A combined approach to temporal logics for time granularity

Massimo Franceschet[1] and Angelo Montanari[2]

[1]Department of Sciences, University of Chieti-Pescara, Italy
E-mail: `francesc@sci.unich.it`
[2]Department of Mathematics and Computer Science,
University of Udine, Italy
E-mail: `montana@dimi.uniud.it`

Abstract

The ability of providing and relating temporal representations at different 'grain levels' of the same reality is an important research theme in Computer Science and Artificial Intelligence and a major requirement for many applications, including formal specifications, temporal databases, data mining, problem solving, and natural language understanding. In particular, the addition of a granularity notion to a temporal logic makes it possible to specify in a concise way reactive systems whose behaviour can be naturally modeled with respect to a (possibly infinite) set of differently-grained temporal domains. In this paper, we provide the monadic second-order theory of upward unbounded layered structures [11], which consist of a finest domain and an infinite number of coarser and coarser domains, with an expressively complete and elementarily decidable temporal logic counterpart. We obtain our result in two steps. First, we define a new class of combined automata, called temporalized automata, which can be proved to be the automata-theoretic counterpart of temporalized logics, and show that relevant properties, such as closure under Boolean operations, decidability, and expressive equivalence with respect to temporal logics, transfer from component automata to temporalized ones. Then, we exploit the correspondence between temporalized logics and automata to reduce the task of finding a temporal logic counterpart of the monadic second-order theory of upward unbounded layered structures to the easier one of finding a temporalized automata counterpart of them.

1 Introduction

Any time granularity can be viewed as the partitioning of a temporal domain in groups of elements, where each group is perceived as an indivisible unit (a granule). The description of a fact can use these granules to provide it with a temporal qualification, at the appropriate abstraction level. However, adding the concept of time granularity to a formalism does not merely mean that one can use different temporal units to represent temporal quantities in a unique flat model, but it involves semantic issues related to the problem of assigning a proper meaning to the association of statements with the different temporal domains of a layered model and of switching from one domain to a coarser/finer one [9]. Montanari et al. investigated the monadic second-order theory of k-refinable *downward unbounded layered structures* (DULSs, for short), which are infinitely refinable structures consisting of a coarsest domain and an infinite number of finer and finer domains, and the monadic second-order theory of k-refinable *upward unbounded layered structures* (UULSs), which consists of a finest domain and an infinite number of coarser and coarser domains [11].

The original motivation of our research was the design of a temporal logic, embedding a notion of time granularity, suitable for the specification of complex concurrent systems whose components evolve according to different time units (granular reactive systems). However, there are significant similarities between the problems we encountered in studying time granularity, and those addressed by current research on combining logics, theories, and structures [6]. Furthermore, we recently established interesting connections between temporal logics for time granularity and automata theory that suggests a complementary point of view: besides an important feature of a representation language, time granularity can be viewed as a a formal setting to investigate the definability of meaningful timing properties that cannot be captured by using flat temporal logics. For instance, temporal logics over k-refinable UULSs allow one to express conditions like "P holds at all time points k^i, for all natural numbers i, of a given temporal domain", which cannot be expressed by using either propositional or quantified temporal logics over a flat temporal domain, while temporal logics over DULSs allow one to constrain a given property to hold true 'densely' over a given time interval [11]. The relationships between the theories of time granularity and the standard theories of timed state sequences, underlying real-time logics, have been explored in [12], where Montanari et al. showed that the latter ones can actually be embedded into the former ones. Finally, there exists a natural link between structures and theories of time granularity and those developed for representing and reasoning about time intervals [13]. A complete and up-to-date illustration of all these connections can be found in [3].

The nonelementary decidability of the theories of DULSs and UULSs was proved by means of nontrivial encodings into Rabin and systolic tree automata, respectively. In this paper, we provide the monadic second-order theory of UULSs with an expressively complete and elementarily decidable temporal logic counterpart. Finding the temporal logic counterpart of monadic theories is a difficult task, involving a non-elementary blow up in the length of formulas. Ehrenfeucht games have been successfully exploited to deal with such a correspondence problem for first-order monadic theories [8] and well-behaved fragments of second-order ones, e.g. the path fragment of the monadic second-order theory of infinite binary trees [7]. As for the theories of time granularity, by means of suitable applications of Ehrenfeucht games, we obtained an expressively complete and elementarily decidable combined temporal logic counterpart of the path fragment of the monadic second-order theory of DULSs [4], while Montanari et al. extended Kamp's theorem to deal with the first-order fragment of the theory of UULSs [10]. Unfortunately, these techniques produce rather involved proofs and do not naturally lift to the full second-order case. In this paper, we follow a different approach. Instead of trying to establish a direct correspondence between monadic second-order theories for time granularity and temporal logics, we connect them via automata. Firstly, we define a new class of combined automata, called temporalized automata, which can be proved to be the automata-theoretic counterpart of temporalized logics [2], and show that relevant properties, such as closure under Boolean operations, decidability, and expressive equivalence with respect to temporal logics, transfer from component automata to temporalized ones. Then, on the basis of the established correspondence between temporalized logics and automata, we reduce the task of finding a temporal logic counterpart of the monadic second-order theory of UULSs to the easier one of finding a temporalized automata counterpart of them. The mapping of monadic formulas into automata (the difficult direction) can indeed greatly benefit from automata closure properties. In [5], we exploit a similar technique to solve the (simpler) problem of providing the monadic second-order theory of DULSs with a temporal logic counterpart.

2 Temporalized logics and automata

In this section we recall the definition of temporalization and we define temporalized automata. Moreover, we prove the equivalence of temporalized automata and temporalized logics. We will consider *temporal logics* over a finite set of propositional letters $\mathcal{P} = \{P, Q, \ldots\}$. Given a temporal logic $\mathbf{T}$, we use $\mathcal{L}_{\mathbf{T}}$ to denote the language of $\mathbf{T}$ and $OP(\mathbf{T})$ to denote the set of temporal operators of $\mathbf{T}$. Moreover, let $\mathsf{K}_{\mathbf{T}}$ be the set of structures (models) over which $\mathbf{T}$ is interpreted and, given $\varphi \in \mathcal{L}_{\mathbf{T}}$, let $\mathcal{M}(\varphi) \subseteq \mathsf{K}_{\mathbf{T}}$ be the set of models of φ. Examples of temporal logics we will consider are *Propositional Linear Temporal Logic* (PLTL) and *Directed Computational Tree Logic* ($\mathrm{CTL}^*_{\mathrm{k}}$), and their quantified versions QLTL and $\mathrm{QCTL}^*_{\mathrm{k}}$ [1]. Quantified versions of propositional temporal logics add to the language quantified formulas of the form $\exists Q \varphi$, where φ is a formula and Q is a proposition letter appearing free in φ. We will also consider the existential fragment EQLTL (resp. $\mathrm{EQCTL}^*_{\mathrm{k}}$) of QLTL (resp. $\mathrm{QCTL}^*_{\mathrm{k}}$) consisting of formulas of the form $\exists Q_1 \ldots \exists Q_n \varphi$, where φ is a PLTL-formula (resp. $\mathrm{CTL}^*_{\mathrm{k}}$).

Temporalization is a simple mode of combining logics in which one logic is embedded into a temporal logic [2]. We consider the embedding of a temporal logic $\mathbf{T_2}$ into a temporal logic $\mathbf{T_1}$. We partition the set of $\mathbf{T_2}$-formulas into *Boolean combinations* $BC_{\mathbf{T_2}}$ and *monolithic formulas* $ML_{\mathbf{T_2}}$: φ belongs to $BC_{\mathbf{T_2}}$ if its outermost operator is a Boolean connective; otherwise, it belongs to $ML_{\mathbf{T_2}}$. We assume that $OP(\mathbf{T_1}) \cap OP(\mathbf{T_2}) = \emptyset$.

Definition 2.1 (*Temporalization – Syntax*)
The combined language $\mathcal{L}_{\mathbf{T_1}(\mathbf{T_2})}$ *of the* temporalization $\mathbf{T_1}(\mathbf{T_2})$ *of* $\mathbf{T_2}$ *by means of* $\mathbf{T_1}$ *over the set of proposition letters* $\mathcal{P}$ *is obtained by replacing the atomic formation rule of* $\mathcal{L}_{\mathbf{T_1}}$*, i.e., the rule stating that every proposition letter is a formula, by the following rule: every monolithic formula* $\varphi \in ML_{\mathbf{T_2}}$ *is an* $\mathcal{L}_{\mathbf{T_1}(\mathbf{T_2})}$*-formula.* $\square$

A *model* for $\mathbf{T_1}(\mathbf{T_2})$ is a triple (W, R, g), where (W, R) is a frame for $\mathbf{T_1}$ and $g : W \to \mathsf{K}_{\mathbf{T_2}}$ a total function mapping worlds in W into models for $\mathbf{T_2}$.

Definition 2.2 (*Temporalization – Semantics*)
Given a model $\mathcal{M} = (W, \mathcal{R}, g)$ *and a state* $w \in W$*, the semantics of the temporalized logic* $\mathbf{T_1}(\mathbf{T_2})$ *is obtained by replacing the semantic clause for proposition letters of* $\mathbf{T_1}$ *by the following clause:* $\mathcal{M}, w \models_{\mathbf{T_1}(\mathbf{T_2})} \varphi$ *iff* $g(w) \models_{\mathbf{T_2}} \varphi$*, for every monolithic formula* $\varphi \in ML_{\mathbf{T_2}}$*.*
$\square$

In the following, we provide temporalized logics with an operational counterpart in terms of automata. For the sake of simplicity, we first define automata and prove results over sequence structures; then, we generalize definitions and results to tree structures (we believe that our machinery can actually be extended to cope with more general structures, such as arbitrary graphs). Let $\Sigma = \{a, b, \ldots\}$ be a finite alphabet. We denote by $\mathcal{S}(\Sigma)$ be the set of Σ-labeled infinite sequences $(\mathbb{N}, <, V)$, with $V : \mathbb{N} \to \Sigma$. We will use the following general definition of sequence automata.

Definition 2.3 (*Sequence automata*)
A sequence automaton *A over Σ consists of (i) a Labeled Transition System $(Q, q_0, \Delta, M, \Omega)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation, Ω is a finite alphabet, and $M \subseteq Q \times \Omega$ is a labeling of states, and (ii) an acceptance condition AC. Given a Σ-labeled infinite sequence $w = (\mathbb{N}, <, V)$, a* run *of A on w is a function $\sigma : \mathbb{N} \rightarrow Q$ such that $\sigma(0) = q_0$ and $(\sigma(i), V(i), \sigma(i+1)) \in \Delta$, for every $i \geq 0$. The automaton A accepts w if there is a run σ of A on w such that $AC(\sigma)$, i.e., the acceptance condition holds on σ. The language accepted by A, denoted by $\mathcal{L}(A)$, is the set of Σ-labeled infinite sequences accepted by A.* □

A class of sequence automata $\mathcal{A}$ is a set of automata that share the acceptance condition AC. (We do not explicitly specify the acceptance condition for sequence automata since all the results do not rest on any particular acceptance condition.) An example of sequence automata class is the class of Büchi sequence automata.

Example 2.4 (*Büchi sequence automata*)
A Büchi sequence automaton is a sequence automaton $A = (Q, q_0, \Delta, M, \Omega)$ such that $\Omega = \{\texttt{final}\}$. We call final a state q such that $(q, \texttt{final}) \in M$. The acceptance condition for A states that A accepts a Σ-labelled infinite sequence x iff there is a run σ of A over x such that some final state occurs infinitely often in σ. □

We are now ready to introduce temporalized automata. Let us assume that $\mathcal{A}_2$ is a class of sequence automata, over the finite alphabet Σ, accepting subsets of $\mathcal{S}(\Sigma)$; moreover, let $\Gamma(\Sigma)$ be a finite alphabet whose symbols $A, B, \ldots$ represent automata in $\mathcal{A}_2$; finally, let $\mathcal{A}_1$ be a class of sequence automata, over the finite alphabet $\Gamma(\Sigma)$, accepting subsets of $\mathcal{S}(\Gamma(\Sigma))$. Given $\mathcal{A}_1$ and $\mathcal{A}_2$ as above, we define a class of temporalized automata $\mathcal{A}_1(\mathcal{A}_2)$ that combines the two component automata classes in a suitable way. Let $\mathcal{S}(\mathcal{S}(\Sigma))$ be the set of infinite sequences of Σ-labeled infinite sequences, that is, temporalized models $(\mathbb{N}, <, g)$ where $g : \mathbb{N} \rightarrow \mathcal{S}(\Sigma)$ is a total function mapping elements of $\mathbb{N}$ into Σ-labeled infinite sequences in $\mathcal{S}(\Sigma)$. Automata in the combined class $\mathcal{A}_1(\mathcal{A}_2)$ accept in $\mathcal{S}(\mathcal{S}(\Sigma))$.

Definition 2.5 (*Temporalized automata*)
A temporalized automaton *A over $\Gamma(\Sigma)$ is a quintuple $(Q, q_0, \Delta, M, \Omega)$ as for sequence automata (Definition 2.3). The* combined acceptance condition *for A is the following. Given $\alpha = (\mathbb{N}, <, g) \in \mathcal{S}(\mathcal{S}(\Sigma))$, a* run *of A on α is a function $\sigma : \mathbb{N} \rightarrow Q$ such that $\sigma(0) = q_0$ and, for every $i \geq 0$, $(\sigma(i), B, \sigma(i+1)) \in \Delta$ for some $B \in \Gamma(\Sigma)$ such that $g(i) \in \mathcal{L}(B)$. The automaton A accepts α if there is a run σ of A on α such that $AC(\sigma)$, where AC is the acceptance condition of $\mathcal{A}_1$-automata. The language accepted by A, denoted by $\mathcal{L}(A)$, is the subset of $\mathcal{S}(\mathcal{S}(\Sigma))$ accepted by A. We denote by $\mathcal{A}_1(\mathcal{A}_2)$ the class of temporalized automata.* □

Given a temporalized automaton $A \in \mathcal{A}_1(\mathcal{A}_2)$, we denote by $A^{\uparrow}$ the automaton in $\mathcal{A}_1$ with the same labeling transition system as A and with the acceptance condition of $\mathcal{A}_1$. Hence, while A accepts in $\mathcal{S}(\mathcal{S}(\Sigma))$, its *abstraction* $A^{\uparrow}$ recognizes in $\mathcal{S}(\Gamma(\Sigma))$. Moreover, given an automaton $A \in \mathcal{A}_1$, we denote by $A^{\downarrow}$ the automaton in $\mathcal{A}_1(\mathcal{A}_2)$ with the same labelling transition system as A and with the combined acceptance condition of $\mathcal{A}_1(\mathcal{A}_2)$. Thus, while A accepts in $\mathcal{S}(\Gamma(\Sigma))$, its *concrete* counterpart $A^{\downarrow}$ recognizes in $\mathcal{S}(\mathcal{S}(\Sigma))$. With the aid of these notations, the combined acceptance condition for temporalized automata can be rewritten as follows. Let $\alpha = (\mathbb{N}, <, g) \in \mathcal{S}(\mathcal{S}(\Sigma))$. We say that a temporalized automaton A accepts α if and only if there is $\beta = (\mathbb{N}, <, V) \in \mathcal{S}(\Gamma(\Sigma))$ such that $\beta \in \mathcal{L}(A^{\uparrow})$ and, for every $i \in \mathbb{N}$, $g(i) \in \mathcal{L}(V(i))$. We will often use this alternative, but equivalent, definition of acceptance condition for temporalized automata.

We define the *transfer problem* for temporalized automata as follows: assuming that automata classes $\mathcal{A}_1$ and $\mathcal{A}_2$ enjoy some property, does $\mathcal{A}_1(\mathcal{A}_2)$ enjoy the same property? In the following we assume that $\mathbf{T_1}$ and $\mathbf{T_2}$ are temporal logics interpreted over $\mathcal{S}(\Sigma)$. Hence, the temporalized logic $\mathbf{T_1}(\mathbf{T_2})$ is interpreted over $\mathcal{S}(\mathcal{S}(\Sigma))$. We say that a class of automata $\mathcal{A}$ is *embedable* into a temporal logic $\mathbf{T}$, denoted $\mathcal{A} \rightarrow \mathbf{T}$, if there is an *effective* translation τ of $\mathcal{A}$-automata into $\mathbf{T}$-formulas such that, for every $\mathcal{A}$-automaton A, $\mathcal{L}(A) = \mathcal{M}(\tau(A))$. Similarly, we define $\mathbf{T} \rightarrow \mathcal{A}$. Finally, we say that $\mathcal{A}$ is *expressively equivalent* to $\mathbf{T}$, written $\mathcal{A} \rightleftarrows \mathbf{T}$, if both $\mathbf{T} \rightarrow \mathcal{A}$ and $\mathcal{A} \rightarrow \mathbf{T}$. We have the following theorem and corollary.

Theorem 2.6 (*Transfer theorem*)
Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be sequence automata classes.

1. Closure *under Boolean operations (union, intersection, and complementation) transfers to temporalized automata: if $\mathcal{A}_1$ and $\mathcal{A}_2$ are (effectively) closed under Boolean operations, then $\mathcal{A}_1(\mathcal{A}_2)$ is (effectively) closed under Boolean operations.*

2. Decidability *(of the emptiness problem) transfers to temporalized automata: if $\mathcal{A}_1$ and $\mathcal{A}_2$ are decidable, then $\mathcal{A}_1(\mathcal{A}_2)$ is decidable.*

3. Expressive equivalence *w.r.t. temporal logic transfers to temporalized automata: given $\mathcal{A}_2$ closed under Boolean operations, if $\mathcal{A}_1 \leftrightarrows \mathbf{T_1}$ and $\mathcal{A}_2 \leftrightarrows \mathbf{T_2}$, then $\mathcal{A}_1(\mathcal{A}_2) \leftrightarrows \mathbf{T_1}(\mathbf{T_2})$.*

Corollary 2.7 *If $\mathbf{T_1} \rightarrow \mathcal{A}_1$, $\mathbf{T_2} \rightarrow \mathcal{A}_2$, and both $\mathcal{A}_1$ and $\mathcal{A}_2$ are decidable, then $\mathbf{T_1}(\mathbf{T_2})$ is decidable.*

We conclude this section by introducing a more general notion of automata: tree automata. Let $k \geq 2$ and T_k be the set $\{0, \dots, k-1\}^*$. Given $x, y \in T_k$, we say that x is a *prefix* of y, denoted $x <_P y$, if $xw = y$, for some nonempty $w \in T_k$. Note that the prefix relation $<_P$ is a partial ordering over T_k. A set $D \subseteq T_k$ is an *tree domain* if:

1. D is *prefix closed*, that is, $x \in D$ and $y <_P x$ implies $y \in D$, for every $x, y \in T_k$;

2. for every $x \in T_k$, if $xi \in D$, then $xj \in D$ for every $0 \leq j < i \leq k-1$.

A *tree* is a pair $(W, <_P)$ where W is a k-ary tree domain. We denote by $\mathcal{T}_k(\Sigma)$ the set of Σ*-labeled k-ary trees* $(W, <_P, V)$ such that $V : W \rightarrow \Sigma$.

Definition 2.8 (*Tree automata*)
A tree automaton *A over Σ consists of (i) a Labeled Transition System $(Q, q_0, \Delta, M, \Omega)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, $\Delta \subseteq Q \times \Sigma \times (\bigcup_{i=1}^{k} Q^i)$ is a transition relation, Ω is a finite alphabet, and $M \subseteq Q \times \Omega$ is a labeling of states, and (ii) an acceptance condition AC. Given a Σ-labeled tree $t = (W, <_P, V)$, a* run *of A on t is function $\sigma : W \rightarrow Q$ such that $\sigma(\epsilon) = q_0$ and, for every $x \in W$ with $m \leq k$ sons $x0, \dots x(m-1)$, we*

have $(\sigma(x), V(x), \sigma(x0), \ldots, \sigma(x(m-1))) \in \Delta$. The automaton A accepts t if there is a run σ of A on t such that $AC(\sigma)$, i.e., the acceptance condition holds on σ. The language accepted by A, denoted by $\mathcal{L}(A)$, is the set of Σ-labeled k-ary trees accepted by A. $\square$

An example of tree automata class is the class of finite tree automata.

Example 2.9 (*Finite tree automata*)
A (top-down) finite tree automaton is a tree automaton $A = (Q, q_0, \Delta, M, \Omega)$ such that $\Omega = \{\texttt{final}\}$. We call final a state q such that $(q, \texttt{final}) \in M$. The automaton A accepts finite trees. *The acceptance condition for A states that A accepts a finite tree t iff there is a run σ of A over t such that all the leaves of σ are labelled with final states.* $\square$

Theorem 2.6 and Corollary 2.7 immediately generalize to tree automata. Corollary 2.7 allows us to prove the decidability of many temporalized logics. For instance, it is known that QLTL and PLTL over infinite sequences can be embedded into Büchi sequence automata, and that Büchi sequence automata are decidable. Moreover, $\mathrm{QCTL}^*_\mathrm{k}$ and $\mathrm{CTL}^*_\mathrm{k}$ over finite trees can be embedded into finite tree automata, and finite tree automata are decidable. It follows that any temporalized logic $\mathbf{T_1}(\mathbf{T_2})$, with $\mathbf{T_1}, \mathbf{T_2} \in \{\mathrm{PLTL}, \mathrm{QLTL}, \mathrm{QCTL}^*_\mathrm{k}, \mathrm{CTL}^*_\mathrm{k}\}$, is decidable. As a matter of fact, the decidability of PLTL(PLTL) was already proved in [2] following a different approach.

3 Temporalized logics and automata for time granularity

In the following, we use temporalized automata to find a combined temporal logic counterpart of the monadic second-order theory of UULSs. The result rests on an alternative view of UULSs as infinite sequences of finite *increasing* k-ary trees. We define a suitable combination of Büchi and finite tree automata and use it to obtain a combined temporal logic which is both elementarily decidable and expressively complete with respect to the monadic second-order theory of UULSs. In fact, the combined model we use to encode an UULS specializes that of temporalization since the innermost submodels are *not* independent from the outermost model. The monadic second-order language for time granularity $\mathrm{MSO}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$ is defined as follows.

Definition 3.1 (*Monadic second-order language*)
Let $\mathrm{MSO}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$ *be the second-order language with equality built up as follows: (i)* atomic formulas *are of the forms $x = y$, $x < y$, $\downarrow_i(x) = y$, $x \in X$ and $x \in P_a$, where $0 \leq i \leq k-1$, x, y are individual variables, X is a set variable, and $a \in \Sigma$; (ii)* formulas *are built up from atomic formulas by means of the Boolean connectives $\neg$ and $\wedge$, and the quantifier $\exists$ ranging over both individual and set variables.* $\square$

We interpret $\mathrm{MSO}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$ over UULSs. For all $i \geq 0$, let $T^i = \{j_i \mid j \geq 0\}$. A Σ-labeled k-refinable UULS is a tuple $\langle \bigcup_{i \geq 0} T^i, (\downarrow_i)_{i=0}^{k-1}, <, (P_a)_{a \in \Sigma} \rangle$, that intuitively represents a k-ary infinite tree generated from the leaves (cf. Figure 1). The sets in $\{T^i\}_{i \geq 0}$ represent the layers of the tree, $\downarrow_i$ is a projection function such that $\downarrow_i(a_0) = \perp$ and $\downarrow_i(a_b) = c_d$ if and only if $b > 0$, $b = d + 1$ and $c = a \cdot k + i$, with $i = 0, \ldots, k-1$, $<$ is the total ordering of $\bigcup_{i \geq 0} T^i$ given by the *inorder* (left-root-right) visit of the nodes, and, for all $a \in \Sigma$, P_a is the

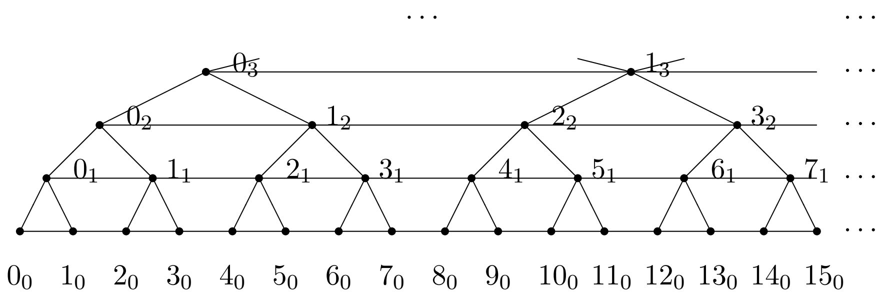

Figure 1: A 2-refinable UULS.

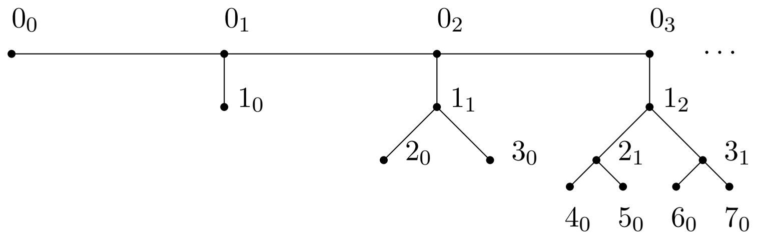

Figure 2: Mapping an UULS into an increasing tree sequence.

set of points in $\bigcup_{i \geq 0} T^i$ labeled with symbol a. Given a formula $\varphi \in \text{MSO}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$, we denote by $\mathcal{M}(\varphi)$ the set of models of φ.

We proceed by giving an alternative characterization of UULSs in terms of tree sequences. To this end, we need to introduce the notions of hanger tree and of increasing tree sequence. A *k-ary hanger tree* is a complete finite tree whose root has exactly $k-1$ sons, and each of them is the root of a finite and complete k-ary tree. A *k-ary Increasing Tree Sequence* (ITS) is an infinite sequence of k-ary hanger trees such that the i-th tree of the sequence has height i (cf. Figure 2). An ITS can be represented as a combined model $(\mathbb{N}, <, g)$ such that, for every $i \in \mathbb{N}$, $g(i)$ is a k-ary hanger tree of height i. Notice that the combined model $(\mathbb{N}, <, g)$ for an ITS is *not* a temporalized model, since the height of the tree $g(i)$ depends on i. Let $ITS_k(\Sigma)$ be the set of all Σ-labeled k-ary ITSs. We show that a Σ-labeled UULS corresponds to a Σ-labeled ITS and vice versa. Intuitively, given an UULS t, the tree sequence is obtained by taking the trees rooted at the points of the leftmost path $0_0, 0_1, \ldots$ of t, and deleting from each tree the subtree rooted at the leftmost son of the root (if any). Formally, let t be a k-ary UULS. For every node x in t, we define t_x to be the *finite* tree rooted at x. For every $i \geq 0$, let $\hat{t}_{0_i}$ be the hanger tree obtained from t_{0_i} by deleting, whenever $i > 0$, the subtree $t_{0_{i-1}}$ from it. The ITS $(\mathbb{N}, <, g)$ that corresponds to the UULS t is obtained by defining, for every $i \geq 0$, $g(i) = \hat{t}_{0_i}$. The embedding of UULSs into ITSs is depicted in Figure 2. Similarly, an ITS can be easily turned into an UULS.

Let $\mathcal{B}$ be the class of Büchi sequence automata (cf. Example 2.4) and $\mathcal{D}_k$ be the class of (top-down) finite tree automata (cf. Example 2.9). We define the class of *tree sequence automata* as $\mathcal{B}(\mathcal{D}_k)$. Automata in $\mathcal{B}(\mathcal{D}_k)$ accept infinite sequences of finite trees. Since both the component automata classes are known to be closed under Boolean operations and decidable, so is the temporalized class $\mathcal{B}(\mathcal{D}_k)$ (cf. Theorem 2.6). Apparently, our purposes do not fit in with finite tree automata, since we are interested in sequences of increasing hanger trees, and not in sequences of finite trees. However, an ITS is just a particular sequence of finite trees, and thus tree sequence automata in $\mathcal{B}(\mathcal{D}_k)$ work over ITSs as well. In the following, we study the expressive power and the decidability of tree sequence automata over *increasing tree sequences*. Notice that the class $\mathcal{B}(\mathcal{D}_k)$ over ITSs is *not* a temporalized automata class, because ITSs are not temporalized models. The following theorem proves that tree sequence automata are as expressive as the monadic second-order theory of UULSs (or, equivalently, of increasing tree sequences).

Theorem 3.2 (*Expressiveness of tree sequence automata over UULSs*)
It holds that $\mathcal{B}(\mathcal{D}_k) \leftrightarrows \mathrm{MSO}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$ *over UULSs, that is,*

1. *for every automaton* $A \in \mathcal{B}(\mathcal{D}_k)$*, there is a formula* φ_A *in* $\mathrm{MSO}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$ *such that* $\mathcal{L}(A) \cap ITS_k(\Sigma) = \mathcal{M}(\varphi_A)$*;*
2. *for every formula* φ *in* $\mathrm{MSO}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$*, there is an automaton* $A_\varphi \in \mathcal{B}(\mathcal{D}_k)$ *such that* $\mathcal{M}(\varphi) = \mathcal{L}(A_\varphi) \cap ITS_k(\Sigma)$*.*

Let us now provide $\mathrm{MSO}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$ over UULSs with an expressive and elementarily decidable temporal logic counterpart. First, consider $\mathrm{PLTL}(\mathrm{CTL}^*_\mathrm{k})$ over ITSs, where $\mathrm{CTL}^*_\mathrm{k}$ is interpreted over finite trees with a strong interpretation of the next operator $\mathbf{X}$. It is possible to prove that $\mathrm{PLTL}(\mathrm{CTL}^*_\mathrm{k})$ over UULSs is as expressive as the monadic path fragment $\mathrm{MPL}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$ (the proof is similar to the one given in [4] for DULSs). Exploiting tree sequence automata, we can extend such a result to the full second-order theory of UULSs. We have that QLTL $\leftrightarrows \mathcal{B}$ and $\mathrm{QCTL}^*_\mathrm{k} \leftrightarrows \mathcal{D}_k$ (the proof of the latter equivalence is similar to the corresponding proof for infinite trees). By Theorem 2.6, we have that $\mathrm{QLTL}(\mathrm{QCTL}^*_\mathrm{k}) \leftrightarrows \mathcal{B}(\mathcal{D}_k)$ over finite tree sequences. Since the set of ITSs is a subset of the set of finite tree sequences, we have that $\mathrm{QLTL}(\mathrm{QCTL}^*_\mathrm{k}) \leftrightarrows \mathcal{B}(\mathcal{D}_k)$ over ITSs as well. By Theorem 3.2, we have that $\mathrm{QLTL}(\mathrm{QCTL}^*_\mathrm{k}) \leftrightarrows \mathrm{MSO}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$ over ITSs. Similarly, one can prove that $\mathrm{EQLTL}(\mathrm{EQCTL}^*_\mathrm{k}) \leftrightarrows \mathcal{B}(\mathcal{D}_k)$ over ITSs, and thus $\mathrm{EQLTL}(\mathrm{EQCTL}^*_\mathrm{k}) \leftrightarrows \mathrm{MSO}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$ over ITSs. The decidability of $\mathrm{QLTL}(\mathrm{QCTL}^*_\mathrm{k})$ and $\mathrm{EQLTL}(\mathrm{EQCTL}^*_\mathrm{k})$ immediately follows from that of $\mathrm{MSO}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$. In the following, we prove that $\mathrm{EQLTL}(\mathrm{EQCTL}^*_\mathrm{k})$ is elementarily decidable. Unfortunately, $\mathrm{MSO}_\Sigma[<, (\downarrow_i)_{i=0}^{k-1}]$ is nonelementarily decidable, and thus to prove that $\mathrm{EQLTL}(\mathrm{EQCTL}^*_\mathrm{k})$ is elementarily decidable we must follow a different path. It is easy to show that the encoding of $\mathrm{EQLTL}(\mathrm{EQCTL}^*_\mathrm{k})$-formulas into $\mathcal{B}(\mathcal{D}_k)$-automata is elementary (recall that the embeddings EQLTL $\to \mathcal{B}$ and $\mathrm{EQCTL}^*_\mathrm{k} \to \mathcal{D}_k$ are elementary). We will prove that the emptiness problem for $\mathcal{B}(\mathcal{D}_k)$ over ITSs is elementarily decidable. This result allows us to conclude that $\mathrm{EQLTL}(\mathrm{EQCTL}^*_\mathrm{k})$ over ITSs is *elementarily* decidable.

We now focus on the decidability and complexity of the emptiness problem for tree sequence automata in $\mathcal{B}(\mathcal{D}_k)$ over ITSs. Such a problem can be formulated as follows: given an automaton $A \in \mathcal{B}(\mathcal{D}_k)$, is there a k-ary increasing tree sequence accepted by A? Equivalently, does $\mathcal{L}(A) \cap ITS_k(\Sigma) \neq \emptyset$? Its (nonelementary) decidability immediately follows from Theorem 3.2, since, given an automaton A, we can build an equivalent monadic formula φ_A and check its satisfiability over UULSs. In the following, we give a necessary and sufficient condition that solves the problem in an *elementary* way.

Let $A = (Q, q_0, \Delta, M, \{\texttt{final}\})$ be an automaton in $\mathcal{B}(\mathcal{D}_k)$ over the alphabet $\Gamma(\Sigma) \subset \mathcal{D}_k$. Clearly, $\mathcal{L}(A) \neq \emptyset$ is necessary for $\mathcal{L}(A) \cap ITS_k(\Sigma) \neq \emptyset$. However, it is not sufficient. By definition of combined acceptance condition for A, we have that $\mathcal{L}(A) \neq \emptyset$ if and only if there is a finite sequence $q_0, q_1, \ldots q_m$ of distinct states in Q, a finite sequence $X_0, X_1, \ldots X_m$ of $\mathcal{D}_k$-automata and $j \in \{0, \ldots m\}$ such that:

1. $\Delta(q_i, X_i, q_{i+1})$, for every $i = 0, \ldots m-1$, and $\Delta(q_m, X_m, q_j)$;
2. q_j is a final state;
3. $\mathcal{L}(X_i) \neq \emptyset$, for every $i = 0, \ldots m$

To obtain a necessary and sufficient condition for $\mathcal{L}(A) \cap ITS_k(\Sigma) \neq \emptyset$, we have to strengthen condition (3) as follows. Let $T_k^i(\Sigma)$ be the set of k-ary hanger trees of height i:

3'. (3'a) $\mathcal{L}(X_i) \cap T_k^i(\Sigma) \neq \emptyset$, for every $i = 0, \ldots j-1$, and (3'b) $\mathcal{L}(X_i) \cap T_k^{i+y \cdot l}(\Sigma) \neq \emptyset$, for every $i = j, \ldots m$ and $y \geq 0$, where $l = m - j + 1$.

The conjunction of conditions (1,2,3') is a necessary and sufficient condition for $\mathcal{L}(A) \cap ITS_k(\Sigma) \neq \emptyset$. We show that conditions (1,2,3') are elementarily decidable. Clearly, there are elementarily many runs in A satisfying conditions (1,2). The following nontrivial Lemma 3.3 shows that condition 3' is elementarily decidable.

Lemma 3.3 *Let X be a finite tree automaton, and $a, l \geq 0$. Then, the problem $\mathcal{L}(X) \cap T_k^{a+y \cdot l}(\Sigma) \neq \emptyset$, for every $y \geq 0$, is elementarily decidable.*

Proof.

Let $X = (Q, q_0, \Delta, M, \{\texttt{final}\})$ over $\Gamma(\Sigma)$. If $l = 0$, then the problem reduces to checking $\mathcal{L}(X) \cap T_k^a(\Sigma) \neq \emptyset$, for some $a \geq 0$. For every $a \geq 0$, the set T_k^a is finite and hence regular. Since finite tree automata are elementarily closed under Boolean operations and are elementarily decidable, we conclude that in this case the condition is elementarily effective.

Suppose now $l > 0$. For the sake of simplicity, we first give the proof for finite *sequence* automata, and then we discuss how to modify it to cope with the case of finite tree automata. Hence, let X be a finite sequence automaton. We have to give an elementarily effective procedure that checks whether X recognizes at least one sequence of length a, at least one of length $a + l$, at least one of length $a + 2l$, and so on. Without loss of generality, we may assume that the set of final states of X is the singleton containing $q_{fin} \in Q$. Hence, the problem reduces to check, for every $y \geq 0$, the existence of a path from q_{in} to q_{fin} of length $a + y \cdot l$ in the state-transition graph associated with X. We thus need to solve the following problem of Graph Theory, which we call the *Periodic Path Problem* (PPP for short):

> Let $G = (N, E)$ be a finite directed graph, q_1, q_2 be two nodes in N, and $a, l \geq 0$ be two natural numbers. Is there a path in G from q_1 to q_2 of length $a + y \cdot l$, for every $y \geq 0$?

In the following, we further reduce the PPP to a problem of Number Theory. Let $\Pi_{q_1,q_2}(G)$ be the set of paths from q_1 to q_2 in the graph G. Given $\pi \in \Pi_{q_1,q_2}(G)$, we denote by π the path obtained from π by removing all its cyclic subpaths. If π is acyclic, then $\pi \quad = \pi$; otherwise, if $\pi = \alpha q' \beta q' \gamma$, then $\pi \quad = \alpha \quad q' \gamma \quad$. Let $\sim_{q_1,q_2}$ be the relation on $\Pi_{q_1,q_2}(G)$ such that $\pi_1 \sim_{q_1,q_2} \pi_2$ if and only if $\pi_1 \quad = \pi_2 \,$. Note that $\sim_{q_1,q_2}$ is an equivalence relation of finite index. For every equivalence class $[\pi]_{\sim_{q_1,q_2}}$, we need a formula expressing the length of a generic path in the class. Note that every path in $[\pi]_{\sim_{q_1,q_2}}$ differs from any other path in the same class only because of some cyclic subpaths. More precisely, let μ be the shortest path in $[\pi]_{\sim_{q_1,q_2}}$ and let $C_1, \dots, C_n$ be the the cycles intersecting π of length $w_1, \dots, w_n$, respectively. Obviously, μ does not cycle through any C_i. Every path in $[\pi]_{\sim_{q_1,q_2}}$ starts from q_1, cycles an arbitrary number of times (possibly zero) through every C_i, and finally reaches q_2. It is easy to see that the length of an arbitrary path $\sigma \in [\pi]_{\sim_{q_1,q_2}}$ is given by the parametric formula:

$$|\sigma| = |\mu| + \sum_{i=1}^{n} x_i \cdot w_i,$$

where $x_i \geq 0$ in the number of times the path σ cycles through C_i.
Let $[\pi_1]_{\sim_{q_1,q_2}}, \dots, [\pi_m]_{\sim_{q_1,q_2}}$ be the equivalence classes of $\sim_{q_1,q_2}$. For every $j = 1, \dots, m$, let μ_j be the shortest path in $[\pi_j]_{\sim_{q_1,q_2}}$ and let $C^j_1, \dots, C^j_n$ be the the cycles intersecting π_j of length $w^j_1, \dots, w^j_n$, respectively. Moreover, let

$$Y_j = \{y \geq 0 \mid \exists x_1, \dots, x_n \geq 0 \, (|\mu_j| + \sum_{i=1}^{n} x_i \cdot w^j_i = a + y \cdot l)\}.$$

The PPP reduces to the following problem of Number Theory:

Do the sets $Y_1, \dots, Y_m$ cover the set of natural numbers, that is, does $\bigcup_{j=1}^{m} Y_j = \mathbb{N}$?

We now solve this latter problem. Let $w_i \geq 0$, for $i = 1, \dots, n$. We are interested in the form of the set $S = \{\sum_{i=1}^{n} x_i \cdot w_i \mid x_i \geq 0\}$. Let $W = (w_1, \dots w_n)$ and let $d = MCD(W)$ (the maximum common divisor of $\{w_1, \dots, w_n\}$). We distinguish the cases $d = 1$ and $d \neq 1$.

If $d = 1$, then it is easy to see that:

$$S = E \cup \{j \mid j \geq k\},$$

where E is a finite set of *exceptions* such that $max(E) < k$, and $k = (w_r - 1) \cdot (w_s - 1)$, with $w_r = min(W)$ (the minimum of $\{w_1, \dots w_n\}$) and $w_s = min(W \setminus \{w_r\})$.

If $d \neq 1$, then consider the set $S' = \{\sum_{i=1}^{n} x_i \cdot w_i/d \mid x_i \geq 0\}$. Clearly, it holds that $MCD(w_1/d, \dots w_n/d) = 1$ and, thus, as above, $S' = E' \cup \{j \mid j \geq k'\}$ for some finite set E' and $k' \in \mathbb{N}$. Therefore, in this case, we have that

$$S = E' \cdot d \cup \{j \mid j \geq k' \cdot d \wedge d \texttt{ DIV j}\},$$

where $d \texttt{ DIV j}$ means that d is a divisor of j. Summing up, the set S can be described in a compact way as follows:

$$S = E \cup \{k + j \cdot d \mid j \in \mathbb{N}\},$$

for some finite (computable) set E, some (computable) $k \in \mathbb{N}$, and $d = MCD(W)$, that is, the set S can be expressed as the union of a finite and computable set of exceptions and an arithmetic progression.

Now we consider the equation $\sum_{i=1}^{n} x_i \cdot w_i = y \cdot l$. Our aim is to describe the set $Y = \{y \geq 0 \mid \exists x_1, \dots x_n \geq 0 \, (\sum_{i=1}^{n} x_i \cdot w_i = y \cdot l)\}$ in a similar way. Let $e = MCD(d, l)$, $l = l' \cdot e$, and $d = d' \cdot e$. We have that:

$$
\begin{array}{ll}
y \in Y & \text{iff} \\
y \cdot l \in S & \text{iff} \\
y \cdot l \in E \lor y \cdot l \geq k \land d \texttt{ DIV y} \cdot \texttt{l} & \text{iff} \\
y \cdot l \in E \lor y \geq \lceil k/l \rceil \land d' \cdot e \texttt{ DIV y} \cdot \texttt{l}' \cdot \texttt{e} & \text{iff} \\
y \cdot l \in E \lor y \geq \lceil k/l \rceil \land d' \texttt{ DIV y} &
\end{array}
$$

Therefore, the set Y is the union of a finite and computable set and an arithmetic progression, that is,

$$Y = E' \cup \{k' + j \cdot d' \mid j \in \mathbb{N}\},$$

for some finite (computable) set E', some (computable) $k' \in \mathbb{N}$, and $d' = d/MCD(d, l)$. The set $Y = \{y \geq 0 \mid \exists x_1, \dots x_n \geq 0\,(\sum_{i=1}^{n} x_i \cdot w_i = a + y \cdot l)\}$, with $a \in \mathbb{N}$, can be described in the same way.

We have shown that, for $i = 1, \dots, m$, every Y_i has the form $E_i \cup \{k_i + y \cdot d_i \mid y \geq 0\}$ for some finite set E_i, and some $k_i, d_i \in \mathbb{N}$. We now give a solution to the problem of establishing whether or not $\bigcup_{i=1}^{m} Y_i = \mathbb{N}$. Let $k_r = min\{k_1, \dots, k_m\}$ and $D = mcm(d_1, \dots, d_m)$ (the minimum common multiple of $\{d_1, \dots, d_m\}$). The algorithm works as follows: for every $k < k_r$, we check whether $k \in Y_i$ for some $i = 1, \dots, m$. If this is not the case, the problem has no solution. Otherwise, we verify whether, for every $j = 0, \dots, D-1$, $k_r + j \in Y_i$ for some $i = 1, \dots, m$. If this is the case, then we have a solution, otherwise, there is no solution. Note that a solution can be described in terms of an ultimately periodic word $w = uv^{\omega}$, with $u, v \in \{1, \dots, m\}^*$, such that, for every $i \geq 0$, $w(i) = j$ means that a path from q_1 to q_2 in the graph G belongs to the j-th equivalence class $[\pi_j]_{\sim_{q_1,q_2}}$.

The above algorithm solves the periodic path problem in doubly exponential time in the number n of nodes of the graph G. The number of equivalence classes of the relation $\sim_{q_1,q_2}$ over the set of paths from q_1 to q_2 in G may be exponential in n. Thus, we have m sets $Y_1, \dots, Y_m$, each one associated with a relevant equivalence class, and $m = \mathcal{O}(2^n)$. Every set Y_i can be represented in polynomial time as $E_i \cup \{k_i + y \cdot d_i \mid y \geq 0\}$, for some finite E_i and some $k_i, d_i \in \mathbb{N}$. Note that the cardinality of E_i is bounded by k_i, $k_i = \mathcal{O}(n^2)$, and $d_i = \mathcal{O}(n)$. The final step of the procedure makes $k_0 + D$ membership tests with respect to some set Y_i, where $k_0 = min\{d_1, \dots, d_m\}$ and $D = mcm(d_1, \dots, d_m)$. Each test is performed in $\mathcal{O}(1)$. Moreover, D is bounded by ${d_0}^m$, where $d_0 = max\{d_1, \dots, d_m\}$, and thus $D = \mathcal{O}(2^{2^n})$. Hence, the procedure works in time doubly exponential.

The general case of finite trees is similar. Let X be a finite tree automaton. A path from q_1 to q_2 corresponds to a run of X such that the run tree is complete and k-ary, the root of the run tree is labeled with state q_1 and the leaves of the run tree are labeled with state q_2. A cycle is a path from q to q. The problem is to find, for every $y \geq 0$, a path from the initial state q_{in} to the final state q_{fin} of length $a + y \cdot l$. The rest of the proof proceeds along the same reasoning path followed for sequence automata. ■

It follows that, given a $\mathcal{B}(\mathcal{D}_k)$-automaton A, we have an algorithm to solve the problem $\mathcal{L}(A) \cap ITS_k(\Sigma) \neq \emptyset$ whose time is doubly exponential in the size of A.

Theorem 3.4 *The emptiness problem for finite tree sequence automata over UULSs is in 2EXPTIME.*

Since EQLTL(EQCTL$^*_k$)-formulas can be elementarily converted into $\mathcal{B}(\mathcal{D}_k)$-automata, we have the following.

Theorem 3.5 *The satisfiability problem for* EQLTL($\mathrm{EQCTL}^*_\mathrm{k}$) *over UULSs is elementarily decidable.*

References

[1] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B*, pages 995–1072. Elsevier Science Publishers B.V., 1990.

[2] M. Finger and D. M. Gabbay. Adding a temporal dimension to a logic system. *Journal of Logic Language and Information*, 1:203–233, 1992.

[3] M. Franceschet. *Dividing and Conquering Time Granularity.* PhD thesis, Department of Mathematics and Computer Science, Udine, Italy, October 2001.

[4] M. Franceschet and A. Montanari. Branching within time: an expressively complete and elementarily decidable temporal logic for time granularity. *Journal of Language and Computation*, 2001. To appear.

[5] M. Franceschet and A. Montanari. Towards an automata-theoretic counterpart of combined temporal logics. In *Proceedings of the International Workshop on Verification and Computational Logic*, pages 55–74, 2001.

[6] M. Franceschet, A. Montanari, and M. de Rijke. Model checking for combined logics. In *Proceedings of the International Conference on Temporal Logic*, pages 65–73, 2000.

[7] T. Hafer and W. Thomas. Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. In T. Ottmann, editor, *Proceedings of the International Colloquium Automata, Languages and Programming*, volume 267 of *Lecture Notes in Computer Science*, pages 269–279, Karlsruhe, Germany, 1987. Springer.

[8] N. Immerman and D. Kozen. Definability with bounded number of bound variables. *Information and Computation*, 83(2):121–139, 1989.

[9] A. Montanari. *Metric and Layered Temporal Logic for Time Granularity.* ILLC Dissertation Series 1996-02, Institute for Logic, Language and Computation, University of Amsterdam, 1996.

[10] A. Montanari, A. Peron, and A. Policriti. Extending Kamp's theorem with binary operators to model time granularity. *Journal of Logic and Computation.* To appear.

[11] A. Montanari, A. Peron, and A. Policriti. Decidable theories of ω-layered metric temporal structures. *Logic Journal of the IGPL*, 7(1):79–102, 1999.

[12] A. Montanari, A. Peron, and A. Policriti. The taming (timing) of the states. *Logic Journal of the IGPL*, 8(5):681–699, 2000.

[13] N. Vitacolonna. Granularità e logiche a intervalli: risultati di decidibilità. Master's thesis, Department of Mathematics and Computer Science, University of Udine – Italy, 2001. In Italian.