{X}_0(q_2) = \mathcal{X}(q_3) = \mathcal{X}_0(q_3) = \{x, y\}$

- $(q, \kappa) \xrightarrow[u]{i/o} (q', \kappa')$, with $q \xrightarrow[u]{i/o} q' \in runs(\mathcal{M})$, $u = (x, c) \Rightarrow \kappa'(x) = c$, and:

  $$\forall y \in \mathcal{X}(q'): \qquad u \neq (y, d) \quad \Longrightarrow \quad \kappa'(y) = \kappa(y),$$

  is a **discrete transition**. Moreover, if $i = \mathsf{to}[x]$, then $\kappa(x) = 0$ and the transition is called a **timeout transition**. Otherwise, it is an **input transition**.

Missing symbols in $(q, \kappa) \xrightarrow{d} (q, \kappa - d)$ and $(q, \kappa) \xrightarrow[u]{i/o} (q', \kappa')$ are quantified existentially.

Definition 4.3.1 (Timed run). A **timed run** is a sequence of configuration transitions, beginning and ending with a delay transition.

Definition 4.3.2 (Untimed projection of a run). The **untimed projection** of a timed run ρ, denoted $untimed(\rho)$, is the run obtained by removing ρ's valuations and delay transitions.

Definition 4.3.3 (Feasible runs). A run π is said to be **feasible** if there exists a timed run ρ such that $untimed(\rho) = \pi$.

Definition 4.3.4 (Enabled timers). We define the set $\mathcal{X}_0(q)$ of **enabled timers** of (g)MMT state q as:

$$\mathcal{X}_0(q) = \{x \in \mathcal{X}(q) \mid \exists (q_\mathcal{I}, \emptyset) \xrightarrow{w} (q, \kappa) \colon \kappa(x) = 0\}.$$

Definition 4.3.5 (Complete (g)MMT). We say that an (g)MMT $\mathcal{M}$ is **complete** if each state $q \in Q$ has an outgoing transition for each of the actions that can be taken from that state. Formally:

$$\forall q \in Q, i \in A \colon q \xrightarrow{i} \in runs(\mathcal{M}) \qquad \Longleftrightarrow \qquad i \in I \cup TO(\mathcal{X}_0(q)).$$

Definition 4.3.6 (Connected (g)MMT). An (g)MMT $\mathcal{M}$ is **connected** iff, for each state $q \in Q$ there exists an action sequence $\sigma \in A^*$ such that $q_\mathcal{I} \xrightarrow{\sigma} q \in runs(\mathcal{M})$.

Definition 4.3.7 (Partial MMT). A **partial (g)MMT** is an (g)MMT that may or may not be complete.

Definition 4.3.8 (*s*-learnable (g)MMT). An (g)MMT $\mathcal{M}$ is ***s*-learnable** if it is complete, and every run of $\mathcal{M}$ is feasible.

Unlike Bruyère et al. [2024], we always require that (g)MMTs are valid. We therefore don't need to explicitly require that s-learnable models are valid (sound, in their terms).

Example 4.3.1. Figure 4.1 shows an example of a complete MMT with timers $X = \{x, y\}$. Timer x is active in q_1, q_2 and q_3, while timer y is only active in states q_2 and q_3. A timer can only be active in a state if it was first started in a preceding transition. The transition for input a from state q_0 to state q_1 starts timer x with value 3. The MMT is not s-learnable, because its run $a\ a\ \mathsf{to}[x]\ \mathsf{to}[x]$ is not feasible.

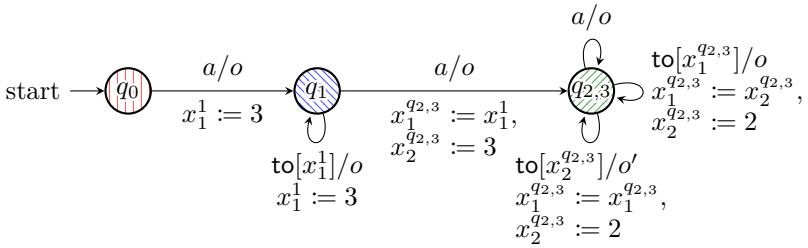


Figure 4.2: A gMMT that is symbolically equivalent to the MMT of Figure 4.1

4.4 Symbolic Words and Symbolic Equivalence

Definition 4.4.1 (Symbolic words). Let $\mathcal{M}$ be an MMT, and let $w = i_1 \dots i_n$ be a word over A that is the label of a run $\pi = q_0 \xrightarrow[u_1]{i_1} q_1 \xrightarrow[u_2]{i_2} \dots \xrightarrow[u_n]{i_n} q_n \in runs(\mathcal{M})$. The **symbolic word (sw)** of w is the word $\overline{w} = \mathtt{i}_1 \dots \mathtt{i}_n$ over $\mathtt{A}$ such that, for every $k \in \{1, \dots, n\}$:

- $\mathtt{i}_k = i_k$ if $i_k \in I$, and
- $\mathtt{i}_k = \mathsf{to}[j]$, where $j < k$ is the index of the last transition that (re)starts timer x if $i_k = \mathsf{to}[x]$.

A given symbolic word $\mathtt{w} = \mathtt{i}_1 \dots \mathtt{i}_n$ over $\mathtt{A}$ that can be converted into a word w over A if there is a run $q_\mathcal{I} \xrightarrow{w} \in runs(\mathcal{M})$. Appendix B of Bruyère et al. [2024] explains how such a run can be used to convert the symbolic word into non-symbolic word w.

Let $\mathsf{A} \subseteq A^*$ be a set of words over A. We define $\overline{\mathsf{A}} = \{\overline{w} \mid w \in \mathsf{A}\}$.

We generalize the MMT and gMMT transition sequence functions to also work for symbolic words that are run from the (g)MMT's initial state. Since these symbolic words $\mathtt{w}$ are run from the initial state, we know that if $q_\mathcal{I} \xrightarrow{\mathtt{w}}$ is feasible in (g)MMT $\mathcal{M}$, then for the non-symbolic word w such that $\overline{w} = \mathtt{w}$, $q_\mathcal{I} \xrightarrow{w} \in runs(\mathcal{M})$. We define, for all (g)MMTs $\mathcal{M}$ and all symbolic words $\mathtt{w} \in (\mathtt{A})^*$:

$$\delta^*(\mathtt{w}) := \begin{cases} \delta^*(q_\mathcal{I}, w) & \text{if } \overline{w} = \mathtt{w} \wedge q_\mathcal{I} \xrightarrow{\mathtt{w}} \text{ is feasible in } \mathcal{M} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Bruyère et al. [2024] defined notions of **symbolic equivalence** between two MMTs, and between a gMMT and an MMT. We write $\mathcal{M} \approx_{sym} \mathcal{N}$ to denote that $\mathcal{M}$ and $\mathcal{N}$ are symbolically equivalent.

Example 4.4.1. Figure 4.2 shows an example of a gMMT that is symbolically equivalent to the MMT of Figure 4.1.

4.5 Race Conditions and Race Avoidance

The MMT learning method accounts for the potential occurrence of **race conditions** when it runs timed input words on the SUL. For MMTs, a race condition occurs in a timed run when either:

- two timers expire simultaneously (i.e. the sum of the delays between them is zero), or
- a timer x expires while another transition simultaneously (re)starts or stops x (i.e. timer x reaches zero while the next transition (re)starts or stops x).

In the first case, it cannot be observed which timer expired. In the second case, it is undefined whether a timeout would occur for a timer x before it is either (re)started or stopped and the x-timeout is prevented.

Definition 4.5.1 (Race avoiding MMT). Let $\mathcal{M}$ be a sound MMT. We say that $\mathcal{M}$ is **race avoiding** if any feasible run $\pi = q_\mathcal{I} \xrightarrow{i_1} q_1 \xrightarrow{i_2} \dots \xrightarrow{i_2} \dots \xrightarrow{i_n} q_n$ is the untimed projection of a timed run $\rho = (q_\mathcal{I}, \emptyset) \xrightarrow{d_1} (q_\mathcal{I}, \emptyset) \xrightarrow{i_1} (q_1, \kappa_1) \xrightarrow{d_2} \dots \xrightarrow{i_n} (q_n, \kappa_n) \xrightarrow{d_{n+1}} (q_n, \kappa_n - d_{n+1})$, such that:

- all delays are non-zero: $d_j > 0$ for any $j \in \{1, \ldots, n+1\}$,
- timers always time out precisely when π wants to process their timeout: for any $(\kappa_j - d_{j+1})$ and $x \in \mathcal{X}(q_j)$ with $j \in \{1, \ldots, n-1\}$, we have $(\kappa_j - d_{j+1})(x) = 0$ iff $i_{j+1} = \mathsf{to}[x]$, and
- no timer times out in $\kappa_n - d_{n+1}$: $(\kappa_n - d_{n+1})(x) \neq 0$ for all $x \in \mathcal{X}(q_n)$.

4.6 Auxiliary Functions That Describe Timer Behavior

We define some auxiliary functions that will help us describe the way that (g)MMT timers behave along untimed runs.

For gMMTs, we start with a function that yields for a given timer the timer to which it renames along a given run:

Definition 4.6.1. Let $\mathcal{M}$ be a gMMT with $\pi = q_0 \xrightarrow[\mathfrak{r}_1]{i_1} q_1 \xrightarrow[\mathfrak{r}_2]{i_2} \ldots \xrightarrow[\mathfrak{r}_n]{i_n} q_n \in runs(\mathcal{M})$ and $x_0 \in \mathcal{X}(q_0)$. Then:

$$renameTo_{q_0 \xrightarrow{i_1 \ldots i_n} q_n}(x_0) = x_n,$$

iff $x_n \in \mathcal{X}(q_n)$ and there exist timers $x_1, \ldots, x_{n-1} \in X$ such that:

$$\forall j \in \{1, \ldots, n\} \colon \mathfrak{r}_j(x_j) = x_{j-1}.$$

Otherwise:

$$renameTo_{q_0 \xrightarrow{i_1 \ldots i_n} q_n}(x_0) = \bot.$$

We also define a function that returns the timer that a given timer is renamed from along a given run:

Definition 4.6.2. Let $\mathcal{M}$ be a gMMT with $\pi = q_0 \xrightarrow[\mathfrak{r}_1]{i_1} q_1 \xrightarrow[\mathfrak{r}_2]{i_2} \ldots \xrightarrow[\mathfrak{r}_n]{i_n} q_n \in runs(\mathcal{M})$ and $x_n \in \mathcal{X}(q_n)$. Then:

$$renamesTo_{q_0 \xrightarrow{i_1 \ldots i_n} q_n}(x_n) = x_0,$$

iff $x_0 \in \mathcal{X}(q_0)$ and there exist timers $x_1, \ldots, x_{n-1} \in X$ such that:

$$\forall j \in \{1, \ldots, n\} \colon \mathfrak{r}_j(x_j) = x_{j-1}.$$

Otherwise:

$$renamesTo_{q_0 \xrightarrow{i_1 \ldots i_n} q_n}(x_n) = \bot.$$

We define a function that yields the index along a given MMT run at which a given timer was last started, granted that this timer remains active in the remainder of the run:

Definition 4.6.3. Let $\mathcal{M}$ be an MMT with $\pi = q_0 \xrightarrow[u_1]{i_1} q_1 \xrightarrow[u_2]{i_2} \ldots \xrightarrow[u_n]{i_n} q_n \in runs(\mathcal{M})$ and $x \in \mathcal{X}(q_n)$. Let $k \in \mathbb{N}$. Then:

$$lastStartedAt_{q_0 \xrightarrow{i_0 \ldots i_n} q_n}(x) = k,$$

iff there exists a $k \in \mathbb{N}$ such that:

- $\tau(q_{k-1}, i_k) = (x, c)$ for some $c \in \mathbb{N}^{>0}$,
- $\forall l \in \{k+1, \ldots, n\} \colon x \in \mathcal{X}(q_l)$, and
- $\forall l \in \{k+1, \ldots, n\} \colon (u_l = \bot) \vee (u_l \in (X \times \mathbb{N}^{>0}) \wedge (\pi_1(u_l) \neq x))$.

Otherwise:

$$\mathit{lastStartedAt}_{q_0 \xrightarrow{i_0 \dots i_n} q_n}(x) = \bot.$$

We also define a gMMT-version of the previous function:

Definition 4.6.4. Let $\mathcal{M}$ be a gMMT with $\pi = q_0 \xrightarrow[\mathfrak{r}_1]{i_1} q_1 \xrightarrow[\mathfrak{r}_2]{i_2} \dots \xrightarrow[\mathfrak{r}_n]{i_n} q_n \in \mathit{runs}(\mathcal{M})$ and $x \in \mathcal{X}(q_n)$. Then:

$$\mathit{lastStartedAt}_{q_0 \xrightarrow{i_0 \dots i_n} q_n}(x) = k,$$

iff there exists a $k \in \mathbb{N}$ such that:

- $\exists y \in X \colon \mathfrak{r}_k(y) \in \mathbb{N}^{>0}$,
- $\mathit{renamesTo}_{q_k \xrightarrow{i_{k+1} \dots i_n} q_n}(y) = x$, and
- $\forall l \in \{k+1, \dots, n\} \colon \mathfrak{r}_l(z) \notin \mathbb{N}^{>0}$, where $z = \mathit{renameTo}_{q_k \xrightarrow{i_{k+1} \dots i_l} q_l}(y)$.

Otherwise:

$$\mathit{lastStartedAt}_{q_0 \xrightarrow{i_0 \dots i_n} q_n}(x) = \bot.$$

We define a function that yields the timer that is started in the final transition along the given MMT run:

Definition 4.6.5. Let $\mathcal{M}$ be an MMT with $\pi = q_0 \xrightarrow{i_1} q_1 \xrightarrow{i_2} \dots \xrightarrow[u]{i_n} q_n \in \mathit{runs}(\mathcal{M})$. Then:

- if $u = (x, c)$ with $x \in \mathcal{X}(q_n)$ and $c \in \mathbb{N}^{>0}$, then:

  $$\mathit{timerStartedAt}(q_0 \xrightarrow{i_1 \dots i_n}) = x,$$

  and
- if $u = \bot$, then:

  $$\mathit{timerStartedAt}(q_0 \xrightarrow{i_1 \dots i_n}) = \bot.$$

Finally, we define a gMMT-version of the previous function:

Definition 4.6.6. Let $\mathcal{M}$ be a gMMT with $\pi = q_0 \xrightarrow{i_1} q_1 \xrightarrow{i_2} \dots \xrightarrow[\mathfrak{r}]{i_n} q_n \in \mathit{runs}(\mathcal{M})$. Then:

- if $\mathfrak{r}(x) \in \mathbb{N}^{>0}$ for some $x \in \mathcal{X}(q_n)$, then:

  $$\mathit{timerStartedAt}(q_0 \xrightarrow{i_1 \dots i_n}) = x,$$

  and
- if $\neg \exists x \in \mathcal{X}(q_n) \colon \mathfrak{r}(x) \in \mathbb{N}^{>0}$, then:

  $$\mathit{timerStartedAt}(q_0 \xrightarrow{i_1 \dots i_n}) = \bot.$$

Chapter 5

k-A-Complete Conformance Testing of MMTs

In this chapter, we introduce our conformance testing method for MMTs. This method follows the same principle as our method for MM1Ts. We start by introducing the notions of timer-observable (t-observable) MMTs and gMMTs, followed by an algorithm that can make s-learnableMMTs t-observable. Next, we define observation trees and functional simulations for (g)MMTs. We then define the notion of explored states, which we subsequently use to define the requisite notions of apartness. Our stratifications for observation tree MMTs are the next topic, followed by the requirements that we impose on the specifications. The final section introduces our conformance testing method for MMTs.

5.1 t-Observable (g)MMTs

We introduce the notion of timer-observable (t-observable) (g)MMTs. The idea is that the model doesn't perform any timer-related actions that cannot be outwardly observed. As such, every timer update performed by a t-observable model starts a spanning run, and timers are only ever active in states that are traversed by at least one spanning for that timer.

Definition 5.1.1 (t-Observable MMT). An MMT $\mathcal{M}$ is t-observable iff:

1. every run of $\mathcal{M}$ is feasible,
2. $\forall q \in Q, x \in X\colon x \in \mathcal{X}(q)$ iff there is an x-spanning run that traverses q, and
3. $\forall q \in Q, x \in X\colon x \in \mathcal{X}_0(q)$ iff $\delta(q, \mathsf{to}[x])\downarrow$.

We define t-observability for gMMTs by adapting the second requirement to account for timer renamings:

Definition 5.1.2 (t-Observable gMMT). A gMMT $\mathcal{M}$ is t-observable iff it meets the first and third requirements for t-observable MMTs, as well as the requirement:
$\forall q_n \in Q, x_n \in X\colon x_n \in \mathcal{X}(q_n)$ iff $\mathcal{M}$ has a spanning run that starts with the sub-run:

$$\pi = q_0 \xrightarrow[\mathfrak{r}_1]{i_1} q_1 \xrightarrow[\mathfrak{r}_2]{i_2} \dots \xrightarrow[\mathfrak{r}_n]{i_n} q_n$$

and there exist timers $x_1, \dots, x_{n-1}$ such that:

1. $\mathfrak{r}_1(x_1) = c$ for some $c \in \mathbb{N}^{>0}$,
2. $\mathfrak{r}_j(x_j) = x_{j-1}$ for all $j \in \{2, \dots, n\}$.

Let $\mathcal{M}$ be a t-observable (g)MMT. We require that every run is feasible, to ensure that the runs that should make $\mathcal{M}$'s timer behavior observable can actually be traversed. The second requirement is there

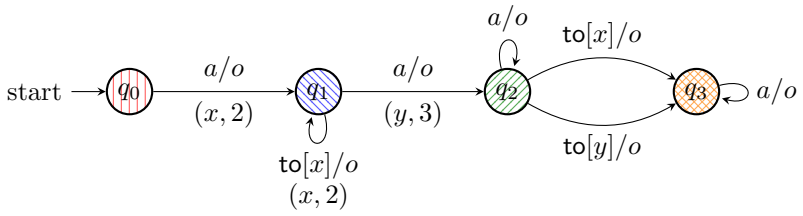


Figure 5.1: An MMT that is neither t-observable nor s-learnable, because $\delta(q_2, \mathsf{to}[y])\downarrow$ while $y \notin \mathcal{X}_0(q_2)$. In this model, $\mathcal{X}(q_0) = \mathcal{X}_0(q_0) = \mathcal{X}(q_3) = \mathcal{X}_0(q_3) = \emptyset$, $\mathcal{X}(q_1) = \mathcal{X}_0(q_1) = \mathcal{X}_0(q_2) = \{x\}$, and $\mathcal{X}(q_2) = \{x, y\}$

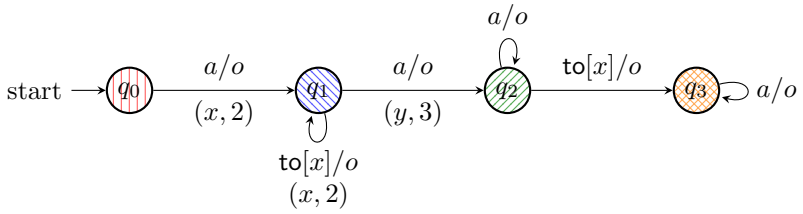


Figure 5.2: An MMT that is s-learnable, but not t-observable because the timer update in the transition from q_1 to q_2 is not observable. In this model, $\mathcal{X}(q_0) = \mathcal{X}_0(q_0) = \mathcal{X}(q_3) = \mathcal{X}_0(q_3) = \emptyset$, $\mathcal{X}(q_1) = \mathcal{X}_0(q_1) = \mathcal{X}_0(q_2) = \{x\}$, and $\mathcal{X}(q_2) = \{x, y\}$

to prevent $\mathcal{M}$ from performing timer behavior that cannot be observed from any runs. This requirement also implies that every timer update that starts a timer x starts an x-spanning run. The third and final requirement ensures that if that if there is a timed run that terminates in a state q for which $x \in \mathcal{X}(q)$ and x has the value 0, then this is made observable by the fact that $\delta(q, \mathsf{to}[x])\downarrow$. The third requirement also ensures that timeouts for timers x can only be performed in states q in which their value can be 0.
The first and third t-observability requirements always hold for s-learnable (g)MMTs: the first is a direct requirement for s-learnability, and the third is implied by the completeness criterium imposed by s-learnability. Symbolically-learnable (g)MMTs aren't guaranteed to satisfy t-observability's second requirement. In Section 5.2, we introduce an algorithm that produces for any s-learnable MMT a symbolically equivalent MMT that is also t-observable. It works by only adding the timer updates and active timers that can be observed, ensuring satisfaction of the second requirement.

Example 5.1.1. Figure 5.1 shows an example of an MMT that is neither t-observable, nor s-learnable. This model is not s-learnable, because $\delta(q_2, \mathsf{to}[y])\downarrow$, while $y \notin \mathcal{X}_0(q_2)$. This violates the completeness requirement. It also violates the third requirement of t-observability. Figure 5.2 shows a copy of the MMT of Figure 5.1 without this $\mathsf{to}[y]$-transition. Indeed, the updated model is s-learnable. It is still not t-observable however, since timer y is started in the a-transition from q_1 to q_2, and since $y \in \mathcal{X}(q_2)$. This violates the second requirement for t-observability, because none of this can be made observable by interacting with the MMT. We explain in Section 5.2 how we would convert this MMT into a symbolically equivalent, t-observable and s-learnable MMT.

5.2 Making s-Learnable MMTs t-Observable

Algorithm 8 describes our procedure for making s-learnable MMTs t-observable. Let $\mathcal{M}$ be an arbitrary s-learnable MMT. The procedure generates an s-learnable, t-observable MMT $\mathcal{N}$ that is symbolically equivalent to $\mathcal{M}$.

Algorithm 8: Procedure for making an MMT t-observable

Procedure *makeTObservable*$(\mathcal{M})$**:**
- $p_{\mathcal{I}}^{\mathcal{N}} \leftarrow$ a fresh MMT state;
- $P^{\mathcal{N}} \leftarrow \{p_{\mathcal{I}}^{\mathcal{N}}\}$;
- $X^{\mathcal{N}} \leftarrow \emptyset$;
- $I^{\mathcal{N}} \leftarrow I^{\mathcal{M}}$;
- $O^{\mathcal{N}} \leftarrow O^{\mathcal{M}}$;
- $\mathcal{X}^{\mathcal{N}} \leftarrow \emptyset; \delta^{\mathcal{N}} \leftarrow \emptyset; \lambda^{\mathcal{N}} \leftarrow \emptyset; \tau^{\mathcal{N}} \leftarrow \emptyset$;
- $\mathcal{N} \leftarrow (P^{\mathcal{N}}, p_{\mathcal{I}}^{\mathcal{N}}, X^{\mathcal{N}}, I^{\mathcal{N}}, O^{\mathcal{N}}, \mathcal{X}^{\mathcal{N}}, \delta^{\mathcal{N}}, \lambda^{\mathcal{N}}, \tau^{\mathcal{N}})$;
- `// Copy` $\mathcal{M}$`'s graph structure to` $\mathcal{N}$
- $f \leftarrow \{(p_{\mathcal{I}}^{\mathcal{N}}, q_{\mathcal{I}})\}$;
- **forall** $q \in Q^{\mathcal{M}} \setminus \{q_{\mathcal{I}}\}$ **do**
  - $p \leftarrow$ a fresh MMT state;
  - $P^{\mathcal{N}} \leftarrow P^{\mathcal{N}} \cup \{p\}$;
  - $f(p) \leftarrow q$;
- **end**
- $S \leftarrow \emptyset$;
- **forall** $p \in P^{\mathcal{N}}$ **do**
  - **forall** $i \in I \cup \{\mathsf{to}[x] \in TO(X^{\mathcal{M}}) \mid \delta^{\mathcal{M}}(f(p), \mathsf{to}[x])\downarrow\}$ **do**
    - $\delta^{\mathcal{N}}(p, i) \leftarrow f^{-1}(\delta^{\mathcal{M}}(f(p), i))$;
    - $\lambda^{\mathcal{N}}(p, i) \leftarrow \lambda^{\mathcal{M}}(f(p), i)$;
    - **if** $\exists x\colon i = \mathsf{to}[x]$ **then**
      - $X^{\mathcal{N}} \leftarrow X^{\mathcal{N}} \cup \{x\}$;
      - $\mathcal{X}^{\mathcal{N}}(p) \leftarrow \mathcal{X}^{\mathcal{N}}(p) \cup \{x\}$;
      - $S \leftarrow S \cup \{(p, x)\}$;
    - **end**
  - **end**
- **end**
- `/* Mark all timers` x `as active in all states that are covered by` x`-spannings */`
- `/* And add all the timer updates that start spanning runs */`
- **forall** $(p, x) \in S$ **do**
  - $R \leftarrow$ a fresh first-in-first-out queue;
  - $E \leftarrow \{p\}$;
  - $R.\mathsf{enqueue}(p)$;
  - **while** $\neg R.\mathsf{isEmpty}()$ **do**
    - $p' \leftarrow R.\mathsf{dequeue}()$;
    - **forall** $p'' \in P^{\mathcal{N}}$ **do**
      - $q'' \leftarrow f(p'')$;
      - **forall** $i \in I \cup TO(X^{\mathcal{N}})\colon \delta^{\mathcal{N}}(p'', i) = p'$ **do**
        - **if** $\tau^{\mathcal{M}}(q'', i) = (x, c)$ **then**
          - $\tau^{\mathcal{N}}(p'', i) \leftarrow (x, c)$;
        - **end**
        - **else if** $p'' \notin E \wedge x \in \mathcal{X}(q'')$ **then**
          - **if** $\tau^{\mathcal{M}}(q'', i) = \bot \vee (\tau^{\mathcal{M}}(q'', i) = (y, c) \wedge y \neq x)$ **then**
            - $\mathcal{X}(p'') \leftarrow \mathcal{X}(p'') \cup \{x\}$;
            - $E \leftarrow E \cup \{p''\}$;
            - $R.\mathsf{enqueue}(p'')$;
          - **end**
        - **end**
      - **end**
    - **end**
  - **end**
- **end**
- **return** $\mathcal{N}$;

Algorithm 8 starts by copying $\mathcal{M}$'s underlying graph structure to a new MMT $\mathcal{N}$. It first adds one state p to $\mathcal{N}$ for every state $q \in Q^{\mathcal{M}}$. The correspondence between all states p and q is captured in a map $f\colon P^{\mathcal{N}} \to Q^{\mathcal{M}}$, i.e., $f(p) = q$ for all $q \in Q^{\mathcal{M}}$. The procedure next completes $\mathcal{N}$'s graph structure by adding for every pair of a state $p \in P^{\mathcal{N}}$ and an action $i \in I \cup \{\mathsf{to}[x] \in TO(X^{\mathcal{M}}) \mid \delta^{\mathcal{M}}(f(p), \mathsf{to}[x])\downarrow\}$ that can be taken from $f(p)$:

1. the successor state $f^{-1}(\delta^{\mathcal{M}}(f(p), i))$,
2. the output symbol $\lambda^{\mathcal{M}}(f(p), i)$, and
3. if $\exists x\colon i = \mathsf{to}[x]$, then:
   (a) timer x is added to $X^{\mathcal{N}}$ (if it wasn't in $X^{\mathcal{N}}$ already),
   (b) x is added to $\mathcal{X}^{\mathcal{N}}(p)$, and
   (c) (p, x) is added to a set S, which is used in the final part of the procedure.

Algorithm 8 now uses S in its final step. For every $(p, x) \in S$, it performs a backwards breadth-first search[Zuse, 1972] to:

- mark x as active in all states that are covered by x-spanning runs that terminate with a timeout from p, and to
- add all timer updates that start x-spanning runs that terminate with a timeout from p.

The result is a t-observable MMT $\mathcal{N}$, such that $\mathcal{N} \approx_{sym} \mathcal{M}$.

Example 5.2.1. Figure 5.3 shows how Algorithm 8's state mapping $f\colon P^{\mathcal{N}} \to Q^{\mathcal{M}}$ relates the states of the s-learnable MMT $\mathcal{M}$ on the top with those of the s-learnable and t-observable MMT $\mathcal{N}$ on the bottom. We stated in Example 5.1.1 that Algorithm 8 would return $\mathcal{N}$ (from Figure 5.1) if it were to be given $\mathcal{M}$ (from Figure 5.2) as an input. It would produce $\mathcal{N}$ by first copying $\mathcal{M}$'s states and its state transitions with their inputs and outputs to a fresh MMT $\mathcal{N}$ (lines 2 through 26). While doing so, the algorithm also composes the set $S = \{(p_1, x), (p_2, x)\}$, where $(p_1, x) \in S$ indicates that $p_1 \xrightarrow{\mathsf{to}[x]} \in runs(\mathcal{N})$, and $(p_2, x) \in S$ indicates that $p_2 \xrightarrow{\mathsf{to}[x]} \in runs(\mathcal{N})$. The algorithm finishes by performing its backwards breadth-first-search for both (p_1, x) and (p_2, x). For (p_1, x), this results in the addition of the timer update $(x, 2)$ to the a-transition from p_0 to p_1, as well as in x becoming active in p_1. For (p_2, x), it results in x becoming active in p_2, and in the same additions as for (p_1, x) if the loop of lines 27 through 49 processes (p_2, x) before (p_1, x). Timer updates for $\mathcal{M}$'s timer y are never added, and y isn't made active in any state of $\mathcal{N}$, because $\mathcal{M}$ has no timeouts for y. The addition of any timer behavior for y would therefore violate the second requirement of t-observability.

We can prove that:

Theorem 5.2.1. Algorithm 8 only returns valid MMTs when it is given valid MMTs.

The proof of Theorem 5.2.1 can be found in Appendix C.4.5.

Theorem 5.2.2. Let $\mathcal{M}$ be an s-learnable MMT. Let $\mathcal{N}$ be the MMT that Algorithm 8 returns when it is called on $\mathcal{M}$. Then $\mathcal{N} \approx_{sym} \mathcal{M}$.

Proof. Lemma C.4.4 tells us that, for all action sequences $\sigma \in (A^{\mathcal{N}})^*$, $\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma)\downarrow \iff \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma)\downarrow$. Let $\sigma \in (A^{\mathcal{N}})^*$ such that $\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma)\downarrow$ and $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma)\downarrow$. Let $p_{k-1} = \delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma)$, and $q_{k-1} = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma)\downarrow$. Lemma C.4.4 tells us that $q_{k-1} = f(p_{k-1})$. Lemma C.4.8 now tells us that:

$$p_{k-1} \xrightarrow[(x,c)]{i_k} p_k \xrightarrow{i_{k+1} \ldots i_j} p_j \in runs(\mathcal{N}) \text{ is } x\text{-spanning} \iff$$
$$f(p_{k-1}) \xrightarrow[(x,c)]{i_k} f(p_k) \xrightarrow{i_{k+1} \ldots i_j} f(p_j) \in runs(\mathcal{M}) \text{ is } x\text{-spanning.}$$

This implies that, for all symbolic words $\mathtt{w} = \mathtt{i}_1 \ldots \mathtt{i}_n$ over $\mathtt{A}^{\mathcal{N}}$:

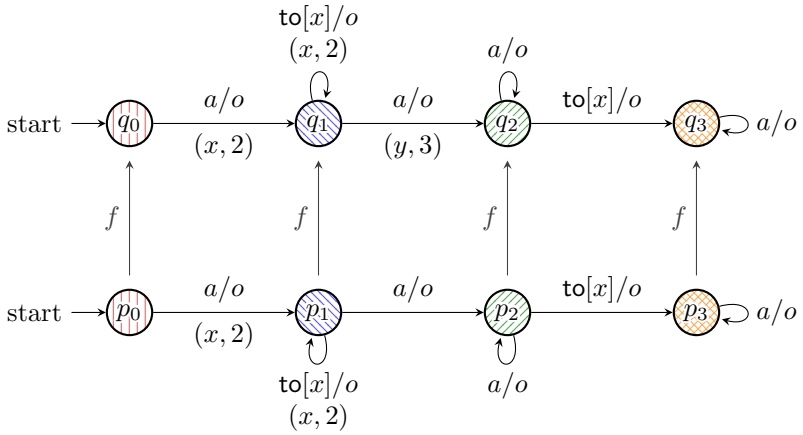


Figure 5.3: Two symbolically equivalent, s-learnable MMTs. The MMT on the bottom is t-observable, the MMT on the top is not. Algorithm 8 would return the MMT on the bottom if it were given the MMT on the top. The gray arrows and the state coloring indicate the correspondence of the states of these models given by Algorithm 8's state mapping f. In these MMTs, $\mathcal{X}(\{q_0, q_3, p_0, p_3\}) = \mathcal{X}_0(\{q_0, q_3, p_0, p_3\}) = \emptyset$, $\mathcal{X}(\{q_1, p_1, p_2\}) = \mathcal{X}_0(\{q_1, p_1, q_2, p_2\}) = \{x\}$, and $\mathcal{X}(q_2) = \{x, y\}$.

- $p_{\mathcal{I}}^{\mathcal{N}} \xrightarrow[u_1]{\mathtt{i}_1/o_1} p_1 \dots \xrightarrow[u_n]{\mathtt{i}_n/o_n} p_n$ is feasible in $\mathcal{N}$ iff $f(p_{\mathcal{I}}^{\mathcal{N}}) \xrightarrow[u'_1]{\mathtt{i}_1/o'_1} f(p_1) \dots \xrightarrow[u'_n]{\mathtt{i}_n/o'_n} f(p_n)$ is feasible in $\mathcal{M}$, and
- if $p_{k-1} \xrightarrow{\mathtt{i}_k \dots \mathtt{i}_j} p_j$ is spanning then $u_k = (x, c) \wedge u'_k = (x', c') \wedge c = c'$.

Moreover, Lemma C.4.4 tells us that $o_j = o'_j$ for all $j \in \{1, \dots, n\}$.

We can thus conclude that $\mathcal{N} \approx_{sym} \mathcal{M}$.

□

Theorem 5.2.3. Let $\mathcal{M}$ be an MMT, and let $\mathcal{N}$ be the MMT that Algorithm 8 returns when it is called on $\mathcal{M}$. If $\mathcal{M}$ is connected, then $\mathcal{N}$ is connected as well.

The proof of Theorem 5.2.3 can be found in Appendix C.4.6. We can now prove that Algorithm 8 only returns t-observable MMTs, granted that the provided MMT is s-learnable:

Theorem 5.2.4. Let $\mathcal{M}$ be an s-learnable MMT, and let $\mathcal{N}$ be the MMT that Algorithm 8 returns when it is called on $\mathcal{M}$. Then $\mathcal{N}$ is t-observable.

The proof of Theorem 5.2.4 can be found in Appendix C.4.7.

Theorem 5.2.4 now enabled us to prove that:

Theorem 5.2.5. Let $\mathcal{M}$ be an s-learnable MMT, and let $\mathcal{N}$ be the MMT that Algorithm 8 returns when it is called on $\mathcal{M}$. Then $\mathcal{N}$ is complete.

The proof of Theorem 5.2.5 can be found in Appendix C.4.8.

Finally, these properties let us conclude that Algorithm 8 preserves s-learnability.

Corollary 5.2.1. Let $\mathcal{M}$ be an s-learnable MMT, and let $\mathcal{N}$ be the MMT that Algorithm 8 returns when it is called on $\mathcal{M}$. Then $\mathcal{N}$ is s-learnable.

Proof. Since $\mathcal{M}$ is s-learnable, it is complete. Therefore, by Theorem 5.2.5, $\mathcal{N}$ is complete as well. Theorem 5.2.4 tells us that since $\mathcal{M}$ is s-learnable, $\mathcal{N}$ is t-observable. This implies that every run of $\mathcal{N}$ is feasible which, together with $\mathcal{N}$'s completeness, means that $\mathcal{N}$ is s-learnable. □

5.2.1 Why not all s-Learnable MMTs are t-Observable

We compared the property of t-observability with that of s-learnability in Section 5.1. Bruyère et al. [2024] introduced a method that makes a complete MMT $\mathcal{M}$ s-learnable by computing what they call its zone MMT, $zone(\mathcal{M})$. Zone MMT $zone(\mathcal{M})$ has for each of its states a pair (q, Z), with $q \in Q^{\mathcal{M}}$ a state of $\mathcal{M}$, and $Z \subseteq \mathsf{Val}(\mathcal{X}(q))$ a **zone**: a set of valuations over $\mathcal{X}(q)$. The set $\mathcal{X}^{zone(\mathcal{M})}((q, Z))$ of timers active in (q, Z) is simply defined as $\mathcal{X}^{zone(\mathcal{M})}((q, Z)) = \mathcal{X}^{\mathcal{M}}(q)$, the set of timers active in q. Therefore, if $\mathcal{M}$ violates the second t-observability requirement, then $zone(\mathcal{M})$ might violate this requirement as well. This implies that MMTs that are made s-learnable with Bruyère et al. [2024]'s method are not guaranteed to be t-observable. We highlight that whereas Bruyère et al. [2024]'s process for making MMTs s-learnable can lead to an increase in the model's state space, our method for making s-learnable MMTs t-observable always returns MMTs with the same number of states as the original model.

5.3 Observation Trees and Functional Simulations

Much like our conformance testing method for MM1Ts from Chapter 3, our method for MMTs relies on the notions of observation trees and functional simulations. We again use an observation tree which simulates both the specification, and the SUT. Our method uses MMTs for the SUTs, and gMMTs for the specifications. We therefore define functional simulations for both of these model types. Tree-shaped, partial MMTs form the basis of the observation trees for both MMTs and gMMTs.

Definition 5.3.1 (Tree MMT). A partial MMT $\mathcal{T}$ is a **tree** iff, for each state $q \in Q$ there is a unique action sequence $\mathsf{access}(q) \in A^*$, such that $\delta^*(q_{\mathcal{I}}, \mathsf{access}(q)) = q$. Each tree state $q \in Q$ has a unique parent state $\mathsf{parent}(q) \in Q$, such that $\mathsf{parent}(q) \xrightarrow{i} q \in runs(\mathcal{T})$ and $i \in A$.

We use Bruyère et al. [2024]'s observation trees:

Definition 5.3.2 (Observation tree MMT). An **observation tree** MMT $\mathcal{T}$ is a t-observable tree MMT $\mathcal{T} = (Q, q_{\mathcal{I}}, X, I, O, \mathcal{X}, \delta, \lambda, \tau)$ such that:

- $X = \{x_q \mid q \in Q \setminus \{q_{\mathcal{I}}\}\}$,
- $\forall q \xrightarrow[(x,c)]{i} q'$ with $i \in I$: $x = x_{q'}$,

The use of our notion of t-observability allowed us to define observation trees a bit more succinctly than in Bruyère et al. [2024].

We use Bruyère et al. [2024]'s functional simulations, which specify for observation tree MMTs how they simulate s-learnable MMTs:

Definition 5.3.3 (Functional MMT simulation). Let $\mathcal{T}$ be an observation tree, and let $\mathcal{M}$ be an s-learnable MMT with the same set of inputs I. A **functional MMT simulation** $\langle f_s, f_t, f_u \rangle : \mathcal{T} \to \mathcal{M}$ is a triple of a map $f_s : Q^{\mathcal{T}} \to Q^{\mathcal{M}}$, a map $f_t : \cup_{q \in Q^{\mathcal{T}}} \mathcal{X}^{\mathcal{T}}(q) \to X^{\mathcal{M}}$, and a map $f_u : Q^{\mathcal{T}} \times A^{\mathcal{T}} \to U^{\mathcal{M}}$. We lift f_t to actions such that:

- $f_t(i) = i$ for every $i \in I$, and
- $f_t(\mathsf{to}[x]) = \mathsf{to}[f_t(x)]$ for every $x \in \mathsf{dom}(f_t)$.

We require that $\langle f_s, f_t, f_u \rangle$ preserves initial states, active timers, and transitions:

$$f_s(q_I^{\mathcal{T}}) = q_{\mathcal{I}} \qquad \text{(FMS0)}$$

$$\forall q \in Q^{\mathcal{T}}, \forall x \in \mathcal{X}^{\mathcal{T}}(q) : f_t(x) \in \mathcal{X}^{\mathcal{M}}(f_s(q)) \qquad \text{(FMS1)}$$

$$\forall q \in Q^{\mathcal{T}}, \forall x, y \in \mathcal{X}^{\mathcal{T}}(q) : x \neq y \;\Rightarrow\; f_t(x) \neq f_t(y) \qquad \text{(FMS2)}$$

$$\forall q \xrightarrow[(x,c)]{i/o} q' : f_s(q) \xrightarrow[(f_t(x),c)]{f_t(i)/o} f_s(q') \qquad \text{(FMS3)}$$

$$\forall q \xrightarrow[\bot]{i/o} q' : f_s(q) \xrightarrow{f_t(i)/o} f_s(q') \qquad \text{(FMS4)}$$

We define f_u as, for all $q \in Q^{\mathcal{T}}$ and all $i \in A^{\mathcal{T}}$:

$$f_u(q,i) = \tau^{\mathcal{M}}(f_s(q), f_t(i)).$$

We use condition (FMS3) to lift f_u to timer updates. For all timer updates $u \in U^{\mathcal{T}}$:

$$f_u(u) = \begin{cases} (f_t(x), c) & \text{if } u = (x,c) \\ \bot & \text{if } u = \bot. \end{cases}$$

We use conditions (FMS3) and (FMS4) to lift $\langle f_s, f_t, f_u \rangle$ to runs. Let $\pi = q_0 \xrightarrow[u_1]{i_1/o_1} q_1 \dots \xrightarrow[u_n]{i_n/o_n} q_n \in runs(\mathcal{T})$. Then:

$$\langle f_s, f_t, f_u \rangle(\pi) = f_s(q_0) \xrightarrow[f_u(q_0,i_1)]{f_t(i_1)/o_1} f_s(q_1) \dots \xrightarrow[f_u(q_{n-1},i_n)]{f_t(i_n)/o_n} f_s(q_n) \in runs(\mathcal{M}).$$

We use this lifting to add the following requirement for $\langle f_s, f_t, f_u \rangle$:

$$\forall \pi \in runs(\mathcal{T}) \colon \langle f_s, f_t, f_u \rangle(\pi) \text{ is } y-\text{spanning} \;\Rightarrow\; \exists x \colon \pi \text{ is } x-\text{spanning} \;\wedge\; f_t(x) = y. \tag{FMS5}$$

We say that $\mathcal{T}$ is an **observation tree for** $\mathcal{M}$ if there exists a functional MMT simulation $\langle f_s, f_t, f_u \rangle \colon \mathcal{T} \to \mathcal{M}$.

This definition is the exact same as the one Bruyère et al. [2024] provides for functional simulations between observation trees and MMTs, apart from our addition and use of the lifting of the timer map f_t to timer updates.

Lemma 5.3.1. Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be an s-learnable MMT, and let $\langle f_s, f_t, f_u \rangle \colon \mathcal{T} \to \mathcal{M}$ be a functional MMT simulation. Then:

$$\forall \pi \in runs(\mathcal{T}) \colon \pi \text{ is } x\text{-spanning} \Rightarrow \langle f_s, f_t, f_u \rangle(\pi) \text{ is } f_t(x)\text{-spanning}.$$

The proof of Lemma 5.3.1 can be found in Appendix C.1.1.

Mind that Bruyère et al. [2024]'s Corollary 3.5 also applies to this notion of functional MMT simulations.

Functional gMMT simulations need to account for timer renaming, due to which a single timer of the observation tree can correspond to multiple distinct timers of the gMMT. We extended the active timer mapping with an argument for the "current" observation tree state, so that observation tree timers that remain active across multiple consecutive state transitions can be mapped to the correct timers of the gMMT.

Definition 5.3.4 (Functional gMMT simulations). Let $\mathcal{T}$ be an observation tree, and let $\mathcal{M}$ be an s-learnable gMMT with the same set of inputs I. A **functional gMMT simulation** $\langle f_s, f_t, f_u \rangle \colon \mathcal{T} \to \mathcal{M}$ is a triple of a map $f_s \colon Q^{\mathcal{T}} \to Q^{\mathcal{M}}$, a map $f_t \colon Q^{\mathcal{T}} \times \cup_{q \in Q^{\mathcal{T}}} \mathcal{X}^{\mathcal{T}}(q) \to X^{\mathcal{M}}$, and a map $f_u \colon Q^{\mathcal{T}} \times A^{\mathcal{T}} \to (X^{\mathcal{M}} \to (X^{\mathcal{M}} \cup \mathbb{N}^{>0}))$. We lift f_t to actions such that, for every $q \in Q^{\mathcal{T}}$:

- $f_t(q,i) = i$ for every $i \in I$, and
- $f_t(q, \mathsf{to}[x]) = \mathsf{to}[f_t(q,x)]$ for every $x \in \mathcal{X}^{\mathcal{T}}(q)$.

We require that $\langle f_s, f_t, f_u \rangle$ preserves initial states, active timers, transitions, timer updates and timer renaming:

$$f_s(q_{\mathcal{I}}^{\mathcal{T}}) = q_{\mathcal{I}} \tag{FGS0}$$

$$\forall q \in Q^{\mathcal{T}}, \forall x \in \mathcal{X}^{\mathcal{T}}(q) \colon f_t(q,x) \in \mathcal{X}^{\mathcal{M}}(f_s(q)) \tag{FGS1}$$

$$\forall q \in Q^{\mathcal{T}}, \forall x,y \in \mathcal{X}^{\mathcal{T}}(q) \colon x \neq y \;\Rightarrow\; f_t(q,x) \neq f_t(q,y) \tag{FGS2}$$

$$\forall q \xrightarrow[(x,c)]{i/o} q' \colon f_s(q) \xrightarrow[\mathfrak{r}]{f_t(q,i)/o} f_s(q') \wedge (\mathfrak{r}(f_t(q',x)) = c) \wedge (\forall y \in (\mathcal{X}^{\mathcal{T}}(q') \setminus \{x\}) \colon \mathfrak{r}(f_t(q',y)) = f_t(q,y)) \tag{FGS3}$$

$$\forall q \xrightarrow[\bot]{i/o} q' \colon f_s(q) \xrightarrow[\mathfrak{r}]{f_t(q,i)/o} f_s(q') \wedge (\forall x \in \mathcal{X}^{\mathcal{T}}(q') \colon \mathfrak{r}(f_t(q',x)) = f_t(q,x)) \tag{FGS4}$$

We define f_u as, for all $q \in Q^{\mathcal{T}}$ and all $i \in A^{\mathcal{T}}$:

$$f_u(q,i) = \tau^{\mathcal{M}}(f_s(q), f_t(i)).$$

We use condition (FGS3) to lift the timer map f_t to timer updates. For all observation tree states q, q' and timer updates $u \in U^{\mathcal{T}}$ such that $\exists i \in (I^{\mathcal{T}} \cup TO(\mathcal{X}_0(q)))\colon \delta(q,i) = q' \wedge \tau(q,i) = u$, we get:

$$f_u(q,q',u) = \begin{cases} \{(f_t(q',y), f_t(q,y)) \mid \forall y \in \mathcal{X}^{\mathcal{T}}(q') \setminus \{x\}\} \cup \{(f_t(q',x),c)\} & \text{if } u = (x,c) \\ \{(f_t(q',x), f_t(q,x)) \mid \forall x \in \mathcal{X}^{\mathcal{T}}(q')\} & \text{if } u = \perp. \end{cases}$$

We use conditions (FGS3) and (FGS4) to lift $\langle f_s, f_t, f_u\rangle$ to runs. Let $\pi = q_0 \xrightarrow[u_1]{i_1/o_1} q_1 \dots \xrightarrow[u_n]{i_n/o_n} q_n \in runs(\mathcal{T})$. Then:

$$\langle f_s, f_t, f_u\rangle(\pi) = f_s(q_0) \xrightarrow[f_u(q_0,i_1)]{f_t(q_0,i_1)/o_1} f_s(q_1) \dots \xrightarrow[f_u(q_{n-1},i_n)]{f_t(q_{n-1},i_n)/o_n} f_s(q_n) \in runs(\mathcal{M}).$$

We use this lifting to add the following requirement for $\langle f_s, f_t, f_u\rangle$:

$$\forall (\pi \xrightarrow{\mathsf{to}[x]} q) \in runs(\mathcal{T})\colon \langle f_s, f_t, f_u\rangle(\pi \xrightarrow{\mathsf{to}[x]} q) \text{ is spanning } \Rightarrow\ (\pi \xrightarrow{\mathsf{to}[x]} q) \text{ is } x-\text{spanning}. \tag{FGS5}$$

Functional MMT simulations $\langle g_s, g_t, g_u\rangle\colon \mathcal{T} \to \mathcal{N}$ have the property that, if $q, q' \in Q^{\mathcal{T}} \wedge g_s(q) = g_s(q') \wedge x \in \mathcal{X}^{\mathcal{T}}(q) \wedge x \in \mathcal{X}^{\mathcal{T}}(q')$, then x represents the same timer $g_t(x)$ in $g_s(q')$ as in $g_s(q)$. The first five requirements for functional gMMT simulations do not impose an analogous requirement, as f_t takes both a timer and a state of $\mathcal{T}$ for its arguments. We add the following rule to achieve an analogous effect in functional gMMT simulations:

$$\forall q, q' \in Q^{\mathcal{T}}, \forall x \in \mathcal{X}^{\mathcal{T}}(q), \forall y \in \mathcal{X}^{\mathcal{T}}(q')\colon f_s(q) = f_s(q') \wedge x = y \Rightarrow f_t(q,x) = f_t(q',y). \tag{FGS6}$$

We say that $\mathcal{T}$ is an **observation tree for** $\mathcal{M}$ if there exists a functional gMMT simulation $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$.

Lemma 5.3.2. Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be an s-learnable gMMT, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Then:

$$\forall \pi \in runs(\mathcal{T})\colon \pi \text{ is spanning} \Rightarrow \langle f_s, f_t, f_u\rangle(\pi) \text{ is spanning}.$$

The proof of Lemma 5.3.2 can be found in Appendix C.1.2.

Lemma 5.3.3. Let $\mathcal{T}$ be an observation tree MMT, let $\mathcal{M}$ be a s-learnable gMMT, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Let $\pi = q_0 \xrightarrow[u_1]{i_1/o_1} q_1 \dots \xrightarrow[u_n]{i_n/o_n} q_n \in runs(\mathcal{T})$. Then we have the run $\rho = \langle f_s, f_t, f_u\rangle(\pi) \in runs(\mathcal{M})$. Let $0 \le k \le j \le n$. If $f_t(q_k, x) \in \mathcal{X}^{\mathcal{M}}(f_s(q_k))$, then:

$$renameTo^{\mathcal{M}}_{f_s(q_k) \xrightarrow{f_t(q_k,i_{k+1})\dots f_t(q_{j-1},i_j)} f_s(q_j)}(f_t(q_k,x))\downarrow$$

$$\Longrightarrow$$

$$\left(renameTo^{\mathcal{M}}_{f_s(q_k) \xrightarrow{f_t(q_k,i_{k+1})\dots f_t(q_{j-1},i_j)} f_s(q_j)}(f_t(q_k,x)) = f_t(q_j,x)\right).$$

The proof of Lemma 5.3.3 can be found in Appendix C.1.3.

Lemma 5.3.4. Let $\mathcal{T}$ be an observation tree MMT, let $\mathcal{M}$ be a s-learnable gMMT, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Let $\pi = q_0 \xrightarrow[u_1]{i_1/o_1} q_1 \dots \xrightarrow[u_n]{i_n/o_n} q_n \in runs(\mathcal{T})$. Then we have the run $\rho = \langle f_s, f_t, f_u\rangle(\pi) \in runs(\mathcal{M})$. Let $0 \le k \le j \le n$. If $f_t(q_j, x) \in \mathcal{X}^{\mathcal{M}}(f_s(q_j))$, then:

$$renamesTo^{\mathcal{M}}_{f_s(q_k) \xrightarrow{f_t(q_k,i_{k+1})\dots f_t(q_{j-1},i_j)} f_s(q_j)}(f_t(q_j,x))$$

$$\Longrightarrow$$

$$\left(renamesTo^{\mathcal{M}}_{f_s(q_k) \xrightarrow{f_t(q_k,i_{k+1})\dots f_t(q_{j-1},i_j)} f_s(q_j)}(f_t(q_j,x)) = f_t(q_k,x)\right).$$

The proof of Lemma 5.3.4 can be found in Appendix C.1.4.

5.4 Explored States

Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be a (g)MMT, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional simulation. We sometimes track how much of a (g)MMT's timer-related behavior is captured by the states of an observation tree, because doing so can allow us to tell two observation tree states apart. We discuss the apartness of states in Section 5.5. The first notion we rely on is that of the **enabled explored** observation tree state. This notion was introduced in Bruyère et al. [2024], where such states are called "explored states".

Definition 5.4.1 (Enabled explored states). Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be a (g)MMT, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional simulation. State q is **enabled explored** if $|\mathcal{X}_0^{\mathcal{T}}(q)| = |\mathcal{X}_0^{\mathcal{M}}(f_s(q))|$. The set $\mathcal{E}_{\mathcal{M}}^{\mathcal{T}}$ denotes the maximal set of enabled explored states of $\mathcal{T}$ that induces a subtree that contains $q_{\mathcal{I}}^{\mathcal{T}}$, i.e.:

$$q_{\mathcal{I}}^{\mathcal{T}} \in \mathcal{E}_{\mathcal{M}}^{\mathcal{T}} \qquad \wedge \qquad (\forall q, q' \in Q^{\mathcal{T}}, i \in A^{\mathcal{T}}\colon q' \in \mathcal{E}_{\mathcal{M}}^{\mathcal{T}} \wedge q \xrightarrow{i} q' \implies q \in \mathcal{E}_{\mathcal{M}}^{\mathcal{T}}).$$

We also rely on a similar notion of **active explored** states. The main idea is similar to the one for enabled explored states: an observation tree state q is active explored if each timer that is active in $f_s(q)$ is represented by a timer that is active in q. We however add the requirement that all timer updates that occurred before $f_s(q)$ is reached in $\mathcal{M}$ via $\langle f_s, f_t, f_u\rangle(\mathsf{access}(q))$ are represented by timer updates in $\mathcal{T}$. We thus know for every active explored state not just what timers are active, but also what timer updates caused these timers to be active.

Definition 5.4.2 (Active explored states). Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be a (g)MMT, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional simulation. State q_n is **active explored** if:

1. $|\mathcal{X}^{\mathcal{T}}(q_n)| = |\mathcal{X}^{\mathcal{M}}(f_s(q_n))|$, and
2. for $\mathcal{T}$'s unique run $q_{\mathcal{I}}^{\mathcal{T}} \xrightarrow[u_1]{i_1/o_1} q_1 \dots \xrightarrow[u_n]{i_n/o_n} q_n$ for $\mathsf{access}(q_n)$:
   - if $\mathcal{M}$ is an MMT:
   $$\forall j \in \{1, \dots, n\}\colon \qquad u_j = \bot \iff f_u(q_{j-1}, i_j) = \bot.$$
   - if $\mathcal{M}$ is a gMMT:
   $$\forall j \in \{1, \dots, n\}\colon \qquad u_j = \bot \iff \neg\exists x\colon f_u(q_{j-1}, i_j)(x) \in \mathbb{N}^{>0}.$$

The set $\mathcal{A}_{\mathcal{M}}^{\mathcal{T}}$ denotes the maximal set of active explored states of $\mathcal{T}$ that induces a subtree that contains $q_{\mathcal{I}}^{\mathcal{T}}$, i.e.:

$$q_{\mathcal{I}}^{\mathcal{T}} \in \mathcal{A}_{\mathcal{M}}^{\mathcal{T}} \qquad \wedge \qquad (\forall q, q' \in Q^{\mathcal{T}}, i \in A^{\mathcal{T}}\colon \quad q' \in \mathcal{A}_{\mathcal{M}}^{\mathcal{T}} \wedge q \xrightarrow{i} q' \implies q \in \mathcal{A}_{\mathcal{M}}^{\mathcal{T}}).$$

Let $\mathcal{T}$ be an observation tree for an MMT $\mathcal{M}$, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional MMT simulation. We can quite easily see from the definitions of enabled and active explored states and of functional MMT simulations that if all states along a run $\pi \in runs(\mathcal{T})$ of $\mathcal{T}$ are both enabled and active explored with respect to $\mathcal{M}$, then π contains a matching timer update for all timer updates found along π's corresponding run in $\mathcal{M}$.

Proposition 5.4.1. Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be an s-learnable (g)MMT, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional simulation. Let $\pi = q_0 \xrightarrow[u_1]{i_1/o_1} \dots \xrightarrow[u_n]{i_n/o_n} q_n \in runs(\mathcal{T})$. If:

$$\forall i \in \{0, \dots, n\}\colon \qquad q_i \in \mathcal{E}_{\mathcal{M}}^{\mathcal{T}} \quad \wedge \quad q_i \in \mathcal{A}_{\mathcal{M}}^{\mathcal{T}},$$

then there is a run:

$$\rho = p_0 \xrightarrow[u'_1]{i'_1/o'_1} \dots \xrightarrow[u'_n]{i'_n/o'_n} p_n \in runs(\mathcal{M}),$$

such that:

- $\forall i \in \{0, \ldots, n\}\colon f_s(q_i) = p_i$, and
- If $\mathcal{M}$ is an:
  - MMT: $\forall j \in \{1, \ldots, n\}\colon f_t(i_j) = i'_j \wedge f_u(u_j) = u'_j$.
  - gMMT: $\forall j \in \{1, \ldots, n\}\colon f_t(q_{j-1}, i_j) = i'_j \wedge f_u(q_{j-1}, q_j, u_j) = u'_j$.

5.5 Timer Matchings and Apartness

The notion of apartness for timers from Bruyère et al. [2024] applies to t-observable (g)MMTs. If two distinct timers x and y are both active in the same state q of a t-observable (g)MMT, then they must have been started in different transitions of any run that traverses q. For t-observable (g)MMTs, if $x, y \in \mathcal{X}^{\mathcal{M}}(q)$, then there are always x- and y-spanning runs that traverse q. It is thus possible to observe the fact that x and y are active in q. The fact that x and y were last (re)started in different transitions implies that it is possible to interact with the (g)MMT in a way that reveals x and y to be distinct timers.

Definition 5.5.1 (Apartness of timers of t-observable (g)MMTs). Let $\mathcal{M}$ be a t-observable (g)MMT, and let $x, y \in X^{\mathcal{M}}$. We say that x and y are **apart** for state $q \in Q^{\mathcal{M}}$, denoted $x\ {}^{t}\#_q\ y$, iff $x, y \in \mathcal{X}^{\mathcal{M}}(q)$ and $x \neq y$. We write $x\ {}^{t}\#\ y$ iff $\exists q' \in Q^{\mathcal{M}}\colon x\ {}^{t}\#_{q'}\ y$.

We, like Bruyère et al. [2024], rely on the concept of matchings to encode the equivalence of timers.

Definition 5.5.2 (Matching). Let S and T be two sets of (g)MMT states. A relation $m \subseteq S \times T$ is a **matching** from S to T if it is an injective partial function. We write $m\colon S \leftrightarrow T$ if m is a matching from S to T. A matching m is **maximal** if it is total or surjective.

Let $\mathcal{M}$ be a t-observable (g)MMT with $q, q' \in Q^{\mathcal{M}}$, and let $m\colon \mathcal{X}^{\mathcal{M}}(q) \leftrightarrow \mathcal{X}^{\mathcal{M}}(q')$ be a matching, denoted by abuse of notation as $m\colon q \leftrightarrow q'$. We say that m is **valid** if it never matches timers to timers from which they are apart, i.e., $\forall x \in \mathsf{dom}(m)\colon \neg(x\ {}^{t}\#\ m(x))$. Like Bruyère et al. [2024], we also lift matchings m to actions:

$$m(i) = \begin{cases} i & \text{if } i \in I \\ \mathsf{to}[m(x)] & \text{if } i = \mathsf{to}[x] \text{ with } x \in \mathsf{dom}(m). \end{cases}$$

Let $\pi = q_0 \xrightarrow{i_1} q_1 \xrightarrow{i_2} \ldots \xrightarrow{i_n} q_n \in runs(\mathcal{M})$ and $\pi' = q'_0 \xrightarrow{i'_1} q'_1 \xrightarrow{i'_2} \ldots \xrightarrow{i'_n} q_n \in runs(\mathcal{M})$ be two feasible runs of $\mathcal{M}$. If there is a valid matching $m\colon q_0 \leftrightarrow q'_0$ between q_0 and q'_0, then we follow Bruyère et al.'s example in lifting m to the runs π and π', granted that π and π' are matching for m. The conditions under which π and π' are matching, and the way that m is lifted to π and π' depend on whether $\mathcal{M}$ is an observation tree, an MMT, or a gMMT:

For observation tree MMTs For observation tree MMTs, we have the same conditions for when a matching matches two runs, and the same lifting of matchings to runs as those used in Bruyère et al. [2024]: For π' to match π, we require that for all $j \in \{1, \ldots, n\}$:

- If $i_j \in I$, then $i'_j = i_j$.
- If $i_j = \mathsf{to}[x]$ for some $x \in X$ then there are two possibilities:
  1. If $x \in \mathcal{X}(q_0)$, then $i'_j = \mathsf{to}[m(x)]$; and
  2. If $x = x_{q_k}$ for some $0 < k < j$ (x is started along the run), then $i'_j = \mathsf{to}[x_{q'_k}]$ with the same k.

  In the first case, i'_j must use the "same" timer according to m. In the second case, i'_j must use the "same" timer according to the updates along the runs.

When π and π' match, we write $m^{\pi}_{\pi'}\colon \pi \leftrightarrow \pi'$ with $m^{\pi}_{\pi'} \coloneqq m \cup \{(x_{q_k}, x_{q'_k}) \mid 0 < k \leq n\}$ and $i'_j = m^{\pi}_{\pi'}(i_j)$ for every j.

For t-observable MMTs For π' to match π, we require that for all $j \in \{1, \ldots, n\}$:

- If $i_j \in I$, then $i'_j = i_j$.
- If $i_j = \mathsf{to}[x]$ for some $x \in X$ then there are two possibilities related to $k = \mathit{lastStartedAt}_{q_0 \xrightarrow{i_1 \ldots i_{j-1}} q_{j-1}}(x)$:
  1. If $k = 0$, then $i'_j = \mathsf{to}[m(x)]$ and $\mathit{lastStartedAt}_{q'_0 \xrightarrow{i'_1 \ldots i'_{j-1}} q'_{j-1}}(m(x)) = 0$; and
  2. If $k > 0$, then $q'_{k-1} \xrightarrow{i'_k \ldots i'_j} q'_j$ is spanning.

  In the first case, i'_j must use the "same" timer according to m. The timer may not be (re)started along π', since it isn't (re)started along π. In the second case, i'_j must use the "same" timer according to the spannings along the runs.

Assume that $m^{\pi}_{\pi'} \colon \pi \leftrightarrow \pi'$. We again lift m to π and π'. We have to account for the fact that either of these runs can traverse the same state multiple times. Our approach is to map timers that have timeouts in π based on the indices of π at which these timeouts occur. If $\delta(q_{j-1}, \mathsf{to}[x])\downarrow$ and x was last (re)started at index k of π, then we know from the fact that π and π' are matching for m that there is a timer x', such that $\delta(q'_{j-1}, \mathsf{to}[x'])\downarrow$ and x' was last restarted at index k of π'. We thus define $m^{\pi}_{\pi'}$ such that in this scenario, $m^{\pi}_{\pi'}(x, j-1) = x'$:

$$m^{\pi}_{\pi'}(x, l) \coloneqq \begin{cases} m(x) & \text{if } l = 0 \vee k = 0 \\ \pi_1(\tau(q'_{k-1}, i'_k)) & \text{if } l > 0 \wedge k > 0 \\ \text{undefined} & \text{if } k = \bot \end{cases}$$

where $k = \mathit{lastStartedAt}_{q_0 \xrightarrow{i_1 \ldots i_l} q_l}(x)$ and $i'_j = m^{\pi}_{\pi'}(i_j, j-1)$ for every j.

For t-observable gMMTs For π' to match π, we require that for all $j \in \{1, \ldots, n\}$:

- If $i_j \in I$, then $i'_j = i_j$.
- If $i_j = \mathsf{to}[x]$ for some $x \in X$ then there are two possibilities related to $k = \mathit{lastStartedAt}_{q_0 \xrightarrow{i_1 \ldots i_{j-1}} q_{j-1}}(x)$:
  1. If $k = 0$, then $i'_j = \mathsf{to}[x']$ and $\mathit{lastStartedAt}_{q'_0 \xrightarrow{i'_1 \ldots i'_{j-1}} q'_{j-1}}(x') = 0$, where:
  $$x' = \mathit{renameTo}_{q'_0 \xrightarrow{i'_1 \ldots i'_{j-1}} q'_{j-1}}(m(\mathit{renamesTo}_{q_0 \xrightarrow{i_1 \ldots i_{j-1}} q_{j-1}}(x)));$$
  and
  2. If $k > 0$, then $q'_{k-1} \xrightarrow{i'_k \ldots i'_j} q'_j$ is spanning.

  In the first case, i'_j must use the "same" timer according to m. We need to account for timer renamings that may occur along the two runs. The timer may not be (re)started along π', since it isn't (re)started along π. In the second case, i'_j must use the "same" timer according to the spannings along the runs. The notion of spanning gMMT runs implicitly accounts for timer renamings.

Assume that π and π' are matching for $m^{\pi}_{\pi'} \colon \pi \leftrightarrow \pi'$. We again lift m to π and π'. The idea behind our approach is the same as in the one for t-observable MMTs. The only functional differences follow from the need to account for timer renamings:

$$m^{\pi}_{\pi'}(x, l) \coloneqq \begin{cases} m(x) & \text{if } l = 0 \\ \mathit{renameTo}_{q'_0 \xrightarrow{i'_1 \ldots i'_l} q'_l}(m(\mathit{renamesTo}_{q_0 \xrightarrow{i_1 \ldots i_l} q_l}(x))) & \text{if } l > 0 \wedge k = 0 \\ \mathit{renameTo}_{q'_k \xrightarrow{i'_{k+1} \ldots i'_l} q'_l}(\mathit{timerStartedAt}(q'_0 \xrightarrow{i'_1 \ldots i'_k})) & \text{if } l > 0 \wedge k > 0 \\ \text{undefined} & \text{if } k = \bot \end{cases}$$

where $k = \mathit{lastStartedAt}_{q_0 \xrightarrow{i_1 \dots i_l} q_l}(x)$, and $i'_j = m^{\pi}_{\pi'}(i_j, j-1)$ for every j.

Lemma 5.5.1. Let $\mathcal{M}$ be a t-observable gMMT, and let $x \in X^{\mathcal{M}}$ be a timer of $\mathcal{M}$. If the runs $\pi = q_0 \xrightarrow{i_1 \dots i_n} q_n \in \mathit{runs}(\mathcal{M})$ and $\pi' = q'_0 \xrightarrow{i'_1 \dots i'_n} q'_n \in \mathit{runs}(\mathcal{M})$ are matching with $m^{\pi}_{\pi'} \colon \pi \leftrightarrow \pi'$ and $q_{k-1} \xrightarrow{i_k \dots (i_j = \mathsf{to}[x])} q_j$ is a spanning sub-run of π, then $q'_{k-1} \xrightarrow{i'_k \dots (i'_j = \mathsf{to}[m^{\pi}_{\pi'}(x, j-1)])} q'_j$ is a spanning sub-run of π'.

Proof. Since $q_{k-1} \xrightarrow{i_k \dots (i_j = \mathsf{to}[x])} q_j$ is a spanning sub-run of π:

1. π has action $i_j = \mathsf{to}[x]$ for the timer $x \in X$, and
2. $\mathit{lastStartedAt}_{q_0 \xrightarrow{i_1 \dots i_{j-1}} q_{j-1}}(x) = k$.

The fact that $q_{k-1} \xrightarrow{i_k \dots (i_j = \mathsf{to}[x])} q_j$ is a spanning sub-run of π implies that $k > 0$, since it tells us that timer x was started in one of π's transitions. Since π starts from q_0, this implies that $1 \leq k \leq j-1$. The fact that π and π' are matching thus directly tells us that $q'_{k-1} \xrightarrow{i'_k \dots (i'_j = \mathsf{to}[m^{\pi}_{\pi'}(x, j-1)])} q'_j$ is a spanning sub-run of π'. □

5.5.1 Reading Runs

For a fixed run $\pi \in \mathit{runs}(\mathcal{M})$ and matching m, there is at most one run $\pi' \in \mathit{runs}(\mathcal{M})$ such that $m^{\pi}_{\pi'} \colon \pi \leftrightarrow \pi'$. This is also the case in Bruyère et al. [2024]. We follow Bruyère et al. [2024]'s example in denoting this unique run π' by $\mathit{read}^m_{\pi}(q'_0)$ if it exists. Otherwise, $\mathit{read}^m_{\pi}(q'_0)$ is left undefined. We say that this function "reads" π from q'_0, using m to rename the timers.

5.5.2 Apartness of States

Similarly to how we have different notions of matchings for observation tree MMTs, for t-observable MMTs, and for t-observable gMMTs, we also have corresponding notions of apartness for all three of these model types. All three of these notions follow the same principle as the apartness of observation tree states from Bruyère et al. [2024].

We also use Bruyère et al. [2024]'s distinction between structural and behavioral apartness:
Two states q_0 and q'_0 are **apart** under a matching m, if we have runs $\pi = q_0 \xrightarrow{w}$ and $\pi' = \mathit{read}^m_{\pi}(q'_0)$ such that:

- π and π' exhibit different behavior, in which case we say that the apartness is **behavioral**; or
- For observation tree $\mathcal{T}$, $m^{\pi}_{\pi'}$ is invalid in the sense that it matches two distinct observation tree timers $x, m^{\pi}_{\pi'} \in X$ that are first started at different points along the same run in $\mathcal{T}$, and that can therefore not represent the same timer of $\mathcal{M}$. We then say that the apartness is **structural**. Formally, we can say that there is a structural apartness when, for some timer $x \in \mathsf{dom}(m^{\pi}_{\pi'})$, $x \ {}^{t}\# \ m^{\pi}_{\pi'}(x)$.

For observation tree MMTs, we use apartness to constructively determine when two states must represent distinct states of the model that is "observed" by the observation tree, based on the limited information in the observation tree. For MMTs and gMMTs, we use apartness to denote when two states cannot be said to exhibit the same observable behavior. We therefore only need structural apartness for the apartness of observation tree MMT states, and not for the apartness of (g)MMT states.

The precise definition of apartness depends on the type of the model. We used the state apartness defined in Bruyère et al. [2024] as a basis for our apartness for observation tree MMT states. We have:

Definition 5.5.3 (Apartness of observation tree MMT states). Let $\mathcal{T}$ be an observation tree MMT for an s-learnable, t-observable (g)MMT $\mathcal{M}$. Two states $q_0, q'_0 \in Q^{\mathcal{T}}$ are m-apart with $m \colon q_0 \leftrightarrow q'_0$, denoted $q_0 \#^m q'_0$, if there are $\pi = q_0 \xrightarrow{i_1} \dots \xrightarrow[u]{i_n/o} q_n$ and $\pi' = q'_0 \xrightarrow{i'_1} \dots \xrightarrow[u']{i'_n/o'} q'_n$ with $m^{\pi}_{\pi'} \colon \pi \leftrightarrow \pi'$, and:

- **Structural apartness** there exists $x \in \mathsf{dom}(m^{\pi}_{\pi'})$, such that $x \ {}^{t}\# \ m^{\pi}_{\pi'}(x)$, or

- **Behavioral apartness** one of the following holds:

$$
\begin{array}{ll}
o \neq o' & \text{(outputs)} \\
u = (x, c) \wedge u' = (x', c') \wedge c \neq c' & \text{(constants)} \\
q_n, q'_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}} \wedge (u = \bot \Leftrightarrow u' \neq \bot) & \text{(updating)} \\
q_n, q'_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}} \wedge |\mathcal{X}(q_n)| \neq |\mathcal{X}(q'_n)| & \text{(active sizes)} \\
q_n, q'_n \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}} \wedge |\mathcal{X}_0(q_n)| \neq |\mathcal{X}_0(q'_n)| & \text{(enabled sizes)} \\
q_n, q'_n \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}} \wedge \exists x \in \mathsf{dom}(m^{\pi}_{\pi'}) \colon (x \in \mathcal{X}_0(q_n) \Leftrightarrow m^{\pi}_{\pi'}(x) \notin \mathcal{X}_0(q'_n)) & \text{(enabled)}
\end{array}
$$

The word $\sigma = i_1 \dots i_n \in A^{\mathcal{T}*}$ is called a **witness** of $q_0 \#^m q'_0$. We write $q_0 \# q'_0$ if $q_0 \#^m q'_0$ for all maximal matchings $m \colon q_0 \leftrightarrow q'_0$.

The difference between our notion of observation tree apartness and the one from Bruyère et al. [2024] is our addition of the (updating) and (active sizes) conditions. Note that these rules rely on the notion of active explored states, which wasn't used in Bruyère et al. [2024].

Our notion of apartness of t-observable MMT states is very similar to the one for observation trees, the only difference being the omission of the sets of active and enabled explored states:

Definition 5.5.4 (Apartness of t-observable MMT states). Two states q_0, q'_0 are m-apart with $m \colon q_0 \leftrightarrow q'_0$, denoted $q_0 \#^m q'_0$, if there are $\pi = q_0 \xrightarrow{i_1} \dots \xrightarrow[u]{i_n/o} q_n$ and $\pi' = q'_0 \xrightarrow{i'_1} \dots \xrightarrow[u']{i'_n/o'} q'_n$ with $m^{\pi}_{\pi'} \colon \pi \leftrightarrow \pi'$, and:

- **Behavioral apartness** one of the following holds:

$$
\begin{array}{ll}
o \neq o' & \text{(outputs)} \\
u = (x, c) \wedge u' = (x', c') \wedge c \neq c' & \text{(constants)} \\
u = \bot \Leftrightarrow u' \neq \bot & \text{(updating)} \\
|\mathcal{X}(q_n)| \neq |\mathcal{X}(q'_n)| & \text{(active sizes)} \\
|\mathcal{X}_0(q_n)| \neq |\mathcal{X}_0(q'_n)| & \text{(enabled sizes)} \\
\exists x \in X \colon m^{\pi}_{\pi'}(x, n)\!\downarrow \wedge (x \in \mathcal{X}_0(q_n) \Leftrightarrow m^{\pi}_{\pi'}(x, n) \notin \mathcal{X}_0(q'_n)) & \text{(enabled)}
\end{array}
$$

The word $\sigma = i_1 \dots i_n \in A^{\mathcal{T}*}$ is called a **witness** of $q_0 \#^m q'_0$, which we denote by $\sigma \vdash q_0 \#^m q'_0$. We write $q_0 \# q'_0$ if $q_0 \#^m q'_0$ for all maximal matchings $m \colon q_0 \leftrightarrow q'_0$.

Our notion of apartness of t-observable gMMT states is mostly the same as the one for t-observable MMT states:

Definition 5.5.5 (Apartness of t-observable gMMT states). Two states q_0, q'_0 are m-apart with $m \colon q_0 \leftrightarrow q'_0$, denoted $q_0 \#^m q'_0$, if there are $\pi = q_0 \xrightarrow{i_1} \dots \xrightarrow[\mathfrak{r}]{i_n/o} q_n$ and $\pi' = q'_0 \xrightarrow{i'_1} \dots \xrightarrow[\mathfrak{r}']{i'_n/o'} q'_n$ with $m^{\pi}_{\pi'} \colon \pi \leftrightarrow \pi'$, and:

- **Behavioral apartness** one of the following holds:

$$
\begin{array}{ll}
o \neq o' & \text{(outputs)} \\
(\exists x \in \mathcal{X}(q_n) \colon \mathfrak{r}(x) = c \in \mathbb{N}^{>0}) \wedge (\exists x' \in \mathcal{X}(q'_n) \colon \mathfrak{r}'(x') = c' \in \mathbb{N}^{>0}) \wedge c \neq c' & \text{(constants)} \\
(\exists x \in \mathcal{X}(q_n) \colon \mathfrak{r}(x) \in \mathbb{N}^{>0}) \Leftrightarrow (\neg \exists x \in \mathcal{X}(q'_n) \colon \mathfrak{r}'(x) \in \mathbb{N}^{>0}) & \text{(updating)} \\
|\mathcal{X}(q_n)| \neq |\mathcal{X}(q'_n)| & \text{(active sizes)} \\
|\mathcal{X}_0(q_n)| \neq |\mathcal{X}_0(q'_n)| & \text{(enabled sizes)} \\
\exists x \in X \colon m^{\pi}_{\pi'}(x, n)\!\downarrow \wedge (x \in \mathcal{X}_0(q_n) \Leftrightarrow m^{\pi}_{\pi'}(x, n) \notin \mathcal{X}_0(q'_n)) & \text{(enabled)}
\end{array}
$$

The word $\sigma = i_1 \dots i_n \in A^{\mathcal{T}*}$ is called a **witness** of $q_0 \#^m q'_0$, which we denote by $\sigma \vdash q_0 \#^m q'_0$. We write $q_0 \# q'_0$ if $q_0 \#^m q'_0$ for all maximal matchings $m \colon q_0 \leftrightarrow q'_0$.

We sometimes rely on the notion of **minimum-length witnesses** of apartness:

Definition 5.5.6. Let $\mathcal{M}$ be a (g)MMT with $q, q' \in Q$. Let $\sigma \in A^*$ such that there exists a matching $m\colon q \leftrightarrow q'$ for which $\sigma \vdash q \#^m q'$. Then σ is a **minimum-length witness** of $q \#^m q'$ iff σ has no proper prefix ρ such that $\rho \vdash q \#^m q'$.

There is an upper bound on the length of any minimum-length witnesses for the apartness of two states of all three of these model types:

Lemma 5.5.2. Let $\mathcal{M}$ be a t-observable (observation tree) (g)MMT with $|Q| = n$, and let $q, q' \in Q$. Let $m\colon q \leftrightarrow q'$. If $q \#^m q'$, then the maximum length that any minimum-length witness $\sigma \in (A)^*$ of $\sigma \vdash q \#^m q'$ may need to have is n.

Proof. We know from the notions of apartness of t-observable (observation tree) (g)MMT states that $q \#^m q'$ can be the result of either:

1. conditions on states that are reached from q and q', as is the case for the (active sizes), (enabled sizes) and (enabled) conditions for apartness; or of
2. conditions on state transitions that are reached from q and q', as is the case for the (outputs), (constants) and (updating) conditions for apartness.

In the first case, the witness may need to be able to reach any of $\mathcal{M}$'s n states. Since we already start in one of $\mathcal{M}$'s states, we only need to be able to perform at most $n-1$ state transitions to reach any of $\mathcal{M}$'s remaining states. In the second case, the witness may need to be able to reach any of $\mathcal{M}$'s states, from which it needs to be able to take one more state transition. Across both cases, we only need the minimum-length witness to be $n - 1 + 1 = n$ actions in length. □

5.6 Stratification

In this short section, we explain how we adapt observation tree stratifications to the MMT setting.

Definition 5.6.1 (Stratification for MMTs). Let $\mathcal{T}$ be an observation tree MMT for an s-learnable (g)MMT $\mathcal{M}$. Then $\mathcal{T}$ and $\mathcal{M}$ have the same set of inputs, I. Let $C \subseteq I \cup TO(\mathbb{N}^{>0})$ be a nonempty, finite, prefix closed set of symbolic words. Then C induces a **stratification** of $Q^{\mathcal{T}}$ as follows:

1. A state q of $\mathcal{T}$ is called a **basis state** iff $\overline{\mathsf{access}(q)} \in C$. We write B to denote the set of basis states: $B \coloneqq \{q \in Q^{\mathcal{T}} \mid \overline{\mathsf{access}(q)} \in C\}$. Note that, since C is nonempty and prefix closed, initial state $q_I^{\mathcal{T}}$ is in the basis, and all states on the path leading to a basis state are basis states as well.
2. We write F^0 for the set of immediate successors of basis states that are not basis states themselves: $F^0 \coloneqq \{q' \in Q^{\mathcal{T}} \setminus B \mid \exists q \in B, i \in A^{\mathcal{T}}\colon q' = \delta(q, i)\}$. We refer to F^0 as the **0-level frontier**.
3. For $k > 0$, the ***k*-level frontier** F^k is the set of immediate successors of $k-1$-level frontier states: $F^k \coloneqq \{q' \in Q^{\mathcal{T}} \mid \exists q \in F^{k-1}, i \in A^{\mathcal{T}}\colon q' = \delta(q, i)\}$.

We often use $F^{<k}$ to denote the set $F^0 \cup \cdots \cup F^{k-1}$ of the states in the first k frontiers, and $F^{\leq k}$ to denote the set $F^0 \cup \cdots \cup F^k$ of all states in the first $k+1$ frontiers.
We say that basis B is **complete** if:

- for each $\mathtt{w} \in C$ there is a state $q \in B$ with $\delta^*(\mathtt{w}) = q$, and if
- for each $q \in B$:
  - for each $i \in I\colon \delta(q, i)\!\downarrow$, and
  - $q \in \mathcal{E}_{\mathcal{M}}^{\mathcal{T}}$.

For $k \in \mathbb{N}$, the k-level frontier is **complete** if for each $q \in F^k$:

- for each $i \in I$: $\delta(q, i)\downarrow$, and
- $q \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$.

For every state $q \in Q^{\mathcal{T}}$, we define the **candidate set** $\mathcal{C}(q)$ as the set of basis states for which there is at least one matching for which they are not apart from q: $\mathcal{C}(q) \coloneqq \{q' \in B \mid \neg(q' \# q)\}$. A state $q \in Q^{\mathcal{T}}$ is **identified** if its candidate set is a singleton, and **isolated** if its candidate set is empty.

5.7 Requirements for the Specification

Our conformance testing procedure imposes several requirements on the specification. In this section, we specify what properties we expect the specification to have. We also explain why we use gMMTs instead of MMTs for the specifications.

5.7.1 Requirements

As with our k-A-complete conformance testing method for MM1Ts, our method for MMTs requires that the specification model is connected, minimal, s-learnable, and t-observable.

We define minimality for gMMTs in terms of state apartness:

Definition 5.7.1 (Minimal (g)MMT). An (g)MMT $\mathcal{M}$ is **minimal** iff, for all pairs of states $q, q' \in Q$, $\neg(q \# q') \Leftrightarrow q = q'$.

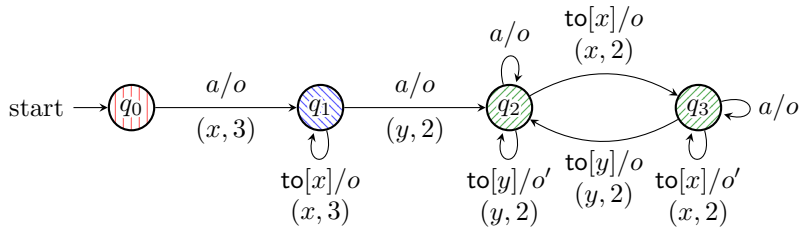
Our MMT conformance testing procedure requires the specification to be provided in the form of a gMMT, rather than an MMT. This requirement actually makes our testing procedure more flexible, since MMTs can easily be converted into symbolically equivalent gMMTs. Bruyère et al. [2024] also provides a way to convert gMMTs into symbolically equivalent MMTs, but this generally leads to a factorial blowup in the size of the state space. Our main reason for supporting gMMT specifications is, however, the fact that not all MMTs can easily be minimized. We explain the problem in Section 5.7.2.

5.7.2 Why gMMTs Should be Easier to Minimize Than MMTs

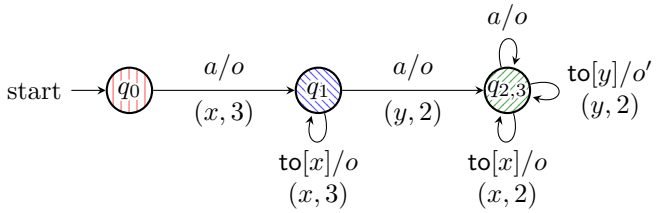
It is not generally possible to minimize an MMT $\mathcal{M}$ such that all of its states are pairwise apart under all maximal matchings, since any two states q_1 and q_2 for which there exists a maximal matching $m\colon q_1 \leftrightarrow q_2$ such that $\neg(q_1 \#^m q_2)$ would have to be represented by the same state $q_{1,2}$ of the minimal MMT. All transitions of the minimal MMT that represent transitions of $\mathcal{M}$ that enter q_1 and q_2 would now have to enter $q_{1,2}$. This could lead to issues, since if there is a run $\pi \in runs(\mathcal{M})$ that traverses q_1 and that requires q_1 to exhibit certain behavior for a timer x, and there is a run $\pi' \in runs(\mathcal{M})$ that traverses q_2 and that requires q_2 to exhibit certain behavior for the same timer x, then the combined state $q_{1,2}$ would have to exhibit the same behavior for x as both q_1 and q_2. This would impose an additional requirement that might not be met by any of the maximal matchings $m\colon q_1 \leftrightarrow q_2$ for which $\neg(q_1 \#^m q_2)$.

Example 5.7.1. Let $\mathcal{M}$ be the MMT of Figure 5.4(a). States $q_2, q_3 \in Q^{\mathcal{M}}$ would exhibit the exact same behavior if we were to swap for them the behavior of timers $x, y \in X^{\mathcal{M}}$. Formally, for the maximal matching $m = \{(x, y), (y, x)\}\colon \neg(q_2 \#^m q_3)$. Therefore, $\neg(q_2 \# q_3)$, which means that $\mathcal{M}$ is not minimal. To minimize $\mathcal{M}$, we would want to merge states q_2 and q_3. To do so, however, would require that we select timers for the two timeout transitions of the new combined state $q_{2,3}$. If we simply select timer x for either transition, and y for the other, then $q_{2,3}$ doesn't exhibit the same symbolic behavior as both q_2 and q_3, which means that the resulting MMT is not symbolically equivalent to $\mathcal{M}$.

We might for instance produce the MMT $\mathcal{M}_n$ of Figure 5.4(b), which we obtained by removing state q_3, and by redirecting the transition from state q_2 to state q_3 into state q_2, which we renamed to $q_{2,3}$. MMT $\mathcal{M}_n$ is minimal, since all of its states are pairwise apart under all maximal matchings. However, $\mathcal{M}_n \not\simeq_{sym} \mathcal{M}$, since the symbolic word $\mathtt{w} = a\ a\ \mathsf{to}[1]\ \mathsf{to}[2]$ would yield for $\mathcal{M}$ a run labelled with the output sequence $o\ o\ o\ o$, and for $\mathcal{M}_n$ a run labelled with the output sequence $o\ o\ o\ o'$. We could alternatively swap timers x and y to make $q_{2,3}$ resemble q_3, but this would clearly result in an MMT that isn't equivalent to $\mathcal{M}$ either. We can thus see that not all MMTs can be minimized by replacing sets of equivalent states with one state that is equivalent to all of them.



(a) An MMT, with $\mathcal{X}(q_0) = \mathcal{X}_0(q_0) = \emptyset$, $\mathcal{X}(q_1) = \mathcal{X}_0(q_1) = \{x\}$, and $\mathcal{X}(q_2) = \mathcal{X}_0(q_2) = \mathcal{X}(q_3) = \mathcal{X}_0(q_3) = \{x, y\}$



(b) A minimal MMT that is similar, but not symbolically equivalent to the MMT above it

Figure 5.4: Two MMTs that are similar, but not symbolically equivalent.

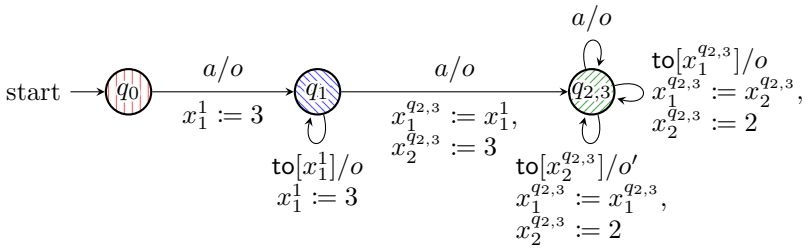


Figure 5.5: A gMMT that is symbolically equivalent to the MMT of Figure 5.4(a)

This problem is not found in gMMTs, since one could rename any timer x in a transition that enters a combined gMMT state $q_{1,2}$ to the timer of $q_{1,2}$ that represents the same behavior from $q_{1,2}$ that x exhibits from either of q_1 and q_2. Therefore, if there is a run $\pi \in runs(\mathcal{M})$ that traverses q_1 and that requires q_1 to exhibit certain behavior for a timer x, and there is a run $\pi' \in runs(\mathcal{M})$ that traverses q_2 and that requires q_2 to exhibit certain behavior for the same timer x, then the combined state $q_{1,2}$ of the minimal gMMT could be made to exhibit the same behavior for a timer x_1 that x exhibits from q_1 in π, while also exhibiting the same behavior for a timer x_2 that x exhibits from q_2.

Example 5.7.2. Let $\mathcal{M}$ be the gMMT of Figure 5.5. This gMMT is minimal, since all of its states are pairwise apart under all maximal matchings. The gMMT is symbolically equivalent to the MMT of Figure 5.4(a). In particular, for the symbolic word $a\ a\ \mathsf{to}[1]\ \mathsf{to}[2]$, they both have the outputs $o\ o\ o\ o$.

5.8 The Testing Procedure

This testing procedure is based on the one we defined in Section 3.3. We non-deterministically expand a tree MMT that serves as an observation tree for both specification gMMT $\mathcal{S}$ and SUT MMT $\mathcal{M}$, until we can either conclude that $\mathcal{M} \approx_{sym} \mathcal{S}$, or we discover that $\mathcal{M} \not\approx_{sym} \mathcal{S}$. We assume the specification to be a connected, minimal, s-learnable, and t-observable gMMT. We assume that any MMT that describes the behavior of the SUT is s-learnable, t-observable, and race-avoiding.

Algorithm 9 shows the main testing procedure.

Algorithm 9: Procedure for testing MMTs

$\mathcal{T} \leftarrow$ a fresh, partial MMT with an initial state $q_I^{\mathcal{T}}$;
$I^{\mathcal{T}} \leftarrow I^{\mathcal{S}}$;
$B \leftarrow \{q_I^{\mathcal{T}}\}$; $\mathcal{E} \leftarrow \{q_I^{\mathcal{T}}\}$; $\mathcal{A} \leftarrow \{q_I^{\mathcal{T}}\}$; $\mathcal{A}_p \leftarrow \emptyset$;
// Complete the basis induced by state cover C
for $\sigma \in C$ **do**
  $c \leftarrow addTransitionsFromSpecSeq_{\mathcal{M}}^{\mathcal{S}}(\sigma)$;
  if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
  $B \leftarrow B \cup \{\delta^{\mathcal{T}^*}(q_I^{\mathcal{T}}, \overline{\sigma})\}$;
end for
while *any of the rules can still be applied* **do**
  [] $\neg(r \# r')$, *for some* $r, r' \in B$ *for which* $r \neq r' \rightarrow$ ▷ Rule (IdentifyBasisStates)
    $c \leftarrow makeObsTreeStatesApart^{\mathcal{S}}(r, r')$;
    if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
  [] $\delta^{\mathcal{T}}(q, i)\uparrow$ *for some* $q \in B \cup F^{<k}$ *and* $i \in I \rightarrow$ ▷ Rule (ExtendFrontiersWithInputs)
    $c \leftarrow addTransition_{\mathcal{M}}^{\mathcal{S}}(q, i)$;
    if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
  [] $q \in (B \cup F^{<k}) \setminus \mathcal{E} \wedge q_{-1} \in \mathcal{E}$ *for some* $q_{-1} \in Q^{\mathcal{T}}$, $i \in I \cup TO(\mathcal{X}_0(q_{-1}))$, *and* $q = \delta^{\mathcal{T}}(q_{-1}, i) \rightarrow$ ▷ Rule (ExtendEnabledExplored)
    $c \leftarrow makeEnabledExplored^{\mathcal{S}}(q)$;
    if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
  [] $q \in (\mathcal{A}_p \cup B \cup F^{\leq k}) \setminus \mathcal{A} \wedge q_{-1} \in \mathcal{A} \wedge canBeMadeActiveExplored(q_{-1}, i, q)$ *for some* $q_{-1} \in Q^{\mathcal{T}}$, $i \in I \cup TO(\mathcal{X}_0(q_{-1}))$, $q = \delta^{\mathcal{T}}(q_{-1}, i) \rightarrow$ ▷ Rule (ExtendActiveExplored)
    if $q \in \mathcal{A}_p$ **then**
      $\mathcal{A}_p \leftarrow \mathcal{A}_p \setminus \{q\}$;
    end
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{q\}$;
  [] $q \in (\mathcal{A}_p \cup B \cup F^{\leq k}) \setminus \mathcal{A} \wedge q_{-1} \in \mathcal{A} \wedge \neg canBeMadeActiveExplored(q_{-1}, i, q) \wedge \delta^{\mathcal{T}}(q', i')\uparrow$, *for some* $q_{-1} \in Q^{\mathcal{T}}$, $i \in I \cup TO(\mathcal{X}_0(q_{-1}))$, $i' \in I$, $q = \delta^{\mathcal{T}}(q_{-1}, i)$, *and* $q' \in Q^{\mathcal{T}}$: $|\mathsf{access}(q')| - |\mathsf{access}(q)| \leq maxNumSUTStates \rightarrow$ ▷ Rule (FindingInputActions)
    $c \leftarrow addTransition_{\mathcal{M}}^{\mathcal{S}}(q', i')$;
    if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
  [] $q \in (\mathcal{A}_p \cup B \cup F^{\leq k}) \setminus \mathcal{A} \wedge q_{-1} \in \mathcal{A} \wedge \neg canBeMadeActiveExplored(q_{-1}, i, q) \wedge (q' \notin \mathcal{E})$, *for some* $q_{-1} \in Q^{\mathcal{T}}$, $i \in I \cup TO(\mathcal{X}_0(q_{-1}))$, $q = \delta^{\mathcal{T}}(q_{-1}, i)$, *and* $q' \in Q^{\mathcal{T}}$: $|\mathsf{access}(q')| - |\mathsf{access}(q)| \leq maxNumSUTStates \rightarrow$ ▷ Rule (FindingTimeoutActions)
    $c \leftarrow makeEnabledExplored^{\mathcal{S}}(q')$;
    if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
  [] $\neg(t \# r) \wedge r \neq r_t$, *for some* $t \in F^{\leq k}$, $r, r_t \in B$, *for which* $r_t = basisStateFor_{(\mathcal{T},C)}^{\mathcal{S}}(t) \rightarrow$ ▷ Rule (IdentifyFrontiers)
    $c \leftarrow makeObsTreeStatesApart^{\mathcal{S}}(t, r)$;
    if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
  [] $r \# t'$, $\neg(r \# t'')$, $\neg(t' \# t'')$, *and* t'' *is identified, for some* $r \in B$, $t' \in F^k$ *and* $t'' \in F^{<k} \rightarrow$ ▷ Rule (ExtendCoTransitivity)
    $c \leftarrow makeObsTreeStatesApart^{\mathcal{S}}(t', t'')$;
    if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
end
return yes;

Algorithm 10: Sub-procedure for determining whether frontier states that should be reachable from a given observation tree state are present in the observation tree

```
1  Procedure areAllStatesInNStepsPresentAndEnabledExplored(q ∈ Q^T, n ∈ ℕ):
2      if n = 0 then
3          return yes;
4      end
5      if q ∉ E then
6          return no;
7      end
8      forall i ∈ I ∪ TO(X_0^T(q)) do
9          if δ^T(q, i)↑ then
10             return no;
11         end
12         q' ← δ^T(q, i);
13         if ¬areAllStatesInNStepsPresentAndEnabledExplored(q', n − 1) then
14             return no;
15         end
16     end
17     return yes;
```

Algorithm 11: Sub-procedure that states whether for a given triple of an observation tree parent state, an action and a successor state, the successor state can be marked as active explored with respect to both the specification and the SUT

```
1  Procedure canBeMadeActiveExplored(q_{-1} ∈ Q^T, i ∈ I^T, q ∈ Q^T):
2      return areAllStatesInNStepsPresentAndEnabledExplored(q, maxNumSUTStates) ∨ (τ^T(q_{-1}, i) ≠
         ⊥ ∧ (∀x ∈ X(q_{-1}): x ∈ X(q)));
```

Algorithm 12: Sub-procedure for finding the basis state that corresponds to a given observation tree state

```
1  Procedure basisStateFor^S_{(T,C)}(q ∈ Q^T):
2      s ← δ^{S*}(s_I^S, access(q));
3      σ ← ρ ∈ C: δ^{S*}(s_I^S, ρ) = s;
4      r ← δ^{T*}(q_I^T, σ);
5      return r;
```

The procedure follows the exact same principle as our MM1T testing procedure from Section 3.3. It uses four rules that are based on the four rules of Algorithm 1. Algorithm 9 also has four additional rules that work towards making certain states of the observation tree enabled and active explored.

In this section, we use a fixed SUT, a fixed observation tree MMT $\mathcal{T}$, and a fixed specification $\mathcal{S}$. We also fix the functional gMMT simulation $\langle f_s, f_t, f_u \rangle \colon \mathcal{T} \to \mathcal{S}$, and the functional MMT simulation $\langle g_s, g_t, g_u \rangle \colon \mathcal{T} \to \mathcal{M}$.

We use the following two sets of enabled and active explored states:

Definition 5.8.1.

$$\mathcal{E} \subseteq \{q \in Q^{\mathcal{T}} \mid (q \in \mathcal{E}^{\mathcal{T}}_{\mathcal{S}}) \wedge (q \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}})\}.$$

Definition 5.8.2.

$$\mathcal{A} \subseteq \{q \in Q^{\mathcal{T}} \mid (q \in \mathcal{A}^{\mathcal{T}}_{\mathcal{S}}) \wedge (q \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}})\}.$$

5.8.1 k-A-Complete Test Suites for MMTs

The notion of fault domains for MMTs is similar to that of fault domains for Mealy machines and MM1Ts. The only differences are that we use symbolic equivalence as our notion of equivalence, and that we distinguish between MMTs and gMMTs.

Definition 5.8.3 (Fault domains and $\mathcal{U}$-completeness). Let $\mathcal{S}$ be an (g)MMT. A **fault domain** is a set $\mathcal{U}$ of MMTs. A test suite $TTS^{\mathcal{S}}$ for $\mathcal{S}$ is **$\mathcal{U}$-complete** if, for each $\mathcal{M} \in \mathcal{U}$, $\mathcal{M}$ only passes $TTS^{\mathcal{S}}$ if $\mathcal{M} \approx_{sym} \mathcal{S}$.

We once again define the relevant fault domains:

Definition 5.8.4. Let $k \in \mathbb{N}$, and let $\mathsf{A} \subseteq \mathtt{A}^*$. Then $\mathcal{U}_k^{\mathsf{A}}$ is the set of all MMTs $\mathcal{M}$ for which, for each state $q \in Q^{\mathcal{M}}$ there are $\sigma \in \mathsf{A}$ and $\rho \in A^{\leq k}$ such that $\delta^*(\sigma) \cdot \rho = q$.

Definition 5.8.5. Let $\mathsf{A} \subseteq \mathtt{A}^*$. Then $\mathcal{U}^{\mathsf{A}}$ is the set of all MMTs $\mathcal{M}$ for which there are $\sigma, \rho \in \mathsf{A}$ with $\sigma \neq \rho$ and $\delta^{\mathcal{M}^*}(\sigma) \approx_{sym} \delta^{\mathcal{M}^*}(\rho)$.

We can now define the relevant notion of k-A-completeness:

Definition 5.8.6 (k-A-complete test suites for MMTs). Let $\mathcal{S}$ be a (g)MMT with a set of inputs I, let $k \in \mathbb{N}$, and let $\mathsf{A} \subseteq I^*$. Then test suite $TS^{\mathcal{S}}$ is **k-A-complete** for $\mathcal{S}$ if, for any SUT MMT $\mathcal{M} \in \mathcal{U}_k^{\mathsf{A}} \cup \mathcal{U}^{\mathsf{A}}$:

$$\mathcal{M} \text{ passes } TS^{\mathcal{S}} \iff \mathcal{M} \approx_{untimed} \mathcal{S}.$$

We rely on knowledge of this fault domain to define our procedure.

5.8.2 The SUT's Maximum Size

Let k be a natural number, and C a minimal prefix-closed state cover for the specification. We will prove our procedure to be k-$\overline{C}$-complete in Section 5.8.9. For now, we just make the assumption that the SUT is an MMT from the corresponding fault domain, $\mathcal{U}_k^{\overline{C}} \cup \mathcal{U}^{\overline{C}}$. This assumption allows us to determine an upper bound on size of the SUT's state space. We use the following property:

Lemma 5.8.1. Let $\mathcal{S}$ and $\mathcal{M}$ be MMTs, and let $\mathcal{T}$ be an observation tree for both $\mathcal{S}$ and $\mathcal{M}$. Let C be a prefix-closed state cover for $\mathcal{S}$, and let B be the basis of a stratification of $Q^{\mathcal{T}}$ induced by $\overline{C}$. Suppose that all states of B are identified. Then $\mathcal{M} \notin \mathcal{U}^{\overline{C}}$.

Proof. Let $\langle g_s, g_t, g_u \rangle \colon \mathcal{T} \to \mathcal{M}$. Suppose that $\sigma, \rho \in \overline{C}$ with $\sigma \neq \rho$. Since B is the basis of a stratification induced by $\overline{C}$, $q = \delta^{\mathcal{T}^*}(\sigma) \in B$ and $q' = \delta^{\mathcal{T}^*}(\rho) \in B$. The fact that $\mathcal{T}$ is a tree thus implies that $q \neq q'$. Since all states of B are identified, they are all pairwise apart. We thus know that $q \# q'$. Lemma C.3.1 now tells us that $g_s(q) \neq g_s(q')$. By Lemma C.6.5, $g_s(q) = \delta^{\mathcal{M}^*}(\sigma)$ and $g_s(q') = \delta^{\mathcal{M}^*}(\rho)$. This implies that $\delta^{\mathcal{M}^*}(\sigma) \neq \delta^{\mathcal{M}^*}(\rho)$, which implies that $\mathcal{M} \notin \mathcal{U}^{\overline{C}}$. □

Therefore, by our assumption that the SUT is an MMT from $\mathcal{U}_k^{\overline{C}} \cup \mathcal{U}^{\overline{C}}$ and by Lemma 5.8.1, the SUT is an MMT from $\mathcal{U}_k^{\overline{C}}$. The maximum number of states of the SUT is thus given by:

$$maxNumSUTStates = \left(\sum_{j=0}^{k-1} l^j \right) (nl - n + 1) + n,$$

where $n = |\mathsf{A}|$ with A a prefix-closed set $\mathsf{A} \subseteq \mathtt{A}^*$, and $l = |A|$. This is the same upper bound that Vaandrager et al. [2024] gives for the Mealy machine equivalent of $\mathcal{U}_k^{\mathsf{A}}$.

We will now discuss the various sub-procedures used by Algorithm 9.

5.8.3 Making an Observation Tree State Enabled Explored

Algorithm 13 describes our procedure for making a given observation tree state enabled explored. This procedure relies on the symbolic output ($\mathbf{OQ}^{\mathcal{M}}$) and waiting ($\mathbf{WQ}^{\mathcal{M}}$) queries that were introduced in Bruyère et al. [2024]. These queries are defined as follows [Bruyère et al., 2024]:

Definition 5.8.7 (Symbolic queries). Let $\mathcal{M}$ be an MMT. Then we can use the following two symbolic queries:

- $\mathbf{OQ}^{\mathcal{M}}(\mathtt{w})$, with $\mathtt{w}$ a symbolic word such that $q_{\mathcal{I}} \xrightarrow{\mathtt{w}} \in runs(\mathcal{M})$, returns the outputs of $q_{\mathcal{I}} \xrightarrow{\mathtt{w}}$.
- $\mathbf{WQ}^{\mathcal{M}}(\mathtt{w})$, with $\mathtt{w}$ a symbolic word such that $q_{\mathcal{I}} \xrightarrow{i_1} \dots \xrightarrow{i_n} q_n \in runs(\mathcal{M})$ with $\overline{i_1 \dots i_n} = \mathtt{w}$, returns the set of all pairs (j, c) such that $q_{j-1} \xrightarrow{i_j \dots i_n \ \mathsf{to}[x]}$ is x-spanning.

Bruyère et al. explain how these symbolic queries can be implemented via concrete, timed input word-based output queries in appendix E of Bruyère et al. [2024]. When running these timed input words on the SUT, it is possible for timeouts to occur that are unexpected based on the limited information in $\mathcal{T}$. To account for this, Bruyère et al. construct the timed input words such that for any timeout that they observe, they can determine which observation tree transition last started the timer, and with what constant. With this information, they can identify the timer for which they observed the timeout. The occurrence of race conditions could prevent the correct identification of newly discovered timers. Bruyère et al. therefore require the SUT to be race-avoiding, and they compose the timed input words in a way that is guaranteed to prevent races between any known timers. Whenever an **unexpected timeout** occurs, they identify the timer, the transition that last (re)started it and the constant to which it was set; they add the corresponding spanning run to the observation tree. They then repeat the symbolic query on this newly extended observation tree. A timed run for a symbolic word with n transitions can have at most n distinct unexpected timeouts, since each of its transitions can (re)start at most one timer. Both of the symbolic queries can therefore always be completed with a finite number of timed runs.

We cannot simply use the approach from Bruyère et al. [2024] to add any unexpected timeouts to our observation tree $\mathcal{T}$, since we would also have to check whether there is a conflict between the specification and the SUT's behavior for the spanning induced by these unexpected timeouts. To account for this, a simple solution would be make a copy $\mathcal{T}_u$ of the observation tree $\mathcal{T}$ before we evaluate the first timed input word, to only compute the timed input words based on $\mathcal{T}_u$, and to only add any unexpected timeouts to $\mathcal{T}_u$. This way, Bruyère et al. [2024]'s method for dealing with unexpected timeouts still functions. We can discard $\mathcal{T}_u$ when we are done with the symbolic query, but it would be more efficient to reuse the same observation tree copy $\mathcal{T}_u$ for all symbolic queries, and to add any information that is added to $\mathcal{T}$ during the testing process to $\mathcal{T}_u$. This way, we retain any unexpected timeouts that are learnt during the testing process, so that they never need to be rediscovered.

Either way, we can can effectively ignore the existence of unexpected timeouts when it comes to the behavior and correctness of our testing method.

The use of these symbolic output and waiting queries is the most direct way in which our testing procedure touches on the timed MMT semantics. The use of the timed semantics is in all cases abstracted behind the use of either of these two queries.

Algorithm 13 starts by performing the waiting query $\mathbf{WQ}^{\mathcal{M}}(\overline{\mathsf{access}(q)})$. For each index-constant pair (j, c) returned by the waiting query, the procedure immediately terminates with a counterexample in case either $\mathcal{S}$

doesn't (re)set a timer in its corresponding transition, or $\mathcal{S}$ (re)sets a timer to a constant other than c. The procedure then retrieves the timer x'_j that was started in index j of $\mathsf{access}(q)$. It adds a timeout action for x'_j from $\mathcal{T}$ in case one doesn't exist yet. To this end, the procedure first performs a symbolic output query to determine whether the output for the new transition is different from the one that the specification has for the corresponding transition. In that case, the procedure terminates with a counterexample. Otherwise, it adds the transition for x'_j to $\mathcal{T}$.
The procedure also adds the timer update in $\mathcal{T}$ if this is needed. It next marks x'_j as active in all states covered by the new x'_j-spanning.
The procedure finally records q as being enabled explored in case $\mathcal{M}$'s run for $\overline{\mathsf{access}(q)}$ and $\mathcal{S}$'s run for $\overline{\mathsf{access}(q)}$ (re)start timers at exactly the same indices. Otherwise, it concludes that there is conflict between the timer updates of the specification and the SUT. It then returns a counterexample that would reveal this conflict if it is used from both the specification, and the SUT.

Lemma 5.8.2. Calling $\mathit{makeEnabledExplored}^{\mathcal{S}}(q)$ for a state $q \in Q^{\mathcal{T}}$ makes q enabled explored and finds all timeouts that correpond to timeouts of the SUT after $\mathit{makeEnabledExplored}^{\mathcal{S}}$ terminates, if q wasn't enabled explored already. The procedure yields a counterexample in case it finds a conflict between the specification and the SUT. Otherwise, once the procedure is done, $q \in \mathcal{E}$.

The proof of Lemma 5.8.2 can be found in Appendix C.9.1.

Algorithm 13: Sub-procedure for making an observation tree state enabled explored

Procedure *makeEnabledExplored*$^{\mathcal{S}}(q \in Q^{\mathcal{T}})$**:**
  if $q = q_{\mathcal{I}}^{\mathcal{T}}$ **then return** `yes`;
  if $q \in \mathcal{E}$ **then**
    return `yes`;
  end
  $s \leftarrow \delta^{\mathcal{S}^*}(s_{\mathcal{I}}^{\mathcal{S}}, \mathsf{access}(q))$;
  $w \leftarrow \mathbf{WQ}^{\mathcal{M}}(\overline{\mathsf{access}(q)})$;
  forall $(j, c) \in w$ **do**
    `// Check for inconsistencies with the specification's corresponding timer update`
    $u_j^{\mathcal{S}} \leftarrow$ the timer update at index j along $\mathcal{S}$'s run for $\overline{\mathsf{access}(q)}$;
    if $u_j^{\mathcal{S}} = \bot \vee \pi_2(u_j^{\mathcal{S}}) \neq c$ **then return** $\overline{\mathsf{access}(q)}\ \mathsf{to}[j]$;
    `// Retrieve` $\mathcal{T}$`'s corresponding action, timer, and source state`
    $q_{j-1} \leftarrow$ the state at index $j-1$ along $\mathcal{T}$'s run for $\mathsf{access}(q)$;
    $i_j \leftarrow$ the action at index j along $\mathcal{T}$'s run for $\mathsf{access}(q)$;
    $q_j \leftarrow \delta^{\mathcal{T}}(q_{j-1}, i_j)$;
    $x'_j \leftarrow \bot$;
    if $\exists x \colon i_j = \mathsf{to}[x]$ **then** $x'_j \leftarrow x \colon i_j = \mathsf{to}[x]$;
    else $x'_j \leftarrow x_{q_j}$;
    `// Extend` $\mathcal{T}$ `with a timeout transition if needed`
    if $\delta^{\mathcal{T}}(q, \mathsf{to}[x'_j])\uparrow$ **then**
      $o \leftarrow$ the final element of $\mathbf{OQ}^{\mathcal{M}}(\overline{\mathsf{access}(q)}\ \mathsf{to}[j])$;
      if $o \neq \lambda^{\mathcal{S}^*}(s, \mathsf{to}[\pi_1(u_j^{\mathcal{S}})])$ **then return** $\overline{\mathsf{access}(q)}\ \mathsf{to}[j]$;
      $q' \leftarrow$ a fresh MMT state;
      $Q^{\mathcal{T}} \leftarrow Q^{\mathcal{T}} \cup \{q'\}$;
      $X^{\mathcal{T}} \leftarrow X^{\mathcal{T}} \cup \{x_{q'}\}$;
      $O^{\mathcal{T}} \leftarrow O^{\mathcal{T}} \cup \{o\}$;
      $\delta^{\mathcal{T}}(q, \mathsf{to}[x'_j]) \leftarrow q'$;
      $\lambda^{\mathcal{T}}(q, \mathsf{to}[x'_j]) \leftarrow o$;
    end
    `// Record the timer update in` $\mathcal{T}$ `if needed`
    if *there is no timer update at index* j *along* $\mathcal{T}$*'s run for* $\mathsf{access}(q)$ **then**
      $\tau^{\mathcal{T}}(q_{j-1}, i_j) \leftarrow (x'_j, c)$;
    end
    Mark x'_j as active in all states along the x'_j spanning run between q_{j-1} and q.
  end
  `// Determine whether the specification and SUT states have timeouts for corresponding timers`
  $e^q \leftarrow \{\pi_1((j, c)) \mid \forall (j, c) \in w\}$;
  $e^s \leftarrow \{lastStartedAt^{\mathcal{S}}_{s_{\mathcal{I}}^{\mathcal{S}} \xrightarrow{\overline{\mathsf{access}(q)}} s}(x) \mid \forall x \in X^{\mathcal{S}} \colon \delta^{\mathcal{S}}(s, \mathsf{to}[x])\downarrow\}$;
  if $e^q \neq e^s$ **then**
    $j \leftarrow$ an arbitrary element of $(e^q \cup e^s) \setminus (e^q \cap e^s)$;
    return $\overline{\mathsf{access}(q)}\ \mathsf{to}[j]$;
  end
  `// Mark the observation tree state as enabled explored`
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{q\}$;
  return `yes`;

5.8.4 Extending the Observation Tree With a Single Transition

Algorithm 14 describes our method for extending the observation tree with a new transition. The procedure takes an observation tree state q and an action i that is to be added from q. The procedure immediately terminates if $\mathcal{T}$ already as an i-transition from q, but not before using Algorithm 13 to ensure that $\delta^{\mathcal{T}}(q,i) \in \mathcal{E}$. Otherwise, if i is a timeout, then it uses Algorithm 13 to make q enabled explored, thereby ensuring that afterwards, q will indeed have an outgoing transition for timeout i. The procedure also makes the state $\delta^{\mathcal{T}}(q,i)$ enabled explored, granted that there was no conflict.
The final case is the one in which i is an input action for which $\mathcal{T}$ doesn't yet have an outgoing transition from $\mathcal{T}$. The procedure then uses an output query to find the SUT's output for this transition. It adds the transition to $\mathcal{T}$ in case there is no conflict between the specification's and the SUT's counterpart for this transition. Finally, the procedure uses Algorithm 13 to make the new state $\delta^{\mathcal{T}}(q,i)$ enabled explored.
The procedure would return a counterexample in case any conflicts between the specification and the SUT are found during its operation.

Algorithm 14: Sub-procedure for extending the observation tree with a single transition

Procedure $addTransition^{\mathcal{S}}_{\mathcal{M}}(q \in Q^{\mathcal{T}}, i \in A^{\mathcal{T}})$**:**
  if $\delta^{\mathcal{T}}(q,i)\downarrow$ **then**
    $q' = \delta^{\mathcal{T}}(q,i)\downarrow$;
    $c \leftarrow makeEnabledExplored^{\mathcal{S}}(q')$;
    if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
    return `no`;
  end
  if $\exists x\colon i = \mathsf{to}[x]$ **then**
    $c \leftarrow makeEnabledExplored^{\mathcal{S}}(q)$;
    if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
    $x \leftarrow x\colon i = \mathsf{to}[x]$;
    $q' \leftarrow \delta^{\mathcal{T}}(q, \mathsf{to}[x])$;
    $c' \leftarrow makeEnabledExplored^{\mathcal{S}}(q')$;
    if $c' \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c';
    return q';
  end
  `// Check the transition's output symbol`
  $s \leftarrow \delta^{\mathcal{S}^*}(s^{\mathcal{S}}_{\mathcal{I}}, \mathsf{access}(q))$;
  $i^{\mathcal{S}} \leftarrow i$;
  $o^{\mathcal{S}} \leftarrow \lambda^{\mathcal{S}}(s, i^{\mathcal{S}})$;
  $o \leftarrow$ the final element of $\mathbf{OQ}^{\mathcal{M}}(\overline{\mathsf{access}(q)\ i})$;
  if $o \neq o^{\mathcal{S}}$ **then return** $\overline{\mathsf{access}(q)\ i}$;
  `// Add the transition to` $\mathcal{T}$
  $q' \leftarrow$ a fresh MMT state;
  $Q^{\mathcal{T}} \leftarrow Q^{\mathcal{T}} \cup \{q'\}$;
  $X^{\mathcal{T}} \leftarrow X^{\mathcal{T}} \cup \{x_{q'}\}$;
  $O^{\mathcal{T}} \leftarrow O^{\mathcal{T}} \cup \{o\}$;
  $\delta^{\mathcal{T}}(q,i) \leftarrow q'$;
  $\lambda^{\mathcal{T}}(q,i) \leftarrow o$;
  $c \leftarrow makeEnabledExplored^{\mathcal{S}}(q')$;
  if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
  return `yes`;

Lemma 5.8.3. Calling $addTransition^{\mathcal{S}}_{\mathcal{M}}(q, i)$ ensures that $\delta^{\mathcal{T}}(q, i)\downarrow$ by the time that $addTransition^{\mathcal{S}}_{\mathcal{M}}$ terminates, granted that there is no conflict between the specification and the SUT. Otherwise, it would return a counterexample.

Proof. We already argued that in all cases, $\delta^{\mathcal{T}}(q, i)\downarrow$ either already holds before $addTransition^{\mathcal{S}}_{\mathcal{M}}(q, i)$ is called, or it will hold afterwards, granted that there was no conflict. If there was, then the procedure would return a counterexample. □

Lemma 5.8.4. Calling $addTransition^{\mathcal{S}}_{\mathcal{M}}(q, i)$ ensures that $\delta^{\mathcal{T}}(q, i) \in \mathcal{E}$ by the time that $addTransition^{\mathcal{S}}_{\mathcal{M}}$ terminates, granted that there is no conflict between the specification and the SUT. Otherwise, it would return a counterexample.

Proof. In all cases, $addTransition^{\mathcal{S}}_{\mathcal{M}}$ calls $makeEnabledExplored^{\mathcal{S}}$ on $\delta^{\mathcal{T}}(q, i)$. Therefore, by Lemma 5.8.2, $\delta^{\mathcal{T}}(q, i) \in \mathcal{E}$ once $makeEnabledExplored^{\mathcal{S}}$ terminates. Since the call to $makeEnabledExplored^{\mathcal{S}}$ terminates before $addTransition^{\mathcal{S}}_{\mathcal{M}}$ terminates, $\delta^{\mathcal{T}}(q, i) \in \mathcal{E}$ will hold by the time that $addTransition^{\mathcal{S}}_{\mathcal{M}}$ terminates, as required. $addTransition^{\mathcal{S}}_{\mathcal{M}}$ would return a counterexample if any conflict between the specification and the SUT was found during its operation. □

5.8.5 Extending the Observation Tree With a Sequence of Transitions

Algorithm 15 shows a sub-procedure of Algorithm 9 which extends the observation tree with all transitions induced by a given sequence of actions from the specification.

For each action, the procedure first determines which action from the observation tree it would correspond to. It then adds the observation tree action, after which it processes the next specification action. The procedure relies on the fact that it starts analyzing the given action sequence from the perspective of the inital state, in which no timers are ever active. For input actions, it simply adds transitions for the inputs it is given to the observation tree state. For timeout actions, it first determines which observation tree timer the specified specification timer corresponds to. To do so, the sub-procedure uses a map $l^{\mathcal{S}} \colon X^{\mathcal{S}} \to X^{\mathcal{T}}$ that yields for every timer of the specification that was encountered so far the last observation tree timer that was started at the same index along the observation tree's run. If the next action $i^{\mathcal{T}}$ that the sub-procedure should add to the observation tree is an input action, then Algorithm 15 simply uses Algorithm 14 to add the transition for $i^{\mathcal{T}}$ if it doesn't already exist. If $i^{\mathcal{T}}$ is a timeout action, then we can't use Algorithm 14 to add the transition, because we don't yet know whether the SUT actually has a corresponding timeout. If not, then the fact that we would want to add this timeout to the observation tree would indicate a conflict between the observable behavior of the specification and the SUT that would imply that the two are not symbolically equivalent. Algorithm 15 therefore uses the sub-procedure Algorithm 13 to both add all timeouts that could exist from the observation tree state, but also to find out whether either the specification or the SUT has timeouts from their current states for which the other has no counterparts. If so, then Algorithm 13 returns a counterexample, and then Algorithm 15 does so as well.
The procedure stops in case it discovers a conflict in the behavior of the specification and the SUT. The procedure then terminates, returning a counterexample.

Lemma 5.8.5. Calling sub-procedure:

$$addTransitionsFromSpecSeq(\sigma)$$

ensures that $\mathcal{T}$ will have a run for its counterpart of σ if the specification and SUT have no coflicts along this run. Otherwise, $addTransitionsFromSpecSeq(\sigma)$ would return a counterexample. Otherwise, it would return a counterexample.

Proof. $addTransitionsFromSpecSeq(\sigma)$ calls $addTransition^{\mathcal{S}}_{\mathcal{M}}$ on all steps along $\mathcal{T}$'s run for its counterpart of σ. Therefore, by Lemma 5.8.3, $\mathcal{T}$ has a run for $\mathcal{T}$'s counterpart to σ once $addTransitionsFromSpecSeq(\sigma)$ terminates, granted that there were no conflicts. □

Lemma 5.8.6. Calling $addTransitionsFromSpecSeq(\sigma)$ ensures that all states that are traversed for σ are enabled explored after the sub-procedure terminates, granted that not conflicts between the specification and the SUT were found during its operation.

Algorithm 15: Sub-procedure for extending the observation tree with multiple transitions given by a specification action sequence that starts in the specification's initial state

Procedure $addTransitionsFromSpecSeq^{\mathcal{S}}_{\mathcal{M}}(\sigma \in (A^{\mathcal{S}})^*)$**:**
- $s \leftarrow s^{\mathcal{S}}_{\mathcal{I}}$;
- $q \leftarrow q^{\mathcal{T}}_{\mathcal{I}}$;
- $l^{\mathcal{S}} \leftarrow \emptyset$;
- $j \leftarrow 1$;
- $x^{\mathcal{T}}_c \leftarrow \bot$;
- **while** $\sigma \neq \epsilon$ **do**
  - `// Retrieve the actions`
  - $i^{\mathcal{S}} \leftarrow \mathsf{head}(\sigma)$;
  - $i^{\mathcal{T}} \leftarrow i^{\mathcal{S}}$;
  - **if** $\exists x\colon i^{\mathcal{S}} = \mathsf{to}[x]$ **then**
    - `// Ensure that the corresponding timer is enabled in` q
    - $c \leftarrow makeEnabledExplored^{\mathcal{S}}(q)$;
    - **if** $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
    - $x^{\mathcal{S}} \leftarrow x\colon i^{\mathcal{S}} = \mathsf{to}[x]$;
    - $x^{\mathcal{T}} \leftarrow l^{\mathcal{S}}(x^{\mathcal{S}})$;
    - $i^{\mathcal{T}} \leftarrow \mathsf{to}[x^{\mathcal{T}}]$;
  - **end**
  - `// Add the transition`
  - $c \leftarrow addTransition^{\mathcal{S}}_{\mathcal{M}}(q, i^{\mathcal{T}})$;
  - **if** $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
  - `// Update timer mapping` $l^{\mathcal{S}}$
  - $\mathfrak{r} \leftarrow \tau^{\mathcal{S}}(s, i^{\mathcal{S}})$;
  - **forall** $x' \in \mathsf{dom}(\mathfrak{r})$ **do**
    - **if** $\mathfrak{r}(x') \in \mathbb{N}^{>0}$ **then**
      - **if** $x^{\mathcal{T}}_c = \bot$ **then return** $\overline{\mathsf{access}(q)}\ \mathsf{to}[j]$;
      - $l^{\mathcal{S}}(x') \leftarrow x^{\mathcal{T}}_c$;
    - **end**
    - **else**
      - $x \leftarrow \mathfrak{r}(x')$;
      - $l^{\mathcal{S}}(x') \leftarrow l^{\mathcal{S}}(x)$;
    - **end**
  - **end**
  - `// Prepare for the next iteration`
  - $s \leftarrow \delta^{\mathcal{S}}(s, i^{\mathcal{S}})$;
  - $q \leftarrow \delta^{\mathcal{T}}(q, i^{\mathcal{T}})$;
  - $\sigma \leftarrow \mathsf{tail}(\sigma)$;
  - **if** $\tau^{\mathcal{T}}(q, i^{\mathcal{T}}) \neq \bot$ **then**
    - $x^{\mathcal{T}}_c \leftarrow \pi_1(\tau^{\mathcal{T}}(q, i^{\mathcal{T}}))$;
  - **end**
  - **else**
    - $x^{\mathcal{T}}_c \leftarrow \bot$;
  - **end**
  - $j \leftarrow j + 1$;
- **end**
- **return** `yes`;

Proof. *addTransitionsFromSpecSeq*(σ) calls $addTransition^{\mathcal{S}}_{\mathcal{M}}$ on all steps along $\mathcal{T}$'s run for its counterpart of σ. Therefore, by Lemma 5.8.4, all states along $\mathcal{T}$'s run for $\mathcal{T}$'s counterpart to σ will be enabled explored once *addTransitionsFromSpecSeq*(σ) terminates, granted that not conflicts between the specification and the SUT were found during its operation. □

5.8.6 Making Observation Tree States Active Explored

Algorithm 16 shows a version of Algorithm 15 that not only adds the transitions from a sequences of specification actions to the observation tree, it also adds them to Algorithm 9's set $\mathcal{A}_p$ of states that should be made active explored. The sub-procedure starts by using Algorithm 15. If there were no conflicts between

Algorithm 16: Sub-procedure for extending the observation tree with multiple transitions given by a specification action sequence that starts in the specification's initial state. The new states will eventually be made active explored by Algorithm 9

Procedure $addTransitionsFromSpecSeqAndMakeActiveExplored^{\mathcal{S}}_{\mathcal{M}}(\sigma \in (A^{\mathcal{S}})^*)$**:**
  $c \leftarrow addTransitionsFromSpecSeq^{\mathcal{S}}_{\mathcal{M}}(\sigma)$;
  if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
  $T \leftarrow \{q_1, \ldots, q_n\} \colon q_I^{\mathcal{T}} \xrightarrow{\mathsf{head}(\sigma)} q_1 \xrightarrow{\mathsf{tail}(\sigma)} q_n \in runs(\mathcal{T})$;
  $A \leftarrow \{q \in Q^{\mathcal{T}} \mid \forall q \in T \colon q \notin \mathcal{A}\}$;
  $\mathcal{A}_p \leftarrow \mathcal{A}_p \cup A$;
  return yes;

the behavior of the specification and the SUT, then it finishes by taking all states along the observation tree run that corresponds to the specification action sequence, and adding each of them that is not in Algorithm 9's set of active explored states $\mathcal{A}$ to $\mathcal{A}_p$.

Lemma 5.8.7. Let $q \in Q^{\mathcal{T}}$. If $q \in \mathcal{A}$, then no more states will ever be made active in q.

Proof. We know from the definition of $\mathcal{A}$ that $q \in \mathcal{A}$ implies that $q \in \mathcal{A}^{\mathcal{T}}_{\mathcal{S}}$ and $q \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$. This in turn implies that $|\mathcal{X}^{\mathcal{T}}(q)| = |\mathcal{X}^{\mathcal{S}}(f_s(q))|$ and $|\mathcal{X}^{\mathcal{T}}(q)| = |\mathcal{X}^{\mathcal{M}}(g_s(q))|$.
Therefore, (FGS1) and (FGS2) tells us that for all $x \in \mathcal{X}^{\mathcal{S}}(f_s(q))$, there is exactly one $y \in \mathcal{X}^{\mathcal{T}}(q)$ such that $f_t(q, y) = x$.
It similarly tells us that, per (FMS1) and (FMS2), all $x' \in \mathcal{X}^{\mathcal{M}}(g_s(q))$ have exactly one $y \in \mathcal{X}^{\mathcal{T}}(q)$ such that $g_t(y) = x'$. This means that there will never be a reason to make any additional states active in q, as required. □

The next lemma relies on the notion of ancestors of observation tree states.

Definition 5.8.8. Let $\mathcal{T}$ be an observation tree. The **ancestors** of a state $q \in Q^{\mathcal{T}}$ are the states of $Q^{\mathcal{T}}$ along $\mathcal{T}$'s unique path from its initial state to q. Formally:

$$ancestors(q) \coloneqq \{q_a \in Q^{\mathcal{T}} \mid \exists \sigma \in A^{\geq 1} \wedge q_a \xrightarrow{\sigma} q \in runs(\mathcal{T})\}.$$

Lemma 5.8.8. For all states in $\mathcal{A}_p$, all ancestors are either in $\mathcal{A}_p$, or in $\mathcal{A}$.

Proof. This property follows from the way in which we add states to $\mathcal{A}_p$: Only the sub-procedure:

addTransitionsFromSpecSeqAndMakeActiveExplored

adds states to $\mathcal{A}_p$. When $\mathcal{A}_p$ adds a state q to $\mathcal{A}_p$, it also adds all of q's ancestors to $\mathcal{A}_p$. We thus know that for all states in $\mathcal{A}_p$, all ancestors are either in $\mathcal{A}_p$, or in $\mathcal{A}$. □

Lemma 5.8.9. Let $q, q \in Q^{\mathcal{T}}$, and let $i \in A^{\mathcal{T}}$. If $q \in \mathcal{A} \wedge (\tau^{\mathcal{T}}(q, i) \neq \bot \wedge (\mathcal{X}^{\mathcal{T}}(q) \subseteq \mathcal{X}^{\mathcal{T}}(q')))$, then $q' \in \mathcal{A}$.

The proof of Lemma 5.8.9 can be found in Appendix C.9.2.

Lemma 5.8.10. Let $\mathcal{U}$ be an MMT fault domain for the SUT, and let $m = \max_{\mathcal{M} \in \mathcal{U}} |Q^{\mathcal{M}}|$. Suppose that $m \geq |Q^{\mathcal{S}}|$, and that $\mathcal{M} \in \mathcal{U}$. Let $q \in Q^{\mathcal{T}}$, and let q_{-1} be q's parent state. If $q_{-1} \in \mathcal{A}$, and:

$$areAllStatesInNStepsPresentAndEnabledExplored(q, m),$$

is called, then $q \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ and $q \in \mathcal{A}^{\mathcal{T}}_{\mathcal{S}}$ will hold upon termination.

The proof of Lemma 5.8.10 can be found in Appendix C.9.3.
We use this lemma to prove that, for our fault domain $\mathcal{U}^{\overline{C}}_k \cup \mathcal{U}^{\overline{C}}$:

Corollary 5.8.1. Let $k \in \mathbb{N}^{>0}$, and let C be a minimal and prefix-closed state cover for $\mathcal{S}$. Suppose that $\mathcal{M} \in \mathcal{U}^{\overline{C}}_k \cup \mathcal{U}^{\overline{C}}$. Let $q \in Q^{\mathcal{T}}$, and let q_{-1} be q's parent state. If $q_{-1} \in \mathcal{A}$, and:

$$areAllStatesInNStepsPresentAndEnabledExplored(q, maxNumSUTStates),$$

is called, then $q \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ and $q \in \mathcal{A}^{\mathcal{T}}_{\mathcal{S}}$ will hold upon termination.

Proof. The property follows directly from Lemma 5.8.10, since:

$$\max_{\mathcal{M} \in \mathcal{U}^{\overline{C}}_k \cup \mathcal{U}^{\overline{C}}} |Q^{\mathcal{M}}| = maxNumSUTStates \geq |C| = |Q^{\mathcal{S}}|,$$

which implies that $\max_{\mathcal{M} \in \mathcal{U}^{\overline{C}}_k \cup \mathcal{U}^{\overline{C}}} |Q^{\mathcal{M}}| \geq |Q^{\mathcal{S}}|$, as required. □

Lemma 5.8.11. All states in $\mathcal{A}_p \cup B \cup F^{\leq k}$ are eventually added to $\mathcal{A}$, granted that no conflicts between the specification and the SUT arise before then.

The proof of Lemma 5.8.11 can be found in Appendix C.9.4.

Lemma 5.8.12. All states that are added to $\mathcal{A}_p$ are eventually made both enabled and active explored, or there will be a counterexample.

Proof. We only add states to $\mathcal{A}_p$ in the sub-procedure:

$$addTransitionsFromSpecSeqAndMakeActiveExplored^{\mathcal{S}}_{\mathcal{M}}.$$

This procedure uses $addTransitionsFromSpecSeq^{\mathcal{S}}_{\mathcal{M}}$ which ensures by Lemma 5.8.6 that all of these states will be enabled explored, granted that there were no conflicts.
Lemma 5.8.11 tells us that all of these states will eventually be enabled explored, granted that no conflicts between the specification and the SUT will be found. □

Lemma 5.8.13. Calling sub-procedure:

$$addTransitionsFromSpecSeqAndMakeActiveExplored(\sigma)$$

ensures that σ will be a run in $\mathcal{T}$, or there will be a counterexample.

Proof. Sub-procedure addTransitionsFromSpecSeqAndMakeActiveExplored(σ) calls:

$$addTransitionsFromSpecSeq(\sigma).$$

Lemma 5.8.5 now tells us that therefore, σ will have a run in $\mathcal{T}$. □

Lemma 5.8.14. All states traversed by sequences passed to:

$$addTransitionsFromSpecSeqAndMakeActiveExplored^{\mathcal{S}}_{\mathcal{M}}$$

are eventually both enabled and active explored, or there is a counterexample.

Proof. Lemma 5.8.12 tells us that all states of $\mathcal{T}$ that are traversed when addTransitionsFromSpecSeqAndMakeActiveExplored is called on σ are enabled explored once the sub-procedure terminates.
Sub-procedure addTransitionsFromSpecSeqAndMakeActiveExplored adds all states that are traversed by σ from $\mathcal{T}$'s initial state to $\mathcal{A}_p$. Lemma 5.8.12 now tells us that all states traversed by the sequence σ passed to addTransitionsFromSpecSeqAndMakeActiveExplored are eventually both enabled and active explored, granted that no conflict is found. □

5.8.7 Making Two Observation Tree States Apart

We show our sub-procedure for making two observation tree states apart in Algorithm 17.

Algorithm 17: Sub-procedure for making two observation tree states apart when their specification counterparts are apart

Procedure $makeObsTreeStatesApart^{\mathcal{S}}(q, q' \in Q^{\mathcal{T}})$**:**
  if $q \# q'$ **then**
    return;
  end
  $s \leftarrow \delta^{\mathcal{S}^*}(s_{\mathcal{I}}^{\mathcal{S}}, \overline{\mathsf{access}(q)})$;
  $s' \leftarrow \delta^{\mathcal{S}^*}(s_{\mathcal{I}}^{\mathcal{S}}, \overline{\mathsf{access}(q')})$;
  $\sigma_a \leftarrow \rho \in C\colon \delta^{\mathcal{S}^*}(s_{\mathcal{I}}^{\mathcal{S}}, \rho) = s$;
  $\sigma'_a \leftarrow \rho \in C\colon \delta^{\mathcal{S}^*}(s_{\mathcal{I}}^{\mathcal{S}}, \rho) = s'$;
  forall *maximal matchings* $m\colon s \leftrightarrow s'$ **do**
    $\sigma_w \leftarrow$ a minimal-length witness of $s \#^m s'$;
    $\sigma'_w \leftarrow$ the action sequence for $read^m_{s \xrightarrow{\sigma_w}}(s')$;
    $c \leftarrow addTransitionsFromSpecSeqAndMakeActiveExplored^{\mathcal{S}}_{\mathcal{T}}(\sigma_a \cdot \sigma_w)$;
    if $c \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c;
    $c' \leftarrow addTransitionsFromSpecSeqAndMakeActiveExplored^{\mathcal{S}}_{\mathcal{T}}(\sigma'_a \cdot \sigma'_w)$;
    if $c' \in (I \cup TO(\mathbb{N}^{>0}))^*$ **then return** c';
  end
return `yes`;

Let $\mathcal{S}$ be an s-learnable gMMT, and let $\mathcal{T}$ be an observation tree. Let $\langle f_s, f_t, f_u \rangle\colon \mathcal{T} \to \mathcal{S}$ be a functional gMMT simulation. Let $q, q' \in Q^{\mathcal{T}}$ be the two states of $\mathcal{T}$ that we want to make apart. Let $s = f_s(q)$, and let $s' = f_s(q')$. We require that $s \# s'$.

Algorithm 17 first checks whether $q \# q'$ already holds. If so, then it simply terminates right away. Otherwise, it proceeds by finding the specification states s and s'. For every maximal matching $m\colon s \leftrightarrow s'$, it finds a witness σ_w of $s \#^m s'$. It then reads $\sigma'_w = read^m_{s \xrightarrow{\sigma_w}}(s')$, and it uses:

$$addTransitionsFromSpecSeqAndMakeActiveExplored^{\mathcal{S}}_{\mathcal{T}}$$

to add the corresponding input sequences from q and q', as well as to ensure that every observation tree state encountered along these sequences are eventually both enabled and active explored. Let $\pi \in runs(\mathcal{T})$ and $\pi' \in runs(\mathcal{T})$ be the runs that this adds to $\mathcal{T}$. Proposition 5.4.1 tells us that once the states along π and π' are all enabled and active explored, all timer updates along $s \xrightarrow{\sigma_w} \in runs(\mathcal{S})$ and $s' \xrightarrow{\sigma'_w} \in runs(\mathcal{S})$ will be represented in π and π'. The actions along π will then function as a witness for the apartness between q and q' under a certain matching $m'\colon q \leftrightarrow q'$ for the same reason for which σ_w functions as a witness for the apartness between s and s' under m. We prove that when we use minimum-length witnesses for σ_w, doing this for all maximal matchings $m\colon s \leftrightarrow s'$ will ensure that once the states along π and π' are enabled explored, $q \# q'$.

Lemma 5.8.15. Let $\mathcal{T}$ be an observation tree, let $\mathcal{S}$ be an s-learnable gMMT, and let $\langle f_s, f_t, f_u \rangle\colon \mathcal{T} \to \mathcal{S}$ be a functional gMMT simulation. Let C be a prefix-closed state cover for $\mathcal{S}$. Let $q_0, q'_0 \in Q^{\mathcal{T}}$. Let $s = f_s(q_0)$ and $s' = f_s(q'_0)$. Let $\sigma_a = c \in C\colon \delta^{\mathcal{S}^*}(s_{\mathcal{I}}^{\mathcal{S}}, c) = s$, and let $\sigma'_a = c \in C\colon \delta^{\mathcal{S}^*}(s_{\mathcal{I}}^{\mathcal{S}}, c) = s'$. Suppose that $s \# s'$ and $\neg(q_0 \# q'_0)$. If for all maximal matchings $m\colon s \leftrightarrow s'$ there is at least one run $\rho = s \xrightarrow{\sigma_w} \in runs(\mathcal{S})$ and $\rho' = read^m_{s \xrightarrow{\sigma_w}}(s') = s' \xrightarrow{\sigma'_w} \in runs(\mathcal{S})$ such that $\sigma'_w = i'_1 \dots i'_n \in (A^{\mathcal{S}})^*$ and $\sigma_w = i_1 \dots i_n \in (A^{\mathcal{S}})^*$ is a minimum-length witness of $s \#^m s'$, then calling:

$$addTransitionsFromSpecSeqAndMakeActiveExplored^{\mathcal{S}}_{\mathcal{M}}(\sigma_a \cdot \sigma_w)$$

and:

$$addTransitionsFromSpecSeqAndMakeActiveExplored^{\mathcal{S}}_{\mathcal{M}}(\sigma'_a \cdot \sigma'_w)$$

ensures that eventually, $q_0 \# q'_0$, granted that no conflict between the specification and the SUT is found before then.

The proof of Lemma 5.8.15 can be found in Appendix C.8.

We prove the correctness of Algorithm 17 as a corollary of Lemma 5.8.15:

Corollary 5.8.2. Let $\mathcal{T}$ be an observation tree MMT, let $\mathcal{S}$ be an s-learnable gMMT, and let $\langle f_s, f_t, f_u \rangle \colon \mathcal{T} \to \mathcal{S}$ be a functional gMMT simulation. Let $q, q' \in Q^{\mathcal{T}}$.

If $f_s(q) \# f_s(q')$, then calling $makeObsTreeStatesApart^{\mathcal{S}}(q, q')$ either returns a counterexample symbolic word because it finds a conflict between the specification and the SUT, or it ensures that eventually, $q \# q'$ will hold, granted that no conflict between the specification and the SUT is found before then.

Proof. The procedure of Algorithm 17 immediately terminates if $q \# q'$ already holds at the beginning, since nothing would need to be done. Otherwise, it makes q and q' apart. It then starts by taking $s = f_s(q)$ and $s' = f_s(q')$. For all maximal matchings $m \colon s \leftrightarrow s'$, the procedure takes a minimal-length witness σ_w of $s \#^m s'$. This implies that $\rho = s \xrightarrow{\sigma_w} \in runs(\mathcal{S})$, and that $\rho' = read^m_{s \xrightarrow{\sigma_w}}(s') = s' \xrightarrow{\sigma'_w} \in runs(\mathcal{S})$.

Let C be a prefix-closed state cover for $\mathcal{S}$. Let $q_0, q'_0 \in Q^{\mathcal{T}}$. Let $\sigma_a = c \in C \colon \delta^{\mathcal{S}^*}(s^{\mathcal{S}}_{\mathcal{I}}, c) = s$, and let $\sigma'_a = c \in C \colon \delta^{\mathcal{S}^*}(s^{\mathcal{S}}_{\mathcal{I}}, c) = s'$.
The procedure runs:

$$addTransitionsFromSpecSeqAndMakeActiveExplored^{\mathcal{S}}_{\mathcal{M}}(\sigma_a \cdot \sigma_w)$$

and:

$$addTransitionsFromSpecSeqAndMakeActiveExplored^{\mathcal{S}}_{\mathcal{M}}(\sigma'_a \cdot \sigma'_w)$$

to try to extend $\mathcal{T}$ with the runs π and π'. If it succeeds, then Lemma 5.8.15 tells us that $q \# q'$ will eventually hold, granted that no conflict between the specification and the SUT is found before then. The procedure would fail to add π and π' to $\mathcal{T}$ if, and only if adding these runs to $\mathcal{T}$ reveals a conflict between the specification $\mathcal{S}$ and the SUT. It would then return a counterexample. □

5.8.8 Termination

We can prove that the procedure of Algorithm 9 terminates within a finite number of rule applications:

Lemma 5.8.16. The procedure of Algorithm 9 always terminates within a finite number of rule applications.

The proof of Lemma 5.8.16 can be found in Appendix C.9.5.

5.8.9 k-A-Completeness of the Procedure

As with our method for MM1Ts from Chapter 3, our approach to proving the k-A-completeness of our procedure is inspired by Vaandrager et al. [2024]'s sufficient condition for the k-A-completeness of test suites for Mealy machines. We will use the following theorem to prove that for any natural number k and any minimal and prefix-closed state cover C for the specification, Algorithm 9 is a valid and k-$\overline{C}$-complete conformance testing procedure for MMTs:

Theorem 5.8.1. Let $k \in \mathbb{N}$. Let $\mathcal{S}$ be a minimal, s-learnable, t-observable gMMT, and let $C \subseteq A^{\mathcal{S}}$ be a minimal and prefix-closed state cover for $\mathcal{S}$. Let $\mathcal{M}$ be an s-learnable, t-observable MMT from $\mathcal{U}^{\overline{C}}_k \cup \mathcal{U}^{\overline{C}}$ that has the same set of inputs I as $\mathcal{S}$. Let $\mathcal{T}$ be an observation tree for both $\mathcal{M}$ and $\mathcal{S}$, and let $B, F^0, F^1, \ldots$ be the stratification of $Q^{\mathcal{T}}$ induced by $\overline{C}$. Suppose that B and $F^{<k}$ are complete, the states in B and $F^{\leq k}$ are all identified, $B \cup F^{\leq k} \subseteq \mathcal{A}^{\mathcal{T}}_{\mathcal{S}}$, $B \cup F^{\leq k} \subseteq \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$, and the following condition holds:

$$\forall t' \in F^k, t'' \in F^{<k} \colon \qquad \mathcal{C}(t') = \mathcal{C}(t'') \quad \vee \quad t' \# t''. \tag{5.1}$$

Then $\mathcal{M} \approx_{sym} \mathcal{S}$.

The proof of Theorem 5.8.1 can be found in Appendix C.6.1.

As in the MM1T setting from Chapter 3, Algorithm 9 does not directly guarantee that the condition of Equation (5.1) will hold. Its ExtendCoTransitivity rule instead guarantees that once the procedure is done, the following condition will hold:

$$\forall r \in B, t' \in F^k, t'' \in F^{<k}: \qquad r \# t' \implies r \# t'' \vee t' \# t''. \tag{5.2}$$

We use the following property to prove that Equation (5.2) implies Equation (5.1):

Lemma 5.8.17. Let $\mathcal{S}$ be an s-learnable gMMT, and let $\mathcal{T}$ be an observation tree for $\mathcal{S}$. Let B be the basis of a stratification of $Q^{\mathcal{T}}$. Suppose that $q, q' \in Q^{\mathcal{T}}$ and q is identified. Then:

$$\mathcal{C}(q) = \mathcal{C}(q') \vee q \# q' \iff (\forall r \in B : r \# q \implies r \# q' \vee q \# q').$$

The proof of Lemma 5.8.17 can be found in Appendix C.6.2.

Let k be a natural number, and let A be a prefix-closed state cover of the specification. The validity of Algorithm 9 as a k-A-complete testing procedure for MMTs is a corollary of Theorem 5.8.1:

Corollary 5.8.3. Let $\mathcal{S}$ be a minimal, s-learnable, t-observable gMMT. Let C be a minimal and prefix-closed state cover for $\mathcal{S}$. Let k be a natural number, and let $\mathcal{M}$ be an s-learnable MMT from $\mathcal{U}_k^{\overline{C}} \cup \mathcal{U}^{\overline{C}}$. The procedure of Algorithm 9 returns `yes` iff $\mathcal{M} \approx_{sym} \mathcal{S}$, and it returns a counterexample in the form of an action sequence iff $\mathcal{M} \not\approx_{sym} \mathcal{S}$.

Proof. We know from Lemma 5.8.16 that Algorithm 9 always terminates within a finite number of rule applications. We see on line Line 37 that the algorithm returns `yes` once none of its rules can be applied anymore. When that happens:

- Lemma C.6.8 tells us that f_s is bijective when restricted to B. We thus know that for all distinct $r, r' \in B$, $f_s(r) \neq f_s(r')$. The fact that $\mathcal{S}$ is minimal now implies that $f_s(r) \# f_s(r')$. Thus, since the IdentifyBasisStates rule performs $makeObsTreeStatesApart^{\mathcal{S}}(r, r')$ without this yielding a counterexample, Corollary 5.8.2 tells us that $r \# r'$. Therefore, all distinct basis states are pairwise apart, which means that all basis states are identified.

- Algorithm 1's initial loop extends for each $\mathtt{w} \in \overline{C}$ the observation tree $\mathcal{T}$ with a new state $q = \delta^{\mathcal{T}^*}(\mathtt{w})$ that it then adds to B. It uses the ExtendActiveExplored rule to make every basis state enabled explored, per Lemma 5.8.2. The ExtendFrontiersWithInputs rule adds outgoing transitions for all inputs from I to all basis states. The basis is therefore complete.
  The ExtendFrontiersWithInputs rule uses $addTransition^{\mathcal{S}}_{\mathcal{M}}$ to add from each state $q \in F^{<k}$ an outgoing transition for each input from I. The ExtendActiveExplored rule makes every state that this adds to $\mathcal{T}$ enabled explored, per Lemma 5.8.2. Making observation tree states enabled explored extends the observation tree with new states. Each of these new states that falls within $F^{<k}$ is also made enabled explored through the use of the ExtendActiveExplored rule. We thus know that all states from $F^{<k}$ are complete.

- Let $t \in F^{\leq k}$. By Lemma C.6.9, there is a basis state $r_t \in B$ for which $r_t \in \mathcal{C}(t)$ and $f_s(r_t) = f_s(t)$. Let $r \in B$ be a basis state for which $r \neq r_t$. Lemma C.6.8 tells us that since $r \neq r_t$, $f_s(r) \neq f_s(r_t)$, which then implies that $f_s(r) \neq f_s(t)$. The fact that $\mathcal{S}$ is minimal now implies that $f_s(r) \# f_s(t)$. Thus, since the IdentifyFrontiers rule performs $makeObsTreeStatesApart^{\mathcal{S}}(t, r)$ without this yielding a counterexample, Corollary 5.8.2 tells us that $t \# r$. Therefore, $r \notin \mathcal{C}(t)$. This implies that $\mathcal{C}(t) = \{r_t\}$, which means that t is identified. We thus know that all states from $F^{\leq k}$ are identified.

- by Lemma 5.8.11, $B \cup F^{\leq k} \subseteq \mathcal{A}$. This means that $B \cup F^{\leq k} \subseteq \mathcal{A}^{\mathcal{T}}_{\mathcal{S}}$ and $B \cup F^{\leq k} \subseteq \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$, as required.

- Let $r \in B$, $t' \in F^k$ and $t'' \in F^{<k}$. We show that Equation (5.2) holds. If $r \# t''$ or $t' \# t''$ holds, then the condition from Equation (5.2) already holds as well. So suppose that $\neg(r \# t'')$ and $\neg(t' \# t'')$. We know from the previous item that t'' is identified. Therefore, since $\neg(r \# t'')$, $\mathcal{C}(t'') = \{r\}$. Thus, by Lemma C.6.9, $f_s(t'') = f_s(r)$. By Lemma C.3.2, $r \# t'$ implies that $f_s(r) \neq f_s(t')$. Therefore,

$f_s(t'') \neq f_s(t')$. The fact that $\mathcal{S}$ is minimal now implies that $f_s(t'') \# f_s(t')$. Thus, since the Extend-CoTransitivity rule performs $makeObsTreeStatesApart^{\mathcal{S}}(t', t'')$ without this yielding a counterexample, Corollary 5.8.2 tells us that $t' \# t''$. Equation (5.2) therefore holds. Since Equation (5.2) holds in all cases, Lemma 5.8.17 tells us that the following condition holds in all cases:

$$\forall t' \in F^k, t'' \in F^{<k}: \qquad \mathcal{C}(t') = \mathcal{C}(t'') \quad \vee \quad t' \# t''.$$

Theorem 5.8.1 thus tells us that if Algorithm 9 terminates because none of its rules can be applied anymore, then $\mathcal{M} \approx_{sym} \mathcal{S}$.
The only circumstance under which Algorithm 9 terminates before all eight of its rules are exhausted is if it finds a conflict between the outputs, timer updates, number of active and/or enabled timers and the timers that are enabled between $\mathcal{M}$ and $\mathcal{S}$, in which case the algorithm returns a counterexample symbolic word. The presence of such a conflict would then indeed imply that $\mathcal{M} \not\approx_{sym} \mathcal{S}$. □

We can now prove that Algorithm 9 is k-$\overline{C}$-complete, where k is an arbitrary natural number, and C is a minimal and prefix-closed state cover for the specification:

Corollary 5.8.4. Let $\mathcal{S}$ be a minimal, s-learnable, t-observable gMMT, and let C be a minimal and prefix-closed state cover for $\mathcal{S}$. Let k be a natural number. Then Algorithm 1 is k-$\overline{C}$-complete.

Proof. Let $\mathcal{M}$ be an MMT in $\mathcal{U}_k^{\overline{C}} \cup \mathcal{U}^{\overline{C}}$. Then Corollary 5.8.3 tells us that Algorithm 9 returns `yes` iff $\mathcal{M} \approx_{sym} \mathcal{S}$, as required. □

Chapter 6

Conclusions and Future Work

In this thesis, we developed k-A-complete conformance testing procedures for both the MM1Ts of Vaandrager et al. [2023], and the MMTs of Bruyère et al. [2024]. We developed these procedures to provide the basis for approximate equivalence oracles with proven correctness guarantees for the active MM1T learning method from Vaandrager et al. [2023], and the active MMT learning method from Bruyère et al. [2024]. We additionally provide a way to minimize MM1Ts, the existence of which ensures that any MM1T can be used as a specification for our MM1T testing procedure. The procedure also returns a counterexample input sequence iff the specification and the SUT are not untimed equivalent, granted that the SUT is in $\mathcal{U}_k^C \cup \mathcal{U}^C$ (with k a natural number and C a minimal and prefix-closed state cover for the specification). Our work has thus resulted in the first approximate MM1T equivalence oracle with proven correctness guarantees. The MM1T learning method can use our approximate oracle to inherit these correctness guarantees.

Our proofs for the validity and k-A-completeness of our MMT testing procedure rely on our notion of t-observability. This notion allows us to distinguish observation tree states based on additional behavior, compared to the MMT learning method's apartness for MMT states. We provide an algorithm that makes any s-learnable MMT t-observable. In doing so, we remove one of the two remaining barriers that could make the hypotheses of the MMT learning procedure unsuitable as specifications for our testing procedure. The sole remaining barrier is that of minimality: there is currently no proven method for minimizing MMTs, and finding one would allow our MMT conformance testing procedure to function as the first approximation of an MMT equivalence oracle with proven correctness guarantees. Our decision to use gMMTs, rather than MMTs for the specifications makes our procedure more flexible in use, since MMTs can easily be converted into symbolically equivalent gMMTs. Bruyère et al. [2024] also provides a way to convert gMMTs into symbolically equivalent MMTs, but this generally leads to a factorial blowup in the size of the state space. We also argue that minimal MMTs should be easier to obtain than minimal gMMTs.

Future Work Our first suggestion for future enhancements is to find an algorithm that can minimize MMTs into symbolically equivalent and minimal (g)MMTs.

It would also be interesting to analyze the efficiency of our testing procedures. We would like to determine an upper bound on the number of time units that it may take our two procedures to determine whether a given specification and SUT are equivalent. We think it would also be worthwhile to implement these procedures. Doing so would allow us to benchmark their efficiency.

We can further increase the efficiency of at least our MMT conformance testing procedure. For example, our MMT testing procedure may at times make states of the observation tree active explored by extending the observation tree, in order to ensure that two given observation tree states are apart. It even does so when the condition for this apartness does not in any way rely on states being active explored, such as for an apartness based on differences in output symbols. We can modify our MMT testing procedure such that it will no longer extend the observation tree, just to make states active explored in situations in which we know this not to be necessary.

Vaandrager et al. [2022] use adaptive distinguishing sequences (ADs) [Lee and Yannakakis, 1994] to optimize two of the non-deterministic rules of their active learning method. They note that although their method's asymptotic complexity is unaffected by this change, its running time on actual benchmarks re-

ceives "an effective boost". It would be interesting to incorporate ADs into our testing procedures, and to evaluate whether this leads to notable performance improvements.

Vaandrager et al. [2024] prove that for Mealy machines, k-A-completeness subsumes k-completeness, which means that any testing method that is k-A-complete is also k-complete. A natural question to answer is whether this is also the case for MM1Ts and MMTs which, if true, would prove that our testing procedures are k-complete in addition to being k-A-complete.

Bibliography

D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2): 87–106, Nov. 1987. ISSN 08905401. URL https://doi.org/10.1016/0890-5401(87)90052-6.

V. Bruyère, B. Garhewal, G. A. Pérez, G. Staquet, and F. W. Vaandrager. Active Learning of Mealy Machines with Timers. *CoRR*, abs/2403.02019, 2024. URL https://doi.org/10.48550/arXiv.2403.02019.

T. S. Chow. Testing Software Design Modeled by Finite-State Machines. *IEEE Trans. Software Eng.*, 4(3): 178–187, 1978. URL https://doi.org/10.1109/TSE.1978.231496.

R. Dorofeeva, K. El-Fakih, and N. Yevtushenko. An Improved Conformance Testing Method. In *Formal Techniques for Networked and Distributed Systems - FORTE 2005, 25th IFIP WG 6.1 International Conference, Taipei, Taiwan, October 2-5, 2005, Proceedings*, volume 3731 of *Lecture Notes in Computer Science*, pages 204–218. Springer, 2005. URL https://doi.org/10.1007/11562436_16.

R. Dorofeeva, K. El-Fakih, S. Maag, A. R. Cavalli, and N. Yevtushenko. FSM-based conformance testing methods: A survey annotated with experimental evaluation. *Information and Software Technology*, 52 (12):1286–1297, Dec. 2010a. ISSN 0950-5849. URL https://doi.org/10.1016/j.infsof.2010.07.001.

R. Dorofeeva, K. El-Fakih, S. Maag, A. R. Cavalli, and N. Yevtushenko. FSM-based conformance testing methods: A survey annotated with experimental evaluation. *Information and Software Technology*, 52 (12):1286–1297, 2010b. URL https://doi.org/10.1016/j.infsof.2010.07.001.

S. Fujiwara, G. von Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test Selection Based on Finite State Models. *IEEE Trans. Software Eng.*, 17(6):591–603, 1991. URL https://doi.org/10.1109/32.87284.

H. Geuvers and B. Jacobs. Relating Apartness and Bisimulation. *Logical Methods in Computer Science*, 17, 2021. URL https://doi.org/10.46298/LMCS-17(3:15)2021.

M. N. Irfan, C. Oriat, and R. Groz. Angluin style finite state machine inference with non-optimal counterexamples. In *Proceedings of the first international workshop on model inference in testing*, pages 11–19, 2010. URL https://doi.org/10.1145/1868044.1868046.

M. Isberner, F. Howar, and B. Steffen. The TTT Algorithm: A Redundancy-Free Approach to Active Automata Learning. In B. Bonakdarpour and S. A. Smolka, editors, *Runtime Verification*, pages 307–322, Cham, 2014. Springer International Publishing. ISBN 978-3-319-11164-3. doi: 10.1007/978-3-319-11164-3_26.

D. Lee and M. Yannakakis. Testing finite-state machines: State identification and verification. *IEEE Transactions on computers*, 43(3):306–320, 1994. URL https://doi.org/10.1109/12.272431.

G. Luo, A. Petrenko, and G. v. Bochmann. Selecting Test Sequences for Partially-Specified Nondeterministic Finite State Machines. In T. Mizuno, T. Higashino, and N. Shiratori, editors, *Protocol Test Systems: 7th workshop 7th IFIP WG 6.1 international workshop on protocol text systems*, pages 95–110. Springer US, Boston, MA, 1995. ISBN 978-0-387-34883-4. URL https://doi.org/10.1007/978-0-387-34883-4_6.

G. H. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 1955. URL https://doi.org/10.1002/j.1538-7305.1955.tb03788.x.

M. Shahbaz and R. Groz. Inferring Mealy Machines. In A. Cavalcanti and D. R. Dams, editors, *FM 2009: Formal Methods*, pages 207–222, Berlin, Heidelberg, 2009. Springer. ISBN 978-3-642-05089-3. URL https://doi.org/10.1007/978-3-642-05089-3_14.

A. S. Troelstra and H. Schwichtenberg. *Basic proof theory*. Number 43 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2 edition, 2000. URL https://doi.org/10.1017/CBO9781139168717.

F. Vaandrager. A New Perspective on Conformance Testing Based on Apartness. In *Logics and Type Systems in Theory and Practice: Essays Dedicated to Herman Geuvers on The Occasion of His 60th Birthday*, volume 14560 of *Lecture notes in computer science*, pages 225–240. Springer, 2024. URL https://doi.org/10.1007/978-3-031-61716-4_15.

F. Vaandrager, M. Ebrahimi, and R. Bloem. Learning Mealy machines with one timer. *Information and Computation*, 295:105013, 2023. ISSN 0890-5401. URL https://doi.org/10.1016/j.ic.2023.105013. Special Issue: Selected papers of the 15th International Conference on Language and Automata Theory and Applications, LATA 2021.

F. W. Vaandrager, B. Garhewal, J. Rot, and T. Wißmann. A New Approach for Active Automata Learning Based on Apartness. In *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*, volume 13243 of *Lecture Notes in Computer Science*, pages 223–243. Springer, 2022. URL https://doi.org/10.1007/978-3-030-99524-9_12.

F. W. Vaandrager, P. Fiterau-Brostean, and I. Melse. Completeness of FSM Test Suites Reconsidered. *CoRR*, abs/2410.19405, 2024. URL https://doi.org/10.48550/arXiv.2410.19405.

M. P. Vasilevskii. Failure diagnosis of automata. *Cybernetics*, 9(4):653–665, 1973.

K. Zuse. Der Plankalkül. *Berichte der Gesellschaft für Mathematik und Datenverarbeitung*, 63:235–285, 1972. URL http://zuse.zib.de/item/gHI1cNsUuQweHB6.

Appendix A

MM1T Material from the Literature

This appendix provides some MM1T-related material from the literature that our results from Chapter 3 rely on. We included this material for the reader's convenience. All of this material is taken from Vaandrager et al. [2023].

A.1 Expressing MM1Ts and Mealy Machines in Terms of One Another

There is a mapping that expresses MM1Ts in terms of Mealy machines, as well as a mapping that can express certain Mealy machines in terms of MM1Ts. The first mapping takes an MM1T $\mathcal{M}$ and returns a Mealy machine $\mathsf{Mealy}(\mathcal{M})$ that uses the same states, inputs and transitions as $\mathcal{M}$. We add self-loop transitions to all states in Q_{off} to make the Mealy machine input complete. We assign a special output nil to each of these self-loops. The remaining transitions receive an output given by a pair of the output from $\mathcal{M}$, together with the timer update.
The inverse operation assigns a tuple $\mathsf{MM1T}(\mathcal{N})$ to every Mealy machine $\mathcal{N}$ for which the input and output sets meet certain requirements. This tuple is not generally guaranteed to be a valid MM1T. For example, $\mathcal{N}$ may have a $\mathsf{timeout}$ transition from its initial state, which would mean that $\mathsf{MM1T}(\mathcal{N})$ would have one as well.

Definition A.1.1. Let $\mathcal{M} = (Q, q_{\mathcal{I}}, I, O, \delta, \lambda, \tau)$ be an MM1T. Then $\mathsf{Mealy}(\mathcal{M})$ is the Mealy machine described by the tuple $(Q, q_{\mathcal{I}}, I, O', \delta', \lambda')$, where:

$$O' = (O \times (\mathbb{N}^{>0} \cup \{\bot\})) \cup \{\mathsf{nil}\}$$

$$\delta'(q,i) = \begin{cases} \delta(q,i) & \text{if } \delta(q,i)\downarrow \\ q & \text{otherwise} \end{cases}$$

$$\lambda'(q,i) = \begin{cases} (\lambda(q,i), \tau_{\bot}(q,i)) & \text{if } \lambda(q,i)\downarrow \\ \mathsf{nil} & \text{otherwise} \end{cases}$$

Conversely, suppose $\mathcal{N} = (P, p_{\mathcal{I}}, I, O, \delta, \lambda)$ is a Mealy machine with $\mathsf{timeout} \in I$ and $O \subseteq (\Omega \times (\mathbb{N}^{>0} \cup \{\bot\})) \cup \{\mathsf{nil}\}$, for some set Ω. Then the above construction can be reversed to obtain the MM1T $\mathsf{MM1T}(\mathcal{N}) =$

$(P_{off} \cup P_{on}, p_{\mathcal{I}}, I, O', \delta', \lambda', \tau)$, where:

$$
\begin{aligned}
O' &= \{o \in \Omega \mid \exists n \in \mathbb{N}^{>0} \cup \{\bot\} \colon (o, n) \in O\} \\
P_{off} &= \{p \ \in P \mid \lambda(p, \mathsf{timeout}) = \mathsf{nil}\} \\
P_{on} &= P \setminus P_{off} \\
\delta'(p,i) &= \begin{cases} \delta(p,i) & \text{if } p \in P_{on} \vee i \neq \mathsf{timeout} \\ \text{undefined} & \text{otherwise} \end{cases} \\
\lambda'(p,i) &= \begin{cases} \pi_1(\lambda(p,i)) & \text{if } \lambda(p,i) \neq \mathsf{nil} \\ \text{undefined} & \text{otherwise} \end{cases} \\
\tau(p,i) &= \begin{cases} \pi_2(\lambda(p,i)) & \text{if } \lambda(p,i) \in O' \times \mathbb{N}^{>0} \\ \text{undefined} & \text{otherwise} \end{cases}
\end{aligned}
$$

A.2 Bisimulations Between MM1Ts

Bisimulations for MM1Ts can be defined as follows:

Definition A.2.1. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be MM1Ts with the same inputs, where $\mathcal{M}_j = (Q_j, q^j_{\mathcal{I}}, I, O_j, \delta_j, \lambda_j, \tau_j)$, for $j = 1, 2$. A **bisimulation** between $\mathcal{M}_1$ and $\mathcal{M}_2$ is a relation $R \subseteq Q_1 \times Q_2$ satisfying, for every $q_1 \in Q_1, q_2 \in Q_2$ and $i \in I$:

$$q^1_{\mathcal{I}} \ R \ q^2_{\mathcal{I}} \tag{A.1}$$

$$q_1 \ R \ q_2 \ \wedge \ \delta_1(q_1, i)\downarrow \quad \Longrightarrow \quad \delta_2(q_2, i)\downarrow \ \wedge \ \delta_1(q_1, i) \ R \ \delta_2(q_2, i) \ \wedge \ \lambda_1(q_1, i) = \lambda_2(q_2, i) \ \wedge \ \tau_1(q_1, i) = \tau_2(q_2, i) \tag{A.2}$$

$$q_1 \ R \ q_2 \ \wedge \ \delta_2(q_2, i)\downarrow \quad \Longrightarrow \quad \delta_1(q_1, i)\downarrow \tag{A.3}$$

We write $\mathcal{M}_1 \simeq \mathcal{M}_2$ iff there is a bisimulation R between $\mathcal{M}_1$ and $\mathcal{M}_2$.

The following property holds:

Lemma A.2.1. Let $\mathcal{M}$ and $\mathcal{N}$ be two MM1Ts with the same inputs. Then $\mathcal{M} \simeq \mathcal{N}$ iff $\mathcal{M} \approx_{untimed} \mathcal{N}$.

Appendix B

Proofs Related to MM1Ts

This appendix contains some proofs related to MM1Ts that we omitted from the main text to increase the documents's readability.

B.1 Properties and Proofs Related to the k-A-Completeness of the MM1T Testing Procedure

The properties and proofs used in Vaandrager et al. [2024] to prove a sufficient condition for the k-A-completeness of Mealy machine test suites form the basis of our proofs for the k-A-completeness of our MM1T conformance testing procedure. We highlight the additions we made compared to the work from Vaandrager et al. [2024] in green, and the remaining differences between our results and theirs in blue. We first provide the auxiliary lemmas, before we conclude with our proofs of Theorem 3.3.1 and Lemma 3.3.2.

Lemma B.1.1. Let $\mathcal{M}$ and $\mathcal{N}$ be MM1Ts, and let $f\colon \mathcal{M} \to \mathcal{N}$ be a functional simulation. Let $q, q' \in Q^{\mathcal{M}}$ and $\sigma \in I^*$. Then:

$$q \xrightarrow{\sigma} q' \quad \Longrightarrow \quad f(q) \xrightarrow{\sigma} f(q').$$

Proof. Suppose that $q \xrightarrow{\sigma} q'$. Then $q' = \delta^{\mathcal{M}^*}(q, \sigma)$. We need to prove that $\delta^{\mathcal{N}^*}(f(q), \sigma) = f(q')$. We prove the property by an induction on the length of σ:

- Base case: $\sigma = \epsilon$. Then $q' = q$. We can see that:

  $$\delta^{\mathcal{N}^*}(f(q), \sigma) = \delta^{\mathcal{N}^*}(f(q), \epsilon) = f(q) = f(q'),$$

  as required.

- Inductive step case: $\sigma = \rho \cdot i$ with $\rho \in I^*$ and $i \in I$. We use the induction hypothesis:

  $$q \xrightarrow{\rho} q' \quad \Longrightarrow \quad f(q) \xrightarrow{\rho} f(q').$$

  The induction hypothesis tells us that $\delta^{\mathcal{N}^*}(f(q), \rho) = f(q')$. Let $q'' = \delta^{\mathcal{M}}(q', i)$. The definition of functional MM1T simulations tells us that therefore, $\delta^{\mathcal{N}}(f(q'), i) = f(q'')$. We thus know that $\delta^{\mathcal{N}^*}(f(q), \rho \cdot i) = f(q'')$. We can also see that $\delta^{\mathcal{M}^*}(q, \rho \cdot i) = q''$. Therefore:

  $$q \xrightarrow{\rho \cdot i} q'' \quad \Longrightarrow \quad f(q) \xrightarrow{\rho \cdot i} f(q''),$$

  as required.

□

The following auxiliary lemma is the only lemma in this appendix that we didn't adapt from a lemma from Vaandrager et al. [2024]:

Lemma B.1.2. Let $\mathcal{M}$ be an MM1T, and let $\mathcal{T}$ be an observation tree for $\mathcal{M}$. Let $f\colon \mathcal{T} \to \mathcal{M}$ be a functional simulation. Let $q_0, q_n \in Q^{\mathcal{T}}$. Let $\sigma = i_1 \dots i_n \in I^*$. Let $\pi = q_0 \xrightarrow{\sigma} q_n$. Suppose that the basis of a stratification of $Q^{\mathcal{T}}$ is complete, and that for all $l \in \{0, \dots, n\}$, if $q_l \in F^j$, then F^j is complete. Then:

$$f(q_0) \xrightarrow{\sigma} f(q_n) \quad \Longrightarrow \quad q_0 \xrightarrow{\sigma} .$$

Proof. Suppose that $f(q_0) \xrightarrow{\sigma} f(q_n)$. Then $f(q_n) = \delta^{\mathcal{M}^*}(f(q_0), \sigma)$. We need to prove that $\delta^{\mathcal{T}^*}(q_0, \sigma)\downarrow$. We prove the property by an induction on the length of σ:

- Base case: $\sigma = \epsilon$. We trivially get that:

  $$q_0 \xrightarrow{\sigma} = q_0 \xrightarrow{\epsilon} = q_0,$$

  as required.

- Inductive step case: $\sigma = i_1 \dots i_{k+1}$ and $k + 1 \leq n$, for some $k \in \mathbb{N}$. Let $\rho = i_1 \dots i_k$. We use the induction hypothesis:

  $$f(q_0) \xrightarrow{\rho} f(q_k) \quad \Longrightarrow \quad q_0 \xrightarrow{\rho} .$$

  The induction hypothesis tells us that $q_0 \xrightarrow{\rho}$. Let $q_k = \delta^{\mathcal{T}^*}(q_0, \rho)$. We know from our final assumption that whether q_k is in the basis or in a frontier, this basis or frontier is complete. This implies in either case that:

  - if $i_{k+1} = \mathsf{timeout}$, then we know that $f(q_k) \in Q^{\mathcal{M}}_{on}$. Then since f is a functional simulation, $q_k \in Q^{\mathcal{T}}_{on}$. Thus, since q_k is in a basis or frontier that is complete, q_k has an outgoing $\mathsf{timeout}$ transition. This implies that $q_k \xrightarrow{i_{k+1}}$, which implies that $q_k \xrightarrow{\sigma}$, as required.
  - if $i_{k+1} \neq \mathsf{timeout}$, then since q_k is in a basis or frontier that is complete, $q_k \xrightarrow{i_{k+1}}$. This implies that $q_k \xrightarrow{\sigma}$, as required.

□

Lemma B.1.3. Let $\mathcal{M}$ be an MM1T, and let $\mathcal{T}$ be an observation tree for $\mathcal{M}$. Let $f\colon \mathcal{T} \to \mathcal{M}$ be a functional simulation. Let B be the basis of a stratification of $Q^{\mathcal{T}}$. If all states of B are pairwise apart, then f restricted to B is injective.

Proof. Let q, q' be two distinct states of B. Since $q \# q'$, Lemma 3.2.4 tells us that $f(q) \not\approx_{untimed} f(q')$. This implies that $f(q) \neq f(q')$, which tells us that f restricted to B is injective when all of B's states are pairwise apart. □

Lemma B.1.4. Let $\mathcal{M}$ be an MM1T, and let $\mathcal{T}$ be an observation tree for $\mathcal{M}$. Let $f\colon \mathcal{T} \to \mathcal{M}$ be a functional simulation. Let B be the basis of a stratification of $Q^{\mathcal{T}}$, such that $|B| = |Q^{\mathcal{M}}|$. If all states of B are pairwise apart, then f restricted to B is a bijection.

Proof. Lemma B.1.3 tells us that f restricted to B is injective. Since $|B| = |Q^{\mathcal{M}}|$, we may conclude that f is a bijection between B and $Q^{\mathcal{M}}$. □

Lemma B.1.5. Let $\mathcal{M}$ be an MM1T, and let $\mathcal{T}$ be an observation tree for $\mathcal{M}$. Let $f\colon \mathcal{T} \to \mathcal{M}$ be a functional simulation. Let B be the basis of a stratification of $Q^{\mathcal{T}}$, such that $|B| = |Q^{\mathcal{M}}|$, and such that all states of B are pairwise apart. Let $q \in Q^{\mathcal{T}}$. Then:

$$\exists r \in B\colon \qquad r \in \mathcal{C}(q) \quad \wedge \quad f(q) = f(r).$$

Proof. Let $f(q) = u$. Lemma B.1.4 tells us that f restricted to B is a bijection. Let $r \in B$ be the unique state with $f(r) = u$. Since $f(q) = f(r)$, Lemma 3.2.4 implies that q and r are not apart. Hence $r \in \mathcal{C}(q)$. □

Lemma B.1.6. Let $\mathcal{S}$ and $\mathcal{M}$ be MM1Ts, and let $\mathcal{T}$ be an observation tree for both $\mathcal{S}$ and $\mathcal{M}$. Let C be a prefix-closed state cover for $\mathcal{S}$, and let B be the basis of a stratification of $Q^{\mathcal{T}}$ induced by C. Suppose that all states of B are identified. Then $\mathcal{M} \notin \mathcal{U}^C$.

Proof. Let $f : \mathcal{T} \to \mathcal{M}$. Suppose that $\sigma, \rho \in C$ with $\sigma \neq \rho$. Since B is the basis of a stratification induced by C, $q = \delta^{\mathcal{T}^*}(q_I^{\mathcal{T}}, \sigma) \in B$ and $q' = \delta^{\mathcal{T}^*}(q_I^{\mathcal{T}}, \rho) \in B$. The fact that $\mathcal{T}$ is a tree thus implies that $q \neq q'$. Since all states of B are identified, they are all pairwise apart. We thus know that $q \# q'$. Lemma 3.2.4 now tells us that $f(q) \not\approx_{untimed} f(q')$. By Lemma B.1.1, $f(q) = \delta^{\mathcal{M}^*}(q_I^{\mathcal{M}}, \sigma)$ and $f(q') = \delta^{\mathcal{M}^*}(q_I^{\mathcal{M}}, \rho)$. This implies that $\delta^{\mathcal{M}^*}(q_I^{\mathcal{M}}, \sigma) \not\approx_{untimed} \delta^{\mathcal{M}^*}(q_I^{\mathcal{M}}, \rho)$, which implies that $\mathcal{M} \notin \mathcal{U}^C$. □

B.1.1 The proof of Theorem 3.3.1

This proof of Theorem 3.3.1 is obtained by slightly modifying Vaandrager et al. [2024]'s proof for its Theorem 4.5. Every difference between the two proofs is needed to either account for the difference between the notions of state equivalence that we use for Mealy machines and for MM1Ts, or between the differences in the notions of bisimulation or stratification for these two model types. Remember that we use trace equivalence for equivalences between Mealy machine states, and untimed equivalence for equivalences between MM1T states. We use the standard notion of bisimulation for Mealy machines. For MM1Ts, we use the bisimulation introduced in Vaandrager et al. [2023]. We also include this bisimulation in Appendix A.2.

We performed four small additions, and one further change to the proof for Theorem 4.5. The four additions are all needed to account for differences between the notions of bisimulation, of stratification, and of completeness of the model. The remaining difference is needed because unlike bisimulations for complete Mealy machines, bisimulations for complete MM1Ts still have to account for transitions that are undefined.

Proof. Lemma B.1.6 tells us that $\mathcal{M} \notin \mathcal{U}^C$. Therefore, $\mathcal{M} \in \mathcal{U}_k^C$. The fact that B is a basis induced by a minimal prefix-closed state cover C implies that $\mathsf{access}(B) = C$. We thus know that since $\mathcal{M} \in \mathcal{U}_k^C$, there are:

$$\forall q \in Q^{\mathcal{M}} : \quad \exists \sigma \in (\mathsf{access}(B) = C), \exists \rho \in I^{\leq k} : \delta^{\mathcal{M}^*}(q_I^{\mathcal{M}}, \sigma \cdot \rho) = q. \tag{B.1}$$

Let $f : \mathcal{T} \to \mathcal{S}$ and $g : \mathcal{T} \to \mathcal{M}$ be functional simulations. We define a relation $R \subseteq Q^{\mathcal{S}} \times Q^{\mathcal{M}}$ as:

$$(s, q) \in R \quad \iff \quad \exists t \in B \cup F^{<k} : \quad f(t) = s \wedge g(t) = q.$$

We claim that R is a bisimulation between $\mathcal{S}$ and $\mathcal{M}$.

1. Since f is a functional simulation from $\mathcal{T}$ to $\mathcal{S}$, $f(q_I^{\mathcal{T}}) = s_I^{\mathcal{S}}$, and since g is a functional simulation from $\mathcal{T}$ to $\mathcal{M}$, $g(q_I^{\mathcal{T}}) = q_I^{\mathcal{M}}$. Using $q_I^{\mathcal{T}} \in B$, this implies that $(s_I^{\mathcal{S}}, q_I^{\mathcal{M}}) \in R$.

2. Suppose that $(s, q) \in R$ and $i \in I$. We need to show that if either $\lambda^{\mathcal{S}}(s, i)\downarrow$ or $\lambda^{\mathcal{M}}(q, i)\downarrow$, then: $\lambda^{\mathcal{S}}(s, i) = \lambda^{\mathcal{M}}(q, i)$, $\tau^{\mathcal{S}}(s, i) = \tau^{\mathcal{M}}(q, i)$ and $(\delta^{\mathcal{S}}(s, i), \delta^{\mathcal{M}}(q, i)) \in R$.
   Since $(s, q) \in R$, there exists a state $t \in B \cup F^{<k}$ such that $f(t) = s$ and $g(t) = q$. Since $B, F^{<k}$ are all complete and $\mathcal{T}$ is an observation tree for both $\mathcal{M}$ and $\mathcal{S}$, $\delta^{\mathcal{S}}(s, i)\downarrow$ iff $\delta^{\mathcal{T}}(t, i)\downarrow$ iff $\delta^{\mathcal{M}}(q, i)\downarrow$.
   We thus know that if either of $\delta^{\mathcal{S}}(s, i)$ or $\delta^{\mathcal{M}}(q, i)$ is undefined, then so is the other. In that case, there are no properties that need to hold for (s, q) and i in order for R to be a bisimulation. We will thus assume that $\delta^{\mathcal{S}}(s, i)$ and $\delta^{\mathcal{M}}(q, i)$ are both defined in the remainder of item 2. Let $s' = \delta^{\mathcal{S}}(s, i)$ and $q' = \delta^{\mathcal{M}}(q, i)$.

   - Since f and g are functional simulations, $\lambda^{\mathcal{T}}(t, i) = \lambda^{\mathcal{S}}(s, i)$ and $\lambda^{\mathcal{T}}(t, i) = \lambda^{\mathcal{M}}(q, i)$. This tells us that $\lambda^{\mathcal{S}}(s, i) = \lambda^{\mathcal{M}}(q, i)$, as required.
   - Since f and g are functional simulations, $\tau^{\mathcal{T}}(t, i) = \tau^{\mathcal{S}}(s, i)$ and $\tau^{\mathcal{T}}(t, i) = \tau^{\mathcal{M}}(q, i)$. This tells us that $\tau^{\mathcal{S}}(s, i) = \tau^{\mathcal{M}}(q, i)$, as required.

- Let $t' = \delta^{\mathcal{T}}(t, i)$. Since f and g are functional simulations, $f(t') = s'$ and $g(t') = q'$. In order to prove $(s', q') \in R$, we consider two cases:
  (a) if $t' \in B \cup F^{<k}$, then, since $f(t') = s'$ and $g(t') = q'$, $(s', q') \in R$ follows directly from the definition of R.
  (b) $t' \in F^k$. Equation (B.1) tells us that there are sequences $\sigma \in \mathsf{access}(B)$ and $\rho \in I^{\leq k}$ such that $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}}, \sigma \cdot \rho) = q'$. By the assumption that $B, F^{<k}$ are all complete, the fact that $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}}, \sigma \cdot \rho)\downarrow$ implies by Lemma B.1.2 that $t'' = \delta^{\mathcal{T}^*}(q_{\mathcal{I}}^{\mathcal{T}}, \sigma \cdot \rho)$ is defined. By Lemma B.1.1, $g(t'') = q'$. Then, by Lemma 3.2.4, t' and t'' are not apart. We claim that t' and t'' have the same candidate set:
     i. $t'' \in B$. Then since B is a basis and all basis states are identified, $\mathcal{C}(t'') = \{t''\}$. Since $\neg(t' \# t'')$ and t' is identified, $\mathcal{C}(t') = \{t''\}$. Hence, $\mathcal{C}(t') = \mathcal{C}(t'')$.
     ii. $t'' \in F^{<k}$. Then by Equation (3.1) and since $\neg(t' \# t'')$, $\mathcal{C}(t') = \mathcal{C}(t'')$.
     Since t' is identified, $\mathcal{C}(t') = \{r\}$, for some $r \in B$. By Lemma B.1.5, $f(t') = f(r)$. Since $\mathcal{C}(t') = \mathcal{C}(t'')$, $\mathcal{C}(t'') = \mathcal{C}(t') = \{r\}$, which also implies that t'' is identified. Applying Lemma B.1.5 now tells us that $f(t'') = f(r)$. Hence $f(t'') = f(t') = s'$. This in turn implies that $(s', q') \in R$, which completes the proof that R is a bisimulation.

We have thus proven that $\mathcal{S} \simeq \mathcal{M}$. The theorem now follows by application of Lemma A.2.1. □

B.1.2 Proof of Lemma 3.3.2

Proof. We prove both directions of the bi-implication:

- Assume that $\mathcal{C}(q) = \mathcal{C}(q') \vee q \# q'$. Suppose that $r \in B$ with $r \# q$. We need to show that $r \# q' \vee q \# q'$. If the $q \# q'$ from the assumption holds, then we are already done. So suppose that $\neg(q \# q')$ and $\mathcal{C}(q) = \mathcal{C}(q')$. Then, since $r \# q$, $r \notin \mathcal{C}(q)$. Therefore, $r \notin \mathcal{C}(q')$. This implies that $r \# q'$, as required.

- Assume that $(\forall r \in B \colon r \# q \implies r \# q' \vee q \# q')$. Suppose that $\neg(q \# q')$. We need to show that $\mathcal{C}(q) = \mathcal{C}(q')$. Since q is identified, all basis states except one are apart from q. Let r be the unique basis state that is not apart from q. By our assumption, q' is apart from all states in $B \setminus \{r\}$. Thus $\mathcal{C}(q') \subseteq \{r\}$. By Lemma B.1.5, $\mathcal{C}(q')$ contains at least one state. Therefore, we conclude that $\mathcal{C}(q') = \{r\}$. This implies that $\mathcal{C}(q) = \mathcal{C}(q')$, as required.

□

B.2 Proof of Lemma 3.1.1

Lemma 3.1.1 imposes three conditions on MM1T $\mathsf{MM1T}(M)$. We split this lemma into three properties which we prove in the next three subsections.

Before we do so however, we first prove the auxiliary lemma that two MM1T states cannot be untimed equivalent if there is an input sequence for which one of them reaches a state in which the timer is on, and the other reaches a state in which the timer is off:

Lemma B.2.1. Let q and p be states of possibly two different MM1Ts with the same set of inputs I, and let $\sigma \in I^*$ be an input sequence such that $\delta^*(q, \sigma)\downarrow$ and $\delta^*(p, \sigma)\downarrow$. Let $q' = \delta^*(q, \sigma)$ and $p' = \delta^*(p, \sigma)$. If the timer is either on in q' and off in p', or off in q' and on in p', then $q \not\approx_{untimed} p$.

Proof. We can see from the definition of the $uWord$ function that $uWord_{q'}(\mathsf{timeout})\downarrow$ iff the timer is on in q'. We can also see that $uWord_{q'}(\mathsf{timeout})\downarrow$ iff $\delta(q', \mathsf{timeout})\downarrow$. Since $\delta(q', \mathsf{timeout}) = \delta^*(q, \sigma\ \mathsf{timeout})$, we know that $\delta(q', \mathsf{timeout})\downarrow$ iff $\delta^*(q, \sigma\ \mathsf{timeout})\downarrow$. We know from the definition of the $uWord$ function that $\delta^*(q, \sigma\ \mathsf{timeout})\downarrow$ iff $uWord_q(\sigma\ \mathsf{timeout})\downarrow$. Therefore, $uWord_q(\sigma\ \mathsf{timeout})\downarrow$ iff the timer is on in q'.
We can use a similar argument to show that $uWord_p(\sigma\ \mathsf{timeout})\downarrow$ iff the timer is on in p'. Therefore:

- if the timer is on in q' and off in p', then $uWord_q(\sigma\ \mathsf{timeout})\downarrow$ and $uWord_p(\sigma\ \mathsf{timeout})\uparrow$, and

- if the timer is off in q' and on in p', then $uWord_q(\sigma\ \mathsf{timeout})\uparrow$ and $uWord_p(\sigma\ \mathsf{timeout})\downarrow$.

In either case, $uWord_q \neq uWord_p$, which means that $q \not\approx_{untimed} p$. □

The first of the three properties induced by Lemma 3.1.1 is:

Lemma B.2.2. Let $\mathcal{M}$ be an MM1T. Let M be a Mealy machine obtained by minimizing Mealy machine $\mathsf{Mealy}(\mathcal{M})$. Then, $\mathsf{MM1T}(M)$ is a valid MM1T.

We prove Lemma B.2.2 in Appendix B.2.1.
The second property is:

Lemma B.2.3. Let $\mathcal{M}$ be an MM1T. Let M be a Mealy machine obtained by minimizing Mealy machine $\mathsf{Mealy}(\mathcal{M})$. Then, $\mathsf{MM1T}(M)$ is minimal.

We prove Lemma B.2.3 in Appendix B.2.2.
The final property is:

Lemma B.2.4. Let $\mathcal{M}$ be an MM1T. Let M be a Mealy machine obtained by minimizing Mealy machine $\mathsf{Mealy}(\mathcal{M})$. Then, $\mathsf{MM1T}(M) \approx_{untimed} \mathcal{M}$.

We prove Lemma B.2.4 in Appendix B.2.3.
The fact that Lemma 3.1.1 holds follows directly from the fact that lemmas B.2.2, B.2.3 and B.2.4 hold.

B.2.1 Proof of Lemma B.2.2

Proof. Methods for minimizing Mealy machines often partition the Mealy machine's set of states into a set of blocks, where all states in the same block are equivalent to one another. Each of the original Mealy machines's states occurs in exactly one of the partition's blocks. The minimal Mealy machine obtained from the method then represents each of these blocks with exactly one state, and the transitions between its states are transitions that can exist between the states of the blocks represented by its states.
We prove that $\mathsf{MM1T}(M)$ is a valid MM1T:

1. The definition of $\mathsf{MM1T}(M)$ tells us that $Q_{on}^{\mathsf{MM1T}(M)}$ and $Q_{off}^{\mathsf{MM1T}(M)}$ are disjoint.
2. The initial state of $\mathsf{Mealy}(\mathcal{M})$ is defined to be the initial state of $\mathcal{M}$, which is $q_{\mathcal{I}}$. We thus know that in M, $q_{\mathcal{I}}$ is in a block B_0 that is represented by a state q_0^M for which $\lambda^M(q_0^M, \mathsf{timeout}) = \mathsf{nil}$. We thus get:

$$
\begin{aligned}
& q_{\mathcal{I}} \in Q_{off}^{\mathcal{M}} && \text{(since } q_{\mathcal{I}} \text{ is an initial state)} \\
\Leftrightarrow\ & \delta^{\mathcal{M}}(q_{\mathcal{I}}, \mathsf{timeout})\uparrow && \text{(Rule 2.1 applies to } \mathcal{M}) \\
\Leftrightarrow\ & \lambda^{\mathcal{M}}(q_{\mathcal{I}}, \mathsf{timeout})\uparrow && \text{(Rule 2.2 applies to } \mathcal{M}) \\
\Leftrightarrow\ & \lambda^{\mathsf{Mealy}(\mathcal{M})}(q_{\mathcal{I}}, \mathsf{timeout}) = \mathsf{nil} && \text{(by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})}) \\
\Leftrightarrow\ & \forall b \in B_{q_0^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(b, \mathsf{timeout}) = \mathsf{nil} && (B_{q_0^M} \text{ is a partition's block with } q_{\mathcal{I}} \in B_{q_0^M}) \\
\Leftrightarrow\ & \lambda^M(q_0^M, \mathsf{timeout}) = \mathsf{nil} && (B_{q_0^M} \text{ is a partition's block for } q_0^M) \\
\Leftrightarrow\ & q_0^M \in Q_{off}^{\mathsf{MM1T}(M)} && \text{(by definition of } Q_{off}^{\mathsf{MM1T}(M)})
\end{aligned}
$$

3. Model $\mathsf{MM1T}(M)$ is defined to have the same set of inputs as M. Since M is a Mealy machine, it must have a finite set of inputs, which thus implies that $\mathsf{MM1T}(M)$ has a finite set of inputs as well.
Since $\mathsf{Mealy}(\mathcal{M})$ is input complete, we know that $\mathsf{Mealy}(\mathcal{M})$ has for every state of $\mathcal{M}$ an outgoing transition for each of $\mathcal{M}$'s inputs, including $\mathsf{timeout}$. Thus, since M is a minimal version of $\mathsf{Mealy}(\mathcal{M})$, it must also have an outgoing transition for $\mathsf{timeout}$ for all of from all of its states. We thus know that $\mathsf{timeout} \in I^M$. Then, since $I^{\mathsf{MM1T}(M)} := I^M$, we know that $\mathsf{timeout} \in I^{\mathsf{MM1T}(M)}$.
4. We can see from the definition of $\mathsf{MM1T}(M)$'s transition function that $\mathsf{MM1T}(M)$ satisfies Rule 2.1.

5. We use the following auxiliary property in some of the next items:

$$
\begin{aligned}
&\forall b \in B_{q^M} : b \in Q^{\mathcal{M}}_{on} \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathcal{M}}(b, \mathsf{timeout})\downarrow && \text{(Rule 2.1 applies to } \mathcal{M}) \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(b, \mathsf{timeout}) \neq \mathsf{nil} && \text{(by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})}) \\
&\Leftrightarrow \lambda^{M}(q^M, \mathsf{timeout}) \neq \mathsf{nil} && (B_{q^M} \text{ is a partition's block for } q^M) \\
&\Leftrightarrow q^M \in Q^{\mathsf{MM1T}(M)}_{on} && \text{(by definition of } Q^{\mathsf{MM1T}(M)}_{on})
\end{aligned}
$$

6. We use the following auxiliary property in some of the next items:

$$
\begin{aligned}
&\forall b \in B_{q^M} : b \in Q^{\mathcal{M}}_{off} \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathcal{M}}(b, \mathsf{timeout})\uparrow && \text{(Rule 2.1 applies to } \mathcal{M}) \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(b, \mathsf{timeout}) = \mathsf{nil} && \text{(by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})}) \\
&\Leftrightarrow \lambda^{M}(q^M, \mathsf{timeout}) = \mathsf{nil} && (B_{q^M} \text{ is a partition's block for } q^M) \\
&\Leftrightarrow q^M \in Q^{\mathsf{MM1T}(M)}_{off} && \text{(by definition of } Q^{\mathsf{MM1T}(M)}_{off})
\end{aligned}
$$

7. The fact that Rule 2.2 holds for $\mathsf{MM1T}(M)$ follows from:

$$
\begin{aligned}
&\lambda^{\mathsf{MM1T}(M)}(q^M, i)\downarrow \\
&\Leftrightarrow \lambda^{M}(q^M, i) \neq \mathsf{nil} && \text{(by definition of } \mathsf{MM1T}(M)) \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(b, i) \neq \mathsf{nil} && (B_{q^M} \text{ is a partition's block for } q^M) \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathcal{M}}(b, i)\downarrow && \text{(by definition of } \mathsf{Mealy}(\mathcal{M})) \\
&\Leftrightarrow \forall b \in B_{q^M} : \big(b \in Q^{\mathcal{M}}_{on} \vee i \neq \mathsf{timeout}\big) && \text{(Rule 2.1 applies to } \mathcal{M}) \\
&\Leftrightarrow \big(\forall b \in B_{q^M} : b \in Q^{\mathcal{M}}_{on}\big) \vee i \neq \mathsf{timeout} \\
&\Leftrightarrow q^M \in Q^{\mathsf{MM1T}(M)}_{on} \vee i \neq \mathsf{timeout} && \text{(by Item 5)} \\
&\Leftrightarrow \delta^{\mathsf{MM1T}(M)}(q^M, i)\downarrow && \text{(by definition of } \delta^{\mathsf{MM1T}}(M))
\end{aligned}
$$

8. The fact that Rule 2.3 holds for $\mathsf{MM1T}(M)$ follows from:

$$
\begin{aligned}
&\tau^{\mathsf{MM1T}(M)}(q^M, i)\downarrow \\
&\Leftrightarrow \lambda^{M}(q^M, i) \in O^{\mathsf{MM1T}(M)} \times \mathbb{N}^{>0} && \text{(by definition of } \mathsf{MM1T}(M)) \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(b, i) \in O^{\mathsf{MM1T}(M)} \times \mathbb{N}^{>0} && (B_{q^M} \text{ is a partition's block for } q^M) \\
&\Rightarrow \forall b \in B_{q^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(b, i) \in O^{\mathcal{M}} \times \mathbb{N}^{>0} && \left(\begin{array}{l}\text{by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})} \\ \text{since } \lambda^{\mathsf{Mealy}(\mathcal{M})}(b, i) \neq \mathsf{nil} \\ \text{since } \lambda^{\mathcal{M}}(b, i) = \\ \quad \pi_1(\lambda^{\mathsf{Mealy}(\mathcal{M})}(b, i))\end{array}\right) \\
&\Leftrightarrow \forall b \in B_{q^M} : \tau^{\mathcal{M}}(b, i)\downarrow && \text{(by definition of } \mathsf{Mealy}(\mathcal{M})) \\
&\Rightarrow \forall b \in B_{q^M} : \delta^{\mathcal{M}}(b, i) \in Q^{\mathcal{M}}_{on} && \text{(Rule 2.3 applies to } \mathcal{M}) \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathcal{M}}(\delta^{\mathcal{M}}(b, i), \mathsf{timeout})\downarrow && \text{(Rule 2.1 applies to } \mathcal{M}) \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(\delta^{\mathcal{M}}(b, i), \mathsf{timeout}) \neq \mathsf{nil} && \text{(by definition of } \mathsf{Mealy}(\mathcal{M})) \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(\delta^{\mathsf{Mealy}(\mathcal{M})}(b, i), \mathsf{timeout}) \neq \mathsf{nil} && \left(\begin{array}{l}\text{by definition of } \mathsf{Mealy}(\mathcal{M}), \text{ and} \\ \text{because } \delta^{\mathcal{M}}(b, i) \in Q^{\mathcal{M}}_{on} \Rightarrow \delta^{\mathcal{M}}(b, i)\downarrow\end{array}\right) \\
&\Leftrightarrow \lambda^{M}(\delta^{M}(q^M, i), \mathsf{timeout}) \neq \mathsf{nil} && (B_{q^M} \text{ is a partition's block for } q^M)
\end{aligned}
$$

We had established that:

$$\forall b \in B_{q^M} : \delta^{\mathcal{M}}(b, i) \in Q^{\mathcal{M}}_{on}$$

This implies that:

$$\begin{aligned}
&\forall b \in B_{q^M} : \delta^{\mathcal{M}}(b, i) \in Q^{\mathcal{M}}_{on} \\
&\Rightarrow \forall b \in B_{q^M} : \delta^{\mathcal{M}}(b, i)\downarrow \\
&\Leftrightarrow \forall b \in B_{q^M} : (b \in Q^{\mathcal{M}}_{on} \vee i \neq \mathsf{timeout}) && \text{(Rule 2.1 applies to } \mathcal{M}) \\
&\Leftrightarrow (\forall b \in B_{q^M} : b \in Q^{\mathcal{M}}_{on}) \vee i \neq \mathsf{timeout} \\
&\Leftrightarrow q^M \in Q^{\mathsf{MM1T}(M)}_{on} \vee i \neq \mathsf{timeout} && \text{(by Item 5)}
\end{aligned}$$

We can now finish the proof for this rule:

$$\begin{aligned}
&\lambda^M(\delta^M(q^M, i), \mathsf{timeout}) \neq \mathsf{nil} \\
&\Leftrightarrow \lambda^M(\delta^{\mathsf{MM1T}(M)}(q^M, i), \mathsf{timeout}) \neq \mathsf{nil} && \text{(since } (q^M \in Q^{\mathsf{MM1T}(M)}_{on}) \vee (i \neq \mathsf{timeout})) \\
&\Leftrightarrow \delta^{\mathsf{MM1T}(M)}(q^M, i) \in Q^{\mathsf{MM1T}(M)}_{on} && \text{(by definition of } Q^{\mathsf{MM1T}(M)}_{on})
\end{aligned}$$

9. For $\mathsf{MM1T}(M)$, Rule 2.4 starts from the position that:

$$\begin{aligned}
&q^M \in Q^{\mathsf{MM1T}(M)}_{off} \wedge \delta^{\mathsf{MM1T}(M)}(q^M, i) \in Q^{\mathsf{MM1T}(M)}_{on} \\
&\Rightarrow q^M \in Q^{\mathsf{MM1T}(M)}_{off} \wedge \delta^{\mathsf{MM1T}(M)}(q^M, i)\downarrow && (\delta^{\mathsf{MM1T}(M)}(q^M, i) \in Q^{\mathsf{MM1T}(M)}) \\
&\Leftrightarrow \delta^{\mathsf{MM1T}(M)}(q^M, \mathsf{timeout})\uparrow \wedge\ \delta^{\mathsf{MM1T}(M)}(q^M, i)\downarrow && \text{(Rule 2.1 applies to } \mathsf{MM1T}(M)) \\
&\Leftrightarrow i \neq \mathsf{timeout}
\end{aligned}$$

We discuss the right-hand-side of the conjunction of Rule 2.4's initial position separately:

$$\begin{aligned}
&\delta^{\mathsf{MM1T}(M)}(q^M, i) \in Q^{\mathsf{MM1T}(M)}_{on} \\
&\Leftrightarrow \lambda^M(\delta^{\mathsf{MM1T}(M)}(q^M, i), \mathsf{timeout}) \neq \mathsf{nil} && \text{(by definition of } Q^{\mathsf{MM1T}(M)}_{on}) \\
&\Leftrightarrow \lambda^M(\delta^M(q^M, i), \mathsf{timeout}) \neq \mathsf{nil} && \begin{pmatrix}\text{by definition of } \delta^{\mathsf{MM1T}(M)} \\ \text{since } \delta^{\mathsf{MM1T}(M)}(q^M, i)\downarrow\end{pmatrix} \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(\delta^{\mathsf{Mealy}(\mathcal{M})}(b, i), \mathsf{timeout}) \neq \mathsf{nil} && (B_{q^M} \text{ is a partition's block for } q^M) \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathcal{M}}(\delta^{\mathsf{Mealy}(\mathcal{M})}(b, i), \mathsf{timeout})\downarrow && \text{(by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})}) \\
&\Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathcal{M}}(\delta^{\mathcal{M}}(b, i), \mathsf{timeout})\downarrow && \begin{pmatrix}\text{by definition of } \delta^{\mathsf{Mealy}(\mathcal{M})} \\ \text{rule 2.3 applies to } \mathcal{M} \\ i \neq \mathsf{timeout}\end{pmatrix} \\
&\Leftrightarrow \forall b \in B_{q^M} : \delta^{\mathcal{M}}(\delta^{\mathcal{M}}(b, i), \mathsf{timeout})\downarrow && \text{(Rule 2.2 applies to } \mathcal{M}) \\
&\Leftrightarrow \forall b \in B_{q^M} : \delta^{\mathcal{M}}(b, i) \in Q^{\mathcal{M}}_{on} && \text{(Rule 2.1 applies to } \mathcal{M})
\end{aligned}$$

The fact that Rule 2.4 holds for $\mathsf{MM1T}(M)$ follows from:

$$
\begin{aligned}
& q^M \in Q_{\mathit{off}}^{\mathsf{MM1T}(M)} \wedge \delta^{\mathsf{MM1T}(M)}(q^M, i) \in Q_{on}^{\mathsf{MM1T}(M)} \\
& \Leftrightarrow q^M \in Q_{\mathit{off}}^{\mathsf{MM1T}(M)} \wedge \left(\forall b \in B_{q^M} : \delta^{\mathcal{M}}(b, i) \in Q_{on}^{\mathcal{M}}\right) && \text{(the previous discussion)} \\
& \Leftrightarrow \left(\forall b \in B_{q^M} : b \in Q_{\mathit{off}}\mathcal{M}\right) \wedge \left(\forall b \in B_{q^M} : \delta^{\mathcal{M}}(b, i) \in Q_{on}^{\mathcal{M}}\right) && \text{(by Item 6)} \\
& \Leftrightarrow \forall b \in B_{q^M} : \left(b \in Q_{\mathit{off}}^{\mathcal{M}} \wedge \delta^{\mathcal{M}}(b, i) \in Q_{on}^{\mathcal{M}}\right) \\
& \Rightarrow \forall b \in B_{q^M} : \tau^{\mathcal{M}}(b, i)\downarrow && \text{(Rule 2.4 applies to } \mathcal{M}) \\
& \Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(b, i) \in O^{\mathcal{M}} \times \mathbb{N}^{>0} && \text{(by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})}) \\
& \Leftrightarrow \lambda^M(q^M, i) \in O^{\mathcal{M}} \times \mathbb{N}^{>0} && (B_{q^M} \text{ is a partition's block for } q^M) \\
& \Rightarrow \lambda^M(q^M, i) \in O^{\mathsf{MM1T}(M)} \times \mathbb{N}^{>0} && \left(\begin{array}{l}\text{by definition of } \lambda^{\mathsf{MM1T}(M)} \\ \text{since } \lambda^M(q^M, i) \neq \mathsf{nil} \\ \text{since } \lambda^{\mathsf{MM1T}(M)}(q^M, i) = \\ \quad \pi_1(\lambda^M(q^M, i))\end{array}\right) \\
& \Leftrightarrow \tau^{\mathsf{MM1T}(M)}(q^M, i)\downarrow && \text{(by definition of } \tau^{\mathsf{MM1T}(M)})
\end{aligned}
$$

10. For $\mathsf{MM1T}(M)$, Rule 2.5 starts from the position that:

$$
\begin{aligned}
& \delta^{\mathsf{MM1T}(M)}(q^M, \mathsf{timeout}) \in Q_{on}^{\mathsf{MM1T}(M)} \\
& \Rightarrow \delta^{\mathsf{MM1T}(M)}(q^M, \mathsf{timeout})\downarrow && (\delta^{\mathsf{MM1T}(M)}(q^M, \mathsf{timeout}) \in Q^{\mathsf{MM1T}(M)}) \\
& \Leftrightarrow q^M \in Q_{on}^{\mathsf{MM1T}(M)} && \text{(Rule 2.1 applies to } \mathsf{MM1T}(M)) \\
& \Leftrightarrow \forall b \in B_{q^M} : b \in Q_{on}^{\mathcal{M}} && \text{(by Item 5)} \\
& \Leftrightarrow \forall b \in B_{q^M} : \delta^{\mathcal{M}}(b, \mathsf{timeout})\downarrow && \text{(Rule 2.1 applies to } \mathcal{M})
\end{aligned}
$$

The fact that Rule 2.5 holds for $\mathsf{MM1T}(M)$ now follows from:

$$
\begin{aligned}
& \delta^{\mathsf{MM1T}(M)}(q^M, \mathsf{timeout}) \in Q_{on}^{\mathsf{MM1T}(M)} \\
& \Leftrightarrow \lambda^M(\delta^{\mathsf{MM1T}(M)}(q^M, \mathsf{timeout}), \mathsf{timeout}) \neq \mathsf{nil} && \text{(by definition of } Q_{on}^{\mathsf{MM1T}(M)}) \\
& \Leftrightarrow \lambda^M(\delta^M(q^M, \mathsf{timeout}), \mathsf{timeout}) \neq \mathsf{nil} && \left(\begin{array}{l}\text{by definition of } \delta^{\mathsf{MM1T}(M)} \\ \text{since } \delta^{\mathsf{MM1T}(M)}(q^M, \mathsf{timeout})\downarrow\end{array}\right) \\
& \Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(\delta^{\mathsf{Mealy}(\mathcal{M})}(b, \mathsf{timeout}), \mathsf{timeout}) \neq \mathsf{nil} && (B_{q^M} \text{ is a partition's block for } q^M) \\
& \Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(\delta^{\mathcal{M}}(b, \mathsf{timeout}), \mathsf{timeout}) \neq \mathsf{nil} && \left(\begin{array}{l}\text{by definition of } \delta^{\mathsf{Mealy}(\mathcal{M})} \\ \text{the previous discussion}\end{array}\right) \\
& \Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathcal{M}}(\delta^{\mathcal{M}}(b, \mathsf{timeout}), \mathsf{timeout})\downarrow && \text{(by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})}) \\
& \Leftrightarrow \forall b \in B_{q^M} : \delta^{\mathcal{M}}(\delta^{\mathcal{M}}(b, \mathsf{timeout}), \mathsf{timeout})\downarrow && \text{(Rule 2.2 applies to } \mathcal{M}) \\
& \Leftrightarrow \forall b \in B_{q^M} : \delta^{\mathcal{M}}(b, \mathsf{timeout}) \in Q_{on}^{\mathcal{M}} && \text{(Rule 2.1 applies to } \mathcal{M}) \\
& \Rightarrow \forall b \in B_{q^M} : \tau^{\mathcal{M}}(b, \mathsf{timeout})\downarrow && \text{(Rule 2.5 applies to } \mathcal{M}) \\
& \Leftrightarrow \forall b \in B_{q^M} : \lambda^{\mathsf{Mealy}(\mathcal{M})}(b, \mathsf{timeout}) \in O^{\mathcal{M}} \times \mathbb{N}^{>0} && \text{(by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})}) \\
& \Leftrightarrow \lambda^M(q^M, \mathsf{timeout}) \in O^{\mathcal{M}} \times \mathbb{N}^{>0} && (B_{q^M} \text{ is a partition's block for } q^M) \\
& \Rightarrow \lambda^M(q^M, \mathsf{timeout}) \in O^{\mathsf{MM1T}(M)} \times \mathbb{N}^{>0} && \left(\begin{array}{l}\text{by definition of } \lambda^{\mathsf{MM1T}(M)} \\ \text{since } \lambda^M(q^M, \mathsf{timeout}) \neq \mathsf{nil} \\ \text{since } \lambda^{\mathsf{MM1T}(M)}(q^M, \mathsf{timeout}) = \\ \quad \pi_1(\lambda^M(q^M, \mathsf{timeout}))\end{array}\right) \\
& \Leftrightarrow \tau^{\mathsf{MM1T}(M)}(q^M, \mathsf{timeout})\downarrow && \text{(by definition of } \tau^{\mathsf{MM1T}(M)})
\end{aligned}
$$

Model $\mathsf{MM1T}(M)$ thus meets every requirement for being a valid MM1T. □

B.2.2 Proof of Lemma B.2.3

Proof. MM1T $\mathsf{MM1T}(M)$ with set of states Q^M is minimal iff, for all $q_1^M, q_2^M \in Q^M$ for which $q_1^M \neq q_2^M$, $q_1^M \not\approx_{untimed} q_2^M$. Let q_1^M and q_2^M be any two states of $\mathsf{MM1T}(M)$ for which $q_1^M \neq q_2^M$.
We know that Mealy machine M is minimal, and that M has the same set of states as $\mathsf{MM1T}(M)$. This tells us that there exists an input sequence $\sigma \in I^*$ for which:

$$\lambda^{M^*}(q_1^M, \sigma) \neq \lambda^{M^*}(q_2^M, \sigma).$$

We thus know that there is at least one input $i \in I$ and one input sequence $\rho \in I^*$ such that $\sigma = \rho\ i$, and:

$$\lambda^M(q_1', i) \neq \lambda^M(q_2', i), \tag{B.2}$$

where $q_1' = \delta^{M^*}(q_1^M, \rho)$ and $q_2' = \delta^{M^*}(q_2^M, \rho)$. Since M is a minimized version of $\mathsf{Mealy}(\mathcal{M})$, $M \approx_{trace} \mathsf{Mealy}(\mathcal{M})$. This implies that we can compare the outputs of M and $\mathsf{Mealy}(\mathcal{M})$, which tells us that there is a set Ω such that:

$$O^M \subseteq (\Omega \times (\mathbb{N}^{>0} \cup \{\bot\})) \cup \{\mathsf{nil}\}.$$

This tells us that the inequality from Equation (B.2) must be caused by one of the following cases:

1. Either $\lambda^M(q_1', i) = \mathsf{nil}$ and $\lambda^M(q_2', i) \neq \mathsf{nil}$, or $\lambda^M(q_1', i) \neq \mathsf{nil}$ and $\lambda^M(q_2', i) = \mathsf{nil}$.
   If $\lambda^M(q_1', i) = \mathsf{nil}$ and $\lambda^M(q_2', i) \neq \mathsf{nil}$, then $\lambda^{\mathsf{MM1T}(M)}(q_1', i)\uparrow$, per the definition of $\mathsf{MM1T}(M)$. Rule 2.1 tells us that $q_1' \in Q_{off}^{\mathsf{MM1T}(M)}$ and $i = \mathsf{timeout}$. The fact that $\lambda^M(q_2', i) \neq \mathsf{nil}$ tells us that $\lambda^{\mathsf{MM1T}(M)}(q_2', i)\downarrow$.
   Since $i = \mathsf{timeout}$, we know from Rule 2.1 that $q_2' \in Q_{on}^{\mathsf{MM1T}(M)}$. Lemma B.2.1 now tells us that, since $q_1' = \delta^{M^*}(q_1^M, \rho)$, $q_2' = \delta^{M^*}(q_2^M, \rho)$, $q_1' \in Q_{off}^{\mathsf{MM1T}(M)}$ and $q_2' \in Q_{on}^{\mathsf{MM1T}(M)}$, $q_1 \not\approx_{untimed} q_2$.
   We can use a similar argument to show that, if $\lambda^M(q_1', i) \neq \mathsf{nil}$ and $\lambda^M(q_2', i) = \mathsf{nil}$, then $q_1 \not\approx_{untimed} q_2$.

2. If $\lambda^M(q_1', i) \neq \mathsf{nil}$, $\lambda^M(q_2', i) \neq \mathsf{nil}$ and $\pi_1(\lambda^M(q_1', i)) \neq \pi_1(\lambda^M(q_2', i))$, then:

   $$\begin{aligned}
   &\lambda^M(q_1', i) \neq \mathsf{nil} \wedge \lambda^M(q_2', i) \neq \mathsf{nil} \wedge \pi_1(\lambda^M(q_1', i)) \neq \pi_1(\lambda^M(q_2', i)) \\
   &\Rightarrow \lambda^{\mathsf{MM1T}(M)}(q_1', i) \neq \lambda^{\mathsf{MM1T}(M)}(q_2', i) && \text{(by definition of } \lambda^{\mathsf{MM1T}(M)}) \\
   &\Rightarrow uWord_{q_1'}^{\mathsf{MM1T}(M)}(i) \neq uWord_{q_2'}^{\mathsf{MM1T}(M)}(i) && \text{(by definition of } uWord) \\
   &\Rightarrow uWord_{q_1}^{\mathsf{MM1T}(M)}(\rho\ i) \neq uWord_{q_2}^{\mathsf{MM1T}(M)}(\rho\ i) && \text{(by definition of } q_1', q_2' \text{ and } uWord) \\
   &\Rightarrow q_1 \not\approx_{untimed} q_2 && \text{(by definition of untimed equivalence)}
   \end{aligned}$$

3. If $\lambda^M(q_1', i) \neq \mathsf{nil}$, $\lambda^M(q_2', i) \neq \mathsf{nil}$ and $\pi_2(\lambda^M(q_1', i)) \neq \pi_2(\lambda^M(q_2', i))$, then we get a second case distinction:

   - Either $\pi_2(\lambda^M(q_1', i)) \in \mathbb{N}^{>0}$ and $\pi_2(\lambda^M(q_2', i)) = \bot$, or $\pi_2(\lambda^M(q_1', i)) = \bot$ and $\pi_2(\lambda^M(q_2', i)) \in \mathbb{N}^{>0}$.
     If $\pi_2(\lambda^M(q_1', i)) \in \mathbb{N}^{>0}$ and $\pi_2(\lambda^M(q_2', i)) = \bot$, then:

     $$\tau^{\mathsf{MM1T}(M)}(q_1', i) = \pi_2(\lambda^M(q_1', i)), \qquad \text{(by definition of } \tau^{\mathsf{MM1T}(M)})$$

     and:

     $$\tau^{\mathsf{MM1T}(M)}(q_2', i)\uparrow, \qquad \text{(by definition of } \tau^{\mathsf{MM1T}(M)})$$

     which implies that $\tau^{\mathsf{MM1T}(M)}(q_1', i) \neq \tau^{\mathsf{MM1T}(M)}(q_2', i)$. We can now see that:

     $$\begin{aligned}
     &\tau^{\mathsf{MM1T}(M)}(q_1', i) \neq \tau^{\mathsf{MM1T}(M)}(q_2', i) \\
     &\Rightarrow uWord_{q_1'}^{\mathsf{MM1T}(M)}(i) \neq uWord_{q_2'}^{\mathsf{MM1T}(M)}(i) && \text{(by definition of } uWord) \\
     &\Rightarrow uWord_{q_1}^{\mathsf{MM1T}(M)}(\rho\ i) \neq uWord_{q_2}^{\mathsf{MM1T}(M)}(\rho\ i) && \text{(by definition of } q_1', q_2' \text{ and } uWord) \\
     &\Rightarrow q_1 \not\approx_{untimed} q_2 && \text{(by definition of untimed equivalence)}
     \end{aligned}$$

We can use a similar argument to show that, if $\pi_2(\lambda^M(q_1', i)) = \bot$ and $\pi_2(\lambda^M(q_2', i)) \in \mathbb{N}^{>0}$, then $q_1 \not\approx_{untimed} q_2$.

- If $\pi_2(\lambda^M(q_1', i)) \in \mathbb{N}^{>0}$, $\pi_2(\lambda^M(q_2', i)) \in \mathbb{N}^{>0}$ and $\pi_2(\lambda^M(q_1', i)) \neq \pi_2(\lambda^M(q_2', i))$, then:

$$
\begin{aligned}
&\pi_2(\lambda^M(q_1', i)) \neq \pi_2(\lambda^M(q_2', i)) \\
&\Rightarrow \tau^{\mathsf{MM1T}(M)}(q_1', i) \neq \tau^{\mathsf{MM1T}(M)}(q_2', i) && \text{(by definition of } \tau^{\mathsf{MM1T}(M)}\text{)} \\
&\Rightarrow uWord_{q_1'}^{\mathsf{MM1T}(M)}(i) \neq uWord_{q_2'}^{\mathsf{MM1T}(M)}(i) && \text{(by definition of } uWord\text{)} \\
&\Rightarrow uWord_{q_1}^{\mathsf{MM1T}(M)}(\rho\ i) \neq uWord_{q_2}^{\mathsf{MM1T}(M)}(\rho\ i) && \text{(by definition of } q_1', q_2' \text{ and } uWord\text{)} \\
&\Rightarrow q_1 \not\approx_{untimed} q_2 && \text{(by definition of untimed equivalence)}
\end{aligned}
$$

We thus see that in all cases, if $\lambda^M(q_1', i) \neq \mathsf{nil}$, $\lambda^M(q_2', i) \neq \mathsf{nil}$ and $\pi_2(\lambda^M(q_1', i)) \neq \pi_2(\lambda^M(q_2', i))$, then $q_1 \not\approx_{untimed} q_2$.

4. $\lambda^M(q_1', i) \neq \mathsf{nil}$, $\lambda^M(q_2', i) \neq \mathsf{nil}$, $\pi_1(\lambda^M(q_1', i)) \neq \pi_1(\lambda^M(q_2', i))$ and $\pi_2(\lambda^M(q_1', i)) \neq \pi_2(\lambda^M(q_2', i))$, then the conditions of items 2 and 3 both hold. This then implies that $q_1 \not\approx_{untimed} q_2$.

Therefore, $\mathsf{MM1T}(M)$ is minimal. □

B.2.3 Proof of Lemma B.2.4

Proof. We know from the definition of $\mathsf{MM1T}(M)$ that $\mathsf{MM1T}(M)$ has the same set of states Q^M as M, that M and $\mathsf{MM1T}(M)$ have the same initial state q_0^M, and that M and $\mathsf{MM1T}(M)$ have the same set of inputs I^M. We similarly see from the definition of $\mathsf{Mealy}(\mathcal{M})$ that $\mathsf{Mealy}(\mathcal{M})$ and $\mathcal{M}$ have the same set of states $Q^{\mathcal{M}}$, the same initial state $q_{\mathcal{I}}^{\mathcal{M}}$, and the same set of inputs $I^{\mathcal{M}}$.

We prove the lemma by showing by induction on the input sequence that for all input sequences σ:

1. $uWord_{q_0^M}^{\mathsf{MM1T}(M)}(\sigma) = uWord_{q_{\mathcal{I}}^{\mathcal{M}}}^{\mathcal{M}}(\sigma)$,
2. $\delta^{\mathsf{MM1T}(M)^*}(q_0^M, \sigma)\downarrow \Rightarrow \delta^{M^*}(q_0^M, \sigma) = \delta^{\mathsf{MM1T}(M)^*}(q_0^M, \sigma)$, and
3. $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}}, \sigma)\downarrow \Rightarrow \delta^{\mathsf{Mealy}(\mathcal{M})^*}(q_{\mathcal{I}}^{\mathcal{M}}, \sigma) = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}}, \sigma)$.

The inductive proof is as follows:

- Base case: For input sequence ϵ:
  1. $uWord_{q_0^M}^{\mathsf{MM1T}(M)}(\epsilon) = \epsilon = uWord_{q_{\mathcal{I}}^{\mathcal{M}}}^{\mathcal{M}}(\epsilon)$, (by definition of $uWord$)
  2. $\delta^{M^*}(q_0^M, \epsilon) = q_0^M = \delta^{\mathsf{MM1T}(M)^*}(q_0^M, \epsilon)$, and (by definition of δ^*)
  3. $\delta^{\mathsf{Mealy}(\mathcal{M})^*}(q_{\mathcal{I}}^{\mathcal{M}}, \epsilon) = q_{\mathcal{I}}^{\mathcal{M}} = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}}, \epsilon)$. (by definition of δ^*)
- Inductive step case: Let $\sigma = \rho\ i$ for some $\rho \in I^{*M} \cup I^{*\mathsf{Mealy}(\mathcal{M})}$ and $i \in I^M \cup I^{\mathcal{M}}$. We use the induction hypothesis (IH):
  1. $uWord_{q_0^M}^{\mathsf{MM1T}(M)}(\rho) = uWord_{q_{\mathcal{I}}^{\mathcal{M}}}^{\mathcal{M}}(\rho)$,
  2. $\delta^{\mathsf{MM1T}(M)^*}(q_0^M, \rho)\downarrow \Rightarrow \delta^{M^*}(q_0^M, \rho) = \delta^{\mathsf{MM1T}(M)^*}(q_0^M, \rho)$, and
  3. $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}}, \rho)\downarrow \Rightarrow \delta^{\mathsf{Mealy}(\mathcal{M})^*}(q_{\mathcal{I}}^{\mathcal{M}}, \rho) = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}}, \rho)$.

  We perform a case distinction on whether $uWord_{q_0^M}^{\mathsf{MM1T}(M)}(\rho\ i)\downarrow \vee\ uWord_{q_{\mathcal{I}}^{\mathcal{M}}}^{\mathcal{M}}(\rho\ i)\downarrow$:

– if $uWord^{\mathsf{MM1T}(M)}_{q_0^M}(\rho\ i)\downarrow \vee\ uWord^{\mathcal{M}}_{q_{\mathcal{I}}^{\mathcal{M}}}(\rho\ i)\downarrow$, then:

$$
\begin{aligned}
&uWord^{\mathsf{MM1T}(M)}_{q_0^M}(\rho\ i)\downarrow \vee\ uWord^{\mathcal{M}}_{q_{\mathcal{I}}^{\mathcal{M}}}(\rho\ i)\downarrow \\
&\Rightarrow uWord^{\mathsf{MM1T}(M)}_{q_0^M}(\rho)\downarrow \vee\ uWord^{\mathcal{M}}_{q_{\mathcal{I}}^{\mathcal{M}}}(\rho)\downarrow \quad \text{(by definition of } uWord) \\
&\Rightarrow uWord^{\mathsf{MM1T}(M)}_{q_0^M}(\rho)\downarrow \wedge\ uWord^{\mathcal{M}}_{q_{\mathcal{I}}^{\mathcal{M}}}(\rho)\downarrow \quad (IH.1 \Rightarrow (uWord^{\mathsf{MM1T}(M)}_{q_0^M}(\rho)\downarrow \Leftrightarrow uWord^{\mathcal{M}}_{q_{\mathcal{I}}^{\mathcal{M}}}(\rho)\downarrow)) \\
&\Rightarrow \delta^{\mathsf{MM1T}(M)^*}(q_0^M,\rho)\downarrow \wedge\ \delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}},\rho)\downarrow \quad \text{(by definition of } uWord)
\end{aligned}
$$

Let $q^M = \delta^{\mathsf{MM1T}(M)^*}(q_0^M,\rho) \overset{IH.2}{=} \delta^{M^*}(q_0^M,\rho)$, and let $q^{\mathcal{M}} = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}},\rho) \overset{IH.3}{=} \delta^{\mathsf{Mealy}(\mathcal{M})^*}(q_{\mathcal{I}}^{\mathcal{M}},\rho)$. Minimization algorithms always yield minimal versions of their input models that are equivalent to the model that they were given. We may thus assume that $M \approx_{trace} \mathsf{Mealy}(\mathcal{M})$. For M and $\mathsf{Mealy}(\mathcal{M})$ to be trace equivalent implies that:

$$
\begin{aligned}
&\lambda^{M^*}(q_0^M,\rho\ i) = \lambda^{\mathsf{Mealy}(\mathcal{M})^*}(q_{\mathcal{I}}^{\mathcal{M}},\rho\ i) \\
&\Leftrightarrow \lambda^{M^*}(q_0^M,\rho) = \lambda^{\mathsf{Mealy}(\mathcal{M})^*}(q_{\mathcal{I}}^{\mathcal{M}},\rho) \wedge \lambda^M(\delta^{M^*}(q_0^M,\rho),i) = \lambda^{\mathsf{Mealy}(\mathcal{M})}(\delta^{\mathsf{Mealy}(\mathcal{M})^*}(q_{\mathcal{I}}^{\mathcal{M}},\rho),i) \\
&\Leftrightarrow \lambda^{M^*}(q_0^M,\rho) = \lambda^{\mathsf{Mealy}(\mathcal{M})^*}(q_{\mathcal{I}}^{\mathcal{M}},\rho) \wedge \lambda^M(q^M,i) = \lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}},i) \qquad (IH)
\end{aligned}
$$

We now use $\lambda^M(q^M,i) = \lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}},i)$ to show that $\lambda^{\mathsf{MM1T}(M)}(q^M,i) = \lambda^{\mathcal{M}}(q^{\mathcal{M}},i)$:

$$
\begin{aligned}
&\lambda^{\mathsf{MM1T}(M)}(q^M,i) \\
&= \begin{cases} \pi_1(\lambda^M(q^M,i)) & \text{if } \lambda^M(q^M,i) \neq \mathsf{nil} \\ \text{undefined} & \text{otherwise} \end{cases} && \text{(by definition of } \lambda^{\mathsf{MM1T}(M)}) \\
&= \begin{cases} \pi_1(\lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}},i)) & \text{if } \lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}},i) \neq \mathsf{nil} \\ \text{undefined} & \text{otherwise} \end{cases} && \text{(the previous discussion)} \\
&= \begin{cases} \pi_1(\lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}},i)) & \text{if } \lambda^{\mathcal{M}}(q^{\mathcal{M}},i)\downarrow \\ \text{undefined} & \text{otherwise} \end{cases} && \text{(by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})}) \\
&= \begin{cases} \lambda^{\mathcal{M}}(q^{\mathcal{M}},i) & \text{if } \lambda^{\mathcal{M}}(q^{\mathcal{M}},i)\downarrow \\ \text{undefined} & \text{otherwise} \end{cases} && \text{(by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})}) \\
&= \lambda^{\mathcal{M}}(q^{\mathcal{M}},i)
\end{aligned}
$$

Before we can show that $\tau^{\mathsf{MM1T}(M)}(q^M,i) = \tau^{\mathcal{M}}(q^{\mathcal{M}},i)$, we first need to show that:

$$\lambda^M(q^M,i) \in O^{\mathsf{MM1T}(M)} \times \mathbb{N}^{>0} \Leftrightarrow \lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}},i) \in O^{\mathcal{M}} \times \mathbb{N}^{>0}$$

We get the first direction of this biconditional from:

$$
\begin{aligned}
&\lambda^M(q^M,i) \in O^{\mathsf{MM1T}(M)} \times \mathbb{N}^{>0} \\
&\Leftrightarrow \lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}},i) \in O^{\mathsf{MM1T}(M)} \times \mathbb{N}^{>0} && \text{(since } \lambda^M(q^M,i) = \lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}},i)) \\
&\Leftrightarrow \lambda^{\mathcal{M}}(q^{\mathcal{M}},i) \in O^{\mathsf{MM1T}(M)} \wedge \tau^{\mathcal{M}}_{\perp}(q^{\mathcal{M}},i) \in \mathbb{N}^{>0} && \text{(by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})}) \\
&\Rightarrow \lambda^{\mathcal{M}}(q^{\mathcal{M}},i) \in O^{\mathcal{M}} \wedge \tau^{\mathcal{M}}_{\perp}(q^{\mathcal{M}},i) \in \mathbb{N}^{>0} && (\lambda^{\mathcal{M}} \text{ is } \mathcal{M}\text{'s output function}) \\
&\Leftrightarrow \lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}},i) \in O^{\mathcal{M}} \times \mathbb{N}^{>0} && \begin{pmatrix}\text{by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})} \\ \text{since } \lambda^{\mathcal{M}}(q^{\mathcal{M}},i)\downarrow\end{pmatrix}
\end{aligned}
$$

We get the second direction of this biconditional from:

$$
\begin{aligned}
&\lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}}, i) \in O^{\mathcal{M}} \times \mathbb{N}^{>0} \\
&\Leftrightarrow \lambda^{M}(q^{M}, i) \in O^{\mathcal{M}} \times \mathbb{N}^{>0} && \text{(since } \lambda^{M}(q^{M}, i) = \lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}}, i)\text{)} \\
&\Leftrightarrow \pi_1(\lambda^{M}(q^{M}, i)) \in O^{\mathcal{M}} \wedge \pi_2(\lambda^{M}(q^{M}, i)) \in \mathbb{N}^{>0} && \begin{pmatrix}\text{by definition of } \pi_1 \\ \text{by definition of } \pi_2\end{pmatrix} \\
&\Leftrightarrow \lambda^{\mathsf{MM1T}(M)}(q^{M}, i) \in O^{\mathcal{M}} \wedge \pi_2(\lambda^{M}(q^{M}, i)) \in \mathbb{N}^{>0} && \begin{pmatrix}\text{by definition of } \lambda^{\mathsf{MM1T}(M)} \\ \text{since } \lambda^{M}(q^{M}, i) \neq \mathsf{nil}\end{pmatrix} \\
&\Rightarrow \lambda^{\mathsf{MM1T}(M)}(q^{M}, i) \in O^{\mathsf{MM1T}(M)} \wedge \pi_2(\lambda^{M}(q^{M}, i)) \in \mathbb{N}^{>0} && (\lambda^{\mathsf{MM1T}(M)} \text{ is } \mathsf{MM1T}(M)\text{'s output function}) \\
&\Leftrightarrow \pi_1(\lambda^{M}(q^{M}, i)) \in O^{\mathsf{MM1T}(M)} \wedge \pi_2(\lambda^{M}(q^{M}, i)) \in \mathbb{N}^{>0} && \begin{pmatrix}\text{by definition of } \lambda^{\mathsf{MM1T}(M)} \\ \text{since } \lambda^{M}(q^{M}, i) \neq \mathsf{nil}\end{pmatrix} \\
&\Leftrightarrow \lambda^{M}(q^{M}, i) \in O^{\mathsf{MM1T}(M)} \times \mathbb{N}^{>0} && \begin{pmatrix}\text{by definition of } \pi_1 \\ \text{by definition of } \pi_2\end{pmatrix}
\end{aligned}
$$

We can now show that $\tau^{\mathsf{MM1T}(M)}(q^{M}, i) = \tau^{\mathcal{M}}(q^{\mathcal{M}}, i)$:

$$
\begin{aligned}
&\tau^{\mathsf{MM1T}(M)}(q^{M}, i) \\
&= \begin{cases} \pi_2(\lambda^{M}(q^{M}, i)) & \text{if } \lambda^{M}(q^{M}, i) \in O^{\mathsf{MM1T}(M)} \times \mathbb{N}^{>0} \\ \text{undefined} & \text{otherwise} \end{cases} && \text{(by definition of } \tau^{\mathsf{MM1T}(M)}\text{)} \\
&= \begin{cases} \pi_2(\lambda^{M}(q^{M}, i)) & \text{if } \lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}}, i) \in O^{\mathcal{M}} \times \mathbb{N}^{>0} \\ \text{undefined} & \text{otherwise} \end{cases} && \text{(the previous discussion)} \\
&= \begin{cases} \pi_2(\lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}}, i)) & \text{if } \lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}}, i) \in O^{\mathcal{M}} \times \mathbb{N}^{>0} \\ \text{undefined} & \text{otherwise} \end{cases} && \text{(an earlier discussion)} \\
&= \begin{cases} \pi_2(\lambda^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}}, i)) & \text{if } \tau^{\mathcal{M}}(q^{\mathcal{M}}, i)\downarrow \\ \text{undefined} & \text{otherwise} \end{cases} && \text{(by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})}\text{)} \\
&= \begin{cases} \tau^{\mathcal{M}}(q^{\mathcal{M}}, i) & \text{if } \tau^{\mathcal{M}}(q^{\mathcal{M}}, i)\downarrow \\ \text{undefined} & \text{otherwise} \end{cases} && \begin{pmatrix}\text{by definition of } \lambda^{\mathsf{Mealy}(\mathcal{M})} \\ \text{since } \tau^{\mathcal{M}}(q^{\mathcal{M}}, i)\downarrow\end{pmatrix} \\
&= \tau^{\mathcal{M}}(q^{\mathcal{M}}, i)
\end{aligned}
$$

We can now show that:

$$
\begin{aligned}
&uWord^{\mathsf{MM1T}(M)}_{q_0^M}(\rho\ i) \\
&= uWord^{\mathsf{MM1T}(M)}_{q_0^M}(\rho)\ (i, \lambda^{\mathsf{MM1T}(M)}(q^{M}, i), \tau^{\mathsf{MM1T}(M)}_{\perp}(q^{M}, i)) && \text{(by definition of } uWord\text{)} \\
&= uWord^{\mathcal{M}}_{q_{\mathcal{I}}^{\mathcal{M}}}(\rho)\ (i, \lambda^{\mathsf{MM1T}(M)}(q^{M}, i), \tau^{\mathsf{MM1T}(M)}_{\perp}(q^{M}, i)) && (IH) \\
&= uWord^{\mathcal{M}}_{q_{\mathcal{I}}^{\mathcal{M}}}(\rho)\ (i, \lambda^{\mathcal{M}}(q^{\mathcal{M}}, i), \tau^{\mathcal{M}}_{\perp}(q^{\mathcal{M}}, i)) && \text{(the previous discussions)} \\
&= uWord^{\mathcal{M}}_{q_{\mathcal{I}}^{\mathcal{M}}}(\rho\ i) && \text{(by definition of } uWord\text{)}
\end{aligned}
$$

Since $\delta^{\mathsf{MM1T}(M)^*}(q_0^M, \rho\ i)\downarrow$, we need to show that:

$$
\begin{aligned}
&\delta^{\mathsf{MM1T}(M)^*}(q_0^M, \rho\ i) \\
&= \delta^{\mathsf{MM1T}(M)}(\delta^{\mathsf{MM1T}(M)^*}(q_0^M, \rho), i) && \text{(by definition of } \delta^{\mathsf{MM1T}(M)^*}) \\
&= \delta^{\mathsf{MM1T}(M)}(q^M, i) && \text{(by definition of } q^M) \\
&= \begin{cases} \delta^M(q^M, i) & \text{if } q^M \in Q_{on}^{\mathsf{MM1T}(M)} \vee i \neq \mathsf{timeout} \\ \text{undefined} & \text{otherwise} \end{cases} && \text{(by definition of } \delta^{\mathsf{MM1T}(M)}) \\
&= \delta^M(q^M, i) && \begin{pmatrix} \delta^{\mathsf{MM1T}(M)}(q^M, i)\downarrow \\ \text{Rule 2.1 applies to } \mathsf{MM1T}(M) \end{pmatrix} \\
&= \delta^M(\delta^{M^*}(q_0^M, \rho), i) && \text{(by definition of } q^M) \\
&= \delta^{M^*}(q_0^M, \rho\ i) && \text{(by definition of } \delta^{M^*})
\end{aligned}
$$

Since $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}}, \rho\ i)\downarrow$, we need to show that:

$$
\begin{aligned}
&\delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}}, \rho\ i) \\
&= \delta^{\mathcal{M}}(\delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}}, \rho), i) && \text{(by definition of } \delta^{\mathcal{M}^*}) \\
&= \delta^{\mathcal{M}}(q^{\mathcal{M}}, i) && \text{(by definition of } q^{\mathcal{M}}) \\
&= \delta^{\mathsf{Mealy}(\mathcal{M})}(q^{\mathcal{M}}, i) && \begin{pmatrix} \text{by definition of } \delta^{\mathsf{Mealy}(\mathcal{M})} \\ \delta^{\mathcal{M}}(q^{\mathcal{M}}, i)\downarrow \end{pmatrix} \\
&= \delta^{\mathsf{Mealy}(\mathcal{M})}(\delta^{\mathsf{Mealy}(\mathcal{M})^*}(q_{\mathcal{I}}^{\mathcal{M}}, \rho), i) && \text{(by definition of } q^{\mathcal{M}}) \\
&= \delta^{\mathsf{Mealy}(\mathcal{M})^*}(q_{\mathcal{I}}^{\mathcal{M}}, \rho\ i) && \text{(by definition of } \delta^{\mathsf{Mealy}(\mathcal{M})^*})
\end{aligned}
$$

– if $uWord_{q_0^M}^{\mathsf{MM1T}(M)}(\rho\ i)\uparrow \wedge\ uWord_{q_{\mathcal{I}}^{\mathcal{M}}}^{\mathcal{M}}(\rho\ i)\uparrow$, then $uWord_{q_0^M}^{\mathsf{MM1T}(M)}(\rho\ i) = uWord_{q_{\mathcal{I}}^{\mathcal{M}}}^{\mathcal{M}}(\rho\ i)$ trivially holds. We also get that:

$$
\begin{aligned}
&uWord_{q_0^M}^{\mathsf{MM1T}(M)}(\rho\ i)\uparrow \wedge\ uWord_{q_{\mathcal{I}}^{\mathcal{M}}}^{\mathcal{M}}(\rho\ i)\uparrow \\
&\Leftrightarrow \delta^{\mathsf{MM1T}(M)^*}(q_0^M, \rho\ i)\uparrow \wedge\ \delta^{\mathcal{M}^*}(q_{\mathcal{I}}^{\mathcal{M}}, \rho\ i)\uparrow \quad \text{(by definition of } uWord)
\end{aligned}
$$

So we are done.

□

Appendix C

Definitions, Properties and Proofs Related to (g)MMTs

This appendix contains a collection of definitions, properties and proofs that we cut from Chapter 5 because we didn't need them to explain the testing procedure. We included them here, because we use them to prove the validity of our method.

C.1 Proofs Related to Functional Simulations

Proofs for (g)MMT properties related to functional simulations.

C.1.1 Proof of Lemma 5.3.1

Proof. Let $\pi = q_{k-1} \xrightarrow[u_k]{i_k} q_k \xrightarrow{i_{k+1}\dots i_j} q_j$ be any x-spanning sub-run of any run $\rho = q_0 \xrightarrow{i_1\dots i_n} q_n \in \mathit{runs}(\mathcal{T})$. Then:

1. $i_j = \mathsf{to}[x]$, and
2. $\mathit{lastStartedAt}^{\mathcal{T}}_{q_0 \xrightarrow{i_1\dots i_{j-1}} q_{j-1}}(x) = k$.

We can infer the following:

- We know that since π is x-spanning, $i_{k+1}\dots i_{j-1}$ are not $\mathsf{to}[x]$. Therefore, we get from (FMS2) that $f_t(i_{k+1})\dots f_t(i_{j-1})$ are not $\mathsf{to}[f_t(x)]$.
- We get from (FMS1) that since x is active in states $q_k \dots q_{j-1}$, $f_t(x)$ is active in states $f_s(q_k)\dots f_s(q_{j-1})$.
- We get from (FMS3) that since x is (re)started in u_k, $f_t(x)$ is (re)started in $f_u(q_{k-1}, i_k)$.

These three conditions tell us that $\mathit{lastStartedAt}^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(i_1)\dots f_t(i_{j-1})} f_s(q_{j-1})}(f_t(x)) = k$. Since $f_t(i_j) = f_t(\mathsf{to}[x]) = \mathsf{to}[f_t(x)]$, we thus get that $\langle f_s, f_t, f_u\rangle(\pi)$ is $f_t(x)$-spanning. □

C.1.2 Proof of Lemma 5.3.2

Proof. Let $\pi = q_{k-1} \xrightarrow[u_k]{i_k} q_k \xrightarrow{i_{k+1}\dots i_j} q_j$ be any x-spanning sub-run of any run $\rho = q_0 \xrightarrow{i_1\dots i_n} q_n \in \mathit{runs}(\mathcal{T})$. Then:

1. $i_j = \mathsf{to}[x]$, and
2. $\mathit{lastStartedAt}^{\mathcal{T}}_{q_0 \xrightarrow{i_1\dots i_{j-1}} q_{j-1}}(x) = k$.

We can infer the following:

- We know that since π is x-spanning, $i_{k+1} \dots i_{j-1}$ are not $\mathsf{to}[x]$. Therefore, we get from (FGS2):

  $$\forall l \in \{k+1, \dots, j-1\} \colon (f_t(q_{l-1}, i_l) \neq f_t(q_{l-1}, \mathsf{to}[x]) = \mathsf{to}[f_t(q_{l-1}, x)]).$$

- We get from (FGS1) that since x is active in states $q_k \dots q_{j-1}$:

  $$\forall l \in \{k, \dots, j-1\} \colon (f_t(q_l, x) \in \mathcal{X}^{\mathcal{M}}(f_s(q_l))).$$

- We get from (FGS3) that since x is (re)started in u_k, $f_t(q_k, x)$ is (re)started in $f_u(q_{k-1}, i_k)$.

These three conditions tell us that $lastStartedAt^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(q_0,i_1)\dots f_t(q_{j-2},i_{j-1})} f_s(q_{j-1})}(f_t(q_{j-1}, x)) = k$. Since:

$$f_t(q_{j-1}, i_j) = f_t(q_{j-1}, \mathsf{to}[x]) = \mathsf{to}[f_t(q_{j-1}, x)],$$

we thus get that $\langle f_s, f_t, f_u \rangle(\pi)$ is spanning. □

C.1.3 Proof of Lemma 5.3.3

Proof. We use a proof by induction on j:

- Base case: $j = k$, then:

  $$\begin{aligned} &renameTo^{\mathcal{M}}_{f_s(q_k) \xrightarrow{f_t(q_k,i_{k+1})\dots f_t(q_{j-1},i_j)} f_s(q_j)}(f_t(q_k, x)) = renameTo^{\mathcal{M}}_{f_s(q_k)}(f_t(q_k, x)) \\ &= f_t(q_k, x) = f_t(q_j, x), \end{aligned}$$

  as required.

- Inductive step case: $j > k$. We use the induction hypothesis (IH):

  $$renameTo^{\mathcal{M}}_{f_s(q_k) \xrightarrow{f_t(q_k,i_{k+1})\dots f_t(q_{j-1},i_j)} f_s(q_j)}(f_t(q_k, x)) = f_t(q_j, x).$$

  We get:

  $$\begin{aligned} &renameTo^{\mathcal{M}}_{f_s(q_k) \xrightarrow{f_t(q_k,i_{k+1})\dots f_t(q_j,i_{j+1})} f_s(q_{j+1})}(f_t(q_k, x)) \\ &= renameTo^{\mathcal{M}}_{f_s(q_j) \xrightarrow{f_t(q_j,i_{j+1})} f_s(q_{j+1})}(renameTo^{\mathcal{M}}_{f_s(q_k) \xrightarrow{f_t(q_k,i_{k+1})\dots f_t(q_{j-1},i_j)} f_s(q_j)}(f_t(q_k, x))) \\ &= renameTo^{\mathcal{M}}_{f_s(q_j) \xrightarrow{f_t(q_j,i_{j+1})} f_s(q_{j+1})}(f_t(q_j, x)) && (IH) \\ &= f_t(q_{j+1}, x), && \begin{pmatrix}(FGS3) \wedge \\ (FGS4)\end{pmatrix} \end{aligned}$$

  as required.

□

C.1.4 Proof of Lemma 5.3.4

Proof. We use a proof by induction on j:

- Base case: $j = k$, then:

  $$\begin{aligned} &renamesTo^{\mathcal{M}}_{f_s(q_k) \xrightarrow{f_t(q_k,i_{k+1})\dots f_t(q_{j-1},i_j)} f_s(q_j)}(f_t(q_j, x)) = renamesTo^{\mathcal{M}}_{f_s(q_j)}(f_t(q_j, x)) \\ &= f_t(q_j, x) = f_t(q_k, x), \end{aligned}$$

  as required.

- Inductive step case: $j > k$. We use the induction hypothesis (IH):

  $$renamesTo^{\mathcal{M}}_{f_s(q_k) \xrightarrow{f_t(q_k,i_{k+1})\dots f_t(q_{j-1},i_j)} f_s(q_j)}(f_t(q_j,x)) = f_t(q_k,x).$$

  We get:

  $$\begin{aligned}
  &renamesTo^{\mathcal{M}}_{f_s(q_{k-1}) \xrightarrow{f_t(q_{k-1},i_k)\dots f_t(q_{j-1},i_j)} f_s(q_j)}(f_t(q_j,x)) \\
  &\quad = renamesTo^{\mathcal{M}}_{f_s(q_{k-1}) \xrightarrow{f_t(q_{k-1},i_k)} f_s(q_k)}(renamesTo^{\mathcal{M}}_{f_s(q_k) \xrightarrow{f_t(q_k,i_{k+1})\dots f_t(q_{j-1},i_j)} f_s(q_j)}(f_t(q_j,x))) \\
  &\quad = renamesTo^{\mathcal{M}}_{f_s(q_{k-1}) \xrightarrow{f_t(q_{k-1},i_k)} f_s(q_k)}(f_t(q_k,x)) && (IH) \\
  &\quad = f_t(q_{k-1},x), && \begin{pmatrix}(FGS3) \wedge \\ (FGS4)\end{pmatrix}
  \end{aligned}$$

  as required.

□

C.2 Properties and Proofs Related to Observation Tree Runs

This section contains some of the lemmas and proofs concerning runs for the observation tree MMTs from Chapter 5.

Lemma C.2.1. Let $\mathcal{T}$ be an observation tree MMT, and let $x \in X$ be a timer of $\mathcal{T}$. If the runs $\pi = q_0 \xrightarrow{i_1 \dots i_n} q_n$ and $\pi' = q'_0 \xrightarrow{i'_1 \dots i'_n} q'_n$ of $\mathcal{T}$ are matching with $m^{\pi}_{\pi'} : \pi \leftrightarrow \pi'$ and $q_{k-1} \xrightarrow{i_k \dots i_j} q_j$ is an x-spanning sub-run of π, then $q'_{k-1} \xrightarrow{i'_k \dots i'_j} q'_j$ is an $m^{\pi}_{\pi'}(x)$-spanning sub-run of π'.

Proof. Since $q_{k-1} \xrightarrow{i_k \dots i_j} q_j$ is an x-spanning sub-run of π:

1. π has action $i_j = \mathsf{to}[x]$ for the timer $x \in X$, and
2. $lastStartedAt_{q_0 \xrightarrow{i_1 \dots i_{j-1}} q_{j-1}}(x) = k$.

The fact that π and π' are matching tells us that $i'_j = \mathsf{to}[m^{\pi}_{\pi'}(x)]$. For all actions within π''s sub-run $q'_k \xrightarrow{i'_{k+1} \dots i'_{j-1}} q'_{j-1}$, we know that:

- if they are input actions, then if they (re)start a timer y, the fact that $\mathcal{T}$ is an observation tree implies that this is the first transition in which y was started.
  We now show by contradiction that such a timer y cannot be the timer $m^{\pi}_{\pi'}(x)$. The timer x was either already active in π's initial state, or it was started somewhere along π. We perform a case distinction:
  - if x was already active in π's initial state, then $m^{\pi}_{\pi'}(x) = m(x)$ was active in π''s initial state. The idea that $y = m^{\pi}_{\pi'}(x)$ would thus conflict with the fact that y is first started in this later action within $q'_k \xrightarrow{i'_{k+1} \dots i'_{j-1}} q'_{j-1}$. Therefore, in the first case, $y \neq m^{\pi}_{\pi'}(x)$.
  - if x first becomes active in action i_l of π, then we know from the fact that x is (re)started in i_k that $l \leq k$. We know from the fact that $\mathcal{T}$ is an observation tree that $x = x_{q_l}$. The definition of $m^{\pi}_{\pi'}$ now gives us that $m^{\pi}_{\pi'}(x) = m^{\pi}_{\pi'}(x_{q_l}) = x_{q'_l}$. Therefore, $m^{\pi}_{\pi'}(x)$ could not have first become active due to any action other than i'_l. Since $l \leq k$, we know that $m^{\pi}_{\pi'}(x)$ could not have first become active due to an input action within $q'_k \xrightarrow{i'_{k+1} \dots i'_{j-1}} q'_{j-1}$. The idea that $y = m^{\pi}_{\pi'}(x)$ would thus conflict with the fact that y is started within $q'_k \xrightarrow{i'_{k+1} \dots i'_{j-1}} q'_{j-1}$. Therefore, in the second case, $y \neq m^{\pi}_{\pi'}(x)$.

- if they are timeout actions, then:
  - if they are for $m^{\pi}_{\pi'}(x)$, then π must have a timeout for x at the same index. This cannot be the case, since this timeout for x would be within $q_k \xrightarrow{i_{k+1}\dots i_{j+1}} q_{j+1}$, and thus within an x-spanning run.
  - if they are timeout actions for a timer other than $m^{\pi}_{\pi'}(x)$, then they cannot (re)start $m^{\pi}_{\pi'}(x)$, since MMT timeout actions can only restart the timer for which the timeout occurs.

We thus know that $m^{\pi}_{\pi'}(x)$ is not (re)started within $q'_k \xrightarrow{i'_{k+1}\dots i'_{j-1}} q'_{j-1}$.
We need to show that action i'_k (re)starts $m^{\pi}_{\pi'}(x)$. We know that i'_k must either be an input action, in which case $\mathcal{T}$ being an observation tree implies that timer $m^{\pi}_{\pi'}(x)$ is first started in this action, or it is a timeout for timer $m^{\pi}_{\pi'}(x)$. We perform a case distinction:

- In the first case, we know from the fact that π and π' are matching that i_k must be an input action as well. Since $\mathcal{T}$ is an observation tree, this would imply that $x = x_{q_k}$. We then know from the fact that π and π' are matching that $m^{\pi}_{\pi'}(x) = m^{\pi}_{\pi'}(x_{q_k}) = x_{q'_k}$. This then implies that $m^{\pi}_{\pi'}(x)$ has to be started at index k of π', as $\mathcal{T}$ is an observation tree and q'_k is the target state of the transition for that action.

- In the second case, the fact that π and π' are matching implies that $i_k = \mathsf{to}[x]$. Therefore, if timer $m^{\pi}_{\pi'}(x)$ isn't started in action k of π', then the fact that $i'_k = \mathsf{to}[m^{\pi}_{\pi'}(x)]$ implies that $m^{\pi}_{\pi'}(x)$ would have to be started at some point within $q'_k \xrightarrow{i'_{k+1}\dots i'_{j-1}} q'_{j-1}$. But since $\mathcal{T}$ is an observation tree and $m^{\pi}_{\pi'}(x)$ first became active along π, that could only be done by actions that are timeouts for $m^{\pi}_{\pi'}(x)$. The fact that π and π' are matching means that there would then have to be a timeout action for x within $q_k \xrightarrow{i_{k+1}\dots i_{j-1}} q_{j-1}$, which cannot be the case as $q_{k-1} \xrightarrow{i_k \dots i_j} q_j$ is x-spanning. We thus know that in the second case, timer $m^{\pi}_{\pi'}(x)$ must be (re)started in index k of π'.

We already knew that $i'_j = \mathsf{to}[m^{\pi}_{\pi'}(x)]$, and we now also know that $\mathit{lastStartedAt}_{q'_0 \xrightarrow{i'_1 \dots i'_{j-1}} q'_{j-1}}(m^{\pi}_{\pi'}(x)) = k$.

The sub-run $q'_{k-1} \xrightarrow{i'_k \dots i'_j} q'_j$ of π' is therefore $m^{\pi}_{\pi'}(x)$-spanning. □

Lemma C.2.2. Let $\mathcal{T}$ be an observation tree MMT, and let $x \in X$ be a timer of $\mathcal{T}$. If the runs $\pi = q_0 \xrightarrow{i_1 \dots i_n} q_n$ and $\pi' = q'_0 \xrightarrow{i'_1 \dots i'_n} q'_n$ of $\mathcal{T}$ are matching with $m^{\pi}_{\pi'} \colon \pi \leftrightarrow \pi'$. Then:

$$\forall j \in \{1, \dots, n\} \colon \qquad i_j \in TO(X) \quad \iff \quad i'_j \in TO(X).$$

Proof. Let $j \in \{1, \dots, n\}$. We cover both directions of the bi-implication:

- If $i_j \in TO(X)$, then $m^{\pi}_{\pi'} \colon \pi \leftrightarrow \pi'$ tells us that $i'_j \in TO(X)$, as required.
- If $i'_j \in TO(X)$, then we use a proof by contradiction to show that $i_j \in TO(X)$:
  Suppose that $i_j \notin TO(X)$. Then $i_j \in I$. The fact that $m^{\pi}_{\pi'} \colon \pi \leftrightarrow \pi'$ then tells us that $i'_j \in I$, which contradicts $i'_j \in TO(X)$. Therefore, $i_j \in TO(X)$, as required.

□

Lemma C.2.3 (Run matchings for (g)MMT observation trees are injective). Let $\mathcal{T}$ be an observation tree with states $q_0, q'_0 \in Q$, and with a matching $m \colon q_0 \leftrightarrow q'_0$. Let $\pi = q_0 \xrightarrow{i_1 \dots i_n} q_n \in \mathit{runs}(\mathcal{T})$ and $\pi' = q'_0 \xrightarrow{i'_1 \dots i'_n} q'_n \in \mathit{runs}(\mathcal{T})$ be matching runs, with $m^{\pi}_{\pi'} \colon \pi \leftrightarrow \pi'$. Then $m^{\pi}_{\pi'}$ is injective.

Proof. The run matching $m^{\pi}_{\pi'}$ is defined as:

$$m^{\pi}_{\pi'} \coloneqq m \cup \{(x_{q_k}, x_{q'_k}) \mid 0 < k \leq n\}.$$

Let $f = \{(x_{q_k}, x_{q'_k}) \mid 0 < k \leq n\}$. Then $m^\pi_{\pi'} = m \cup f$. Clearly, $\mathsf{dom}(m^\pi_{\pi'}) = \mathsf{dom}(m) \cup \mathsf{dom}(f)$, where $\mathsf{dom}(m) \subseteq \mathcal{X}(q_0)$ and $\mathsf{dom}(f) = \{x_{q_k} \mid 0 < k \leq n\}$. This implies that $\mathsf{dom}(m) \cap \mathsf{dom}(f) = \emptyset$.
Since timer matchings are always injective, m is injective. We can also see that:

$$\forall x, y \in \mathsf{dom}(f)\colon \qquad x \neq y \implies f(x) \neq f(y),$$

which means that f is injective.
Finally, the fact that $m\colon \mathcal{X}(q_0) \rightharpoonup \mathcal{X}(q'_0)$ implies that $m(\mathsf{dom}(m)) \subseteq \mathcal{X}(q'_0)$, while $f(\mathsf{dom}(f)) = \{x_{q'_k} \mid 0 < k \leq n\}$. Thus, $m(\mathsf{dom}(m)) \cap f(\mathsf{dom}(f)) = \emptyset$. This implies that for all $x, y \in \mathsf{dom}(m^\pi_{\pi'})$, if $x \in \mathsf{dom}(m)$ and $y \in \mathsf{dom}(f)$, then $m^\pi_{\pi'}(x) \neq m^\pi_{\pi'}(y)$.

We have shown that $m^\pi_{\pi'}$ is injective, as required. □

Lemma C.2.4. Let $\mathcal{T}$ be an observation tree MMT, and let $x \in X$ be a timer of $\mathcal{T}$. If the runs $\pi = q_0 \xrightarrow{i_1 \dots i_n} q_n$ and $\pi' = q'_0 \xrightarrow{i'_1 \dots i'_n} q'_n$ of $\mathcal{T}$ are matching with $m^\pi_{\pi'}\colon \pi \leftrightarrow \pi'$ and $q'_{k-1} \xrightarrow{i'_k \dots i'_j} q'_j$ is an $m^\pi_{\pi'}(x)$-spanning sub-run of π', then $q_{k-1} \xrightarrow{i_k \dots i_j} q_j$ is an x-spanning sub-run of π.

Proof. Let $q'_{k-1} \xrightarrow{i'_k \dots i'_j} q'_j$ be an $m^\pi_{\pi'}(x)$-spanning sub-run of π'. Lemma C.2.2 tells us that since $i'_j \in TO(X)$, $i_j \in TO(X)$. Let $i_j = \mathsf{to}[y]$. Since observation trees are MMTs, we know that a timeout for a timer y must always terminate a y-spanning run. The fact that $\mathcal{T}$ is tree-shaped implies that for each timeout action in $\mathcal{T}$, there is precisely one spanning terminated by that timeout. Let $q_{k'-1} \xrightarrow{i_{k'} \dots i_j} q_j$ be the y-spanning that terminates in our action i_j of π. Then Lemma C.2.1 tells us that since $m^\pi_{\pi'}\colon \pi \leftrightarrow \pi'$, $q'_{k'-1} \xrightarrow{i'_{k'} \dots i'_j} q'_j$ is $m^\pi_{\pi'}(y)$-spanning. Since $m^\pi_{\pi'}(y) = i'_j = m^\pi_{\pi'}(x)$, $m^\pi_{\pi'}(y) = m^\pi_{\pi'}(x)$. Lemma C.2.3 now tells us that $y = x$, which tells us that $k' = k$. This implies that since $q_{k'-1} \xrightarrow{i'_k \dots i_j} q_j$ is y-spanning, $q_{k-1} \xrightarrow{i_k \dots i_j} q_j$ is x-spanning. Furthermore, since $q'_{k-1} \xrightarrow{i'_k \dots i'_j} q'_j$ is a sub-run of π', index k falls within π'. Index k thus falls within π, which implies that $q_{k-1} \xrightarrow{i_k \dots i_j} q_j$ is an x-spanning sub-run of π, as required. □

C.3 Properties and Proofs Related to Apartness

This section contains some of the lemmas and proofs that are centered around the notions of apartness that Chapter 5 introduced for (observation tree) (g)MMTs.

We also use the following two lemmas:

Lemma C.3.1. Let $\mathcal{T}$ be an observation tree MMT, let $\mathcal{M}$ be an s-learnable MMT, and let $\langle f_s, f_t, f_u \rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional MMT simulation. Let $q_0, q'_0 \in Q^\mathcal{T}$. If $q_0, q'_0 \in \mathcal{A}^\mathcal{T}_\mathcal{M}$, then $q_0 \# q'_0 \Rightarrow f_s(q_0) \neq f_s(q'_0)$.

The proof of Lemma C.3.1 can be found in Appendix C.3.1.

Lemma C.3.2. Let $\mathcal{T}$ be an observation tree MMT, let $\mathcal{M}$ be an s-learnable gMMT, and let $\langle f_s, f_t, f_u \rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Let $q_0, q'_0 \in Q^\mathcal{T}$. If $q_0, q'_0 \in \mathcal{A}^\mathcal{T}_\mathcal{M}$, then $q_0 \# q'_0 \Rightarrow f_s(q_0) \neq f_s(q'_0)$.

The proof of Lemma C.3.2 can be found in Appendix C.3.2.
We use these lemmas in our proof of Theorem 5.8.1, which can be found in Appendix C.6.1.
We use the following lemma in two proofs: one in the current appendix, and one in Appendix C.8.

Lemma C.3.3. Let $\mathcal{T}$ be an observation tree MMT, and let $\mathcal{M}$ be an s-learnable gMMT. Let $q_0, q'_0 \in Q^\mathcal{T}$, with $q_0, q'_0 \in \mathcal{A}^\mathcal{T}_\mathcal{M}$. Let $\langle f_s, f_t, f_u \rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation, for which $|\mathcal{X}^\mathcal{M}(f_s(q_0))| = |\mathcal{X}^\mathcal{M}(f_s(q'_0))|$. Let $\pi = q_0, \xrightarrow{i_1 \dots i_n} q_n \in \mathit{runs}(\mathcal{T})$ and $\pi' = q'_0 \xrightarrow{i'_1 \dots i'_n} q'_n \in \mathit{runs}(\mathcal{T})$. Let $\rho = \langle f_s, f_t, f_u \rangle(\pi)$, $\rho' = \langle f_s, f_t, f_u \rangle(\pi')$, and maximal matching $m'\colon f_s(q_0) \leftrightarrow f_s(q'_0)$ with $m'^\rho_{\rho'}\colon \rho \leftrightarrow \rho'$. Let:

$$m = \{(x, y) \in \mathcal{X}^\mathcal{T}(q_0) \times \mathcal{X}^\mathcal{T}(q'_0) \mid (f_t(q_0, x), f_t(q'_0, y)) \in m'\}.$$

be a maximal matching $m\colon q_0 \leftrightarrow q_0'$ such that $m_{\pi'}^{\pi}\colon \pi \leftrightarrow \pi'$. Then:

$$m'^{\rho}_{\rho'}(f_t(q_l,x),l) = \begin{cases} f_t(q_l', m_{\pi'}^{\pi}(x)) & \text{if } k \geq 0 \\ \text{undefined} & \text{if } k = \bot \end{cases}$$

with $k = \mathit{lastStartedAt}^{\mathcal{M}}_{f_s(q_0)\xrightarrow{f_t(q_0,i_1)\dots f_t(q_{l-1},i_l)}f_s(q_l)}(f_t(q_l,x)) = \mathit{lastStartedAt}^{\mathcal{T}}_{q_0\xrightarrow{i_1\dots i_l}q_l}(x)$.

Proof. For all $x \in \mathcal{X}^{\mathcal{T}}(q_0)$, condition (FGS1) implies that $f_t(q_0,x) \in \mathcal{X}^{\mathcal{M}}(f_s(q_0))$. Now, m' being maximal implies that since $|\mathcal{X}^{\mathcal{M}}(f_s(q_0))| = |\mathcal{X}^{\mathcal{M}}(f_s(q_0'))|$, there is a timer $z \in \mathcal{X}^{\mathcal{M}}(f_s(q_0'))$: $m'(f_t(q_0,x)) = z$. Since $z \in \mathcal{X}^{\mathcal{M}}(f_s(q_0'))$, $q_0' \in \mathcal{A}_{\mathcal{M}}^{\mathcal{T}}$ and (FGS1), we know that there is a timer $y \in \mathcal{X}^{\mathcal{T}}(q_0')$ such that $f_t(q_0',y) = z = m'(f_t(q_0,x))$. So $m'(f_t(q_0,x)) = f_t(q_0',y) = f_t(q_0',m(x))$, for all $x \in \mathcal{X}^{\mathcal{T}}(q_0)$.
Partial function m needs to be injective to be a valid matching. We get from (FGS2) that if $x \neq x'$, then $f_t(q_0,x) \neq f_t(q_0,x')$. We know from the fact that m' is a matching that m' is injective, which tells us that $m'(f_t(q_0,x)) \neq m'(f_t(q_0,x'))$. This tells us that $f_t(q_0',m(x)) \neq f_t(q_0',m(x'))$, which tells us that $m(x) \neq m(x')$. Partial function m is therefore injective, which makes it a valid matching.

Since ρ and ρ' are matching under m', matching m' is extended to ρ and ρ' as:

$$m'^{\rho}_{\rho'}(x,l) = \begin{cases} m'(x) & \text{if } l = 0 \\ \mathit{renameTo}_{f_s(q_0')\xrightarrow{f_t(q_0',i_1')\dots f_t(q_{l-1}',i_l')}f_s(q_l')}(m'(\mathit{renamesTo}_{f_s(q_0)\xrightarrow{f_t(q_0,i_1)\dots f_t(q_{l-1},i_l)}f_s(q_l)}(x))) & \text{if } l > 0 \wedge k = 0 \\ \mathit{renameTo}_{f_s(q_k')\xrightarrow{f_t(q_k',i_{k+1}')\dots f_t(q_{l-1}',i_l')}f_s(q_l')}(\mathit{timerStartedAt}(f_s(q_0') \xrightarrow{f_t(q_0',i_1')\dots f_t(q_{k-1}',i_k')})) & \text{if } l > 0 \wedge k > 0 \\ \text{undefined} & \text{if } k = \bot \end{cases}$$

where $k = \mathit{lastStartedAt}^{\mathcal{M}}_{f_s(q_0)\xrightarrow{f_t(q_0,i_1)\dots f_t(q_{l-1},i_l)}f_s(q_l)}(x)$, and $f_t(q_{j-1},i_j)' = m'^{\rho}_{\rho'}(f_t(q_{j-1},i_j),j-1)$ for every j.
We simplify $m'^{\rho}_{\rho'}$ for timers that are explicitly obtained by applying the timer map f_t to timers from $X^{\mathcal{T}}$.

We can now show that:

$$
m'^{\rho}_{\rho'}(f_t(q_l,x),l) =
$$

$$
\begin{cases}
m'(f_t(q_0,x)) & \text{if } l=0 \\
\mathit{renameTo}_{f_s(q'_0)\xrightarrow{f_t(q'_0,i'_1)\cdots f_t(q'_{l-1},i'_l)}f_s(q'_l)}\ (m'(\mathit{renamesTo}_{f_s(q_0)\xrightarrow{f_t(q_0,i_1)\cdots f_t(q_{l-1},i_l)}f_s(q_l)}\ (f_t(q_l,x)))) & \text{if } l>0\wedge k=0 \\
\mathit{renameTo}_{f_s(q'_k)\xrightarrow{f_t(q'_k,i'_{k+1})\cdots f_t(q'_{l-1},i'_l)}f_s(q'_l)}\ (\mathit{timerStartedAt}(f_s(q'_0)\xrightarrow{f_t(q'_0,i'_1)\cdots f_t(q'_{k-1},i'_k)})) & \text{if } l>0\wedge k>0 \\
\text{undefined} & \text{if } k=\bot
\end{cases}
$$

$$
\begin{cases}
m'(f_t(q_0,x)) & \text{if } l=0 \\
\mathit{renameTo}_{f_s(q'_0)\xrightarrow{f_t(q'_0,i'_1)\cdots f_t(q'_{l-1},i'_l)}f_s(q'_l)}\ (m'(f_t(q_0,x))) & \text{if } l>0\wedge k=0 \\
\mathit{renameTo}_{f_s(q'_k)\xrightarrow{f_t(q'_k,i'_{k+1})\cdots f_t(q'_{l-1},i'_l)}f_s(q'_l)}\ (f_t(q'_k,\mathit{timerStartedAt}(q'_0\xrightarrow{i'_1\ldots i'_k}))) & \text{if } l>0\wedge k>0 \\
\text{undefined} & \text{if } k=\bot
\end{cases}
\quad \begin{pmatrix}(FS3)\wedge\\(FS4)\end{pmatrix}
$$

$$
\begin{cases}
f_t(q'_0,m(x)) & \text{if } l=0 \\
\mathit{renameTo}_{f_s(q'_0)\xrightarrow{f_t(q'_0,i'_1)\cdots f_t(q'_{l-1},i'_l)}f_s(q'_l)}\ (f_t(q'_0,m(x))) & \text{if } l>0\wedge k=0 \\
\mathit{renameTo}_{f_s(q'_k)\xrightarrow{f_t(q'_k,i'_{k+1})\cdots f_t(q'_{l-1},i'_l)}f_s(q'_l)}\ (f_t(q'_k,\mathit{timerStartedAt}(q'_0\xrightarrow{i'_1\ldots i'_k}))) & \text{if } l>0\wedge k>0 \\
\text{undefined} & \text{if } k=\bot
\end{cases}
\quad \begin{pmatrix}m'(f_t(q_0,x))\\=\\f_t(q'_0,m(x))\end{pmatrix}
$$

$$
\begin{cases}
f_t(q'_0,m(x)) & \text{if } l=0 \\
f_t(q'_l,m(x)) & \text{if } l>0\wedge k=0 \\
f_t(q'_l,\mathit{timerStartedAt}(q'_0\xrightarrow{i'_1\ldots i'_k})) & \text{if } l>0\wedge k>0 \\
\text{undefined} & \text{if } k=\bot
\end{cases}
\quad \begin{pmatrix}(FS3)\wedge\\(FS4)\end{pmatrix}
$$

$$
\begin{cases}
f_t(q'_0,m(x)) & \text{if } l=0 \\
f_t(q'_l,m(x)) & \text{if } l>0\wedge k=0 \\
f_t(q'_l,m^{\pi}_{\pi'}(\mathit{timerStartedAt}(q_0\xrightarrow{i_1\ldots i_k}))) & \text{if } l>0\wedge k>0 \\
\text{undefined} & \text{if } k=\bot
\end{cases}
\quad \begin{pmatrix}\text{The timer } x \text{ was either already}\\\text{active in } q_0\text{, or it was started in } \pi\end{pmatrix}
$$

$$
\begin{cases}
f_t(q'_0,m(x)) & \text{if } l=0 \\
f_t(q'_l,m(x)) & \text{if } l>0\wedge k=0 \\
f_t(q'_l,m^{\pi}_{\pi'}(x)) & \text{if } l>0\wedge k>0 \\
\text{undefined} & \text{if } k=\bot
\end{cases}
\quad \begin{pmatrix}\text{The timer } x \text{ was either already}\\\text{active in } q_0\text{, or it was started in } \pi\end{pmatrix}
$$

$$
\begin{cases}
f_t(q'_l,m(x)) & \text{if } l=0\vee k=0 \\
f_t(q'_l,m^{\pi}_{\pi'}(x)) & \text{if } l>0\wedge k>0 \\
\text{undefined} & \text{if } k=\bot
\end{cases}
\quad (l=0\Rightarrow k=0)
$$

$$
\begin{cases}
f_t(q'_l,m^{\pi}_{\pi'}(x)) & \text{if } k\geq 0 \\
\text{undefined} & \text{if } k=\bot
\end{cases}
\quad (m\subseteq m^{\pi}_{\pi'})
$$

where $f_t(q_{j-1}, i_j)' = m'^{\rho}_{\rho'}(f_t(q_{j-1}, i_j), j-1)$ for every j; and:

$$k = lastStartedAt^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(q_0,i_1)\dots f_t(q_{l-1},i_l)} f_s(q_l)}(f_t(q_l, x)) = lastStartedAt^{\mathcal{T}}_{q_0 \xrightarrow{i_1 \dots i_l} q_l}(x),$$

per (FGS5). □

C.3.1 Proof of Lemma C.3.1

Proof. We can conclude that $f_s(q_0) \neq f_s(q'_0)$ if $f_s(q_0) \# f_s(q'_0)$. For $f_s(q_0) \# f_s(q'_0)$ to be the case would mean that for all maximal matchings $m' \colon f_s(q_0) \leftrightarrow f_s(q'_0) \colon (f_s(q_0) \#^{m'} f_s(q'_0))$. For all maximal $m' \colon f_s(q_0) \leftrightarrow f_s(q'_0)$, there would have to exist an action sequence $\sigma' \in (A^{\mathcal{M}})^*$ such that $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$. We will show that $q_0 \# q'_0$ implies that for all maximal $m' \colon f_s(q_0) \leftrightarrow f_s(q'_0)$, there either exists an action sequence that shows that $f_s(q_0) \#^{m'} f_s(q'_0)$, or that $f_s(q_0) \neq f_s(q'_0)$ follows directly from a structural apartness between q_0 and q'_0.

The apartness $q_0 \# q'_0$ may hold as a consequence of either of the following two cases:

- (active sizes): then $q_0, q'_0 \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ and $|\mathcal{X}^{\mathcal{T}}(q_0)| \neq |\mathcal{X}^{\mathcal{T}}(q'_0)|$. Then $q_0 \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{M}}(f_s(q_0))|$, and $q'_0 \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}(q'_0)| = |\mathcal{X}^{\mathcal{M}}(f_s(q'_0))|$. Therefore:

  $$|\mathcal{X}^{\mathcal{M}}(f_s(q_0))| = |\mathcal{X}^{\mathcal{T}}(q_0)| \neq |\mathcal{X}^{\mathcal{T}}(q'_0)| = |\mathcal{X}^{\mathcal{M}}(f_s(q'_0))|.$$

  This then means that $f_s(q_0) \# f_s(q'_0)$.

- (enabled sizes): then $q_0, q'_0 \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ and $|\mathcal{X}^{\mathcal{T}}_0(q_0)| \neq |\mathcal{X}^{\mathcal{T}}_0(q'_0)|$. Then $q_0 \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}_0(q_0)| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q_0))|$, and $q'_0 \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}_0(q'_0)| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q'_0))|$. Therefore:

  $$|\mathcal{X}^{\mathcal{M}}_0(f_s(q_0))| = |\mathcal{X}^{\mathcal{T}}_0(q_0)| \neq |\mathcal{X}^{\mathcal{T}}_0(q'_0)| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q'_0))|.$$

  This then means that $f_s(q_0) \# f_s(q'_0)$.

For the remainder of this proof, we will assume that neither of these conditions for apartness holds, since otherwise there would be nothing more for us to show.
We know from $q_0, q'_0 \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ that, since $q_0 \# q'_0$ is not a consequence of (active sizes): $|\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{T}}(q'_0)|$. Since $q_0, q'_0 \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$, we know that $|\mathcal{X}^{\mathcal{M}}(f_s(q_0))| = |\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{T}}(q'_0)| = |\mathcal{X}^{\mathcal{M}}(f_s(q'_0))|$.

Let $m' \colon f_s(q_0) \leftrightarrow f_s(q'_0)$ be an arbitrary maximal matching between $f_s(q_0)$ and $f_s(q'_0)$. We define the matching $m \colon q_0 \leftrightarrow q'_0$:

$$m = \{(x, y) \in \mathcal{X}^{\mathcal{T}}(q_0) \times \mathcal{X}^{\mathcal{T}}(q'_0) \mid (f_t(x), f_t(y)) \in m'\}.$$

For all $x \in \mathcal{X}^{\mathcal{T}}(q_0)$, condition (FMS1) implies that $f_t(x) \in \mathcal{X}^{\mathcal{M}}(f_s(q_0))$. Now, m' being maximal implies that since $|\mathcal{X}^{\mathcal{M}}(f_s(q_0))| = |\mathcal{X}^{\mathcal{M}}(f_s(q'_0))|$, there is a timer $z \in \mathcal{X}^{\mathcal{M}}(f_s(q'_0))$: $m'(f_t(x)) = z$. Since $z \in \mathcal{X}^{\mathcal{M}}(f_s(q'_0))$, $q'_0 \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ and (FMS1), we know that there is a timer $y \in \mathcal{X}^{\mathcal{T}}(q'_0)$ such that $f_t(y) = z = m'(f_t(x))$. So $m'(f_t(x)) = f_t(y) = f_t(m(x))$, for all $x \in \mathcal{X}^{\mathcal{T}}(q_0)$.
Partial function m needs to be injective to be a valid matching. We get from (FMS2) that if $x \neq x'$, then $f_t(x) \neq f_t(x')$. We know from the fact that m' is a matching that m' is injective, which tells us that $m'(f_t(x)) \neq m'(f_t(x'))$. This tells us that $f_t(m(x)) \neq f_t(m(x'))$, which tells us that $m(x) \neq m(x')$. Partial function m is therefore injective, which makes it a valid matching.

The apartness $q_0 \# q'_0$ implies that there exists an action sequence $\sigma = i_1 \dots i_n$ that is a witness of $q_0 \#^m q'_0$. Lemma 5.5.2 tells us that n never needs to exceed $|Q^{\mathcal{M}}|$. We therefore assume that $n \leq |Q^{\mathcal{M}}|$. We get the runs:

$$\pi = q_0 \xrightarrow{i_1} \dots \xrightarrow[u]{i_n/o} q_n \in runs(\mathcal{T})$$

and:

$$\pi' = q'_0 \xrightarrow{i'_1} \dots \xrightarrow[u']{i'_n/o'} q'_n \in runs(\mathcal{T})$$

with $m^{\pi}_{\pi'} : \pi \leftrightarrow \pi'$. We use f_t to lift σ to $\mathcal{M}$, which yields the action sequence $\sigma' = f_t(i_1) \dots f_t(i_n)$. This gives us the runs:

$$\rho = f_s(q_0) \xrightarrow{f_t(i_1)} \dots \xrightarrow[f_t(u)]{f_t(i_n)/o} f_s(q_n) \in runs(\mathcal{M}),$$

and:

$$\rho' = f_s(q'_0) \xrightarrow{f_t(i'_1)} \dots \xrightarrow[f_t(u')]{f_t(i'_n)/o'} f_s(q'_n) \in runs(\mathcal{M}).$$

The runs ρ and ρ' are matching under m', since for all $j \in \{1, \dots, n\}$:

- If $f_t(i_j) \in I$, then $f_t(i_j) = i_j \in I$. We then know from the definition of $m^{\pi}_{\pi'}$ that $i'_j = m^{\pi}_{\pi'}(i_j) = i_j \in I$. Therefore:

  $$f_t(i'_j) = i'_j = i_j = f_t(i_j),$$

  as required.

- If $f_t(i_j) = f_t(\mathsf{to}[x]) = \mathsf{to}[f_t(x)]$ for some $f_t(x) \in X^{\mathcal{M}}$, then $i_j = \mathsf{to}[x]$. There are two possibilities related to $k = lastStartedAt^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(i_1) \dots f_t(i_{j-1})} f_s(q_{j-1})}(f_t(x))$:

  1. If $k = 0$, then $f_t(x) \in \mathcal{X}^{\mathcal{M}}(f_s(q_0))$. Therefore, since $|\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{M}}(f_s(q_0))|$, we know that $\exists y \in \mathcal{X}^{\mathcal{T}}(q_0) : f_t(y) = f_t(x)$. Condition (FMS2) tells us that $y = x$, which thus implies that $x \in \mathcal{X}^{\mathcal{T}}(q_0)$. The fact that π and π' are matching for m now tells us that $i'_j = \mathsf{to}[m(x)]$. Therefore:

     $$f_t(i'_j) = f_t(\mathsf{to}[m(x)]) = \mathsf{to}[f_t(m(x))] = \mathsf{to}[m'(f_t(x))].$$

     Let $k' = lastStartedAt^{\mathcal{M}}_{f_s(q'_0) \xrightarrow{f_t(i'_1) \dots f_t(i'_{j-1})} f_s(q'_{j-1})}(m'(f_t(x)))$. The fact that $k' = 0$ follows by contradiction:
     Suppose that $k' > 0$. Then $f_s(q'_{k'-1}) \xrightarrow{f_t(i'_{k'}) \dots (f_t(i'_j) = f_t(m(x)))} f_s(q'_j)$ is a spanning run. Then (FMS5) tells us that since $\pi' \in runs(\mathcal{T})$, $q'_{k'-1} \xrightarrow{i'_{k'} \dots i'_j} q'_j$ is $m(x)$-spanning. We therefore know from Lemma C.2.4 that $q_{k'-1} \xrightarrow{i_{k'} \dots i_j} q_j$ is x-spanning. Lemma 5.3.1 now tells us that:

     $$f_s(q_{k'-1}) \xrightarrow{f_t(i_{k'}) \dots f_t(i_j)} f_s(q_j)$$

     is spanning, where $i_j = \mathsf{to}[x]$ tells us that $f_t(x) \in \mathcal{X}^{\mathcal{M}}(f_s(q_{j-1}))$. Therefore:

     $$k = lastStartedAt^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(i_1) \dots f_t(i_{j-1})} f_s(q_{j-1})}(f_t(x)) = k' > 0.$$

     This contradicts $k = 0$. Hence, $k' = 0$, as required.

  2. If $k > 0$, then $f_s(q_{k-1}) \xrightarrow{f_t(i_k) \dots f_t(i_j)} f_s(q_j)$ is an $f_t(x)$-spanning sub-run of ρ. Condition (FMS5) then tells us that $q_{k-1} \xrightarrow{i_k \dots i_j} q_j$ is an x-spanning sub-run of π. We know from Lemma C.2.1 that since π and π' are matching under $m^{\pi}_{\pi'}$, $q'_{k-1} \xrightarrow{i'_k \dots i'_j} q'_j$ is an $m^{\pi}_{\pi'}(x)$-spanning sub-run of π'. Lemma 5.3.1 tells us that therefore, $f_s(q'_{k-1}) \xrightarrow{f_t(i'_k) \dots f_t(i'_j)} f_s(q'_j)$ is a spanning sub-run of ρ', as required.

Since ρ and ρ' are matching under m', we can extend matching m' to ρ and ρ':

$$m'^{\rho}_{\rho'}(x, l) = \begin{cases} m'(x) & \text{if } l = 0 \vee k = 0 \\ \pi_1(\tau^{\mathcal{M}}(f_s(q'_{k-1}), f_t(i_k)')) & \text{if } l > 0 \wedge k > 0 \\ \text{undefined} & \text{if } k = \bot \end{cases}$$

where $k = \mathit{lastStartedAt}^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(i_1)\dots f_t(i_l)} f_s(q_l)}(x)$ and $f_t(i_j)' = m'^{\rho}_{\rho'}(f_t(i_j), j-1)$ for every j. We simplify $m'^{\rho}_{\rho'}$ for timers that are explicitly obtained by applying the timer map f_t to timers from $X^{\mathcal{T}}$. The resulting equation will help us in the proof's final stage. We get:

$$
\begin{aligned}
m'^{\rho}_{\rho'}(f_t(x), l) &= \begin{cases} m'(f_t(x)) & \text{if } l = 0 \vee k = 0 \\ \pi_1(\tau^{\mathcal{M}}(f_s(q'_{k-1}), f_t(i_k)')) & \text{if } l > 0 \wedge k > 0 \\ \text{undefined} & \text{if } k = \bot \end{cases} \\
&= \begin{cases} f_t(m(x)) & \text{if } l = 0 \vee k = 0 \\ f_t(\pi_1(\tau^{\mathcal{T}}(q'_{k-1}, i'_k))) & \text{if } l > 0 \wedge k > 0 \\ \text{undefined} & \text{if } k = \bot \end{cases} && \begin{pmatrix} m'(f_t(x)) = f_t(m(x)) \\ \wedge\, (FMS3) \end{pmatrix} \\
&= \begin{cases} f_t(m(x)) & \text{if } l = 0 \vee k = 0 \\ f_t(m^{\pi}_{\pi'}(x)) & \text{if } l > 0 \wedge k > 0 \\ \text{undefined} & \text{if } k = \bot \end{cases} && \begin{pmatrix} \text{The timer } x \text{ was either already} \\ \text{active in } q_0, \text{ or it was started in } \pi \end{pmatrix} \\
&= \begin{cases} f_t(m^{\pi}_{\pi'}(x)) & \text{if } k \in \mathbb{N} \\ \text{undefined} & \text{if } k = \bot \end{cases} && (m \subseteq m^{\pi}_{\pi'})
\end{aligned}
$$

where $f_t(i_j)' = m'^{\rho}_{\rho'}(f_t(i_j), j-1)$ for every j; and:

$$
k = \mathit{lastStartedAt}^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(i_1)\dots f_t(i_l)} f_s(q_l)}(f_t(x)) = \mathit{lastStartedAt}^{\mathcal{T}}_{q_0 \xrightarrow{i_1 \dots i_l} q_l}(x),
$$

per (FMS5).

We now look into the conditions(s) that make $\sigma \vdash q_0 \#^m q'_0$. We show that either σ' is a witness of $f_s(q) \#^{m'} f_s(q')$, or that $m^{\pi}_{\pi'}$ is invalid and can therefore not be relied upon to say that $f_s(q_0)$ and $f_s(q'_0)$ may be the same state of $\mathcal{M}$:

- The apartness is structural, then we know that $m^{\pi}_{\pi'}$ is invalid in the sense that it matches observation tree timers x and $m^{\pi}_{\pi'}$ that are started in different points along the same run, and that can therefore not represent the same timer of $\mathcal{M}$. Matching m is therefore not a candidate for a matching for which its existence shows that $f_s(q_0)$ and $f_s(q'_0)$ may be the same state of $\mathcal{M}$.
- (outputs): then $o \neq o'$. Then (FMS3) and (FMS4) tell us that:

  $$f_s(q_{n-1}) \xrightarrow{f_t(i_n)/o} \quad \wedge \quad f_s(q'_{n-1}) \xrightarrow{f_t(i'_n)/o'} .$$

  Since $o \neq o'$, we have that $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$.
- (constants): then $u = (x, c)$, $u' = (x', c')$ and $c \neq c'$. Then (FMS3) tells us that $f_t(u) = (f_t(x), c)$ and $f_t(u') = (f_t(x'), c')$. Since $c \neq c'$, we have that $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$.
- (updating): then $q_n, q'_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ and $u = \bot \Leftrightarrow u' \neq \bot$. We perform a case distinction:
  1. $u = \bot \wedge u' = (x', c')$, then (FMS3) tells us that $f_t(u') = (f_t(x'), c') \neq \bot$. The fact that $q_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ tells us that $u = \bot$ implies that $f_t(u) = \bot$.
  2. $u = (x, c) \wedge u' = \bot$, then (FMS3) tells us that $f_t(u) = (f_t(x), c) \neq \bot$. The fact that $q'_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ tells us that $u' = \bot$ implies that $f_t(u') = \bot$.

  In both cases, $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$.

- (active sizes): then $q_n, q'_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ and $|\mathcal{X}^{\mathcal{T}}(q_n)| \neq |\mathcal{X}^{\mathcal{T}}(q'_n)|$. Then $q_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}(q_n)| = |\mathcal{X}^{\mathcal{M}}(f_s(q_n))|$, and $q'_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}(q'_n)| = |\mathcal{X}^{\mathcal{M}}(f_s(q'_n))|$. Therefore:

  $$|\mathcal{X}^{\mathcal{M}}(f_s(q_n))| = |\mathcal{X}^{\mathcal{T}}(q_n)| \neq |\mathcal{X}^{\mathcal{T}}(q'_n)| = |\mathcal{X}^{\mathcal{M}}(f_s(q'_n))|.$$

  This then means that $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$.

- (enabled sizes): then $q_n, q'_n \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ and $|\mathcal{X}^{\mathcal{T}}_0(q_n)| \neq |\mathcal{X}^{\mathcal{T}}_0(q'_n)|$. Then $q_n \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}_0(q_n)| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q_n))|$, and $q'_n \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}_0(q'_n)| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q'_n))|$. Therefore:

  $$|\mathcal{X}^{\mathcal{M}}_0(f_s(q_n))| = |\mathcal{X}^{\mathcal{T}}_0(q_n)| \neq |\mathcal{X}^{\mathcal{T}}_0(q'_n)| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q'_n))|.$$

  This then means that $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$.

- (enabled): then, $q_n, q'_n \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ and $\exists x \in \mathsf{dom}(m^{\pi}_{\pi'})\colon (x \in \mathcal{X}_0(q_n) \Leftrightarrow m^{\pi}_{\pi'}(x) \notin \mathcal{X}_0(q'_n))$. We thus know that for such a timer x, either:

  1. $x \in \mathcal{X}^{\mathcal{T}}_0(q_n) \wedge m^{\pi}_{\pi'}(x) \notin \mathcal{X}_0(q'_n)$, or
  2. $m^{\pi}_{\pi'}(x) \in \mathcal{X}_0(q'_n) \wedge x \notin \mathcal{X}^{\mathcal{T}}_0(q_n)$

  We perform a case distinction:

  1. in the first case, (FMS3) and (FMS4) give us that $x \in \mathcal{X}^{\mathcal{T}}_0(q_n)$ implies that $f_t(x) \in \mathcal{X}^{\mathcal{M}}_0(f_s(q_n))$. Conditions (FMS3) and (FMS4) also imply that, since $q'_n \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$, $m^{\pi}_{\pi'}(x) \notin \mathcal{X}_0(q'_n)$ implies that $f_t(m^{\pi}_{\pi'}(x)) \notin \mathcal{X}_0(f_s(q'_n))$. Thus, since $f_t(m^{\pi}_{\pi'}(x)) = m'^{\rho}_{\rho'}(f_t(x), n)$, we get that $m'^{\rho}_{\rho'}(f_t(x), n) \notin \mathcal{X}_0(f_s(q'_n))$. This then implies that $m'^{\rho}_{\rho'}(f_t(x), n)\downarrow$. We may thus conclude that in the first case, $\exists f_t(x) \in X^{\mathcal{M}}\colon (m'^{\rho}_{\rho'}(f_t(x), n)\downarrow \wedge\ (f_t(x) \in \mathcal{X}^{\mathcal{M}}_0(f_s(q_n)) \wedge m'^{\rho}_{\rho'}(f_t(x), n) \notin \mathcal{X}_0(f_s(q'_n))))$.
  2. in the second case, $x \notin \mathcal{X}^{\mathcal{T}}_0(q_n)$ implies that $f_t(x) \notin \mathcal{X}^{\mathcal{M}}_0(f_s(q_n))$ due to $q_n \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$, (FMS3) and (FMS4). We have that $m^{\pi}_{\pi'}(x) \in \mathcal{X}_0(q'_n)$ implies that $f_t(m^{\pi}_{\pi'}(x)) \in \mathcal{X}_0(f_s(q'_n))$, per (FMS3) and (FMS4). Therefore, since $f_t(m^{\pi}_{\pi'}(x)) = m'^{\rho}_{\rho'}(f_t(x), n)$, we get that $m'^{\rho}_{\rho'}(f_t(x), n) \in \mathcal{X}_0(f_s(q'_n))$. This then implies that $m'^{\rho}_{\rho'}(f_t(x), n)\downarrow$. We may thus conclude that in the second case, $\exists f_t(x) \in X^{\mathcal{M}}\colon (m'^{\rho}_{\rho'}(f_t(x), n)\downarrow \wedge\ (f_t(x) \notin \mathcal{X}^{\mathcal{M}}_0(f_s(q_n)) \wedge m'^{\rho}_{\rho'}(f_t(x), n) \in \mathcal{X}_0(f_s(q'_n))))$.

  Therefore, in all cases, $\exists f_t(x) \in X^{\mathcal{M}}\colon (m'^{\rho}_{\rho'}(f_t(x), n)\downarrow \wedge\ (f_t(x) \in \mathcal{X}^{\mathcal{M}}_0(f_s(q_n)) \Leftrightarrow m'^{\rho}_{\rho'}(f_t(x), n) \notin \mathcal{X}_0(f_s(q'_n))))$, which tells us that $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$.

We may thus conclude that for all maximal matchings $m'\colon f_s(q_0) \leftrightarrow f_s(q'_0)$, $f_s(q_0) \neq f_s(q'_0)$ either follows directly, or $f_s(q_0) \#^{m'} f_s(q'_0)$. Therefore, $f_s(q_0) \neq f_s(q'_0)$, as required. □

C.3.2 Proof of Lemma C.3.2

Proof. We can conclude that $f_s(q_0) \neq f_s(q'_0)$ if $f_s(q_0) \# f_s(q'_0)$. For $f_s(q_0) \# f_s(q'_0)$ to be the case would mean that for all maximal matchings $m'\colon f_s(q_0) \leftrightarrow f_s(q'_0)\colon (f_s(q_0) \#^{m'} f_s(q'_0))$. For all maximal $m'\colon f_s(q_0) \leftrightarrow f_s(q'_0)$, there would have to exist an action sequence $\sigma' \in (A^{\mathcal{M}})^*$ such that $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$. We will show that $q_0 \# q'_0$ implies that for all maximal $m'\colon f_s(q_0) \leftrightarrow f_s(q'_0)$, there either exists an action sequence that shows that $f_s(q_0) \#^{m'} f_s(q'_0)$, or that $f_s(q_0) \neq f_s(q'_0)$ follows directly from a structural apartness between q_0 and q'_0.

The apartness $q_0 \# q'_0$ may hold as a consequence of either of the following two cases:

- (active sizes): then $q_0, q'_0 \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ and $|\mathcal{X}^{\mathcal{T}}(q_0)| \neq |\mathcal{X}^{\mathcal{T}}(q'_0)|$. Then $q_0 \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{M}}(f_s(q_0))|$, and $q'_0 \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}(q'_0)| = |\mathcal{X}^{\mathcal{M}}(f_s(q'_0))|$. Therefore:

  $$|\mathcal{X}^{\mathcal{M}}(f_s(q_0))| = |\mathcal{X}^{\mathcal{T}}(q_0)| \neq |\mathcal{X}^{\mathcal{T}}(q'_0)| = |\mathcal{X}^{\mathcal{M}}(f_s(q'_0))|.$$

  This then means that $f_s(q_0) \# f_s(q'_0)$.

- (enabled sizes): then $q_0, q_0' \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ and $|\mathcal{X}^{\mathcal{T}}_0(q_0)| \neq |\mathcal{X}^{\mathcal{T}}_0(q_0')|$. Then $q_0 \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}_0(q_0)| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q_0))|$, and $q_0' \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}_0(q_0')| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q_0'))|$. Therefore:

  $$|\mathcal{X}^{\mathcal{M}}_0(f_s(q_0))| = |\mathcal{X}^{\mathcal{T}}_0(q_0)| \neq |\mathcal{X}^{\mathcal{T}}_0(q_0')| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q_0'))|.$$

  This then means that $f_s(q_0) \# f_s(q_0')$.

For the remainder of this proof, we will assume that neither of these conditions for apartness holds, since otherwise there would be nothing more for us to show.
We know from $q_0, q_0' \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ that, since $q_0 \# q_0'$ is not a consequence of (active sizes): $|\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{T}}(q_0')|$. Since $q_0, q_0' \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$, we know that $|\mathcal{X}^{\mathcal{M}}(f_s(q_0))| = |\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{T}}(q_0')| = |\mathcal{X}^{\mathcal{M}}(f_s(q_0'))|$.

Let $m'\colon f_s(q_0) \leftrightarrow f_s(q_0')$ be an arbitrary maximal matching between $f_s(q_0)$ and $f_s(q_0')$. We define the matching: $m\colon q_0 \leftrightarrow q_0'$:

$$m = \{(x, y) \in \mathcal{X}^{\mathcal{T}}(q_0) \times \mathcal{X}^{\mathcal{T}}(q_0') \mid (f_t(q_0, x), f_t(q_0', y)) \in m'\}.$$

For all $x \in \mathcal{X}^{\mathcal{T}}(q_0)$, condition (FGS1) implies that $f_t(q_0, x) \in \mathcal{X}^{\mathcal{M}}(f_s(q_0))$. Now, m' being maximal implies that since $|\mathcal{X}^{\mathcal{M}}(f_s(q_0))| = |\mathcal{X}^{\mathcal{M}}(f_s(q_0'))|$, there is a timer $z \in \mathcal{X}^{\mathcal{M}}(f_s(q_0'))$: $m'(f_t(q_0, x)) = z$. Since $z \in \mathcal{X}^{\mathcal{M}}(f_s(q_0'))$, $q_0' \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ and (FGS1), we know that there is a timer $y \in \mathcal{X}^{\mathcal{T}}(q_0')$ such that $f_t(q_0', y) = z = m'(f_t(q_0, x))$. So $m'(f_t(q_0, x)) = f_t(q_0', y) = f_t(q_0', m(x))$, for all $x \in \mathcal{X}^{\mathcal{T}}(q_0)$.
Partial function m needs to be injective to be a valid matching. We get from (FGS2) that if $x \neq x'$, then $f_t(q_0, x) \neq f_t(q_0, x')$. We know from the fact that m' is a matching that m' is injective, which tells us that $m'(f_t(q_0, x)) \neq m'(f_t(q_0, x'))$. This tells us that $f_t(q_0', m(x)) \neq f_t(q_0', m(x'))$, which tells us that $m(x) \neq m(x')$. Partial function m is therefore injective, which makes it a valid matching.

The apartness $q_0 \# q_0'$ implies that there exists an action sequence $\sigma = i_1 \dots i_n \in (A^{\mathcal{T}})^*$ that is a witness of $q_0 \#^m q_0'$. Lemma 5.5.2 tells us that n never needs to exceed $|Q^{\mathcal{M}}|$. We therefore assume that $n \leq |Q^{\mathcal{M}}|$. We get the runs:

$$\pi = q_0 \xrightarrow{i_1} \dots \xrightarrow[u]{i_n/o} q_n \in runs(\mathcal{T})$$

and:

$$\pi' = q_0' \xrightarrow{i_1'} \dots \xrightarrow[u']{i_n'/o'} q_n' \in runs(\mathcal{T}),$$

with $m^{\pi}_{\pi'}\colon \pi \leftrightarrow \pi'$. We use f_t to lift σ to $\mathcal{M}$, which yields the action sequence $\sigma' = f_t(q_0, i_1) \dots f_t(q_{n-1}, i_n)$. This gives us the runs:

$$\rho = f_s(q_0) \xrightarrow{f_t(q_0, i_1)} \dots \xrightarrow[f_t(q_{n-1}, q_n, u)]{f_t(q_{n-1}, i_n)/o} f_s(q_n) \in runs(\mathcal{M}),$$

and:

$$\rho' = f_s(q_0') \xrightarrow{f_t(q_0', i_1')} \dots \xrightarrow[f_t(q_{n-1}', q_n', u')]{f_t(q_{n-1}', i_n')/o'} f_s(q_n') \in runs(\mathcal{M}).$$

The runs ρ and ρ' are matching under m', since for all $j \in \{1, \dots, n\}$:

- If $f_t(q_{j-1}, i_j) \in I$, then $f_t(q_{j-1}, i_j) = i_j \in I$. We then know from the definition of $m^{\pi}_{\pi'}$ that $i_j' = m^{\pi}_{\pi'}(i_j) = i_j \in I$. Therefore:

  $$f_t(q_{j-1}', i_j') = i_j' = i_j = f_t(q_{j-1}, i_j),$$

  as required.

- If $f_t(q_{j-1}, i_j) = f_t(q_{j-1}, \mathsf{to}[x]) = \mathsf{to}[f_t(q_{j-1}, x)]$ for some $f_t(q_{j-1}, x) \in X^{\mathcal{M}}$, then $i_j = \mathsf{to}[x]$. There are two possibilities related to $k = lastStartedAt^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(q_0, i_1) \dots f_t(q_{j-2}, i_{j-1})} f_s(q_{j-1})}(f_t(q_{j-1}, x))$:

1. If $k = 0$, then $renamesTo^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(q_0,i_1)\dots f_t(q_{j-2},i_{j-1})} f_s(q_{j-1})}(f_t(q_{j-1}, x)) \in \mathcal{X}^{\mathcal{M}}(f_s(q_0))$. Therefore, since $|\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{M}}(f_s(q_0))|$, we know that:

   $$\exists y \in \mathcal{X}^{\mathcal{T}}(q_0)\colon f_t(q_0, y) = renamesTo^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(q_0,i_1)\dots f_t(q_{j-2},i_{j-1})} f_s(q_{j-1})}(f_t(q_{j-1}, x)) \in \mathcal{X}^{\mathcal{M}}(f_s(q_0)).$$

   Condition (FGS2) tells us that $y = x$, which thus implies that $x \in \mathcal{X}^{\mathcal{T}}(q_0)$. The fact that π and π' are matching for m now tells us that $i'_j = \mathsf{to}[m(x)]$. Therefore:

   $$\begin{aligned}
   f_t(q'_{j-1}, i'_j) &= f_t(q'_{j-1}, \mathsf{to}[m(x)]) \\
   &= \mathsf{to}[f_t(q'_{j-1}, m(x))] \\
   &= \mathsf{to}[renameTo^{\mathcal{M}}_{f_s(q'_0) \xrightarrow{f_t(q'_0,i'_1)\dots f_t(q_{j-2},i'_{j-1})} f_s(q'_{j-1})}(f_t(q'_0, m(x)))] && \text{(Lemma 5.3.3)} \\
   &= \mathsf{to}[renameTo^{\mathcal{M}}_{f_s(q'_0) \xrightarrow{f_t(q'_0,i'_1)\dots f_t(q_{j-2},i'_{j-1})} f_s(q'_{j-1})}(m'(f_t(q_0, x)))] && \begin{pmatrix} m'(f_t(q_0,x)) \\ f_t(q'_0, m(x)) \end{pmatrix} = \\
   &= \mathsf{to}[renameTo^{\mathcal{M}}_{f_s(q'_0) \xrightarrow{f_t(q'_0,i'_1)\dots f_t(q_{j-2},i'_{j-1})} f_s(q'_{j-1})}(\dots \\
   &\quad \dots\ m'(renamesTo^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(q_0,i_1)\dots f_t(q_{j-2},i_{j-1})} f_s(q_{j-1})}(f_t(q_{j-1}, x))))]. && \text{(Lemma 5.3.4)}
   \end{aligned}$$

   Let $k' = lastStartedAt^{\mathcal{M}}_{f_s(q'_0) \xrightarrow{f_t(q'_0,i'_1)\dots f_t(q'_{j-2},i'_{j-1})} f_s(q'_{j-1})}(m'(f_t(q'_{j-1}, x)))$. The fact that $k' = 0$ follows by contradiction:

   Suppose that $k' > 0$. Then $f_s(q'_{k'-1}) \xrightarrow{f_t(q'_{k'-1},i'_{k'})\dots(f_t(q'_{j-1},i'_j)=f_t(q'_{j-1},m(x)))} f_s(q'_j)$ is a spanning run. Then (FGS5) tells us that since $\pi' \in runs(\mathcal{T})$, $q'_{k'-1} \xrightarrow{i'_{k'}\dots i'_j} q'_j$ is $m(x)$-spanning. We therefore know from Lemma C.2.4 that $q_{k'-1} \xrightarrow{i_{k'}\dots i_j} q_j$ is x-spanning. Lemma 5.3.2 now tells us that $f_s(q_{k'-1}) \xrightarrow{f_t(q_{k'-1},i_{k'})\dots f_t(q_{j-1},i_j)} f_s(q_j)$ is spanning, where $i_j = \mathsf{to}[x]$ tells us that $f_t(q_{j-1}, x) \in \mathcal{X}^{\mathcal{M}}(f_s(q_{j-1}))$. Therefore:

   $$k = lastStartedAt^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(q_0,i_1)\dots f_t(q_{j-2},i_{j-1})} f_s(q_{j-1})}(f_t(q_{j-1}, x)) = k' > 0.$$

   This contradicts $k = 0$. Hence, $k' = 0$, as required.

2. If $k > 0$, then $f_s(q_{k-1}) \xrightarrow{f_t(q_{k-1},i_k)\dots f_t(q_{j-1},i_j)} f_s(q_j)$ is a spanning sub-run of ρ. Condition (FGS5) then tells us that $q_{k-1} \xrightarrow{i_k\dots i_j} q_j$ is an x-spanning sub-run of π. We know from Lemma C.2.1 that since π and π' are matching under $m^{\pi}_{\pi'}$, $q'_{k-1} \xrightarrow{i'_k\dots i'_j} q'_j$ is an $m^{\pi}_{\pi'}(x)$-spanning sub-run of π'. Lemma 5.3.2 tells us that therefore, $f_s(q'_{k-1}) \xrightarrow{f_t(i'_k)\dots f_t(i'_j)} f_s(q'_j)$ is a spanning sub-run of ρ', as required.

Since ρ and ρ' are matching under m', we can extend matching m' to ρ and ρ':

$$m'^{\rho}_{\rho'}(x, l) = \begin{cases}
m'(x) & \text{if } l = 0 \\
renameTo_{f_s(q'_0) \xrightarrow{f_t(q'_0,i'_1)\dots f_t(q'_{l-1},i'_l)} f_s(q'_l)}(m'(renamesTo_{f_s(q_0) \xrightarrow{f_t(q_0,i_1)\dots f_t(q_{l-1},i_l)} f_s(q_l)}(x))) & \text{if } l > 0 \wedge k = 0 \\
renameTo_{f_s(q'_k) \xrightarrow{f_t(q'_k,i'_{k+1})\dots f_t(q'_{l-1},i'_l)} f_s(q'_l)}(timerStartedAt(f_s(q'_0) \xrightarrow{f_t(q'_0,i'_1)\dots f_t(q'_{k-1},i'_k)})) & \text{if } l > 0 \wedge k > 0 \\
\text{undefined} & \text{if } k = \bot
\end{cases}$$

where $k = \mathit{lastStartedAt}^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(q_0,i_1)\dots f_t(q_{l-1},i_l)} f_s(q_l)}(x)$, and $f_t(q_{j-1}, i_j)' = m'^{\rho}_{\rho'}(f_t(q_{j-1}, i_j), j-1)$ for every j.

We simplify $m'^{\rho}_{\rho'}$ for timers that are explicitly obtained by applying the timer map f_t to timers from $X^{\mathcal{T}}$. The resulting equation will help us in the proof's final stage. We get from Lemma C.3.3 that since $|f_s(q_0)| = |f_s(q'_0)|$:

$$m'^{\rho}_{\rho'}(f_t(q_l, x), l) = \begin{cases} f_t(q'_l, m^{\pi}_{\pi'}(x)) & \text{if } k \geq 0 \\ \text{undefined} & \text{if } k = \bot \end{cases}$$

where $f_t(q_{j-1}, i_j)' = m'^{\rho}_{\rho'}(f_t(q_{j-1}, i_j), j-1)$ for every j; and:

$$k = \mathit{lastStartedAt}^{\mathcal{M}}_{f_s(q_0) \xrightarrow{f_t(q_0,i_1)\dots f_t(q_{l-1},i_l)} f_s(q_l)}(f_t(q_l, x)) = \mathit{lastStartedAt}^{\mathcal{T}}_{q_0 \xrightarrow{i_1 \dots i_l} q_l}(x),$$

per (FGS5).

We now look into the conditions(s) that make $\sigma \vdash q_0 \#^m q'_0$. We show that either σ' is a witness of $f_s(q) \#^{m'} f_s(q')$, or that $m^{\pi}_{\pi'}$ is invalid and can therefore not be relied upon to say that $f_s(q_0)$ and $f_s(q'_0)$ may be the same state of $\mathcal{M}$:

- The apartness is structural, then we know that $m^{\pi}_{\pi'}$ is invalid in the sense that it matches observation tree timers x and $m^{\pi}_{\pi'}$ that are started in different points along the same run, and that can therefore not represent the same timer of $\mathcal{M}$. Matching m is therefore not a candidate for a matching for which its existence shows that $f_s(q_0)$ and $f_s(q'_0)$ may be the same state of $\mathcal{M}$.
- (outputs): then $o \neq o'$. Then (FGS3) and (FGS4) tell us that:

  $$f_s(q_{n-1}) \xrightarrow{f_t(q_{n-1},i_n)/o} \quad \wedge \quad f_s(q'_{n-1}) \xrightarrow{f_t(q'_{n-1},i'_n)/o'} .$$

  Since $o \neq o'$, we have that $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$.
- (constants): then $u = (x, c)$, $u' = (x', c')$ and $c \neq c'$. Then (FGS3) implies $f_t(q_{n-1}, q_n, u)(f_t(q_n, x)) = c$ and $f_t(q'_{n-1}, q'_n, u')(f_t(q'_n, x')) = c'$. Since $c \neq c'$, we have that $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$.
- (updating): then $q_n, q'_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ and $u = \bot \Leftrightarrow u' \neq \bot$. We perform a case distinction:
  1. $u = \bot \wedge u' = (x', c')$, then (FGS3) tells us that $f_t(q'_{n-1}, q'_n, u')(f_t(q'_n, x')) = c' \in \mathbb{N}^{>0}$. The fact that $q_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ tells us that $u = \bot$ implies that $\neg \exists x \in \mathcal{X}^{\mathcal{M}}(q_n)\colon f_t(q_{n-1}, q_n, u)(f_t(q_n, x)) \in \mathbb{N}^{>0}$.
  2. $u = (x, c) \wedge u' = \bot$, then (FGS3) tells us that $f_t(q_{n-1}, q_n, u)(f_t(q_n, x)) = c \in \mathbb{N}^{>0}$. The fact that $q'_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ tells us that $u' = \bot$ implies that $\neg \exists x' \in \mathcal{X}^{\mathcal{M}}(q'_n)\colon f_t(q'_{n-1}, q'_n, u')(f_t(q'_n, x')) \in \mathbb{N}^{>0}$.

  In both cases, $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$.
- (active sizes): then $q_n, q'_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ and $|\mathcal{X}^{\mathcal{T}}(q_n)| \neq |\mathcal{X}^{\mathcal{T}}(q'_n)|$. Then $q_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}(q_n)| = |\mathcal{X}^{\mathcal{M}}(f_s(q_n))|$, and $q'_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}(q'_n)| = |\mathcal{X}^{\mathcal{M}}(f_s(q'_n))|$. Therefore:

  $$|\mathcal{X}^{\mathcal{M}}(f_s(q_n))| = |\mathcal{X}^{\mathcal{T}}(q_n)| \neq |\mathcal{X}^{\mathcal{T}}(q'_n)| = |\mathcal{X}^{\mathcal{M}}(f_s(q'_n))|.$$

  This then means that $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$.
- (enabled sizes): then $q_n, q'_n \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ and $|\mathcal{X}^{\mathcal{T}}_0(q_n)| \neq |\mathcal{X}^{\mathcal{T}}_0(q'_n)|$. Then $q_n \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}_0(q_n)| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q_n))|$, and $q'_n \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}_0(q'_n)| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q'_n))|$. Therefore:

  $$|\mathcal{X}^{\mathcal{M}}_0(f_s(q_n))| = |\mathcal{X}^{\mathcal{T}}_0(q_n)| \neq |\mathcal{X}^{\mathcal{T}}_0(q'_n)| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q'_n))|.$$

  This then means that $\sigma' \vdash f_s(q_0) \#^{m'} f_s(q'_0)$.
- (enabled) then $q_n, q'_n \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ and $\exists x \in \mathsf{dom}(m^{\pi}_{\pi'})\colon (x \in \mathcal{X}_0(q_n) \Leftrightarrow m^{\pi}_{\pi'}(x) \notin \mathcal{X}_0(q'_n))$. We thus know that for such a timer x, either:

1. $x \in \mathcal{X}_0^{\mathcal{T}}(q_n) \wedge m_{\pi'}^{\pi}(x) \notin \mathcal{X}_0(q'_n)$, or
2. $m_{\pi'}^{\pi}(x) \in \mathcal{X}_0(q'_n) \wedge x \notin \mathcal{X}_0^{\mathcal{T}}(q_n)$

We perform a case distinction:

1. in the first case, (FGS3) and (FGS4) give us that $x \in \mathcal{X}_0^{\mathcal{T}}(q_n)$ implies that $f_t(q_n, x) \in \mathcal{X}_0^{\mathcal{M}}(f_s(q_n))$. Conditions (FGS3) and (FGS4) also imply that, since $q'_n \in \mathcal{E}_{\mathcal{M}}^{\mathcal{T}}$, $m_{\pi'}^{\pi}(x) \notin \mathcal{X}_0(q'_n)$ implies that $f_t(q'_n, m_{\pi'}^{\pi}(x)) \notin \mathcal{X}_0(f_s(q'_n))$. Therefore, since $f_t(q'_n, m_{\pi'}^{\pi}(x)) = m_{\rho'}^{\prime\rho}(f_t(q_n, x), n)$, we get that $m_{\rho'}^{\prime\rho}(f_t(q_n, x), n) \notin \mathcal{X}_0(f_s(q'_n))$. This then implies that $m_{\rho'}^{\prime\rho}(f_t(q_n, x), n)\downarrow$. We may thus conclude that in the first case, $\exists f_t(q_n, x) \in X^{\mathcal{M}} \colon (m_{\rho'}^{\prime\rho}(f_t(q_n, x), n)\downarrow \wedge \ (f_t(q_n, x) \in \mathcal{X}_0^{\mathcal{M}}(f_s(q_n)) \wedge m_{\rho'}^{\prime\rho}(f_t(q_n, x), n) \notin \mathcal{X}_0(f_s(q'_n))))$.
2. in the second case, $x \notin \mathcal{X}_0^{\mathcal{T}}(q_n)$ implies that $f_t(q_n, x) \notin \mathcal{X}_0^{\mathcal{M}}(f_s(q_n))$ due to $q_n \in \mathcal{E}_{\mathcal{M}}^{\mathcal{T}}$, (FGS3) and (FGS4). We have that $m_{\pi'}^{\pi}(x) \in \mathcal{X}_0(q'_n)$ implies that $f_t(q'_n, m_{\pi'}^{\pi}(x)) \in \mathcal{X}_0(f_s(q'_n))$, per (FGS3) and (FGS4). Therefore, since $f_t(q'_n, m_{\pi'}^{\pi}(x)) = m_{\rho'}^{\prime\rho}(f_t(q_n, x), n)$, we get that $m_{\rho'}^{\prime\rho}(f_t(q_n, x), n) \in \mathcal{X}_0(f_s(q'_n))$. This then implies that $m_{\rho'}^{\prime\rho}(f_t(q_n, x), n)\downarrow$. We may thus conclude that in the second case, $\exists f_t(q_n, x) \in X^{\mathcal{M}} \colon (m_{\rho'}^{\prime\rho}(f_t(q_n, x), n)\downarrow \wedge \ (f_t(q_n, x) \notin \mathcal{X}_0^{\mathcal{M}}(f_s(q_n)) \wedge m_{\rho'}^{\prime\rho}(f_t(q_n, x), n) \in \mathcal{X}_0(f_s(q'_n))))$.

Therefore, in all cases:

$$\exists f_t(q_n, x) \in X^{\mathcal{M}} \colon (m_{\rho'}^{\prime\rho}(f_t(q_n, x), n)\downarrow \wedge \ (f_t(q_n, x) \in \mathcal{X}_0^{\mathcal{M}}(f_s(q_n)) \Leftrightarrow m_{\rho'}^{\prime\rho}(f_t(q_n, x), n) \notin \mathcal{X}_0(f_s(q'_n)))),$$

which tells us that $\sigma' \vdash f_s(q_0) \ \#^{m'} \ f_s(q'_0)$.

We may thus conclude that for all maximal matchings $m' \colon f_s(q_0) \leftrightarrow f_s(q'_0)$, $f_s(q_0) \neq f_s(q'_0)$ either follows directly, or $f_s(q_0) \ \#^{m'} \ f_s(q'_0)$. Therefore, $f_s(q_0) \neq f_s(q'_0)$, as required. □

C.4 Properties and Proofs for the Algorithm for Making MMTs t-Observable

This appendix contains many of the properties and proofs for the properties of Section 5.2.

We use the following auxiliary lemmas:

Lemma C.4.1. Algorithm 8's state map $f \colon P^{\mathcal{N}} \to Q^{\mathcal{M}}$ is bijective.

Proof. On Line 9, Algorithm 8 maps $\mathcal{N}$'s initial state $p_{\mathcal{I}}^{\mathcal{N}}$ to $\mathcal{M}$'s initial state $q_{\mathcal{I}}$. On lines 10 through 14, every state $q \in Q^{\mathcal{M}}$ is mapped to by a fresh state p that is added to $\mathcal{N}$'s state set $P^{\mathcal{N}}$. The state sets $Q^{\mathcal{M}}$ and $P^{\mathcal{N}}$ remain untouched after Line 14. State map f is therefore a bijection between $Q^{\mathcal{M}}$ and $P^{\mathcal{N}}$, from Line 14 onwards. □

We can also see that state map f preserves initial states:

Lemma C.4.2. Let $\mathcal{M}$ be an MMT. Then Algorithm 8 yields for $\mathcal{M}$ an MMT $\mathcal{N}$ such that, for its state map $f \colon Q^{\mathcal{M}} \to P^{\mathcal{N}}$:

$$f(p_{\mathcal{I}}^{\mathcal{N}}) = q_{\mathcal{I}}.$$

Proof. This follows directly from Line 9 of Algorithm 8. □

Figure C.1 shows a diagram that illustrates the correspondence between the transitions of the MMTs that are passed to Algorithm 8, and those from the MMTs that are returned by Algorithm 8 (apart from the timer updates). We can prove that this diagram commutes:

Lemma C.4.3. The diagram of Figure C.1 is commutative. Formally, this means that for any MMT $\mathcal{M}$, Algorithm 8 yields an MMT $\mathcal{N}$ such that, for its state map $f \colon P^{\mathcal{N}} \to Q^{\mathcal{M}}$:

$$\forall p, p' \in P^{\mathcal{N}}, i \in A^{\mathcal{N}}, o \in O^{\mathcal{N}} \colon \qquad p \xrightarrow{i/o} p' \quad \Longleftrightarrow \quad f(p) \xrightarrow{i/o} f(p').$$

$$\begin{array}{ccc} p & \xrightarrow{i/o} & p' \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f} \\ f(p) & \xrightarrow{i/o} & f(p') \end{array}$$

Figure C.1: A commutative diagram for the state map $f\colon P^{\mathcal{N}} \to Q^{\mathcal{M}}$ used by Algorithm 8. See Lemma C.4.3 for a proof of commutativity.

The proof of Lemma C.4.3 can be found in Appendix C.4.1. Intuitively, f preserves the transition structure. Since f also preserves the initial states per Lemma C.4.2, we can prove that the MMTs passed to and returned by Algorithm 8 accept the exact same action sequences. We also prove that, for all action sequences, both MMTs yield the same outputs and their states are always matched by f:

Lemma C.4.4. Let $\mathcal{M}$ be an MMT. Then Algorithm 8 yields for $\mathcal{M}$ an MMT $\mathcal{N}$ such that, for its state map $f\colon Q^{\mathcal{M}} \to P^{\mathcal{N}}$:

$$\begin{aligned} &\forall \sigma \in (A^{\mathcal{N}})^*\colon \\ &\quad \delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma)\downarrow \Leftrightarrow \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma)\downarrow \wedge \\ &\quad \lambda^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma) = \lambda^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma) \wedge \\ &\quad (\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma)\downarrow \Rightarrow f(\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma)) = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma)). \end{aligned}$$

The proof of Lemma C.4.4 can be found in Appendix C.4.2.

Lemma C.4.5. Let $\mathcal{M}$ be an MMT. Then Algorithm 8 yields for $\mathcal{M}$ an MMT $\mathcal{N}$ such that, for its state map $f\colon Q^{\mathcal{M}} \to P^{\mathcal{N}}$:

$$\forall p \in P^{\mathcal{N}}, x \in X^{\mathcal{N}}\colon \qquad x \in \mathcal{X}^{\mathcal{N}}(p) \quad \Longrightarrow \quad x \in \mathcal{X}^{\mathcal{M}}(f(p)).$$

Proof. There are two locations in which Algorithm 8 marks timers x as active in states p of $\mathcal{N}$:

- In the first, it adds timer x to $\mathcal{X}^{\mathcal{N}}(p)$ iff $f(p)$ has a timeout for x. Therefore, Rule 4.5 and Rule 4.6 imply that $x \in \mathcal{X}^{\mathcal{M}}(f(p))$.
- In the second, it performs a backwards breadth-first-search on $\mathcal{N}$, in which it only marks timers x as active in states p when $x \in \mathcal{X}^{\mathcal{M}}(f(p))$.

The property therefore holds. □

Lemma C.4.6. Let $\mathcal{M}$ be an MMT, and let S be the set of state-timer pairs used in Algorithm 8. If $(p, x) \in S$, then for all x-spanning runs $\rho = f(p_n) \xrightarrow[u']{i_n} f(p_{n-1}) \dots \xrightarrow{i_1} f(p) \xrightarrow{i} f(p') \in runs(\mathcal{M})$, the loop of lines 27 through 49 ensures that $\pi = p_n \xrightarrow[u_n]{i_n} p_{n-1} \dots \xrightarrow[u_1]{i_1} p \xrightarrow{i} p' \in runs(\mathcal{N})$ is x-spanning as well.

The proof of Lemma C.4.6 can be found in Appendix C.4.3.

Lemma C.4.7. Let $\mathcal{M}$ be an MMT. The loop of lines 27 through 49 of Algorithm 8 only marks x as active in state $p \in P^{\mathcal{N}}$ if there is an x-spanning run $\pi \in runs(\mathcal{N})$ that traverses p.

Proof. The loop of lines 27 through 49 only marks timers x as active in states $p_0 \in P^{\mathcal{N}}$ on Line 41. To reach Line 41 for x and p_0, there must be a $(p_n, x) \in S$ with the same timer x. The loop performs a backwards breadth-first-search through $\mathcal{N}$'s transition structure. It would only reach p with a run $\pi = p \xrightarrow[u_1]{i_1} q_1 \dots \xrightarrow[u_n]{i_n} p_n \in runs(\mathcal{N})$ if:

$$\forall l \in \{1, \dots, n\}\colon \qquad u_1 \neq (x, c) \quad \wedge \quad x \in \mathcal{X}^{\mathcal{M}}(f(q_l)),$$

and $x \in \mathcal{X}^{\mathcal{M}}(f(q))$. The procedure then also marks x as active in all states along π. Let $\pi' = f(p) \xrightarrow{i_1} f(p_1) \dots \xrightarrow{i_n} f(p_n) \in runs(\mathcal{M})$. Timer x being active in $f(q_l)$ implies that there is a run $\rho \in runs(\mathcal{M})$ that (re)starts x, and which terminates in $f(q_l)$. Therefore, $\rho \cdot \pi' \in runs(\mathcal{M})$ is an x-spanning run. Thus, Lemma C.4.6 tells us that the run in $\mathcal{T}$ for which mapping all states with f results in the run $\rho \cdot \pi'$ is x-spanning. This run traverses p, as required.

□

Lemma C.4.8. Let $\mathcal{M}$ be a complete MMT. Then Algorithm 8 makes it so that:

$$\forall x \in X^{\mathcal{N}}, k \in \{1, \dots, n-1\}, j \in \{2, n\}\colon$$
$$p_{k-1} \xrightarrow[(x,c)]{i_k} q_k \xrightarrow{i_{k+1} \dots i_j} p_j \text{ is } x\text{-spanning} \iff f(p_{k-1}) \xrightarrow[(x,c)]{i_k} f(q_k) \xrightarrow{i_{k+1} \dots i_j} f(p_j) \text{ is } x\text{-spanning}.$$

The proof of Lemma C.4.8 can be found in Appendix C.4.4.

Lemma C.4.9. Let $\mathcal{M}$ be a complete MMT. Then Algorithm 8 makes it so that:

$$\forall p \in P^{\mathcal{N}}, x \in X^{\mathcal{N}}\colon \quad x \in \mathcal{X}^{\mathcal{N}}(p) \implies (\exists \pi \in runs(\mathcal{N})\colon \pi \text{ is } x\text{-spanning} \wedge \pi \text{ traverses } p).$$

Proof. Let $p \in P^{\mathcal{N}}$ and $x \in X^{\mathcal{N}}$. Suppose that $x \in \mathcal{X}^{\mathcal{N}}(p)$. There are two locations in which Algorithm 8 could have made x active in p:

1. In the first location, on Line 22, Algorithm 8 would add x to $\mathcal{X}^{\mathcal{N}}(p)$ iff it previously added a timeout transition for x from p on Line 18. The algorithm would only do so if $f(p) \xrightarrow{\mathsf{to}[x]} \in runs(\mathcal{M})$. Therefore, we know that there is an x-spanning run $f(q_{k-1}) \xrightarrow{i_k \dots i_{j-1}} f(p) \xrightarrow{\mathsf{to}[x]} f(p_j) \in runs(\mathcal{M})$. Lemma C.4.8 now tells us that $q_{k-1} \xrightarrow{i_k \dots i_{j-1}} p \xrightarrow{\mathsf{to}[x]} p_j \in runs(\mathcal{N})$ is an x-spanning run as well. This means that in the first case, there is indeed a run $\pi \in runs(\mathcal{N})$ that is both x-spanning, and that traverses p as required.
2. The second location is the loop of lines 27 through 49. Lemma C.4.7 tells us that here, timers x are only ever marked as active in states $p \in P^{\mathcal{N}}$ if $\mathcal{N}$ has an x-spanning run that traverses p, as required.

□

C.4.1 Proof of Lemma C.4.3

Proof. We know from Lemma C.4.1 that f is a bijection between $P^{\mathcal{N}}$ and $Q^{\mathcal{M}}$. We know from lines 16 through 26 that, for all $p \in P^{\mathcal{N}}$ and all $i \in I \cup \{\mathsf{to}[x] \in TO(X^{\mathcal{M}}) \mid \delta^{\mathcal{M}}(f(p), \mathsf{to}[x])\downarrow\}$:

- $p' = \delta^{\mathcal{N}}(p, i) \coloneqq f^{-1}(\delta^{\mathcal{M}}(f(p), i))$, and
- $o = \lambda^{\mathcal{N}}(p, i) \coloneqq \lambda^{\mathcal{M}}(f(p), i)$.

This implies that:

$$\forall p, p' \in P^{\mathcal{N}}, i \in A^{\mathcal{N}}, o \in O^{\mathcal{N}}\colon \quad f(p) \xrightarrow{i/o} f(p') \implies p \xrightarrow{i/o} p'.$$

Since lines 16 through 26 is the only place where Algorithm 8 adds transitions to $\mathcal{N}$, we know that:

$$\forall p, p' \in P^{\mathcal{N}}, i \in A^{\mathcal{N}}, o \in O^{\mathcal{N}}\colon \quad p \xrightarrow{i/o} p' \implies f(p) \xrightarrow{i/o} f(p').$$

Therefore:

$$\forall p, p' \in P^{\mathcal{N}}, i \in A^{\mathcal{N}}, o \in O^{\mathcal{N}}\colon \quad p \xrightarrow{i/o} p' \iff f(p) \xrightarrow{i/o} f(p'),$$

as required. □

C.4.2 Proof of Lemma C.4.4

Proof. We use an induction on $\sigma \in (A^{\mathcal{N}})^*$ to obtain the property:

- Base case: if $\sigma = \epsilon$, then $\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma) = p_{\mathcal{I}}^{\mathcal{N}}$ and $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma) = q_{\mathcal{I}}$. Therefore:
  - $\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma)\downarrow \Leftrightarrow \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma)\downarrow$,
  - $\lambda^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma) = \epsilon = \lambda^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma)$, and
  - Since $\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma) = p_{\mathcal{I}}^{\mathcal{N}}$ and $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma) = q_{\mathcal{I}}$, Lemma C.4.2 tells us that:

    $$f(\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma)) = f(p_{\mathcal{I}}^{\mathcal{N}}) = q_{\mathcal{I}} = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma),$$

    as required.
- Inductive step case: if $\sigma = \rho\ i$ for some $\rho \in (A^{\mathcal{N}})^*$ and $i \in A^{\mathcal{N}}$. We use the induction hypothesis:

  $$\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho)\downarrow \Leftrightarrow \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho)\downarrow \wedge$$
  $$\lambda^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho) = \lambda^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho) \wedge$$
  $$(\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho)\downarrow \Rightarrow f(\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho)) = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho)).$$

  We perform a case distinction on whether $\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho)\downarrow$:
  - If $\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho)\downarrow$, then the induction hypothesis tells us that $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho)\downarrow$. Let $p = \delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho)$ and let $q = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho)$. The induction hypothesis tells us that $f(p) = q$. Lemma C.4.3 tells us that therefore:

    $$\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho\ i)\downarrow \iff \delta^{\mathcal{N}}(p, i)\downarrow \iff \delta^{\mathcal{M}}(f(p), i)\downarrow \iff \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho\ i)\downarrow,$$

    as required.
  - If $\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho)\uparrow$, then the induction hypothesis tells us that $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho)\uparrow$. Undefined transition sequences cannot become defined when they are extended with additional transitions. We thus know that:

    $$\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho)\uparrow \implies \delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho\ i)\uparrow,$$

    and that:

    $$\delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho)\uparrow \implies \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho\ i)\uparrow.$$

  We see that $\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho\ i)\downarrow \iff \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho\ i)\downarrow$, as required. If $\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho\ i)\uparrow$ and $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho\ i)\uparrow$, then there is nothing more for us to show for this inductive step case. We will therefore assume that $\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho\ i)\downarrow$ and $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho\ i)\downarrow$ in the remainder of the step case. The induction hypothesis tells us that therefore, $f(\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho)) = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho)$. Let $p = \delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho)$, and let $q = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho)$. We thus know that $f(p) = q$. Therefore, we know from Lemma C.4.3 that:
  - the outputs are equal:

    $$\begin{aligned}\lambda^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho\ i) &= \lambda^{\mathcal{N}}(p, i) \\ &= \lambda^{\mathcal{M}}(f(p), i) && \text{(Lemma C.4.3)} \\ &= \lambda^{\mathcal{M}}(q, i) = \lambda^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho\ i),\end{aligned}$$

    as required.
  - the f-mapping is preserved:

    $$\begin{aligned}f(\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho\ i)) &= f(\delta^{\mathcal{N}}(p, i)) \\ &= \delta^{\mathcal{M}}(f(p), i) && \text{(Lemma C.4.3)} \\ &= \delta^{\mathcal{M}}(q, i) = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho\ i),\end{aligned}$$

    as required.

We thus see that:

$$\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho\ i)\downarrow \Leftrightarrow \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho\ i)\downarrow \wedge$$
$$\lambda^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho\ i) = \lambda^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho\ i) \wedge$$
$$(\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho\ i)\downarrow \Rightarrow f(\delta^{\mathcal{N}^*}(p_{\mathcal{I}}^{\mathcal{N}}, \rho\ i)) = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \rho\ i)).$$

as required.

We have thus shown by induction on the input sequence that the property indeed holds. □

C.4.3 Proof of Lemma C.4.6

Proof. For each $(p, x) \in S$, lines 27 through 49 perform a backwards breadth-first-search through $\mathcal{N}$'s transition structure that starts from p. Let $\rho = f(p_n) \xrightarrow[u]{i_n} f(p_{n-1}) \dots \xrightarrow{i_1} f(p) \xrightarrow{i} f(p') \in runs(\mathcal{M})$ be an x-spanning run, and let $\sigma = i_1 \dots i_k$ be the inverse of a suffix of the action sequence $i_n \dots i_1$ that labels the transitions of ρ. Set E is initialized as $\{p\}$. We perform an induction on the content of the first-in-first-out queue R, which is initialized as $R = [p]$:

- Base case: $R = [p]$. The search performs certain checks for all $p_a \in P$, $i_a \in I \cup TO(X^{\mathcal{N}})$ for which $\delta^{\mathcal{N}}(p_a, i_a) = p$. This includes the case in which $p_a = p_1$ and $i_a = i_1$, which is the only relevant case for this proof. So suppose that $i_a = i_1$ and $p_a = p_1$. Since ρ is x-spanning, $\tau^{\mathcal{M}}(f(p_1), i_1) = (y, c) \Rightarrow y \neq x$. Since ρ is x-spanning and ρ traverses $f(p_1)$, $x \in \mathcal{X}^{\mathcal{M}}(f(p_1))$. Since $E = \{p\}$, $p_1 \notin E$. Therefore, x is marked as active in p_1, p_1 is appended to R, and p_1 is added to E. Timer update u_1 wasn't changed from $\perp$, as required.

- Inductive step case: $p_{k+1} = R.\mathsf{dequeue}()$ with $k \in \{1, \dots, n-1\}$. We use the induction hypothesis:

  - $\forall l \in \{1, \dots, k\}$: $\quad x \in \mathcal{X}^{\mathcal{N}}(p_l) \wedge x \in \mathcal{X}^{\mathcal{M}}(f(p_l))$,
  - $\forall l \in \{1, \dots, k\}$: $\quad u_l = (y, c) \implies y \neq x$, *and*
  - $\forall l \in \{1, \dots, k\}$: $\quad p_l \in E$.
  - $\forall l \in \{1, \dots, k\}$: $\quad p_l \in E \implies x \in \mathcal{X}^{\mathcal{N}}(p_l) \wedge x \in \mathcal{X}^{\mathcal{M}}(f(p_l)) \wedge u_l = (y, c) \Longrightarrow y \neq x$.
  - $\forall l \in \{1, \dots, k\}$: $\quad p_l \in R \vee p_l \in E$.

  Then the search performs certain checks for all $p_a \in P$, $i_a \in I \cup TO(X^{\mathcal{N}})$ for which $\delta^{\mathcal{N}}(p_a, i_a) = p_k$. Since p_k is a state along π, this includes the case in which $p_a = p_{k+1}$ and $i_a = i_{k+1}$, which is the only relevant case for this proof. So suppose that $i_a = i_{k+1}$ and $p_a = p_{k+1}$. Then:

  - if $k + 1 = n$, then since ρ is x-spanning, $\tau^{\mathcal{M}}(f(p_{k+1}), i_{k+1}) = (x, c)$, for some $c \in \mathbb{N}^{>0}$. The procedure then marks $u = \tau^{\mathcal{N}}(p_n, i_n) = (x, c)$, which then makes π an x-spanning run per the induction hypothesis.
  - if $k+1 < n$, then since ρ is x-spanning, $\tau^{\mathcal{M}}(f(p_{k+1}), i_{k+1}) = (y, c) \Rightarrow y \neq x$. Since ρ is x-spanning and ρ traverses $f(p_{k+1})$, $x \in \mathcal{X}^{\mathcal{M}}(f(p_{k+1}))$. Therefore:
    - if $p_{k+1} \notin E$, then x is marked as active in p_{k+1}. State p_{k+1} is appended to R, and added to E. Timer update u_{k+1} wasn't changed from $\perp$, as required.
    - if $p_{k+1} \in E$, then the induction hypothesis tells us that $x \in \mathcal{X}^{\mathcal{T}}(p_{k+1})$, $u_{k+1} = (y, c) \Rightarrow y \neq x$ and $p_{k+1} \in E$, as required.

We have shown by induction on R that $u = (x, c)$, and that $\forall l \in \{1, \dots, n\} : x \in \mathcal{X}^{\mathcal{T}}(p_l)$. The fact that π is x-spanning now follows from the facts that $i = \mathsf{to}[x]$, and that since $(p, x) \in S$, x was added to $\mathcal{X}^{\mathcal{T}}(p)$ on Line 22. □

C.4.4 Proof of Lemma C.4.8

Proof. We know from Lemma C.4.3 that $p_{j-1} \xrightarrow{\mathsf{to}[x]} \in runs(\mathcal{N})$ iff $f(p_{j-1}) \xrightarrow{\mathsf{to}[x]} \in runs(\mathcal{M})$. Therefore, if either:

- $\pi = p_{k-1} \xrightarrow{i_k \dots i_j} p_j \in runs(\mathcal{N})$ is an x-spanning run, or:
- $\pi' = f(p_{k-1}) \xrightarrow{i_k \dots i_j} f(p_j)$ is an x-spanning run,

then we know that $p_{j-1} \xrightarrow{\mathsf{to}[x]} \in runs(\mathcal{N})$ and $f(p_{j-1}) \xrightarrow{\mathsf{to}[x]} \in runs(\mathcal{M})$, and that:

$$\forall l \in \{k, \dots, j-2\} \colon p_l \xrightarrow{\mathsf{to}[x]} \notin runs(\mathcal{N}) \wedge f(p_l) \xrightarrow{\mathsf{to}[x]} \notin runs(\mathcal{M}).$$

Algorithm 8 only adds timeout actions to $\mathcal{N}$ on Line 18. The fact that $p_{j-1} \xrightarrow{\mathsf{to}[x]} \in runs(\mathcal{N})$ therefore implies that this timeout was added on Line 18, which implies that (p_{j-1}, x) was added to S on Line 23.

In order to prove the lemma, we will show that if either π or π' is x-spanning, then:

$$p_{k-1} \xrightarrow[(x,c)]{i_k} p_k, \quad f(p_{k-1}) \xrightarrow[(x,c)]{i_k} f(p_k),$$

and:

$$\forall l \in \{k, \dots, j-1\} \colon \qquad x \in \mathcal{X}^{\mathcal{N}}(q_l) \quad \wedge \quad x \in \mathcal{X}^{\mathcal{M}}(f(q_l)),$$

which would mean that π and π' are both x-spanning. We use the following case distinction:

- If $\pi = p_{k-1} \xrightarrow[u]{i_k} q_k \xrightarrow{i_{k+1} \dots i_j} p_j$ is an x-spanning run, then the update $u = (x, c)$ that starts this spanning must have been added on Line 37, as that is the only place where Algorithm 8 adds timer updates to $\mathcal{N}$. Algorithm 8 only adds timer updates when they also exist in the corresponding transitions of $\mathcal{M}$. We thus know that $f(p_{k-1}) \xrightarrow[(x,c)]{i_k} f(q_k) \in runs(\mathcal{M})$.

  The fact that π is an x-spanning run implies that:

  $$\forall l \in \{k, \dots, j-1\} \colon \quad x \in \mathcal{X}^{\mathcal{N}}(q_l).$$

  Lemma C.4.5 now tells us that:

  $$\forall l \in \{k, \dots, j-1\} \colon \quad x \in \mathcal{X}^{\mathcal{M}}(f(q_l)).$$

  We thus know that $f(p_{k-1}) \xrightarrow{i_k \dots i_j} f(p_j)$ is an x-spanning run, as required.

- In the second case, $\pi' = f(p_{k-1}) \xrightarrow[u]{i_k} f(q_k) \xrightarrow{i_{k+1} \dots i_j} f(p_j)$ is an x-spanning run. We already established that $(p_{j-1}, x) \in S$. The loop of lines 27 through 49 is therefore run for $(p_{j-1}, x) \in S$. Lemma C.4.6 now tells us that since π' is an x-spanning run of $\mathcal{M}$, the loop of lines 27 through 49 will ensure that $p_{k-1} \xrightarrow[u]{i_k} q_k \xrightarrow{i_{k+1} \dots i_j} p_j$ is x-spanning as well, as required.

The property therefore holds. □

C.4.5 Proof of Theorem 5.2.1

Proof. Algorithm 8 always returns a tuple with the structure of an MMT. It remains for us to show that these MMTs are always valid. We discuss each of the six rules for valid MMTs:

- We use a proof by contradiction to show that Rule 4.1 holds for $\mathcal{N}$: Assume that $\mathcal{X}^{\mathcal{N}}(p_{\mathcal{I}}^{\mathcal{N}}) \neq \emptyset$. This then implies that $\exists x \in X^{\mathcal{N}} \colon x \in \mathcal{X}^{\mathcal{N}}(p_{\mathcal{I}}^{\mathcal{N}})$. Lemma C.4.5 tells us that therefore, $x \in \mathcal{X}^{\mathcal{M}}(f(p_{\mathcal{I}}^{\mathcal{N}}))$. Lemma C.4.2 now tells us that $x \in \mathcal{X}^{\mathcal{M}}(q_{\mathcal{I}})$. Since $\mathcal{M}$ is valid, $x \in \mathcal{X}^{\mathcal{M}}(q_{\mathcal{I}})$ contradicts Rule 4.1. Therefore, $\mathcal{X}^{\mathcal{N}}(p_{\mathcal{I}}^{\mathcal{N}}) = \emptyset$, as required.

- Rule 4.2 holds for $\mathcal{N}$, since in the only location in which outputs and transitions for states of $\mathcal{N}$ and actions are set, they are both set for the same states and actions.

- Suppose that $p \xrightarrow[\bot]{i} p'$. Rule 4.3 then holds for $\mathcal{N}$, since Lemma C.4.9 tells us that a timer x is only active in a state p if $\mathcal{N}$ has an x-spanning run that traverses p. This implies that for x to be active in p', but not in p could only happen if the i-transition starts x. So in this case, in which no timers are started in the transition, timer x is never active in p' if it is not active in p, and so the property holds.

- Suppose that $p \xrightarrow[(x,c)]{i} p'$. Rule 4.4 then holds for $\mathcal{N}$, since the only location at which Algorithm 8 adds timer updates for timers x to transitions is in the final loop of lines 27 through 49, in which it performs a backwards breadth-first-search that marks x as active in all states that it encounters, until it finds a previous transition that (re)starts x. We thus know that x is active in p'.
  Lemma C.4.9 tells us that timers y are only active in p' if $\mathcal{N}$ has a y-spanning run that traverses p'. The i-transition cannot (re)start timer y, since it already (re)starts x. Therefore, y can only be active in p' if the y-spanning traverses p as well, and $y \in \mathcal{X}^{\mathcal{N}}(p)$. Therefore, $\mathcal{X}^{\mathcal{N}}(p') \setminus \{x\} \subseteq \mathcal{X}^{\mathcal{N}}(p)$, as required.

- Suppose that $p \xrightarrow[\bot]{\mathsf{to}[x]} p'$. The:

  - left-hand-side of the conjunction of Rule 4.5 holds, because in the first location in which Algorithm 8 makes timers active, it makes timers x active in states p iff it adds a timeout transition for x from p.
  - right-hand-side of the conjunction of Rule 4.5 follows from an argument by contradiction: Assume that $x \in \mathcal{X}^{\mathcal{N}}(p')$. There are two locations in which Algorithm 8 could have marked x as active in p':
    * In the first, it would only mark x as active in p' if it also adds a timeout transition for x from p'. The algorithm only adds such a transition if $f(p') \xrightarrow{\mathsf{to}[x]} \in runs(\mathcal{M})$. Since $\mathcal{M}$ is valid, $f(p') \xrightarrow{\mathsf{to}[x]} \in runs(\mathcal{M})$ would only hold if $x \in \mathcal{X}^{\mathcal{M}}(f(p'))$, per Rule 4.5 and Rule 4.6. Lemma C.4.3 tells us that $f(p) \xrightarrow{\mathsf{to}[x]} f(p')$. Therefore, Rule 4.5 and Rule 4.6 tell us that since $x \in \mathcal{X}^{\mathcal{M}}(f(p'))$, $f(p) \xrightarrow[(x,c)]{\mathsf{to}[x]} f(p')$. Algorithm 8 would add a timer update for x to the $p \xrightarrow{\mathsf{to}[x]} p'$-transition. This leads to a contradiction, since $p \xrightarrow[\bot]{\mathsf{to}[x]} p'$.
    * In the second, it would mark x as active in p' iff $\mathcal{M}$ has an x-spanning run that traverses $f(p')$. Since $\mathcal{M}$ is valid, we know from Rule 4.5 and Rule 4.6 that this is only the case if the transition's counterpart in $\mathcal{M}$ starts timer x, i.e. if $f(p) \xrightarrow[(x,c)]{\mathsf{to}[x]} f(p')$. Algorithm 8 would add a timer update for x to the $p \xrightarrow{\mathsf{to}[x]} p'$-transition. This leads to a contradiction, since $p \xrightarrow[\bot]{\mathsf{to}[x]} p'$.

    We can thus conclude that $x \notin \mathcal{X}^{\mathcal{N}}(p')$, as required.

  We can thus conclude that Rule 4.5 holds for $\mathcal{N}$.

- Suppose that $p \xrightarrow[(y,c)]{\mathsf{to}[x]} p'$. The:

  - left-hand-side of the conjunction of Rule 4.6 holds, because in the first location in which Algorithm 8 makes timers active, it makes timers x active in states p iff it adds a timeout transition for x from p.
  - right-hand-side of the conjunction of Rule 4.6 holds, because Algorithm 8 only adds a timer update for a timer y in this $\mathsf{to}[x]$-transition if its counterpart transition $f(p) \xrightarrow[(x,y)]{\mathsf{to}[x]} f(p')$ in $\mathcal{N}$ has a timer update for y as well. Since $\mathcal{M}$ is valid, we know from Rule 4.6 that $y = x$, as required.

We can thus conclude that Rule 4.6 holds for $\mathcal{N}$.

We can thus conclude that Algorithm 8 always returns valid MMTs when it is given valid MMTs. □

C.4.6 Proof of Theorem 5.2.3

The proof of Theorem 5.2.3 relies on the following auxiliary lemmas:

Lemma C.4.10. Let $\mathcal{M}$ be an MMT, and let $\mathcal{N}$ be the MMT that Algorithm 8 returns when it is called on $\mathcal{M}$. Then:

$$A^{\mathcal{N}} = \{i \in A^{\mathcal{M}} \mid \exists q \in Q^{\mathcal{M}} \colon q \xrightarrow{i} \in runs(\mathcal{M})\}.$$

Proof. We can see on Line 5 of Algorithm 8 that $I^{\mathcal{N}} = I^{\mathcal{M}}$. All that remains for us to prove is that:

$$\forall x \in X^{\mathcal{M}} \colon \quad (\exists q \colon q \xrightarrow{\mathsf{to}[x]} \in runs(\mathcal{M})) \implies x \in X^{\mathcal{N}}. \tag{C.1}$$

The loop of lines 16 through 26 iterates over each of $\mathcal{M}$'s state-action pairs. For each of them, it checks whether the action is a timeout, upon which it adds the timeout's timer to $X^{\mathcal{N}}$ on Line 21. The condition of Equation (C.1) therefore holds. □

Lemma C.4.11. Let $\mathcal{M}$ be an MMT, and let $\mathcal{N}$ be the MMT that Algorithm 8 returns when it is called on $\mathcal{M}$. Then:

$$\forall q \in Q^{\mathcal{M}}, \sigma \in (A^{\mathcal{M}})^* \colon \quad q \xrightarrow{\sigma} \in runs(\mathcal{M}) \implies \sigma \in (A^{\mathcal{N}})^*.$$

Proof. Let $q \in Q^{\mathcal{M}}$, and let $\sigma = i_1 \dots i_n \in (A^{\mathcal{M}})^*$. Suppose $q \xrightarrow{\sigma} \in runs(\mathcal{M})$. Then Lemma C.4.10 implies that:

$$\{i_1, \dots, i_n\} \subseteq A^{\mathcal{N}}.$$

This implies that $\sigma \in (A^{\mathcal{N}})^*$, as required. □

The proof of Theorem 5.2.3 is given by:

Proof. Since $\mathcal{M}$ is connected:

$$\forall q \in Q^{\mathcal{M}} \colon \quad \exists \sigma \in (A^{\mathcal{M}})^* \colon q_{\mathcal{I}} \xrightarrow{\sigma} q \in runs(\mathcal{M}).$$

Thus, by Lemma C.4.11:

$$\forall q \in Q^{\mathcal{M}} \colon \quad \exists \sigma \in (A^{\mathcal{N}})^* \colon q_{\mathcal{I}} \xrightarrow{\sigma} q \in runs(\mathcal{M}).$$

Since f is bijective per Lemma C.4.1:

$$\forall p \in P^{\mathcal{N}} \colon \quad \exists q \in Q^{\mathcal{M}} \colon \quad f(p) = q \wedge (\exists \sigma \in (A^{\mathcal{N}})^* \colon q_{\mathcal{I}} \xrightarrow{\sigma} q \in runs(\mathcal{M})).$$

By Lemma C.4.4:

$$\begin{aligned} &\forall p \in P^{\mathcal{N}} \colon \\ &\quad \exists q \in Q^{\mathcal{M}} \colon \quad f(p) = q \wedge (\exists \sigma \in (A^{\mathcal{N}})^* \colon q_{\mathcal{I}} \xrightarrow{\sigma} q \in runs(\mathcal{M}) \wedge (\exists p' \colon p^{\mathcal{N}}_{\mathcal{I}} \xrightarrow{\sigma} p' \in runs(\mathcal{N}))). \end{aligned} \tag{C.2}$$

In Equation (C.2), $q_{\mathcal{I}} \xrightarrow{\sigma} \in runs(\mathcal{M})$ implies that $\delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \sigma)\downarrow$. Thus, by Lemma C.4.4, $f(p') = q$. Since f is injective per Lemma C.4.1, $p' = p$. We get:

$$\forall p \in P^{\mathcal{N}} \colon \quad \exists q \in Q^{\mathcal{M}} \colon \quad f(p) = q \wedge (\exists \sigma \in (A^{\mathcal{N}})^* \colon q_{\mathcal{I}} \xrightarrow{\sigma} q \in runs(\mathcal{M}) \wedge p^{\mathcal{N}}_{\mathcal{I}} \xrightarrow{\sigma} p \in runs(\mathcal{N})).$$

Therefore:

$$\forall p \in P^{\mathcal{N}} \colon \quad \exists \sigma \in (A^{\mathcal{N}})^* \colon p^{\mathcal{N}}_{\mathcal{I}} \xrightarrow{\sigma} p \in runs(\mathcal{N}),$$

as required. □

C.4.7 Proof of Theorem 5.2.4

Proof. We prove that each of the three conditions holds:

1. The fact that $\mathcal{M}$ is s-learnable implies that all runs of $\mathcal{M}$ are feasible. Theorem 5.2.2 tells us that $\mathcal{N} \approx_{sym} \mathcal{M}$. Bruyère et al. [2024] has proven that therefore, $\mathcal{N}$ and $\mathcal{M}$ satisfy their notion of timed equivalence. What exactly it means for $\mathcal{N}$ and $\mathcal{M}$ to be timed equivalent is irrelevant to this proof. What does matter is that since $\mathcal{N}$ and $\mathcal{M}$ are timed equivalent, they accept the exact same timed input words, which implies that thus that the same runs are feasible in both of them. Since $\mathcal{N}$ has the same transition structure as $\mathcal{M}$, including all timer updates that start spanning runs, this implies that since all runs of $\mathcal{M}$ are feasible, all runs of $\mathcal{N}$ are feasible as well.

2. Lemma C.4.9 tells us that:

$$\forall x \in X^{\mathcal{N}}, p \in P^{\mathcal{N}}: \qquad x \in \mathcal{X}^{\mathcal{N}}(p) \iff \text{there is an } x\text{-spanning run that traverses } p,$$

   as required.

3. Let $p \in P^{\mathcal{N}}$, and let $x \in X^{\mathcal{N}}$. By Theorem 5.2.3, there exists an action sequence $\sigma \in (A^{\mathcal{N}})^*$, such that $\delta^{\mathcal{N}*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma) = p$. Lemma C.4.4 tells us that $\delta^{\mathcal{M}}(q_{\mathcal{I}}, \sigma)\downarrow$, and that $\delta^{\mathcal{M}}(q_{\mathcal{I}}, \sigma) = f(\delta^{\mathcal{N}*}(p_{\mathcal{I}}^{\mathcal{N}}, \sigma))$. Let $q = \delta^{\mathcal{M}}(q_{\mathcal{I}}, \sigma)$. We can see that $f(p) = q$. Lemma C.4.3 tells us that therefore:

$$\forall x \in X^{\mathcal{N}}: \qquad \delta^{\mathcal{N}}(p, \mathsf{to}[x])\downarrow \iff \delta^{\mathcal{M}}(q, \mathsf{to}[x])\downarrow.$$

   Since $\mathcal{M}$ is s-learnable, it is complete, which implies that:

$$\forall q \in Q^{\mathcal{M}}, x \in X^{\mathcal{M}}: \qquad x \in \mathcal{X}_0^{\mathcal{M}}(q) \iff \delta^{\mathcal{M}}(q, \mathsf{to}[x])\downarrow.$$

   We thus know that:

$$\forall p \in P^{\mathcal{N}}, x \in X^{\mathcal{N}}: \qquad x \in \mathcal{X}_0^{\mathcal{M}}(q) \iff \delta^{\mathcal{N}}(p, \mathsf{to}[x])\downarrow. \tag{C.3}$$

   Lemma C.4.8 tells us that in the runs $\pi = p_{\mathcal{I}}^{\mathcal{N}} \xrightarrow{\sigma} p \in runs(\mathcal{N})$ and $\pi' = q_{\mathcal{I}} \xrightarrow{\sigma} q \in runs(\mathcal{M})$, π has timer update (x, c) at index i iff π' has timer update (x, c) at index i. Since both runs also have identical timeouts at the same indices, we know that:

$$\forall q \in Q^{\mathcal{M}}, x \in X^{\mathcal{M}}: \qquad x \notin \mathcal{X}_0^{\mathcal{M}}(q) \implies x \notin \mathcal{X}_0^{\mathcal{N}}(p),$$

   and that:

$$\forall x \in \mathcal{X}_0^{\mathcal{M}}(q): \qquad x \in \mathcal{X}^{\mathcal{N}}(p) \implies x \in \mathcal{X}_0^{\mathcal{N}}(p).$$

   We can also see that:

$$\begin{aligned} \forall q \in Q^{\mathcal{M}}, x \in X^{\mathcal{M}}: \qquad & x \in \mathcal{X}_0^{\mathcal{M}}(q) \\ \implies\ & \delta^{\mathcal{M}}(q, \mathsf{to}[x])\downarrow \\ \implies\ & \delta^{\mathcal{N}}(p, \mathsf{to}[x])\downarrow \\ \implies\ & x \in \mathcal{X}^{\mathcal{N}}(p). \qquad \text{(Rule 4.5 and Rule 4.6)} \end{aligned}$$

   Therefore:

$$\forall q \in Q^{\mathcal{M}}, x \in X^{\mathcal{M}}: \qquad x \in \mathcal{X}_0^{\mathcal{M}}(q) \implies x \in \mathcal{X}_0^{\mathcal{N}}(p),$$

   We can thus conclude that:

$$\forall q \in Q^{\mathcal{M}}, x \in X^{\mathcal{M}}: \qquad x \in \mathcal{X}_0^{\mathcal{M}}(q) \iff x \in \mathcal{X}_0^{\mathcal{N}}(p).$$

   Equation (C.3) now tells us that:

$$\forall p \in P^{\mathcal{N}}, x \in X^{\mathcal{N}}: \qquad x \in \mathcal{X}_0^{\mathcal{N}}(p) \iff \delta^{\mathcal{N}}(p, \mathsf{to}[x])\downarrow,$$

   as required.

Since all three conditions hold, $\mathcal{N}$ is t-observable. □

C.4.8 Proof of Theorem 5.2.5

The proof of Theorem 5.2.5 is given by:

Proof. Theorem 5.2.4 tells us that since $\mathcal{M}$ is s-learnable, $\mathcal{N}$ is t-observable. This implies that:

$$\forall p \in P^{\mathcal{N}}, x \in X^{\mathcal{N}}: \qquad x \in \mathcal{X}_0^{\mathcal{N}}(p) \iff \delta^{\mathcal{N}}(p, \mathsf{to}[x])\downarrow.$$

All that remains for us to prove is that:

$$\forall p \in P^{\mathcal{N}}, i \in I^{\mathcal{N}}: \qquad p \xrightarrow{i} \in runs(\mathcal{N}). \tag{C.4}$$

Since $\mathcal{M}$ is s-learnable, it is complete. Therefore:

$$\forall q \in Q^{\mathcal{M}}, i \in I^{\mathcal{M}}: \qquad q \xrightarrow{i} \in runs(\mathcal{M}).$$

Therefore, since f is bijective per Lemma C.4.1:

$$\forall p \in P^{\mathcal{N}}, i \in I^{\mathcal{M}}: \qquad \exists q \in Q^{\mathcal{M}}: q \xrightarrow{i} \in runs(\mathcal{M}) \wedge f(p) = q.$$

This implies that:

$$\forall p \in P^{\mathcal{N}}, i \in I^{\mathcal{M}}: \qquad f(p) \xrightarrow{i} \in runs(\mathcal{M}).$$

By Lemma C.4.10:

$$\forall p \in P^{\mathcal{N}}, i \in I^{\mathcal{N}}: \qquad f(p) \xrightarrow{i} \in runs(\mathcal{M}).$$

Lemma C.4.3 now tells us that:

$$\forall p \in P^{\mathcal{N}}, i \in I^{\mathcal{N}}: \qquad p \xrightarrow{i} \in runs(\mathcal{N}).$$

This was the property from Equation (C.4) that we needed to prove. So we are done. □

C.5 (g)MMT Bisimulations

In this appendix, we introduce bisimulations between t-observablegMMTs and t-observable MMTs. We use these bisimulations to prove the correctness of the MMT testing procedure that we introduce in Chapter 5. The current section provides lemmas that we use for this proof, which can be found in Appendix C.6.1.

Let $\mathcal{M}$ be a t-observable gMMT, and let $\mathcal{N}$ be a t-observable MMT. This notion of bisimulation matches states $q \in Q^{\mathcal{M}}$ with states $p \in P^{\mathcal{N}}$, along with a timer mapping $\mu\colon (\mathcal{X}^{\mathcal{M}}(q) \to \mathcal{X}^{\mathcal{N}}(p))$ that relates q and p's active timers based on the points along the runs at which they were last (re)started.

Definition C.5.1 (Bisimulations between t-observable gMMTs and t-observable MMTs). Let $\mathcal{M}$ be a t-observable gMMT, and let $\mathcal{N}$ be a t-observable MMT. A **bisimulation** between $\mathcal{M}$ and $\mathcal{N}$ is a relation $R \subseteq \{(q, p, \mu) \mid \forall q \in Q^{\mathcal{M}}, p \in P^{\mathcal{N}}, \mu\colon (\mathcal{X}^{\mathcal{M}}(q) \to \mathcal{X}^{\mathcal{N}}(p))\}$.
Let R be a bisimulation. We usually denote elements $(q, p, \mu) \in R$ by $q\ R^{\mu}\ p$. We lift the timer map μ to actions such that:

- $\mu(i) = i$ for every $i \in I$, and
- $\mu(\mathsf{to}[x]) = \mathsf{to}[\mu(x)]$ for every $x \in \mathsf{dom}(\mu)$.

We use the following two functions to update the timer map for a given timer update:

$$updateMap^{\mathcal{M},\mu}_{\mathcal{N},\mathfrak{r}}(q', p') = \{(y', y) \in \mathcal{X}^{\mathcal{M}}(q') \times \mathcal{X}^{\mathcal{N}}(p') \mid y = \mu(\mathfrak{r}(y'))\}$$
$$updateMap^{\mathcal{M},\mu}_{\mathcal{N},\mathfrak{r},(x,x')}(q', p') = \{(y', y) \in \mathcal{X}^{\mathcal{M}}(q') \times \mathcal{X}^{\mathcal{N}}(p') \mid y' \neq x \wedge y = \mu(\mathfrak{r}(y'))\} \cup \{(x, x')\}$$

We require that R satisfies the following conditions:

$$q_{\mathcal{I}} \; R^{\emptyset} \; p^{\mathcal{N}}_{\mathcal{I}} \tag{B0}$$

$$q \; R^{\mu} \; p \implies \forall x, y \in \mathcal{X}^{\mathcal{M}}(q)\colon (x \neq y \implies \mu(x) \neq \mu(y)) \tag{B1}$$

$$q \; R^{\mu} \; p \implies \forall x' \in \mathcal{X}^{\mathcal{N}}(p)\colon (\exists x \in \mathcal{X}^{\mathcal{M}}(q)\colon \mu(x) = x') \tag{B2}$$

$$q \; R^{\mu} \; p \wedge q \xrightarrow[\mathfrak{r}]{i/o} q' \wedge (\neg \exists x \in \mathcal{X}^{\mathcal{M}}(q')\colon \mathfrak{r}(x) \in \mathbb{N}^{>0}) \implies p \xrightarrow[\bot]{\mu(i)/o'} p' \tag{B3a}$$

$$\wedge\, o = o' \tag{B3b}$$

$$\wedge\, q' \; R^{updateMap^{\mathcal{M},\mu}_{\mathcal{N},\mathfrak{r}}(q',p')} \; p' \tag{B3c}$$

$$q \; R^{\mu} \; p \wedge q \xrightarrow[\mathfrak{r}]{i/o} q' \wedge (\mathfrak{r}(x) = c \in \mathbb{N}^{>0}) \implies p \xrightarrow[(x',c')]{\mu(i)/o'} p' \tag{B4a}$$

$$\wedge\, o = o' \tag{B4b}$$

$$\wedge\, c = c' \tag{B4c}$$

$$\wedge\, q' \; R^{updateMap^{\mathcal{M},\mu}_{\mathcal{N},\mathfrak{r},(x,x')}(q',p')} \; p' \tag{B4d}$$

$$q \; R^{\mu} \; p \wedge p \xrightarrow{\mathtt{to}[y']} \implies q \xrightarrow{\mathtt{to}[\mu^{-1}(y')]} \tag{B5}$$

$$q \; R^{\mu} \; p \wedge p \xrightarrow[\bot]{i} \implies q \xrightarrow[\mathfrak{r}]{\mu^{-1}(i)} q' \wedge (\neg \exists x \in \mathcal{X}^{\mathcal{M}}(q')\colon \mathfrak{r}(x) \in \mathbb{N}^{>0}) \tag{B6}$$

$$q \; R^{\mu} \; p \wedge p \xrightarrow[(x',c')]{i} \implies q \xrightarrow[\mathfrak{r}]{\mu^{-1}(i)} q' \wedge (\exists x \in \mathcal{X}^{\mathcal{M}}(q')\colon \mathfrak{r}(x) \in \mathbb{N}^{>0}) \tag{B7}$$

We write $\mathcal{M} \simeq \mathcal{N}$ iff there is a bisimulation R between $\mathcal{M}$ and $\mathcal{N}$.
The following partial function combines the two update functions:

$$updateMapFrom^{\mathcal{M},\mu}_{\mathcal{N},i}(q,p) = \begin{cases} updateMap^{\mathcal{M},\mu}_{\mathcal{N},\mathfrak{r}}(q',p') & \text{if } \neg\exists x\colon \mathfrak{r}(x) \in \mathbb{N}^{>0} \\ updateMap^{\mathcal{M},\mu}_{\mathcal{N},\mathfrak{r},(x,x')}(q',p') & \text{if } \mathfrak{r}(x) \in \mathbb{N}^{>0} \wedge u \in \{x'\} \times \mathbb{N}^{>0} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

where $\mathfrak{r} = \tau^{\mathcal{M}}(q,i)$, $u = \tau^{\mathcal{N}}(p,\mu(i))$, $q' = \delta^{\mathcal{M}}(q,i)$, and $p' = \delta^{\mathcal{N}}(p,\mu(i))$.

Assume that a bisimulation relates $\mathcal{M}$ and $\mathcal{N}$. If a transition from q (re)starts a timer $x \in X^{\mathcal{M}}$, then we know that the matching transition for p (re)starts a timer $y \in X^{\mathcal{N}}$. The bisimulation renames x's timers as needed in the timer mapping μ, so that if $\mathcal{M}$ has a timeout for the timer that started as x and that has been renamed to x', then y must have a timeout for $\mu(x') = y$. This ensures that both models only have matching timeouts for timers that were started at the same points along their runs. Two models can therefore only be bisimilar if they accept the same symbolic words.

We use the following lemma to prove certain properties about these bisimulations:

Lemma C.5.1. Let $\mathcal{M}$ be a t-observable gMMT, and let $\mathcal{N}$ be a t-observable MMT such that $\mathcal{M} \simeq \mathcal{N}$. If $\pi = q_{\mathcal{I}} \xrightarrow[\mathfrak{r}_1]{\mathtt{i}_1/o_1} q_1 \dots \xrightarrow[\mathfrak{r}_n]{\mathtt{i}_n/o_n} q_n$ is a feasible run in $\mathcal{M}$ and $\pi' = p^{\mathcal{N}}_{\mathcal{I}} \xrightarrow[u'_1]{\mathtt{i}_1/o'_1} p_1 \dots \xrightarrow[u'_n]{\mathtt{i}_n/o'_n} p_n$ is a feasible run in $\mathcal{N}$, then, for every $j \in \{0,\dots,n\}$:

1. $\exists \mu_j\colon q_j \; R^{\mu_j} \; p_j$,
2. Moreover:
   (a) $\forall x \in \mathcal{X}^{\mathcal{M}}(q_j)\colon lastStartedAt^{\mathcal{M}}_{q_{\mathcal{I}} \xrightarrow{\mathtt{i}_1 \dots \mathtt{i}_j} q_j}(x) = lastStartedAt^{\mathcal{N}}_{p^{\mathcal{N}}_{\mathcal{I}} \xrightarrow{\mathtt{i}_1 \dots \mathtt{i}_j} p_j}(\mu_j(x))$, and
   (b) if $j < n$, then $\mu_j(i_{j+1}) = i'_{j+1}$, where $q_{\mathcal{I}} \xrightarrow{\sigma = i_1 \dots i_{j+1}} \in runs(\mathcal{M})$ is the run of $\mathcal{M}$ for which $\overline{\sigma} = \mathtt{i}_1 \dots \mathtt{i}_{j+1}$ and $p^{\mathcal{N}}_{\mathcal{I}} \xrightarrow{\sigma' = i'_1 \dots i'_{j+1}} \in runs(\mathcal{N})$ is the run of $\mathcal{N}$ for which $\overline{\sigma'} = \mathtt{i}_1 \dots \mathtt{i}_{j+1}$.

The proof of Lemma C.5.1 can be found in Appendix C.5.1.

Bruyère et al. [2024] provided a version of symbolic equivalence that expresses when a gMMT exhibits the same symbolic behavior as an MMT. We use Lemma C.5.1 to prove that when a t-observable gMMT is bisimilar to a t-observable MMT, then they are also symbolically equivalent:

Lemma C.5.2. Let $\mathcal{M}$ be a t-observable gMMT, and let $\mathcal{N}$ be a t-observable MMT with the same actions, A. Then $\mathcal{M} \simeq \mathcal{N} \Rightarrow \mathcal{M} \approx_{sym} \mathcal{N}$.

The proof of Lemma C.5.2 can be found in Appendix C.5.2.

C.5.1 Proof of Lemma C.5.1

Proof. Suppose that $\pi = q_{\mathcal{I}} \xrightarrow[\mathfrak{r}_1]{\mathtt{i}_1/o_1} q_1 \dots \xrightarrow[\mathfrak{r}_n]{\mathtt{i}_n/o_n} q_n$ is a feasible run in $\mathcal{M}$ and $\pi' = p^{\mathcal{N}}_{\mathcal{I}} \xrightarrow[u'_1]{\mathtt{i}_1/o'_1} p_1 \dots \xrightarrow[u'_n]{\mathtt{i}_n/o'_n} p_n$ is a feasible run in $\mathcal{N}$. Then, for every $j \in \{0, \dots, n\}$:

- Base case: $j = 0$. We have that:
  1. Lemma item 1 follows from (B0), as this gives us: $q_{\mathcal{I}}\ R^{\emptyset}\ p^{\mathcal{N}}_{\mathcal{I}}$.
  2. Moreover:
     (a) Lemma item 2a vacuously holds, since initial (g)MMT states never have active timers.
     (b) Lemma item 2b follows from the fact that since $\mathcal{X}^{\mathcal{M}}(q_{\mathcal{I}}) = \mathcal{X}^{\mathcal{N}}(p^{\mathcal{N}}_{\mathcal{I}}) = \emptyset$, $(i_1 = i'_1) \in I$, which implies that $\mu_j(i_{j+1}) = i_{j+1} = i'_{j+1}$.
- Inductive step case: $0 < j < n$. We use the induction hypothesis (IH):
  1. $\exists \mu_j \colon q_j\ R^{\mu_j}\ p_j$,
  2. *Moreover:*
     (a) $\forall x \in \mathcal{X}^{\mathcal{M}}(q_j)$: $lastStartedAt^{\mathcal{M}}_{q_{\mathcal{I}} \xrightarrow{\mathtt{i}_1 \dots \mathtt{i}_j} q_j}(x) = lastStartedAt^{\mathcal{N}}_{p^{\mathcal{N}}_{\mathcal{I}} \xrightarrow{\mathtt{i}_1 \dots \mathtt{i}_j} p_j}(\mu_j(x))$, *and*
     (b) $\mu_j(i_{j+1}) = i'_{j+1}$, *where* $\rho = q_{\mathcal{I}} \xrightarrow{\sigma = i_1 \dots i_{j+1}} \in runs(\mathcal{M})$ *is the run of* $\mathcal{M}$ *for which* $\overline{\sigma} = \mathtt{i}_1 \dots \mathtt{i}_{j+1}$ *and* $\rho' = p^{\mathcal{N}}_{\mathcal{I}} \xrightarrow{\sigma' = i'_1 \dots i'_{j+1}} \in runs(\mathcal{N})$ *is the run of* $\mathcal{N}$ *for which* $\overline{\sigma'} = \mathtt{i}_1 \dots \mathtt{i}_{j+1}$.

  Let $q = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, \mathtt{i}_1 \dots \mathtt{i}_j) = \delta^{\mathcal{M}^*}(q_{\mathcal{I}}, i_1 \dots i_j)$, and $p = \delta^{\mathcal{N}^*}(p^{\mathcal{N}}_{\mathcal{I}}, \mathtt{i}_1 \dots \mathtt{i}_j) = \delta^{\mathcal{N}^*}(p^{\mathcal{N}}_{\mathcal{I}}, i'_1 \dots i'_j)$.
  For the next step, $j+1$, we get:
  1. We perform a case distinction on $\mathtt{i}_{j+1}$:
     – if $\mathtt{i}_{j+1} \in I$, then $i_{j+1} = i'_{j+1} = \mathtt{i}_{j+1}$. Let $q_{j+1} = \delta^{\mathcal{M}}(q, i_{j+1})$, and let $p_{j+1} = \delta^{\mathcal{N}}(p, i'_{j+1})$. We know from $q_j\ R^{\mu_j}\ p_j$ (IH item 1) that $q_{j+1}\ R^{\mu_{j+1}}\ p_{j+1}$, where:
     $$\mu_{j+1} = updateMap^{\mathcal{M},\mu_j}_{\mathcal{N},\mathfrak{r}_{j+1}}(q_{j+1}, p_{j+1}).$$
     – if $\mathtt{i}_{j+1} = \mathtt{to}[k]$ for some $k \leq j$, then let $x = timerStartedAt^{\mathcal{M}}(q_{\mathcal{I}} \xrightarrow{i_1 \dots i_k})$ be the timer started at index k of ρ, and let $x' = timerStartedAt^{\mathcal{N}}(p^{\mathcal{N}}_{\mathcal{I}} \xrightarrow{i'_1 \dots i'_k})$ be the the timer started at index k of ρ'. Let $i_{j+1} = \mathtt{to}[x]$, and let $i'_{j+1} = \mathtt{to}[x']$. IH item 2b tells us that $\mu_j(i_{j+1}) = i'_{j+1}$.
     Let $q_{j+1} = \delta^{\mathcal{M}}(q, i_{j+1})$, and let $p_{j+1} = \delta^{\mathcal{N}}(p, i'_{j+1})$. We know from $q_j\ R^{\mu_j}\ p_j$ (IH item 1) that $q_{j+1}\ R^{\mu_{j+1}}\ p_{j+1}$, where $\mu_{j+1} = updateMap^{\mathcal{M},\mu_j}_{\mathcal{N},\mathfrak{r}_{j+1},(x,x')}(q_{j+1}, p_{j+1})$.
  2. Moreover, having acquired $q_{j+1}\ R^{\mu_{j+1}}\ p_{j+1}$ from Item 1:
     (a) For all $y \in \mathcal{X}^{\mathcal{M}}(q_{j+1})$, either:
     – $\mathfrak{r}_{j+1}(y) = c \in \mathbb{N}^{>0}$. Then $q_j\ R^{\mu_j}\ p_j$ (IH item 1) tells us that $\exists y' \colon u'_{j+1} = (y', c') \wedge c = c' \wedge \mu_{j+1}(y) = y'$. Then:
     $$lastStartedAt^{\mathcal{N}}_{p^{\mathcal{N}}_{\mathcal{I}} \xrightarrow{\mathtt{i}_1 \dots \mathtt{i}_{j+1}} p_{j+1}}(\mu_{j+1}(y)) = j+1 = lastStartedAt^{\mathcal{M}}_{q_{\mathcal{I}} \xrightarrow{\mathtt{i}_1 \dots \mathtt{i}_{j+1}} q_{j+1}}(y),$$
     as required; or

– $\mathfrak{r}_{j+1}(y) = x$ for some $x \in \mathcal{X}^{\mathcal{M}}(q)$. This implies that:
  * if $\neg\exists z\colon \mathfrak{r}_{j+1}(z) \in \mathbb{N}^{>0}$, then $q_j\ R^{\mu_j}\ p_j$ (IH item 1) tells us that $u'_{j+1} = \bot$.
  * if $\exists z\colon \mathfrak{r}_{j+1}(z) \in \mathbb{N}^{>0}$, then $q_j\ R^{\mu_j}\ p_j$ (IH item 1) tells us that $u'_{j+1} \in \{\mu_{j+1}(z)\} \times \mathbb{N}^{>0}$. The fact that $z \neq y$ implies that $\mu_{j+1}(z) \neq \mu_{j+1}(y)$, which tells us that $u'_{j+1} \notin \{\mu_{j+1}(y)\} \times \mathbb{N}^{>0}$.

  We thus see that in both cases, $u'_{j+1} \notin \{\mu_{j+1}(y)\} \times \mathbb{N}^{>0}$, which implies that:

$$\begin{aligned} \mu_{j+1}(y) &= \mu_j(\mathfrak{r}_{j+1}(y)) && \text{(by definition of } \mathit{updateMap}\text{)} \\ &= \mu_j(x). && (\mathfrak{r}_{j+1}(y) = x) \end{aligned}$$

  Therefore:

$$\begin{aligned} & \mathit{lastStartedAt}^{\mathcal{N}}_{p_{\mathcal{I}}^{\mathcal{N}} \xrightarrow{\mathtt{i}_1 \cdots \mathtt{i}_{j+1}} p_{j+1}}(\mu_{j+1}(y)) \\ &= \mathit{lastStartedAt}^{\mathcal{N}}_{p_{\mathcal{I}}^{\mathcal{N}} \xrightarrow{\mathtt{i}_1 \cdots \mathtt{i}_{j}} p_{j}}(\mu_{j}(x)) && (\mu_{j+1}(y) = \mu_j(x)) \\ &= \mathit{lastStartedAt}^{\mathcal{M}}_{q_{\mathcal{I}} \xrightarrow{\mathtt{i}_1 \cdots \mathtt{i}_{j}} q_{j}}(x) && \text{(IH item 2a)} \\ &= \mathit{lastStartedAt}^{\mathcal{M}}_{q_{\mathcal{I}} \xrightarrow{\mathtt{i}_1 \cdots \mathtt{i}_{j+1}} q_{j+1}}(y) && (\mathfrak{r}_{j+1}(y) = x) \end{aligned},$$

  as required.

(b) If $j + 1 < n$, then we perform a case distinction on $\mathtt{i}_{j+2}$:
  – if $i_{j+2} \in I$, then $i_{j+2} = i'_{j+2} = \mathtt{i}_{j+2}$. Then $\mu_{j+1}(i_{j+2}) = i_{j+2} = i'_{j+2}$,
  – if $i_{j+2} = \mathsf{to}[x]$ for a timer $x \in \mathcal{X}^{\mathcal{M}}(q')$ and $i'_{j+2} = \mathsf{to}[x']$ for a timer $x' \in \mathcal{X}^{\mathcal{N}}(p')$, then we know that x and x' must have been last started at the same indices of their respective runs:

$$\mathit{lastStartedAt}^{\mathcal{N}}_{p_{\mathcal{I}}^{\mathcal{N}} \xrightarrow{i_1 \cdots i_{j+1}} p_{j+1}}(x') = \mathit{lastStartedAt}^{\mathcal{M}}_{q_{\mathcal{I}} \xrightarrow{i_1 \cdots i_{j+1}} q_{j+1}}(x). \tag{C.5}$$

  We can combine this information with that of Item 2a to see that:

$$\begin{aligned} & \mathit{lastStartedAt}^{\mathcal{N}}_{p_{\mathcal{I}}^{\mathcal{N}} \xrightarrow{\mathtt{i}_1 \cdots \mathtt{i}_{j+1}} p_{j+1}}(\mu_{j+1}(x)) \\ &= \mathit{lastStartedAt}^{\mathcal{M}}_{q_{\mathcal{I}} \xrightarrow{\mathtt{i}_1 \cdots \mathtt{i}_{j+1}} q_{j+1}}(x) && \text{(Item 2a)} \\ &= \mathit{lastStartedAt}^{\mathcal{M}}_{q_{\mathcal{I}} \xrightarrow{i_1 \cdots i_{j+1}} q_{j+1}}(x) \\ &= \mathit{lastStartedAt}^{\mathcal{N}}_{p_{\mathcal{I}}^{\mathcal{N}} \xrightarrow{i_1 \cdots i_{j+1}} p_{j+1}}(x') && \text{(Equation (C.5))} \\ &= \mathit{lastStartedAt}^{\mathcal{N}}_{p_{\mathcal{I}}^{\mathcal{N}} \xrightarrow{\mathtt{i}_1 \cdots \mathtt{i}_{j+1}} p_{j+1}}(x') \end{aligned}$$

  We thus know that $\mu_{j+1}(x)$ and x' were last started at the same index of the same run. Since at most one timer can be started in a single transition step, we know that $\mu_{j+1}(x) = x'$. Therefore:

$$\mu_{j+1}(i_{j+2}) = \mu_{j+1}(\mathsf{to}[x]) = \mathsf{to}[\mu_{j+1}(x)] = \mathsf{to}[x'] = i'_{j+2},$$

  as required.

□

C.5.2 Proof of Lemma C.5.2

Proof. We use a proof by induction on the lengths of the symbolic words to show that $\mathcal{M} \simeq \mathcal{N} \Rightarrow \mathcal{M} \approx_{sym} \mathcal{N}$:

- Base case: $\mathtt{w} = \epsilon$. We get the runs $q_\mathcal{I} \xrightarrow{\epsilon} = q_\mathcal{I}$ and $p_\mathcal{I}^\mathcal{N} \xrightarrow{\epsilon} = p_\mathcal{I}^\mathcal{N}$ for $\mathcal{M}$ and $\mathcal{N}$, respectively. Such "empty" runs are feasible for any (g)MMT. The base case holds, since there are no outputs for $\mathtt{w}$ and $\mathtt{w}$ can't start any spannings.
- Inductive step case: Let $\mathtt{w} = \mathtt{i}_1 \dots \mathtt{i}_{n+1}$ be a symbolic word over $\mathtt{A}$. We use the induction hypothesis *(IH)*:

  1. *$\pi = q_\mathcal{I} \xrightarrow[\mathtt{r}_1]{\mathtt{i}_1/o_1} q_1 \dots \xrightarrow[\mathtt{r}_n]{\mathtt{i}_n/o_n} q_n$ is a feasible run in $\mathcal{M}$ iff $\pi' = p_\mathcal{I}^\mathcal{N} \xrightarrow[u'_1]{\mathtt{i}_1/o'_1} p_1 \dots \xrightarrow[u'_n]{\mathtt{i}_n/o'_n} p_n$ is a feasible run in $\mathcal{N}$.*
  2. *If π is feasible for $\mathcal{M}$ and π' is feasible for $\mathcal{N}$, then the following conditions also hold:*
     (a) $o_j = o'_j$ for all $j \in \{1, \dots, n\}$,
     (b) $q_{k-1} \xrightarrow{\mathtt{i}_k \dots \mathtt{i}_j} q_j$ is spanning and $j \leq n \Rightarrow \exists x \colon \mathtt{r}_k(x) = c \wedge u'_k = (x', c') \wedge c = c'$.

  We assume that the runs for $\mathtt{i}_1 \dots \mathtt{i}_n$ (IH item 1) are indeed feasible, as otherwise the runs for $\mathtt{w}$ wouldn't be feasible and there would be nothing for us to show in this step case. Lemma C.5.1 tells us that therefore, $q_n \; R^\mu \; p_n$. Let $\rho = q_\mathcal{I} \xrightarrow{\sigma = i_1 \dots i_n} \in runs(\mathcal{M})$ be the run of $\mathcal{M}$ for which $\overline{\sigma} = \mathtt{i}_1 \dots \mathtt{i}_n$, and let $\rho' = p_\mathcal{I}^\mathcal{N} \xrightarrow{\sigma' = i'_1 \dots i'_n} \in runs(\mathcal{N})$ be the run of $\mathcal{N}$ for which $\overline{\sigma'} = \mathtt{i}_1 \dots \mathtt{i}_n$. We are to determine whether $\pi_+ = q_\mathcal{I} \xrightarrow[\mathtt{r}_1]{\mathtt{i}_1/o_1} q_1 \dots \xrightarrow[\mathtt{r}_{n+1}]{\mathtt{i}_{n+1}/o_{n+1}} q_{n+1}$ is a feasible run in $\mathcal{M}$ iff $\pi'_+ = p_\mathcal{I}^\mathcal{N} \xrightarrow[u'_1]{\mathtt{i}_1/o'_1} p_1 \dots \xrightarrow[u'_{n+1}]{\mathtt{i}_{n+1}/o'_{n+1}} p_{n+1}$ is a feasible run in $\mathcal{N}$. We perform a case distinction on $\mathtt{i}_{n+1}$:

  1. if $\mathtt{i}_{n+1} \in I$, then let $i_{n+1} = i'_{n+1} = \mathtt{i}_{n+1}$. We can see that π_+ is indeed feasible for $\mathcal{M}$ and π'_+ is indeed feasible for $\mathcal{N}$, since the actions are input actions. Therefore, $\mu(i_{n+1}) = i_{n+1} = i'_{n+1}$.
  2. if $\mathtt{i}_{n+1} = \mathtt{to}[k]$ for some $0 < k \leq n$, then let

  $$x_\perp = \begin{cases} timerStartedAt^\mathcal{M}(q_\mathcal{I} \xrightarrow{i_1 \dots i_k}) & \text{if } timerStartedAt^\mathcal{M}(q_\mathcal{I} \xrightarrow{i_1 \dots i_k})\downarrow \\ \perp & \text{otherwise,} \end{cases}$$

  and let:

  $$x'_\perp = \begin{cases} timerStartedAt^\mathcal{N}(p_\mathcal{I}^\mathcal{N} \xrightarrow{i'_1 \dots i'_k}) & \text{if } timerStartedAt^\mathcal{N}(p_\mathcal{I}^\mathcal{N} \xrightarrow{i'_1 \dots i'_k})\downarrow \\ \perp & \text{otherwise.} \end{cases}$$

  We know from Lemma C.5.1 that $\exists \mu_{k-1} \colon q_{k-1} \; R^{\mu_{k-1}} \; p_{k-1}$, which implies that $x_\perp = \perp \Leftrightarrow x'_\perp = \perp$. This means that π (re)sets a timer at index k iff π' (re)sets a timer at index k. Therefore, if π_+ is infeasible in $\mathcal{M}$ because $x_\perp = \perp$, then $x'_\perp = \perp$, making π'_+ infeasible for $\mathcal{N}$. The converse also holds. There is nothing more for us to show in case π_+ or π'_+ is infeasible, so we will assume that $x_\perp \in \mathcal{X}^\mathcal{M}(q_k)$ and $x'_\perp \in \mathcal{X}^\mathcal{N}(p_k)$ for the remainder of this item.
  Let $x = renameTo^\mathcal{M}_{q_k \xrightarrow{i_{k+1} \dots i_n} q_n}(x_\perp)$, and let $x' = x'_\perp$. Even though $x_\perp \in \mathcal{X}^\mathcal{M}(q_k)$, it is still possible for π_+ to be infeasible for $\mathcal{M}$ if $\delta^\mathcal{M}(q_n, \mathtt{to}[x])\uparrow$. Likewise, $\delta^\mathcal{N}(p_n, \mathtt{to}[x'])\uparrow$ would mean that π'_+ is not feasible for $\mathcal{N}$, even though $x'_\perp \in \mathcal{X}^\mathcal{N}(p_k)$. We now show that $\delta^\mathcal{M}(q_n, \mathtt{to}[x])\uparrow \Leftrightarrow \delta^\mathcal{N}(p_n, \mathtt{to}[x'])\uparrow$:
     - We first show by contradiction that if $\delta^\mathcal{M}(q_n, \mathtt{to}[x])\uparrow$, then $\delta^\mathcal{N}(p_n, \mathtt{to}[x'])\uparrow$:
       Assume $\delta^\mathcal{M}(q_n, \mathtt{to}[x])\uparrow$. If $\delta^\mathcal{N}(p_n, \mathtt{to}[x'])\downarrow$, then $q_n \; R^\mu \; p_n$ implies that there exists a timer obtained from $\delta^\mathcal{M}(q_n, \mathtt{to}[y])\downarrow$ such that $\mu(y) = x'$. We know from Lemma C.5.1 that y must then have last been (re)started at the same index k of ρ at which x' was last (re)started in ρ'. But since (g)MMTs can (re)start at most one timer in a single transition, this implies that $y = x$. The contradiction now follows from the fact that $\delta^\mathcal{M}(q_n, \mathtt{to}[y])\downarrow$, while $\delta^\mathcal{M}(q_n, \mathtt{to}[x])\uparrow$. Hence, $\delta^\mathcal{M}(q_n, \mathtt{to}[x])\uparrow \; \Rightarrow \delta^\mathcal{N}(p_n, \mathtt{to}[x'])\uparrow$.

– We use a similar argument to show that $\delta^{\mathcal{N}}(p_n, \mathtt{to}[x'])\uparrow \Rightarrow \delta^{\mathcal{M}}(q_n, \mathtt{to}[x])\uparrow$:
Assume $\delta^{\mathcal{N}}(p_n, \mathtt{to}[x'])\uparrow$. If $\delta^{\mathcal{M}}(q_n, \mathtt{to}[x])\downarrow$, then $q_n \ R^{\mu} \ p_n$ implies that $\delta^{\mathcal{N}}(p_n, \mathtt{to}[\mu(x)])\downarrow$. We know from Lemma C.5.1 that $\mu(x)$ must then have last been (re)started at the same index k of ρ' at which x was last (re)started in ρ. Since (g)MMTs can (re)start at most one timer in a single transition, this implies that $\mu(x) = x'$. The contradiction now follows from the fact that $\delta^{\mathcal{N}}(p_n, \mathtt{to}[\mu(x)])\downarrow$, while $\delta^{\mathcal{N}}(p_n, \mathtt{to}[x'])\uparrow$. Hence, $\delta^{\mathcal{N}}(p_n, \mathtt{to}[x'])\uparrow \Rightarrow \delta^{\mathcal{M}}(q_n, \mathtt{to}[x])\uparrow$.

Therefore, we know that π_+ is a feasible run in $\mathcal{M}$ iff π'_+ is a feasible run in $\mathcal{N}$.
If π_+ is feasible for $\mathcal{M}$ and π'_+ is feasible for $\mathcal{N}$, then let $i_{n+1} = \mathtt{to}[x]$ and $i'_{n+1} = \mathtt{to}[x']$. Then $q_n \ R^{\mu} \ p_n$ implies that $\mu(i_{n+1}) = i'_{n+1}$, per Lemma C.5.1.

We can see that regardless of $\mathtt{i}_{n+1}$'s value, run π_+ is feasible for $\mathcal{M}$ iff run π'_+ is feasible for $\mathcal{N}$. This tells us that IH item 1 holds for the next step.
From this point on, we assume that run π_+ is feasible for $\mathcal{M}$ and that run π'_+ is feasible for $\mathcal{N}$, since otherwise there is nothing more for us to show for this step.
We use the fact that $q_n \ R^{\mu} \ p_n$ to show that the conditions of the induction hypothesis also hold for the next step:

– $q_n \ R^{\mu} \ p_n$ tells us that $\lambda^{\mathcal{M}}(q_n, i_{n+1}) = \lambda^{\mathcal{N}}(p_n, \mu(i_{n+1}))$. Therefore:

$$o_{n+1} = \lambda^{\mathcal{M}}(q_n, i_{n+1}) = \lambda^{\mathcal{N}}(p_n, \mu(i_{n+1})) = \lambda^{\mathcal{N}}(p_n, i'_{n+1}) = o'_{n+1},$$

as required (IH item 2a),

– Regarding the transition for $\mathtt{i}_{n+1}$, there are some relevant cases with regard to spannings:

1. If the $n+1$th transition for $\mathcal{M}$'s run π_+ potentially starts a spanning by (re)starting a timer $x \in \mathcal{X}^{\mathcal{M}}(q_{n+1})$ to a constant $c \in \mathbb{N}^{>0}$, i.e. $\exists x\colon \mathfrak{r}_{n+1}(x) = c \in \mathbb{N}^{>0}$, then $q_n \ R^{\mu} \ p_n$ tells us that:

$$\exists x' \in \mathcal{X}^{\mathcal{N}}(p_{n+1})\colon (u'_{n+1} = (x', c') \wedge c = c'),$$

as required.

2. If $q_{k-1} \xrightarrow{\mathtt{i}_k \dots \mathtt{i}_{n+1}} q_{n+1}$ is spanning, then $\exists x\colon \mathfrak{r}_k(x) = c \in \mathbb{N}^{>0}$. Lemma C.5.1 tells us that since π_+ is a feasible run in $\mathcal{M}$ and π'_+ is a feasible run in $\mathcal{N}$, $\exists \mu_{k-1}\colon q_{k-1} \ R^{\mu_{k-1}} \ p_{k-1}$. This tells us that:

$$\exists x' \in \mathcal{X}^{\mathcal{N}}(p_k)\colon (u'_k = (x', c') \wedge c = c'),$$

as required.

(IH item 2b).

We thus know that the conditions of the induction hypothesis also hold for the next step.

We have thus shown by an induction on the lengths of the symbolic words that $\mathcal{M} \simeq \mathcal{N} \Rightarrow \mathcal{M} \approx_{sym} \mathcal{N}$. □

C.6 Properties and Proofs Related to the k-A-Completeness of the MMT Testing Procedure

The properties and proofs used in Vaandrager et al. [2024] to prove a sufficient condition for the k-A-completeness of Mealy machine test suites form the basis of our proofs for the k-A-completeness of our MMT conformance testing procedure. We highlight the additions we made compared to the work from Vaandrager et al. [2024] in green, and the remaining differences between our results and theirs in blue. We first provide the auxiliary lemmas, before we conclude with our proofs of Theorem 5.8.1 and Lemma 5.8.17.

Lemma C.6.1. Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be an s-learnable MMT, and let $\langle f_s, f_t, f_u \rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional MMT simulation. Let $q_0, q_n \in Q^{\mathcal{T}}$. Let $\sigma = i_1 \dots i_n \in (A^{\mathcal{T}})^*$, and let $\sigma' = f_t(i_1) \dots f_t(i_n) \in (A^{\mathcal{M}})^*$. Then:

$$q_0 \xrightarrow{\sigma} q_n \quad \Longrightarrow \quad f_s(q_0) \xrightarrow{\sigma'} f_s(q_n).$$

Proof. Suppose that $q_0 \xrightarrow{\sigma} q_n \in runs(\mathcal{T})$. Then $q_n = \delta^{\mathcal{T}^*}(q_0, \sigma)$. We need to prove that $\delta^{\mathcal{M}^*}(f_s(q_0), \sigma') = f_s(q_n)$. We prove the property by an induction on the length of σ:

- Base case: $\sigma = \epsilon$. Then $q_n = q_0$ and $\sigma' = \epsilon$. We can see that:

  $$\delta^{\mathcal{M}^*}(f_s(q_0), \sigma') = \delta^{\mathcal{M}^*}(f_s(q_n), \epsilon) = f_s(q_n),$$

  as required.

- Inductive step case: $\sigma = i_1 \dots i_{n+1} \in (A^{\mathcal{T}})^*$. Let $\rho = i_1 \dots i_n$, and let $\rho' = f_t(i_1) \dots f_t(i_n) \in (A^{\mathcal{M}})^*$. We use the induction hypothesis:

  $$q_0 \xrightarrow{\rho} q_n \implies f_s(q_0) \xrightarrow{\rho'} f_s(q_n).$$

  The induction hypothesis tells us that $\delta^{\mathcal{M}^*}(f_s(q_0), \rho') = f_s(q_n)$. Let $q_{n+1} = \delta^{\mathcal{M}}(q_n, i_{n+1})$. The definition of functional MMT simulations tells us that therefore, $\delta^{\mathcal{M}}(f_s(q_n), f_t(i_{n+1})) = f_s(q_{n+1})$. We thus know that $\delta^{\mathcal{M}^*}(f_s(q_0), \rho' \cdot f_t(i_{n+1})) = f_s(q_{n+1})$. We can also see that $\delta^{\mathcal{M}^*}(q_0, \rho \cdot i_{n+1}) = q_{n+1}$. Therefore:

  $$q_0 \xrightarrow{\rho \cdot i_{n+1}} q_{n+1} \implies f_s(q_0) \xrightarrow{\rho' \cdot f_t(i_{n+1})} f_s(q_{n+1}),$$

  as required.

□

Lemma C.6.2. Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be an s-learnable gMMT, and let $\langle f_s, f_t, f_u \rangle \colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Let $q_0, q_n \in Q^{\mathcal{T}}$. Let $\sigma = i_1 \dots i_n \in (A^{\mathcal{T}})^*$, and let $\sigma' = f_t(q_0, i_1) \dots f_t(q_{n-1}, i_n) \in (A^{\mathcal{M}})^*$. Then:

$$q_0 \xrightarrow{\sigma} q_n \implies f_s(q_0) \xrightarrow{\sigma'} f_s(q_n).$$

Proof. Suppose that $q_0 \xrightarrow{\sigma} q_n$. Then $q_n = \delta^{\mathcal{M}^*}(q_0, \sigma)$. We need to prove that $\delta^{\mathcal{M}^*}(f_s(q_0), \sigma') = f_s(q_n)$. We prove the property by an induction on the length of σ:

- Base case: $\sigma = \epsilon$. Then $q_n = q_0$ and $\sigma' = \epsilon$. We can see that:

  $$\delta^{\mathcal{M}^*}(f_s(q_0), \sigma') = \delta^{\mathcal{M}^*}(f_s(q_n), \epsilon) = f_s(q_n),$$

  as required.

- Inductive step case: $\sigma = i_1 \dots i_{n+1} \in (A^{\mathcal{T}})^*$. Let $\rho = i_1 \dots i_n$, and let $\rho' = f_t(q_0, i_1) \dots f_t(q_{n-1}, i_n) \in (A^{\mathcal{M}})^*$. We use the induction hypothesis:

  $$q_0 \xrightarrow{\rho} q_n \implies f_s(q_0) \xrightarrow{\rho'} f_s(q_n).$$

  The induction hypothesis tells us that $\delta^{\mathcal{M}^*}(f_s(q_0), \rho') = f_s(q_n)$. Let $q_{n+1} = \delta^{\mathcal{M}}(q_n, i_{n+1})$. The definition of functional gMMT simulations tells us that therefore, $\delta^{\mathcal{M}}(f_s(q_n), f_t(q_n, i_{n+1})) = f_s(q_{n+1})$. We thus know that $\delta^{\mathcal{M}^*}(f_s(q_0), \rho' \cdot f_t(i_{n+1})) = f_s(q_{n+1})$. We can also see that $\delta^{\mathcal{M}^*}(q_0, \rho \cdot i_{n+1}) = q_{n+1}$. Therefore:

  $$q_0 \xrightarrow{\rho \cdot i_{n+1}} q_{n+1} \implies f_s(q_0) \xrightarrow{\rho' \cdot f_t(q_n, i_{n+1})} f_s(q_{n+1}),$$

  as required.

□

Lemma C.6.3. Let $\mathcal{T}$ be an observation tree MMT with a state $q \in Q^{\mathcal{T}}$, let $\mathcal{M}$ be an s-learnable MMT, and let $\langle f_s, f_t, f_u \rangle \colon \mathcal{T} \to \mathcal{M}$ be a functional MMT simulation. Then:

$$q \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}} \wedge q \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}} \implies$$
$$(\forall x \in \mathcal{X}^{\mathcal{M}}(f_s(q)) \colon \exists (y \in \mathcal{X}^{\mathcal{T}}(q) \colon f_t(y) = x \wedge (f_s(q) \xrightarrow{\mathsf{to}[f_t(y)]} \in runs(\mathcal{M}) \quad \Leftrightarrow \quad q \xrightarrow{\mathsf{to}[y]} \in runs(\mathcal{T})))).$$

Proof. We know from $q \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ that $|\mathcal{X}^{\mathcal{T}}(q)| = |\mathcal{X}^{\mathcal{M}}(f_s(q))|$. Taken with (FMS1) and (FMS2), this tells us that $\forall x \in \mathcal{X}^{\mathcal{M}}(f_s(q))$, there is a unique $y \in \mathcal{X}^{\mathcal{T}}(q) \colon f_t(y) = x$. Let $y \in \mathcal{X}^{\mathcal{T}}(q)$, and let $x = f_t(y) \in \mathcal{X}^{\mathcal{M}}(f_s(q))$. We know from (FMS3) and (FMS4) that if $q \xrightarrow{\mathsf{to}[y]} \in runs(\mathcal{T})$, then $f_s(q) \xrightarrow{\mathsf{to}[f_t(y)]} \in runs(\mathcal{M})$, as required. Therefore, since $\mathcal{M}$ is s-learnable and thus complete, $q \xrightarrow{\mathsf{to}[y]} \in runs(\mathcal{T}) \Rightarrow f_s(q) \xrightarrow{\mathsf{to}[f_t(y)]} \in runs(\mathcal{M}) \Rightarrow f_t(y) \in \mathcal{X}^{\mathcal{M}}_0(f_s(q))$. Since $\mathcal{T}$ is an observation tree, $q \xrightarrow{\mathsf{to}[y]} \in runs(\mathcal{T}) \Leftrightarrow y \in \mathcal{X}^{\mathcal{T}}_0(q)$. We thus know that $y \in \mathcal{X}^{\mathcal{T}}_0(q) \Rightarrow f_t(y) \in \mathcal{X}^{\mathcal{M}}_0(f_s(q))$.
The fact that $q \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ implies that $|\mathcal{X}^{\mathcal{T}}_0(q)| = |\mathcal{X}^{\mathcal{M}}_0(f_s(q))|$. Taken with $y \in \mathcal{X}^{\mathcal{T}}_0(q) \Rightarrow f_t(y) \in \mathcal{X}^{\mathcal{M}}_0(f_s(q))$ and (FMS2), this implies that $y \in \mathcal{X}^{\mathcal{T}}_0(q) \Leftrightarrow f_t(y) \in \mathcal{X}^{\mathcal{M}}_0(f_s(q))$. Suppose $f_s(q) \xrightarrow{\mathsf{to}[f_t(y)]} \in runs(\mathcal{M})$. Then $f_t(y) \in \mathcal{X}^{\mathcal{M}}_0(f_s(q))$, as we showed earlier. Then $y \in \mathcal{X}^{\mathcal{T}}_0(q) \Leftrightarrow f_t(y) \in \mathcal{X}^{\mathcal{M}}_0(f_s(q))$ implies that $y \in \mathcal{X}^{\mathcal{T}}_0(q)$. Since $\mathcal{T}$ is an observation tree and thus t-observable, $y \in \mathcal{X}^{\mathcal{T}}_0(q) \Rightarrow q \xrightarrow{y} \in runs(\mathcal{T})$. Therefore, $f_s(q) \xrightarrow{\mathsf{to}[f_t(y)]} \in runs(\mathcal{M})$ implies $q \xrightarrow{\mathsf{to}[y]} \in runs(\mathcal{T})$, as required.
We have thus shown that $f_s(q) \xrightarrow{\mathsf{to}[f_t(y)]} \in runs(\mathcal{M}) \Leftrightarrow q \xrightarrow{\mathsf{to}[y]} \in runs(\mathcal{T})$, as required. □

Lemma C.6.4. Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be an s-learnable MMT, and let $\langle f_s, f_t, f_u \rangle \colon \mathcal{T} \to \mathcal{M}$ be a functional MMT simulation. Let $q_0, q_n \in Q^{\mathcal{T}}$. Let $\sigma = i_1 \dots i_n \in (A^{\mathcal{T}})^*$, and let $\sigma' = f_t(i_1) \dots f_t(i_n) \in (A^{\mathcal{M}})^*$. Let $\pi = q_0 \xrightarrow{\sigma} q_n \in runs(\mathcal{T})$. Suppose that the basis of a stratification of $Q^{\mathcal{T}}$ is complete, and that for all $l \in \{0, \dots, n\}$:

- $q_l \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$, and
- if $q_l \in F^j$, then F^j is complete.

Then:

$$f_s(q_0) \xrightarrow{\sigma'} f_s(q_n) \quad \implies \quad q_0 \xrightarrow{\sigma} \in runs(\mathcal{T}).$$

Proof. Suppose that $f_s(q_0) \xrightarrow{\sigma'} f_s(q_n) \in runs(\mathcal{M})$. Then $f_s(q_n) = \delta^{\mathcal{M}^*}(f_s(q_0), \sigma')$. We need to prove that $\delta^{\mathcal{T}^*}(q_0, \sigma)\downarrow$. We prove the property by an induction on the length of σ':

- Base case: $\sigma' = \epsilon$. Then $\sigma = \epsilon$. We trivially get that:

  $$q_0 \xrightarrow{\sigma} = q_0 \xrightarrow{\epsilon} = q_0 \in runs(\mathcal{T}),$$

  as required.

- Inductive step case: $\sigma' = f_t(i_1) \dots f_t(i_{k+1}) \in (A^{\mathcal{M}})^*$ and $k + 1 \leq n$, for some $k \in \mathbb{N}$. Let $\rho' = f_t(i_1) \dots f_t(i_k) \in (A^{\mathcal{M}})^*$. Condition (FMS2) tells us that action sequence $\rho = i_1 \dots i_k \in (A^{\mathcal{T}})^*$ is unique. We use the induction hypothesis:

  $$f_s(q_0) \xrightarrow{\rho'} f_s(q_k) \quad \implies \quad q_0 \xrightarrow{\rho} \in runs(\mathcal{T}).$$

  The induction hypothesis tells us that $q_0 \xrightarrow{\rho} \in runs(\mathcal{T})$. Let $q_k = \delta^{\mathcal{T}^*}(q_0, \rho)$. We know from our final assumption that whether q_k is in the basis or in a frontier, this basis or frontier is complete. This implies in either case that $q_k \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$, and that:

  - if $f_t(i_{k+1}) \in I$, then $i_{k+1} \in I$. The fact that q_k is in a complete basis or frontier now implies that $q_k \xrightarrow{i_{k+1}} \in runs(\mathcal{T})$. Therefore, $q_0 \xrightarrow{i_0 \dots i_{k+1}} \in runs(\mathcal{T})$, as required.

– if $f_t(i_{k+1}) \in TO(\mathcal{X}^{\mathcal{M}}(f_s(q_k)))$, then let $f_t(i_{k+1}) = \mathsf{to}[x]$. Since $q_k \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ and $q_k \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$, Lemma C.6.3 tells us that $\exists y\colon f_t(y) = x \wedge q_k \xrightarrow{\mathsf{to}[y]} \in runs(\mathcal{T})$. This tells us that $i_{k+1} = \mathsf{to}[y]$, and that $q_k \xrightarrow{i_{k+1}} \in runs(\mathcal{T})$. Therefore, $q_0 \xrightarrow{i_0 \dots i_{k+1}} \in runs(\mathcal{T})$, as required.

□

Lemma C.6.5. Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be an s-learnable MMT, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional MMT simulation. Let $q_{\mathcal{I}}, q_n \in Q^{\mathcal{T}}$. Let $\mathtt{w} = \mathtt{i}_1 \dots \mathtt{i}_n \in I \cup TO(\mathbb{N}^{>0})$. Then:

$$q_{\mathcal{I}} \xrightarrow{\mathtt{w}} q_n \quad \Longrightarrow \quad f_s(q_{\mathcal{I}}) \xrightarrow{\mathtt{w}} f_s(q_n).$$

Proof. Let $\sigma = i_1 \dots i_n$ such that $\overline{\sigma} = \mathtt{w}$, and let $\sigma' = f_t(i_1) \dots f_t(i_n) \in (A^{\mathcal{M}})^*$. Suppose $q_{\mathcal{I}} \xrightarrow{\mathtt{w}} q_n$. This implies that $q_{\mathcal{I}} \xrightarrow{\mathtt{w}}$ is feasible in $\mathcal{T}$. We thus know that $\pi = q_{\mathcal{I}} \xrightarrow{\sigma} q_n \in runs(\mathcal{T})$. Thus, by Lemma C.6.1, $\pi' = f_s(q_{\mathcal{I}}) \xrightarrow{\sigma'} f_s(q_n) \in runs(\mathcal{M})$. We know from Lemma 5.3.1 that any spanning sub-runs of π are matched in π', and from (FMS5) that any spanning sub-runs of π' are matched in π. Therefore, since π and π' are runs from the initial states $q^{\mathcal{T}}_{\mathcal{I}}$ and $f_s(q^{\mathcal{T}}_{\mathcal{I}}) = q_{\mathcal{I}}$ of respectively $\mathcal{T}$ and $\mathcal{M}$, $f_s(q_{\mathcal{I}}) \xrightarrow{\mathtt{w}} f_s(q_n)$. □

Lemma C.6.6. Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be an s-learnable gMMT, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Let $q_{\mathcal{I}}, q_n \in Q^{\mathcal{T}}$. Let $\mathtt{w} = \mathtt{i}_1 \dots \mathtt{i}_n \in I \cup TO(\mathbb{N}^{>0})$. Then:

$$q_{\mathcal{I}} \xrightarrow{\mathtt{w}} q_n \quad \Longrightarrow \quad f_s(q_{\mathcal{I}}) \xrightarrow{\mathtt{w}} f_s(q_n).$$

Proof. Let $\sigma = i_1 \dots i_n$ such that $\overline{\sigma} = \mathtt{w}$, and let $\sigma' = f_t(q_0, i_1) \dots f_t(q_{n-1}, i_n) \in (A^{\mathcal{M}})^*$. Suppose $q_{\mathcal{I}} \xrightarrow{\mathtt{w}} q_n$. This implies that $q_{\mathcal{I}} \xrightarrow{\mathtt{w}}$ is feasible in $\mathcal{T}$. We thus know that $\pi = q_{\mathcal{I}} \xrightarrow{\sigma} q_n \in runs(\mathcal{T})$. Thus, by Lemma C.6.2, $\pi' = f_s(q_{\mathcal{I}}) \xrightarrow{\sigma'} f_s(q_n) \in runs(\mathcal{M})$. We know from Lemma 5.3.2 that any spanning sub-runs of π are matched in π', and from (FGS5) that any spanning sub-runs of π' are matched in π. Therefore, since π and π' are runs from the initial states $q^{\mathcal{T}}_{\mathcal{I}}$ and $f_s(q^{\mathcal{T}}_{\mathcal{I}}) = q_{\mathcal{I}}$ of respectively $\mathcal{T}$ and $\mathcal{M}$, $f_s(q_{\mathcal{I}}) \xrightarrow{\mathtt{w}} f_s(q_n)$. □

Lemma C.6.7. Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be an s-learnable gMMT, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Let C be a prefix-closed state cover for $\mathcal{M}$, and let B be the basis of a stratification of $Q^{\mathcal{T}}$ induced by $\overline{C}$. Then f_s restricted to B is injective.

Proof. Let $r, r' \in B$, such that $r \neq r'$. Let $\sigma = \mathsf{access}(r)$ and $\rho = \mathsf{access}(r')$. Then by $r \neq r'$, $\sigma \neq \rho$. Since $\sigma, \rho \in \mathsf{access}(B)$, $\overline{\sigma}, \overline{\rho} \in \overline{C}$. The fact that C is a minimal prefix-closed state cover for $\mathcal{M}$ implies that since $\overline{\sigma} \neq \overline{\rho}$, $\delta^{\mathcal{M}^*}(\overline{\sigma}) \neq \delta^{\mathcal{M}^*}(\overline{\rho})$. Thus, by Lemma C.6.6, $f_s(r) = f_s(\delta^{\mathcal{T}^*}(\overline{\sigma})) = \delta^{\mathcal{M}^*}(\overline{\sigma})$ and $f_s(r') = f_s(\delta^{\mathcal{T}^*}(\overline{\rho})) = \delta^{\mathcal{M}^*}(\overline{\rho})$. Therefore, since $\delta^{\mathcal{M}^*}(\overline{\sigma}) \neq \delta^{\mathcal{M}^*}(\overline{\rho})$, $f_s(r) \neq f_s(r')$, as required. □

Lemma C.6.8. Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be a minimal s-learnable gMMT, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Let C be a minimal and prefix-closed state cover for $\mathcal{M}$, and let B be the basis of a stratification of $Q^{\mathcal{T}}$ induced by $\overline{C}$. Then f_s restricted to B is a bijection.

Proof. Lemma C.6.7 tells us that f_s restricted to B is injective. Since C is minimal, $|B| = |Q^{\mathcal{M}}|$. We may thus conclude that f_s is a bijection between B and $Q^{\mathcal{M}}$. □

Lemma C.6.9. Let $\mathcal{T}$ be an observation tree, let $\mathcal{M}$ be an s-learnable gMMT, and let $\langle f_s, f_t, f_u\rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Let C be a minimal and prefix-closed state cover for $\mathcal{M}$, and let B be the basis of a stratification of $Q^{\mathcal{T}}$ induced by $\overline{C}$. Let $q \in Q^{\mathcal{T}}$. Then:

$$\exists r \in B\colon \qquad r \in \mathcal{C}(q) \quad \wedge \quad f_s(q) = f_s(r).$$

Proof. Let $f_s(q) = u$. Lemma C.6.8 tells us that f_s restricted to B is a bijection. Let $r \in B$ be the unique state with $f_s(r) = u$. Since $f_s(q) = f_s(r)$, Lemma C.3.2 implies that q and r are not apart for all maximal matchings. Hence $r \in \mathcal{C}(q)$. □

C.6.1 The proof of Theorem 5.8.1

Proof. Lemma 5.8.1 tells us that $\mathcal{M} \notin \mathcal{U}^{\overline{C}}$. Therefore, $\mathcal{M} \in \mathcal{U}_k^{\overline{C}}$. The fact that B is a basis induced by a minimal prefix-closed state cover $\overline{C}$ implies that $\overline{\mathsf{access}(B)} = \overline{C}$. We thus know that since $\mathcal{M} \in \mathcal{U}_k^{\overline{C}}$, there are:

$$\forall q \in Q^{\mathcal{M}}\colon \qquad \exists \mathtt{w} \in (\overline{\mathsf{access}(B)} = \overline{C}), \exists \sigma \in (A^{\mathcal{M}})^{\leq k}\colon \delta^{\mathcal{M}^*}(\mathtt{w}) \cdot \sigma. \tag{C.6}$$

Let $\langle f_s, f_t, f_u \rangle\colon \mathcal{T} \to \mathcal{S}$ be a functional gMMT simulation, and let $\langle g_s, g_t, g_u \rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional MMT simulation. We define a relation $R \subseteq \{(s, q, \mu) \mid \forall s \in Q^{\mathcal{S}}, q \in Q^{\mathcal{M}}, \mu\colon (\mathcal{X}^{\mathcal{S}}(s) \to \mathcal{X}^{\mathcal{M}}(q))\}$:

$$(s, q, \mu) \in R \Leftrightarrow \exists t \in B \cup F^{<k}\colon f_s(t) = s \wedge g_s(t) = q,$$

where $\mu = \{(x, x') \in \mathcal{X}^{\mathcal{S}}(s) \times \mathcal{X}^{\mathcal{M}}(q) \mid \exists y \in \mathcal{X}^{\mathcal{T}}(t)\colon f_t(t, y) = x \wedge g_t(y) = x'\}$.
We claim that R is a bisimulation between $\mathcal{S}$ and $\mathcal{M}$.

1. Condition (FGS0) tells us that $f_s(q_I^{\mathcal{T}}) = s_I^{\mathcal{S}}$, and condition (FMS0) tells us that $g_s(q_I^{\mathcal{T}}) = q_I^{\mathcal{M}}$. Since $q_I^{\mathcal{T}} \in B$, this implies that $s_I^{\mathcal{S}}\ R^{\emptyset}\ q_I^{\mathcal{M}}$ (B0), as required.

2. Suppose that $s\ R^{\mu}\ q$. Then there exists a state $t \in B \cup F^{<k}$ such that $f_s(t) = s$ and $g_s(t) = q$.

   - Let $x, y \in \mathcal{X}^{\mathcal{S}}(s)\colon x \neq y$. Then $t \in \mathcal{A}_{\mathcal{S}}^{\mathcal{T}}$ implies that there exist timers $x_0, y_0 \in \mathcal{X}^{\mathcal{T}}(t)\colon f_t(t, x_0) = x \wedge f_t(t, y_0) = y$. The fact that $x \neq y$ now implies that $x_0 \neq y_0$, which implies that $g_t(x_0) \neq g_t(y_0)$ per (FMS2). The definition of μ tells us that since $\mu(x) = g_t(x_0)$ and $\mu(y) = g_t(y_0)$, $\mu(x) \neq \mu(y)$ (B1).
   - Let $x' \in \mathcal{X}^{\mathcal{M}}(q)$. Then $t \in \mathcal{A}_{\mathcal{M}}^{\mathcal{T}}$ tells us that $\exists y \in \mathcal{X}^{\mathcal{T}}(t)\colon g_t(y) = x'$. Condition (FGS1) tells us that $f_t(t, y) \in \mathcal{X}^{\mathcal{S}}(s)$. The definition of μ tells us that therefore, $\mu(f_t(t, y)) = g_t(y) = x'$ (B2).
   - Let $i \in I \cup TO(\mathcal{X}_0^{\mathcal{T}}(t))$, let $i^{\mathcal{S}} = f_t(t, i)$, and let $i^{\mathcal{M}} = g_t(i)$. Let $t' = \delta^{\mathcal{T}}(t, i)$, let $s' = \delta^{\mathcal{S}}(s, i^{\mathcal{S}})$ when $\delta^{\mathcal{S}}(s, i^{\mathcal{S}})\downarrow$, and let $q' = \delta^{\mathcal{M}}(q, i^{\mathcal{M}})$ when $\delta^{\mathcal{M}}(q, i^{\mathcal{M}})\downarrow$.
     - Since $t \in B \cup F^{<k}$ and the first k frontiers are complete, we know that $t \in \mathcal{E}_{\mathcal{S}}^{\mathcal{T}}$ and $t \in \mathcal{E}_{\mathcal{M}}^{\mathcal{T}}$. This implies that $|\mathcal{X}_0^{\mathcal{S}}(s)| = |\mathcal{X}_0^{\mathcal{T}}(t)| = |\mathcal{X}_0^{\mathcal{M}}(q)|$.
       For all timers $x \in \mathcal{X}^{\mathcal{S}}(s)$, $t \in \mathcal{A}_{\mathcal{S}}^{\mathcal{T}}$ and conditions (FGS1) and (FGS2) tell us that there exists exactly one timer $y \in \mathcal{X}^{\mathcal{T}}(t)\colon f_t(t, y) = x$. The fact that $\mathcal{S}$ and $\mathcal{T}$ are t-observable therefore implies that, if $\delta^{\mathcal{S}}(s, \mathsf{to}[x])\downarrow$, then (FGS2), (FGS3), (FGS4) and $|\mathcal{X}_0^{\mathcal{S}}(s)| = |\mathcal{X}_0^{\mathcal{T}}(t)|$ additionally tell us that $\delta^{\mathcal{T}}(t, \mathsf{to}[y])\downarrow$. Conditions (FMS3) and (FMS4) now tell us that since $\delta^{\mathcal{T}}(t, \mathsf{to}[y])\downarrow$, $\delta^{\mathcal{M}}(q, \mathsf{to}[g_t(y)])\downarrow$. We thus know that $\delta^{\mathcal{S}}(s, i^{\mathcal{S}})\downarrow \Rightarrow \delta^{\mathcal{M}}(q, i^{\mathcal{M}})\downarrow$.
       For all timers $x \in \mathcal{X}^{\mathcal{M}}(q)$, $t \in \mathcal{A}_{\mathcal{M}}^{\mathcal{T}}$ and conditions (FMS1) and (FMS2) tells us that there exists exactly one timer $y \in \mathcal{X}^{\mathcal{T}}(t)\colon g_t(y) = x$. The fact that $\mathcal{M}$ and $\mathcal{T}$ are t-observable therefore implies that, if $\delta^{\mathcal{M}}(q, \mathsf{to}[x])\downarrow$, then (FMS2), (FMS3), (FMS4) and $|\mathcal{X}_0^{\mathcal{M}}(q)| = |\mathcal{X}_0^{\mathcal{T}}(t)|$ additionally tell us that $\delta^{\mathcal{T}}(t, \mathsf{to}[y])\downarrow$. Conditions (FGS3) and (FGS4) now tell us that since $\delta^{\mathcal{T}}(t, \mathsf{to}[y])\downarrow$, $\delta^{\mathcal{S}}(s, \mathsf{to}[f_t(t, y)])\downarrow$. We thus know that $\delta^{\mathcal{M}}(q, i^{\mathcal{M}})\downarrow \Rightarrow \delta^{\mathcal{S}}(s, i^{\mathcal{S}})\downarrow$.
       Therefore, $\delta^{\mathcal{S}}(s, i^{\mathcal{S}})\downarrow \Leftrightarrow \delta^{\mathcal{M}}(q, i^{\mathcal{M}})\downarrow$ (B5), as required.
     - Then $\mu(i^{\mathcal{S}}) = i^{\mathcal{M}}$, since:
       - if $i \in I$, then $f_t(t, i) = i$ and $g_t(i) = i$. Then $i^{\mathcal{S}} = f_t(t, i) = i \in I$. Since $i^{\mathcal{S}} \in I$:

         $$\mu(i^{\mathcal{S}}) = i^{\mathcal{S}} = i = g_t(i) = i^{\mathcal{M}}.$$

       - if $i \in TO(\mathcal{X}_0^{\mathcal{T}}(t))$, then $i = \mathsf{to}[x]$ for some timer $x \in \mathcal{X}_0^{\mathcal{T}}(t)$. We get:

         $$i^{\mathcal{S}} = f_t(t, i) = f_t(t, \mathsf{to}[x]) = \mathsf{to}[f_t(t, x)],$$

and:

$$i^{\mathcal{M}} = g_t(i) = g_t(\mathsf{to}[x]) = \mathsf{to}[g_t(x)].$$

Therefore:

$$\mu(i^{\mathcal{S}}) = \mu(\mathsf{to}[f_t(t,x)]) = \mathsf{to}[\mu(f_t(t,x))] = \mathsf{to}[g_t(x)] = i^{\mathcal{M}}.$$

– Conditions (FGS3) and (FGS4) tell us that $\lambda^{\mathcal{T}}(t,i) = \lambda^{\mathcal{S}}(s,i^{\mathcal{S}})$, and conditions (FMS3) and (FMS4) tell us that $\lambda^{\mathcal{T}}(t,i) = \lambda^{\mathcal{M}}(q,i^{\mathcal{M}})$. Therefore:

$$\lambda^{\mathcal{S}}(s,i^{\mathcal{S}}) = \lambda^{\mathcal{T}}(t,i) = \lambda^{\mathcal{M}}(q,i^{\mathcal{M}}) = \lambda^{\mathcal{M}}(q,\mu(i^{\mathcal{S}})),$$

as required for (B3b) and (B4b).

– We know from the fact that $t \in B \cup F^{<k}$ that $t' \in \mathcal{A}^{\mathcal{T}}_{\mathcal{S}}$ and $t' \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$. We know from $t' \in \mathcal{A}^{\mathcal{T}}_{\mathcal{S}}$ that t (re)sets a timer x for action i iff s (re)sets the timer $f_t(t',x)$ for action $i^{\mathcal{S}}$. Similarly, we know from the fact that $t' \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ that t (re)sets a timer x for action i iff q (re)sets the timer $g_t(x)$ for action $i^{\mathcal{M}}$. We thus know that s (re)sets the timer $f_t(t',x)$ for $i^{\mathcal{S}}$ iff q (re)sets the timer $g_t(x)$ for $i^{\mathcal{M}}$ ((B3a), (B4a), (B6) and (B7)), as required.

– We know from the previous item that s (re)sets the timer $f_t(t',x)$ for $i^{\mathcal{S}}$ iff t (re)sets the timer x for i iff q (re)sets the timer $g_t(x)$ for $i^{\mathcal{M}}$. Condition (FGS3) specifies that if t (re)sets x to c for i, then s (re)sets $f_t(t',x)$ to c for $i^{\mathcal{S}}$. Similarly, condition (FMS3) tells us that when t (re)sets x to c for i, then q (re)sets $g_t(x)$ to c for $i^{\mathcal{M}}$. We thus know that s (re)sets $f_t(t',x)$ to c for $i^{\mathcal{S}}$ iff q (re)sets $g_t(x)$ to c for $i^{\mathcal{M}}$ (B4c), as required.

– Conditions (FGS3) and (FGS4) tell us that:

$$f_s(t') = \delta^{\mathcal{S}}(f_s(t), f_t(t,i)) = \delta^{\mathcal{S}}(s,i^{\mathcal{S}}) = s',$$

and conditions (FMS3) and (FMS4) tell us that:

$$g_s(t') = \delta^{\mathcal{M}}(g_s(t), g_t(i)) = \delta^{\mathcal{M}}(q,i^{\mathcal{M}}) = q'.$$

In order to prove that there is a mapping μ' such that $s' \; R^{\mu'} \; q'$, we first show that there exists a state $t_1 \in B \cup F^{<k}$ such that $f_s(t_1) = s'$ and $g_s(t_1) = q'$, as required. We consider two cases:

(a) if $t' \in B \cup F^{<k}$, then we already know that $f_s(t') = s'$ and $g_s(t') = q'$.

(b) $t' \in F^k$. Equation (C.6) tells us that there are $\mathtt{w} \in \overline{\mathsf{access}(B)}$ and $\sigma \in (A^{\mathcal{M}})^{\leq k}$ such that $\delta^{\mathcal{M}^*}(\delta^{\mathcal{M}^*}(\mathtt{w}),\sigma) = q'$. Since $\delta^{\mathcal{M}^*}(\mathtt{w})\!\downarrow$, $q^{\mathcal{M}}_{\mathcal{I}} \xrightarrow{\mathtt{w}}$ is feasible in $\mathcal{M}$, which implies that $\mathit{symWordToAcSeq}^{\mathcal{M}}(\mathtt{w})\!\downarrow$. Let $\rho = i_1 \dots i_n \in (A^{\mathcal{T}})^*$ be the unique action sequence of $\mathcal{T}$ for which $g_t(i_1)\dots g_t(i_n) = \mathit{symWordToAcSeq}^{\mathcal{M}}(\mathtt{w}) \cdot \sigma$. By the assumption that $B, F^{<k}$ are all complete and $B \cup F^{<k} \subseteq \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$, Lemma C.6.4 tells us that $t'' = \delta^{\mathcal{T}^*}(q^{\mathcal{T}}_{\mathcal{I}},\rho)$ is defined. By Lemma C.6.1, $g_s(t'') = q'$. Then, by Lemma C.3.1, t' and t'' are not apart. We claim that t' and t'' have the same candidate set:

i. $t'' \in B$. Then since B is a basis and all basis states are identified, $\mathcal{C}(t'') = \{t''\}$. Since $\neg(t' \# t'')$ and t' is identified, $\mathcal{C}(t') = \{t''\}$. Hence, $\mathcal{C}(t') = \mathcal{C}(t'')$.

ii. $t'' \in F^{<k}$. Then by Equation (5.1) and since $\neg(t' \# t'')$, $\mathcal{C}(t') = \mathcal{C}(t'')$.

Since t' is identified, $\mathcal{C}(t') = \{r\}$, for some $r \in B$. By Lemma C.6.9, $f_s(t') = f_s(r)$. Since $\mathcal{C}(t') = \mathcal{C}(t'')$, $\mathcal{C}(t'') = \mathcal{C}(t') = \{r\}$. Applying Lemma C.6.9 now tells us that $f(t'') = f(r)$. Hence $f(t'') = f(t') = s'$.

In both cases, we know that there exists a state $t_1 \in B \cup F^{<k}$ such that $f_s(t_1) = s' = f_s(t')$ and $g_s(t_1) = q' = g_s(t')$.

For all $x \in \mathcal{X}^{\mathcal{T}}(t')$, (FMS1) tells us that $g_t(x) \in \mathcal{X}^{\mathcal{M}}(g_s(t'))$. We know from $g_s(t_1) = g_s(t')$ that $\mathcal{X}^{\mathcal{M}}(g_s(t_1)) = \mathcal{X}^{\mathcal{M}}(g_s(t'))$. Since $t_1 \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$, we know that $|\mathcal{X}^{\mathcal{T}}(t_1)| = |\mathcal{X}^{\mathcal{M}}(g_s(t_1))|$. This implies that $\forall y' \in \mathcal{X}^{\mathcal{M}}(g_s(t_1)) \colon \exists y \in \mathcal{X}^{\mathcal{T}}(t_1) \colon g_t(y) = y'$, per (FMS1) and (FMS2). Condition (FMS2) further adds that for all $y' \in \mathcal{X}^{\mathcal{M}}(g_s(t_1))$, there is exactly one timer $y \in \mathcal{X}(t_1)$ such

that $g_t(y) = y'$. Since $\mathcal{X}^{\mathcal{M}}(g_s(t_1)) = \mathcal{X}^{\mathcal{M}}(g_s(t'))$, we know that $g_t(x) \in \mathcal{X}^{\mathcal{M}}(g_s(t_1))$. This thus implies that there exists a timer $y \in \mathcal{X}^{\mathcal{T}}(t_1)$ such that $g_t(y) = g_t(x)$. Condition (FMS2) tells us that $y = x$, which implies that $x \in \mathcal{X}^{\mathcal{T}}(t_1)$. We thus know that $\mathcal{X}^{\mathcal{T}}(t') \subseteq \mathcal{X}^{\mathcal{T}}(t_1)$. This property also holds in the other direction, since $t' \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$. We therefore know that $\mathcal{X}^{\mathcal{T}}(t_1) \subseteq \mathcal{X}^{\mathcal{T}}(t')$. These two facts combine to tell us that:

$$\mathcal{X}^{\mathcal{T}}(t_1) = \mathcal{X}^{\mathcal{T}}(t'). \tag{C.7}$$

Equation (C.7), (FGS6) and $f_s(t_1) = f_s(t')$ tell us that:

$$\forall x \in \mathcal{X}^{\mathcal{T}}(t') \colon f_t(t_1, x) = f_t(t', x). \tag{C.8}$$

We already established that s (re)starts a timer for $i^{\mathcal{S}}$ iff q (re)starts a timer for $i^{\mathcal{M}}$. We now perform a case distinction on whether timers are (re)started in these transitions, in order to show that in both cases, the timer map $\mu' = \mathit{updateMapFrom}^{\mathcal{S},\mu}_{\mathcal{M},i^{\mathcal{S}}}(s,q)$ is a timer map for which $(s', q', \mu') \in R$:

* If no timers are (re)started in the transitions for $i^{\mathcal{S}}$ and $i^{\mathcal{M}}$, then:

$$\begin{aligned}
&\mu' \\
&= \mathit{updateMap}^{\mathcal{S},\mu}_{\mathcal{M},\mathfrak{r}}(s', q') \in (\mathcal{X}^{\mathcal{S}}(s') \times \mathcal{X}^{\mathcal{M}}(q')) && \text{(since no timers are started)} \\
&= \{(y', y) \in (\mathcal{X}^{\mathcal{S}}(s') \times \mathcal{X}^{\mathcal{M}}(q')) \mid y = \mu(\mathfrak{r}(y'))\} && \text{(by definition of } \mathit{updateMap}\text{)} \\
&= \{(y', y) \mid \exists z \in \mathcal{X}^{\mathcal{T}}(t) \colon f_t(t, z) = \mathfrak{r}(y') \wedge g_t(z) = y\}. && \text{(by definition of } \mu\text{)}
\end{aligned}$$

Since $\mathsf{dom}(\mathfrak{r}) = \mathcal{X}^{\mathcal{S}}(f_s(t'))$ per Rule 4.10 and $f_s(t_1) = f_s(t')$, we know that $\mathsf{dom}(\mathfrak{r}) = \mathcal{X}^{\mathcal{S}}(f_s(t_1))$. We thus get from (FGS4) and Equation (C.8) that $\forall z \in \mathcal{X}^{\mathcal{T}}(t') \colon f_t(t, z) = \mathfrak{r}(f_t(t', z)) = \mathfrak{r}(f_t(t_1, z))$. We thus know that:

$$\begin{aligned}
&\{(y', y) \mid \exists z \in \mathcal{X}^{\mathcal{T}}(t) \colon f_t(t, z) = \mathfrak{r}(y') \wedge g_t(z) = y\} \\
&= \{(y', y) \mid \exists z \in \mathcal{X}^{\mathcal{T}}(t) \colon f_t(t, z) = \mathfrak{r}(y') = \mathfrak{r}(f_t(t_1, z)) \wedge g_t(z) = y\}.
\end{aligned}$$

Rule 4.10 requires that $\mathfrak{r}$ is injective, which tells us that in the above, $y' = f_t(t_1, z)$. The fact that $\mathfrak{r}(f_t(t_1, z))\downarrow$ implies that $f_t(t_1, z) \in \mathcal{X}^{\mathcal{S}}(f_s(t_1))$. The facts that $t_1 \in \mathcal{A}^{\mathcal{T}}_{\mathcal{S}}$ and (FGS1) now imply that $\exists z' \in \mathcal{X}^{\mathcal{T}}(t_1) \colon f_t(t_1, z') = f_t(t_1, z)$. If $z \neq z'$, then (FGS2) tells us that $f_t(t_1, z') \neq f_t(t_1, z)$. Therefore, $z' = z$, which implies that $z \in \mathcal{X}^{\mathcal{T}}(t_1)$. This fact, together with the fact that $y' = f_t(t_1, z)$, implies that:

$$\begin{aligned}
&\{(y', y) \mid \exists z \in \mathcal{X}^{\mathcal{T}}(t) \colon f_t(t, z) = \mathfrak{r}(y') = \mathfrak{r}(f_t(t_1, z)) \wedge g_t(z) = y\} \\
&= \{(y', y) \in \mathcal{X}^{\mathcal{S}}(s') \times \mathcal{X}^{\mathcal{M}}(q') \mid \exists z \in \mathcal{X}^{\mathcal{T}}(t_1) \colon f_t(t_1, z) = y' \wedge g_t(z) = y\}.
\end{aligned}$$

Since $f_s(t_1) = s'$ and $g_s(t_1) = q'$, we know that $(s', q', \mu') \in R$. This means that $s' \; R^{\mu'} \; q'$, as required for (B3c).

* If s (re)sets the timer $f_t(t', x)$ for $i^{\mathcal{S}}$ and q (re)sets the timer $g_t(x)$ for $i^{\mathcal{M}}$, then:

$$\begin{aligned}
&\mu' \\
&= \mathit{updateMap}^{\mathcal{S},\mu}_{\mathcal{M},\mathfrak{r},(f_t(t',x),g_t(x))}(s', q') \in (\mathcal{X}^{\mathcal{S}}(s') \times \mathcal{X}^{\mathcal{M}}(q')) \\
&= \{(y', y) \mid y' \neq f_t(t', x) \wedge y = \mu(\mathfrak{r}(y'))\} \cup \{(f_t(t', x), g_t(x))\} \\
&= \{(y', y) \mid \exists z \in \mathcal{X}(t) \colon y' \neq f_t(t', x) \wedge f_t(t, z) = \mathfrak{r}(y') \wedge g_t(z) = y\} \cup \\
&\qquad \{(f_t(t', x), g_t(x))\}.
\end{aligned}$$

The first steps are the same as the ones in the case in which no timers were (re)set:

$$\begin{aligned}
&\{(y', y) \mid \exists z \in \mathcal{X}(t)\colon y' \neq f_t(t', x) \wedge f_t(t, z) = \mathfrak{r}(y') \wedge g_t(z) = y\} \cup \\
&\qquad \{(f_t(t', x), g_t(x))\} \\
&= \{(y', y) \mid \exists z \in \mathcal{X}(t)\colon y' \neq f_t(t', x) \wedge f_t(t, z) = \mathfrak{r}(y') = \mathfrak{r}(f_t(t_1, z)) \wedge g_t(z) = y\} \cup \\
&\qquad \{(f_t(t', x), g_t(x))\} \\
&= \{(y', y) \mid \exists z' \in \mathcal{X}^{\mathcal{T}}(t_1)\colon y' \neq f_t(t', x) \wedge f_t(t_1, z') = y' \wedge g_t(z') = y\} \cup \\
&\qquad \{(f_t(t', x), g_t(x))\}.
\end{aligned}$$

We now integrate $\{(f_t(t', x), g_t(x))\}$ into the rest of the expression:

$$\begin{aligned}
&\{(y', y) \mid \exists z' \in \mathcal{X}^{\mathcal{T}}(t_1)\colon y' \neq f_t(t', x) \wedge f_t(t_1, z') = y' \wedge g_t(z') = y\} \cup \\
&\qquad \{(f_t(t', x), g_t(x))\} \\
&= \{(y', y) \mid \exists z' \in \mathcal{X}^{\mathcal{T}}(t_1)\colon y' \neq f_t(t', x) \wedge f_t(t_1, z') = y' \wedge g_t(z') = y\} \cup \\
&\qquad \{(y', y) \mid f_t(t', x) = y' \wedge g_t(x) = y\} \\
&= \{(y', y) \mid \exists z' \in \mathcal{X}^{\mathcal{T}}(t_1)\colon y' \neq f_t(t_1, x) \wedge f_t(t_1, z') = y' \wedge g_t(z') = y\} \cup \qquad \text{(Equation (C.8))} \\
&\qquad \{(y', y) \mid f_t(t_1, x) = y' \wedge g_t(x) = y\} \\
&= \{(y', y) \mid \exists z' \in (\mathcal{X}^{\mathcal{T}}(t_1) \setminus \{x\})\colon f_t(t_1, z') = y' \wedge g_t(z') = y\} \cup \\
&\qquad \{(y', y) \mid \exists z' \in \{x\} \wedge f_t(t_1, z') = y' \wedge g_t(z') = y\} \\
&= \{(y', y) \in \mathcal{X}^{\mathcal{S}}(s') \times \mathcal{X}^{\mathcal{M}}(q') \mid \exists z' \in \mathcal{X}^{\mathcal{T}}(t_1)\colon f_t(t_1, z') = y' \wedge g_t(z') = y\}.
\end{aligned}$$

Since $f_s(t_1) = s'$ and $g_s(t_1) = q'$, we know that $(s', q', \mu') \in R$. This means that $s'\ R^{\mu'}\ q'$, as required for (B4d).

We have thus proven that $\mathcal{S} \simeq \mathcal{M}$. The theorem now follows by application of Lemma C.5.2. □

C.6.2 Proof of Lemma 5.8.17

Proof. We prove both directions of the bi-implication:

- Assume that $\mathcal{C}(q) = \mathcal{C}(q') \vee q \# q'$. Suppose that $r \in B$ with $r \# q$. We need to show that $r \# q' \vee q \# q'$. If the $q \# q'$ from the assumption holds, then we are already done. So suppose that $\neg(q \# q')$ and $\mathcal{C}(q) = \mathcal{C}(q')$. Then, since $r \# q$, $r \notin \mathcal{C}(q)$. Therefore, $r \notin \mathcal{C}(q')$. This implies that $r \# q'$, as required.

- Assume that $(\forall r \in B\colon r \# q \implies r \# q' \vee q \# q')$. Suppose that $\neg(q \# q')$. We need to show that $\mathcal{C}(q) = \mathcal{C}(q')$. Since q is identified, all basis states except one are apart from q. Let r be the unique basis state that is not apart from q. By our assumption, q' is apart from all states in $B \setminus \{r\}$. Thus $\mathcal{C}(q') \subseteq \{r\}$. By Lemma C.6.9, $\mathcal{C}(q')$ contains at least one state. Therefore, we conclude that $\mathcal{C}(q') = \{r\}$. This implies that $\mathcal{C}(q) = \mathcal{C}(q')$, as required.

□

C.7 Auxiliary Properties Concerning Observation Tree MMTs and Functional (g)MMT Simulations

The fact that observation tree MMTs are trees implies that:

Proposition C.7.1. Let $\mathcal{T}$ be an observation tree MMT with a run $\pi \in \mathit{runs}(\mathcal{T})$. Then π can traverse any state of Q at most once.

Proposition C.7.2. Let $\mathcal{M}$ be a (partial) MMT with a run $\pi = q_0 \xrightarrow{i_1 \dots i_n} q_n \in \mathit{runs}(\mathcal{M})$. Let $x \in \mathcal{X}(q_n)$ for some $x \in X$, and let $l = \mathit{firstStartedAt}_\pi(x)$. Then:

$$\forall j \in \{1, \dots, l-1\} \colon x \notin \mathcal{X}(q_j).$$

Lemma C.7.1. Let $\mathcal{M}$ be a (partial) MMT with a run $\pi = q_0 \xrightarrow{i_1 \dots i_n} q_n \in \mathit{runs}(\mathcal{M})$. Let $x \in \mathcal{X}(q_n)$ for some $x \in X$, and let $l = \mathit{firstStartedAt}_\pi(x)$. Then:

$$l > 0 \implies i_l \in I$$

Proof. Proof by contradiction: If $i_l = \mathsf{to}[y]$ for some timer $y \in X$, then Rule 4.6 tells us that $y = x$. Rule 4.5 and Rule 4.6 then tell us that $x \in \mathcal{X}(q_{l-1})$. Proposition C.7.2 tells us that therefore, $\mathit{firstStartedAt}_\pi(x) \neq l$, which contradicts $\mathit{firstStartedAt}_\pi(x) = l$. Therefore, $i_l \in I$. □

Lemma C.7.2. Let $\mathcal{T}$ be an observation tree MMT with a state $q_n \in Q$. Let $\pi = q_\mathcal{I}^\mathcal{T} \xrightarrow[u_1]{i_1} q_1 \dots \xrightarrow[u_n]{i_n} q_n \in \mathit{runs}(\mathcal{T})$. Then:

$$\forall j \in \{1, \dots, n\} \colon i_j \in I \wedge u_j = (x, c) \implies (\forall l \in \{1, \dots, n\} \colon l \neq j \implies \neg(i_l \in I \wedge u_l = (x', c') \wedge x' = x)).$$

Proof. Let $i_j \in I$ such that $u_j = (x, c)$. Then the fact that $\mathcal{T}$ is an observation tree MMT implies that $x = x_{q_j}$. For all $l \in \{1, \dots, n\} \colon l \neq j$, we know from the fact that $\mathcal{T}$ is an observation tree MMT that if $i_l \in I$ and $u_l = (x', c')$, then $x' = x_{q_l}$. Since $l \neq j$, Proposition C.7.1 tells us that $q_l \neq q_j$. Therefore, $x_{q_l} \neq x_{q_j}$. This means that $x' \neq x$, as required. □

Lemma C.7.3. Let $\mathcal{T}$ be an observation tree MMT with a state $q_n \in Q$. Let $\pi = q_0 \xrightarrow{i_1 \dots i_j} q_j \in \mathit{runs}(\mathcal{T})$. If timer $x \in \mathcal{X}(q_j)$, then:

$$\mathit{firstStartedAt}_\pi(x) = l \wedge l > 0 \implies x = x_{q_l}.$$

Proof. If $\mathit{firstStartedAt}_\pi(x) = l$ and $l > 0$, then Lemma C.7.1 tells us that $i_l \in I$. The fact that $\mathcal{T}$ is an observation tree then implies that $q_{l-1} \xrightarrow[(x,c)]{i_l} q_l \in \mathit{runs}(\mathcal{T})$ with $x = x_{q_l}$, as required. □

Lemma C.7.4. Let $\mathcal{T}$ be an observation tree MMT. Let $\pi = q_0 \xrightarrow{i_1 \dots i_j} q_j \in \mathit{runs}(\mathcal{T})$. Then:

$$\forall x \in \mathcal{X}(q_j), l \in \{1, \dots, j\} \colon x \notin \mathcal{X}(q_0) \wedge x = x_{q_l} \implies \mathit{firstStartedAt}_\pi(x) = l.$$

Proof. Let $x \in \mathcal{X}(q_j)$ and $l \in \{1, \dots, j\}$ such that $x \notin \mathcal{X}(q_0)$ and $x = x_{q_l}$. Let $k = \mathit{firstStartedAt}_\pi(x)$. Then $x \notin \mathcal{X}(q_0)$ implies that $0 < k \leq n$. Lemma C.7.1 tells us that $i_k \in I$. We therefore know that $q_{k-1} \xrightarrow[(x_{q_l}, c)]{i_k} q_k \in \mathit{runs}(\mathcal{T})$. The fact that $\mathcal{T}$ is an observation tree implies that $x_{q_l} = x_{q_k}$. Every distinct state of an observation tree MMT is associated with a distinct timer. This fact and Proposition C.7.1 together tells us that since $x_{q_l} = x_{q_k}$, $q_k = q_l$ and $k = l$. This implies that $\mathit{firstStartedAt}_\pi(x) = l$, as required. □

Definition C.7.1. Let $\mathcal{T}$ be an observation tree MMT with a state $q \in Q^\mathcal{M}$. The **descendants** of q are the states:

$$D^\mathcal{T}(q) \coloneqq \{q' \in Q^\mathcal{T} \mid \exists \sigma \in (A^\mathcal{T})^{>1} \colon q \xrightarrow{\sigma} q' \in \mathit{runs}(\mathcal{T})\}.$$

Lemma C.7.5. Let $\mathcal{T}$ be an observation tree MMT with states $q, q_a \in Q$. Then:

$$\forall x \in \mathcal{X}(q) \colon x = x_{q_a} \implies q_a \notin D^\mathcal{T}(q).$$

Proof. Let $q, q_a \in Q$, let $x = x_{q_a} \in \mathcal{X}(q)$, and let $n = |\mathsf{access}(q)|$. Let $\sigma = \mathsf{access}(q) = i_1 \dots i_n$. Then there are states $q_1 \dots q_n \in Q$ such that $\pi = q_\mathcal{I}^\mathcal{T} \xrightarrow{i_1} q_1 \xrightarrow{i_2 \dots i_n} q_n \in \mathit{runs}(\mathcal{T})$. This implies that $q_n = q$, and that $q_a = q_l$ for some $l \in \{1, \dots, n\}$. Since $q_a = q_l$, $x_{q_a} = x_{q_l}$. Lemma C.7.4 and Rule 4.1 therefore tell us that $\mathit{firstStartedAt}_\pi(x_{q_a}) = l$. We thus know that $q_a = q_l$ is traversed by π. Proposition C.7.1 now tells us that $q_a \notin D^\mathcal{T}(q)$, as required. □

Lemma C.7.6. Let $\mathcal{T}$ be an observation tree MMT with a run $q_0 \xrightarrow{i_1 \dots i_n} q_n \in runs(\mathcal{T})$. Then:

$$\forall x \in X\colon \\ x \in \mathcal{X}(q_0) \ \wedge \ x \in \mathcal{X}(q_n) \quad \Longrightarrow \quad (\forall j \in \{1, \dots, n\}\colon x \in \mathcal{X}(q_j) \wedge (\tau(q_{j-1}, i_j) = (x, c) \Longrightarrow i_j = \mathsf{to}[x])).$$

Proof. Let $x \in X$ such that $x \in \mathcal{X}(q_0)$ and $x \in \mathcal{X}(q_n)$. For all $j \in \{1, \dots, n\}$:

1. We prove that $x \in \mathcal{X}(q_j)$ by a contradiction. Suppose that $x \notin \mathcal{X}(q_j)$. Then $x \in \mathcal{X}(q_n)$ implies that x must be activated somewhere along $\rho = q_j \xrightarrow{i_{j+1} \dots i_n} q_n$. Let $l = \mathit{firstStartedAt}_\rho(x)$. Then Lemma C.7.1 tells us that since $l > 0$, $i_l \in I$. We thus know that $q_{l-1} \xrightarrow[(x,c)]{i_l} q_l \in runs(\mathcal{T})$. The fact that $\mathcal{T}$ is an observation tree then tells us that $x = x_{q_l}$.
   Lemma C.7.5 tells us that since $x = x_{q_l}$ and $x \in \mathcal{X}^{\mathcal{T}}(q_0)$, $q_l \notin D^{\mathcal{T}}(q_0)$. This contradicts the fact that q_l is encountered in π after q_0. Therefore, $x \in \mathcal{X}(q_j)$.

2. If $\tau(q_{j-1}, i_j) = (x, c)$, then we prove by contradiction that $i_j = \mathsf{to}[x]$. Suppose that $i_j \neq \mathsf{to}[x]$. Then there are two possibilities:

   - If $i_j \in I$, then we know from the fact that $\mathcal{T}$ is an observation tree that $q_{j-1} \xrightarrow[(x,c)]{i_j} q_j \in runs(\mathcal{T})$ with $x = x_{q_j}$. Item 1 and $x \in \mathcal{X}(q_0)$ thus tell us that for all $h \in \{0, \dots, n\}\colon x_{q_j} \in \mathcal{X}(q_h)$. Therefore, $x_{q_j} \in \mathcal{X}(q_{j-1})$. Lemma C.7.5 tells us that $q_j \notin D^{\mathcal{T}}(q_{j-1})$. This contradicts the fact that q_{j-1} has a state transition to q_j.
   - If $i_j = \mathsf{to}[x']$ with $x' \neq x$, then Rule 4.6 states that since $\tau(q_{j-1}, i_j) \neq \bot$, $\tau(q_{j-1}, i_j) = (x', c')$. This contradicts with $\tau(q_{j-1}, i_j) = (x, c)$, since it would require that $x' = x$.

   We see that all cases in which $i_j \neq \mathsf{to}[x]$ lead to a contradiction. Therefore, $i_j = \mathsf{to}[x]$, as required.

□

Lemma C.7.7. Let $\mathcal{T}$ be an observation tree MMT, and let $\mathcal{M}$ be a gMMT. Let $\langle f_s, f_t, f_u \rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Let $m'\colon f_s(q_0) \leftrightarrow f_s(q'_0)$. Let:

- $\pi = q_0 \xrightarrow{i_1 \dots i_n} q_n \in runs(\mathcal{T})$ and $\pi' = q'_0 \xrightarrow{i'_1 \dots i'_n} q'_n \in runs(\mathcal{T})$.
- $\rho = \langle f_s, f_t, f_u \rangle(\pi) \in runs(\mathcal{M})$ and $\rho' = \langle f_s, f_t, f_u \rangle(\pi') \in runs(\mathcal{M})$, such that $m'\colon \rho \leftrightarrow \rho'$.

Then:

$$\forall j \in \{1, \dots, n\}\colon \qquad i_j \in I^{\mathcal{T}} \quad \Longleftrightarrow \quad i'_j \in I^{\mathcal{T}}.$$

Proof. Let $i_j \in A^{\mathcal{T}}$. If $i_j \in I^{\mathcal{T}}$, then $f_t(q_{j-1}, q_j) \in I^{\mathcal{M}}$. Now, since $m'\colon \rho \leftrightarrow \rho'$, $f_t(q'_{j-1}, i'_j) \in I^{\mathcal{M}}$. Therefore, $i'_j \in I^{\mathcal{T}}$. A similar argument would reveal that $i'_j \in I^{\mathcal{T}} \Longrightarrow i_j \in I^{\mathcal{T}}$. Therefore, $i_j \in I^{\mathcal{T}} \Longleftrightarrow i'_j \in I^{\mathcal{T}}$, as required. □

Lemma C.7.8. Let $\mathcal{T}$ be an observation tree MMT, and let $\mathcal{M}$ be a gMMT. Let $\langle f_s, f_t, f_u \rangle\colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Let $m'\colon f_s(q_0) \leftrightarrow f_s(q'_0)$. Let:

- $\pi = q_0 \xrightarrow{i_1 \dots i_n} q_n \in runs(\mathcal{T})$ and $\pi' = q'_0 \xrightarrow{i'_1 \dots i'_n} q'_n \in runs(\mathcal{T})$.
- $\rho = \langle f_s, f_t, f_u \rangle(\pi) \in runs(\mathcal{M})$ and $\rho' = \langle f_s, f_t, f_u \rangle(\pi') \in runs(\mathcal{M})$, such that $m'\colon \rho \leftrightarrow \rho'$.

Then:

$$\forall k \in \{1, \dots, n-1\}, j \in \{2, n\}\colon \qquad q_{k-1} \xrightarrow{i_k \dots i_j} q_j \text{ is spanning} \quad \Longrightarrow \quad q'_{k-1} \xrightarrow{i'_k \dots i'_j} q'_j \text{ is spanning.}$$

Proof. Let $q_{k-1} \xrightarrow{i_k \dots i_j} q_j$ be a spanning sub-run of π. Then Lemma 5.3.2 implies that ρ's sub-run:

$$f_s(q_{k-1}) \xrightarrow{f_t(q_{k-1},i_k)\dots f_t(q_{j-1},i_j)} f_s(q_j)$$

is spanning as well. The fact that $m' \colon \rho \leftrightarrow \rho'$ now implies that since $k > 0$:

$$f_s(q'_{k-1}) \xrightarrow{f_t(q'_{k-1},i'_k)\dots f_t(q'_{j-1},i'_j)} f_s(q'_j)$$

is also spanning. Condition (FGS5) now implies that $q'_{k-1} \xrightarrow{i'_k \dots i'_j} q'_j$ is a spanning sub-run of π', as required. □

Lemma C.7.9. Let $\mathcal{T}$ be an observation tree MMT, and let $\mathcal{M}$ be a gMMT. Let $\langle f_s, f_t, f_u \rangle \colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Let $q_0, q'_0 \in Q^{\mathcal{T}}$. Let $m \colon q_0 \leftrightarrow q'_0$ and $m' \colon f_s(q_0) \leftrightarrow f_s(q'_0)$. Let:

- $\pi = q_0 \xrightarrow{i_1 \dots i_n} q_n \in runs(\mathcal{T})$ and $\pi' = q'_0 \xrightarrow{i'_1 \dots i'_n} q'_n \in runs(\mathcal{T})$.
- $\rho = \langle f_s, f_t, f_u \rangle(\pi) \in runs(\mathcal{M})$ and $\rho' = \langle f_s, f_t, f_u \rangle(\pi') \in runs(\mathcal{M})$, such that $m' \colon \rho \leftrightarrow \rho'$.

Then:

$$\begin{aligned} &\forall x, y, y' \in X^{\mathcal{T}}, k \in \{1, \dots, n\}, j \in \{k, \dots, n\} \colon \\ &\quad i_k = \mathsf{to}[x] \wedge i_j = \mathsf{to}[x] \wedge i'_j = \mathsf{to}[y] \quad \Longrightarrow \quad i'_k = \mathsf{to}[y]. \end{aligned}$$

Proof. The fact that $i_k \in TO(X^{\mathcal{T}})$ implies per Lemma C.7.7 that $\exists y' \in X^{\mathcal{T}} \colon i'_k = \mathsf{to}[y']$. The remaining question is whether $y' = y$. Rule 4.6 implies that since $i_k = \mathsf{to}[x]$, $x = \mathcal{X}(q_{k-1})$ and since $i_j = \mathsf{to}[x]$, $x \in \mathcal{X}(q_{j-1})$. Lemma C.7.6 therefore tells us that:

1. $\forall l \in \{k, \dots, j-1\} \colon x \in \mathcal{X}(q_l)$, and
2. $\forall l \in \{k, \dots, j-1\} \colon \tau^{\mathcal{T}}(q_{l-1}, i_l) = (x, c) \Longrightarrow i_l = \mathsf{to}[x]$.

We perform a case distinction on k:

- If $k = j$, then $i'_k = i'_j = \mathsf{to}[y]$, as required.
- Otherwise, we know that $k < j$. Item 2 tells us that between i_k and i_j, x can only be reset by timeout actions for x, and not by input actions or by timeouts for other timers. Let $i_h, i_l = \mathsf{to}[x]$ be two consecutive timeouts for x in π, where $k \leq h < l \leq j$. Lemma C.7.7 tells us that since $i_h \in TO(X^{\mathcal{T}})$, $i'_h \in TO(X^{\mathcal{T}})$.
  The fact that i_h and i_l are consecutive timeouts for x in π implies that, since x is active in all intermediate states per Item 1, $q_{h-1} \xrightarrow{i_h \dots i_l} q_l$ is an x-spanning sub-run of π. Lemma C.7.8 tells us that therefore, $q'_{h-1} \xrightarrow{i'_h \dots i'_l} q'_l$ is a spanning sub-run of π'.
  We prove the property by an induction on l:
  - Base case: If $l = j$, then we know that $q'_{h-1} \xrightarrow{i'_h \dots i'_j} q'_j$ is a spanning sub-run of π'. Rule 4.6 thus implies that, since $i'_j = \mathsf{to}[y]$ and $i'_h \in TO(X^{\mathcal{T}})$, $i'_h = \mathsf{to}[y]$.
  - Inductive step case: If $l < j$, then we use the following induction hypothesis:
    IH: $i'_l = \mathsf{to}[y]$.

    The induction hypothesis tells us that the spanning sub-run $q'_{h-1} \xrightarrow{i'_h \dots i'_l} q'_l$ is y-spanning. Rule 4.6 thus implies that since $i'_h \in TO(X^{\mathcal{T}})$, $i'_h = \mathsf{to}[y]$.

  We thus know that, for all $l \leq j$ with $k \leq h < l \leq j$ and with i_h and i_l being consecutive timeouts for x, $i'_h = \mathsf{to}[y]$. Since this is the case for all $h \geq k$, $i'_k = \mathsf{to}[y]$, as required.

We thus know that in all cases, $i'_k = \mathsf{to}[y]$, as required. □

Lemma C.7.10. Let $\mathcal{T}$ be an observation tree MMT, and let $\mathcal{M}$ be a (g)MMT. Let $\langle f_s, f_t, f_u\rangle \colon \mathcal{T} \to \mathcal{M}$ be a functional simulation. Let $q_0, q_0' \in Q^{\mathcal{T}}$. Let $m \colon q_0 \leftrightarrow q_0'$ and $m' \colon f_s(q_0) \leftrightarrow f_s(q_0')$. Let:

- $\pi = q_0 \xrightarrow{i_1 \dots i_n} q_n \in runs(\mathcal{T})$ and $\pi' = q_0' \xrightarrow{i_1' \dots i_n'} q_n' \in runs(\mathcal{T})$.
- $\rho = \langle f_s, f_t, f_u\rangle(\pi) \in runs(\mathcal{M})$ and $\rho' = \langle f_s, f_t, f_u\rangle(\pi') \in runs(\mathcal{M})$, such that $m' \colon \rho \leftrightarrow \rho'$.

Then:

$$\begin{aligned}&\forall k, k' \in \{1, n-1\}, j \in \{2, n\} \colon \\ &\quad (q_{k-1} \xrightarrow{i_k \dots i_j} q_j \text{ is } x\text{-spanning} \ \wedge \ q'_{k'-1} \xrightarrow{i'_{k'} \dots i'_j} q'_j \text{ is } x'\text{-spanning} \wedge x \notin \mathsf{dom}(m)) \implies \\ &\quad (\exists h \in \{1, \dots, k\} \colon \quad x = x_{q_h} \wedge x' = x_{q'_h}).\end{aligned}$$

Proof. Lemma C.7.8 tells us that since $q_{k-1} \xrightarrow{i_k \dots i_j} q_j$ is a spanning sub-run of π, $q'_{k-1} \xrightarrow{i'_k \dots i'_j} q'_j$ is a spanning sub-run of π'. This implies that $k' = k$. We perform a case distinction on i_k:

- If $i_k \in I^{\mathcal{T}}$, then Lemma C.7.7 tells us that $i'_k \in I^{\mathcal{T}}$. The fact that $\mathcal{T}$ is an observation tree MMT then implies that $q_{k-1} \xrightarrow[(x,c)]{i_k} q_k, q'_{k-1} \xrightarrow[(x',c')]{i'_k} q'_k \in runs(\mathcal{T})$, and that $x = x_{q_k}$ and $x' = x_{q'_k}$, as required.
- If $\exists y \in X^{\mathcal{T}} \colon i_k = \mathsf{to}[y]$, then the fact that the transition starts an x-spanning run and Rule 4.6 together imply that $i_k = \mathsf{to}[x]$. Since $i_k \in TO(X^{\mathcal{T}})$, Lemma C.7.7 tells us that $i'_k \in TO(X^{\mathcal{T}})$. The fact that the transition starts an x'-spanning run and Rule 4.6 now imply that $i'_k = \mathsf{to}[x']$.
  Let $l = \mathit{firstStartedAt}^{\mathcal{T}}_{q_0 \xrightarrow{i_1 \dots i_k} q_k}(x)$. Since $x \notin \mathsf{dom}(m)$, we know that $0 < l < k$. Lemma C.7.3 now tells us that $x = x_{q_l}$, and Lemma C.7.1 tells us that $i_l \in I^{\mathcal{T}}$.
  Since $x \in \mathcal{X}^{\mathcal{T}}(q_l)$ and $x \in \mathcal{X}^{\mathcal{T}}(q_{k-1})$, Lemma C.7.6 us that $\forall h \in \{l+1, \dots, k-1\} \colon x \in \mathcal{X}^{\mathcal{T}}(q_h) \wedge (\tau^{\mathcal{T}}(q_{h-1}, i_h) = (x, c) \implies i_h = \mathsf{to}[x])$. Therefore, since i_l starts x and $i_k = \mathsf{to}[x]$, there is a sequence of x-spanning runs in π, the first of which is started by i_l and the last of which is ended by i_k. Lemma C.7.8 therefore tells us that within π' there is a sequence of spannings runs, the first of which is started by i'_l and the last of which is ended by i'_k. Therefore, there exists a timer $y \in X^{\mathcal{T}}$ such that i'_l starts y and $i'_k = \mathsf{to}[y]$. Since i'_k starts an x'-spanning run, $y = x'$. Lemma C.7.7 tells us that since $i_l \in I^{\mathcal{T}}$, $i'_l \in I^{\mathcal{T}}$. Therefore, since $\mathcal{T}$ is an observation tree MMT, $q'_{l-1} \xrightarrow[(x',c')]{i'_l} q'_l \in runs(\mathcal{T})$ with $x' = x_{q'_l}$. We thus know that $\exists h \in \{1, \dots, k\} \colon x = x_{q_h}$ and $x' = x_{q'_h}$, as required.

□

C.8 Proof of Lemma 5.8.15

The proof of Lemma 5.8.15 relies on the following auxiliary lemma:

Lemma C.8.1. Let $\mathcal{T}$ be an observation tree MMT with a state $q \in Q^{\mathcal{T}}$, let $\mathcal{M}$ be an s-learnable gMMT, and let $\langle f_s, f_t, f_u\rangle \colon \mathcal{T} \to \mathcal{M}$ be a functional gMMT simulation. Then:

$$\begin{aligned}&q \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}} \wedge q \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}} \implies \\ &\quad (\forall x \in \mathcal{X}^{\mathcal{M}}(f_s(q)) \colon \exists (y \in \mathcal{X}^{\mathcal{T}}(q) \colon f_t(q, y) = x \wedge (f_t(q, y) \in \mathcal{X}^{\mathcal{M}}_0(f_s(q)) \iff y \in \mathcal{X}^{\mathcal{T}}_0(q)))).\end{aligned}$$

Proof. We know from $q \in \mathcal{A}^{\mathcal{T}}_{\mathcal{M}}$ that $|\mathcal{X}^{\mathcal{T}}(q)| = |\mathcal{X}^{\mathcal{M}}(f_s(q))|$. Taken with (FGS1) and (FGS2), this tells us that $\forall x \in \mathcal{X}^{\mathcal{M}}(f_s(q)) \colon (\exists y \in \mathcal{X}^{\mathcal{T}}(q) \colon f_t(q, y) = x)$. Let $y \in \mathcal{X}^{\mathcal{T}}(q)$, and let $x = f_t(q, y) \in \mathcal{X}^{\mathcal{M}}(f_s(q))$. The fact that $\mathcal{M}$ is s-learnable implies that $\mathcal{M}$ is complete, which implies that $f_s(q) \xrightarrow{\mathsf{to}[f_t(q,y)]} \in runs(\mathcal{M}) \Leftrightarrow f_t(q, y) \in \mathcal{X}^{\mathcal{M}}_0(f_s(q))$. We know from (FGS3) and (FGS4) that $q \xrightarrow{\mathsf{to}[y]} \in runs(\mathcal{T}) \Rightarrow f_s(q) \xrightarrow{\mathsf{to}[f_t(q,y)]} \in runs(\mathcal{M})$. Therefore, since $\mathcal{M}$ is s-learnable and thus complete, $q \xrightarrow{\mathsf{to}[y]} \in runs(\mathcal{T}) \Rightarrow f_s(q) \xrightarrow{\mathsf{to}[q, f_t(y)]} \in$

$runs(\mathcal{M}) \Rightarrow f_t(q,y) \in \mathcal{X}_0^{\mathcal{M}}(f_s(q))$. Since $\mathcal{T}$ is an observation tree, $q \xrightarrow{\mathsf{to}[y]} \in runs(\mathcal{T}) \Leftrightarrow y \in \mathcal{X}_0^{\mathcal{T}}(q)$. We thus know that $y \in \mathcal{X}_0^{\mathcal{T}}(q) \Rightarrow f_t(q,y) \in \mathcal{X}_0^{\mathcal{M}}(f_s(q))$.

The fact that $q \in \mathcal{E}_{\mathcal{M}}^{\mathcal{T}}$ implies that $|\mathcal{X}_0^{\mathcal{T}}(q)| = |\mathcal{X}_0^{\mathcal{M}}(f_s(q))|$. Taken with $y \in \mathcal{X}_0^{\mathcal{T}}(q) \Rightarrow f_t(q,y) \in \mathcal{X}_0^{\mathcal{M}}(f_s(q))$ and (FGS2), this implies that $y \in \mathcal{X}_0^{\mathcal{T}}(q) \Leftrightarrow f_t(q,y) \in \mathcal{X}_0^{\mathcal{M}}(f_s(q))$, as required. □

C.8.1 The proof of Lemma 5.8.15

Proof. For any maximal matching $m\colon s \leftrightarrow s'$, we know from $s \# s'$ that there exists an action sequence $\sigma_w \in (A^{\mathcal{S}})^*$ such that $\sigma_w \vdash s \#^m s'$. Let σ_w be a minimum-length witness of $s \#^m s'$. Let $\rho = s \xrightarrow{\sigma_w} \in runs(\mathcal{S})$ and $\rho' = read^m_{\sigma_w}(s') = s' \xrightarrow{\sigma'_w} \in runs(\mathcal{S})$, such that $\sigma'_w \in (A^{\mathcal{S}})^*$ and $\sigma_w \in (A^{\mathcal{S}})^*$ is a minimum-length witness of $s \overset{s}{\#}^m s'$. The fact that $\sigma_w \vdash s \#^m s'$ also implies that $m\colon \rho \leftrightarrow \rho'$.

Let:

$$\pi = q_0 \xrightarrow[u_1]{i_1/o_1} \dots \xrightarrow[u_n]{i_n/o_n} q_n,$$

be the run that lets us define ρ as:

$$\rho = \langle f_s, f_t, f_u \rangle(\pi) \in runs(\mathcal{S}).$$

Let:

$$\pi' = q'_0 \xrightarrow[u'_1]{i'_1/o'_1} \dots \xrightarrow[u'_n]{i'_n/o'_n} q'_n,$$

be the run that lets us define ρ' as:

$$\rho' = \langle f_s, f_t, f_u \rangle(\pi') \in runs(\mathcal{S}).$$

Lemma 5.8.13 tells us that calling:

- $addTransitionsFromSpecSeqAndMakeActiveExplored^{\mathcal{S}}_{\mathcal{M}}(\sigma_a \cdot \sigma_w)$ and
- $addTransitionsFromSpecSeqAndMakeActiveExplored^{\mathcal{S}}_{\mathcal{M}}(\sigma'_a \cdot \sigma'_w)$

would make it so that $\pi \in runs(\mathcal{T})$ and $\pi' \in runs(\mathcal{T})$. Lemma 5.8.14 tells us that these procedure calls will ensure that eventually:

$$\forall i \in \{0,\dots,n\}\colon \{q_i, q'_i\} \subseteq \mathcal{A} \wedge \{q_i, q'_i\} \subseteq \mathcal{E}. \tag{C.9}$$

Proposition 5.4.1 tells us that once that is the case:

$$\rho = f_s(q_0) \xrightarrow[f_u(q_0,q_1,u_1)]{f_t(q_0,i_1)/o_1} \dots \xrightarrow[f_u(q_{n-1},q_n,u_n)]{f_t(q_{n-1},i_n)/o_n} f_s(q_n) = \langle f_s, f_t, f_u \rangle(\pi) \in runs(\mathcal{S})$$

and:

$$\rho' = f_s(q'_0) \xrightarrow[f_u(q'_0,q'_1,u'_1)]{f_t(q'_0,i'_1)/o'_1} \dots \xrightarrow[f_u(q'_{n-1},q'_n,u'_n)]{f_t(q'_{n-1},i'_n)/o'_n} f_s(q'_n) = \langle f_s, f_t, f_u \rangle(\pi') \in runs(\mathcal{S}),$$

which means that π and π' will contain all timer updates present in both the specification, and the SUT. We get from Equation (C.9) that $q_0, q'_0 \in \mathcal{A}_{\mathcal{S}}^{\mathcal{T}}$ and $q_0, q'_0 \in \mathcal{E}_{\mathcal{S}}^{\mathcal{T}}$. The apartness $q_0 \# q'_0$ might therefore hold as a consequence of either of the following two cases:

- since $q_0 \in \mathcal{A}_{\mathcal{S}}^{\mathcal{T}}$ implies that $|\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{S}}(f_s(q_0))|$, and $q'_0 \in \mathcal{A}_{\mathcal{S}}^{\mathcal{T}}$ implies that $|\mathcal{X}^{\mathcal{T}}(q'_0)| = |\mathcal{X}^{\mathcal{S}}(f_s(q'_0))|$; we know that if $|\mathcal{X}^{\mathcal{S}}(f_s(q_0))| \neq |\mathcal{X}^{\mathcal{S}}(f_s(q'_0))|$, then:

  $$|\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{S}}(f_s(q_0))| \neq |\mathcal{X}^{\mathcal{S}}(f_s(q'_0))| = |\mathcal{X}^{\mathcal{T}}(q'_0)|.$$

  This would then mean that $q_0 \# q'_0$ per (active sizes).

- since $q_0 \in \mathcal{E}_{\mathcal{S}}^{\mathcal{T}}$ implies that $|\mathcal{X}_0^{\mathcal{T}}(q_0)| = |\mathcal{X}_0^{\mathcal{S}}(f_s(q_0))|$, and $q_0' \in \mathcal{E}_{\mathcal{S}}^{\mathcal{T}}$ implies that $|\mathcal{X}_0^{\mathcal{T}}(q_0')| = |\mathcal{X}_0^{\mathcal{S}}(f_s(q_0'))|$; we know that if $|\mathcal{X}_0^{\mathcal{S}}(f_s(q_0))| \neq |\mathcal{X}_0^{\mathcal{S}}(f_s(q_0'))|$, then:

  $$|\mathcal{X}_0^{\mathcal{T}}(q_0)| = |\mathcal{X}_0^{\mathcal{S}}(f_s(q_0))| \neq |\mathcal{X}_0^{\mathcal{S}}(f_s(q_0'))| = |\mathcal{X}_0^{\mathcal{T}}(q_0')|.$$

  This would then mean that $q_0 \# q_0'$ per (enabled sizes).

For the remainder of this proof, we will assume that neither of these conditions for apartness holds, since otherwise there would be nothing more for us to show. We therefore know from $q_0, q_0' \in \mathcal{A}_{\mathcal{S}}^{\mathcal{T}}$ and $q_0, q_0' \in \mathcal{E}_{\mathcal{S}}^{\mathcal{T}}$ that:

$$|\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{S}}(f_s(q_0))| = |\mathcal{X}^{\mathcal{S}}(f_s(q_0'))| = |\mathcal{X}^{\mathcal{T}}(q_0')|$$

and:

$$|\mathcal{X}_0^{\mathcal{T}}(q_0)| = |\mathcal{X}_0^{\mathcal{S}}(f_s(q_0))| = |\mathcal{X}_0^{\mathcal{S}}(f_s(q_0'))| = |\mathcal{X}_0^{\mathcal{T}}(q_0')|.$$

We define the mapping $m' \colon \mathcal{X}^{\mathcal{T}}(q_0) \to \mathcal{X}^{\mathcal{T}}(q_0')$ as:

$$m' = \{(x, y) \in \mathcal{X}^{\mathcal{T}}(q_0) \times \mathcal{X}^{\mathcal{T}}(q_0') \mid (f_t(q_0, x), f_t(q_0', y)) \in m\}.$$

The fact that the (active sizes) condition for apartness doesn't hold implies that $|\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{S}}(f_s(q_0))| = |\mathcal{X}^{\mathcal{S}}(f_s(q_0'))| = |\mathcal{X}^{\mathcal{T}}(q_0')|$. We know from (FGS2) that for a fixed state $q \in Q^{\mathcal{T}}$, f_t is injective with respect to the timers of $\mathcal{X}^{\mathcal{T}}(q)$. We know from (FGS1), (FGS2) and $|\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{S}}(f_s(q_0))| = |\mathcal{X}^{\mathcal{S}}(f_s(q_0'))| = |\mathcal{X}^{\mathcal{T}}(q_0')|$ that for $q \in \{q_0, q_0'\}$, f_t is surjective for the timers of $\mathcal{X}^{\mathcal{T}}(q)$. Timer map f_t is thus bijective with respect to the timers of $\mathcal{X}^{\mathcal{T}}(q)$ when $q \in \{q_0, q_0'\}$. This implies that mapping m' is different for every distinct maximal matching $m' \colon s \leftrightarrow s'$. The fact that $|\mathcal{X}^{\mathcal{T}}(q_0)| = |\mathcal{X}^{\mathcal{S}}(f_s(q_0))| = |\mathcal{X}^{\mathcal{S}}(f_s(q_0'))| = |\mathcal{X}^{\mathcal{T}}(q_0')|$ implies that there are exactly as many unique maximal matchings between q_0 and q_0' as there are unique maximal matchings $m \colon s \leftrightarrow s'$. Therefore, if we can show for any m that m' is a valid maximal matching with $q_0 \#^{m'} q_0'$, then q_0 and q_0' are apart for all maximal matchings that exist between q_0 and q_0'. This would mean that $q_0 \# q_0'$, as required.
For all $x \in \mathcal{X}^{\mathcal{T}}(q_0)$, condition (FGS1) implies that $f_t(q_0, x) \in \mathcal{X}^{\mathcal{S}}(f_s(q_0))$. Now, m being maximal implies that since $|\mathcal{X}^{\mathcal{S}}(f_s(q_0))| = |\mathcal{X}^{\mathcal{S}}(f_s(q_0'))|$, there is a timer $z \in \mathcal{X}^{\mathcal{S}}(f_s(q_0'))$: $m'(f_t(q_0, x)) = z$. Since $z \in \mathcal{X}^{\mathcal{S}}(f_s(q_0'))$, $q_0' \in \mathcal{A}_{\mathcal{S}}^{\mathcal{T}}$ and (FGS1), we know that there is a timer $y \in \mathcal{X}^{\mathcal{T}}(q_0')$ such that $f_t(q_0', y) = z = m'(f_t(q_0, x))$. So:

$$\forall x \in \mathcal{X}^{\mathcal{T}}(x_0') \colon m(f_t(q_0, x)) = f_t(q_0', y) = f_t(q_0', m'(x)). \tag{C.10}$$

Partial function m' needs to be injective to be a valid matching. We get from (FGS2) that if $x \neq x'$, then $f_t(q_0, x) \neq f_t(q_0, x')$. We know from the fact that m is a matching that m is injective, which tells us that $m(f_t(q_0, x)) \neq m(f_t(q_0, x'))$. This tells us that $f_t(q_0', m'(x)) \neq f_t(q_0', m'(x'))$, which tells us that $m'(x) \neq m'(x')$. Partial function m' is therefore injective, which makes it a valid matching.

We know from Lemma 5.5.2 that any witnesses of apartness between s and s' need to be at most $|Q^{\mathcal{S}}|$ elements long. Since we specify that σ_w is minimum-length, we know that it falls within that range, so $n = |\sigma_w| \leq |Q^{\mathcal{S}}|$.

The runs π and π' are matching under m', since for all $j \in \{1, \ldots, n\}$:

- If $i_j \in I$, then $f_t(q_{j-1}, i_j) = i_j \in I$. We then know from the definition of $m_{\rho'}^{\rho}$ that:

  $$f_t(q_{j-1}', i_j') = m_{\rho'}^{\rho}(f_t(q_{j-1}, i_j)) = m_{\rho'}^{\rho}(i_j) = i_j \in I.$$

  Therefore:

  $$i_j' = f_t(q_{j-1}', i_j') = i_j,$$

  as required.

- If $i_j = \mathsf{to}[x]$ for some $x \in X^{\mathcal{T}}$, then $f_t(q_{j-1}, i_j) = f_t(q_{j-1}, \mathsf{to}[x]) = \mathsf{to}[f_t(q_{j-1}, x)]$ for some $f_t(q_{j-1}, x) \in X^{\mathcal{S}}$. Let $k = \mathit{lastStartedAt}^{\mathcal{S}}_{f_s(q_0) \xrightarrow{f_t(q_0,i_1)\dots f_t(q_{j-2},i_{j-1})} f_s(q_{j-1})}(f_t(q_{j-1}, x))$. Then:

  $$f_s(q_{k-1}) \xrightarrow{f_t(q_{k-1},i_k)\dots f_t(q_{j-1},i_j)} f_s(q_j) \in \mathit{runs}(\mathcal{S})$$

  is a spanning sub-run of ρ. Condition (FGS5) therefore tells us that when $\pi \in \mathit{runs}(\mathcal{T})$, $q_{k-1} \xrightarrow{i_k \dots i_j} q_j$ is an x-spanning run. If:

  - $k = 0$, then $m \colon \rho \leftrightarrow \rho'$ implies that:

    $$f_t(q'_{j-1}, i'_j) = \mathsf{to}[\mathit{renameTo}^{\mathcal{S}}_{f_s(q'_0) \xrightarrow{f_t(q'_0,i'_1)\dots f_t(q'_{j-2},i'_{j-1})} f_s(q'_{j-1})}(\dots$$
    $$\dots\ m(\mathit{renamesTo}^{\mathcal{S}}_{f_s(q_0) \xrightarrow{f_t(q_0,i_1)\dots f_t(q_{j-2},i_{j-1})} f_s(q_{j-1})}(f_t(q_{j-1}, x))))].$$

    We perform a case distinction on x:

    * In one of two cases, $x \in \mathcal{X}^{\mathcal{T}}(q_0)$. Since $k = 0$, we know that:

      $$\exists x' \in \mathcal{X}^{\mathcal{S}}(f_s(q_0)) \colon \mathit{renameTo}^{\mathcal{S}}_{f_s(q_0) \xrightarrow{f_t(q_0,i_1)\dots f_t(q_{j-2},i_{j-1})} f_s(q_{j-1})}(x') = f_t(q_{j-1}, x).$$

      This implies that $f_t(q'_{j-1}, i'_j) = \mathsf{to}[\mathit{renameTo}^{\mathcal{S}}_{f_s(q'_0) \xrightarrow{f_t(q'_0,i'_1)\dots f_t(q'_{j-2},i'_{j-1})} f_s(q'_{j-1})}(m(x'))]$, and that $x' = f_t(q_0, x)$ per Lemma 5.3.3. We established with Equation (C.10) that $m(f_t(q_0, x)) = f_t(q'_0, m'(x))$. Since $x' = f_t(q_0, x)$, this implies that $m(x') = f_t(q'_0, m'(x))$. Therefore:

      $$\begin{aligned} f_t(q'_{j-1}, i'_j) &= \mathsf{to}[\mathit{renameTo}^{\mathcal{S}}_{f_s(q'_0) \xrightarrow{f_t(q'_0,i'_1)\dots f_t(q'_{j-2},i'_{j-1})} f_s(q'_{j-1})}(m(x'))] \\ &= \mathsf{to}[\mathit{renameTo}^{\mathcal{S}}_{f_s(q'_0) \xrightarrow{f_t(q'_0,i'_1)\dots f_t(q'_{j-2},i'_{j-1})} f_s(q'_{j-1})}(f_t(q'_0, m'(x)))] \\ &= \mathsf{to}[f_t(q'_{j-1}, m'(x))] \qquad \text{(Lemma 5.3.3)} \end{aligned}$$

      Condition (FGS2) now tells us that $i'_j = \mathsf{to}[m'(x)]$, as required.

    * Otherwise, $x \notin \mathcal{X}^{\mathcal{T}}(q_0)$. Then $x = x_{q_l}$ with $0 < l < j$. We know from $\mathit{firstStartedAt}^{\mathcal{T}}_{\pi}(x_{q_l}) = l$ that $k \geq l$, and thus that $k \geq l > 0$. This contradicts our knowledge that $k = 0$. Therefore, it cannot be the case that $x = x_{q_l}$ with $0 < l < j$, so this case is done.

  - $k > 0$, then $m \colon \rho \leftrightarrow \rho'$ implies that $f_s(q'_{k-1}) \xrightarrow{f_t(q'_{k-1},i'_k)\dots f_t(q'_{j-1},i'_j)} f_s(q'_j)$ is spanning. This fact and (FGS5) together tell us that when $\pi' \in \mathit{runs}(\mathcal{T})$, there is a timer $x' \in X^{\mathcal{T}}$ such that $q'_{k-1} \xrightarrow{i'_k \dots i'_j} q'_j$ is x'-spanning. We know that $f_s(q_{k-1}) \xrightarrow{f_t(q_{k-1},i_k)\dots f_t(q_{j-1},i_j)} f_s(q_j)$ is spanning. Together with (FGS5) and $i_j = \mathsf{to}[x]$, this implies that once $\pi \in \mathit{runs}(\mathcal{T})$, $q_{k-1} \xrightarrow{i_k \dots i_j} q_j$ is x-spanning.
    We perform a case distinction on whether $x \in \mathcal{X}^{\mathcal{T}}(q_0)$:

    * In the first case, $x \in \mathcal{X}^{\mathcal{T}}(q_0)$. Since $i_j = \mathsf{to}[x]$, $x \in \mathcal{X}^{\mathcal{T}}(q_{j-1})$ per Rule 4.5 and Rule 4.6. Therefore, since $x \in \mathcal{X}^{\mathcal{T}}(q_0)$, Lemma C.7.6 tells us that $\forall l \in \{1, \dots, j-1\} \colon x \in \mathcal{X}^{\mathcal{T}}(q_l)$ and $\tau^{\mathcal{T}}(q_{l-1}, i_l) = (x, c) \implies i_l = \mathsf{to}[x]$. This implies that since i_k starts an x-spanning run, $i_k = \mathsf{to}[x]$. Lemma C.7.7 tells us that since $i_k \in TO(X^{\mathcal{T}})$, $i'_k \in TO(X^{\mathcal{T}})$. Therefore, since i'_k starts an x'-spanning run, $i'_k = \mathsf{to}[x']$. There must be a first timeout $i_h = \mathsf{to}[x]$ in π, for some $h \in \{1, \dots, k\}$. Since $\mathit{lastStartedAt}^{\mathcal{T}}_{q_0 \xrightarrow{i_1 \dots i_{h-1}} q_{h-1}}(x) = 0$, we see that if:

      · $h = 1$, then:

      $$\begin{aligned} & m^{\rho}_{\rho'}(f_t(q_{h-1}, x), h-1) = m(f_t(q_{h-1}, x)) = m(f_t(q_0, x)) \\ &= f_t(q'_0, m'(x)) \qquad \text{(Equation (C.10))} \\ &= f_t(q'_{h-1}, m'(x)). \end{aligned}$$

· $h > 1$, then:

$$\begin{aligned}
&m^{\rho}_{\rho'}(f_t(q_{h-1}, x), h-1)\\
&= renameTo^{\mathcal{S}}_{f_s(q'_0)\xrightarrow{f_t(q'_0,i'_1)\dots f_t(q'_{h-1},i'_{h-1})}f_s(q'_{h-1})}(\dots\\
&\quad \dots\ m(renamesTo^{\mathcal{S}}_{f_s(q_0)\xrightarrow{f_t(q_0,i_1)\dots f_t(q_{h-1},i_{h-1})}f_s(q_{h-1})}(f_t(q_{h-1}, x)))).
\end{aligned}$$

Now, per Lemma 5.3.3:

$$renamesTo^{\mathcal{S}}_{f_s(q_0)\xrightarrow{f_t(q_0,i_1)\dots f_t(q_{h-1},i_{h-1})}f_s(q_{h-1})}(f_t(q_{h-1}, x)) = f_t(q_0, x).$$

Therefore:

$$\begin{aligned}
&m^{\rho}_{\rho'}(f_t(q_{h-1}, x), h-1)\\
&= renameTo^{\mathcal{S}}_{f_s(q'_0)\xrightarrow{f_t(q'_0,i'_1)\dots f_t(q'_{h-1},i'_{h-1})}f_s(q'_{h-1})}(\dots\\
&\quad \dots\ m(renamesTo^{\mathcal{S}}_{f_s(q_0)\xrightarrow{f_t(q_0,i_1)\dots f_t(q_{h-1},i_{h-1})}f_s(q_{h-1})}(f_t(q_{h-1}, x))))\\
&= renameTo^{\mathcal{S}}_{f_s(q'_0)\xrightarrow{f_t(q'_0,i'_1)\dots f_t(q'_{h-1},i'_{h-1})}f_s(q'_{h-1})}(m(f_t(q_0, x)))\\
&= renameTo^{\mathcal{S}}_{f_s(q'_0)\xrightarrow{f_t(q'_0,i'_1)\dots f_t(q'_{h-1},i'_{h-1})}f_s(q'_{h-1})}(f_t(q'_0, m'(x))) && \text{(Equation (C.10))}\\
&= f_t(q'_{h-1}, m'(x)). && \text{(Lemma 5.3.3)}
\end{aligned}$$

We thus see that in both cases, $f_t(q'_{h-1}, i'_h) = \mathsf{to}[m^{\rho}_{\rho'}(f_t(q_{h-1}, x), h-1)] = \mathsf{to}[f_t(q'_{h-1}, m'(x))]$. Condition (FGS2) tells us that therefore, $i'_h = \mathsf{to}[m'(x)]$. Lemma C.7.9 tells us that since $i_k = i_h = \mathsf{to}[x]$, $i'_h = \mathsf{to}[m'(x)]$ and $i'_k = \mathsf{to}[x']$, $x' = m'(x)$. Since i'_k starts a spanning run that ends with $i'_j = \mathsf{to}[x']$, $i'_j = \mathsf{to}[m'(x)]$, as required.

∗ Otherwise, $x = x_{q_l}$ for some $0 < l < j$. Since $x \notin \mathcal{X}^{\mathcal{T}}(q_0)$, $x \notin \mathsf{dom}(m')$. Lemma C.7.8 tells us that since $q_{k-1} \xrightarrow{i_k \dots i_j} q_j$ is spanning, $q'_{k-1} \xrightarrow{i'_k \dots i'_j} q'_j$ is spanning. Lemma C.7.10 therefore tells us that since $i_j = \mathsf{to}[x]$, $x = x_{q_l}$ and $x \notin \mathsf{dom}(m')$, $i'_j = \mathsf{to}[x_{q'_l}]$. Therefore, $i_j = \mathsf{to}[x_{q_l}]$ and $i'_j = \mathsf{to}[x_{q'_l}]$ for some $0 < l < j$, as required.

We now extend m' to π and π' in the usual way:

$$m'^{\pi}_{\pi'} \coloneqq m \cup \{(x_{q_k}, x_{q'_k}) \mid 0 < k \leq n\},$$

and $i'_j = m'^{\pi}_{\pi'}(i_j)$ for every j.

Let $\sigma = i_1 \dots i_n$, such that $f_t(q_0, i_1) \dots f_t(q_{n-1}, i_n) = \sigma_w$.

We now show that $\sigma \vdash q_0 \ \#^{m'}\ q'_0$. We do so by looking at each of the conditions for apartness for $\sigma_w \vdash f_s(q_0) \ \#^m\ f_s(q'_0)$ other than (active sizes) and (enabled sizes), because we already know that neither of those conditions hold:

- (outputs): then $o \neq o'$. Then (FGS3) and (FGS4) tell us that:

$$q_{n-1} \xrightarrow{i_n/o_n} \in runs(\mathcal{T}) \quad \wedge \quad q'_{n-1} \xrightarrow{i'_n/o'_n} \in runs(\mathcal{T}).$$

Since $o \neq o'$, we get that $\sigma \vdash q_0 \ \#^{m'}\ q'_0$.

- (constants): then:

$$\begin{aligned}
&(\exists x \in \mathcal{X}^{\mathcal{S}}(f_s(q_n)) \colon (f_u(q_{n-1}, q_n, u_n) = f_u(q_{n-1}, q_n))(x) = c \in \mathbb{N}^{>0}) \wedge\\
&(\exists x' \in \mathcal{X}^{\mathcal{S}}(f_s(q'_n)) \colon (f_u(q'_{n-1}, q'_n, u'_n) = f_u(q'_{n-1}, q'_n))(x') = c' \in \mathbb{N}^{>0}) \wedge\\
&c \neq c'
\end{aligned}$$

Then Equation (C.9) implies that $\exists y, y' \in X^{\mathcal{T}} \colon u_n = (y, c) \wedge u'_n = (y', c')$. Since $c \neq c'$, we have that $\sigma \vdash q_0 \ \#^{m'} \ q'_0$.

- (updating): then:

$$(\exists x \in \mathcal{X}^{\mathcal{S}}(f_s(q_n)) \colon (f_u(q_{n-1}, q_n, u_n) = f_u(q_{n-1}, q_n))(x) \in \mathbb{N}^{>0}) \ \Leftrightarrow$$
$$(\neg \exists x' \in \mathcal{X}^{\mathcal{S}}(f_s(q'_n)) \colon (f_u(q'_{n-1}, q'_n, u'_n) = f_u(q'_{n-1}, q'_n))(x') \in \mathbb{N}^{>0})$$

Then Equation (C.9) implies that $u_n \neq \bot \Leftrightarrow u'_n = \bot$, which tells us that $\sigma \vdash q_0 \ \#^{m'} \ q'_0$.

- (enabled): then:

$$\exists x \in X^{\mathcal{S}} \colon m^{\rho}_{\rho'}(x, n)\downarrow \wedge \ (x \in \mathcal{X}^{\mathcal{S}}_0(f_s(q_n)) \Leftrightarrow m^{\rho}_{\rho'}(x, n) \notin \mathcal{X}^{\mathcal{S}}_0(f_s(q'_n))). \tag{C.11}$$

We know from $m^{\rho}_{\rho'}(x, n)\downarrow$ that $x \in \mathcal{X}^{\mathcal{S}}(f_s(q_n))$ and $m^{\rho}_{\rho'}(x, n) \in \mathcal{X}^{\mathcal{S}}(f_s(q'_n))$. Equation (C.9) tells us that eventually, $q_n, q'_n \in \mathcal{A}^{\mathcal{T}}_{\mathcal{S}}$. Once this is the case, there must be timers $y \in \mathcal{X}^{\mathcal{T}}(q_n)$ and $y' \in \mathcal{X}^{\mathcal{T}}(q'_n)$ such that $f_s(q_n, y) = x$ and $f_t(q'_n, y') = m^{\rho}_{\rho'}(x, n) = m^{\rho}_{\rho'}(f_t(q_n, y), n)$, per (FGS1) and (FGS2). Since $m^{\rho}_{\rho'}(x, n)\downarrow$, $k = \mathit{lastStartedAt}^{\mathcal{S}}_{\rho}(x) \in \mathbb{N}$. Lemma C.3.3 therefore tells us that $m^{\rho}_{\rho'}(f_t(q_n, y), n) = f_t(q'_n, m'^{\pi}_{\pi'}(y))$. Since, $f_t(q'_n, y') = m^{\rho}_{\rho'}(f_t(q_n, y), n) = f_t(q'_n, m'^{\pi}_{\pi'}(y))$, (FGS2) tells us that $y' = m'^{\pi}_{\pi'}(y)$.
We know from Equation (C.11) that for a timer $x \in X^{\mathcal{S}}$ that satisfies Equation (C.11), either:

1. $x \in \mathcal{X}^{\mathcal{S}}_0(f_s(q_n)) \wedge m^{\rho}_{\rho'}(x, n) \notin \mathcal{X}^{\mathcal{S}}_0(f_s(q'_n))$, or
2. $m^{\rho}_{\rho'}(x, n) \in \mathcal{X}^{\mathcal{S}}_0(f_s(q'_n)) \wedge x \notin \mathcal{X}^{\mathcal{S}}_0(f_s(q_n))$.

We perform a case distinction:

1. In the first case, Lemma C.8.1 tells us that $y \in \mathcal{X}^{\mathcal{T}}_0(q_n)$ and $y' = m'^{\pi}_{\pi'}(y) \notin \mathcal{X}^{\mathcal{T}}_0(q'_n)$. Therefore, $\exists y \in \mathsf{dom}(m'^{\pi}_{\pi'}) \colon (y \in \mathcal{X}^{\mathcal{T}}_0(q_n) \wedge m'^{\pi}_{\pi'}(y) \notin \mathcal{X}^{\mathcal{T}}_0(q'_n))$.
2. In the second case, Lemma C.8.1 tells us that $y \notin \mathcal{X}^{\mathcal{T}}_0(q_n)$ and $y' = m'^{\pi}_{\pi'}(y) \in \mathcal{X}^{\mathcal{T}}_0(q'_n)$. Therefore, $\exists y \in \mathsf{dom}(m'^{\pi}_{\pi'}) \colon (y \notin \mathcal{X}^{\mathcal{T}}_0(q_n) \wedge m'^{\pi}_{\pi'}(y) \in \mathcal{X}^{\mathcal{T}}_0(q'_n))$.

Therefore, in all cases, $\exists y \in \mathsf{dom}(m'^{\pi}_{\pi'}(y)) \colon (y \notin \mathcal{X}^{\mathcal{T}}_0(q_n) \Leftrightarrow m'^{\pi}_{\pi'}(y) \in \mathcal{X}^{\mathcal{T}}_0(q'_n))$, which tells us that $\sigma \vdash q_0 \ \#^{m'} \ q'_0$.

Therefore, $q_0 \ \# \ q'_0$ will eventually hold. □

C.9 Proofs Related to the MMT Conformance Testing Procedure

Proofs for the MMT conformance testing procedure from Chapter 5.

C.9.1 Proof of Lemma 5.8.2

Proof. We perform a case distinction on the state q that is passed to $\mathit{makeEnabledExplored}^{\mathcal{S}}$:

- If $q = q^{\mathcal{T}}_I$. Then Algorithm 13 terminates immediately, since per Rule 4.1, initial states of observation tree MMTs never have active timers (and therefore never have enabled timers). Therefore, calling Algorithm 13 on q ensures that $q \in \mathcal{E}$, as required.
- If $q \neq q^{\mathcal{T}}_I$, then Algorithm 13 performs the waiting query $\mathbf{WQ}^{\mathcal{M}}(\overline{\mathsf{access}(q)})$. This yields a value $w \subseteq (\mathbb{N}^{>0} \times \mathbb{N}^{>0})$.

  Now, for all $(j, c) \in w$, Algorithm 13 finds the timer updates for $\mathcal{S}$ at index j of $\langle g_s, g_t, g_u \rangle(\mathsf{access}(q))$ and returns with a counterexample if $\mathcal{S}$ has no corresponding timer update at j with constant c. The algorithm identifies the observation tree timer x'_j for which the timeout action should be (or already is) there from q, and then adds the timeout action for x'_j from q if it isn't already there. It also records

the timer update that started x'_j (if needed), and it marks x'_j as spanning in the states between the x'_j timeout and the update that last started x'_j before the timeout, in order to finish the spanning for x'_j, the one timer that corresponds to $(j, c) \in w$. Algorithm 13 does this for all elements of w, which contains an element for *all* timers that can have a timeout in q. Therefore, calling makeEnabledExplored on q makes q enabled explored with respect to the SUT, i.e. $q \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$.
Algorithm 13 yields a counterexample if the total number of timeouts and the positions along the access sequence at which they were started for q is different from the specification counterparts of q and of q's access sequence. This ensures that if $|\mathcal{X}^{\mathcal{S}}_0(f_s(q))| \neq |\mathcal{X}^{\mathcal{M}}_0(g_s(q))|$ or the symbolic words that terminate with timeouts from $f_s(q)$ are different from those from $g_s(q)$, then q is *not* added to $\mathcal{E}$ and that Algorithm 13 returns a counterexample symbolic word that is feasible in either $\mathcal{S}$ or in $\mathcal{M}$, but not in both.
Otherwise, we know that $q \in \mathcal{E}^{\mathcal{T}}_{\mathcal{M}}$ and $q \in \mathcal{E}^{\mathcal{T}}_{\mathcal{S}}$, and that the spannings that terminate with timeouts from q (and therefore those for the SUT) are symbolically the same as those for the specification. This then implies that there are no conflicts between the specification and the SUT with regard to timeouts from q and its specification and SUT counterparts.
Finally, the sub-procedure adds q to $\mathcal{E}$.

We thus know by induction on the state $q \in Q^{\mathcal{T}}$ passed to Algorithm 13 that once the algorithm terminates, $q \in \mathcal{E}$, unless it found a conflict between the specification and the SUT. □

C.9.2 Proof of Lemma 5.8.9

Proof. Let $\tau^{\mathcal{T}}(q, i) = (x, c) \in \mathcal{X}^{\mathcal{T}}(q') \times \mathbb{N}^{>0}$, and let $\mathfrak{r} = \tau^{\mathcal{S}}(f_s(q), f_t(q, i))$. In general, (FGS1) tells us that:

$$|\mathcal{X}^{\mathcal{S}}(f_s(q'))| \geq |\mathcal{X}^{\mathcal{T}}(q')|,$$

and (FMS1) tells us that:

$$|\mathcal{X}^{\mathcal{M}}(g_s(q'))| \geq |\mathcal{X}^{\mathcal{T}}(q')|.$$

We could thus conclude that $q' \in \mathcal{A}$ if we can show that $|\mathcal{X}^{\mathcal{T}}(q')| \geq |\mathcal{X}^{\mathcal{S}}(f_s(q'))|$ and $|\mathcal{X}^{\mathcal{T}}(q')| \geq |\mathcal{X}^{\mathcal{M}}(g_s(q'))|$.
We label our assumption that all states that are active in q are also active in q':

$$\mathcal{X}^{\mathcal{T}}(q) \subseteq \mathcal{X}^{\mathcal{T}}(q'). \tag{C.12}$$

- For the specification: By (FGS3):

  $$\mathfrak{r}(f_t(q', x)) = c \qquad \wedge \qquad \forall y \in \mathcal{X}^{\mathcal{T}}(q') \setminus \{x\} \colon \mathfrak{r}(f_t(q', y)) = f_t(q, y).$$

  This implies that, per Rule 4.10:

  $$(\mathcal{X}^{\mathcal{S}}(f_s(q')) = \mathsf{dom}(\mathfrak{r})) \qquad \subseteq \qquad \{f_t(q', x)\} \cup \{f_t(q', y) \mid \forall y \in \mathcal{X}^{\mathcal{T}}(q') \setminus \{x\}\}.$$

  Condition (FGS2) tells us that:

  $$|\{f_t(q', x)\} \cup \{f_t(q', y) \mid \forall y \in \mathcal{X}^{\mathcal{T}}(q') \setminus \{x\}\}| = |\mathcal{X}^{\mathcal{T}}(q')|.$$

  So $|\mathcal{X}^{\mathcal{S}}(f_s(q'))| \leq |\mathcal{X}^{\mathcal{T}}(q')|$. Therefore, since $|\mathcal{X}^{\mathcal{S}}(f_s(q'))| \geq |\mathcal{X}^{\mathcal{T}}(q')|$, $|\mathcal{X}^{\mathcal{S}}(f_s(q'))| = |\mathcal{X}^{\mathcal{T}}(q')|$, as required.

- For the SUT: We perform a case distinction on $x \in \mathcal{X}^{\mathcal{T}}(q)$:
  - $x \in \mathcal{X}^{\mathcal{T}}(q)$. We know from $x \in \mathcal{X}^{\mathcal{T}}(q)$, Rule 4.4 and Equation (C.12) that:

    $$\mathcal{X}^{\mathcal{T}}(q') = \mathcal{X}^{\mathcal{T}}(q),$$

    By (FMS3):

    $$\tau^{\mathcal{M}}(g_s(q), g_t(i)) = (g_t(x), c).$$

Since $x \in \mathcal{X}^{\mathcal{T}}(q) = \mathcal{X}^{\mathcal{T}}(q')$, we know from (FGS1) that:

$$g_t(x) \in \mathcal{X}^{\mathcal{M}}(g_s(q)).$$

Rule 4.4 now tells us that:

$$\mathcal{X}^{\mathcal{M}}(g_s(q')) \subseteq \mathcal{X}^{\mathcal{M}}(g_s(q)).$$

Therefore:

$$\begin{aligned} |\mathcal{X}^{\mathcal{M}}(g_s(q'))| &\leq |\mathcal{X}^{\mathcal{M}}(g_s(q))| \\ &= |\mathcal{X}^{\mathcal{T}}(q)| && (q \in \mathcal{A}) \\ &= |\mathcal{X}^{\mathcal{T}}(q')|. && (\mathcal{X}^{\mathcal{T}}(q') = \mathcal{X}^{\mathcal{T}}(q)) \end{aligned}$$

So $|\mathcal{X}^{\mathcal{M}}(g_s(q'))| \leq |\mathcal{X}^{\mathcal{T}}(q')|$. Therefore, since $|\mathcal{X}^{\mathcal{M}}(g_s(q'))| \geq |\mathcal{X}^{\mathcal{T}}(q')|$, $|\mathcal{X}^{\mathcal{M}}(g_s(q'))| = |\mathcal{X}^{\mathcal{T}}(q')|$, as required.

– $x \notin \mathcal{X}^{\mathcal{T}}(q)$. Rule 4.4 and Equation (C.12) tell us that:

$$\mathcal{X}^{\mathcal{T}}(q') = \mathcal{X}^{\mathcal{T}}(q) \cup \{x\}.$$

Therefore, since $x \notin \mathcal{X}^{\mathcal{T}}(q)$:

$$|\mathcal{X}^{\mathcal{T}}(q')| = |\mathcal{X}^{\mathcal{T}}(q)| + 1. \tag{C.13}$$

Condition (FMS3) tells us that since $\tau^{\mathcal{T}}(q, i) = (x, c)$, $\tau^{\mathcal{M}}(g_s(q), g_t(i)) = (g_t(x), c)$. Thus, by Rule 4.4:

$$\mathcal{X}^{\mathcal{M}}(g_s(q')) \setminus \{g_t(x)\} \subseteq \mathcal{X}^{\mathcal{M}}(g_s(q)).$$

Therefore:

$$\begin{aligned} |\mathcal{X}^{\mathcal{M}}(g_s(q'))| - 1 &\leq |\mathcal{X}^{\mathcal{M}}(g_s(q))| \\ &= |\mathcal{X}^{\mathcal{T}}(q)| && (q \in \mathcal{A}) \\ &= |\mathcal{X}^{\mathcal{T}}(q')| - 1. && (\textit{Equation (C.13)}) \end{aligned}$$

So $|\mathcal{X}^{\mathcal{M}}(g_s(q'))| \leq |\mathcal{X}^{\mathcal{T}}(q')|$. Therefore, since $|\mathcal{X}^{\mathcal{M}}(g_s(q'))| \geq |\mathcal{X}^{\mathcal{T}}(q')|$, $|\mathcal{X}^{\mathcal{M}}(g_s(q'))| = |\mathcal{X}^{\mathcal{T}}(q')|$, as required.

Since in all cases, $|\mathcal{X}^{\mathcal{S}}(f_s(q'))| = |\mathcal{X}^{\mathcal{T}}(q')| = |\mathcal{X}^{\mathcal{M}}(g_s(q'))|$, $q' \in \mathcal{A}$, as required. □

C.9.3 Proof of Lemma 5.8.10

Proof. Let $\mathcal{M} \in \mathcal{U}$. Suppose that *areAllStatesInNStepsPresentAndEnabledExplored*(q, m) has run. Since any state $q \in Q^{\mathcal{T}}$ always corresponds to one state of $\mathcal{M}$, it could take at most $|Q^{\mathcal{M}}| - 1$ transition steps to reach from q a state that represents any particular state of $Q^{\mathcal{M}}$. Therefore, it could take at most $m - 1$ transition steps to reach from q a state that represents any particular state of any MMT $\mathcal{M} \in \mathcal{U}$. Thus, from q, it could take at most m transition steps to reach a transition in $\mathcal{T}$ that represents any transition, and therefore any timeout transition of the SUT that can be reached via $g_s(q)$. We thus know that since *areAllStatesInNStepsPresentAndEnabledExplored*(q, m) has run, there is for each timeout action that can be reached in $\mathcal{M}$ from $g_s(q)$ a corresponding timeout action in $\mathcal{T}$ that can be reached from q.
Every timeout action of an MMT always terminates at least one spanning run. Let $\rho \in \mathit{runs}(\mathcal{M})$ be a y-spanning run that traverses $g_s(q)$. Then for the run π for which $\langle g_s, g_t, g_u \rangle(\pi) = \rho$, we know from the fact that *areAllStatesInNStepsPresentAndEnabledExplored*(q, m) has run that $\pi \in \mathit{runs}(\mathcal{T})$, and from (FMS5) that π is x-spanning, where $g_t(x) = y$. Since ρ traverses $g_s(q)$, $y \in \mathcal{X}^{\mathcal{M}}(g_s(q))$, and since π traverses q, $x \in \mathcal{X}^{\mathcal{T}}(q)$. The fact that $\mathcal{M}$ is t-observable implies that for all $y \in \mathcal{X}^{\mathcal{M}}(g_s(q))$, there exists a y-spanning run

that traverses $g_s(q)$. We have thus shown that when *areAllStatesInNStepsPresentAndEnabledExplored*(q, m), there is, for each $y \in \mathcal{X}^{\mathcal{M}}(g_s(q))$ an $x \in \mathcal{X}^{\mathcal{T}}(q)$ for which $g_t(x) = y$. Thus, by (FMS2):

$$\textit{areAllStatesInNStepsPresentAndEnabledExplored}(q, m) \implies |\mathcal{X}^{\mathcal{T}}(q)| \geq |\mathcal{X}^{\mathcal{M}}(g_s(q))|.$$

Conditions (FMS1) and (FMS2) imply that in general, $|\mathcal{X}^{\mathcal{T}}(q)| \leq |\mathcal{X}^{\mathcal{M}}(g_s(q))|$. Therefore:

$$\textit{areAllStatesInNStepsPresentAndEnabledExplored}(q, m) \implies |\mathcal{X}^{\mathcal{T}}(q)| = |\mathcal{X}^{\mathcal{M}}(g_s(q))|. \quad \text{(C.14)}$$

We can similarly show that since $|Q^{\mathcal{S}}| \leq m$ and $\mathcal{S}$ is t-observable:

$$\textit{areAllStatesInNStepsPresentAndEnabledExplored}(q, m) \implies |\mathcal{X}^{\mathcal{T}}(q)| = |\mathcal{X}^{\mathcal{S}}(f_s(q))|. \quad \text{(C.15)}$$

Let $q_{\mathcal{I}}^{\mathcal{T}} \xrightarrow[u_1]{i_1} q_1 \ldots \xrightarrow[u_n]{i_n} q \in \textit{runs}(\mathcal{T})$. All that remains for us to prove is that:

$$\forall j \in \{1, \ldots, n\} \colon g_u(q_{j-1}, i_j) = \bot \Leftrightarrow u_j = \bot \Leftrightarrow \exists x \colon f_u(q_{j-1}, i_j)(x) \in \mathbb{N}^{>0}. \quad \text{(C.16)}$$

We get:

$$\forall j \in \{1, \ldots, n-1\} \colon g_u(q_{j-1}, i_j) = \bot \Leftrightarrow u_j = \bot \Leftrightarrow \exists x \colon f_u(q_{j-1}, i_j)(x) \in \mathbb{N}^{>0}$$

from $q_{-1} \in \mathcal{A}$. We showed that if $\rho \in \textit{runs}(\mathcal{M})$ is a spanning run that traverses $g_s(q)$, then there is a spanning run $\pi \in \textit{runs}(\mathcal{T})$ that traverses q, where $\langle g_s, g_t, g_u \rangle(\pi) = \rho$. Therefore, $g_u(q_{n-1}, i_n) \neq \bot \implies u_n \neq \bot$. Lemma 5.3.1 conversely tells us that $u_n \neq \bot \implies g_u(q_{n-1}, i_n) \neq \bot$. Therefore, $u_n = \bot \Leftrightarrow g_u(q_{n-1}, i_n) = \bot$. We can similarly use Lemma 5.3.2 to show that $u_n = \bot \Leftrightarrow \exists x \colon f_u(q_{j-1}, i_j)(x) \in \mathbb{N}^{>0}$. Equation (C.16) therefore holds.

Since Equation (C.16), Equation (C.14) and Equation (C.15) all hold:

$$\textit{areAllStatesInNStepsPresentAndEnabledExplored}(q, m) \implies q \in \mathcal{A}_{\mathcal{M}}^{\mathcal{T}} \wedge q \in \mathcal{A}_{\mathcal{S}}^{\mathcal{T}},$$

as required.

□

C.9.4 Proof of Lemma 5.8.11

Proof. Let $q \in \mathcal{A}_p \cup B \cup F^{\leq k}$ and let $q_{-1} \in Q^{\mathcal{T}}$ be the observation tree state such that $q = \delta^{\mathcal{T}}(q_{-1}, i)$ for some $i \in I$. For q to be added to $\mathcal{A}$ by the ExtendActiveExplored rule requires that $q_{-1} \in \mathcal{A}$. We know that $B$and $F^{\leq k}$ are prefix-closed, and we know from Lemma 5.8.8 that $\mathcal{A}_p$ is prefix-closed. Therefore, if the ExtendActiveExplored rule would add $q' \in B \cup F^{\leq k}$ to $\mathcal{A}$, then it would also try to do so for all of q''s ancestors, apart from $q_{\mathcal{I}}^{\mathcal{T}}$. We perform an induction on q:

- Base case: If $q = q_{\mathcal{I}}^{\mathcal{T}}$, then we cannot use the ExtendActiveExplored rule on q since $q_{\mathcal{I}}^{\mathcal{T}}$ has no parent state. However, Algorithm 9 has already added q to $\mathcal{A}$ because a partial MMT's initial state can never have active timers. Therefore, $q \in \mathcal{A}$, as required.
- Inductive step case: If $q \neq q_{\mathcal{I}}^{\mathcal{T}}$, then q has a parent state q_{-1}. We use the induction hypothesis:
  $IH: \ q_{-1} \in \mathcal{A}$.
  We use the induction hypothesis to start from the position where $q_{-1} \in \mathcal{A}$. If:
  1. *areAllStatesInNStepsPresentAndEnabledExplored*$(q, \textit{maxNumSUTStates})$, then by Corollary 5.8.1, $q \in \mathcal{A}_{\mathcal{M}}^{\mathcal{T}}$ and $q \in \mathcal{A}_{\mathcal{S}}^{\mathcal{T}}$. This then implies that $|\mathcal{X}^{\mathcal{S}}(f_s(q))| = |\mathcal{X}^{\mathcal{T}}(q)| = |\mathcal{X}^{\mathcal{M}}(g_s(q))|$. The ExtendActiveExplored rule would eventually be called on q, upon which it would rightfully add q to $\mathcal{A}$, as required.
  2. $(\tau^{\mathcal{T}}(q_{-1}, i) \neq \bot \wedge (\forall x \in \mathcal{X}(q_{-1}) \colon x \in \mathcal{X}(q)))$. Lemma 5.8.9 now tells us that since $q_{-1} \in \mathcal{A}$, $q \in \mathcal{A}$. The ExtendActiveExplored rule would eventually be called on q, upon which it would rightfully add q to $\mathcal{A}$, as required.

3. neither of the first two conditions hold, then we know that there is at least one state q' that is reached within *maxNumSUTStates* transition steps from q for which at least one of the following two conditions holds:
   - There is an input $i \in I$, for which $\delta^{\mathcal{T}}(q', i)\uparrow$. Then the conditions for the FindingInputActions rule are met, and the procedure from Algorithm 9 will eventually use the rule to add an input transition for i from q'.
   - $q' \notin \mathcal{E}$. Then the conditions for the FindingTimeoutActions rule are met, and the procedure from Algorithm 9 will eventually use the rule to make q' enabled explored.

   This third condition will hold for q for as long as neither of the first two conditions hold. The procedure from Algorithm 9 will therefore keep using the FindingInputActions and FindingTimeoutActions rules until one of the first two conditions holds for q, where the first condition is always guaranteed to hold eventually. Once either of the first two conditions holds, Algorithm 9 will eventually use the ExtendActiveExplored rule to add q to $\mathcal{A}$, granted that no conflicts arise before then.

We thus know that for all $q \in \mathcal{A}_p \cup B \cup F^{\leq k}$, q is either already in $\mathcal{A}$, or it will eventually be added to $\mathcal{A}$, granted that no conflicts arise before then. □

C.9.5 Proof of Lemma 5.8.16

Proof. The IdentifyBasisStates rule can only be applied as many times are there are elements in state cover C. Therefore, since $\mathcal{S}$ has a finite number of elements, C is finite as well, and the IdentifyBasisStates rule can only be applied a finite number of times. The size of the basis is therefore also finite.
The finite size of the basis also imposes a limit on the size that the 0-frontier can reach through repeated application of the ExtendFrontiersWithInputs rule. The fact that the maximum size of the 0-frontier is finite in turn implies that the maximum size of the 1-level frontier is finite, and so on. The ExtendFrontiersWithInputs rule can only be applied a finite number of times, since the maximum size of any frontier is always finite, which implies that the maximum sizes of the first $k+1$ frontiers are always finite.
The ExtendEnabledExplored rule can only be applied as many times as there are elements in the finite set of states $B \cup F^{\leq k}$, since once it runs for one of these states, the state becomes enabled explored, and the rule can never be applied to this state again.
The ExtendActiveExplored rule can only be applied once to any given state $q \in Q^{\mathcal{T}}$, since it adds q to $\mathcal{A}$, after which it can never be run on q again. The total number of states $q \in (\mathcal{A}_p \cup B \cup F^{\leq k})$ on which the rule will be run is finite, since the number of states in $B \cup F^{\leq k}$ is finite, and since only the sub-procedure *addTransitionsFromSpecSeqAndMakeActiveExplored* adds states to $\mathcal{A}_p$. This sub-procedure is only used in *makeEnabledExplored*$^{\mathcal{S}}$ to extend $Q^{\mathcal{T}}$ and $\mathcal{A}_p$. Sub-procedure *makeEnabledExplored*$^{\mathcal{S}}$ is only used to make states from the basis and the first $k+1$ frontiers apart from other states. Since there are only ever finitely many states in the basis and the first $k+1$ frontiers, only a finite number of states may be added to $\mathcal{A}_p$.
We argued in the proof of Lemma 5.8.11 that whenever the FindingInputActions rule or the FindingTimeoutActions rule is applied to an observation tree state $\mathcal{T}$, this observation tree state will be added to $\mathcal{A}$. These two rules can then never be applied on q again. We already argued that the size of the set $(\mathcal{A}_p \cup B \cup F^{\leq k})$ of the states on which these two rules can be applied is always finite. Therefore, the FindingInputActions and FindingTimeoutActions rules can only be applied a finite number of times.
Every application of the IdentifyFrontiers rule makes a state from the $k+1$-level frontier apart from at least one basis state. This rule can only be applied a finite number of times, since the number of states in the $k+1$-level frontier and the number of basis states are both finite.
Every application of the ExtendCoTransitivity rule makes a state from the first k frontiers apart from a state from the $k+1$-level frontier. This rule can only be applied a finite number of times, since the number of states in any frontier is always finite.
We may thus conclude that all eight of Algorithm 9's rules can only be applied a finite number of times. The loop-condition of Line 9 will thus always be met after a finite number of loop iterations. The algorithm will therefore always terminate within a finite number of rule applications. □