2013 20th International Symposium on Temporal Representation and Reasoning

# Complexity of Model Checking over General Linear Time

Tim French, John M$^{c}$Cabe-Dansted and Mark Reynolds
School of Computer Science and Software Engineering
The University of Western Australia
35 Stirling Highway, Crawley WA 6009
Perth, Australia.
Email: {tim.french,john.mccabe-dansted,mark.reynolds}@uwa.edu.au

***Abstract*—Temporal logics over general linear time allow us to capture continuous properties in applications such as distributed systems, natural language, message passing and A.I. modelling of human reasoning. Linear time structures, however, can exhibit a wide range of behaviours that are hard to reason with, or even describe finitely. Recently, a formal language of Model Expressions has been proposed to allow the convenient finite description of an adequately representative range of these generally infinite structures. Given a model described in this Model Expression language and a temporal logic formula, a model checking algorithm decides whether the formula is satisfied at some time in the model. Tools based on such algorithms would support a wide variety of tasks such as verification and counter-example investigation.**

**A previous paper gave an exponential space algorithm for the problem of model checking Until/Since temporal formulas over linear time Model Expressions. Here we prove that the problem is actually PSPACE-complete. We present a new PSPACE algorithm and we show PSPACE-hardness by a reduction from quantified boolean formulas.**

***Keywords*-Model Checking; Complexity; General Linear Flows; Dense Time; Logic**

## I. Introduction

Since [1] first made the radical suggestion that a model with a continuous flow of time might be better for handling concurrent systems than traditional discrete temporal logics, there has been steadily increasing development of appropriate non-discrete formalisms, techniques, tools and their foundations. The important quality assurance task of verification of a system, formalised as model-checking, has developed from simple checking of a Kripke structure against an LTL formula in discrete time [2] to checking all traces of a timed automata in continuous time against temporal metric constraints [3].

Here we also consider model checking of temporal properties in not necessarily discrete time models but we generalise the problem in several important but related ways. We consider any underlying linear models of time and we do not assume that the structure to "check" has been generated by the action of some finite mechanism moving from discrete state to discrete state. So time may be the non-negative reals or the natural numbers but it may instead be the rationals or the integers or some finite sequence of continuous intervals interspersed with discrete points. If a sentence of natural language or some strange formalism can determine the behaviour of a signal over a linear order, we can check it.

Our second generalisation is more important. Timed automata are tremendously useful mathematical objects for verification and design tasks [4], [5] but there are many properties exhibited in the real continuous world that do not seem amenable to easy modelling by their fundamental discrete step by step action. To name just a few consider the following: the approach of Achilles to Zeno's Tortoise [6]; a fractal signal [7] (see Figure 1); the infinitesimal chattering of a sliding controller [8] (like a perfect ABS braking system); or mathematically defined hypothetical signals like one true at precisely the irrational real numbers. An automaton allowing a cycle of shorter and shorter durations may give an indication of the possibility of some Zeno type behaviour but it does not allow specification of what happens after the accumulation point. Furthermore, even developers of more conventional automata-like machines have to sometimes consider the behaviour of their system with an environment that is less well-behaved and they would benefit from being able to describe a wide range of environmental possibilities. In this paper, we allow structures exhibiting any definable boolean signals.

A simple and long-established propositional temporal language adequate for our situation is $L(U, S)$, the propositional temporal language using Kamp's Until and Since connectives [9]. We interpret $L(U, S)$ over general linear flows and call the logic US/L. If we restrict semantics to just the real numbers then the logic is sometimes called RTL—roughly Metric Temporal Logic [10], [11] without the metrics.

Recent work [12] addressed the problem of synthesis of linear models of US/L formulas—finding an algorithm to describe a linear model of any given satisfiable formula. This required a formal language for outputting details of models. To do this, the paper presented a compositional language for model expressions, building upon pioneering work by [13], [14], [15], [16]. An abstract syntax with a few basic composition operators allows an expression to specify how to iteratively build an arbitrarily complex linear structure from simple singleton point structures. This language is expressively adequate, as any satisfiable formula of US/L has a model representable in this language [12].

The work was partially supported by the Australian Research Council.

1530-1311/13 $26.00 © 2013 IEEE
DOI 10.1109/TIME.2013.21

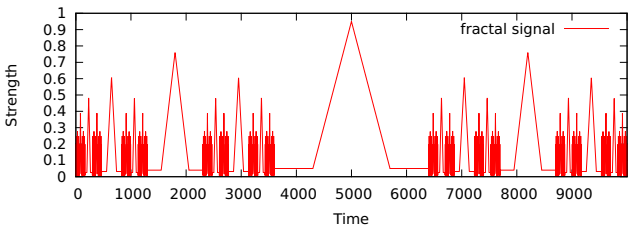


Fig. 1. An example of a fractal signal

Having finite representations for structures also opens the possibility for an algorithm for model checking formulas against those representations. Such a general linear time model checking algorithm was presented for this problem in [17]. The algorithm had complexity exponential in time and space.

Let us consider a example such as the fractal signals of, e.g., [7]. Figure 1 shows a simplified hypothetical fractal signal where all spikes are upwards. Let $p$ hold during periods of increasing signal, $q$ describe a decreasing one, and $r$ represent a constant signal. Assume that these propositions also hold at the start and ends of the appropriate intervals. Using the new notation of [12] can then represent the pattern of increments and decrements in this signal with the Model Expression $\langle\langle r\rangle + \{r,p\} + \langle p\rangle + \{p,q\} + \langle q\rangle + \{q,r\} + \langle r\rangle\rangle$. Useful properties of this signal can be specified in $L(U,S)$. For example, we can reach a region of increment directly from a constant region so $r \wedge U\,(p, r \vee p)$ is satisfied within the model. However, we cannot reach a region of decrement from a region of increment, so $r \wedge U\,(q, r \vee q)$ is not satisfied within the model. Checking such claims is the work of a model-checking algorithm.

In this paper, we will show that model checking US/L formulas against (general linear time) Model Expressions is actually PSPACE-Complete. We show PSPACE-hardness by a reduction from the satisfiability problem for quantified boolean formula. This hardness result is quite robust in the sense that it only needs a fraction of the operators available, and removing more operators results in a problem in P.

We then show that the problem is in PSPACE by giving a polynomial space model checking algorithm. This is essentially an improved variant of the [17] algorithm. However, we do not want to claim that the new algorithm would be better for practical use. For this reason we have not implemented the algorithm although an implementation of the original algorithm is available [18].

There are some similarities between the model checking task here and the task of model checking LTL formulas against "compressed paths" as described in [19]. In fact, we have been able to follow a similar approach to theirs for our proof of PSPACE-hardness. Like model expressions, the compressed path formalism allows concise expressions of structures (but there only finite discrete ones) via duplication of submodels. However, the compressed paths of [19] were intended to represent finite discrete paths efficiently and thus only allowed finite duplications of submodels. By contrast [12] focussed on expressively complete finite representations of infinite/dense structures, for which finite duplication is not required, so the formalism has only operators for infinite repeats of submodels. Thus the actual model-checking algorithms end up being quite different.

In section 2 we define the logic US/L and in section 3, we define model expressions. Section 4 contains our PSPACE-hardness result and section 5 presents the new PSPACE model checking algorithm.

## II. The Logic

In this section, we introduce the logic US/L. We first define the temporal language $L(U,S)$ of Until and Since. Then US/L is just the logic of $L(U,S)$ over the class of all structures with a linear flow of time.

$L(U,S)$ is a propositional language and we work with a fixed countable set $\mathbf{L}$ of atomic propositions (or atoms). A frame $(T,<)$, or flow of time, is any irreflexive linear order. Structures $\mathbf{T} = (T,<,h)$ will have a frame $(T,<)$ and a valuation $h$ for the atoms, i.e. for each atom $p \in \mathbf{L}$, $h(p) \subseteq T$.

The well-formed formulas of $L(U,S)$ are generated from the atomic propositions by the 2-place connectives $U$ and $S$ (both used in a prefix manner) along with the boolean connectives $\neg$ and $\wedge$. That is, we define the set of formulas recursively to contain the atoms and for formulas $\alpha$ and $\beta$, we include $\neg\alpha$, $\alpha \wedge \beta$, $U(\alpha,\beta)$ and $S(\alpha,\beta)$.

Formulas are evaluated at points in structures $\mathbf{T} = (T,<,h)$. We write $\mathbf{T}, x \models \alpha$ when $\alpha$ is true at point $x \in T$. This is defined inductively as follows. Suppose that we have defined the truth of formulas $\alpha$ and $\beta$, at all points of $\mathbf{T}$. Then for all points $x$:

| | | |
|---|---|---|
| $\mathbf{T}, x \models p$ | iff | $x \in h(p)$, for $p$ atomic; |
| $\mathbf{T}, x \models \neg\alpha$ | iff | $\mathbf{T}, x \not\models \alpha$; |
| $\mathbf{T}, x \models \alpha \wedge \beta$ | iff | both $\mathbf{T}, x \models \alpha$ and $\mathbf{T}, x \models \beta$; |
| $\mathbf{T}, x \models U(\alpha,\beta)$ | iff | there is $y > x$ in $T$ such that $\mathbf{T}, y \models \alpha$ and for all $z \in T$ such that $x < z < y$ we have $\mathbf{T}, z \models \beta$; and |
| $\mathbf{T}, x \models S(\alpha,\beta)$ | iff | there is $y < x$ in $T$ such that $\mathbf{T}, y \models \alpha$ and for all $z \in T$ such that $y < z < x$ we have $\mathbf{T}, z \models \beta$. |

Along with standard logical abbreviations $\alpha \vee \beta = \neg(\alpha \wedge \beta)$, $\alpha \rightarrow \beta = \neg\alpha \vee \beta$, and $\alpha \leftrightarrow \beta = (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$, we use the following temporal abbreviations in illustrating the logic: $F\alpha = U(\alpha, \top)$, "alpha will be true (sometime in the future)" $G\alpha = \neg F(\neg\alpha)$, "alpha will always hold (in the future)"; and their past time versions $P$ and $H$.

## III. Building Structures

As in [12], we define a notation which allows the description of temporal structures. Each expression tells us how to build a structure out of simple basic structures via four ways of putting structures together to form larger ones.

The general idea is simple. Using singleton structures—the flow of time is one point—we build up to more complex

structures by the iterative application of any of the operations. There are four operations available:

1) concatenation of two structures, consisting of one followed by the other;
2) $\omega$ repeats of some structure laid end to end towards the past;
3) $\omega$ repeats laid end to end towards the future;
4) and making a densely thorough *shuffle* of copies from a finite set of structures.

These operations are well-known from the study of linear orders (see e.g. [14]).

*Model Expressions* are an abstract syntax to define models using the following set of primitive operators based on the four operations:

$$\mathcal{I} ::= \ a \mid \lambda \mid \mathcal{I} + \mathcal{J} \mid \overleftarrow{\mathcal{I}} \mid \overrightarrow{\mathcal{I}} \mid \langle \mathcal{I}_0, \cdots, \mathcal{I}_n \rangle$$

where $a \in \Sigma = 2^{\mathbf{L}}$ so the letter indicates the atoms true at a point. We refer to these operators, respectively, as *a letter*, *the empty order*, *concatenation*, *lead*, *trail*, and *shuffle*.

**Definition 1.** [Correspondence] A model expression $\mathcal{I}$ *corresponds* to a structure as follows:

- $\lambda$ is the empty sequence and corresponds to the (pseudo) frame[1] $(\emptyset, <, h)$ where $<$ and $h$ are empty relations.
- $a$ corresponds to any single point structure $(\{x\}, <, h)$ where $<$ is the empty relation and $h(p) = \{x\}$ if and only if $p \in a$.
- $\mathcal{I}+\mathcal{J}$ corresponds to a structure $(T, <, h)$ if and only if $T$ is the disjoint union of two sets $U$ and $V$ where $\forall u \in U$, $\forall v \in V$, $u < v$ and $\mathcal{I}$ corresponds to $(U, <^U, h^U)$ and $\mathcal{J}$ corresponds to $(V, <^V, h^V)$. The symbols $<^U, h^U$ refer to the restriction of the relations $<$ and $h$ to apply only to elements of $U$.
- $\overleftarrow{\mathcal{I}}$ corresponds to the structure $(T, <, h)$ if and only if $T$ is the disjoint union of sets $\{U_i | i \in \omega\}$ where for all $i$, for all $u \in U_i$, for all $v \in U_{i+1}$, $v < u$, and $\mathcal{I}$ corresponds to $(U_i, <^{U_i}, h^{U_i})$.
- $\overrightarrow{\mathcal{I}}$ corresponds to the structure $(T, <, h)$ if and only if $T$ is the disjoint union of sets $\{U_i | i \in \omega\}$ where for all $i$, for all $u \in U_i$, for all $v \in U_{i+1}$, $u < v$, and $\mathcal{I}$ corresponds to $(U_i, <^{U_i}, h^{U_i})$.
- $\langle \mathcal{I}_0, \cdots, \mathcal{I}_n \rangle$ corresponds to the structure $(T, <, h)$ if and only if $T$ is the disjoint union of sets $\{U_i | i \in \mathbb{Q}\}$ where
  1) for all $i \in \mathbb{Q}$, $(U_i, <^{U_i}, h^{U_i})$ corresponds to some $\mathcal{I}_j$ for $j \leq n$,
  2) for every $j \leq n$, for every $a \neq b \in \mathbb{Q}$, there is some $k$ in the open interval $(a, b)$ where $\mathcal{I}_j$ corresponds to $(U_k, <^{U_k}, h^{U_k})$,
  3) for every $a < b \in \mathbb{Q}$, for all $u \in U_a$, for all $v \in U_b$, $u < v$.

To allow more elegant phrasing we take "$\mathcal{I}$ corresponds to $\mathbf{T}$" to be equivalent to "$\mathbf{T}$ corresponds to $\mathcal{I}$". The MEs are important for modelling US/L formulas. Every satisfiable US/L formula $\phi$ has an ME $\mathcal{I}$ such that a structure $\mathbf{T}$ that corresponds to $\mathcal{I}$ will satisfy $\phi$, yet MEs are minimal in that excluding any operator (except $\lambda$) will render them unable to represent models for some US/L formulas.

We will give an illustration of the non-trivial operations below. The *lead* operation, $\mathcal{I} = \overleftarrow{\mathcal{J}}$ has $\omega$ submodels, each corresponding to $\mathcal{J}$, and each preceding the last, as illustrated in Figure 2.

The *trail* operator is the mirror image of *lead*, whereby $\mathcal{I} = \overrightarrow{\mathcal{J}}$ has $\omega$ structures, each corresponding to $\mathcal{J}$ and each proceeding the earlier structures.

The *shuffle* operator is harder to represent with a diagram. The model expression $\mathcal{I} = \langle \mathcal{I}_1, \ldots \mathcal{I}_n \rangle$ corresponds to a dense, thorough mixture of intervals corresponding to $\mathcal{I}_1$, ..., $\mathcal{I}_n$, without endpoints. We define the shuffle operation using the rationals, $\mathbb{Q}$ as they are a convenient order with the required properties.

The definition of model expressions is not deterministic, as the construct for the shuffle $\langle \mathcal{I}_1, \cdots, \mathcal{I}_n \rangle$ does not specify how the structures corresponding to $\mathcal{I}_1, \cdots, \mathcal{I}_n$ are mapped to $\mathbb{Q}$. This is inconsequential, and as long as the mapping is dense for each $i$ from 1 to $n$, the resulting structures will be models of the same formulas [12].

## IV. Hardness Result

**Definition 2.** We define the *USME-checking problem* as follows: given an ME $\mathcal{I}$ and formula $\phi$, determine whether there exists a structure $\mathbf{T} = (T, <, h)$ corresponding to $\mathcal{I}$ and point $x \in T$ such that $\mathbf{T}, x \vDash \phi$.

The reduction we will use to prove PSPACE-hardness will be from satisfiability of prenex QBF formulas. We now provide a definition of these formulas.

**Definition 3.** A *prenex QBF* is a formula of the form $\exists r_n \forall r_{n-1} \cdots \exists r_2 \forall r_1 \psi$ where $n$ is even and $\psi$ is a Boolean formula using only atoms in the set $\{r_1, \ldots, r_n\}$. A subformula of a prenex QBF formula is a QBF formula. For QBF formulas $\alpha$, $\beta$ and atom $r$ we let $\alpha\,[r/\beta]$ be the formula obtained from $\alpha$ by replacing each occurrence of the atom $r$ with $\beta$. For any Boolean formula $\psi$ and atom $r$, $\forall r \psi$ is equivalent to $\psi\,[r/\top] \wedge \psi\,[r/\bot]$ and $\exists r \psi$ is equivalent to $\neg \forall r \neg \psi$ and hence $\psi\,[r/\top] \vee \psi\,[r/\bot]$.

We see that a prenex QBF is equivalent to a boolean formula with all atoms replaced with $\top$ or $\bot$. As such, a prenex QBF is equivalent to a theorem of Propositional Calculus iff it is satisfiable. We say that a prenex QBF evaluates to true iff it is equivalent to a theorem (or equivalently, if it is satisfiable).

The reduction from QBF satisfiability solving to model checking compressed paths [19] makes use of efficient replication of identical submodels. In our reduction to MEs we will use the $\leftarrow$ operator to replicate submodels; however, this gives us less flexibility. Whereas [19] distinguished between duplicated submodels by following one with an $f_n$ atom and the other with a $t_n$ atom we instead use the Until operator to

[1]Of course, the empty pseudo-frame is not counted as a frame and we do not allow empty structures.

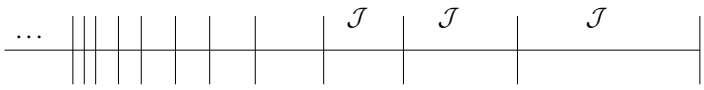


Fig. 2. The lead operation, where $\mathcal{I} = \overleftarrow{\mathcal{J}}$

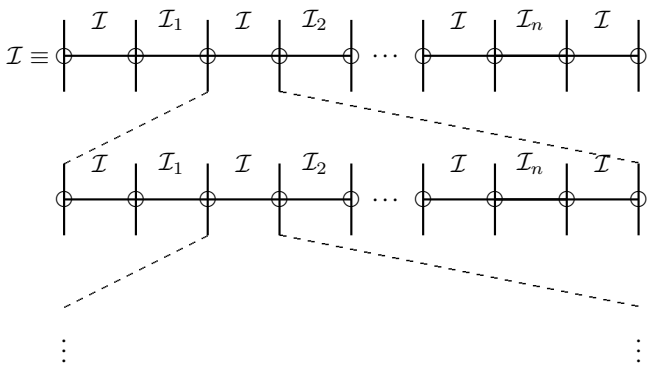


Fig. 3. The shuffle operation, where $\mathcal{I} = \langle \mathcal{I}_1, \cdots, \mathcal{I}_n \rangle$

define a formula $\alpha_i$ to distinguish the last instance from the first. Using this trick we now define a reduction from testing the satisfiability of QBF formula to ME-checking as follows:

**Definition 4.** For the rest of this section we let $\mathcal{J}_0 = \emptyset$ and let $\mathcal{J}_i = \overleftarrow{\{p_i\} + \mathcal{J}_{i-1}} + \{q_i\}$ for each positive $i$, let $\alpha_i = \neg(q_i \vee U(p_i, \neg q_i))$, fix a positive even integer $n$, let $\phi_q = \exists r_n \forall r_{n-1} \cdots \exists r_2 \forall r_1 \phi$ be some prenex QBF formula on the set of atoms $r_1, \ldots, r_n$, and let $\phi'$ be the formula that results when each $r_i$ is replaced by $\alpha_i$ in the Boolean formula $\phi$. We let $\psi_0 = \phi'$ and in general let $\psi_i = U(q_i, p_i \rightarrow \psi_{i-1})$ for positive odd $i$ and let $\psi_i = \neg(U(q_i, \neg(p_i \wedge \psi_{i-1})))$ for positive even $i$. We reduce the problem "Is the QBF formula $\phi_q$ satisfiable" to checking the $L(U,S)$ formula $p_{\text{start}} \rightarrow \psi_n$ against the ME $\{p_{\text{start}}\} + J_n$.

The truth of $\alpha_i$ at $q_i$ points is unimportant to the correctness of the reduction and so we could have defined $\alpha_i$ as being just $\neg U(p_i, \neg q_i)$; however, ensuring $\alpha_i$ is not true at the same time as $q_i$ makes the formal statement and proof of Lemma 6 below simpler. The intuition behind our choice of $\psi_i$ is that $\alpha_j$ will be true throughout the last (prior to $\{q_i\}$) submodel corresponding to $\{p_i\} + \mathcal{J}_{i+1}$, but false throughout the previous such submodels. Thus $U(q_i, p_i \rightarrow \psi_{i-1})$ means "for points satisfying $p_i$, including points where $\alpha_i$ is false and points where $\alpha_i$ is true, it is the case that $\psi_{i-1}$", and so is used as our translation of $\forall r_i$. Likewise the intuition behind $\neg(U(q_i, \neg(p_i \wedge \psi_{i-1})))$ is that it requires $\psi_{i-1}$ at one of the points satisfying $p_i$, and we can choose from one of these points that satisfy $\alpha_i$ or the point which satisfies $\neg\alpha_i$. We will now show that this reduction is correct and thus the ME-checking problem is PSPACE-hard. It can be convenient to treat formulas as atoms, so we provide the definitions:

**Definition 5.** Say $\mathbf{T} = (T, <, h)$ is a structure, then $\mathbf{T}$ with an $L(U,S)$ formula $\alpha$ added as an atom is $\mathbf{T}^{\alpha} = (T, <, h')$ interpreted over the set of atoms $\mathbf{L} \cup \{\alpha\}$ where $h'(p) = h(p)$ for all $p \in \mathbf{L}$ and for each point $x \in T$ we have $x \in h'(\alpha)$ iff $\mathbf{T}, x \vDash \alpha$. Where $\mathcal{I}$ is an ME we let $\mathcal{I}^{\alpha}$ be the ME that results when $\alpha$ is added to every letter of $\mathcal{I}$ (not just letters at which $\alpha$ would be satisfied). Given a set $\Psi = \{\alpha_1, \ldots, \alpha_m\}$ of $L(U,S)$ formulas and an ME $\mathcal{I}$ we define $\mathcal{I}^{\Psi}$ to be the result $\mathcal{I}^{\alpha_1 \cdots \alpha_1}$ of replacing each letter $a$ in $\mathcal{I}$ with $a \cup \Psi$.

Note that if any structure corresponds to $\overleftarrow{\mathcal{I}}$ then it must also correspond to $\overleftarrow{\mathcal{I}} + \mathcal{I}$ [20]. Using this we can easily prove the next lemma inductively.

**Lemma 6.** *For each $i$ we let $\mathbf{T}_i = (T, <, h)$ be a structure corresponding to $\mathcal{J}_i$. The structure $\mathbf{T}^{\alpha_1 \cdots \alpha_i}$ with each $\alpha_i$ added as an atom corresponds to $\mathcal{J}'_i$ where $\mathcal{J}'$ is defined recursively: $\mathcal{J}'_0 = \emptyset$ and for each positive $i$ we let $\mathcal{J}'_i = \overleftarrow{\{p_i\} + \mathcal{J}'_{i-1}} + \{p_i, \alpha_i\} + \mathcal{J}'^{\alpha_i}_{i-1} + \{q_i\}$.*

*Proof:* The lemma is trivially true for $i = 0$. Say that the lemma is correct up to $i-1$, that is $\mathcal{J}'_{i-1}$ corresponds to $\mathbf{T}^{\alpha_1 \cdots \alpha_{i-1}}_{i-1}$. By definition $\overleftarrow{\{p_i\} + \mathcal{J}_{i-1}} + \{q_i\}$ and the equivalent ME $\overleftarrow{\{p_i\} + \mathcal{J}_{i-1}} + \{p_i\} + \mathcal{J}_{i-1} + \{q_i\}$ correspond to $\mathbf{T}_i$. We see that $\mathbf{T}^{\alpha_1 \cdots \alpha_{i-1}}_i$ corresponds to $\overleftarrow{\{p_i\} + \mathcal{J}'_{i-1}} +$

$\{p_i\} + \mathcal{J}'_{i-1} + \{q_i\}$. Recall that $\alpha_i = \neg(q_i \vee U(p_i, \neg q_i))$. At the final point $q_i$ is true so $\alpha_i$ is false. The atom $p_i$ does not occur within $\mathcal{J}'_{i-1}$ so $U(p_i, \neg q_i)$ must be false throughout the $\mathcal{J}'_{i-1}$ immediately preceding $\{q_i\}$; hence $\alpha_i$ is true there. As we have defined Until to be strict, $U(p_i, \neg q_i)$ is also false at the last occurrence of $p_i$. The atom $q_i$ does not occur within $\overleftarrow{\{p_i\} + \mathcal{J}'_{i-1}}$ so we see that $U(p_i, \neg q_i)$ is true when followed by $p_i$. Hence $\mathbf{T}_i^{\alpha_1 \cdots \alpha_{i-1} \alpha_i}$ corresponds to $\mathcal{J}'_i = \overleftarrow{\{p_i\} + \mathcal{J}'_{i-1}} + \{p_i, \alpha_i\} + \mathcal{J}'^{\alpha_i}_{i-1} + \{q_i\}$. By induction the lemma holds for all non-negative integers $i$. ■

From Lemma 6 we see that for any subset $S \subseteq [1, n]$ there exists a point $x \in T$ such that for all $i \in [1, n]$ we have $\mathbf{T}, x \vDash \alpha_i$ iff $i \in S$. This is enough to give us NP-hardness. We see that a USME-checker returns "true", when given $\phi'$ and $\mathcal{I}_n$ as input, precisely when $\phi$ is satisfiable. As we have a polynomial time reduction from Boolean satisfiability solving, an NP-hard problem [21], we know the USME-checking problem is NP-hard; however, the goal is to prove PSPACE-hardness and so we need a reduction from a PSPACE-hard problem. Boolean satisfiability with alternating quantifiers is PSPACE-complete (see for example [22]). Satisfiability of Quantified Boolean Formulas (QBF) is trivially reducible to satisfiability of QBF in prenex normal form, so we will only consider prenex QBF and QBF that are subformulas of a prenex QBF.

**Definition 7.** Given a QBF formula $\beta$ and set $S \subseteq \{1, \ldots, n\}$, we let $\beta^S$ be the formula that results when each free variable $r_j$ in $\beta$ is replaced with $\top$ if $j \in S$ and $\bot$ if $j \notin S$. Let $\phi_0 = \phi$, for each positive odd $i$ let $\phi_i = \forall r_i \phi_{i-1}$ and for each positive even $i$ let $\phi_i = \exists r_i \phi_{i-1}$.

Note that the $\phi_n$ defined above is the same as the prenex QBF $\phi_q = \exists r_n \forall r_{n-1} \cdots \exists r_2 \forall r_1 \phi$. The following lemma then demonstrates the existence of a reduction from satisfiability solving of QBF.

**Lemma 8.** *Let $\mathbf{T} = (T, <, h)$ be a structure corresponding to $\mathcal{J}_n$. Let $\mathbf{T}' = (T, <, h') = \mathbf{T}^{\alpha_1 \cdots \alpha_n}$. Let $i \in \{0, \ldots, n\}$, let $S = \{s_1, \ldots, s_m\}$ be a subset of $\{i+1, \ldots, n\}$, let $\Psi = \{\alpha_{s_1}, \ldots, \alpha_{s_m}\}$. Finally, let $U$ be an interval of points in $\mathbf{T}$ such that $\mathcal{J}'^{\Psi}_i$ corresponds to $\mathbf{T}'_U$ and $z$ be a point immediately prior to $U$. Then $\phi_i^S$ evaluates to true precisely if $\mathbf{T}, z \vDash \psi_i$.*

*Proof:* Assume that $i$ is the smallest $i$ that provides a counterexample to this lemma.

We see that if $i = 0$ then there is one point $x \in U$ and that $\mathbf{T}_U$ corresponds to $\emptyset$. We see that for each $j$ we have $x \in h'(\alpha_j)$ iff $j \in S$, and further for a point $z$ immediately prior to $x$, we have $x \in h'(\alpha_j) \iff \mathbf{T}, x \vDash \alpha_j \iff \mathbf{T}, z \vDash \alpha_j$. Each atom $r_j$ in $\phi_i^S = \phi_0^S = \phi^S$ has been substituted for $\top$ or $\bot$, for $\top$ if $j \in S$ (and $\mathbf{T}, z \vDash \alpha_j$) or $\bot$ if $j \notin S$ (and $\mathbf{T}, z \vDash \neg\alpha_j$). Thus $\mathbf{T}, z \vDash \psi_0$ iff $\phi^S$ evaluates to true.

Now consider $i > 0$. Recall that $\mathbf{T}_U$ corresponds to

$$\mathcal{J}'^{\Psi}_i = \overleftarrow{\{p_i\} \cup \Psi + \mathcal{J}'^{\Psi}_{i-1}} + \{p_i, \alpha_i\} \cup$$

$$\Psi + \mathcal{J}'^{\Psi \cup \alpha_i}_{i-1} + \{q_i\}.$$

We see that for each $x \in h_U(p_i)$ either $\{p_i, \alpha_i\} \cup \Psi$ or $\{p_i\} \cup \Psi$ corresponds to $\mathbf{T}_{\{x\}}$. We see that where $\{p_i\} \cup \Psi$ corresponds to $\mathbf{T}_{\{x\}}$, $x$ is immediately prior to an interval of points $V$ such that $\mathbf{T}_V$ corresponds to $\mathcal{J}'^{\Psi}_{i-1}$. As $i$ is a minimal counter example $i-1$ does not provide a counter example. Thus $\mathbf{T}, x \vDash \psi_{i-1}$ iff $\phi_{i-1}^S$ evaluates to true. Likewise if $\{p_i, \alpha_i\} \cup \Psi$ corresponds to $\mathbf{T}'_{\{x\}}$ we see that $\mathbf{T}, x \vDash \psi_{i-1}$ iff $\phi_{i-1}^{S \cup i}$ evaluates to true. We now further divide the case of $i > 0$ into odd and even cases.

Suppose $i$ is odd. Then $\psi_i = U(q_i, p_i \to \psi_{i-1})$ and $\phi_i = \forall r_i \phi_{i-1}$. We see that $\mathbf{T}, z \vDash \psi_i$ is true precisely if for all $x \in h_U(p_i)$ we have $\mathbf{T}, x \vDash \psi_{i-1}$. From the previous paragraph we see that for the final $x \in h_U(p_i)$ we have $T, x \vDash \psi_{i-1}$ iff $\phi_{i-1}^{S \cup i}$ evaluates to true, and for the other $x \in h_U(p_i)$ we have $T, x \vDash \psi_{i-1}$ iff $\phi_{i-1}^S$ evaluates to true. Thus $\mathbf{T}, z \vDash \psi_{i-1}$ precisely if $\phi_{i-1}^{S \cup i}$ and $\phi_{i-1}^S$ both evaluate to true; or in other words $\phi_i^S = (\forall r \phi_{i-1})^S$ is true. By contradiction, $i$ is even.

As $i$ is even, $\psi_i = \neg(U(q_i, \neg(p_i \wedge \psi_{i-1})))$ and $\phi_i = \exists r_i \phi_{i-1}$. We see that $\mathbf{T}, z \vDash \psi_i$ is true precisely if for some $x \in h_U(p_i)$ we have $\mathbf{T}, x \vDash \psi_{i-1}$. There is such an $x$ iff either $\phi_{i-1}^{S \cup i}$ or $\phi_{i-1}^S$ evaluate to true; or in other words $\phi_i^S = (\exists_{r_i} \phi_{i-1})^S$ is true.

We have now shown that the minimal counter example $i$ cannot be zero, a positive odd number or a positive even number. By contradiction, no counterexample exists. ■

This lemma demonstrates that our reduction from QBF is correct, and gives us the following theorem:

**Theorem 9.** *The USME-Checking problem is PSPACE-hard.*

Note that a more traditional definition of the model checking problem would only require us to determine whether the formula is a satisfied at a particular point. Note that this definition would still be PSPACE-hard as we are only interested in the truth of $\psi_n$ at the first point of the reduction. Also note that $\psi_n$ does not use the Since operator so limiting formulas to those of $L(U)$ (or $L(S)$, as the Since operator is essentially just a mirror image of the until operator) would still leave a PSPACE-hard model-checking problem. In the next section we outline a polynomial space implementation for USME-checking, showing that the USME-checking problem is PSPACE-complete.

Note also that we used the $q_i$ atoms to assist in reasoning about the correctness of the reduction. We do not claim that this reduction is in any sense minimal. Consider, for example, a smaller reduction similar to above but where $\mathcal{J}_0 = \{p_1\}$, for each positive $i$ we have $\mathcal{J}_i = \overleftarrow{\{p_{i+1}\} + \mathcal{J}_{i-1}}$, $\alpha_i = U(p_i, \neg p_{i+1})$ and we model check the formula $\phi_n$ against the model $\mathcal{J}_n$.

Note that this PSPACE-hardness result only requires MEs with $\leftarrow$ and + (not shuffles or $\to$) and does not require the Since operator in the formula. Equivalently we could use $\to$ and + in the ME and exclude Until from the formula. Thus this result is quite robust to restrictions of the ME

and/or formula. An obvious question is whether we can get a result that is even stronger. We now consider some obvious stronger restrictions, and show that all of these restrictions lead to a problem in P. Given that we consider a number of restrictions, and that a full discussion of their precise complexity would require reference to the implementation in [17], we do not consider each restriction in detail, for example we will not discuss whether the restriction is P-hard. Note that, in practice, the unrestricted ME-checking problem is fairly easy. For example, the implementation in [17] performs well on randomly generated problems and uses an amount of time linear in the size of the model.

We have seen that we only need either lead or trail to give PSPACE-hardness. The original decision procedure [17] worked by expanding the ME to add formulas as atomic propositions. Adding each formula was polynomial but interactions between Until and $\leftarrow$ (likewise $S$ and $\rightarrow$) could roughly triple the size of the ME after adding a formula leading to exponential growth of the ME. No other interactions led to growth in the size of the ME. Thus the size of the final ME is the same as the input ME if we exclude both lead and trail, and so we see the following proposition is true.

**Proposition 10.** *If we exclude both lead and trail from MEs (or both until and since from formulas) the ME-checking problem becomes polynomial*

If we exclude "+" from MEs we can see by induction that Until and Since formulas are true either everywhere or nowhere, and so adding formulas as atoms would not grow the size of the ME. It is easy to see the following proposition must be true.

**Proposition 11.** *If we exclude "+" from MEs, the USME-checking problem becomes polynomial.*

Our original PSPACE hardness result relied only on "+" and one of lead or trail. We have now considered excluding both those operators. In each case the model checking problem became polynomial. Note that our PSPACE-hardness result only required one of Until and Since, and if we exclude both, the USME-checking problem becomes polynomial. An obvious question is whether we can get a hardness result using only temporal connectives such as $F$, (and its dual $G$) or its past time equivalent $H$. We will now outline why we cannot do so.

**Theorem 12.** *Model-checking a formula $\phi$ in the fragment $L(F,P)$ of $L(U,S)$ against an ME $\mathcal{I}$ is in P.*

*Proof:* We see that as time advances an $F$ formula can become false, but can never become true. Thus we can choose an ordering $F\beta_1, \ldots, F\beta_n$ of the $n$ $F$-subformulas of $\phi$ such that for all $i$, $j$ with $i \leq j$ we have $F\beta_j \rightarrow F\beta_i$ true throughout $\mathcal{I}$. This means that the set of formulas presatisfied at some point is one of the $n+1$ values $\emptyset$,$\{F\beta_1\}, \ldots, \{F\beta_1, \ldots, F\beta_n\}$. We can likewise order the $m$ $H$-formulas. This means that the set of pre and post-satisfied formulas takes one of $(n+1)(m+1)$ values, and will result in the implementation described in the next section terminating in polynomial time. ■

## V. A Polynomial Space Implementation

In this section we outline how to model check a formula $\phi$ against an ME $\mathcal{I}$ using only a polynomial amount of space. We now define presatisfaction. Intuitively $\mathbf{T} = (T, <, h)$ *presatisfies* $U(\alpha, \beta)$ means $U(\alpha, \beta)$ would be true at a point immediately prior to all points in $T$.

**Definition 13.** We say that a structure $\mathbf{T} = (T, <, h)$ *presatisfies* $U(\alpha, \beta)$ iff there exists a $y \in T$ such that $\mathbf{T}, y \vDash \alpha$ and for each $x \in T$ we have $x < y \rightarrow \mathbf{T}, x \vDash \beta$.

Postsatisfaction is similar but is the mirror image, so informally $\mathbf{T} = (T, <, h)$ *postsatisfies* $S(\alpha, \beta)$ means $S(\alpha, \beta)$ would be true at a point immediately after all points in $T$.

As with [17], we iterate over the formulas from smallest to largest; however, we do not actually modify the ME to add atoms. The algorithm in [17] pushed down formulas as atoms into an ME $\mathcal{I}$ so the result only depended on $\mathcal{I}$ and the formulas of the form $U(\alpha, \beta)$ that were presatisfied after the interval that corresponding to $\mathcal{I}$ and the formulas of the form $S(\alpha, \beta)$ that were postsatisfied before the interval; in this formulation we store the set $\Psi$ of such formulas instead of extending $\mathcal{I}$ with new atoms.

**Definition 14.** Let $\mathbb{S}$ be the set of formulas of the form $S(\alpha, \beta)$ and $\mathbb{U}$ be the set of formulas of the form $U(\alpha, \beta)$. Let $\phi_1, \ldots, \phi_n$ be an enumeration of subformulas of the fixed formula $\phi$ and negations of strict subformulas of $\phi$ such that for any $i, j \in [1, n]$ if $\phi_i$ is a subformula of $\phi_j$ then $i \leq j$. We let $\phi_{\leq j}$ be the set $\{\phi_1, \ldots, \phi_j\}$, and let define equivalence up to $j$ on sets of formulas $S$ and $T$ as follows $S \approx_{\leq j} T$ iff $S \cap \phi_{\leq j} = T \cap \phi_{\leq j}$.

Below we define a recursive function $\mathfrak{A}$. Informally, $\mathfrak{A}(\mathcal{I}, \Phi, \phi) = (\Theta, \Psi)$ is intended to represent the statement: if we have an interval $\mathbf{T}_V$ that corresponds to $\mathcal{I}$, $\Phi$ is the set of formulas in $\mathbb{U}$ presatisfied immediately after $V$ in $\mathbf{T}$ and formulas in $\mathbb{S}$ postsatisfied immediately before $V$ in $\mathbf{T}$ then it must be the case that the set of formulas satisfied within $\mathbf{T}_V$ is $\Theta$ and the set formulas in $\mathbb{U}$ presatisfied immediately before $V$ in $\mathbf{T}$ and formulas in $\mathbb{S}$ postsatisfied immediately after $V$ in $\mathbf{T}$ is $\Psi$. The formula $\phi$ indicates that we are only interested in whether $\phi \in \Theta$ and so we can limit ourselves to subformulas of $\phi$ and their negations. The algorithm works by generating increasing accurate approximations to $(\Theta, \Psi)$ that are accurate up to some subformula $\phi_j$.

**Definition 15.** Let $\mathcal{K}$ be an ME, and $\Phi$ be a set of formulae. We define $\mathfrak{A}(\mathcal{K}, \Phi, \phi)$ to be the pair of sets of formulas $(\Theta, \Psi)$ as follows.

We consider various possible forms of $\mathcal{K}$. The first case we consider is a letter. In the following construction we build increasingly accurate approximations of $(\Theta, \Psi)$: for each $j$ we have $\Theta_j \approx_{\leq j} \Theta$ and $\Psi_j \approx_{\leq j} \Psi$.

Case 0. $\mathcal{K} = \lambda$. Since $\mathcal{K}$ corresponds to the empty pseudo frame it cannot satisfy any formula so we let $\Theta = \emptyset$, likewise

it cannot affect pre or postsatisfaction so $\Psi = \Phi$.

Case 1. $\mathcal{K}$ is a letter:

- $\Theta_0 = \emptyset$, $\Psi_0 = \emptyset$
- for each $i$:
  - We let $\Theta_i$ be a set such that $\Theta_i \setminus \{\phi_i\} = \Theta_{i-1} \setminus \{\phi_i\}$ and: if $\phi_i$ is of the form $\neg\alpha$ then $\phi_i \in \Theta_i$ iff $\alpha \notin \Theta_{i-1}$; if $\phi_i$ is of the form $\alpha \wedge \beta$ then $\phi_i \in \Theta_i$ iff $\alpha \in \Theta_{i-1} \wedge \beta \in \Theta_{i-1}$; if $\phi_i$ is an atom then $\phi_i \in \Theta_i$ iff $\phi_i \in \mathcal{K}$; if $\phi_i \in \mathbb{U} \cup \mathbb{S}$ then $\phi_i \in \Theta_i$ iff $\phi \in \Phi$.
  - $\phi_i \in \Psi$ iff $\phi_i$ is of the form $U(\alpha, \beta)$ or $S(\alpha, \beta)$ and either $\alpha \in \Theta_i$ or $(\beta \in \Theta_i) \wedge (\phi_i \in \Phi)$.
  - We let $\Theta$ be the minimal expansion of $\Theta_n$ such that for each $\phi_i$, we have $\phi_i \in \Theta$ iff $\phi_i \in \Theta_n$ and $\neg\phi_i \in \Theta$ iff $\phi_i \notin \Theta_n$. We add the negations into $\Theta$ so that when we have an ME with multiple letters we can express "$\alpha$ occurs everywhere" as $\neg\alpha \notin \Theta$.

Case 2. $\mathcal{K} = \mathcal{I} + \mathcal{J}$: The construction for $+$ is similar. However, we first build approximations for the subMEs $\mathcal{I}$ and $\mathcal{J}$. To understand the following construction note that the formulas postsatisfied following $\mathcal{J}$ are the same as those following $\mathcal{I}+\mathcal{J}$, that is $\Phi \cap \mathbb{U}$. Likewise the formulas presatisfied prior to $\mathcal{I}$ are the same as those presatisfied prior to $\mathcal{I} + \mathcal{J}$, that is $(\Phi \cap \mathbb{S})$. We iteratively call $\mathfrak{A}$ using approximations to $\Phi$. This works because each time we calculate $\Psi_i^{\mathcal{J}}$ (or $\Psi_i^{\mathcal{I}}$) the approximation we pass is correct up to $\phi_i$ and $\mathfrak{A}$ is structured such that some later formula $\phi_j$ for $j > i$ cannot affect whether $\phi_i$ is included in the input. To see that the approximation is correct up to $\phi_i$ take for example the case where $\phi_i \in \mathbb{S}$ and note that $\left(\Psi_{i-1}^{\mathcal{I}} \cap \mathbb{S}\right) = \left(\Psi_i^{\mathcal{I}} \cap \mathbb{S}\right)$ as $\phi_i \notin \mathbb{U}$.

- let $\Psi_0^{\mathcal{I}} = \Psi_0^{\mathcal{J}} = \emptyset$,
- for each $i$ :
  - If $\phi_i \notin \mathbb{U} \cup \mathbb{S}$ then let:

$$\left(\Theta_i^{\mathcal{J}}, \Psi_i^{\mathcal{J}}\right) = \mathfrak{A}\left(\mathcal{J}, (\Phi \cap \mathbb{U}) \cup \left(\Psi_{i-1}^{\mathcal{I}} \cap \mathbb{S}\right), \phi_i\right)$$
$$\left(\Theta_i^{\mathcal{I}}, \Psi_i^{\mathcal{I}}\right) = \mathfrak{A}\left(\mathcal{I}, (\Phi \cap \mathbb{S}) \cup \left(\Psi_{i-1}^{\mathcal{J}} \cap \mathbb{U}\right), \phi_i\right)$$

  - If $\phi_i \in \mathbb{U}$ then let:

$$\left(\Theta_i^{\mathcal{J}}, \Psi_i^{\mathcal{J}}\right) = \mathfrak{A}\left(\mathcal{J}, (\Phi \cap \mathbb{U}) \cup \left(\Psi_{i-1}^{\mathcal{I}} \cap \mathbb{S}\right), \phi_i\right)$$
$$\left(\Theta_i^{\mathcal{I}}, \Psi_i^{\mathcal{I}}\right) = \mathfrak{A}\left(\mathcal{I}, (\Phi \cap \mathbb{S}) \cup \left(\Psi_i^{\mathcal{J}} \cap \mathbb{U}\right), \phi_i\right)$$

  - If $\phi_i \in \mathbb{S}$ then let:

$$\left(\Theta_i^{\mathcal{I}}, \Psi_i^{\mathcal{I}}\right) = \mathfrak{A}\left(\mathcal{I}, (\Phi \cap \mathbb{S}) \cup \left(\Psi_{i-1}^{\mathcal{J}} \cap \mathbb{U}\right), \phi_i\right)$$
$$\left(\Theta_i^{\mathcal{J}}, \Psi_i^{\mathcal{J}}\right) = \mathfrak{A}\left(\mathcal{J}, (\Phi \cap \mathbb{U}) \cup \left(\Psi_i^{\mathcal{I}} \cap \mathbb{S}\right), \phi_i\right)$$

- $\Theta = \Theta_n^{\mathcal{I}} \cup \Theta_n^{\mathcal{J}}$; $\Psi^{\mathcal{I}} = \Psi_n^{\mathcal{I}}$; $\Psi^{\mathcal{J}} = \Psi_n^{\mathcal{J}}$; $\Psi = \left(\Psi_n^{\mathcal{I}} \cap \mathbb{U}\right) \cup \left(\Psi_n^{\mathcal{J}} \cap \mathbb{S}\right)$

Case 3. $\mathcal{K} = \langle \mathcal{I}_0, \ldots, \mathcal{I}_m \rangle$: Note that a shuffle is a dense mixture and so the fragment of a structure $\mathbf{T}$ corresponding $\mathcal{K}$ that whatever occurs before (or after) a substructure corresponding to $\mathcal{I}_i$ corresponds to $\mathcal{K}$ itself. When determining whether $U(\alpha, \beta)$ is presatisfied before a structure corresponding to $\mathcal{K}$, we see that if $\beta$ is not true throughout $\mathcal{K}$ then we see that, since $\mathcal{K}$ represents a dense mixture, even the smallest prefix of the structure corresponding to $\mathcal{K}$ will have $\neg\beta$ and so $U(\alpha, \beta)$ will not be presatisfied. Clearly $U(\alpha, \beta)$ is presatisfied if $\beta$ is found everywhere in $\mathcal{K}$ and $\alpha$ occurs in $\mathcal{K}$ or we append a structure to $\mathbf{T}$ that presatisfies $U(\alpha, \beta)$.

- $\Psi_0 = \emptyset$, $\Theta_0 = \emptyset$
- for each $i$ :
  - If $\phi_i$ is of the form $U(\alpha, \beta)$ or $S(\alpha, \beta)$ then
    - If $\neg\beta \notin \Theta_{i-1} \wedge (\alpha \in \Theta_{i-1} \vee \phi_i \in \Phi)$ then $\Psi_i = \Psi_{i-1} \cup \{\phi_i\}$
    - else $\Psi_i = \Psi_{i-1} \setminus \{\phi_i\}$.
  - for every integer $j \in [0, m]$ we let:

$$\left(\Theta_i^j, \Psi_i^j\right) = \mathfrak{A}\left(\mathcal{I}_j, (\Psi_i \cap \mathbb{S}) \cup (\Psi_i \cap \mathbb{U}), \phi_i\right)$$

  - $\Theta_i = \bigcup_{0 \leq j \leq m} \Theta_i^j$
- $\Theta = \Theta_n$, $\Psi = \Psi_n$.

The original model checking procedure from [17] unwinds $\overleftarrow{\mathcal{I}}$ into something of the form $\overleftarrow{\mathcal{I}_0} + \mathcal{I}_1 + \cdots + \mathcal{I}_m$. The intuition behind integer $j$ below is that it indexes $\mathcal{I}_j$ in this expansion. We handle $\leftarrow$ as an expansion of $+$ so its definition below mostly follows the approach for $\mathcal{I}+\mathcal{J}$. However, note that $\leftarrow$ has a limit point on the left, and so at the left hand the since operator behaves similarly to the shuffle.

Case 4. $\mathcal{K} = \overleftarrow{\mathcal{I}}$:

- let $m$ be the number of instances of $U$ in $\phi$.
- let $\Psi_0^0 = \Psi_0^m = \emptyset$,
- for each $i$ :
  - let $\Psi_i^{m+1} = \Phi \cap \mathbb{U}$.
  - If $\phi_i \notin \mathbb{S}$ then let $\Psi_i^{-1} = \Psi_{i-1}^0$ and for each $j \in [0, m]$ from $m$ down to 0 let:
  - If $\phi_i \in \mathbb{S}$ is of the form $S(\alpha, \beta)$ then
    - If $\neg\beta \notin \Theta_{i-1}^0 \wedge \left(\alpha \in \Theta_{i-1}^0 \vee \phi_i \in \Phi\right)$ then $\Psi_i^{-1} = \Psi_{i-1}^0 \cup \{\phi_i\}$ else $\Psi_i^{-1} = \Psi_{i-1}^0 \setminus \{\phi_i\}$.
    - for each $j$ from 0 up to $m$ let $\left(\Theta_i^j, \Psi_i^j\right) = \mathfrak{A}\left(\mathcal{I}, \left(\Psi_i^{j-1} \cap \mathbb{S}\right) \cup \left(\Psi_{i-1}^{j+1} \cap \mathbb{U}\right), \phi_i\right)$
- $\Theta = \bigcup_{0 \leq j \leq m} \Theta_n^j$, $\Psi = \left(\Psi_n^0 \cap \mathbb{U}\right) \cup \left(\Psi_n^m \cap \mathbb{S}\right)$

Case 5. $\mathcal{K} = \overrightarrow{\mathcal{I}}$: Broadly similar to above, but the mirror image. For example, the intuition behind integer $j$ is now that it indexes $\mathcal{I}_j$ of an expansion $\mathcal{I}_m + \cdots + \mathcal{I}_1 + \overrightarrow{\mathcal{I}_0}$.

- let $m$ be the number of instances of $S$ in $\phi$.
- let $\Psi_0^0 = \Psi_0^m = \emptyset$,
- for each $i$ :
  - let $\Psi_i^{m+1} = \Phi \cap \mathbb{S}$.
  - If $\phi_i \notin \mathbb{U}$ then let $\Psi_i^{-1} = \Psi_{i-1}^0$ and for each $j \in [0, m]$ from $m$ down to 0 let $\left(\Theta_i^j, \Psi_i^j\right) = \mathfrak{A}\left(\mathcal{I}, \left(\Psi_{i-1}^{j-1} \cap \mathbb{U}\right) \cup \left(\Psi_i^{j+1} \cap \mathbb{S}\right), \phi_i\right)$
  - If $\phi_i \in \mathbb{U}$ then
    - If $\neg\beta \notin \Theta_{i-1}^0 \wedge \left(\alpha \in \Theta_{i-1}^0 \vee \phi_i \in \Phi\right)$ then $\Psi_i^{-1} = \Psi_{i-1} \cup \{\phi_i\}$ else $\Psi_i^{-1} = \Psi_{i-1} \setminus \{\phi_i\}$.
    - for each $j$ from 0 up to $m$ let $\left(\Theta_i^j, \Psi_i^j\right) = \mathfrak{A}\left(\mathcal{J}, \left(\Psi_i^{j-1} \cap \mathbb{U}\right) \cup \left(\Psi_{i-1}^{j+1} \cap \mathbb{S}\right), \phi_i\right)$

- $\Theta = \bigcup_{0 \leq j \leq m} \Theta_n^j$, $\Psi = \left(\Psi_n^0 \cap \mathbb{S}\right) \cup \left(\Psi_n^m \cap \mathbb{U}\right)$

Note that we could eliminate the case where $\mathcal{K} = \lambda$, as the algebraic equivalences in [20] could be used to reduce the input ME $\mathcal{I}$ to an equivalent ME without $\lambda$ occurring (or the trivial ME that contains only $\lambda$, for which the model checking problem is trivial as it has no points and thus cannot satisfy any formula).

**Definition 16.** The "Polynomial Space Model Checking Procedure" is as follows: given an ME $\mathcal{I}$ and formula $\phi$ we return "true" iff $\phi \in \Theta$ where $(\Theta, \Psi) = \mathfrak{A}(\mathcal{I}, \Phi, \phi)$.

As this model checker is a reformulation of the model-checker in [17], the proof of correctness is similar. We now show that the USME-checking problem is in PSPACE.

**Theorem 17.** *We can solve the model checking problem in* $\mathcal{O}\left(|\mathcal{I}|^2 |\phi|\right)$ *space.*

*Proof:* Consider an obvious implementation of $\mathfrak{A}$ as a recursive function. We can discard $\Phi_i$ and $\Psi_i$ after $\Phi_{i+1}$ and $\Psi_{i+1}$ have been computed. Each set of formulas can be stored as an array of bits of length $n$, with the $i^{\text{th}}$ bit representing whether $\phi_i$ is in the set and so the amount of space required to store an array of formulas is $|\phi|$. We see that we only need to store a number of such arrays of order $|\mathcal{I}|$ per recursion, and the recursion depth is at most $|\mathcal{I}|$. Hence we can solve the model checking problem in space of order $|\mathcal{I}|^2 |\phi|$. ■

## VI. Conclusion

In this paper we have shown that the model checking problem for US/L formulas over Model Expressions is PSPACE-Complete. We presented a simple recursive procedure which, unlike [17], does not construct an ME or store interim MEs and thus only requires polynomial space with regard to the total length of input $|\mathcal{I}| + |\phi|$. We have also proven a hardness result by a reduction from QBF satisfiability solving, giving us PSPACE-completeness. Interestingly, it is known that the USME-checking problem is polynomial if either the length of the input formula or input model is fixed [17] (but clearly not if neither are fixed, unless P=PSPACE). Our PSPACE-hardness result is quite robust in the sense that it depends only upon $+$, $\leftarrow$, $U$ and the classical operators; removing further operators will result in a polynomial problem. Despite their semantic complexity, shuffles do not contribute to the computational complexity of the problem.

The complexity is quite similar to that of LTL which is also PSPACE-complete for model checking against both Kripke structures [2] and compressed paths [19]. Note that while LTL is simpler than US/L, the Kripke models allow the LTL model checker to non-deterministically pick loops through the Kripke structure, an issue that the USME (and compressed path) checker does not need to handle. Also, MEs do not have disjunctions, and so MEs represent just one possible trace of a system, whereas the traditional model checking problem of LTL has the models represent systems which can have many possible traces. The difficulty in model checking LTL against Kripke structures primarily results from these disjunctions while in the model checking MEs the difficulty comes from the operators $\leftarrow$ and $\rightarrow$ used to represent Zeno properties, and is related to the difficulty with compressed paths.

It is interesting that the complexity of model-checking in this context matches that of synthesis and satisfiability. Synthesis [12] and satisfiability [16] checking $L(U, S)$ are also PSPACE-complete over the reals, and over general linear time as well [23].

## References


[1] H. Barringer, R. Kuiper, and A. Pnueli, "A really abstract concurrent model and its temporal logic," in *Proceedings of 13th ACM Symposium on Principles of Programming Languages*, 1986, pp. 173 – 183.

[2] A. Sistla and E. Clarke, "Complexity of propositional linear temporal logics," *J. ACM*, vol. 32, pp. 733–749, 1985.

[3] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell, "On expressiveness and complexity in real-time model checking," in *Proc. of the 35th Intl. Colloq. on Automata, Languages and Programming, Part II*, ser. ICALP '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 124–135.

[4] R. Alur, C. Courcoubetis, and D. L. Dill, "Model-checking for real-time systems," in *LICS*. IEEE Computer Society, 1990, pp. 414–425.

[5] R. Alur and D. Dill, "Automata for modeling real-time systems," in *Automata, Languages and Programming*, ser. LNCS, M. Paterson, Ed. Springer Berlin / Heidelberg, 1990, vol. 443, pp. 322–335.

[6] N. Huggett, "Zenoś paradoxes," in *The Stanford Encyclopedia of Philosophy*, winter 2010 ed., E. N. Zalta, Ed., 2010.

[7] G. W. Wornell and A. V. Oppenheim, "Estimation of fractal signals from noisy measurements using wavelets," *Signal Processing, IEEE Transactions on*, vol. 40, no. 3, pp. 611–623, 1992.

[8] P. J. Mosterman, "Hybrid dynamic systems: mode transition behavior in hybrid dynamic systems," in *Winter Simulation Conference*, S. E. Chick *et al* Eds. ACM, 2003, pp. 623–631.

[9] H. Kamp, "Tense logic and the theory of linear order," Ph.D. dissertation, University of California, Los Angeles, 1968.

[10] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time systems*, vol. 2, no. 4, pp. 255–299, 1990.

[11] J. Ouaknine and J. Worrell, "Some recent results in metric temporal logic," in *Formal Modeling and Analysis of Timed Systems*. Springer, 2008, pp. 1–13.

[12] T. French, J. C. McCabe-Dansted, and M. Reynolds, "Synthesis for temporal logic over the reals," in *Advances in Modal Logic*, T. Bolander, T. Braüner, S. Ghilardi, and L. S. Moss, Eds. College Publications, 2012, pp. 217–238.

[13] H. Läuchli and J. Leonard, "On the elementary theory of linear order," *Fundamenta Mathematicae*, vol. 59, pp. 109–116, 1966.

[14] J. P. Burgess and Y. Gurevich, "The decision problem for linear temporal logic," *Notre Dame J. Formal Logic*, vol. 26, no. 2, pp. 115–128, 1985.

[15] M. Reynolds, "Continuous temporal models," in *Australian Joint Conference on Artificial Intelligence*, ser. Lecture Notes in Computer Science, M. Stumptner *et al*, Eds., vol. 2256. Springer, 2001, pp. 414–425.

[16] ——, "The complexity of the temporal logic over the reals," *Annals of Pure and Applied Logic*, vol. 161, no. 8, pp. 1063–1096, 2010.

[17] T. French, J. McCabe-Dansted, and M. Reynolds, "Model checking general linear temporal logic," 2013, accepted to appear in TABLEAU.

[18] J. C. McCabe-Dansted, "Model checker for general linear time (online applet and data)," 2012, http://www.csse.uwa.edu.au/~mark/research/Online/mechecker.html.

[19] N. Markey and P. Schnoebelen, "Model checking a path," in *CONCUR 2003-concurrency theory*. Springer, 2003, pp. 251–265.

[20] T. French, J. McCabe-Dansted, and M. Reynolds, "An algebraic system for linear orders," 2013, accepted to appear in TIME 2013.

[21] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the third annual ACM symposium on Theory of computing*, ser. STOC '71. New York, NY, USA: ACM, 1971, pp. 151–158.

[22] L. J. Stockmeyer and A. R. Meyer, "Word problems requiring exponential time(preliminary report)," in *Proceedings of the fifth annual ACM symposium on Theory of computing*, ser. STOC '73. New York, NY, USA: ACM, 1973, pp. 1–9.

[23] M. Reynolds, "The complexity of temporal logics over linear time," *Journal of Studies in Logic*, vol. 3, pp. 19–50, 2010.