Timed Hyperproperties[⋆,⋆⋆]

Hsi-Ming Ho[a,1], Ruoyu Zhou[b], Timothy M. Jones[b]

[a] *Department of Informatics, University of Sussex, Brighton, UK*
[b] *Department of Computer Science and Technology, University of Cambridge, Cambridge, UK*

Abstract

We study the satisfiability and model-checking problems for *timed hyperproperties* specified with HyperMITL, a timed extension of HyperLTL. While the satisfiability problem can be solved similarly as for HyperLTL, we show that the model-checking problem for HyperMITL, unless the specification is alternation-free, is undecidable even when very restricted timing constraints are allowed. On the positive side, we show that model checking HyperMITL with quantifier alternations is possible under certain semantic restrictions. As an intermediate tool, we give an 'asynchronous' interpretation of Wilke's monadic logic of relative distance ($\mathcal{L}_d^{\leftrightarrow}$) and show that it characterises timed languages recognised by timed automata with silent transitions.

Keywords:
Timed Automata; Temporal Logics; Cybersecurity

1. Introduction

1.1. Background

One of the most popular specification formalisms for reactive systems is *Linear Temporal Logic* (LTL), first introduced into computer science by Pnueli [1] in the late 1970s. The success of LTL can be attributed to the fact that its satisfiability and model-checking problems are of lower complexity (PSPACE-complete [2], as compared with non-elementary [3] for the equally expressive *first-order logic of order* (FO[<])). Moreover, LTL enjoys simple translations into automata and excellent tool support (e.g., [4, 5]).

While LTL is adequate for describing features of *individual* execution traces, many security policies are based on relations between *two (or more)* execution traces. A standard example of such properties is *observational determinism* [6, 7, 8]: for every pair of execution traces, if the low-security inputs agree in both execution traces, then the low-security outputs in both execution traces must agree as well. Such properties are called *hyperproperties* [9]: a model of the property is not a single execution trace but a set of execution traces. HyperLTL [10], obtained from LTL by adding *trace quantifiers*, has been proposed as a specification formalism to express hyperproperties. For example, observational determinism can be expressed as a HyperLTL formula:

$$\forall \pi_a \, \forall \pi_b \; \mathbf{G}(I_a = I_b) \Rightarrow \mathbf{G}(O_a = O_b)\,.$$

HyperLTL inherits almost all the benefits of LTL; in particular, tools that support HyperLTL verification can be built by leveraging existing tools for LTL.

[⋆] This work was supported by the Engineering and Physical Sciences Research Council (EPSRC), through grant references EP/K026399/1 and EP/P020011/1.

[⋆⋆] A preliminary version of this paper appeared in the *26th International Symposium on Temporal Representation and Reasoning (TIME 2019).*

Email addresses: Hsi-Ming.Ho@sussex.ac.uk (Hsi-Ming Ho), ruoyu.zhou@cl.cam.ac.uk (Ruoyu Zhou), timothy.jones@cl.cam.ac.uk (Timothy M. Jones)

[1] Most of the work was conducted while the first author was affiliated with the University of Cambridge.

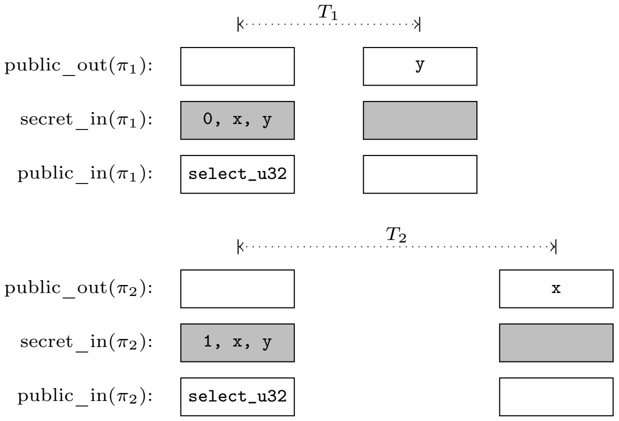


Figure 1: An attacker can infer the value of the secret selection bit `b` if $T_1 \neq T_2$.

For many practical applications, however, in addition to the occurences and orders of events, *timing* has to be accounted for as well. For example, one may want to verify that in every execution trace of the system, whenever a request `req` is issued, the corresponding acknowledgement `ack` is received within the next 5 time units. *Timed automata* [11] and *timed logics* [12, 13, 14] are introduced exactly for this purpose, and it is natural to consider timed variants of observational determinism (and similar properties) in this context. Indeed, timing anomalies caused by different high-security inputs is a well-known attack vector that can be exploited to obtain sensitive information [15]; this kind of *timing side-channel attacks* also play significant roles in recent high-profile exploits such as Meltdown [16] and Spectre [17]. Below is a simple yet realistic example of such attacks at the instruction level.

Example 1 ([18])**.** A C function that selects between two secret variables `x` and `y` based on a secret selection bit `b` (i.e. the low-security user has access to the low-security output—the value of either `x` or `y`—but should not know it is the value of which of `x` or `y`) may be written as follows:

```
uint32_t select_u32(uint32_t b, uint32_t x, uint32_t y)
{
    return b ? x : y;
}
```

This straightforward implementation, however, may result in a timing side channel—depending on which compiler optimisations are applied, the execution time may depend on `b`. If this is the case then an attacker, provided that he/she can observe the time when the output is returned, can infer `b` (depicted in Figure 1).[2] In sensitive applications like cryptography libraries and embedded smartcard software, such functions are usually replaced by functional-equivalent, 'branch-less' versions, with the hope of eliminating potential differences in execution times without sacrificing portability. In this case, one such version is as follows:

```
uint32_t ct_select_u32(uint32_t b, uint32_t x, uint32_t y)
{
    signed bit = 0 - b;
    return (x & bit) | (y & ~bit);
}
```

Unfortunately, such obfuscation efforts can easily be wiped out by more agressive code optimisations. For instance, after compilation by `clang 3.3` (`-O2`), the C function `ct_select_u32` results in the following assembly code, which contains a conditional jump instruction (branch). This piece of code could reveal `b`, as execution times may differ due to branch prediction. We remark that similar issues are of large practical interest and various approaches to detecting them have been proposed, e.g., [19, 20, 21, 22].

[2] While events (and sets of events) are shown as rectangles in the figures, they are assumed to occur instantaneously.

```
ct_select_u32:
    mov    0x4(%esp),%al
    test   %al,%al
    jne    L
    lea    0xc(%esp),%eax
    mov    (%eax),%eax
    ret
L:  lea    0x8(%esp),%eax
    mov    (%eax),%eax
    ret
```

Given the highly sophisticated cache hierarchies, pipeline stalls, etc. in contemporary real machines, the timing side channel in the example above may be difficult to realise and exploit in an actual attack; but such issues may also manifest themselves in hardware designs (e.g., at the register-transfer level), as illustrated by the following example.

Example 2 ([23])**.** An AND gate with two secret inputs A, B, and a public output C (with respective delays T_A, T_B, and T_C; we assume that $T_B > T_A$ and $T_C > T_B - T_A$) can be modelled as the *timed automaton* with two clocks x, y in Figure 2 (see Section 2.2 for a definition of timed automata). Intuitively, the values of A and B are only known after T_A and T_B, respectively; the output $C = A \wedge B$, on the other hand, has an extra delay of T_C from the point when its value is determined. So for example, if A turned out to be 0 (i.e. event A^0 has occurred), the output C must be 0 as well (i.e. C^0 must occur later). But if $A = 1$ and $B = 0$, then C^0 will occur at a later time than in the previous case. In other words, when $C = 0$, a low-security user (to whom A^0 and A^1 should not be observable), provided that he/she can measure the time when C^0 occurs, can also infer the value of A while he/she should not be able to. The pair of traces with $C = 0$ that reveals A is depicted in Figure 3 and Figure 4. In this simple example, however, the timing side channel can be easily removed by adding $y := 0$ on the self-loop on the lower-right location.

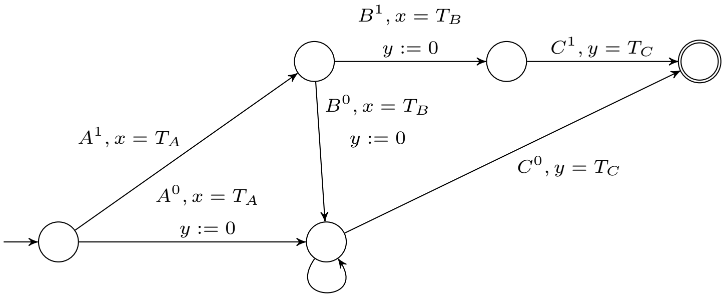


Figure 2: A timed automaton modelling an AND gate with inputs A, B and output C with respective delays T_A, T_B, and T_C.

1.2. Contributions

We propose HyperMITL, obtained by adding trace quantifiers to *Metric Interval Temporal Logic* (MITL) [14], as a specification formalism for *timed hyperproperties*. We consider systems modelled as *timed automata* whose behaviours are sequences of *events* that occur at different instants in time (i.e. *timed words*). It is not hard to see that, as far as satisfiability is concerned, HyperMITL is similar to HyperLTL, i.e. satisfiability is decidable for fragments not containing $\forall\exists$. However, in contrast with HyperLTL (whose model-checking problem is decidable), model checking HyperMITL is undecidable if there is at least one quantifier alternation in the specification, even when the timing constraints used in either the system or the specification are very restricted. Still, the alternation-free fragment of HyperMITL, which is arguably sufficient to capture many timed hyperproperties of practical interest, has a decidable model-checking problem. We also identify a number of subcases where HyperMITL model checking becomes decidable for larger fragments, such as when the specification belongs to a certain subclass of one-clock timed automata, or when there is an *a priori* bound on either the *length of the time domain* or the *variability* of the traces involved. The results reveal the connections between HyperMITL, timed automata with silent transitions [24], and an 'asynchronous' interpretation of the *monadic logic of relative distance* ($\mathcal{L}_d^{\leftrightarrow}$) of Wilke's [25].

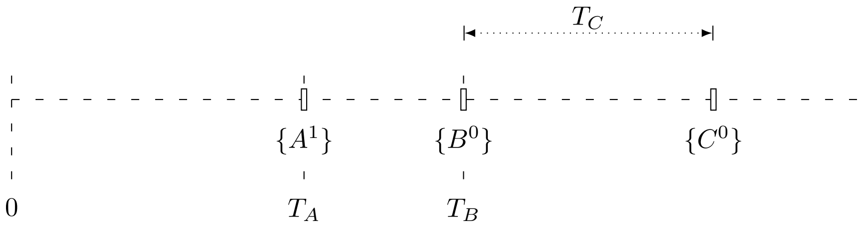


Figure 3: A trace ρ_1 with $A = 1$, $B = 0$, and $C = 0$.

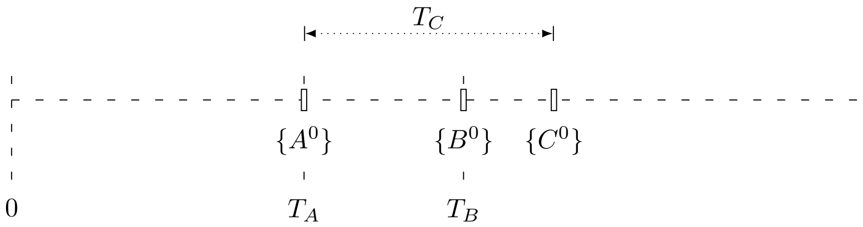


Figure 4: A trace ρ_2 with $A = 0$, $B = 0$, and $C = 0$.

1.3. Related work

Since the pioneering work of Clarkson and Schneider [9], there has been great interest in specifying and verifying hyperproperties in the past few years. The framework based on HyperLTL [10] is possibly the most popular for this purpose, thanks to its expressiveness, flexibility, and relative ease of implementation. In addition to satisfiability [26, 27] and model checking [10, 28], tools for monitoring HyperLTL also exist [29, 30, 31]. Notably, the complexity of monitoring HyperLTL, as well as model checking HyperLTL on restricted (tree-shaped or acyclic) Kripke structures, are studied in [32] and shown to be much lower than those of the general satisfiability and model-checking problems. These results, however, do not apply in the current timed setting—we will see in Section 4.3 that our main undecidability result holds even with these structural restrictions on the system model.

Our formulation of HyperMITL is very closely related to HyperSTL [33] originally proposed in the context of quality assurance of cyber-physical systems. While [33] focusses on testing, we are mainly concerned with the decidability of verification problems. On the other hand, the semantics of HyperSTL is defined over sets of continuous signals, i.e. *state-based*; as noted in [33], however, the price to pay for this extra generality is that implementing a model checker for HyperSTL is difficult in practice, especially for systems modelled in proprietary frameworks, such as Simulink®. Practical reasoning of (*event-based*) HyperMITL, by contrast, is more amenable to implementation based on existing highly optimised timed automata verification back ends, e.g., Uppaal [34].[3] Indeed, a prototype model checker based on Uppaal for the synchronous semantics of HyperMITL (with some restrictions) is reported in [37], although it does not consider the decidability of verification problems. Another relevant work [38], also based on Uppaal, checks noninterference in systems modelled as timed automata (similar to Example 4; see below). Their approach, however, is specifically tailored to noninterference and does not generalise. Some similar (but different) notions of noninterference for timed automata have been considered in [39, 40].

It is also possible to extend hyperlogics in other quantitative dimensions orthogonal to time. HyperPCTL [41] can express *probabilisitic hyperproperties*, e.g., the probability distribution of the low-security outputs is independent of the high-security inputs. In [42], specialised algorithms are developed for verifying *quantitative hyperproperties*, e.g., there is a bound on the number of traces with the same low-security inputs but different low-level outputs. The current paper is complementary to these works.

[3] For more detailed accounts of the state-based and event-based semantics for timed automata and logics, see e.g., [35, 36].

2. Timed hyperproperties

2.1. Timed words

A *timed word* over a non-empty finite set Σ of *events* is a non-empty finite sequence $(\sigma_1, \tau_1) \dots (\sigma_n, \tau_n) \in (\Sigma \times \mathbb{R}_{\geq 0})^*$ where $\tau_1 \dots \tau_n$ is an increasing sequence of non-negative real numbers ('*timestamps*'), i.e. $\tau_i < \tau_{i+1}$ for all i, $1 \leq i < n$.[4] For $t \in \mathbb{R}_{\geq 0}$ and a timed word $\rho = (\sigma_1, \tau_1) \dots (\sigma_n, \tau_n)$, we write $t \in \rho$ iff $t = \tau_i$ for some i, $1 \leq i \leq n$. We denote by $T\Sigma^*$ the set of all timed words over Σ. A *timed language* (or a *trace property*) is a subset of $T\Sigma^*$.

2.2. Timed automata

Let X be a finite set of *clocks* ($\mathbb{R}_{\geq 0}$-valued variables). A *valuation* v for X maps each clock $x \in X$ to a value in $\mathbb{R}_{\geq 0}$. The set $G(X)$ of *clock constraints* (*guards*) g over X is generated by $g := \top \mid g \wedge g \mid x \bowtie c$ where $\bowtie \in \{\leq, <, \geq, >\}$, $x \in X$, and $c \in \mathbb{N}_{\geq 0}$. The satisfaction of a guard g by a valuation v (written $v \models g$) is defined in the usual way. For $t \in \mathbb{R}_{\geq 0}$, we let $v + t$ be the valuation defined by $(v + t)(x) = v(x) + t$ for all $x \in X$. For $\lambda \subseteq X$, we let $v[\lambda \leftarrow 0]$ be the valuation defined by $(v[\lambda \leftarrow 0])(x) = 0$ if $x \in \lambda$, and $(v[\lambda \leftarrow 0])(x) = v(x)$ otherwise.

A *timed automaton* (TA) over Σ is a tuple $\mathcal{A} = \langle \Sigma, S, s_0, X, \Delta, F \rangle$ where S is a finite set of locations, $s_0 \in S$ is the initial location, X is a finite set of clocks, $\Delta \subseteq S \times \Sigma \times G(X) \times 2^X \times S$ is the transition relation, and F is the set of accepting locations. We say that $\mathcal{A}$ is *deterministic* iff for each $s \in S$ and $\sigma \in \Sigma$ and every distinct pair of transitions $(s, \sigma, g^1, \lambda^1, s^1) \in \Delta$ and $(s, \sigma, g^2, \lambda^2, s^2) \in \Delta$, $g^1 \wedge g^2$ is not satisfiable. A *state* of $\mathcal{A}$ is a pair (s, v) of a location $s \in S$ and a valuation v for X. A *run* of $\mathcal{A}$ on a timed word $(\sigma_1, \tau_1) \dots (\sigma_n, \tau_n) \in T\Sigma^*$ is a sequence of states $(s_0, v_0) \dots (s_n, v_n)$ where (i) $v_0(x) = 0$ for all $x \in X$ and (ii) for each i, $0 \leq i < n$, there is a transition $(s_i, \sigma_{i+1}, g, \lambda, s_{i+1})$ such that $v_i + (\tau_{i+1} - \tau_i) \models g$ (let $\tau_0 = 0$) and $v_{i+1} = \big(v_i + (\tau_{i+1} - \tau_i)\big)[\lambda \leftarrow 0]$. A run of $\mathcal{A}$ is *accepting* iff it ends in a state (s, v) with $s \in F$. A timed word is *accepted* by $\mathcal{A}$ iff $\mathcal{A}$ has an accepting run on it. We denote by $[\![\mathcal{A}]\!]$ the timed language of $\mathcal{A}$, i.e. the set of all timed words accepted by $\mathcal{A}$. Some fundamental results on TAs are that the *emptiness* problem (is $[\![\mathcal{A}]\!] = \emptyset$?) is decidable (PSPACE-complete), but the *universality* problem (is $[\![\mathcal{A}]\!] = T\Sigma^*$?) and the *language inclusion* problem (is $[\![\mathcal{A}_1]\!] \subseteq [\![\mathcal{A}_2]\!]$?) are undecidable [11].

A *timed automaton with ϵ-transitions* (TA_ϵ) additionally admits transitions of the form $(s, \epsilon, g, \lambda, s')$ where ϵ is a special '*silent*' event. In a run of a TA_ϵ on a timed word $(\sigma_1, \tau_1) \dots (\sigma_n, \tau_n) \in T\Sigma^*$, such transitions can be taken (one or more times) between each pair of ordinary events $\sigma_i, \sigma_{i+1} \in \Sigma$. In contrast to the untimed case where ϵ-transitions can be removed with standard textbook constructions (e.g., [43]), it is known that TA_ϵ's are strictly more expressive than TAs [24]; in particular, universality becomes decidable for TAs if the automaton in question uses only one clock [44], but it remains undecidable for TA_ϵ's in that case.

2.3. Timed logics

The set of MITL formulae over a finite set of atomic propositions AP is generated by

$$\psi := \top \mid p \mid \psi_1 \wedge \psi_2 \mid \neg\psi \mid \psi_1 \, \overline{\mathbf{U}}_I \, \psi_2 \mid \psi_1 \, \overline{\mathbf{S}}_I \, \psi_2$$

where $p \in \mathsf{AP}$ and $I \subseteq \mathbb{R}_{\geq 0}$ is a *non-singular* interval with endpoints in $\mathbb{N}_{\geq 0} \cup \{\infty\}$. We omit the subscript I when $I = [0, \infty)$ and sometimes write pseudo-arithmetic expressions for constraining intervals, e.g., '< 3' for $[0, 3)$. The other Boolean operators are defined as usual: $\bot \equiv \neg\top$ and $\psi_1 \vee \psi_2 \equiv \neg(\neg\psi_1 \wedge \neg\psi_2)$. We also define the dual temporal operators $\psi_1 \widetilde{\overline{\mathbf{U}}}_I \psi_2 \equiv \neg\big((\neg\psi_1) \, \overline{\mathbf{U}}_I \, (\neg\psi_2)\big)$ and $\psi_1 \widetilde{\overline{\mathbf{S}}}_I \psi_2 \equiv \neg\big((\neg\psi_1) \, \overline{\mathbf{S}}_I \, (\neg\psi_2)\big)$. Using these operators, every MITL formula ψ can be transformed into an MITL formula in *negative normal form*, i.e. $\neg$ is only applied to atomic propositions. To ease the presentation, we will also use the usual shortcuts like $\overline{\mathbf{F}}_I \, \psi \equiv \top \, \overline{\mathbf{U}}_I \, \psi$, $\overline{\mathbf{G}}_I \, \psi \equiv \neg \, \overline{\mathbf{F}}_I \, \neg\psi$, $\mathbf{X}_I \, \psi \equiv \bot \, \overline{\mathbf{U}}_I \, \psi$, and the more popular 'weak-future' variants of temporal

[4] We focus on *finite* timed words in this paper; most of our results (except those rely on the decidability of universality of one-clock TAs over finite timed words) carry over to the case of infinite timed words with some simple modifications. For example, in Section 3, suitable subformulae can be added to rule out the runs that get stuck in self-loops labelled with $\{p^\epsilon\}$.

operators, e.g., $\mathbf{F}\,\psi \equiv \psi \vee \overline{\mathbf{F}}\,\psi$. Given an MITL formula ψ over AP in negative normal form, a timed word $\rho = (\sigma_1, \tau_1) \dots (\sigma_n, \tau_n)$ over $\Sigma_{\mathsf{AP}} = 2^{\mathsf{AP}}$, and $t \in \mathbb{R}_{\geq 0}$, we define the MITL satisfaction relation $\models$ as follows:[5]

- $(\rho, t) \models \top$ iff $t \in \rho$;
- $(\rho, t) \models \bot$ iff $t \notin \rho$;
- $(\rho, t) \models p$ iff $t = \tau_i$ for some i, $1 \leq i \leq n$ and $p \in \sigma_i$;
- $(\rho, t) \models \neg p$ iff $t = \tau_i$ for some i, $1 \leq i \leq n$ and $p \notin \sigma_i$;
- $(\rho, t) \models \psi_1 \wedge \psi_2$ iff $(\rho, t) \models \psi_1$ and $(\rho, t) \models \psi_2$;
- $(\rho, t) \models \psi_1 \vee \psi_2$ iff $(\rho, t) \models \psi_1$ or $(\rho, t) \models \psi_2$;
- $(\rho, t) \models \psi_1\, \overline{\mathbf{U}}_I\, \psi_2$ iff there exists $t' > t$ such that $t' - t \in I$, $(\rho, t') \models \top$, $(\rho, t') \models \psi_2$, and $(\rho, t'') \models \psi_1$ for all t'' such that $t'' \in (t, t')$ and $(\rho, t'') \models \top$;
- $(\rho, t) \models \psi_1 \widetilde{\overline{\mathbf{U}}}_I \psi_2$ iff for all $t' > t$ such that $t' - t \in I$ and $(\rho, t') \models \top$, either $(\rho, t') \models \psi_2$ or $(\rho, t'') \models \psi_1$ for some t'' such that $t'' \in (t, t')$ and $(\rho, t'') \models \top$;
- $(\rho, t) \models \psi_1\, \overline{\mathbf{S}}_I\, \psi_2$ iff there exists t', $0 \leq t' < t$ such that $t - t' \in I$, $(\rho, t') \models \top$, $(\rho, t') \models \psi_2$, and $(\rho, t'') \models \psi_1$ for all t'' such that $t'' \in (t', t)$ and $(\rho, t'') \models \top$;
- $(\rho, t) \models \psi_1 \widetilde{\overline{\mathbf{S}}}_I \psi_2$ iff for all t', $0 \leq t' < t$ such that $t - t' \in I$ and $(\rho, t') \models \top$, either $(\rho, t') \models \psi_2$ or $(\rho, t'') \models \psi_1$ for some t'' such that $t'' \in (t', t)$ and $(\rho, t'') \models \top$.

We say that ρ *satisfies* ψ ($\rho \models \psi$) iff $(\rho, 0) \models \psi$, and we write $[\![\psi]\!]$ for the timed language of ψ, i.e. the set of all timed words satisfying ψ. It is well known that any MITL formula can be translated into a TA accepting the same timed language [47]; this implies that the satisfiability and model-checking problems for MITL are decidable (EXPSPACE-complete).

2.4. Adding trace quantifiers

Let V be an infinite supply of *trace variables*, the set of HyperMITL formulae over AP are generated by

$$\begin{aligned} \phi &:= \exists \pi\, \phi \mid \forall \pi\, \phi \mid \psi \\ \psi &:= \top \mid \top_\pi \mid p_\pi \mid \psi_1 \wedge \psi_2 \mid \neg \psi \mid \psi_1\, \overline{\mathbf{U}}_I\, \psi_2 \mid \psi_1\, \overline{\mathbf{S}}_I\, \psi_2 \end{aligned}$$

where $\pi \in V$, $p \in \mathsf{AP}$, and $I \subseteq \mathbb{R}_{\geq 0}$ is a non-singular interval with endpoints in $\mathbb{N}_{\geq 0} \cup \{\infty\}$ (to ease the notation, we will usually write, e.g., p_a for p_{π_a}). Without loss of generality we forbid the reuse of trace variables, i.e. each trace quantifier must use a fresh trace variable. Syntactic sugar is defined as in MITL, e.g., $\overline{\mathbf{F}}_I\, \psi \equiv \top\, \overline{\mathbf{U}}_I\, \psi$. A HyperMITL formula is *closed* if it does not have free occurrences of trace variables. Following [48], we refer to fragments of HyperMITL by their quantifier patterns, e.g., $\exists^*\forall^*$-HyperMITL. Finally, note that trace quantifiers can be added to TAs in the same manner (in this case, quantified TAs operate over 'stacked' traces; see the semantics for HyperMITL below).

In contrast with TAs and MITL formulae, which define *trace properties*, HyperMITL formulae define *(timed) hyperproperties*, i.e. sets of trace properties. Depending on whether one requires timestamps in quantified traces to match exactly (i.e. all quantified traces must *synchronise*), two possible semantics can be defined accordingly.

[5] The formulation of the pointwise semantics of MITL here deviates slightly from the standard one (cf. [45, 46]) to enable a natural treatment of interleaving of events in different traces.

2.5. Asynchronous semantics

A *trace assignment* Π over Σ is a partial mapping from V to $T\Sigma^*$. We write $\Pi_\emptyset$ for the empty trace assignment and $\Pi[\pi \mapsto \rho]$ for the trace assignment that maps π to ρ and π' to $\Pi(\pi')$ for all $\pi' \neq \pi$. Given a $\mathsf{HyperMITL}$ formula ϕ over AP whose quantifier-free part is in negative normal form, a trace set T over Σ_{AP}, a trace assignment Π over Σ_{AP} with $\mathit{range}(\Pi) \subseteq T$, and $t \in \mathbb{R}_{\geq 0}$, we define the $\mathsf{HyperMITL}$ *asynchronous* satisfaction relation $\models$ as follows (we omit the cases where the definitions are obvious or exactly similar):

- $(T, t) \models_\Pi \top$ iff $t \in \rho$ for some $\rho \in \mathit{range}(\Pi)$;[6]
- $(T, t) \models_\Pi \top_\pi$ iff $t \in \rho$ for $\rho = \Pi(\pi)$;
- $(T, t) \models_\Pi p_\pi$ iff $t \in \rho$ for $\rho = \Pi(\pi)$ and $p \in \sigma_i$ for the event (σ_i, t) in ρ;
- $(T, t) \models_\Pi \psi_1 \overline{\mathbf{U}}_I \psi_2$ iff there exists $t' > t$ such that $t' - t \in I$, $(T, t') \models_\Pi \top$, $(T, t') \models_\Pi \psi_2$, and $(T, t'') \models_\Pi \psi_1$ for all t'' such that $t'' \in (t, t')$ and $(T, t'') \models_\Pi \top$;
- $(T, t) \models_\Pi \psi_1 \widetilde{\overline{\mathbf{U}}}_I \psi_2$ iff for all $t' > t$ such that $t' - t \in I$ and $(T, t') \models_\Pi \top$, either $(T, t') \models_\Pi \psi_2$ or $(T, t'') \models_\Pi \psi_1$ for some t'' such that $t'' \in (t, t')$ and $(T, t'') \models_\Pi \top$;
- $(T, t) \models_\Pi \exists \pi \, \phi$ iff there is a trace $\rho \in T$ such that $(T, t) \models_{\Pi[\pi \mapsto \rho]} \phi$;
- $(T, t) \models_\Pi \forall \pi \, \phi$ iff for all traces $\rho \in T$, $(T, t) \models_{\Pi[\pi \mapsto \rho]} \phi$.

We say that T *satisfies* a closed $\mathsf{HyperMITL}$ formula ϕ in the asynchronous semantics ($T \models \phi$) iff $(T, 0) \models_{\Pi_\emptyset} \phi$.

The asynchronous semantics for $\mathsf{HyperMITL}$ is the natural choice of semantics for the current event-based setting. As the examples below illustrate, allowing explicit interleaving of events may simplify the specification even when no quantitative timing constraint is involved.

Example 3. Consider again the system in Example 2 and a low-security user u_L who can observe $\{B^0, B^1, C^0, C^1\}$ but not $\{A^0, A^1\}$. The property "if B^0 occurs in both π_a and π_b, then the corresponding C^0's must occur simultaneously in both π_a and π_b" (a variant of noninference [49]) can be specified with the following $\mathsf{HyperMITL}$ formula in the asynchronous semantics:

$$\phi_1 = \forall \pi_a \, \forall \pi_b \left(\mathbf{F} \, B^0_a \wedge \mathbf{F} \, B^0_b \Rightarrow \mathbf{F}(C^0_a \wedge C^0_b) \right).$$

In particular, $C^0_a \wedge C^0_b$ holds only when the two $\{C^0\}$-events occur simultaneously in π_a and π_b. It is clear that the system does not satisfy ϕ_1, as there are two traces of the system where B^0 occurs in both, but the occurrences of C^0 are at different times; as we mentioned earlier, this allows u_L to infer A by timing C^0. If, on the other hand, the timing accuracy attainable by u_L is limited in that it can only differentiate events that are more than d time units apart, the system can instead be checked against

$$\phi_2 = \forall \pi_a \, \forall \pi_b \left(\mathbf{F} \, B^0_a \wedge \mathbf{F} \, B^0_b \Rightarrow \mathbf{F} \left(C^0_a \wedge (\mathbf{F}_{\leq d} \, C^0_b \vee \mathbf{O}_{\leq d} \, C^0_b) \right) \right)$$

where the $\mathbf{O}$ operator is the past version of the $\mathbf{F}$ operator. This will be satisfied if $T_B - T_A \leq d$, and since u_L will not be able to infer A, the system may be considered secure in this case. Finally, note that in the original (synchronous) semantics for $\mathsf{HyperLTL}$ [10], ϕ_1 is satisfied by the system, as events are synchronised by their positions rather than times of occurrence.

Example 4 (Noninterference in event-based systems [50])**.** A system operating on sequences of commands issued by different users can be modelled as a deterministic finite automaton $\mathcal{A}$ over $\Sigma = U \times C$ where U is the set of users and C is the set of commands. Additionally, let Obs be the set of observations and $\mathit{out} \colon S \times U \to \mathsf{Obs}$ be the observation function for what can be observed at each location by each user. Let

[6] Note the dependency of the interpretation of $\top$ on Π; in particular, it is possible for a trace set with out-of-sync traces to satisfy $\forall \pi_b \, (p_b \, \overline{\mathbf{U}} \, q_b)$ but not $\forall \pi_a \, \forall \pi_b \, (p_b \, \overline{\mathbf{U}} \, q_b)$.

there be a partition of U into two disjoint sets of users $U_H \subseteq U$ and $U_L \subseteq U$. *Noninterference* requires that for each $w \in \Sigma^*$ where w ends with a command issued by a user in U_L and $\mathcal{A}$ reaches s after reading w, the subsequence w' obtained by removing all the commands issued by the users in U_H results in a location s' such that $out(s', u_L) = out(s, u_L)$ for each user $u_L \in U_L$. For our purpose, we can combine $\mathcal{A}$ and out (in the expected way) into an automaton $\mathcal{A}'$ over Σ_{AP} where $\mathsf{AP} = (U \times C) \uplus (U \times \mathsf{Obs})$ (atomic propositions in $U \times \mathsf{Obs}$ reflect the observations at the location that has just been entered). Checking noninterference then amounts to model checking $\mathcal{A}'$ (whose locations are all accepting) against the following HyperMITL formula in the asynchronous semantics:

$$\phi_3 = \forall \pi_a \, \forall \pi_b \, \big(\mathbf{G}(\top_b \Rightarrow \psi^L_b \wedge \psi^{=}_{U,C}) \wedge \mathbf{G}(\top_a \wedge \bot_b \Rightarrow \psi^H_a) \Rightarrow \mathbf{G}(\top_b \Rightarrow \psi^{=}_{out(U_L)}) \big)$$

(where ψ^L_b asserts that the command in π_b is issued by a user in U_L, $\psi^{=}_{U,C}$ says that the two synchronised commands in π_a and π_b agree on U and C, etc.). Specifically,

- $\mathbf{G}(\top_b \Rightarrow \psi^L_b \wedge \psi^{=}_{U,C})$ asserts that π_b only contains low commands and π_a also contains these commands at exactly the same instants;
- $\mathbf{G}(\top_a \wedge \bot_b \Rightarrow \psi^H_a)$ asserts that all the commands that are only present in π_a are high commands;
- $\mathbf{G}(\top_b \Rightarrow \psi^{=}_{out(U_L)})$ ensures that, after each low command in π_b, the observation of each $u_L \in U_L$ is identical to the observation of u_L after the corresponding low command in π_a, regardless of the high commands that occur in the preceding 'gaps'.

We remark that the asynchronous formulation here is much simpler and clearer than the formulation based on the original HyperLTL semantics in [10].

2.6. Synchronous semantics

A *synchronous* semantics, akin to the HyperLTL one we just mentioned, can also be defined for HyperMITL formulae: each trace quantifier only ranges over traces that synchronise with the traces in the 'current' trace assignment. For example, the second quantifier in $\exists \pi_a \, \exists \pi_b \, \psi$ requires π_b to satisfy $(\pi_a, t) \models \top_a \Leftrightarrow (\pi_b, t) \models \top_b$ for all $t \in \mathbb{R}_{\geq 0}$. The HyperMITL synchronous satisfaction relation $\models^{sync}$ can be defined similarly as in Section 2.5 by requiring that the newly quantified trace π synchronises with all the traces in Π or, alternatively, be expressed directly in the asynchronous semantics. More precisely, for a closed HyperMITL formula $\phi = \mathcal{Q} \, \phi'$ where $\mathcal{Q}$ denotes a block of quantifiers of the same type (i.e. all existential or all universal) and ϕ' is a (possibly open) HyperMITL formula, and a set V of trace variables, let (abusing notation slightly) $sync(\phi, V) = \mathcal{Q} \, \big(\mathbf{G}(\bigwedge_{\pi \in \mathcal{Q} \cup V} \top_\pi) \wedge sync(\phi', \mathcal{Q} \cup V) \big)$ when $\mathcal{Q}$ are existential, $sync(\phi) = \mathcal{Q} \, \big(\mathbf{G}(\bigwedge_{\pi \in \mathcal{Q} \cup V} \top_\pi) \Rightarrow sync(\phi', \mathcal{Q} \cup V) \big)$ when $\mathcal{Q}$ are universal, and $sync(\psi, V) = \psi$ when ψ is quantifier free. The following lemma enables us to check, for a trace set T and ϕ as above, whether $T \models^{sync} \phi$ in terms of $\models$ (we rewrite formulae into prenex normal form as appropriate).

Lemma 1. *For any trace set T over Σ_{AP} and closed* HyperMITL *formula ϕ over* AP, *$T \models^{sync} \phi$ iff $T \models sync(\phi, \emptyset)$.*

Proof. Let ψ be the quantifier-free part of ϕ. It is clear that $(T, 0) \models^{sync}_{\Pi} \psi \Leftrightarrow (T, 0) \models_{\Pi} \psi \Leftrightarrow (T, 0) \models_{\Pi} sync(\psi, V)$ for any V, provided that Π is synchronised. Without loss of generality, consider

- $\phi' = \exists \pi \, \psi$: Assume that $(T, 0) \models^{sync}_{\Pi} \phi'$ for some synchronised Π. This immediately implies that $(T, 0) \models^{sync}_{\Pi[\pi \mapsto \rho]} \psi$ for some $\rho \in T$ where $\Pi[\pi \mapsto \rho]$ is also synchronised. From above we have $(T, 0) \models_{\Pi[\pi \mapsto \rho]} sync(\psi, V)$ where $V = V' \cup \{\pi\}$ and V' is arbitrary, which in turn implies $(T, 0) \models_{\Pi} sync(\phi', V')$. Each step can also be reversed, so the converse holds as well.
- $\phi' = \forall \pi \, \psi$: Assume that $(T, 0) \models^{sync}_{\Pi} \phi'$ for some synchronised Π. If $T = \emptyset$ then $(T, 0) \models_{\Pi} sync(\phi', V')$ vacuously holds. If $T \neq \emptyset$, we have $(T, 0) \models^{sync}_{\Pi[\pi \mapsto \rho]} \psi$ for all $\rho \in T$ that synchronises with Π. For each such ρ we have $(T, 0) \models_{\Pi[\pi \mapsto \rho]} sync(\psi, V)$ where $V = V' \cup \{\pi\}$ and V' is arbitrary, and this implies $(T, 0) \models_{\Pi} sync(\phi', V')$. Each step can also be reversed, so the converse holds as well.

Now observe that the steps above are also valid if ϕ' is used as ψ. The claim holds by setting $V' = \emptyset$. $\square$

While the synchronous semantics may seem quite restricted (intuitively, the chance that two random traces of a timed system have exactly the same timestamps is certainly slim!), one can argue that it already suffices for many applications as in many cases the modelled system admits *stuttering steps*. We will see later that for alternation-free HyperMITL, the asynchronous semantics can be emulated in the synchronous semantics using a 'weak inverse' of Lemma 1.

2.7. Satisfiability and model checking

Given a closed HyperMITL formula ϕ over AP, the *satisfiability* problem asks whether there is a *non-empty* trace set $T \subseteq T\Sigma^*_{\mathsf{AP}}$ satisfying it, i.e. $T \models \phi$ (or $T \models^{sync} \phi$, if the synchronous semantics is assumed). Given a TA $\mathcal{A}$ over Σ_{AP} and a closed HyperMITL formula ϕ over AP, the *model-checking* problem asks whether $[\![\mathcal{A}]\!] \models \phi$ (or $[\![\mathcal{A}]\!] \models^{sync} \phi$). Our focus in this paper is on the *decidability* of these problems, as their complexity (when they are decidable) follow straightforwardly from standard results on MITL [14] and HyperLTL [10, 48].

3. Satisfiability

To emulate interleaving of events (of a concurrent or distributed system, say) in a synchronous setting, it is natural and necessary to introduce stuttering steps. In the context of verification, it is often a desirable trait for a temporal logic to be *stutter-invariant* [51, 52] so that it cannot be used to differentiate traces that ought to be regarded as the same (e.g., in an iterative refinement process, an abstract component of a system may be replaced by a concrete implementation that simulates an abstract step with some additional internal actions). As a simple attempt to reconcile the asynchronous and synchronous semantics of HyperMITL, we can make use of the same idea to enable synchronisation of interleaving traces while preserving the semantics. More precisely, let *stutter*(ρ) for a trace $\rho \in T\Sigma^*_{\mathsf{AP}}$ be the maximal set of traces $\rho' \in T\Sigma^*_{\mathsf{AP}_\epsilon}$ (where $\mathsf{AP}_\epsilon = \mathsf{AP} \cup \{p^\epsilon\}$) such that

- for every event (σ_i, τ_i) in ρ', either $\sigma_i = \{p^\epsilon\}$ or $p^\epsilon \notin \sigma_i$;
- ρ can be obtained from ρ' by deleting all the $\{p^\epsilon\}$-events.

This extends to trace sets $T \subseteq T\Sigma^*_{\mathsf{AP}}$ in the obvious way. For a closed alternation-free HyperMITL formula $\phi = \mathcal{Q}\psi$ over AP, let ψ' be the formula obtained by replacing, e.g., all $\top_\pi$ with $\neg p^\epsilon_\pi$, in ψ, and define *stutter*$(\phi) = \mathcal{Q}\psi''$ (a HyperMITL formula over AP_ϵ) where $\psi'' = \mathbf{G}(\bigvee_{\pi \in \mathcal{Q}} \neg p^\epsilon_\pi) \wedge \big(\bigwedge_{\pi \in \mathcal{Q}} \mathbf{G}(p^\epsilon_\pi \Rightarrow \bigwedge_{p \in \mathsf{AP}} \neg p_\pi)\big) \wedge \psi'$ when $\mathcal{Q}$ are existential and $\psi'' = \mathbf{G}(\bigvee_{\pi \in \mathcal{Q}} \neg p^\epsilon_\pi) \wedge \big(\bigwedge_{\pi \in \mathcal{Q}} \mathbf{G}(p^\epsilon_\pi \Rightarrow \bigwedge_{p \in \mathsf{AP}} \neg p_\pi)\big) \Rightarrow \psi'$ when $\mathcal{Q}$ are universal. Intuitively, ψ'' ensures that the traces involved are *well-formed* (i.e. satisfy the first condition above). The following lemma follows from a simple structural induction.

Lemma 2. *For any trace set T over Σ_{AP} and closed alternation-free* HyperMITL *formula $\phi = \mathcal{Q}\psi$ over* AP *($\mathcal{Q}$ is either a block of existential quantifiers or universal quantifiers and ψ is quantifier free), $T \models \phi$ iff stutter$(T) \models^{sync}$ stutter(ϕ).*

Proof. Without loss of generality let $\mathcal{Q}$ be existential. If $\mathcal{Q}$ consists of a single quantifier $\exists\pi$ then clearly $T \models \phi \Leftrightarrow$ *stutter*$(T) \models^{sync}$ *stutter*(ϕ) ($\models$ and $\models^{sync}$ coincides in this case, and the quantifier-free part of *stutter*(ϕ) can only be satisfied if p^ϵ never holds in π). If $\mathcal{Q}$ consists of multiple quantifiers, then the quantifier-free part of ϕ is satisfied by Π (with *domain*$(\Pi) = \mathcal{Q}$) iff the quantifier-free part of *stutter*(ϕ) is satisfied by its stuttered counterpart Π' such that each trace in Π' is well-formed, Π' is synchronised (i.e. the timestamps of any two traces in Π' match exactly), and $(T, t) \models_{\Pi'} \bigvee_{\pi \in \mathcal{Q}} \neg p^\epsilon_\pi$ for any $t \in \mathbb{R}_{\geq 0}$ such that $(T, t) \models_{\Pi'} \top$. This implies *range*$(\Pi') \models^{sync}$ *stutter*(ϕ), and *stutter*$(T) \models^{sync}$ *stutter*(ϕ) since *range*$(\Pi') \subseteq$ *stutter*(T). The converse is similar. $\square$

The following two lemmas follow from Lemma 2 and the fact that for alternation-free HyperMITL formulae, satisfiability in the synchronous semantics can be reduced, in the same way as HyperLTL, to MITL satisfiability by introducing k copies of fresh atomic propositions (where k is the number of trace quantifiers).

Lemma 3. *The satisfiability problem for* $\exists^*$-HyperMITL *is decidable.*

Lemma 4. *The satisfiability problem for* $\forall^*$-HyperMITL *is decidable.*

Lemma 2, however, does not extend to larger fragments of HyperMITL. For example, consider the trace set T with two traces $\{(\{p\},1)(\{r\},3),(\{q\},2)\}$ and $\phi = \exists\pi_a\,\forall\pi_b\,(\mathbf{F}\,p_a \wedge \neg\,\mathbf{F}\,q_b)$. Now it is obvious that $T \not\models \phi$, but since $(\{p\},1)(\{r\},3) \in \mathit{stutter}(T)$ does not synchronise with any trace $\rho' \in \mathit{stutter}((q,2))$, we have $\mathit{stutter}(T) \models^{\mathit{sync}} \mathit{stutter}(\phi)$ (the definition of $\mathit{stutter}(\cdot)$ is extended to general HyperMITL formulae, as in Lemma 1). Still, it is not hard to see that the crucial observation used in $\exists^*\forall^*$-HyperLTL satisfiability (if $\exists\pi_0 \ldots \exists\pi_k\,\forall\pi'_0 \ldots \forall\pi'_\ell\,\psi$ is satisfiable, it must be satisfied by the trace set $\{\pi_0,\ldots\pi_k\}$ and thus is equisatisfiable with an $\exists^*$-HyperLTL formula) extends to HyperMITL in the asynchronous semantics; the following lemma then follows from Lemma 3.

Lemma 5. *The satisfiability problem for* $\exists^*\forall^*$-HyperMITL *is decidable.*

Finally, note that the undecidability of $\forall\exists$-HyperLTL carries over to HyperMITL: in the synchronous semantics, the reduction from Post's correspondence problem in [48] applies directly with some trivial modifications (as we work with finite traces); undecidability then holds for the case of asynchronous semantics as well, by Lemma 1.

Lemma 6. *The satisfiability problem for* $\forall\exists$-HyperMITL *is undecidable.*

Theorem 1. *The satisfiability problem for* HyperMITL *is decidable if the formula does not contain* $\forall\exists$.

4. Model checking

We now turn to the model-checking problem, which behaves quite differently than in the case of HyperLTL.

4.1. The alternation-free case

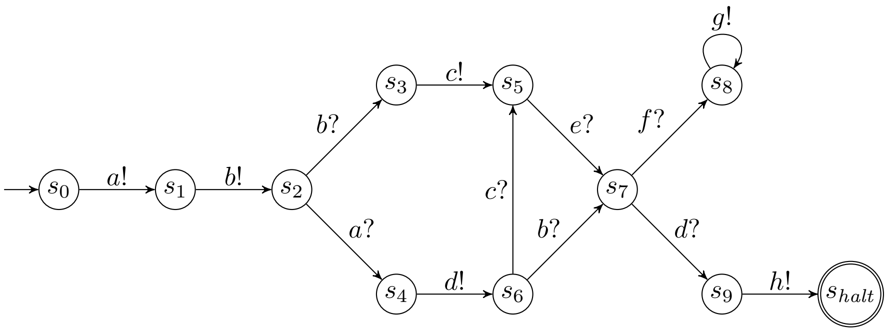
Without loss of generality, we consider only the case of $\exists^*$-HyperMITL in the asynchronous semantics. By Lemma 2, checking $[\![\mathcal{A}]\!] \models \phi$ (for a TA $\mathcal{A}$ over Σ_{AP} and a closed $\exists^*$-HyperMITL formula ϕ over AP) is equivalent to checking $\mathit{stutter}([\![\mathcal{A}]\!]) \models^{\mathit{sync}} \mathit{stutter}(\phi)$. To this end, we define $\mathit{stutter}(\mathcal{A})$ as the TA over $\Sigma_{\mathsf{AP}_\epsilon}$ obtained from $\mathcal{A}$ by adding a self-loop labelled with $\{p^\epsilon\}$ to each location; it should be clear that $[\![\mathit{stutter}(\mathcal{A})]\!] = \mathit{stutter}([\![\mathcal{A}]\!])$. In this way, the problem reduces to model checking $\exists^*$-HyperMITL in the synchronous semantics, which can be reduced to MITL model checking in the same way as model checking $\exists^*$-HyperLTL reduces to LTL model checking (by introducing fresh atomic propositions).

Theorem 2. *Model checking alternation-free* HyperMITL *is decidable.*

4.2. The general case

Recall that the model-checking problem for HyperLTL is decidable even when the specification involves nested quantifiers. This is unfortunately not the case for HyperMITL: allowing only one quantifier alternation already leads to undecidability. To see this, recall that any TA can be written as a formula $\exists X\,\psi$ where X is a set of (new) atomic propositions and ψ is an MITL formula [53, 54]. The undecidable TA universality problem—given a TA $\mathcal{A}$ over Σ, deciding whether $[\![\mathcal{A}]\!] = T\Sigma^*$—can thus be reduced to model checking HyperMITL: one simply checks whether there exists an X-labelling for every timed word over Σ so that ψ is satisfied. Here we show that model checking HyperMITL with quantifier alternations in the asynchronous semantics is essentially a harder problem: as we will see later, it necessarily involves TA_ϵ's and therefore remains undecidable even when *both the model and the specification are deterministic and only one of them uses a single clock (i.e. the other is untimed)*; by contrast, TA universality over finite timed words is decidable when the TA uses only one clock [44].

Instead of detailing the encoding based on TA (TA_ϵ) universality above, we adapt the undecidability proof of the *reactive synthesis* problem for MITL in [55], which itself is by reduction from the halting problem for



$$(s_0, \epsilon) \to (s_1, a) \to (s_2, ab) \to (s_4, b) \to (s_6, bd) \to (s_7, d) \to (s_9, \epsilon) \to (s_{halt}, h)$$

Figure 5: A DCM and its unique halting computation.

deterministic channel machines (DCMs), known to be undecidable [56]. Note that, in contrast to HyperMITL model checking, timed reactive synthesis is decidable when the specification is deterministic [57]; in this sense, quantification over traces is more powerful than quantification over strategies (there is a winning strategy of the controller for all possible strategies of the environment).[7] For our purpose, we introduce the $\lhd_I$ operator, in which we allow I to be singular (note that this is merely syntactic sugar and does not increase the expressiveness of MITL [53, 54]):

- $(T, t) \models_\Pi \lhd_I \phi$ iff there exists t', $0 \le t' < t$ such that $t - t' \in I$, $(T, t') \models_\Pi \top$, $(T, t') \models_\Pi \phi$, and $(T, t'') \not\models_\Pi \phi$ for all t'' such that $t'' \in (t', t)$ and $(T, t'') \models_\Pi \top$.

Intuitively, $\lhd_I \phi$ asserts that the time difference between now and the last time at which ϕ holds is in I. Let $\mathsf{LTL}_\lhd$ be the fragment of MITL where all timed subformulae must be of the form $\lhd_I \phi$, and all ϕ's in such subformulae must be 'pure past' formulae; these requirements ensure that $\mathsf{LTL}_\lhd$, in which we will write the quantifier-free part of the specification, translates into deterministic TAs [58]. We will first do the proof for the case of asynchronous semantics and then adapt it to the case of synchronous semantics.

Theorem 3. *Model checking $\exists^*\forall^*$-HyperMITL and $\forall^*\exists^*$-HyperMITL are undecidable in the asynchronous semantics.*

Proof. A DCM $\mathcal{S} = \langle S, s_0, s_{halt}, M, \Delta \rangle$ can be seen as a finite automaton equipped with an unbounded fifo channel: S is a finite set of locations, s_0 is the initial location, s_{halt} is the halting location (such that $s_{halt} \neq s_0$), M is a finite set of messages, and $\Delta \subseteq S \times \{m!, m? \mid m \in M\} \times S$ is the transition relation satisfying the following determinism hypotheses: (i) $(s, q, s') \in \Delta$ and $(s, q, s'') \in \Delta$ implies $s' = s''$; (ii) if $(s, m!, s') \in \Delta$ then it is the only outgoing transition from s. Without loss of generality, we further assume that there is no incoming transition to s_0, no outgoing transition from s_{halt}, and $(s_0, q, s') \in \Delta$ implies that $q \in \{m! \mid m \in M\}$ and $s' \neq s_{halt}$. The semantics of $\mathcal{S}$ can be described with a graph $G(\mathcal{S})$ with vertices $\{(s, x) \mid s \in S, x \in M^*\}$ and edges defined as follows: (i) $(s, x) \to (s', xm)$ if $(s, m!, s') \in \Delta$; (ii) $(s, mx) \to (s', x)$ if $(s, m?, s') \in \Delta$. In other words, $m!$ 'writes' a copy of m to the tail of the channel and $m?$ 'reads' a copy of m off the head of the channel. We say that $\mathcal{S}$ *halts* if there is a path in $G(\mathcal{S})$ from (s_0, ϵ) to (s_{halt}, x) (a *halting computation* of $\mathcal{S}$) for some $x \in M^*$. An example DCM and its unique halting computation are depicted in Figure 5.

The idea, as in many similar proofs (e.g., [46]), is to encode a halting computation of $\mathcal{S}$ as a trace where each $m?$ is preceded by a corresponding $m!$ exactly 1 time unit earlier, and each $m!$ is followed by an $m?$

[7] Indeed, the quantifier-free part ψ in the encoding mentioned above (based on labelling timed words with propositions in X) is already in $\mathsf{LTL}_\lhd$ and thus is deterministic.

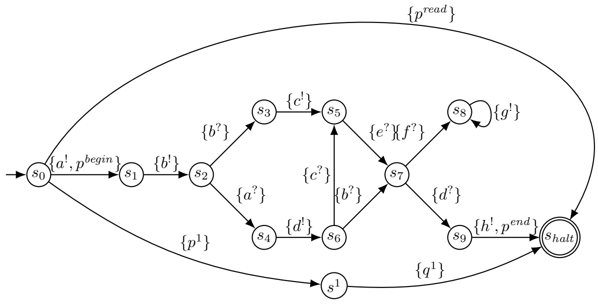


Figure 6: The model $\mathcal{A}$ from the DCM in Figure 5.

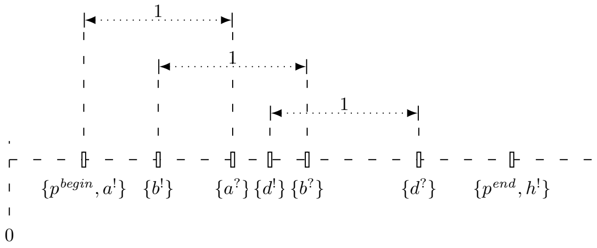


Figure 7: A trace that encodes the halting computation of the DCM in Figure 5. Note that each $m^!$ is followed by a corresponding $m^?$ exactly 1 time unit later.

exactly 1 time unit later if s_{halt} has not been reached yet (in other words, the channel can be non-empty when s_{halt} is reached). To this end, let the model $\mathcal{A}$ be an (untimed) finite automaton over $\Sigma = 2^{\mathsf{AP}}$ where $\mathsf{AP} = \{m^!, m^? \mid m \in M\} \cup \{p^{begin}, p^{end}, p^{read}, p^1, q^1\}$ and whose set of locations is $S \cup \{s_1\}$, where s_1 is a new non-accepting location. The transitions of $\mathcal{A}$ follow $\mathcal{S}$: for each $m \in M$, $s \xrightarrow{\{m^?\}} s'$ is a transition of $\mathcal{A}$ iff $(s, m?, s') \in \Delta$, and similarly for $m!$—except for those going out of s_0 or going into s_{halt}, on which we further require p^{begin} or p^{end} to hold, respectively. Let s_0 be the initial location and s_{halt} be the only accepting location, and finally add transitions $s_0 \xrightarrow{\{p^{read}\}} s_{halt}$ and $s_0 \xrightarrow{\{p^1\}} s_1 \xrightarrow{\{q^1\}} s_{halt}$. It is clear that $\mathcal{A}$ is deterministic and it accepts only three types of traces (see Figure 6):

1. From s_0 through some other locations of $\mathcal{S}$ and finally s_{halt}, i.e. those respecting the transition relation, but not necessarily the semantics, of $\mathcal{S}$.
2. From s_0 to s_{halt} in a single transition (on which p^{read} holds).
3. From s_0 to s_1 and then s_{halt}.

It remains to write a specification ϕ such that $[\![\mathcal{A}]\!] \models \phi$ exactly when $\mathcal{A}$ accepts a trace of type (1) that also respects the semantics of $\mathcal{S}$ (one such trace that corresponds to the unique halting computation of the DCM in Figure 5 is depicted in Figure 7). This is where the traces of types (2) and (3) come into play: for example, if a trace of type (1) issues a read $m?$ without a corresponding write $m!$, then a trace of type (3) can be used to 'pinpoint' the error. More precisely, let $\phi = \exists \pi_a \, \forall \pi_b \, (\psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4)$ where

- $\psi_1 = \mathbf{F}\, p_a^{end}$ ensures that π_a is of type (1);
- $\psi_2 = \mathbf{F}(p_b^{read} \wedge \psi_R) \Rightarrow \mathbf{F}(p_b^{read} \wedge \lhd_{\geq 1} p_a^{begin})$, where $\psi_R = \bigvee\{m_a^? \mid m \in M\}$, is a simple sanity check which ensures that in π_a, each $m^?$ must happen at time $\geq t+1$ if p^{begin} happens at t;
- $\psi_3 = \bigwedge_{m \in M} \Big(\mathbf{F}(q_b^1 \wedge m_a^?) \Rightarrow \big(\mathbf{F}(p_a^{begin} \wedge \mathbf{F}\, p_b^1) \wedge \mathbf{F}(q_b^1 \wedge \lhd_{=1} p_b^1) \Rightarrow \mathbf{F}(p_b^1 \wedge m_a^!) \big) \Big)$ ensures that each $m^?$, if it happens at t, is preceded by a corresponding $m^!$ at $t-1$ in π_a;
- $\psi_4 = \bigwedge_{m \in M} \Big(\mathbf{F}(p_b^1 \wedge m_a^!) \Rightarrow \mathbf{F}(p_a^{end} \wedge \lhd_{<1} p_b^1) \vee \big(\mathbf{F}(q_b^1 \wedge \lhd_{=1} p_b^1) \Rightarrow \mathbf{F}(q_b^1 \wedge m_a^?) \big) \Big)$ ensures that each $m^!$ at t is followed by a corresponding $m^?$ at $t+1$ (unless p^{end} happens first) in π_a.

Now we claim that $\psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4$ can be replaced by a formula equivalent to a one-clock deterministic TA. Observe that the only timed subformulae are $\lhd_{\geq 1} p_a^{begin}$, $\lhd_{=1} p_b^1$, and $\lhd_{<1} p_b^1$. As p^1 and p^{read} cannot happen in the same trace (π_b), it is not hard to see that the reduction remains correct if we replace $\lhd_{\geq 1} p_a^{begin}$ by $\lhd_{\geq 1}(p_a^{begin} \vee p_b^1)$ in ψ_2. Similarly, if we already know that p_b^1 happens after p_a^{begin}, we can replace $\lhd_{=1} p_b^1$ and $\lhd_{<1} p_b^1$ by $\lhd_{=1}(p_a^{begin} \vee p_b^1)$ and $\lhd_{<1}(p_a^{begin} \vee p_b^1)$ (respectively) in ψ_3 and ψ_4 to obtain ψ_3' and ψ_4'. The resulting formula $\psi_1 \wedge \psi_2' \wedge \psi_3' \wedge \psi_4'$ can readily be translated into a one-clock deterministic TA.

We finish this proof with two observations:

1. As each of p^1, q^1, and p^{read} can appear at most once in a single trace, the past subformulae can be replaced by future ones (the resulting formula can no longer be translated into a deterministic TA, however). For example, we may write $\mathbf{F}(p_b^1 \wedge \mathbf{F}_{\leq 1}\, q_b^1 \wedge \mathbf{F}_{\geq 1}\, q_b^1)$ instead of $\mathbf{F}(q_b^1 \wedge \triangleleft_{=1}\, p_b^1)$.
2. It is possible to move all the timing constraints into the model and use an untimed HyperLTL formula as the specification: in the model, ensure that p^1 and q^1 are separated by exactly 1 time unit, and add $s_0 \xrightarrow{\{p^2\}} s_1 \xrightarrow{\{q^2\}} s_{halt}$ such that p^2 and q^2 are separated by < 1 time unit; in the specification, use p^2, q^2 to rule out those π_a's with some $m^?$ at < 1 time unit from p^{begin}. □

Now we consider the synchronous semantics. The corresponding result is weaker in this case, as we will see in the next section that in several subcases the problem becomes decidable. Still, the reduction above can be made to work if the model has one clock and an extra trace quantifier is allowed.

Theorem 4. *Model checking* $\exists^*\forall^*$-HyperMITL *and* $\forall^*\exists^*$-HyperMITL *are undecidable in the synchronous semantics.*

Proof. We use a modified model $\mathcal{A}'$ whose set of locations is $S \cup \{s_1, s_2, s_3, s_4\}$; the transitions are similar to $\mathcal{A}$ in the proof of Theorem 3, but we now use a clock x in the path $s_0 \xrightarrow[x:=0]{\{p^1\}} s_1 \xrightarrow[x\geq 1, x:=0]{\{q^1\}} s_{halt}$, the paths $s_0 \xrightarrow[x:=0]{\{p^2\}} s_2 \xrightarrow[x\leq 1, x:=0]{\{q^2\}} s_{halt}$, $s_0 \xrightarrow[x:=0]{\{p^3\}} s_3 \xrightarrow[x>1, x:=0]{\{q^3\}} s_{halt}$, $s_0 \xrightarrow[x:=0]{\{p^4\}} s_4 \xrightarrow[x<1, x:=0]{\{q^4\}} s_{halt}$ are added, and $s_0 \xrightarrow{\{p^{read}\}} s_{halt}$ is removed. Moreover, a self-loop labelled with $\{p^\epsilon\}$ (and resets no clock) is added to each of s_0, s_1, s_2, s_3, s_4, and s_{halt}. The specification is $\phi' = \exists \pi_a \, \forall \pi_b \, \forall \pi_c \bigwedge_{1 \leq i \leq 9} \psi_i'$ where $\bigwedge_{1 \leq i \leq 9} \psi_i'$ is the following untimed LTL formula:

- $\psi_1' = \mathbf{F}\, p_a^{end}$;
- $\psi_2' = \mathbf{F}(q_b^4 \wedge \psi_R) \Rightarrow \neg\, \mathbf{F}(p_b^4 \wedge p_a^{begin})$ where $\psi_R = \bigvee\{m_a^? \mid m \in M\}$;
- $\psi_3' = \bigwedge_{m \in M} \big(\mathbf{F}(q_b^1 \wedge q_c^2 \wedge m_a^?) \wedge \mathbf{F}(p_b^1 \wedge p_c^2) \Rightarrow \mathbf{F}(p_b^1 \wedge p_c^2 \wedge m_a^!)\big)$;
- $\psi_4' = \mathbf{F}(q_b^3 \wedge \psi_R) \Rightarrow \neg\, \mathbf{F}(p_b^3 \wedge \mathbf{X}\, q_b^3)$;
- $\psi_5' = \mathbf{F}(q_b^3 \wedge q_c^4 \wedge \psi_R) \Rightarrow \neg\, \mathbf{F}(p_b^3 \wedge \mathbf{X}\, p_c^4)$;
- $\psi_6' = \mathbf{F}(p_b^4 \wedge \psi_W) \Rightarrow \neg\, \mathbf{F}(q_b^4 \wedge \neg\, \mathbf{X} \top)$ where $\psi_W = \bigvee\{m_a^! \mid m \in M\}$;
- $\psi_7' = \bigwedge_{m \in M} \Big(\mathbf{F}(p_b^1 \wedge p_c^2 \wedge m_a^!) \wedge \mathbf{F}(q_b^1 \wedge q_c^2) \Rightarrow \mathbf{F}\, \big(q_b^1 \wedge q_c^2 \wedge (m_a^? \vee p_a^\epsilon)\big)\Big)$;
- $\psi_8' = \mathbf{F}(p_b^3 \wedge \psi_W) \Rightarrow \neg\, \mathbf{F}(p_b^3 \wedge \mathbf{X}\, q_b^3)$;
- $\psi_9' = \mathbf{F}(p_b^3 \wedge p_c^4 \wedge \psi_W) \Rightarrow \neg\, \mathbf{F}(q_b^3 \wedge \mathbf{X}\, q_c^4)$.

We now claim that $[\![\mathcal{A}']\!] \models \phi'$ iff $\mathcal{A}'$ accepts a trace that encodes a halting computation of $\mathcal{S}$. Intuitively, ψ_1', ψ_2' play similar roles as ψ_1, ψ_2 in the proof of Theorem 3; ψ_3' ensures that if each $m^?$ at t is preceded by an event at $t-1$ then $m^!$ must hold there, and ψ_4', ψ_5' ensures that each $m^?$ at t is actually preceded by an event at $t-1$. The roles of ψ_6', ψ_7', ψ_8', and ψ_9' are analogous.

- ($\Leftarrow$): Let π_a be the trace that encodes a halting computation of $\mathcal{S}$ in the same way as before, with the exception that it now contains a $\{p_a^\epsilon\}$-event at $t+1$ for each $m_a^!$-event at t that is not read as $\mathcal{A}'$ reaches s_{halt}; in other words, there may be a number of $\{p_a^\epsilon\}$-events after where p_a^{end} holds. Apparently ψ_1' is satisfied, and it is easy to verify that each of ψ_2' to ψ_9' will be satisfied for any π_b and π_c.

- ($\Rightarrow$): As ψ_1' is satisfied we know that π_a must follow the transitions of $\mathcal{S}$ (potentially with some $\{p_a^\epsilon\}$-events before p_a^{begin} and after p_a^{end}). If it is not in the required form (respect the semantics of $\mathcal{S}$ and has some $\{p_a^\epsilon\}$-events matching the leftover $m_a^!$'s), then one of the following must be true:

- Some $m_a^?$ happens before $t+1$ where p_a^{begin} happens at t in π_a: there is a trace π_b where p_b^4 and q_b^4 align with p_a^{begin} and the $m_a^?$ in question. This implies that ψ_2' does not hold.
- Some $m_a^?$ happens at $t+1$ and there is an event at t in π_a, but $m_a^!$ does not hold there: there are traces π_b, π_c where p_b^1 and p_c^2 happen at t and q_b^1 and q_c^2 happen at $t+1$. This implies that ψ_3' does not hold.
- Some $m_a^?$ happens at $t+1$ and there is no event in $[t, t+1)$ in π_a: there is a trace π_b where q_b^3 and p_b^3 align with that $m_a^?$ and the event just before it, so ψ_4' does not hold.
- Some $m_a^?$ happens at $t+1$ and there is no event at t, but there are events in $[0, t)$ and $(t, t+1)$ in π_a: there are traces π_b, π_c where q_b^3 and q_c^4 both happen at $t+1$, p_b^3 aligns with the last event in $[0, t)$, and p_c^4 aligns with the first event in $(t, t+1)$. This implies that ψ_5' does not hold.
- Some $m_a^!$ happens at t and there is no event in $[t+1, \infty)$ in π_a: there is a trace π_b where p_b^4 and q_b^4 align with that $m_a^!$ and the last event in π_a, thus ψ_6' does not hold.
- Some $m_a^!$ happens at t and there is an event at $t+1$ in π_a, but $m_a^!$ or p_a^ϵ do not hold there: ψ_7' does not hold.
- Some $m_a^!$ happens at t and there is no event in $(t, t+1]$ in π_a: ψ_8' does not hold.
- Some $m_a^!$ happens at t and there is no event at t, but there are events in $(t, t+1)$ and $(t+1, \infty)$ in π_a: ψ_9' does not hold.

So, π_a is a valid encoding of a halting computation of $\mathcal{S}$. □

4.3. Restricted models

We conclude this section by showing that the undecidability results above can actually be obtained for trivial systems with only a single location. In particular, the structural restrictions considered in [32] have no effect on the decidability of HyperMITL model checking. To see this, note that we can introduce a fresh atomic proposition p^s for each location s such that p^s holds whenever the transition taken enters s; the transition relation can then be enforced with a simple LTL formula.

Corollary 1. *Model checking* $\exists^*\forall^*$-HyperMITL *and* $\forall^*\exists^*$-HyperMITL *are undecidable in the asynchronous semantics for systems with only one location.*

Corollary 2. *Model checking* $\exists^*\forall^*$-HyperMITL *and* $\forall^*\exists^*$-HyperMITL *are undecidable in the synchronous semantics for systems with only one location.*

5. Decidable subcases

While the negative results in the previous section may be disappointing, we stress again that model checking alternation-free HyperMITL is no harder than MITL model checking, and it can in fact be carried out with algorithms and tools for the latter. In any case, we now identify several subcases where model checking is decidable beyond the alternation-free fragment.

5.1. Untimed model + untimed specification

The first case we consider is when both the model and the specification are untimed, and the asynchronous semantics is assumed. Our algorithm follows the lines of [10] and is essentially based on *self-composition* (cf. [59], and many others; see the references in [10]) of the model; the difficulty here, however, is to handle interleaving of events. Let the model $\mathcal{A}$ be a finite automaton over Σ_{AP} and the specification be a (untimed) closed HyperMITL formula over AP. Without loss of generality, we assume the specification to be $\phi = \exists\pi_1\,\forall\pi_2\,\ldots\exists\pi_{k-1}\,\forall\pi_k\,\psi$, which can be rewritten into $\exists\pi_1\,\neg\exists\pi_2\,\neg\ldots\exists\pi_{k-1}\,\neg\exists\pi_k\,\neg\psi$. We start by translating *stutter*$(\neg\psi)$ (in which we replace all occurrences of $\top_i$ with $\neg p_i^\epsilon$, i.e. regarded here simply as an MITL formula over $(\mathsf{AP}_\epsilon)^k = \{p_i \mid p \in \mathsf{AP}_\epsilon, 1 \le i \le k\}$) into the equivalent finite automaton over $\Sigma_{(\mathsf{AP}_\epsilon)^k}$, and take its product with (i) the automaton for $\mathbf{G}(\bigvee_{1\le i\le k} \neg p_i^\epsilon) \wedge \big(\bigwedge_{1\le i\le k} \mathbf{G}(p_i^\epsilon \Rightarrow \bigwedge_{p\in\mathsf{AP}} \neg p_i)\big)$ and (ii) the automaton obtained from $\mathit{stutter}(\mathcal{A})$ by extending the alphabet to $\Sigma_{(\mathsf{AP}_\epsilon)^k}$ and renaming all the occurrences of p to p_k, to obtain $\mathcal{B}$. Now let $\mathcal{C}$ be the projection of $\mathcal{B}$ onto $(\mathsf{AP}_\epsilon)^{k-1} = \{p_i \mid p \in \mathsf{AP}_\epsilon, 1 \leq i \leq k-1\}$ (this step corresponds to $\exists$ in $\neg\exists\pi_k$). By construction, $\mathcal{B}$ accepts traces that are well-formed in dimensions 1 to $k-1$, and so does $\mathcal{C}$; but $\mathcal{C}$ may accept traces containing $\{p_i^\epsilon \mid 1 \leq i \leq k-1\}$-events. We replace these events by 'real' ϵ's, which are then removed to obtain $\mathcal{C}'$. Finally, we complement $\mathcal{C}'$ to obtain $\mathcal{C}''$ (this step corresponds to $\neg$ in $\neg\exists\pi_k$). We can then start over by taking the product of $\mathcal{C}''$, the automaton for $\mathbf{G}(\bigvee_{1\leq i\leq k-1} \neg p_i^\epsilon) \wedge \big(\bigwedge_{1\leq i\leq k-1} \mathbf{G}(p_i^\epsilon \Rightarrow \bigwedge_{p\in\mathsf{AP}} \neg p_i)\big)$, and the automaton obtained from $\mathit{stutter}(\mathcal{A})$ by extending the alphabet to $\Sigma_{(\mathsf{AP}_\epsilon)^{k-1}}$ and renaming all the occurrences of p to p_{k-1}; the resulting automaton is the new $\mathcal{B}$. We continue this process until the outermost quantifier $\exists\pi_1$ is reached, when we test the emptiness of $\mathcal{B}$ (at this point, it is an automaton over $\Sigma_{\mathsf{AP}_\epsilon}$).

Proposition 1. *Model checking* HyperMITL *is decidable when the model and the specification are both untimed.*

5.2. One clock + one alternation

The model-checking algorithm outlined in the previous case crucially depends on the fact that both $\mathcal{A}$ and ϕ are untimed, hence their product (in the sense detailed in the previous case) can be complemented. When the synchronous semantics is assumed and there is only one quantifier alternation in ϕ, it might be the case that we do not actually need complementation. For example, if $\mathcal{A}$ is untimed and $\phi = \forall\pi_a\, \exists\pi_b\, \exists\pi_c\, \psi$ where ψ translates into a one-clock TA, the corresponding model-checking problem clearly reduces to universality for one-clock TAs, which is decidable but non-primitive recursive [60].[8] This observation applies to other cases as well, such as when $\mathcal{A}$ is a one-clock TA and $\phi = \exists\pi_a\, \forall\pi_b\, \psi$ where ψ is untimed; here model checking reduces to language inclusion between two one-clock TAs.

5.3. Untimed model + MIA specification

The main obstacle in applying the algorithm above to larger fragments of HyperMITL, as should be clear now, is that universal quantifiers amount to complementations, which are not possible in general in the case of TAs. Moreover, we note that the usual strategy of restricting to deterministic models and specifications does not help, as the projection steps (which corresponds to trace quantifiers) in the algorithm necessarily introduces non-determinism. To make the algorithm work for larger fragments, we essentially need a class of automata that is both *closed under projection* and *complementable*. Fortunately, there is a subclass of one-clock TAs that satisfies these conditions. We consider two additional restrictions on one-clock TAs:

- Non-Singular (NS): a one-clock TA is NS if all the guards are non-singular (i.e. must be of the form $x \in I$ where x is the single clock and I is a non-singular interval).
- Reset-on-Testing (RoT): a one-clock TA is RoT if whenever the guard of a transition is not $\top$, x must be reset on that transition.

One-clock TAs satisfying both NS and RoT are called *metric interval automata* (MIAs), which are determinisable [61]. Since the projection operation cannot invalidate NS and RoT, the algorithm above can be applied when the synchronous semantics is assumed, $\mathcal{A}$ is untimed, ψ or $\neg\psi$ translates to a MIA, and only one complementation is involved; in this case it runs in elementary time.

Proposition 2. *Model checking* $\forall^*\exists^*$-HyperMITL *(*$\exists^*\forall^*$-HyperMITL*) is decidable in the synchronous semantics when the model is untimed and ψ ($\neg\psi$) translates into a* MIA *in the specification $\phi = \forall\pi_1 \ldots \exists\pi_k\, \psi$ ($\phi = \exists\pi_1 \ldots \forall\pi_k\, \psi$).*

On the other hand, we can adapt the proof of Theorem 4 to show that model checking an untimed model against an $\exists^*\forall^*$-HyperMITL specification ϕ in the synchronous semantics, when the quantifier-free part ψ (instead of $\neg\psi$) translates into a MIA, remains undecidable.

[8] This case is undecidable in the asynchronous semantics by Theorem 3; as explained above, the algorithm may introduce ϵ-transitions in the asynchronous semantics, while universality for one-clock TA_ϵ's is undecidable [60].

Proposition 3. *Model checking $\exists^*\forall^*$-$\mathsf{HyperMITL}$ is undecidable in the synchronous semantics when the model is untimed and ψ in the specification $\phi = \exists\pi_1 \ldots \forall\pi_k\, \psi$ translates into a* MIA.

Proof. We use $\mathcal{A}''$, obtained by dropping all the timing constraints in $\mathcal{A}'$ in the proof of Theorem 4 and adding two paths $s_0 \xrightarrow{\{p^{3'}\}} s_3' \xrightarrow{\{q^{3'}\}} s_{halt}$ and $s_0 \xrightarrow{\{p^{4'}\}} s_4' \xrightarrow{\{q^{4'}\}} s_{halt}$, as the model. Then, we replace the formula $\bigwedge_{1 \leq i \leq 9} \psi_i'$ by a MIA—as only one clock is allowed, each condition that involves timing must be enforced by a separate 'configuration' of π_b and π_c. For the following construction to work, we additionally require that in π_a (which encodes a halting computation of $\mathcal{S}$), if two events are separated by exactly 1 time unit then they must be a valid write-read pair.

- $\psi_1' = \mathbf{F}\, p_a^{end}$: this can be enforced by an untimed automaton $\mathcal{A}_1$.
- $\psi_2' = \mathbf{F}(q_b^4 \wedge \psi_R) \Rightarrow \neg\,\mathbf{F}(p_b^4 \wedge p_a^{begin})$ where $\psi_R = \bigvee\{m_a^? \mid m \in M\}$: We use $\mathcal{A}_2'$ to enforce that π_b, π_c both go through s_4. Then, let $\mathcal{A}_2$ be the product of $\mathcal{A}_2'$ and the union of the equivalent automata for the following:
  - It is not the case that q_b^4, q_c^4, and ψ_R hold simultaneously at some event.
  - It is not the case that p_b^4, p_c^4, and p_a^{begin} hold simultaneously at some event.
  - p_b^4 and q_b^4 are separated by ≥ 1 time unit.
  - p_c^4 and q_c^4 are separated by ≥ 1 time unit.
- $\psi_3' = \bigwedge_{m \in M} \big(\mathbf{F}(q_b^1 \wedge q_c^2 \wedge m_a^?) \wedge \mathbf{F}(p_b^1 \wedge p_c^2) \Rightarrow \mathbf{F}(p_b^1 \wedge p_c^2 \wedge m_a^!)\big)$ and $\psi_7' = \bigwedge_{m \in M} \Big(\mathbf{F}(p_b^1 \wedge p_c^2 \wedge m_a^!) \wedge \mathbf{F}(q_b^1 \wedge q_c^2) \Rightarrow \mathbf{F}\left(q_b^1 \wedge q_c^2 \wedge (m_a^? \vee p_a^\epsilon)\right)\Big)$: similarly, we use $\mathcal{A}_3'$ to enforce that π_b goes through s_1 and π_c goes through s_2. Let $\mathcal{A}_3$ be the product of $\mathcal{A}_3'$ and the union of the equivalent automata for the following:
  - For the same $m \in M$, p_b^1, p_c^2, and $m_a^!$ hold simultaneously at some event, and q_b^1, q_c^2, and $m_a^? \vee p_a^\epsilon$ hold simultaneously at some event.
  - p_b^1 and q_b^1 are separated by < 1 time unit.
  - p_c^2 and q_c^2 are separated by > 1 time unit.
- $\psi_4' = \mathbf{F}(q_b^3 \wedge \psi_R) \Rightarrow \neg\,\mathbf{F}(p_b^3 \wedge \mathbf{X}\, q_b^3)$: we use $\mathcal{A}_4'$ to enforce that π_b, π_c both go through s_3. Let $\mathcal{A}_4$ be the product of $\mathcal{A}_4'$ and the union of the equivalent automata for the following:
  - It is not the case that q_b^3, q_c^3, and ψ_R hold simultaneously at some event.
  - It is not the case that p_b^3 and p_c^3 hold simultaneously at some event.
  - It is not the case that p_b^3, p_c^3 hold simultaneously at some event and q_b^3, q_c^3 hold simultaneously at the next event.
  - p_b^3 and q_b^3 are separated by ≤ 1 time unit.
  - p_c^3 and q_c^3 are separated by ≤ 1 time unit.
- $\psi_5' = \mathbf{F}(q_b^3 \wedge q_c^4 \wedge \psi_R) \Rightarrow \neg\,\mathbf{F}(p_b^3 \wedge \mathbf{X}\, p_c^4)$: we use $\mathcal{A}_5'$ to enforce that π_b goes through s_3 and π_c goes through s_4. Let $\mathcal{A}_5$ be the product of $\mathcal{A}_5'$ and the union of the equivalent automata for the following:
  - It is not the case that q_b^3, q_4^3, and ψ_R hold simultaneously at some event.
  - It is not the case that p_b^3 holds at some event and p_c^4 holds at the next event.
  - p_b^3 and q_b^3 are separated by ≤ 1 time unit.
  - p_c^4 and q_c^4 are separated by ≥ 1 time unit.

Similarly we have $\mathcal{A}_6$, $\mathcal{A}_8$, and $\mathcal{A}_9$ (using the paths going through s_3' and s_4'). The specification is the product of $\mathcal{A}_1$ and the union of $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_4$, $\mathcal{A}_5$, $\mathcal{A}_6$, $\mathcal{A}_8$, $\mathcal{A}_9$, $\mathcal{A}_{rest}$ where $\mathcal{A}_{rest}$ accepts the remaining configurations of π_b and π_c. It is clear that the resulting automaton can be modified to use only one clock. □

Spec. Model	untimed	NS+RoT	NS	RoT
untimed	Dec. (Proposition 1)	Dec. for $\forall^*\exists^*$ (Proposition 2)	Undec. for $\exists\forall\forall$ (Proposition 3)	Undec. for $\exists\forall\forall$ (Proposition 3)
NS+RoT	Undec. for $\exists\forall\forall$ (Theorem 4)	Undec.	Undec.	Undec.
NS	Undec.	Undec.	Undec.	Undec.
RoT	Undec.	Undec.	Undec.	Undec.

Table 1: Decidability of model checking untimed or one-clock TAs against (one-clock) HyperMITL in the synchronous semantics; NS stands for Non-Singular constraints and RoT stands for Reset-on-Testing.

5.4. Bounded time domains

We now show that when there is an *a priori* bound N (where N is a positive integer) on the length of the time domain, the model-checking problem for full HyperMITL becomes decidable; in fact, in the case of synchronous semantics it reduces to the satisfiability problem for QPTL [2]. From a practical point of view, this implies that *time-bounded* HyperMITL verification (at least for the $\exists^*\forall^*$-fragment, say) can be carried out with highly efficient, off-the-shelf tools that work with LTL and (untimed) automata, such as SPOT [62], GOAL [63], and `Owl` [64].

We assume the asynchronous semantics. For a given N, we consider all traces in which all timestamps are less than N. Denote by $[\![\mathcal{A}]\!]_{[0,N)}$ the set of all such traces in $[\![\mathcal{A}]\!]$; the model-checking problem then becomes deciding whether $[\![\mathcal{A}]\!]_{[0,N)} \models \phi$. As before, we assume ϕ to be $\exists\pi_1 \neg\exists\pi_2 \neg \ldots \exists\pi_{k-1} \neg\exists\pi_k \neg\psi$. Following [65, 66], we can use the *stacking construction* to obtain, from the conjunction ψ' of *stutter*$(\neg\psi)$ and $\mathbf{G}(\bigvee_{\pi\in\mathcal{Q}} \neg p^\epsilon_\pi) \wedge \big(\bigwedge_{\pi\in\mathcal{Q}} \mathbf{G}(p^\epsilon_\pi \Rightarrow \bigwedge_{p\in\mathsf{AP}} \neg p_\pi)\big)$, an equi-satisfiable untimed (QPTL [2]) formula $\overline{\phi} = \exists W\, \overline{\psi'}$ over the stacked alphabet $\overline{(\mathsf{AP}_\epsilon)^k} \cup \overline{Q}$ (where $\overline{(\mathsf{AP}_\epsilon)^k} = \{p_{i,j} \mid p \in \mathsf{AP}_\epsilon,\, 1 \le i \le k,\, 0 \le j < N\}$ and $\overline{Q} = \{q_j \mid 0 \le j < N\}$). We apply the following modifications to $\overline{\phi}$ to obtain $\overline{\phi'}$:

- Introduce atomic propositions $\{p^\epsilon_i \mid 1 \le i \le k\}$ and add the conjunct
$$\wedge_{1\le i\le k}\, \mathbf{G}\Big(\big(\wedge_{0\le j<N}\, (q_j \Rightarrow p^\epsilon_{i,j})\big) \Leftrightarrow p^\epsilon_i\Big)\,;$$
- Introduce atomic propositions $\{q_{i,j} \mid 1 \le i \le k,\, 0 \le j < N\}$ and add the conjunct
$$\wedge_{1\le i\le k}\, \mathbf{G}\,\big(\wedge_{0\le j<N}\, (\neg p^\epsilon_i \wedge q_j \Leftrightarrow q_{i,j})\big)\,;$$
- Project away $\{p^\epsilon_{i,j} \mid 1 \le i \le k,\, 0 \le j < N\}$ and $\overline{Q}$;
- Replace all occurrences of p^ϵ_i by $\bot_i$.

Now, as we mentioned earlier, we can write $\mathcal{A}$ as an $\mathcal{L}^{\leftrightarrow}_d$ formula $\phi_\mathcal{A} = \exists X_\mathcal{A}\, \psi_\mathcal{A}$ where $X_\mathcal{A}$ is a set of atomic propositions such that $\mathsf{AP} \cap X_\mathcal{A} = \emptyset$ and $\psi_\mathcal{A}$ is an MITL formula over $\mathsf{AP} \cup X_\mathcal{A}$. Let $\overline{\phi_\mathcal{A}}$ be its stacked counterpart $\exists\overline{X_\mathcal{A}}\, \exists Y\, \overline{\psi_\mathcal{A}}$; we translate $\overline{\phi_\mathcal{A}}$ back into an untimed automaton $\overline{\mathcal{A}}$ over the stacked alphabet $\overline{\mathsf{AP}} \cup \overline{Q}$. The problem thus reduces to untimed model checking of $\overline{\mathcal{A}}$ against $\exists\pi_1\, \forall\pi_2 \ldots \exists\pi_{k-1}\, \forall\pi_k\, \overline{\phi'}$ in the asynchronous semantics, which is decidable by Proposition 1 ($\overline{\phi'}$ has outermost existential propositional quantifiers, but clearly the equivalent automaton for *stutter*$(\overline{\phi'})$ can be used directly in the algorithm).

Finally, note that the proof is simpler for the case of synchronous semantics: we can simply work with a (non-stuttering) $\mathcal{L}^{\leftrightarrow}_d$ formula in all the intermediate steps without translating it into an automaton, and then check the satisfiability of the final formula by stacking it into a QPTL formula.

Proposition 4. *Model checking* HyperMITL *is decidable when the time domain is* $[0, N)$*, where* N *is a given positive integer.*

5.5. Bounded variability

We end this section by showing that HyperMITL model checking also becomes decidable if there is an *a priori* bound on the *variabiity* of traces. The *variability* of a timed word is the maximum possible number of events in any open unit interval; a timed word is *of bounded variability* k_{var} if its variability is less or equal than k_{var} (where k_{var} is a positive integer). For our purpose, we give an alternative, 'asynchronous' semantics for $\mathcal{L}_d^{\leftrightarrow}$, and show that $\mathcal{L}_d^{\leftrightarrow}$ in this semantics captures exactly the class of timed languages accepted by TA_ϵ's.

Recall from [25] that the set of *monadic logic of distance* ($\mathcal{L}_d$) formulae over AP are generated by

$$\phi := Q_p x \mid Q_{act} x \mid x < x' \mid Xx \mid d(x, x') \sim c \mid \phi_1 \wedge \phi_2 \mid \neg\psi \mid \exists x\, \phi \mid \exists X\, \phi$$

where $p \in \mathsf{AP}$, $i \in \{0, \dots, m-1\}$, $\sim \in \{=, \neq, <, >, \leq, \geq\}$, and $c \in \mathbb{N}_{\geq 0}$. The *relative distance formula* $\overleftarrow{d}(X, x) \sim c$ is defined by

$$\overleftarrow{d}(X, x) \sim c \equiv \exists x' \left(x' < x \wedge Xx' \wedge \neg\exists x'' \left(x' < x'' < x \wedge Xx''\right) \wedge d(x, x') \sim c\right).$$

Intuitively, this says that looking back from x, the last event where X holds is $\sim c$ away; similarly we define $\overrightarrow{d}(X, x) \sim c$. The set of *monadic logic of relative distance* ($\mathcal{L}_d^{\leftrightarrow}$) formulae over AP are $\mathcal{L}_d$ formulae of the form $\exists X_1 \dots \exists X_m\, \psi$ where ψ is built from $Q_p x$, Q_{act}, $x < x'$, $X_i x$, $\overleftarrow{d}(X_i, x) \sim c$, $\overrightarrow{d}(X_i, x) \sim c$ using Boolean operators, first-order quantifiers, and second-order quantifiers for set variables other than $X_1, \dots, X_m$.

With every timed word $\rho = (\sigma_1, \tau_1) \dots (\sigma_n, \tau_n)$ over Σ_{AP} we associate a structure M^ρ whose universe is the non-negative real line $\mathbb{R}_{\geq 0}$. The monadic predicate Q_{act} holds at $t \in \mathbb{R}_{\geq 0}$ if t is an '*action point*', i.e. $t \in \rho$. The monadic predicates $Q_p x$ may only hold at action points, i.e. it holds at $t \in \mathbb{R}_{\geq 0}$ iff $t = \tau_i$ for some i, $1 \leq i \leq n$ and $p \in \sigma_i$. The order relation $x < x'$ is interpreted in the expected way. The *distance predicate* $d(x, x') \sim c$ holds iff $|x - x'| \sim c$. The set variables are quantified over *finite discrete* subsets of the universe, i.e. isolated points scattered along the non-negative real line.

The first observation is that allowing set variables to contain points between action points in ρ does not increase expressiveness; in other words, the class of languages expressible in the monadic second-order logic of order is the same in both the asynchronous semantics and the (traditional) synchronous semantics. This can be proved along the same lines as the proofs of Büchi's theorem [67, 68] and Proposition 1 (exploiting the fact that ϵ-transitions can easily be removed in non-deterministic automata).

Proposition 5. *The untimed fragment of $\mathcal{L}_d$ (obtained from $\mathcal{L}_d$ by disallowing distance predicates) in the asynchronous semantics is as expressive as non-deterministic automata.*

By contrast, $\mathcal{L}_d^{\leftrightarrow}$ becomes more expressive in the asynchronous semantics: it characterises exactly the class of timed languages expressible as TA_ϵ's (in the synchronous semantics, $\mathcal{L}_d^{\leftrightarrow}$ is as expressive as TA).

Proposition 6. *$\mathcal{L}_d^{\leftrightarrow}$ in the asynchronous semantics is as expressive as TA_ϵ.*

Proof (sketch). Following [25], we can get rid of all future relative distance formulae $\overrightarrow{d}(X, x) \sim c$ by introducing set variables that label the truth value of such formulae. When we replace $\exists X_1 \dots \exists X_m\, \psi$ with $\exists \bar{X}\, \exists \bar{E}\, \exists \bar{G}\, \exists \bar{G}'\, \exists K\, \psi'$, where ψ' is a modified formula using the new set variables, we also add conjuncts to ψ' to ensure that the new set variables only contain action points and points where at least one of $X_1, \dots, X_m$ holds. When adding transitions to the resulting automaton, make use of ϵ-transitions when Q_{act} does not hold. For the other direction, when capturing transitions with existentially quantified set variables, add conjuncts to ensure that those for ϵ-transitions are 'out-of-sync' with action points. □

To recover the decidability of HyperMITL model checking in the case of bounded-variable traces, we consider a bounded-variable asynchronous semantics for $\mathcal{L}_d^{\leftrightarrow}$ where all set variables are of bounded variability k_{var}, i.e. for each X, the formula $M^\rho, t \models Xx$ holds for at most k_{var} different t's in every open unit interval. In this case, we can show that $\mathcal{L}_d^{\leftrightarrow}$ is closed under complementation in almost the same way as in the synchronous semantics [25].

Proposition 7. *$\mathcal{L}_d^{\leftrightarrow}$ in the asynchronous semantics is closed under complementation if all set variables are of bounded variability k_{var} where k_{var} is a positive integer.*

Now, it is clear that HyperMITL model checking can be done when all traces are bounded-variable: the algorithm closely follows Proposition 1, but we have to add appropriate conjuncts to ensure that $\{Q_{p_i} \mid p \in \mathsf{AP}\}$ for some i $(1 \leq i \leq k)$ may only hold at points where Q_{act_i} holds. We can now state the main result of this subsection.

Proposition 8. *Model checking* HyperMITL *is decidable when all traces are of bounded variability k_{var} where k_{var} is a positive integer.*

6. Conclusion

We leave as future work to investigate whether a suitable notion of 'timing fuzziness' (e.g., [69, 70, 71]) can be incorporated, either to recover decidability of model checking or better align with practical applications, e.g., monitoring of cyber-physical systems [72, 73]. Another possible direction is to identify a decidable fragment of $\mathcal{L}_d^{\leftrightarrow}$ in the asynchronous semantics (this should be possible, e.g., MITL interpreted continuouly over timed words [74] is decidable) and see how it links back to a similar fragment of HyperMITL.

References

[1] A. Pnueli, The temporal logic of programs, in: FOCS, IEEE, 1977, pp. 46–57.

[2] A. P. Sistla, M. Y. Vardi, P. Wolper, The complementation problem for Büchi automata with applications to temporal logic (extended abstract), in: ICALP, Vol. 194 of LNCS, Springer, 1985, pp. 465–474.

[3] L. Stockmeyer, The complexity of decision problems in automata theory and logic, PhD thesis, TR 133, M.I.T., Cambridge (1974).

[4] G. J. Holzmann, The model checker SPIN, IEEE Transactions on Software Engineering 23 (5) (1997) 279–295.

[5] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, A. Tacchella, NuSMV2: An opensource tool for symbolic model checking, in: CAV, Vol. 2404 of LNCS, Springer, 2002, pp. 359–364.

[6] A. W. Roscoe, Csp and determinism in security modelling, in: S&P, IEEE Computer Society, 1995, pp. 114–127.

[7] S. Zdancewic, A. C. Myers, Observational determinism for concurrent program security, in: CSFW, IEEE Computer Society, 2003, p. 29.

[8] M. Huisman, P. Worah, K. Sunesen, A temporal logic characterisation of observational determinism, in: CSFW, IEEE Computer Society, 2006, p. 3.

[9] M. R. Clarkson, F. B. Schneider, Hyperproperties, Journal of Computer Security 18 (6) (2010) 1157–1210.

[10] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, C. Sánchez, Temporal logics for hyperproperties, in: POST, Vol. 8414 of LNCS, Springer, 2014, pp. 265–284.

[11] R. Alur, D. L. Dill, A theory of timed automata, Theoretical Computer Science 126 (2) (1994) 183–235.

[12] R. Koymans, Specifying real-time properties with metric temporal logic, Real-Time Systems 2 (4) (1990) 255–299.

[13] R. Alur, T. A. Henzinger, A really temporal logic, Journal of the ACM 41 (1) (1994) 164–169.

[14] R. Alur, T. Feder, T. A. Henzinger, The benefits of relaxing punctuality, Journal of the ACM 43 (1) (1996) 116–146.

[15] P. C. Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, in: CRYPTO, Vol. 1109 of LNCS, Springer, 1996, pp. 104–113.

[16] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, M. Hamburg, Meltdown: Reading kernel memory from user space, in: USENIX Security Symposium, USENIX Association, 2018, pp. 973–990.

[17] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, Y. Yarom, Spectre attacks: Exploiting speculative execution, CoRR abs/1801.01203 (2018).

[18] L. Simon, D. Chisnall, R. J. Anderson, What you get is what you C: Controlling side effects in mainstream C compilers, in: EuroS&P, IEEE, 2018, pp. 1–15.

[19] G. Barthe, G. Betarte, J. D. Campo, C. Luna, System-level non-interference of constant-time cryptography. Part I: Model, J. Autom. Reasoning 63 (1) (2019) 1–51.

[20] J. B. Almeida, M. Barbosa, G. Barthe, F. Dupressoir, M. Emmi, Verifying constant-time implementations, in: T. Holz, S. Savage (Eds.), USENIX Security, USENIX Association, 2016, pp. 53–70.

[21] B. Bond, C. Hawblitzel, M. Kapritsos, K. R. M. Leino, J. R. Lorch, B. Parno, A. Rane, S. T. V. Setty, L. Thompson, Vale: Verifying high-performance cryptographic assembly code, in: E. Kirda, T. Ristenpart (Eds.), USENIX Security, USENIX Association, 2017, pp. 917–934.

[22] S. Blazy, D. Pichardie, A. Trieu, Verifying constant-time implementations by abstract interpretation, Journal of Computer Security 27 (1) (2019) 137–163.

[23] Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, K. Ohta, Fault sensitivity analysis, in: S. Mangard, F.-X. Standaert (Eds.), CHES, Vol. 6225 of Lecture Notes in Computer Science, Springer, 2010, pp. 320–334.

[24] B. Bérard, A. Petit, V. Diekert, P. Gastin, Characterization of the expressive power of silent transitions in timed automata, Fundam. Inform 36 (2-3) (1998) 145–182.

[25] T. Wilke, Specifying timed state sequences in powerful decidable logics and timed automata, in: FTRTFT, Vol. 863 of LNCS, Springer, 1994, pp. 694–715.

[26] B. Finkbeiner, C. Hahn, M. Stenger, Eahyper: Satisfiability, implication, and equivalence checking of hyperproperties, in: CAV, Vol. 10427 of Lecture Notes in Computer Science, Springer, 2017, pp. 564–570.

[27] B. Finkbeiner, C. Hahn, T. Hans, Mghyper: Checking satisfiability of HyperLTL formulas beyond the $\exists^*\forall^*$ fragment, in: ATVA, Vol. 11138 of Lecture Notes in Computer Science, Springer, 2018, pp. 521–527.

[28] B. Finkbeiner, M. N. Rabe, C. Sánchez, Algorithms for model checking HyperLTL and $HyperCTL^*$, in: CAV, Vol. 9206 of LNCS, Springer, 2015, pp. 30–48.

[29] S. Agrawal, B. Bonakdarpour, Runtime verification of k-safety hyperproperties in HyperLTL, in: CSF, IEEE Computer Society, 2016, pp. 239–252.

[30] B. Finkbeiner, C. Hahn, M. Stenger, L. Tentrup, Monitoring hyperproperties, in: RV, Vol. 10548 of LNCS, Springer, 2017, pp. 190–207.

[31] B. Finkbeiner, C. Hahn, M. Stenger, L. Tentrup, RVHyper: A runtime verification tool for temporal hyperproperties, in: TACAS, Vol. 10806 of Lecture Notes in Computer Science, Springer, 2018, pp. 194–200.

[32] B. Bonakdarpour, B. Finkbeiner, The complexity of monitoring hyperproperties, in: CSF, IEEE Computer Society, 2018, pp. 162–174.

[33] L. V. Nguyen, J. Kapinski, X. Jin, J. V. Deshmukh, T. T. Johnson, Hyperproperties of real-valued signals, in: MEMOCODE, ACM, 2017, pp. 104–113.

[34] K. G. Larsen, P. Pettersson, W. Yi, Uppaal in a nutshell, International Journal on Software Tools for Technology Transfer 1 (1-2) (1997) 134–152.

[35] R. Alur, T. A. Henzinger, Logics and models of real time: A survey, in: REX, Vol. 600 of LNCS, Springer-Verlag, 1992, pp. 74–106.

[36] J. Ouaknine, J. Worrell, Some recent results in metric temporal logic, in: FORMATS, Vol. 5215 of LNCS, Springer, 2008, pp. 1–13.

[37] J. Heinen, Model checking timed hyperproperties, Master's thesis, Saarland University (2018).

[38] C. Gerking, D. Schubert, E. Bodden, Model checking the information flow security of real-time systems, in: ESSoS, Vol. 10953 of LNCS, Springer, 2018, pp. 27–43.

[39] G. Gardey, J. Mullins, O. H. Roux, Non-interference control synthesis for security timed automata, Electr. Notes Theor. Comput. Sci 180 (1) (2007) 35–53.

[40] P. Vasilikos, F. Nielson, H. R. Nielson, Secure information release in timed automata, in: POST, Vol. 10804 of LNCS, Springer, 2018, pp. 28–52.

[41] E. Ábrahám, B. Bonakdarpour, HyperPCTL: A temporal logic for probabilistic hyperproperties, in: QEST, Vol. 11024 of Lecture Notes in Computer Science, Springer, 2018, pp. 20–35.

[42] B. Finkbeiner, C. Hahn, H. Torfah, Model checking quantitative hyperproperties, in: CAV, Vol. 10981 of Lecture Notes in Computer Science, Springer, 2018, pp. 144–163.

[43] J. E. Hopcroft, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.

[44] J. Ouaknine, J. Worrell, On the language inclusion problem for timed automata: Closing a decidability gap, in: LICS, IEEE Computer Society, 2004, pp. 54–63.

[45] R. Alur, T. A. Henzinger, Real-time logics: Complexity and expressiveness, Information and Computation 104 (1) (1993) 35–77.

[46] J. Ouaknine, J. Worrell, On the decidability and complexity of metric temporal logic over finite words, Logical Methods in Computer Science 3 (1) (2007).

[47] R. Alur, T. A. Henzinger, Back to the future: towards a theory of timed regular languages, in: FOCS, IEEE Computer Society, 1992, pp. 177–186.

[48] B. Finkbeiner, C. Hahn, Deciding hyperproperties, in: CONCUR, Vol. 59 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 13:1–13:14.

[49] J. McLean, A general theory of composition for trace sets closed under selective interleaving functions, in: S&P, IEEE Computer Society, 1994, pp. 79–93.

[50] J. A. Goguen, J. Meseguer, Security policies and security models, in: S&P, IEEE Computer Society, 1982, pp. 11–20.

[51] L. Lamport, What good is temporal logic?, in: R.E.A. Mason (Ed.), IFIP Congress, North-Holland, Amsterdam, 1983, pp. 657–667.

[52] A. Kučera, J. Strejček, The stuttering principle revisited, Acta Informatica 41 (7–8) (2005) 415–434.

[53] T. A. Henzinger, J.-F. Raskin, P.-Y. Schobbens, The regular real-time languages, in: ICALP, Vol. 1443 of LNCS, Springer, 1998, pp. 580–591.

[54] J.-F. Raskin, Logics, automata and classical theories for deciding real time, Ph.D. thesis, FUNDP (Belgium) (1999).

[55] T. Brihaye, M. Estiévenart, G. Geeraerts, H.-M. Ho, B. Monmege, N. Sznajder, Real-time synthesis is hard!, in: FORMATS, Vol. 9884 of LNCS, Springer, 2016, pp. 105–120.

[56] D. Brand, P. Zafiropulo, On communicating finite state machines, Journal of the ACM 30 (1983) 323–342.

[57] D. D'Souza, P. Madhusudan, Timed control synthesis for external specifications, in: STACS, Vol. 2285 of LNCS, Springer, 2002, pp. 571–582.

[58] L. Doyen, G. Geeraerts, J.-F. Raskin, J. Reichert, Realizability of real-time logics, in: FORMATS, Vol. 5813 of LNCS, Springer, 2009, pp. 133–148.

[59] G. Barthe, P. R. D'Argenio, T. Rezk, Secure information flow by self-composition, Mathematical Structures in Computer

Science 21 (6) (2011) 1207–1252.

[60] P. A. Abdulla, J. Deneux, J. Ouaknine, K. Quaas, J. Worrell, Universality analysis for one-clock timed automata, Fundam. Inform 89 (4) (2008) 419–450.

[61] T. Ferrère, The compound interest in relaxing punctuality, in: FM, Vol. 10951 of LNCS, Springer, 2018, pp. 147–164.

[62] A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, L. Xu, Spot 2.0 - a framework for LTL and ω-automata manipulation, in: ATVA, Vol. 9938 of LNCS, Springer, 2016, pp. 122–129.

[63] M.-H. Tsai, Y.-K. Tsay, Y.-S. Hwang, Goal for games, omega-automata, and logics, in: CAV, Vol. 8044 of Lecture Notes in Computer Science, Springer, 2013, pp. 883–889.

[64] J. Kretínský, T. Meggendorfer, S. Sickert, Owl: A library for ω-words, automata, and ltl, in: ATVA, Vol. 11138 of Lecture Notes in Computer Science, Springer, 2018, pp. 543–550.

[65] J. Ouaknine, A. Rabinovich, J. Worrell, Time-bounded verification, in: Proceedings of CONCUR 2009, Vol. 5710 of LNCS, Springer, 2009, pp. 496–510.

[66] H.-M. Ho, On the expressiveness of metric temporal logic over bounded timed words, in: RP, Vol. 8762 of Lecture Notes in Computer Science, Springer, 2014, pp. 138–150.

[67] J. R. Büchi, Weak second-order arithmetic and finite automata, Zeitschrift für mathematische Logik und Grundlagen der Mathematik 6 (1960) 66–92.

[68] J. R. Büchi, On a Decision Method in Restricted Second Order Arithmetic, in: Proceedings of the 1960 International Congress of Logic, Methodology and Philosophy of Science, Stanford University Press, 1962, pp. 1–12.

[69] R. Alur, S. L. Torre, P. Madhusudan, Perturbed timed automata, in: HSCC, Vol. 3414 of LNCS, Springer, 2005, pp. 70–85.

[70] V. Gupta, T. A. Henzinger, R. Jagadeesan, Robust timed automata, in: HART, Vol. 1201 of LNCS, Springer, 1997, pp. 331–345.

[71] A. Donzé, O. Maler, Robust satisfaction of temporal logic over real-valued signals, in: FORMATS, Vol. 6246 of LNCS, Springer, 2010, pp. 92–106.

[72] B. Bonakdarpour, J. V. Deshmukh, M. Pajic, Opportunities and challenges in monitoring cyber-physical systems security, in: ISoLA, Vol. 11247 of LNCS, Springer, 2018, pp. 9–18.

[73] E. Bartocci, J. V. Deshmukh, A. Donzé, G. E. Fainekos, O. Maler, D. Nickovic, S. Sankaranarayanan, Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications, in: Lectures on Runtime Verification, Vol. 10457 of LNCS, Springer, 2018, pp. 135–175.

[74] D. D'Souza, P. Prabhakar, On the expressiveness of MTL in the pointwise and continuous semantics, International Journal on Software Tools for Technology Transfer 9 (1) (2007) 1–4.