Temporalized logics and automata for time granularity

MASSIMO FRANCESCHET
Department of Sciences, University of Chieti-Pescara, Italy
(*e-mail:* francesc@sci.unich.it)

ANGELO MONTANARI
Department of Mathematics and Computer Science, University of Udine, Italy
(*e-mail:* montana@dimi.uniud.it)

submitted 18 March 2002; revised 14 January 2003; accepted 5 September 2003

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 a major requirement for many applications, including formal specification and verification, temporal databases, data mining, problem solving, and natural language understanding. In particular, the addition of a granularity dimension 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.

Suitable extensions of the monadic second-order theory of k successors have been proposed in the literature to capture the notion of time granularity. In this paper, we provide the monadic second-order theories of downward unbounded layered structures, which are infinitely refinable structures consisting of a coarsest domain and an infinite number of finer and finer domains, and of upward unbounded layered structures, which consist of a finest domain and an infinite number of coarser and coarser domains, with expressively complete and elementarily decidable temporal logic counterparts.

We obtain such a 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 the temporal logic counterparts of the given theories of time granularity to the easier one of finding temporalized automata counterparts of them.

1 Introduction

Time granularity is an important, but not always well-understood, research theme in computer science. To acquaint the reader with the basics of the subject, we start the paper with a gentle introduction to research on time granularity. In Section 1.1, we briefly illustrate the intersection of research on time granularity with different areas of computer science, ranging from system specification and verification to natural language understanding, and we give a high-level view of the logical approach to the problem of representing and reasoning about time granularity that we follow in the paper. In Section 1.2, we focus on the topics addressed in the paper, and we outline its main contributions. In Section 1.3, we show that the considered topics present interesting connections with a number of issues relevant to various research directions in computer science logic, including real-time logics, interval logics, and combined logics. We conclude the introduction by a short description of the organization of the rest of the paper.

1.1 Representing and reasoning about time granularity

The ability of providing and relating temporal representations at different 'grain levels' of the same reality is an important research theme in various fields of computer science, including formal specification and verification, temporal databases, data mining, problem solving, and natural language understanding. As for *formal specifications*, there exists a large class of reactive systems whose components have dynamic behavior regulated by very different time constants (granular reactive systems). A good specification language must enable one to specify and verify the components of a granular reactive system and their interactions in a simple and intuitively clear way (Ciapessoni et al. 1993; Corsetti et al. 1991; Corsetti et al. 1991; Fiadeiro and Maibaum 1994; Lamport 1985; Montanari et al. 2002; Montanari et al. 1999; Montanari et al. 2000; Montanari and Policriti 1996). As for *temporal databases*, the common way to represent temporal information is to timestamp either attributes (*attribute timestamping*) or tuples/objects (*tuple-timestamping*). Timestamping is performed taking time values over some fixed granularity. However, it may happen that differently-grained timestamps are associated with different data. This is the case, for instance, when information is collected from distinct sources which are not under the same control. Moreover, users and application programs may require the flexibility of viewing and querying temporal data at different time granularities. To guarantee consistency either the data must be converted into a uniform granularity-independent representation or temporal database operations must be generalized to cope with data associated with different temporal domains. In both cases, a precise semantics for time granularity is needed (Bettini et al. 1997; Chandra et al. 1994; Combi and Pozzi 2001; Dyreson and Snodgrass 1995; Jajodia et al. 1993; Jajodia et al. 1995; Montanari and Pernici 1993; Ning et al. 2002; Niezette and Stevenne 1993; Segev and Chandra 1993; Wijsen 1998; Wijsen 1999). With regard to *data mining*, a huge amount of data is collected every day in the form of event-time sequences. These sequences represent valuable sources of information, not only for what is explicitly recorded, but also for deriving implicit information and predicting the future behavior of the monitored process. This latter activity requires an analysis of the frequency of certain events, the discovery of their regularity, and the identification of sets of events that are linked by particular temporal relationships. Such frequencies, regularity, and relationships are often expressed in terms of multiple granularities, and thus analysis and discovery tools must be able to cope with them (Agrawal and Srikant 1995; Bettini et al. 1998; Bettini et al. 1996b; Dreyer et al. 1994; Mannila et al. 1995). With regard

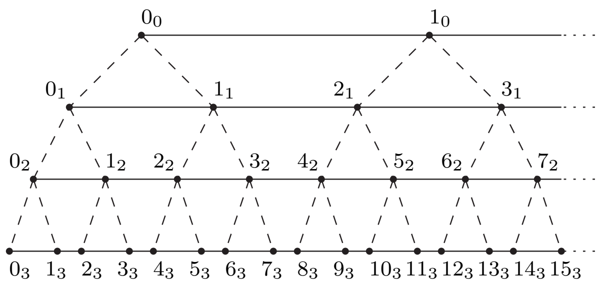

Fig. 1. The 2-refinable 4-layered structure.

to *problem solving*, several problems in scheduling, planning, and diagnosis can be formulated as temporal constraint satisfaction problems provided with a time granularity dimension. Variables are used to represent events occurring at different time granularities and constraints are used to represent temporal relations between events (Bettini et al. 1996a; Cukierman and Delgrande 1998; Euzenat 1995; Ladkin 1987; Montanari et al. 1992; Mota and Robertson 1996; Poesio and Brachman 1991; Shahar 1996). Finally, shifts in the temporal perspective are common in natural language communication, and thus the ability of supporting and relating a variety of temporal models, at different grain sizes, is a relevant feature for the task of *natural language processing* (Blackburn and Bos 2003; Foster et al. 1986; Fum et al. 1989; Kamp and Schiehlen 2001).

According to a commonly accepted perspective, 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). A representation formalism can then use these granules to provide facts, actions or events 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.

Different approaches to represent and to reason about time granularity have been proposed in the literature. In the following, we introduce the distinctive features of the logical approach to time granularity[1]. In the *logical* setting, the different time granularities and their interconnections are represented by means of mathematical structures, called layered structures. A *layered structure* consists of a possibly infinite set of related differently-grained temporal domains. Such a structure identifies the relevant temporal domains and defines the relations between time points belonging to different domains. Suitable operators make it possible to move horizontally *within* a given temporal domain (displacement operators), and to move vertically *across* temporal domains (projection operators). Both classical and temporal logics

[1] In (Franceschet and Montanari 2002) we analyze alternative approaches to time granularity, developed in the context of temporal databases, and we compare them with the logical one.

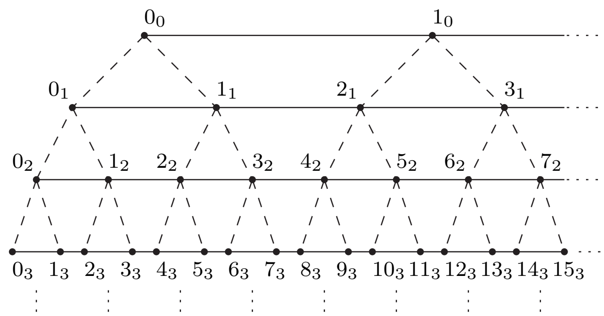

Fig. 2. The 2-refinable downward unbounded layered structure.

can be interpreted over the layered structure. Logical formulas allow one to specify properties involving different time granularities in a single formula by mixing displacement and projection operators. Algorithms are provided to verify whether a given formula is consistent (*satisfiability checking*) as well as to check whether a given formula is satisfied in a particular structure (*model checking*). The logical approach to represent time granularity has been mostly applied in the field of formal specification and verification of concurrent systems. An application of time granularity logics to the specification of a supervisor that automates the activities of a high voltage station, devoted to the end user distribution of the energy generated by power plants, has been accomplished in collaboration with Automation Research Center of the Electricity Board of Italy (ENEL). A short account of this work has been given in (Ciapessoni et al. 1993). Logics for time granularity have also been applied to the specification of real-time monitoring systems (Corsetti et al. 1991), mobile systems (Franceschet et al. 2000), and therapy plans in clinical medicine (Combi et al. 2002).

A systematic logical framework for time granularity, based on a many-level view of temporal structures, with matching logics and decidability results, has been proposed in (Montanari 1996; Montanari and Policriti 1996; Montanari et al. 1999) and later extended in (Franceschet 2002; Franceschet and Montanari 2001a; Franceschet and Montanari 2001b; Franceschet and Montanari 2003). Layered structures with exactly $n \geq 1$ temporal domains such that each time point can be refined into $k \geq 2$ time points of the immediately finer temporal domain, if any, are called k-refinable n-layered structures (n-LSs for short, see Figure 1). They have been investigated in (Montanari and Policriti 1996), where a classical second-order language, with second-order quantification restricted to monadic predicates, has been interpreted over them. The language includes a total order $<$ and k projection functions $\downarrow_0, \ldots, \downarrow_{k-1}$ over the layered temporal universe such that, for every point x, $\downarrow_0(x), \ldots, \downarrow_{k-1}(x)$ are the k elements of the immediately finer temporal domain, if any, into which x is refined. The satisfiability problem for the monadic second-order language over n-LSs has been proved to be decidable by using a reduction to the emptiness problem for Büchi sequence automata. Unfortunately, the decision procedure has a nonelementary complexity.

Layered structures with an infinite number of temporal domains, ω-layered struc-

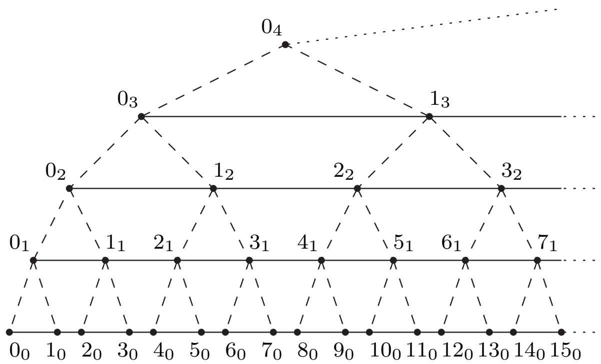

Fig. 3. The 2-refinable upward unbounded layered structure.

tures, have been studied in (Montanari et al. 1999). In particular, the authors investigated *k-refinable downward unbounded layered structures* (DULSs), that is, ω-layered structures consisting of a coarsest domain together with an infinite number of finer and finer domains (see Figure 2), and *k-refinable upward unbounded layered structures* (UULSs), that is, ω-layered structures consisting of a finest temporal domain together with an infinite number of coarser and coarser domains (see Figure 3). A classical monadic second-order language, including a total order $<$ and k projection functions $\downarrow_0, \ldots, \downarrow_{k-1}$, has been interpreted over both UULSs and DULSs. The decidability of the monadic second-order theories of UULSs and DULSs has been proved by reducing the satisfiability problem to the emptiness problem for systolic and Rabin tree automata, respectively. In both cases the decision procedure has a nonelementary complexity.

1.2 Our contributions

Monadic logics for time granularity are quite expressive, but, unfortunately, they have few computational appealing: their decision problem is indeed *nonelementary*. This roughly means that it is possible to algorithmically check satisfiability, but the complexity of the algorithm grows very rapidly and cannot be bounded. Moreover, the corresponding automata (Büchi sequence automata for the theory of finitely-layered structures, Rabin tree automata for downward unbounded structures, and systolic tree automata for upward unbounded ones) do not directly work over layered structures, but rather over collapsed structures into which layered structures can be encoded. Hence, they are not natural and intuitive tools to specify and check properties of time granularity.

In this paper, we follow a different approach. Taking inspiration from combination methods for temporal logics, we start by studying how to combine automata in such a way that properties of the components are inherited by the combination. Then, we reinterpret layered structures as *combined structures*. This intuition reveals to be the keystone of our endeavor. Indeed, it allows us to define combined temporal logics and combined automata over layered structures, and to study their expressive power and computational properties by taking advantage of the transfer

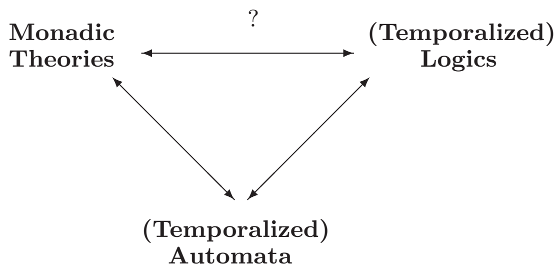

Fig. 4. From monadic theories to (temporalized) logics via (temporalized) automata.

theorems for combined logics and combined automata. The outcome is appealing: the resulting combined temporal logics and automata directly work over layered structures. Moreover, they are expressively equivalent to monadic languages, and they are elementarily decidable.

Finding the temporal logic counterpart of monadic theories is a difficult task, involving a nonelementary 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 (Immerman and Kozen 1989) and well-behaved fragments of second-order ones, e.g. the path fragment of the monadic second-order theory of infinite binary trees (Hafer and Thomas 1987). 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 (Franceschet and Montanari 2003), while Montanari et al. extended Kamp's theorem to deal with the first-order fragment of the theory of UULSs (Montanari et al. 2002). Unfortunately, these techniques produce rather involved proofs and do not naturally lift to the full second-order case.

In this paper, instead of trying to establish a direct correspondence between monadic second-order theories for time granularity and temporal logics, we connect them via automata (cf. Figure 4). 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, 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 theories of DULSs and 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.

As a by-product, the alternative characterization of temporalized logics for time granularity as temporalized automata allows one to reduce logical problems to automata ones. As it is well-known in the area of automated system specification and verification, such a reduction presents several advantages, including the possibility of using automata for both system modeling and specification, and the possibility of checking the system on-the-fly (a detailed account of these advantages can be found in (Franceschet and Montanari 2001b)).

1.3 Related fields

The original motivation of our research was the design of a temporal logic embedding the notion of time granularity, suitable for the specification of complex concurrent systems whose components evolve according to different time units. However, we soon established a fruitful complementary point of view on time granularity: it can be regarded as a powerful setting to investigate the definability of meaningful timing properties over a *single* time domain. Moreover, layered structures and logics provide an interesting embedding framework for flat *real-time* structures and logics, as well as there exists a natural link between structures and theories of time granularity and those developed for representing and reasoning about *time intervals*. Finally, 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. In the following, we briefly explain all these connections.

Granular reactive systems. As pointed out above, we were originally motivated by the design of a temporal logic embedding the notion of time granularity suitable for the specification of granular reactive systems. A *reactive system* is a concurrent program that maintains and interaction with the external environment and that ideally runs forever. Temporal logic has been successfully used for modeling and analyzing the behavior of reactive systems (Emerson 1990). It supports semantic model checking, which can be used to check specifications against system behaviors; it also supports pure syntactic deduction, which may be used to verify the consistency of specifications. Finite-state automata, such as Büchi sequence automata and Rabin tree automata (Thomas 1990), have been proved very useful in order to provide clean and asymptotically optimal satisfiability and model checking algorithms for temporal logics (Kupferman et al. 2000; Vardi and Wolper 1994) as well as to cope with the *state explosion problem* that frightens concurrent system verification (Courcoubetis et al. 1991; Jard and Jeron 1989; Vardi and Wolper 1986).

A *granular reactive systems* is a reactive system whose components have dynamic behaviours regulated by very different time constants (Montanari 1996). As an example, consider a pondage power station consisting of a reservoir, with filling and emptying times of days or weeks, generator units, possibly changing state in a few seconds, and electronic control devices, evolving in microseconds or even less. A complete specification of the power station must include the description of these components and of their interactions. A natural description of the temporal evolution of the reservoir state will probably use days: "During rainy weeks, the level of the reservoir increases 1 meter a day", while the description of the control devices behaviour may use microseconds: "When an alarm comes from the level sensors, send an acknowledge signal in 50 microseconds". We say that systems of such a type have *different time granularities*. It is somewhat unnatural, and sometimes impossible, to compel the specifier to use a unique time granularity, microseconds in the previous example, to describe the behaviour of all the components. A good language must indeed allow the specifier to easily describe all simple and intuitively clear facts (naturalness of the notation). Hence, a specification language for granular reactive systems must support different time granularities to allow one (i) to maintain the specifications of the dynamics of differently-grained components as separate as possible (modular specifications), (ii) to differentiate the refinement degree of the specifications of different system components (flexible specifications), and (iii) to write complex specifications in an incremental way by refining higher-level predicates associated with a given time granularity in terms of more detailed ones at a finer granularity (incremental specifications).

Definability of meaningful timing properties. Time granularity can be viewed not only as an important feature of a representation language, but also as a formal tool to investigate the definability of meaningful timing properties, such as density and exponential grow/decay, over a *single* time domain (Montanari et al. 1999). In this respect, the number of layers (single vs. multiple, finite vs. infinite) of the underlying temporal structure, as well as the nature of their interconnections, play a major role: certain timing properties can be expressed using a single layer; others using a finite number of layers; others only exploiting an infinite number of layers. For instance, temporal logics over binary 2-layered structures suffice to deal with conditions like "P holds at all even times of a given temporal domain" that cannot be expressed using flat propositional temporal logics (Wolper 1983). Moreover, temporal logics over ω-layered structures allow one to express relevant properties of infinite sequences of states over a single temporal domain that cannot be captured by using flat or n-layered 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 finite number of layers, while temporal logics over DULSs allow one to constrain a given property to hold true 'densely' over a given time interval.

On the relationship with real-time logics. Layered structures and logics can be regarded as an embedding framework for flat real-time structures and logics. A *real-time system* is a reactive system with well-defined fixed-time constraints. Systems that control scientific experiments, industrial control systems, automobile-engine fuel-injection systems, and weapon systems are examples of real-time systems. Examples of quantitative timing properties relevant to real-time systems are periodicity, bounded responsiveness, and timing delays. Logics for real-time systems, called real-time logics, are interpreted over *timed state sequences*, that is, state sequences in which every state is associated with a time instant.

Montanari et al. showed that the second-order theory of timed state sequences can be *properly* embedded into the second-order theory of binary UULSs as well as into the second-order theory of binary DULSs (Montanari et al. 2000). The increase in expressive power of the embedding frameworks makes it possible to express and check additional timing properties of real-time systems, which cannot be dealt with by the classical theory. For instance, in the theory of timed state sequences, saying that a state s holds true at time i can be meant to be an abstraction of the fact that state s can be arbitrarily placed in the time interval $[i, i+1)$. The stratification of domains in layered structures naturally supports such an interval interpretation and gives means for reducing the uncertainty involved in the abstraction process. For instance, it allows on to say that a state s belongs to the first (respectively, second) half of the time interval $[i, i+1)$. More generally, the embedding of real-time logics into the granularity framework allows one to deal with *temporal indistinguishability* of states (two or more states associated with the same time) and *temporal gaps* between states (a nonempty time interval between the time associated to two contiguous states). Temporal indistinguishability and temporal gaps can indeed be interpreted as phenomena due to the fact that real-time logics lack the ability to express properties at the right (finer) level of granularity: distinct states, associated with the same time, can always be ordered at the right level of granularity; similarly, time gaps represent intervals in which a state cannot be specified at a finer level of granularity. A finite number of layers is obviously not sufficient to capture timed state sequences: it is not possible to fix a priori any bound on the granularity that a domain must have to allow one to temporally order a given set of states, and thus we need to have an infinite number of temporal domains at our disposal.

On the relationship with interval logics. As pointed out in (Montanari 1996), there exists a natural link between structures and theories of time granularity and those developed for representing and reasoning about time intervals. Differently-grained temporal domains can indeed be interpreted as different ways of partitioning a given discrete/dense time axis into consecutive disjoint intervals. According to this interpretation, every time point can be viewed as a suitable interval over the time axis and projection implements an intervals-subintervals mapping. More precisely, let us define *direct constituents* of a time point x, belonging to a given domain, the time points of the immediately finer domain into which x can be refined, if any, and *indirect constituents* the time points into which the direct constituents of x can be directly or indirectly refined, if any. The mapping of a given time point into its direct or indirect constituents can be viewed as a mapping of a given time interval into (a specific subset of) its subintervals.

The existence of such a natural correspondence between interval and granularity structures hints at the possibility of defining a similar connection at the level of the corresponding theories. For instance, according to such a connection, temporal logics over DULSs allow one to constrain a given property to hold true densely over a given time interval, where P densely holds over a time interval w if P holds over w

and there exists a direct constituent of w over which P densely holds. In particular, establishing a connection between structures and logics for time granularity and those for time intervals would allow one to transfer decidability results from the granularity setting to the interval one. As a matter of fact, most interval temporal logics, including Moszkowski's Interval Temporal Logic (ITL) (Moszkowski 1983), Halpern and Shoham's Modal Logic of Time Intervals (HS) (Halpern and Shoham 1991), Venema's CDT Logic (Venema 1991), and Chaochen and Hansen's Neighborhood Logic (NL) (Zhou and Hansen 1998), are highly undecidable. Decidable fragments of these logics have been obtained by imposing severe restrictions on their expressive power, e.g., the *locality* constraint in (Moszkowski 1983).

Preliminary results can be found in (Montanari et al. 2002), where the authors propose a new interval temporal logic, called Split Logic (SL for short), which is equipped with operators borrowed from HS and CDT, but is interpreted over specific interval structures, called *split-frames*. The distinctive feature of a split-frame is that there is at most one way to chop an interval into two adjacent subintervals, and consequently it does not possess *all* the intervals. They prove the decidability of SL with respect to particular classes of split-frames which can be put in correspondence with the first-order fragments of the monadic theories of time granularity. In particular, *discrete* split-frames with maximal intervals correspond to finitely layered structures, discrete split-frames (with unbounded intervals) can be mapped into upward unbounded layered structures, and *dense* split-frames with maximal intervals can be encoded into downward unbounded layered structures.

The combining logic perspective. There are significant similarities between the problems we addressed in the time granularity setting and those dealt with by current research on logics that model changing contexts and perspectives. The design of these types of logics is emerging as a relevant research topic in the broader area of combination of logics, theories, and structures, at the intersection of logic with artificial intelligence, computer science, and computational linguistics (Gabbay and de Rijke 2000). The reason is that application domains often require rather complex hybrid description and specification languages, while theoretical results and implementable algorithms are at hand only for simple basic components (Gabbay et al. 2003). As for granular reactive systems, their operational behavior can be naturally described as a suitable combination of temporal *evolutions* (sequences of component states) and temporal *refinements* (mapping of a component state into a finite sequence of states belonging to a finer component). According to such a point of view, the model describing the operational behavior of the system and the specification language can be obtained by *combining* simpler models and languages, respectively, and model checking/satisfiability procedures for combined logics can be used.

From the above discussion, it turns out that the time granularity framework is expressive and flexible enough to be used to investigate many interesting topics not explicitly related to time granularity. The aim of this paper is to deepen our understanding of time granularity. The rest of the paper is organized as follows.

In Section 2, we introduce temporalized automata and we show that relevant logical properties, such as closure under Boolean operations and decidability, transfer from component automata to temporalized ones; furthermore, we prove that temporalized automata are as expressive as temporalized logics. In Section 3 we exploit temporalized automata to find the temporal logic counterparts of the given theories of time granularity. Temporalized automata for the theories of DULSs and UULSs are obtained as combinations of Büchi and Rabin automata and of Büchi and finite tree automata, respectively. As a matter of fact, unlike the case of DULSs, the combined model we use to encode an UULS differs from that of pure temporalization since the innermost submodels are not independent from the outermost top-level model. In Section 4, we apply temporalized logics to a real-world case study. Conclusive remarks provide an assessment of the work done and outline some future research directions.

2 Temporalized logics and automata

In this section we recall the definition of temporalization and we define temporalized automata[2]. Moreover, we prove the equivalence of temporalized automata and temporalized logics. We will take into consideration the following well-known temporal logics: *Propositional Linear Temporal Logic* (PLTL), *Quantified Linear Temporal Logic* (QLTL), *Existentially Quantified Linear Temporal Logic* (EQLTL), *Directed Computational Tree Logic* ($\mathrm{CTL}^*_\mathrm{k}$), *Quantified Directed Computational Tree Logic* ($\mathrm{QCTL}^*_\mathrm{k}$), and *Existentially Quantified Directed Computational Tree Logic* ($\mathrm{EQCTL}^*_\mathrm{k}$); moreover, we will take advantage of the following well-known finite-state automata classes: Büchi sequence automata, Rabin tree automata, finite tree automata.

Let $\mathcal{P} = \{P, Q, \ldots\}$ be a set of proposition letters. We consider *temporal logics* over the set of propositional letters $\mathcal{P}$. Given a temporal logic $\mathbf{T}$, we use $\mathcal{L}_\mathbf{T}$ and $\mathsf{K}_\mathbf{T}$ to denote the language and the set of models of $\mathbf{T}$, respectively. Furthermore, we write $OP(\mathbf{T})$ to denote the set of temporal operators of $\mathbf{T}$.

Temporalization is a simple form of logic combination that embeds one component logic into the other (Finger and Gabbay 1992). Let $\mathbf{T}$ be a temporal logic and $\mathbf{L}$ an arbitrary logic. For the sake of simplicity, we constrain $\mathbf{L}$ to be an extension of propositional logic. We partition the set of $\mathbf{L}$-formulas into *Boolean combinations* $BC_\mathbf{L}$ and *monolithic formulas* $ML_\mathbf{L}$: α belongs to $BC_\mathbf{L}$ if its outermost operator is a Boolean connective; otherwise it belongs to $ML_\mathbf{L}$. We assume that $OP(\mathbf{T}) \cap OP(\mathbf{L}) = \emptyset$.

Definition 2.1
(Temporalization – Syntax)

The language $\mathcal{L}_{\mathbf{T}(\mathbf{L})}$ of the *temporalization* $\mathbf{T}(\mathbf{L})$ of $\mathbf{L}$ by means of $\mathbf{T}$ over the set of

[2] We assume the reader to be familiar with basic concepts of modal and temporal logics, and automata. If this is not the case, comprehensive surveys are given in (Emerson 1990) and (Thomas 1990), respectively.

proposition letters $\mathcal{P}$ is obtained by taking the set of formation rules of $\mathcal{L}_{\mathbf{T}}$ and by replacing the atomic formation rule: "every proposition letter $P \in \mathcal{P}$ is a formula" by the rule: "every monolithic formula $\alpha \in \mathcal{L}_{\mathbf{L}}$ is a formula". □

As an example, let $\mathbf{T_1}$ and $\mathbf{T_2}$ be two temporal logics, and let $\{\mathbf{F_1}, \mathbf{G_1}\}$ (resp. $\{\mathbf{F_2}, \mathbf{G_2}\}$) be the temporal operators of $\mathbf{T_1}$ (resp. $\mathbf{T_2}$). The formula $\mathbf{F_1 G_2} p$ is a $\mathbf{T_1}(\mathbf{T_2})$-formula, while the formula $\mathbf{F_1 G_2} p \leftrightarrow \mathbf{G_2 F_1} p$ is not.

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

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 $\mathbf{T}(\mathbf{L})$ is obtained by taking the set of semantic clauses of $\mathbf{T}$ and by replacing the clause for proposition letters: "$\mathcal{M}, w \models P$ if and only if $P \in V(w)$, whenever $P \in \mathcal{P}$" by the clause: "$\mathcal{M}, w \models \alpha$ if and only if $g(w) \models_{\mathbf{L}} \alpha$, whenever $\alpha \in ML_{\mathbf{L}}$".
□

Hereafter, we will restrict our attention to temporalized logics such that both the embedding and the embedded logics are temporal logics.

We now introduce a new class of combined automata, called *temporalized automata*, which can be viewed as 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. We first define automata and prove results over sequence structures; then, we generalize definitions and results to tree structures (as a matter of fact, we believe that our machinery can actually be extended to cope with more general structures, such as graphs). We will use the following general definition of sequence automata. Let $\Sigma = \{a, b, \ldots\}$ be a finite alphabet and let $\mathcal{S}(\Sigma)$ be the set of Σ-labeled infinite sequences, that is, structures of the form $(\mathbb{N}, <, V)$, where $(\mathbb{N}, <)$ is the set of natural numbers, together with the usual ordering relation, and $V : \mathbb{N} \rightarrow \Sigma$ is a valuation function mapping natural numbers into symbols in Σ.

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, as we will see, all the achieved results do not rest on any specific acceptance condition). An example of a class of sequence automata is the class of Büchi automata.

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

Temporalized automata over sequence structures can be defined as follows. Let $\mathcal{A}_2$ be a class of sequence automata which accept sequences in $\mathcal{S}(\Sigma)$; moreover, let $\Gamma(\Sigma)$ be a finite alphabet whose symbols $A, B, \ldots$ denote automata in $\mathcal{A}_2$, and let $\mathcal{A}_1$ be a class of sequence automata which accept ($\Gamma(\Sigma)$-labeled infinite) sequences in $\mathcal{S}(\Gamma(\Sigma))$. Given $\mathcal{A}_1$ and $\mathcal{A}_2$ as above, we define a class of temporalized automata $\mathcal{A}_1(\mathcal{A}_2)$ that combine the two component classes of automata 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 $g : \mathbb{N} \to \mathcal{S}(\Sigma)$ is a total function mapping elements of $\mathbb{N}$ into sequences in $\mathcal{S}(\Sigma)$. Automata in $\mathcal{A}_1(\mathcal{A}_2)$ accept objects in $\mathcal{S}(\mathcal{S}(\Sigma))$. The class of temporalized automata $\mathcal{A}_1(\mathcal{A}_2)$ is formally defined as follows.

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 defined as follows. Given $w = (\mathbb{N}, <, g) \in \mathcal{S}(\mathcal{S}(\Sigma))$, a *run* of A on w is function $\sigma : \mathbb{N} \to Q$ 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 w if there exists a run σ of A on w such that $AC(\sigma)$, where AC is the acceptance condition of $\mathcal{A}_1$. The language recognized by A, denoted by $\mathcal{L}(A)$, is the set of elements in $\mathcal{S}(\mathcal{S}(\Sigma))$ accepted by A. □

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 $\mathcal{A}_1$. 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 labeling transition system as A and with the combined acceptance condition of $\mathcal{A}_1(\mathcal{A}_2)$. While A accepts in $\mathcal{S}(\Gamma(\Sigma))$, its *concretization* $A^{\downarrow}$ recognizes in $\mathcal{S}(\mathcal{S}(\Sigma))$. Taking advantage of these notions, the combined acceptance condition for temporalized automata can be rewritten as follows. Let $w = (\mathbb{N}, <, g) \in \mathcal{S}(\mathcal{S}(\Sigma))$. A temporalized automaton A accepts w if and only if there exists $v = (\mathbb{N}, <, V) \in \mathcal{S}(\Gamma(\Sigma))$ such that $v \in \mathcal{L}(A^{\uparrow})$ and, for every $i \in \mathbb{N}$,

$g(i) \in \mathcal{L}(V(i))$. In the following, we will often use this alternative, but equivalent, formulation of the combined acceptance condition for temporalized automata.

We now show that relevant logical properties transfer from component automata to temporalized ones. The following notation will be used to express the relationships between automata and temporal logics. We write $\mathcal{A} \to \mathbf{T}$ to denote the fact that every automaton A in $\mathcal{A}$ can be converted into a formula φ_A in $\mathbf{T}$ such that $\mathcal{L}(A) = \mathcal{M}(\varphi_A)$, where $\mathcal{M}(\varphi_A)$ is the set of models of φ_A. Conversely, we write $\mathbf{T} \to \mathcal{A}$ to denote the fact that every formula φ in $\mathbf{T}$ can be converted into an equivalent automaton in $\mathcal{A}$. Finally, $\mathcal{A} \leftrightarrows \mathbf{T}$ stands for $\mathcal{A} \to \mathbf{T}$ and $\mathbf{T} \to \mathcal{A}$. The *transfer problem* for temporalized automata can be stated as follows. Assuming that the automata classes $\mathcal{A}_1$ and $\mathcal{A}_2$ enjoy a given logical property, does $\mathcal{A}_1(\mathcal{A}_2)$ enjoy that property? We investigate the transfer problem with respect to the following properties of automata:

1. (Effective) *closure* under Boolean operations (union, intersection, and complementation): if $\mathcal{A}_1$ and $\mathcal{A}_2$ are (effectively) closed under Boolean operations, is $\mathcal{A}_1(\mathcal{A}_2)$ (effectively) closed under Boolean operations?
2. *Decidability*: if $\mathcal{A}_1$ and $\mathcal{A}_2$ are decidable, is $\mathcal{A}_1(\mathcal{A}_2)$ decidable?
3. *Expressive equivalence* with respect to temporal logic: if $\mathcal{A}_1 \leftrightarrows \mathbf{T_1}$ and $\mathcal{A}_2 \leftrightarrows \mathbf{T_2}$, does $\mathcal{A}_1(\mathcal{A}_2) \leftrightarrows \mathbf{T_1}(\mathbf{T_2})$?

The following lemma plays a crucial role. It shows that every temporalized automaton is equivalent to a temporalized automaton whose transitions are labeled with automata that form a partition of the set $\mathcal{S}(\Sigma)$ of Σ-labeled sequences. Hence, *different* labels of the 'partitioned automaton' correspond to (automata accepting) *disjoint* sets of Σ-labeled sequences. Moreover, the partitioned automaton can be effectively constructed from the original one. We will see that a similar partition lemma holds for temporalized logics (cf. Lemma 2.9 below).

Lemma 2.6

(Partition lemma for temporalized automata)

Let A be a temporalized automaton in $\mathcal{A}_1(\mathcal{A}_2)$. If $\mathcal{A}_2$ is closed under Boolean operations (union, intersection, and complementation), then there exists a finite alphabet $\Gamma'(\Sigma) \subseteq \mathcal{A}_2$ and a temporalized automaton A' over $\Gamma'(\Sigma)$ such that $\mathcal{L}(A) = \mathcal{L}(A')$ and the set $\{\mathcal{L}(X) \mid X \in \Gamma'(\Sigma)\}$ is a partition of $\mathcal{S}(\Sigma)$. Moreover, if $\mathcal{A}_2$ is effectively closed under Boolean operations and it is decidable, then A' can be effectively computed from A.

Proof

To construct $\Gamma'(\Sigma)$ and A' we proceed as follows. Let $A = (Q, q_0, \Delta, M, \Omega)$ be a temporalized automaton over $\Gamma(\Sigma) = \{X_1, \ldots X_n\} \subseteq \mathcal{A}_2$. For every $1 \leq i \leq n$ and $j \in \{0, 1\}$, let $X_i^j = X_i$ for $j = 0$ and $X_i^j = \mathcal{S}(\Sigma) \setminus X_i$ for $j = 1$. Given $(j_1, \ldots, j_n) \in \{0, 1\}^n$, let $\mathtt{Cap}_{(j_1,\ldots,j_n)} = \bigcap_{i=1}^n X_i^{j_i}$. We define $\Gamma_1(\Sigma)$ as the set of all and only $\mathtt{Cap}_{(j_1,\ldots,j_n)}$ such that $(j_1, \ldots, j_n) \in \{0, 1\}^n$. Since $\mathcal{A}_2$ is closed under Boolean operations, $\Gamma_1(\Sigma) \subseteq \mathcal{A}_2$. Moreover, let $\Gamma_2(\Sigma) = \{X \in \Gamma_1(\Sigma) \mid \mathcal{L}(X) \neq \emptyset\}$. We set $\Gamma'(\Sigma) = \Gamma_2(\Sigma)$, and, for $1 \leq i \leq n$, $\Gamma'_i(\Sigma) = \{X \in \Gamma'(\Sigma) \mid X \cap X_i \neq \emptyset\}$.

Note that $\{\mathcal{L}(X) \mid X \in \Gamma'(\Sigma)\}$ is a partition of $\mathcal{S}(\Sigma)$. Moreover, for every $1 \leq i \leq n$, $\{\mathcal{L}(X) \mid X \in \Gamma'_i(\Sigma)\}$ is a partition of $\mathcal{L}(X_i)$. We define the temporalized automaton $A' = (Q, q_0, \Delta', M, \Omega)$ over $\Gamma'(\Sigma)$, where Δ' contains all and only the triples $(q_1, X, q_2) \in Q \times \Gamma'(\Sigma) \times Q$ such that $X \in \Gamma'_i(\Sigma)$ and $(q_1, X_i, q_2) \in \Delta$ for some $1 \leq i \leq n$. It is not difficult to see that $\mathcal{L}(A) = \mathcal{L}(A')$. $\square$

We now prove the first transfer theorem: closure under Boolean operations transfers from component automata to temporalized ones.

Theorem 2.7
(Transfer of closure under Boolean operations)
Closure under Boolean operations (union, intersection, and complementation) transfers from component automata to temporalized ones: given two classes $\mathcal{A}_1$ and $\mathcal{A}_2$ of automata which are (effectively) closed under Boolean operations, the class $\mathcal{A}_1(\mathcal{A}_2)$ of temporalized automata is (effectively) closed under Boolean operations.

Proof
Let $X, Y \in \mathcal{A}_1(\mathcal{A}_2)$.

Union We must provide an automaton $A \in \mathcal{A}_1(\mathcal{A}_2)$ that recognizes the language $\mathcal{L}(X) \cup \mathcal{L}(Y)$. Define $A = (X^\uparrow \cup Y^\uparrow)^\downarrow$. We show that $\mathcal{L}(A) = \mathcal{L}(X) \cup \mathcal{L}(Y)$. Let $x = (\mathbb{N}, <, g) \in \mathcal{L}(A)$. Hence, there is $y = (\mathbb{N}, <, V) \in \mathcal{L}(A^\uparrow) = \mathcal{L}(X^\uparrow \cup Y^\uparrow) = \mathcal{L}(X^\uparrow) \cup \mathcal{L}(Y^\uparrow)$ such that, for every $i \in \mathbb{N}$, $g(i) \in \mathcal{L}(V(i))$. Suppose $y \in \mathcal{L}(X^\uparrow)$. It follows that $x \in \mathcal{L}(X)$. Hence $x \in \mathcal{L}(X) \cup \mathcal{L}(Y)$. Similarly if $y \in \mathcal{L}(Y^\uparrow)$. Conversely, suppose that $x = (\mathbb{N}, <, g) \in \mathcal{L}(X) \cup \mathcal{L}(Y)$. If $x \in \mathcal{L}(X)$, then there is $y = (\mathbb{N}, <, V) \in \mathcal{L}(X^\uparrow)$ such that, for every $i \in \mathbb{N}$, $g(i) \in \mathcal{L}(V(i))$. Hence, $y \in \mathcal{L}(X^\uparrow) \cup \mathcal{L}(Y^\uparrow) = \mathcal{L}(X^\uparrow \cup Y^\uparrow) = \mathcal{L}(A^\uparrow)$. It follows that $x \in \mathcal{L}(A)$. Similarly if $x \in \mathcal{L}(Y)$.

Complementation We must provide an automaton $A \in \mathcal{A}_1(\mathcal{A}_2)$ that recognizes the language $\mathcal{S}(\mathcal{S}(\Sigma)) \setminus \mathcal{L}(X)$. Given Lemma 2.6, we may assume that $\{\mathcal{L}(Z) \mid Z \in \Gamma(\Sigma)\}$ forms a partition of $\mathcal{S}(\Sigma)$. We define $A = (\mathcal{S}(\Gamma(\Sigma)) \setminus X^\uparrow)^\downarrow$. We show that $\mathcal{L}(A) = \mathcal{S}(\mathcal{S}(\Sigma)) \setminus \mathcal{L}(X)$. Let $x = (\mathbb{N}, <, g) \in \mathcal{L}(A)$. Hence, there exists $y = (\mathbb{N}, <, V) \in \mathcal{L}(A^\uparrow) = \mathcal{S}(\Gamma(\Sigma)) \setminus \mathcal{L}(X^\uparrow)$ such that, for every $i \in \mathbb{N}$, $g(i) \in \mathcal{L}(V(i))$. Suppose, by contradiction, that $x \in \mathcal{L}(X)$. It follows that there exists $z = (\mathbb{N}, <, V') \in \mathcal{L}(X^\uparrow)$ such that, for every $i \in \mathbb{N}$, $g(i) \in \mathcal{L}(V'(i))$. Hence, for every $i \in \mathbb{N}$, $g(i) \in \mathcal{L}(V(i)) \cap \mathcal{L}(V'(i))$. Since, for every $i \in \mathbb{N}$, $\mathcal{L}(V(i)) \cap \mathcal{L}(V'(i)) = \emptyset$ whenever $V(i) \neq V'(i)$, we conclude that $V(i) = V'(i)$. Hence $V = V'$ and thus $y = z$. This is a contradiction since y and z belong to disjoint sets. It follows that $x \in \mathcal{S}(\mathcal{S}(\Sigma)) \setminus \mathcal{L}(X)$.

We now prove the opposite direction. Let $x = (\mathbb{N}, <, g) \in \mathcal{S}(\mathcal{S}(\Sigma)) \setminus \mathcal{L}(X)$. It follows that, for every $y = (\mathbb{N}, <, V) \in \mathcal{L}(X^\uparrow)$, there exists $i \in \mathbb{N}$ such that $g(i) \notin \mathcal{L}(V(i))$. Suppose, by contradiction, that $x \in \mathcal{S}(\mathcal{S}(\Sigma)) \setminus \mathcal{L}(A)$. It follows that, for every $z = (\mathbb{N}, <, V) \in \mathcal{L}(A^\uparrow) = \mathcal{S}(\Gamma(\Sigma)) \setminus \mathcal{L}(X^\uparrow)$, there exists $i \in \mathbb{N}$ such that $g(i) \notin \mathcal{L}(V(i))$. We can conclude that, for every $v = (\mathbb{N}, <, V) \in \mathcal{S}(\Gamma(\Sigma))$, there exists $i \in \mathbb{N}$ such that $g(i) \notin \mathcal{L}(V(i))$. This is a contradiction: since $\{\mathcal{L}(Z) \mid Z \in \Gamma(\Sigma)\}$ forms a partition of $\mathcal{S}(\Sigma)$, for every $i \in \mathbb{N}$, there is $Y_i \in \Gamma(\Sigma)$ such that $g(i) \in \mathcal{L}(Y_i)$. We have that $(\mathbb{N}, <, V')$, with $V'(i) = Y_i$, is an element of $\mathcal{S}(\Gamma(\Sigma))$ and, for every $i \in \mathbb{N}$, $g(i) \in \mathcal{L}(V'(i))$. We conclude that $x \in \mathcal{L}(A)$.

Intersection It follows from closure under union and complementation using De Morgan's laws. □

It is worth noticing that if $A = (X^{\uparrow} \cap Y^{\uparrow})^{\downarrow}$, then $\mathcal{L}(A) \subseteq \mathcal{L}(X) \cap \mathcal{L}(Y)$, while the opposite inclusion $\mathcal{L}(X) \cap \mathcal{L}(Y) \subseteq \mathcal{L}(A)$ does not hold in general. We give a simple counterexample. Let $\Gamma(\Sigma) = \{B, C\}$, $X^{\uparrow}$ be the automaton accepting sequences starting with the symbol B, and $Y^{\uparrow}$ be the automaton accepting strings starting with the symbol C. Then, $\mathcal{L}(X^{\uparrow} \cap Y^{\uparrow}) = \emptyset$ and hence $\mathcal{L}(A) = \emptyset$. Let $\Sigma = \{a, b\}$, B be the automaton accepting sequences with an odd number of symbols a, and C be the automaton recognizing sequences with a prime number of symbols a. $\mathcal{L}(X) \cap \mathcal{L}(Y)$ contains, for instance, a combined structure starting with a sequence with exactly 13 occurrences of symbol a, and hence it is not empty.

We now focus on the problem of establishing whether decidability transfers from component automata to temporalized ones. Given $A \in \mathcal{A}_1(\mathcal{A}_2)$, it is easy to see that a sufficient condition for $\mathcal{L}(A) = \emptyset$ is that $\mathcal{L}(A^{\uparrow}) = \emptyset$. However, this condition is not necessary, since $\mathcal{L}(A) = \emptyset$ may depend on the fact that some $\mathcal{A}_2$-automata labeling A accept the empty language. However, if we know that A is labeled with $\mathcal{A}_2$-automata recognizing non-empty languages, then the condition $\mathcal{L}(A^{\uparrow}) = \emptyset$ is both necessary and sufficient for $\mathcal{L}(A) = \emptyset$. In the following theorem, we take advantage of these considerations to devise an algorithm that checks emptiness for temporalized automata.

Theorem 2.8
(Transfer of decidability)

Decidability transfers from component automata to temporalized ones: given two decidable classes of automata $\mathcal{A}_1$ and $\mathcal{A}_2$, the class $\mathcal{A}_1(\mathcal{A}_2)$ of temporalized automata is decidable.

Proof
Let A be a temporalized automaton in $\mathcal{A}_1(\mathcal{A}_2)$. We describe an algorithm that returns 1 if $\mathcal{L}(A) = \emptyset$ and 0 otherwise.

Step 1 Verify whether $\mathcal{L}(A^{\uparrow}) = \emptyset$ using the algorithm that checks emptiness for $\mathcal{A}_1$. If $\mathcal{L}(A^{\uparrow}) = \emptyset$, then return 1.

Step 2 For every $X \in \Gamma(\Sigma)$, if $\mathcal{L}(X) = \emptyset$ (this test can be performed by exploiting the algorithm that checks emptiness for $\mathcal{A}_2$), then remove every transition of the form (q_1, X, q_2) from the transition relation of A.

Step 3 Let B be the temporalized automaton obtained from A after Step 2. Check, using the emptiness algorithm for $\mathcal{A}_1$, whether $\mathcal{L}(B^{\uparrow}) = \emptyset$. If $\mathcal{L}(B^{\uparrow}) = \emptyset$, then return 1, else return 0.

The algorithm always terminates returning either 1 or 0. We prove that the algorithm returns 1 if and only if $\mathcal{L}(A) = \emptyset$. Suppose that the algorithm returns 1. If $\mathcal{L}(A^{\uparrow}) = \emptyset$, then $\mathcal{L}(A) = \emptyset$. Suppose now that $\mathcal{L}(A^{\uparrow}) \neq \emptyset$ and $\mathcal{L}(B^{\uparrow}) = \emptyset$. Note that $\mathcal{L}(A) = \mathcal{L}(B)$, since B is obtained from A by cutting off automata accepting the empty language. Assume, by contradiction, that there is $x \in \mathcal{L}(A)$. Since $\mathcal{L}(A) =$

$\mathcal{L}(B)$, we have that $x \in \mathcal{L}(B)$. Hence $\mathcal{L}(B)$ in not empty. Since $\mathcal{L}(B^{\uparrow}) = \emptyset$, we have that $\mathcal{L}(B)$ is empty which is a contradiction. Hence $\mathcal{L}(A) = \emptyset$. Suppose now that the algorithm returns 0. Then $\mathcal{L}(B^{\uparrow})$ contains at least one element, say $x = (\mathbb{N}, <, V)$. Since B uses only non-empty $\mathcal{A}_2$-automata as alphabet symbols, we have that, for every $i \in \mathbb{N}$, $\mathcal{L}(V(i)) \neq \emptyset$. Hence $y = (\mathbb{N}, <, g)$, with g such that, for every $i \in \mathbb{N}$, $g(i)$ equals to some element of $\mathcal{L}(V(i))$, is an element of $\mathcal{L}(A)$. Hence $\mathcal{L}(A) \neq \emptyset$ □

Finally, we consider the problem of establishing whether expressive equivalence with respect to temporal logics transfers from component automata to temporalized ones. We first state a partition lemma for temporalized logics. The proof is similar to the one of Lemma 2.6, and thus omitted.

Lemma 2.9
(Partition Lemma for temporalized logics)
Let φ be a temporalized formula of $\mathbf{T_1}(\mathbf{T_2})$ and $\alpha_1, \ldots, \alpha_n$ be the maximal $\mathbf{T_2}$-formulas of φ. Then, there exists a finite set Λ of $\mathbf{T_2}$-formulas such that:

1. the set $\{\mathcal{M}(\alpha) \mid \alpha \in \Lambda\}$ is a partition of $\bigcup_{i=1}^{n} \mathcal{M}(\alpha_i)$, and
2. the formula φ' obtained by replacing every $\mathbf{T_2}$-formula α_i in φ with $\bigvee\{\alpha \mid \alpha \in \Lambda \text{ and } \mathcal{M}(\alpha) \cap \mathcal{M}(\alpha_i) \neq \emptyset\}$ is equivalent to φ, i.e., $\mathcal{M}(\varphi) = \mathcal{M}(\varphi')$.

The following theorem shows that expressive equivalence with respect to temporal logics transfers from component automata to temporalized ones.

Theorem 2.10
(Transfer of expressive equivalence w.r.t. temporal logic)
Expressive equivalence w.r.t. temporal logic transfers from component automata to temporalized ones: if $\mathcal{A}_1 \leftrightarrows \mathbf{T_1}$ $\mathcal{A}_2 \leftrightarrows \mathbf{T_2}$, and $\mathcal{A}_2$ is closed under Boolean operations, then $\mathcal{A}_1(\mathcal{A}_2) \leftrightarrows \mathbf{T_1}(\mathbf{T_2})$.

Proof
We first prove that $\mathcal{A}_1(\mathcal{A}_2) \rightarrow \mathbf{T_1}(\mathbf{T_2})$. Let $A \in \mathcal{A}_1(\mathcal{A}_2)$ be a temporalized automaton over $\Gamma(\Sigma) = \{X_1, \ldots, X_n\} \subseteq \mathcal{A}_2$. We have to find a temporalized formula $\varphi_A \in \mathbf{T_1}(\mathbf{T_2})$ such that $\mathcal{L}(A) = \mathcal{M}(\varphi_A)$. Since $\mathcal{A}_2$ is closed under Boolean operations, by exploiting Lemma 2.6, we may assume that $\{\mathcal{L}(X_1), \ldots, \mathcal{L}(X_n)\}$ partitions $\mathcal{S}(\Sigma)$. Since $\mathcal{A}_1 \rightarrow \mathbf{T_1}$, there exists a translation τ_1 from $\mathcal{A}_1$-automata to $\mathbf{T_1}$-formulas such that, for every $X \in \mathcal{A}_1$, $\mathcal{L}(X) = \mathcal{M}(\tau_1(X))$. Let $\varphi_{A^{\uparrow}} = \tau_1(A^{\uparrow})$. The formula $\varphi_{A^{\uparrow}}$ uses proposition letters in $\{P_{X_1}, \ldots, P_{X_n}\}$. Moreover, since $\mathcal{A}_2 \rightarrow \mathbf{T_2}$, there exists a translation σ_1 from $\mathcal{A}_2$-automata to $\mathbf{T_2}$-formulas such that, for every $X \in \mathcal{A}_2$, $\mathcal{L}(X) = \mathcal{M}(\sigma_1(X))$. For every $1 \leq i \leq n$, let $\varphi_{X_i} = \sigma_1(X_i)$. For every proposition letter P_{X_i} appearing in $\varphi_{A^{\uparrow}}$, replace P_{X_i} by φ_{X_i} in $\varphi_{A^{\uparrow}}$. Let φ_A be the resulting formula. It is immediate to see that $\varphi_A \in \mathbf{T_1}(\mathbf{T_2})$. We prove that $\mathcal{L}(A) = \mathcal{M}(\varphi_A)$.

($\subseteq$) Let $x = (\mathbb{N}, <, g) \in \mathcal{L}(A)$. This implies that there exists $x^{\uparrow} = (\mathbb{N}, <, V) \in \mathcal{S}(\Gamma(\Sigma))$ such that $x^{\uparrow} \in \mathcal{L}(A^{\uparrow})$ and, for every $i \in \mathbb{N}$, $g(i) \in \mathcal{L}(V(i))$. Since $\mathcal{L}(A^{\uparrow}) = \mathcal{M}(\varphi_{A^{\uparrow}})$, we have that $x^{\uparrow} \in \mathcal{M}(\varphi_{A^{\uparrow}})$. We prove that, for every $i \in \mathbb{N}$ and $j \in$

$\{1,\ldots,n\}$, $x^{\uparrow}, i \models P_{X_j}$ if and only if $x, i \models \varphi_{X_j}$. Let $i \in \mathbb{N}$ and $j \in \{1,\ldots,n\}$. We know that $x^{\uparrow}, i \models P_{X_j}$ if and only if $V(i) = X_j$. We first prove that $V(i) = X_j$ if and only if $g(i) \in \mathcal{L}(X_j)$. The left to right direction immediately follows since $g(i) \in \mathcal{L}(V(i))$. We prove the right to left direction by contradiction. Suppose $g(i) \in \mathcal{L}(X_j)$ and $V(i) = X_k \neq X_j$. Hence $g(i) \in \mathcal{L}(V(i)) = \mathcal{L}(X_k)$ and thus $g(i) \in \mathcal{L}(X_j) \cap \mathcal{L}(X_k)$, which is a contradiction, since $\mathcal{L}(X_j) \cap \mathcal{L}(X_k) = \emptyset$. Hence $V(i) = X_j$. Finally, we have that $g(i) \in \mathcal{L}(X_j)$ if and only if $g(i) \in \mathcal{M}(\varphi_{X_j})$ if and only if $x, i \models \varphi_{X_j}$. Summing up, we have that $x^{\uparrow} \in \mathcal{M}(\varphi_{A^{\uparrow}})$ and, for every $i \in \mathbb{N}$ and $j \in \{1,\ldots,n\}$, $x^{\uparrow}, i \models P_{X_j}$ if and only if $x, i \models \varphi_{X_j}$. It follows that $x \in \mathcal{M}(\varphi_A)$.

($\supseteq$) Let $x = (\mathbb{N}, <, g) \in \mathcal{M}(\varphi_A)$. We define $x^{\uparrow} = (\mathbb{N}, <, V) \in \mathcal{S}(\Gamma(\Sigma))$ in such a way that, for every $i \in \mathbb{N}$, $V(i) = X_j$ if and only if $g(i) \in \mathcal{M}(\varphi_{X_j}) = \mathcal{L}(X_j)$. Notice that $V(i)$ is always and univocally defined, since $\{\mathcal{L}(X_1),\ldots,\mathcal{L}(X_n)\}$ partitions $\mathcal{S}(\Sigma)$. We prove that, for every $i \in \mathbb{N}$ and $j \in \{1,\ldots,n\}$, we have that $x^{\uparrow}, i \models P_{X_j}$ if and only if $x, i \models \varphi_{X_j}$. Let $i \in \mathbb{N}$ and $j \in \{1,\ldots,n\}$. We know that $x^{\uparrow}, i \models P_{X_j}$ if and only if $V(i) = X_j$. We first prove that $V(i) = X_j$ if and only if $g(i) \in \mathcal{L}(X_j)$. The left to right direction immediately follows by definition of $x^{\uparrow}$.The right to left direction follows since $\mathcal{L}(X_j) \cap \mathcal{L}(X_k) = \emptyset$ whenever $k \neq j$. Finally, $g(i) \in \mathcal{L}(X_j)$ if and only if $g(i) \in \mathcal{M}(\varphi_{X_j})$ if and only if $x, i \models \varphi_{X_j}$. Summing up, we have that $x^{\uparrow} \in \mathcal{M}(\varphi_{A^{\uparrow}}) = \mathcal{L}(A^{\uparrow})$ and, for every $i \in \mathbb{N}$, $g(i) \in \mathcal{M}(\varphi_{X_j}) = \mathcal{M}(\varphi_{V(i)}) = \mathcal{L}(V(i))$. Therefore, $x \in \mathcal{L}(A)$.

We now prove that $\mathbf{T_1}(\mathbf{T_2}) \rightarrow \mathcal{A}_1(\mathcal{A}_2)$. Let $\varphi \in \mathbf{T_1}(\mathbf{T_2})$ be a temporalized formula. We have to find a temporalized automaton $A_\varphi \in \mathcal{A}_1(\mathcal{A}_2)$ such that $\mathcal{M}(\varphi) = \mathcal{L}(A_\varphi)$. Let $\alpha_1,\ldots,\alpha_n$ be the maximal $\mathbf{T_2}$-formulas of φ. By exploiting Lemma 2.9, we may assume that there exists a finite set Λ of $\mathbf{T_2}$-formulas such that the set $\{\mathcal{M}(\alpha) \mid \alpha \in \Lambda\}$ forms a partition of $\bigcup_{i=1}^{n} \mathcal{M}(\alpha_i)$, and every maximal $\mathbf{T_2}$-formula α_i in φ has the form $\bigvee\{\alpha \mid \alpha \in \Lambda \text{ and } \mathcal{M}(\alpha) \cap \mathcal{M}(\alpha_i) \neq \emptyset\}$.

Let $\varphi^{\uparrow}$ be the formula obtained from φ by replacing every $\mathbf{T_2}$-formula $\alpha \in \Lambda$ appearing in φ with proposition letter P_α and by adding to the resulting formula the conjunct $P_\beta \vee \neg P_\beta$, where β is the $\mathbf{T_2}$-formula $\neg \bigvee_{i=1}^{n} \alpha_i$. Let $\mathcal{Q} = \{P_\alpha \mid \alpha \in \Lambda \cup \{\beta\}\}$ be the set of proposition letters of $\varphi^{\uparrow}$. Since $\mathbf{T_1} \rightarrow \mathcal{A}_1$, there exists a translation τ_2 from $\mathbf{T_1}$-formulas to $\mathcal{A}_1$-automata such that, for every $\psi \in \mathbf{T_1}$, $\mathcal{M}(\psi) = \mathcal{L}(\tau_2(\psi))$. Let $A_{\varphi^{\uparrow}} = \tau_2(\varphi^{\uparrow})$. The automaton $A_{\varphi^{\uparrow}}$ labels its transitions with symbols in $2^{\mathcal{Q}}$. Moreover, since $\mathbf{T_2} \rightarrow \mathcal{A}_2$, there exists a translation σ_2 from $\mathbf{T_2}$-formulas to $\mathcal{A}_2$-automata such that, for every $\psi \in \mathbf{T_2}$, $\mathcal{M}(\psi) = \mathcal{L}(\sigma_2(\psi))$. For every $\alpha \in \Lambda \cup \{\beta\}$, let $A_\alpha = \sigma_2(\alpha)$. Finally, let A_φ be the automaton obtained by replacing every label $X \subseteq \mathcal{Q}$ on a transition of $A_{\varphi^{\uparrow}}$ with the $\mathcal{A}_2$-automaton $\bigcap_{P_\alpha \in X} A_\alpha = \sigma_2(\bigwedge_{P_\alpha \in X} \alpha)$. We have that $A_\varphi \in \mathcal{A}_1(\mathcal{A}_2)$ and $\mathcal{L}(A_\varphi) = \mathcal{M}(\varphi)$. The proof is similar to the case $\mathcal{L}(A) = \mathcal{M}(\varphi_A)$. Notice that to prove this direction we did not use the hypothesis of closure under Boolean operations of $\mathcal{A}_2$. $\square$

The following corollary shows that, whenever $\mathbf{T_1} \rightarrow \mathcal{A}_1$ and $\mathbf{T_2} \rightarrow \mathcal{A}_2$, the decidability problem for $\mathbf{T_1}(\mathbf{T_2})$ can be reduced to the decidability problems for $\mathcal{A}_1$ and $\mathcal{A}_2$.

Corollary 2.11
If $\mathbf{T_1} \to \mathcal{A}_1$, $\mathbf{T_2} \to \mathcal{A}_2$, and both $\mathcal{A}_1$ and $\mathcal{A}_2$ are decidable, then $\mathbf{T_1}(\mathbf{T_2})$ is decidable.

Theorems 2.7, 2.8 and 2.10 hold for automata that operate on finite sequences as well; moreover, they can be immediately generalized to automata on finite and infinite trees (definitions of all these classes of automata can be found in (Thomas 1990)). They remain valid for automata on temporalized structures that mix sequences and trees.

Corollary 2.11 allows one to prove the decidability of many temporalized logics. For instance, it is well-known that QLTL (and all its fragments) over infinite sequences can be embedded into Büchi sequence automata, $\mathrm{QCTL}^*_{\mathrm{k}}$ (and all its fragments) over infinite k-ary trees can be embedded into Rabin k-ary tree automata, and both Büchi sequence and Rabin k-ary tree automata are decidable. Moreover, QLTL (and all its fragments) over finite sequences can be embedded into finite sequence automata, $\mathrm{QCTL}^*_{\mathrm{k}}$ (and all its fragments) over finite k-ary trees can be embedded into finite k-ary tree automata, and both finite sequence and finite k-ary tree automata are decidable. From Corollary 2.11, it follows that any temporalized logic $\mathbf{T_1}(\mathbf{T_2})$, where $\mathbf{T_1}$ and $\mathbf{T_2}$ are (fragments of) QLTL or $\mathrm{QCTL}^*_{\mathrm{k}}$, interpreted over either finite or infinite sequence or tree structures, are decidable. As a matter of fact, the decidability of PLTL(PLTL) over infinite sequences of infinite sequences was already proved in (Finger and Gabbay 1992) following a different approach.

3 Temporalized logics and automata for time granularity

In the following, we use temporalized automata to find the (combined) temporal logic counterparts of the monadic second-order theories of downward and upward layered structures. Both results rest on an alternative view of DULSs and UULSs as infinite sequences of k-ary trees of a suitable form. More precisely, DULSs can be viewed as infinite sequences of infinite k-ary trees, while UULSs can be interpreted as infinite sequences of finite *increasing* k-ary trees. In Section 3.1 we provide the monadic second-order theory of DULSs with an expressively complete and elementarily decidable temporalized logic counterpart by exploiting a temporalization of Büchi and Rabin automata. Then, in Section 3.2, 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. It is worth remarking that, unlike the case of DULSs, the combined model we use to encode an UULS differs from that of temporalization since the innermost submodels are *not* independent from the outermost top-level model.

The monadic second-order language for time granularity $\mathrm{MSO}_{\mathcal{P}}[<, (\downarrow_i)_{i=0}^{k-1}]$ is defined as follows.

Definition 3.1
(Monadic second-order language)
Let $\mathrm{MSO}_{\mathcal{P}}[<, (\downarrow_i)_{i=0}^{k-1}]$ be the second-order language with equality built up as fol-

lows: (i) *atomic formulas* are of the forms $x = y$, $x < y$, $\downarrow_i (x) = y$, $x \in X$ and $x \in P$, where $0 \leq i \leq k-1$, x, y are individual variables, X is a set variable, and $P \in \mathcal{P}$; (ii) *formulas* are built up starting from atomic formulas by means of the Boolean connectives $\neg$ and $\wedge$, and the quantifier $\exists$ ranging over both individual and set variables. □

We interpret $\mathrm{MSO}_{\mathcal{P}}[<, (\downarrow_i)_{i=0}^{k-1}]$ over DULSs and UULSs. For all $i \geq 0$, let $T^i = \{j_i \mid j \geq 0\}$. A $\mathcal{P}$-labeled k-refinable DULS is a tuple $\langle \bigcup_{i\geq 0} T^i, (\downarrow_i)_{i=0}^{k-1}, <, (P)_{P\in\mathcal{P}}\rangle$. Part of a 2-refinable DULS is depicted in Figure 2. A DULS can be viewed as an infinite sequence of complete k-ary infinite trees, each one rooted at a point of T^0. The sets in $\{T^i\}_{i\geq 0}$ are the layers of the trees, $\downarrow_i$ is a projection function such that $\downarrow_i (a_b) = c_d$ if and only if $d = b+1$ and $c = a \cdot k + i$, with $i = 0, \ldots, k-1$, $<$ is a total ordering over $\bigcup_{i\geq 0} T^i$ given by the *preorder* (root-left-right) visit of the nodes (for elements belonging to the same tree) and by the total linear ordering of trees (for elements belonging to different trees), and, for all $P \in \mathcal{P}$, P is the set of points in $\bigcup_{i\geq 0} T^i$ labeled with letter P. A $\mathcal{P}$-labeled k-refinable UULS is a tuple $\langle \bigcup_{i\geq 0} T^i, (\downarrow_i)_{i=0}^{k-1}, <, (P)_{P\in\mathcal{P}}\rangle$. Part of a 2-refinable UULS is depicted in Figure 3. An UULS can be viewed as a k-ary infinite tree generated from the leaves. The sets in $\{T^i\}_{i\geq 0}$ represent the layers of the tree, $\downarrow_i$ is a projection function such that $\downarrow_i (a_0) = \bot$, for all a, 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 $P \in \mathcal{P}$, P is the set of points in $\bigcup_{i\geq 0} T^i$ labeled with letter P. Given a formula $\varphi \in \mathrm{MSO}_{\mathcal{P}}[<, (\downarrow_i)_{i=0}^{k-1}]$, we denote by $\mathcal{M}(\varphi)$ the set of models of φ.

For technical reasons, it is convenient to work with a different, but equivalent, monadic second-order logic over DULSs that replaces the total ordering $<$ by two partial orderings $<_1$ and $<_2$ defined as follows. Let t be a DULS. According to the interpretation of DULSs as tree sequences, we define $x <_1 y$ if and only if x is the root of some tree t_i of t, y is the root of some tree t_j of t, and $i < j$ over natural numbers. Moreover, $x <_2 y$ if and only if y is different from x and y belongs to the tree rooted at x. In a similar way, it is convenient to work with a different, but equivalent, monadic second-order logic over UULSs that replaces the total ordering $<$ with a partial ordering $<_{pre}$ such that $x <_{pre} y$ if and only if y is different from x and y belongs to the tree rooted at x.

3.1 Downward unbounded layered structures

We start with a formalization of the alternative characterization of DULSs as suitable tree sequences given above. Let $\mathcal{T}_k(\mathcal{P})$ be the set of $\mathcal{P}$-labeled infinite k-ary trees. Let $\mathcal{S}(\mathcal{T}_k(\mathcal{P}))$ be the set of infinite sequences of $\mathcal{P}$-labeled infinite k-ary trees, that is, temporalized models $(\mathbb{N}, <, g)$ where $g : \mathbb{N} \rightarrow \mathcal{T}_k(\mathcal{P})$. $\mathcal{P}$-labeled DULSs correspond to tree sequences in $\mathcal{S}(\mathcal{T}_k(\mathcal{P}))$, and vice versa. On the one hand, $\mathcal{P}$-labeled DULS t can be viewed as an infinite sequence of $\mathcal{P}$-labeled infinite k-ary trees, whose i-th tree, denoted by t_i, is the $\mathcal{P}$-labeled tree rooted at the i-th point i_0 of the coarsest domain T^0 of t (cf. Figure 5). Such a sequence can be represented as

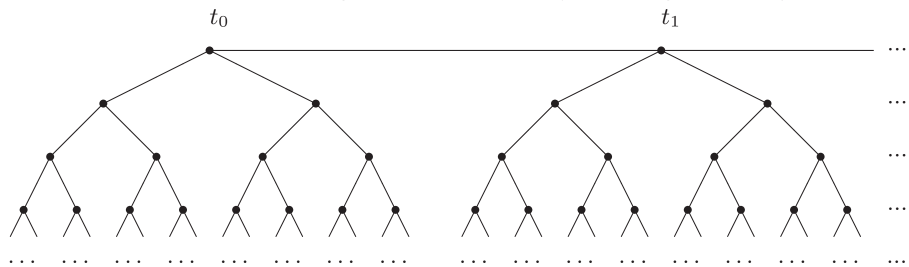

Fig. 5. A tree sequence.

the temporalized model $(\mathbb{N}, <, g) \in \mathcal{S}(\mathcal{T}_k(\mathcal{P}))$ such that, for every $i \in \mathbb{N}$, $g(i) = t_i$. On the other hand, it is immediate to reinterpret infinite sequences of $\mathcal{P}$-labeled infinite k-ary trees in terms of $\mathcal{P}$-labeled DULSs.

Such a correspondence between DULSs and temporalized models enables us to use temporalized logics $\mathbf{T_1}(\mathbf{T_2})$, where $\mathbf{T_1}$ is a linear time logic and $\mathbf{T_2}$ is a branching time logic, to express properties of DULSs. Furthermore, taking advantage of the correspondence between temporalized logic and automata, we can equivalently use temporalized automata $\mathcal{A}_\mathbf{1}(\mathcal{A}_\mathbf{2})$ over DULSs, where $\mathcal{A}_1$ is a class of sequence automata and $\mathcal{A}_2$ is a class of tree automata. In the following, we will focus on the class $\mathcal{B}(\mathcal{R}_k)$ of temporalized automata embedding Rabin k-ary tree automata into Büchi sequence automata. We call automata in this class *infinite tree sequence automata*. Since both $\mathcal{B}$ and $\mathcal{R}_k$ are effectively closed under Boolean operations and decidable, Theorems 2.7 and 2.8 allow us to conclude that the class $\mathcal{B}(\mathcal{R}_k)$ of infinite tree sequence automata is effectively closed under Boolean operations and decidable as well. The complexity of the emptiness problem for infinite tree sequence automata is given by the following theorem.

Theorem 3.2
(Complexity of infinite tree sequence automata)

The emptiness problem for infinite tree sequence automata is decidable in polynomial time in the number of states, and exponential time in the number of accepting pairs.

Proof
For any given $A \in \mathcal{B}(\mathcal{R}_k)$, let n be the number of states of A and N (resp. M) be the maximum number of states (resp. accepting pairs) of a Rabin tree automaton labeling transitions in A. The emptiness of Büchi sequence automata can be checked in polynomial time in the number of states, while the emptiness of Rabin tree automata can be verified in polynomial time in the number of states, and exponential time in the number of accepting pairs. By applying the algorithm used to test the emptiness of temporalized automata in the proof of Theorem 2.8, we have that the complexity of checking whether A accepts the empty language is polynomial in n and N, and exponential in M. $\square$

The following theorem relates infinite tree sequence automata to the monadic second-order theory of DULSs.

Theorem 3.3
(Expressiveness of infinite tree sequence automata)
Infinite tree sequence automata are as expressive as the monadic second-order theory of DULSs.

Proof
The proof can be accomplished following a proof strategy that closely resembles those adopted to prove classical results in the field, such as, for instance, the proof of Büchi's Theorem (cf. (Thomas 1990)). We split it in two parts:

(a) we first show that, for every automaton $A \in \mathcal{B}(\mathcal{R}_k)$ over $\Gamma(\Sigma)$, there exists a formula $\varphi_A \in \mathrm{MSO}_{\mathcal{P}_\Sigma}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ over $\mathcal{P}_\Sigma = \{P_a \mid a \in \Sigma\}$ such that $\mathcal{L}(A) = \mathcal{M}(\varphi_A)$;
(b) then, we show that, for every formula $\varphi \in \mathrm{MSO}_{\mathcal{P}}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ over $\mathcal{P}$, there exists an automaton $A_\varphi \in \mathcal{B}(\mathcal{R}_k)$ over some $\Gamma(2^{\mathcal{P}})$ such that $\mathcal{M}(\varphi) = \mathcal{L}(A_\varphi)$.

We first introduce some auxiliary predicates that can be easily defined in the monadic second-order logic over DULSs. Let $+1$ be a binary predicate such that $+1(x, y)$ if and only if x and y belong to the coarsest domain and y is the immediate successor of x. We will write $x + 1 \in X$ for $\exists y(+1(x, y) \wedge y \in X)$. Moreover, let $\mathtt{T}^0(x)$ be a shorthand for "x belongs to the coarsest domain", $0_0 \in X$ be a shorthand for "the first element of the coarsest domain belongs to X", and $\mathtt{Path}(X, x)$ be a shorthand for the formula stating that "X is a path rooted at x".

Let us prove part (a) for $k = 2$. The generalization to $k > 2$ is straightforward. Let $A = (Q, q_0, \Delta, F)$ be a $\mathcal{B}(\mathcal{R}_2)$-automaton over $\Gamma(\Sigma)$ (finite subset of $\mathcal{R}_2$) accepting tree sequences in $\mathcal{S}(\mathcal{T}_2(\Sigma))$. We produce a sentence $\varphi_A \in \mathrm{MSO}_{\mathcal{P}_\Sigma}[<_1, <_2, \downarrow_0, \downarrow_1]$, that involves monadic predicates in $\mathcal{P}_\Sigma = \{P_a \mid a \in \Sigma\}$ and is interpreted over $\mathcal{S}(\mathcal{T}_2(\Sigma))$, such that $\mathcal{L}(A) = \mathcal{M}(\varphi_A)$. We assume $Q = \{0, \ldots m\}$ and $q_0 = 0$. For every $Z \in \Gamma(\Sigma)$, let $Z = (Q_Z, q_Z^0, \Delta_Z, \Gamma_Z)$ over Σ, with $Q_Z = \{0, \ldots m_Z\}$, $q_Z^0 = 0$, and $\Gamma_Z = \{(L_i^Z, U_i^Z) \mid 1 \leq i \leq r_Z\}$.

The $\mathrm{MSO}_{\mathcal{P}_\Sigma}[<_1, <_2, \downarrow_0, \downarrow_1]$-sentence φ_A that corresponds to the automaton A basically encodes the combined acceptance condition for $\mathcal{B}(\mathcal{R}_2)$-automata. The outermost part of the sentence expresses the existence of an accepting run over the coarsest layer of the tree sequence for the Büchi sequence automaton $A^{\uparrow}$. For all $i \in Q$, the second-order variable X_i denotes the set of positions of the run which are associated with the state i, while, for all $Z \in \Gamma(\Sigma)$ the monadic predicate Q_Z denotes the set of positions of the run that are labeled with the Rabin tree automaton Z. The innermost part $\mathtt{RAC}(x, Z)$ captures the existence of an accepting run over the tree rooted at x for the Rabin tree automaton Z. For $i \in Q_Z$, the second-order variable Y_i denotes the set of positions of the run that are associated with state i. The sentence φ_A is defined as follows:

$$
\begin{aligned}
&(\exists Q_Z)_{Z \in \Gamma(\Sigma)} (\exists X_i)_{i=0}^{m} (\textstyle\bigwedge_{i=0}^{m} \forall x(x \in X_i \rightarrow \mathtt{T}^0(x)) \wedge \\
&\textstyle\bigwedge_{Z \in \Gamma(\Sigma)} \forall x(x \in Q_Z \rightarrow \mathtt{T}^0(x)) \wedge 0_0 \in X_0 \wedge \bigwedge_{i \neq j} \neg \exists y(y \in X_i \wedge y \in X_j) \wedge \\
&\forall x(\mathtt{T}^0(x) \rightarrow \textstyle\bigvee_{(i,Z,j) \in \Delta} (x \in X_i \wedge x \in Q_Z \wedge x + 1 \in X_j)) \wedge \\
&\textstyle\bigvee_{i \in F} \forall x(\mathtt{T}^0(x) \rightarrow \exists y(\mathtt{T}^0(y) \wedge x <_1 y \wedge y \in X_i)) \wedge \\
&\textstyle\bigwedge_{Z \in \Gamma(\Sigma)} \forall x(x \in Q_Z \rightarrow \mathtt{RAC}(x, Z)),
\end{aligned}
$$

where $\mathtt{RAC}(x, Z)$ stands for:

$(\exists Y_i)_{i=0}^{m_Z}(\bigwedge_{i=0}^{m_Z} \forall y(y \in Y_i \rightarrow x \leq_2 y) \wedge x \in Y_0 \wedge \bigwedge_{i \neq j} \neg \exists y(y \in Y_i \wedge y \in Y_j) \wedge$
$\forall y(x \leq_2 y \rightarrow \bigvee_{(i,a,j_0,j_1) \in \Delta_Z}(y \in Y_i \wedge y \in P_a \wedge \downarrow_0 (y) \in Y_{j_0} \wedge \downarrow_1 (y) \in Y_{j_1})) \wedge$
$\forall W(\mathtt{Path}(W, x) \rightarrow \bigvee_{i=0}^{r_Z}(\bigwedge_{j \in L_i^Z} \exists u(u \in W \wedge \forall v(v \in W \wedge u <_2 v \rightarrow v \notin Y_j)) \wedge$
$\bigvee_{j \in U_i^Z} \forall u(u \in W \rightarrow \exists v(v \in W \wedge u <_2 v \wedge v \in Y_j))))).$

We now prove part (b). Let $\mathcal{P} = \{P_1, \ldots P_n\}$. To simplify things, we prove our result for the theory $\mathrm{MSO}_{\mathcal{P}}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}, +1]$ which can be easily shown to be equivalent to $\mathrm{MSO}_{\mathcal{P}}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$. Given a formula $\varphi \in \mathrm{MSO}_{\mathcal{P}}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}, +1]$, that involves monadic predicates in $\mathcal{P}$ and is interpreted over $\mathcal{P}$-labeled tree sequences in $\mathcal{S}(\mathcal{T}_k(\mathcal{P}))$, we build an automaton $A_\varphi \in \mathcal{B}(\mathcal{R}_k)$ over some $\Gamma(2^{\mathcal{P}})$ and accepting in $\mathcal{S}(\mathcal{T}_k(\mathcal{P}))$ such that $\mathcal{L}(A_\varphi) = \mathcal{M}(\varphi)$.

As a first step, we show that the ordering relations $<_1$ and $<_2$ can actually be removed without reducing the expressiveness. We replace $x <_1 y$ by

$$\mathtt{T}^0(x) \wedge \mathtt{T}^0(y) \wedge \forall X(x+1 \in X \wedge \forall z(z \in X \rightarrow z+1 \in X) \rightarrow y \in X)),$$

and $x <_2 y$ by

$$\forall X(\bigwedge_{i=0}^{k-1} \downarrow_i (x) \in X \wedge \forall z(z \in X \rightarrow \bigwedge_{i=0}^{k-1} \downarrow_i (z) \in X) \rightarrow y \in X).$$

Hence, $\mathrm{MSO}_{\mathcal{P}}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}, +1]$ is as expressive as $\mathrm{MSO}_{\mathcal{P}}[(\downarrow_i)_{i=0}^{k-1}, +1]$. Next, we introduce an expressively equivalent variant of $\mathrm{MSO}_{\mathcal{P}}[(\downarrow_i)_{i=0}^{k-1}, +1]$, denoted by $\mathrm{MSO}[(\downarrow_i)_{i=0}^{k-1}, +1]$, which uses free set variables X_i in place of predicate symbols P_i and is interpreted over $\{0,1\}^n$-labeled tree sequences in $\mathcal{S}(\mathcal{T}_k(\{0,1\}^n))$. The idea is to encode a set $X \subseteq \mathcal{P}$ with the string $i_1 \ldots i_n \in \{0,1\}^n$ such that, for $j = 1, \ldots, n$, $i_j = 1$ if and only if $P_j \in X$. We now reduce $\mathrm{MSO}[(\downarrow_i)_{i=0}^{k-1}, +1]$ to a simpler formalism $\mathrm{MSO}_0[(\downarrow_i)_{i=0}^{k-1}, +1]$, where *only* second-order variables X_i occur and atomic formulas are of the forms $X_i \subseteq X_j$ (X_i is a subset of X_j), $\mathtt{Proj}_m(X_i, X_j)$, with $m = 0, \ldots, k-1$ (X_i and X_j are the singletons $\{x\}$ and $\{y\}$, respectively, and $\downarrow_m (x) = y$), and $\mathtt{Succ}(X_i, X_j)$ (X_i and X_j are the singletons $\{x\}$ and $\{y\}$, respectively, and $x + 1 = y$). This step is performed as in the proof of Büchi's Theorem. Finally, given a $\mathrm{MSO}_0[(\downarrow_i)_{i=0}^{k-1}, +1]$-formula $\varphi(X_1, \ldots, X_n)$, we prove, by induction on the structural complexity of φ, that there exists a temporalized automaton A_φ accepting in $\mathcal{S}(\mathcal{T}_k(\{0,1\}^n))$ such that $\mathcal{M}(\varphi) = \mathcal{L}(A_\varphi)$. A corresponding automaton accepting in $\mathcal{S}(\mathcal{T}_k(\mathcal{P}))$ can be obtained in the obvious way. As for atomic formulas, let $\alpha_{i,j}$ be the Rabin tree automaton over $\{0,1\}^n$ for $X_i \subseteq X_j$. The temporalized automaton for $X_i \subseteq X_j$ is depicted in Figure 6 (top). Moreover, let ζ be the Rabin tree automaton over $\{0,1\}^n$ that accepts the singleton set containing a tree labeled with 0^n everywhere, and let $\alpha_{i,j}^m$ be the Rabin tree automaton over $\{0,1\}^n$ for $\mathtt{Proj}_m(X_i, X_j)$. The temporalized automaton for $\mathtt{Proj}_m(X_i, X_j)$ is depicted in Figure 6 (middle). Finally, let α_i be the Rabin tree automaton over $\{0,1\}^n$ that accepts the singleton set containing a tree labeled with $0^{i-1}10^{n-i}$ at the root, and labeled with 0^n elsewhere. The combined automaton for $\mathtt{Succ}(X_i, X_j)$ is depicted in Figure 6 (bottom). The induction step immediately follows from the closure of

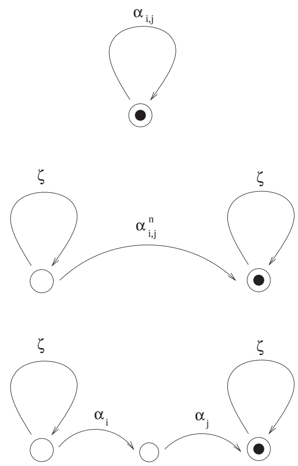

Fig. 6. Temporalized automata for atomic formulas.

$\mathcal{B}(\mathcal{R}_k)$ automata under Boolean operations and projection. Closure under Boolean operations has been already shown; closure under projection can be argued as follows: given a $\mathcal{B}(\mathcal{R}_k)$-automaton A, the corresponding projected $\mathcal{B}(\mathcal{R}_k)$-automaton is obtained by simply projecting every Rabin automaton that labels some transition of A. $\square$

We can exploit infinite tree sequence automata to provide the (full) second-order theory of DULSs with an expressively complete and elementarily decidable temporal logic counterpart. First of all, it is well-known that $\mathcal{B} \leftrightarrows$ QLTL and $\mathcal{B} \leftrightarrows$ EQLTL, as well as $\mathcal{R}_k \leftrightarrows \text{QCTL}^*_\text{k}$ and $\mathcal{R}_k \leftrightarrows \text{EQCTL}^*_\text{k}$ (Emerson 1990). Since Rabin tree automata are closed under Boolean operations, Theorem 2.10 allows us to conclude that both $\text{QLTL}(\text{QCTL}^*_\text{k}) \leftrightarrows \mathcal{B}(\mathcal{R}_k)$ and $\text{EQLTL}(\text{EQCTL}^*_\text{k}) \leftrightarrows \mathcal{B}(\mathcal{R}_k)$[3]. By applying Theorem 3.3, we have that both $\text{QLTL}(\text{QCTL}^*_\text{k}) \leftrightarrows \text{MSO}_\mathcal{P}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ and $\text{EQLTL}(\text{EQCTL}^*_\text{k}) \leftrightarrows \text{MSO}_\mathcal{P}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$. Such a result is summarized by the following theorem.

Theorem 3.4
(Expressiveness of $\text{QLTL}(\text{QCTL}^*_\text{k})$ and $\text{EQLTL}(\text{EQCTL}^*_\text{k})$)
$\text{QLTL}(\text{QCTL}^*_\text{k})$ and $\text{EQLTL}(\text{EQCTL}^*_\text{k})$ are as expressive as $\text{MSO}_\mathcal{P}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$, when interpreted over DULSs.

Furthermore, since $\text{MSO}_\mathcal{P}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ is decidable, both $\text{QLTL}(\text{QCTL}^*_\text{k})$ and $\text{EQLTL}(\text{EQCTL}^*_\text{k})$ are decidable. The next theorem shows that $\text{EQLTL}(\text{EQCTL}^*_\text{k})$ is *elementarily* decidable.

[3] It is worth pointing out that the application of the partition step of Theorem 2.10 to temporal formulas in EQCTL^*_k generates formulas of the form $\neg\exists Q_1 \ldots \exists Q_n \varphi$, where φ is a CTL^*_k-formula, which do not belong to the language of EQCTL^*_k, because such a language is not closed under negation. Nevertheless, formulas of the form $\neg\exists Q_1 \ldots \exists Q_n \varphi$ can be embedded into Rabin tree automata as well. The Rabin tree automaton for $\neg\exists Q_1 \ldots \exists Q_n \varphi$ can indeed be obtained by taking the complementation of the projection, with respect to $Q_1, \ldots Q_n$, of the Rabin tree automaton for φ.

Theorem 3.5
(Complexity of EQLTL(EQCTL$^*_\mathrm{k}$))
The satisfiability problem for EQLTL(EQCTL$^*_\mathrm{k}$) over DULSs is in ELEMENTARY.

Proof
EQLTL(EQCTL$^*_\mathrm{k}$) can be decided by embedding it into $\mathcal{B}(\mathcal{R}_k)$ automata (such an embedding can be accomplished following the approach outlined in the proof of Theorem 2.10). EQLTL can be elementarily embedded into Büchi sequence automata. Indeed, given an EQLTL-formula $\exists Q_1 \ldots \exists Q_n \varphi$, the PLTL-formula φ can be converted into a Büchi sequence automaton A_φ of size $O(2^{|\varphi|})$. A Büchi sequence automaton for $\exists Q_1 \ldots \exists Q_n \varphi$ can be obtained by taking the projection of A_φ with respect to letters $Q_1, \ldots, Q_n$, that is, by deleting letters $Q_1, \ldots, Q_n$ from the transitions of A_φ. The size of the resulting automaton is $O(2^{|\varphi|})$. Similarly, EQCTL$^*_\mathrm{k}$ formulas can be embedded into Rabin tree automata with a doubly exponential number of states and a singly exponential number of accepting pairs in the length of the formula. In particular, as already pointed out, a Rabin tree automaton for formulas of the form $\neg\exists Q_1 \ldots \exists Q_n \varphi$, which are generated by applying the partition step of Theorem 2.10 to EQCTL$^*_\mathrm{k}$ formulas, can be obtained by taking the complementation of the projection, with respect to $Q_1, \ldots Q_n$, of the Rabin tree automaton for φ. The resulting automaton has elementary size. Hence, any EQLTL(EQCTL$^*_\mathrm{k}$) formula can be converted into an equivalent $\mathcal{B}(\mathcal{R}_k)$ automaton of elementary size. Since $\mathcal{B}(\mathcal{R}_k)$ automata are elementarily decidable, we have the thesis. □

We conclude the section by giving some examples of meaningful timing properties that can be expressed in (fragments of) EQLTL(EQCTL$^*_\mathrm{k}$) interpreted over DULSs. As a first example, consider the property 'P densely holds at some node x' meaning that there exists a path rooted at x such that P holds at each node of the path (notice that such a property implies that, for every $i \geq 0$, there exists $y \in \downarrow^i (x)$ such that P holds at y, where, for $i \geq 0$, $\downarrow^i (x)$ is the i-th layer of the tree rooted at x, but not vice versa). This property can be expressed in PLTL(CTL$^*_\mathrm{k}$) by the formula:

$$\Diamond \mathbf{EFEG} P.$$

As another example, the property 'P holds at the origin of every layer' (or, equivalently, 'P holds along the leftmost path of the first tree of the sequence') can be expressed in PLTL(CTL$^*_\mathrm{k}$) as follows:

$$\mathbf{E}(P \wedge \mathbf{GX_0} P).$$

As a third example, the property 'P holds everywhere on every even tree' can be encoded in EQLTL(CTL$^*_\mathrm{k}$) as follows:

$$\exists Q (Q \wedge \bigcirc \neg Q \wedge \Box (Q \leftrightarrow \bigcirc \bigcirc Q) \wedge \Box (Q \rightarrow \mathbf{AG} P)).$$

Notice that such a property cannot be expressed in PLTL(CTL$^*_\mathrm{k}$), since, as it is well-known, PLTL cannot express the property 'P holds on every even point' (Wolper 1983). As a last example, the property 'P holds everywhere on every even layer'

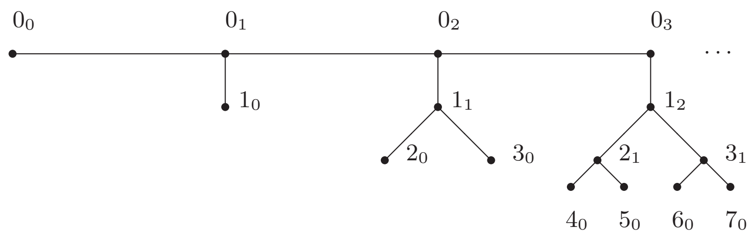

Fig. 7. Mapping an UULS into an increasing tree sequence.

can be encoded in PLTL(EQCTL$^*_\mathrm{k}$) as follows:

$$\Box\exists Q(Q \wedge \mathbf{AX}\neg Q \wedge \mathbf{AG}(Q \leftrightarrow \mathbf{AXAX}Q) \wedge \mathbf{AG}(Q \rightarrow P)).$$

Notice that also this property cannot be expressed in PLTL(CTL$^*_\mathrm{k}$).

Unfortunately, things are not always that easy. As an example, the property 'P holds at exactly one node' can be easily encoded in (the first-order fragment of) $\mathrm{MSO}_\mathcal{P}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ by the formula: $\exists x(x \in P \wedge \forall y(y \neq x \rightarrow y \notin P))$, while it is not easy at all to express it in EQLTL(EQCTL$^*_\mathrm{k}$). Moreover, since $\mathrm{MSO}_\mathcal{P}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ is nonelementarily decidable, while EQLTL(EQCTL$^*_\mathrm{k}$) is elementarily decidable, the translation τ of $\mathrm{MSO}_\mathcal{P}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$ formulas into EQLTL(EQCTL$^*_\mathrm{k}$) formulas is nonelementary. This means that, for every $n \in \mathbb{N}$, there exists an $\mathrm{MSO}_\mathcal{P}[<_1, <_2, (\downarrow_i)_{i=0}^{k-1}]$-formula φ such that the length of $\tau(\varphi)$ is greater than $\kappa(n, |\varphi|)$ (an exponential tower of height n).

3.2 Upward unbounded layered structures

We start by giving an alternative characterization of UULSs in terms of tree sequences. To this end, we need to introduce the notions of almost k-ary tree and of increasing tree sequence. An *almost k-ary finite tree* is a complete finite tree whose root has exactly $k-1$ sons $0, \ldots, k-2$, each of them is the root of a complete finite k-ary tree. Let $\mathcal{H}_k(\mathcal{P})$ be the set of $\mathcal{P}$-labeled almost k-ary finite trees. A $\mathcal{P}$-labeled *increasing k-ary tree sequence* (ITS, for short) is a tree sequence such that, for every $i \in \mathbb{N}$, the i-th tree of the sequence is a $\mathcal{P}$-labeled almost k-ary tree of height i (cf. Figure 7). A $\mathcal{P}$-labeled ITS can be represented as a temporalized model $(\mathbb{N}, <, g)$ such that, for every $i \in \mathbb{N}$, $g(i)$ is the i-th tree of the sequence. Let $ITS_k(\mathcal{P})$ be the set of $\mathcal{P}$-labeled k-ary ITSs. It is worth noting that $ITS_k(\mathcal{P})$ is *not* the class $\mathcal{H}_k(\mathcal{P}$ of temporalized models embedding almost k-ary finite trees into infinite sequences: an increasing tree sequence is a particular sequence of almost k-ary finite trees, but a sequence of almost k-ary finite trees is not necessary increasing, and thus $ITS_k(\mathcal{P}) \subsetneq \mathcal{S}(\mathcal{H}_k(\mathcal{P}))$.

It is not difficult to show that a $\mathcal{P}$-labeled UULS corresponds to a $\mathcal{P}$-labeled ITS, and vice versa. As already pointed out, an UULS can be viewed as an infinite complete k-ary tree generated from the leaves. The corresponding tree sequence can be obtained starting from the first point of the finest layer of the UULS and climbing up along the leftmost path of the structure. The i-th tree in the sequence is obtained by taking the tree rooted at the i-th point of the leftmost path, and by deleting from it the subtree rooted at the leftmost son of its root. More precisely, let t be a k-ary UULS. For every node x in t, we define t_x to be the finite complete k-ary tree rooted at x. For every $i \geq 0$, let $\hat{t}_{0_i}$ be the almost k-ary finite tree obtained from t_{0_i} by deleting, whenever $i > 0$, the subtree $t_{0_{i-1}}$ from it. The ITS $(\mathbb{N}, <, g)$ associated with the UULS t is obtained by defining, for every $i \geq 0$, $g(i) = \hat{t}_{0_i}$. The embedding of a binary UULS into a binary ITS is depicted in Figure 7. Similarly, ITSs can be reinterpreted in terms of UULSs.

On the basis of such a correspondence between UULSs and ITSs, we can use temporalized logics $\mathbf{T_1}(\mathbf{T_2})$, where $\mathbf{T_1}$ is a linear time logic and $\mathbf{T_2}$ is a branching time logic, to express properties of UULSs. More precisely, we interpret $\mathbf{T_1}(\mathbf{T_2})$ over $\mathcal{S}(\mathcal{H}_k(\mathcal{P}))$, but, since we are interested in increasing tree sequences, we study the logical properties of $\mathbf{T_1}(\mathbf{T_2})$, such as expressiveness and decidability, with respect to the proper subset $ITS_k(\mathcal{P})$. Temporalized automata $\mathcal{A}_\mathbf{1}(\mathcal{A}_\mathbf{2})$ over UULSs can be defined in a similar way. Once again, we consider automata in $\mathcal{A}_\mathbf{1}(\mathcal{A}_\mathbf{2})$ accepting in $\mathcal{S}(\mathcal{H}_k(\Sigma))$, but, since we are interested in increasing tree sequences, we study the relevant properties of $\mathcal{A}_\mathbf{1}(\mathcal{A}_\mathbf{2})$, such as closure under Boolean operations, expressiveness, and decidability, with respect to the proper subset $ITS_k(\Sigma)$. In the following, we will focus on the class $\mathcal{B}(\mathcal{C}_k)$ of temporalized automata embedding almost k-ary finite tree automata into Büchi sequence automata. We call automata in $\mathcal{B}(\mathcal{C}_k)$ *finite tree sequence automata*.

Since both $\mathcal{B}$ and $\mathcal{C}_k$ are effectively closed under Boolean operations and decidable, Theorems 2.7 and 2.8 allows us to conclude that $\mathcal{B}(\mathcal{C}_k)$ is effectively closed under Boolean operations and decidable. We show that $\mathcal{B}(\mathcal{C}_k)$-automata are closed under Boolean operations over the set $ITS_k(\Sigma)$ as well. Let $A, B \in \mathcal{B}(\mathcal{C}_k)$. We show that:

- there exists $C \in \mathcal{B}(\mathcal{C}_k)$ such that

$$\mathcal{L}(C) \cap ITS_k(\Sigma) = ITS_k(\Sigma) \setminus \mathcal{L}(A) \quad (\textit{complementation});$$

- there exists $C \in \mathcal{B}(\mathcal{C}_k)$ such that

$$\mathcal{L}(C) \cap ITS_k(\Sigma) = (\mathcal{L}(A) \cup \mathcal{L}(B)) \cap ITS_k(\Sigma) \quad (\textit{union});$$

- there exists $C \in \mathcal{B}(\mathcal{C}_k)$ such that

$$\mathcal{L}(C) \cap ITS_k(\Sigma) = (\mathcal{L}(A) \cap \mathcal{L}(B)) \cap ITS_k(\Sigma) \quad (\textit{intersection}).$$

As it can be easily checked, it suffices to set $C = \overline{A}$ in case of complementation, $C = A \cup B$ in the case of union, and $C = A \cap B$ in the case of intersection.

The following theorem relates finite tree sequence automata to the monadic second-order theory of UULSs.

Theorem 3.6

(Expressiveness of finite tree sequence automata)

Finite tree sequence automata are as expressive as the monadic second-order theory of UULSs.

Proof

The proof is quite similar to that of Theorem 3.3, and thus we only sketch its main steps. We split the proof in two parts:

(a) we first show that, for every automaton $A \in \mathcal{B}(\mathcal{C}_k)$ over $\Gamma(\Sigma)$, there exists a formula $\varphi_A \in \text{MSO}_{\mathcal{P}_\Sigma}[<, (\downarrow_i)_{i=0}^{k-1}]$ over $\mathcal{P}_\Sigma = \{P_a \mid a \in \Sigma\}$ such that $\mathcal{L}(A) \cap ITS_k(\Sigma) = \mathcal{M}(\varphi_A)$;

(b) then we show that, for every formula $\varphi \in \text{MSO}_{\mathcal{P}}[<, (\downarrow_i)_{i=0}^{k-1}]$, there exists an automaton $A_\varphi \in \mathcal{B}(\mathcal{C}_k)$ over some $\Gamma(2^{\mathcal{P}})$ such that $\mathcal{M}(\varphi) = \mathcal{L}(A_\varphi) \cap ITS_k(\mathcal{P})$.

The embedding of automata into formulas is performed by encoding the combined acceptance condition for $\mathcal{B}(\mathcal{C}_k)$-automata into $\text{MSO}_{\mathcal{P}}[<, (\downarrow_i)_{i=0}^{k-1}]$. The Büchi acceptance condition have to be implemented over the leftmost path of the structure, and the finite tree automata acceptance condition have to be constrained to hold over almost k-ary trees rooted at nodes in the leftmost path of the structure. The embedding of formulas into automata takes advantage of the closure properties of $\mathcal{B}(\mathcal{C}_k)$-automata over UULSs. □

We can exploit finite tree sequence automata to provide the (full) second-order theory of UULSs with an expressively complete temporal logic counterpart. We know that $\mathcal{B} \leftrightarrows$ QLTL and $\mathcal{B} \leftrightarrows$ EQLTL, and that $\mathcal{C}_k \leftrightarrows \text{QCTL}^*_\text{k}$ and $\mathcal{C}_k \leftrightarrows \text{EQCTL}^*_\text{k}$. Since almost k-ary finite tree automata are closed under Boolean operations, Theorem 2.10 allows us to conclude that that $\text{QLTL}(\text{QCTL}^*_\text{k}) \leftrightarrows \mathcal{B}(\mathcal{C}_k)$ and $\text{EQLTL}(\text{EQCTL}^*_\text{k}) \leftrightarrows \mathcal{B}(\mathcal{C}_k)$ over infinite sequences of almost k-ary finite trees. Since increasing k-ary tree sequences are infinite sequences of almost k-ary trees, the above equivalences hold over increasing k-ary tree sequences as well. From Theorem 3.6, we have that $\text{QLTL}(\text{QCTL}^*_\text{k}) \leftrightarrows \text{MSO}_{\mathcal{P}}[<_{pre}, (\downarrow_i)_{i=0}^{k-1}]$ and $\text{EQLTL}(\text{EQCTL}^*_\text{k}) \leftrightarrows \text{MSO}_{\mathcal{P}}[<_{pre}, (\downarrow_i)_{i=0}^{k-1}]$. Such a result is summarized by the following theorem.

Theorem 3.7

(Expressiveness of $\text{QLTL}(\text{QCTL}^*_\text{k})$ and $\text{EQLTL}(\text{EQCTL}^*_\text{k})$)

$\text{QLTL}(\text{QCTL}^*_\text{k})$ and $\text{EQLTL}(\text{EQCTL}^*_\text{k})$ are as expressive as $\text{MSO}_{\mathcal{P}}[<_{pre}, (\downarrow_i)_{i=0}^{k-1}]$, when interpreted over UULSs.

The (nonelementary) decidability of $\text{QLTL}(\text{QCTL}^*_\text{k})$ and $\text{EQLTL}(\text{EQCTL}^*_\text{k})$ immediately follows from that of $\text{MSO}_{\mathcal{P}}[<_{pre}, (\downarrow_i)_{i=0}^{k-1}]$ over UULSs. A natural question arises at this point: is $\text{EQLTL}(\text{EQCTL}^*_\text{k})$ elementary decidable as in the case of DULSs? In order to answer this question, we study the decidability and complexity of the emptiness problem for finite tree sequence automata over increasing k-ary tree sequences. Such a problem can be formulated as follows: given an automaton $A \in \mathcal{B}(\mathcal{C}_k)$, is there an increasing k-ary tree sequence accepted by A? (Equivalently, does $\mathcal{L}(A) \cap ITS_k(\Sigma) \neq \emptyset$?) The (nonelementary) decidability of such a problem immediately follows from Theorem 3.6, 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 *elementary* time.

Let $A = (Q, q_0, \Delta, F)$ be an automaton in $\mathcal{B}(\mathcal{C}_k)$ over the alphabet $\Gamma(\Sigma)$ (finite subset of $\mathcal{C}_k$). 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 $\mathcal{C}_k$-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. $q_j \in F$;
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 almost k-ary finite 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 3.8 shows that condition 3' is elementarily decidable.

Lemma 3.8
Let X be a almost k-ary 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 $X = (Q, q_0, \Delta, F)$ 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 almost k-ary finite tree automata are elementarily closed under Boolean operations and 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 almost k-ary 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_0 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):

Given a finite directed graph $G = (N, E)$, two nodes $q_1, q_2 \in N$, and two natural numbers $a, l \geq 0$, the question is: for every $y \geq 0$, is there a path in G from q_1 to q_2 of length $a + y \cdot l$?

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 $\pi^{\circlearrowleft}$ the path obtained by eliminating cyclic subpaths from π. That is, if π is acyclic, then $\pi^{\circlearrowleft} = \pi$. Else, if $\pi = \alpha q' \beta q' \gamma$, then $\pi^{\circlearrowleft} = \alpha^{\circlearrowleft} q' \gamma^{\circlearrowleft}$. 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 $\pi_1^{\circlearrowleft} = \pi_2^{\circlearrowleft}$. 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 for the presence of some cyclic subpaths. More precisely, let μ be the shortest path in $[\pi]_{\sim_{q_1,q_2}}$, let $C_1, \ldots C_n$ be the cycles intersecting π, and let $w_1, \ldots w_n$ be their respective lengths. Note that 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 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 $[\pi_1]_{\sim_{q_1,q_2}}, \ldots, [\pi_m]_{\sim_{q_1,q_2}}$ be the equivalence classes of $\sim_{q_1,q_2}$. For every $j = 1, \ldots m$, let μ_j be the shortest path in $[\pi_j]_{\sim_{q_1,q_2}}$, let $C_1^j, \ldots C_n^j$ be the the cycles intersecting π_j, and let $w_1^j, \ldots w_n^j$ be their respective lengths. Moreover, let

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

The PPP reduces to the following problem of Number Theory:

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

We now solve the latter problem. Let $w_i \geq 0$, for $i = 1, \ldots n$. 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 $W = (w_1, \ldots w_n)$ and let $d = GCD(W)$ (the greatest common divisor of $\{w_1, \ldots, w_n\}$). 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 $\{w_1, \ldots w_n\}$) 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, $GCD(w_1/d, \ldots w_n/d) = 1$ and hence, as above, $S' = E' \cup \{j \mid j \geq k'\}$ for some finite set E' and some $k' \in \mathbb{N}$. Therefore, in this case,

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

where d `DIV j` means that d is a divisor of j.

Summing up, in any case, the set S can be described 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 $d = GCD(W)$. In other words, the set S is 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 $Y = \{y \geq 0 \mid \exists x_1, \ldots x_n \geq 0\,(\sum_{i=1}^{n} x_i \cdot w_i = y \cdot l)\}$ in a similar way. Let $e = GCD(d, l)$, $l = l' \cdot e$ and $d = d' \cdot e$. We have that:

$y \in Y$	iff
$y \cdot l \in S$	iff
$y \cdot l \in E \,\vee\, y \cdot l \geq k \,\wedge\, d\ \mathtt{DIV}\ \mathtt{y \cdot l}$	iff
$y \cdot l \in E \,\vee\, y \geq \lceil k/l \rceil \,\wedge\, d' \cdot e\ \mathtt{DIV}\ \mathtt{y \cdot l' \cdot e}$	iff
$y \cdot l \in E \,\vee\, y \geq \lceil k/l \rceil \,\wedge\, d'\ \mathtt{DIV}\ \mathtt{y}$	

Therefore, the set Y is the union of a finite and computable set and an arithmetic progression, i.e.,

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

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

We have shown that, for $i = 1, \ldots, m$, every Y_i has the form $E_i \cup \{k_i + y \cdot d_i \mid y \geq 0\}$ for some finite E_i, and some $k_i, d_i \in \mathbb{N}$. We now give a solution to the problem $\bigcup_{i=1}^{m} Y_i = \mathbb{N}$. Let $k_r = min\{k_1, \ldots, k_m\}$ and $D = LCM(d_1, \ldots, d_m)$ (the least common multiple of $\{d_1, \ldots, d_m\}$). The algorithm works as follows: for every $k < k_r$, we check whether $k \in Y_i$ for some $i = 1, \ldots, m$. If this is not the case, the problem has no solution. Otherwise, we verify whether, for every $j = 0, \ldots, D-1$, $k_r + j \in Y_i$ for some $i = 1, \ldots, m$. 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 $u, v \in \{1, \ldots m\}^*$, 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 $Y_1, \ldots, Y_m$, 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 $k_0 = min\{d_1, \ldots d_m\}$, and $D = LCM(d_1, \ldots d_m)$. Each test is performed in $\mathcal{O}(1)$. Moreover, D is bounded by ${d_0}^m$, where $d_0 = max\{d_1, \ldots d_m\}$, and hence $D = \mathcal{O}(2^{2^n})$. Hence, the procedure works in doubly exponential time.

The general case of finite trees is similar. Let X be a finite almost k-ary 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_0 to the final state q_{fin} of length $a + y \cdot l$. The rest of the proof follows the same reasoning path of the proof for sequence automata. □

From Lemma 3.8, it follows that, given a $\mathcal{B}(\mathcal{C}_k)$-automaton A, we have an algorithm to solve the problem $\mathcal{L}(A) \cap ITS_k(\Sigma) \neq \emptyset$ in doubly exponential time in the size of A.

Theorem 3.9
The emptiness problem for finite tree sequence automata over UULSs is in 2EXPTIME.

Since EQLTL(EQCTL$^*_\mathrm{k}$) formulas can be elementarily converted into $\mathcal{B}(\mathcal{C}_k)$ automata, we have the desired result.

Theorem 3.10
(Complexity of EQLTL(EQCTL$^*_\mathrm{k}$))
The satisfiability problem for EQLTL(EQCTL$^*_\mathrm{k}$) over UULSs is in ELEMENTARY.

We conclude the section by giving some examples of meaningful timing properties that can be expressed in (fragments of) EQLTL(EQCTL$^*_\mathrm{k}$) interpreted over UULSs. As a first example, consider the property 'P holds at every point of the finest layer T^0 whose distance from the origin of the layer 0_0 is a power of two ($1_0, 2_0, 4_0, 8_0$, and so on)' over a binary UULS. Such a property can be expressed in PLTL(CTL$^*_\mathrm{k}$) as follows:

$$\bigcirc \Box \, \mathbf{EX_1 G}((\mathbf{X}\mathtt{true} \rightarrow \mathbf{X_0}\mathtt{true}) \wedge (\neg \mathbf{X}\mathtt{true} \rightarrow P)).$$

Notice that the property 'P holds on every point 2^i, with $i \in \mathbb{N}$' cannot be expressed in QLTL. As a second example, the property 'P holds on every even point of the leftmost path' can be expressed in EQLTL(CTL$^*_\mathrm{k}$) as follows:

$$\exists Q(Q \wedge \bigcirc \neg Q \wedge \Box(Q \leftrightarrow \bigcirc \bigcirc Q) \wedge \Box(Q \rightarrow P)).$$

As already pointed out, this property cannot be expressed in PLTL(CTL$^*_\mathrm{k}$), since PLTL cannot express the property 'P holds on every even point' (Wolper 1983).

As in the case of DULSs, there are some natural properties of UULSs that cannot be easily captured in EQLTL(EQCTL$^*_\mathrm{k}$). As an example, it is not easy to express the property 'P holds on every even point of the finest domain T^0'.

4 The specification of a high voltage station

In this section, we exemplify the concrete use of temporalized logics as specification formalisms by providing (an excerpt of) the specification of a supervisor that automates the activities of a High Voltage (HV) station devoted to the end user distribution of energy generated by power plants (Montanari 1996). We first show how relevant timing properties of such a system can be expressed in monadic second-order languages, and then we give their simpler temporalized logic formulations.

Each HV station is composed of bays, connecting generation units to the distribution line. A bay consists of circuit breakers and insulators. They are both switches, but an expensive circuit breaker can interrupt current in a very short time (50 millisecond or even less), while a cheap insulator is not able to interrupt a flowing current and it has a switching time of a few seconds. Let us consider a simple HV station consisting of two bars `b1` and `b2` connected to different power units, a distribution line `l`, and two bays `pb` (parallel bay) and `lb` (line bay). The parallel bay shorts circuit between the two bars `b1` and `b2`. It consists of two insulators `ip1` and `ip2`, and one circuit breaker `cbp`. It is in the state `closed` if all its switches are closed; otherwise it is `open`. The line bay connects the distribution line with either the first bar or the second one. It consists of three insulators `ilb1`, `ilb2`, and `il1`, and one circuit breaker `cbl`. It is in the state `closed_on_b1` if `ilb1`, `cbl`, and `il1` are closed, and in the state `closed_on_b2` if `ilb2`, `cbl`, and `il1` are closed.

We focus on the specification of the change of the bar connected to the line from `b1` to `b2`. The supervisor starts its operation by closing the parallel bay, an action that takes about 10 *seconds*; then, it first closes the insulator `ilb2`, an action that takes about 5 *seconds* and then it opens the insulator `ilb1`, and action that takes 5 *seconds* as well; finally, it opens the parallel bay, an action that takes other 10 *seconds*. To model the behavior of the system, we use the predicates `change_b1_b2`, `change_b2_b1`, `close_pb`, `open_pb`, `close_ilb1`, `open_ilb1`, `close_ilb2`, and so on to denote the corresponding commands sent by the supervisor to the various devices. Furthermore, for every system action we identify the time granularity with respect to which it can be considered as an instantaneous action. The change of the bar takes about 30 seconds, opening and closing the parallel bay 10 seconds, switching insulators 5 seconds, and switching circuit breakers 50 milliseconds. Accordingly, we assume a 4-layered structure whose 4 layers correspond to the 4 involved time granularities, namely, `30secs`, `10secs`, `5secs`, and `50millisecs` (in (Franceschet and Montanari 2003) we show how to tailor temporal logics for time granularity over downward unbounded layered structures to deal with n-layered structures).

In the monadic second-order language, the change of the bar is described by the following formula, which specifies the sequence of actions taken by the supervisor:

$$
\begin{aligned}
\forall x.\ (T^{30secs}(x) \wedge \mathit{change\_b1\_b2}(x) \rightarrow \quad & \exists y_1.\ \downarrow_0(x) = y_1 \wedge \mathit{close\_pb}(y_1) \wedge \\
& \exists y_2.\ +1_{10secs}(y_1, y_2) \wedge \\
& \exists y_3.\ \downarrow_0(y_2) = y_3 \wedge \mathit{close\_ilb2}(y_3) \wedge \\
& \exists y_4.\ +1_{5secs}(y_3, y_4) \wedge \mathit{open\_ilb1}(y_4) \wedge \\
& \exists y_5.\ +1_{5secs}(y_4, y_5) \wedge \\
& \exists y_6.\ \downarrow_0(y_6) = y_5 \wedge \mathit{open\_pb}(y_6)),
\end{aligned}
$$

where the definable predicate $+1_g(x, y)$ states that both x and y belong to the layer g and y is the successor x with respect to g. Such a condition can be expressed in temporalized logic in a much more compact and readable way:

$$
\begin{aligned}
\mathbf{G}(\mathit{change\_b1\_b2} \rightarrow \quad & \mathbf{EX_0}\mathit{close\_pb} \wedge \mathbf{EX_1X_0}\mathit{close\_ilb2} \wedge \\
& \mathbf{EX_1X_1}\mathit{open\_ilb1} \wedge \mathbf{EX_2}\mathit{open\_pb})
\end{aligned}
$$

As for the compound operation `close_pb`, let us assume that the supervisor starts in parallel the closure of the circuit breaker, which is completed in 50 milliseconds, and of the first insulator, that takes about 5 seconds; then, once the first insulator is closed, it closes the second one. Such an operation can be specified by the following classical formula:

$$\begin{aligned}\forall x.\ (T^{10secs}(x) \wedge close\_pb(x) \rightarrow\quad & \exists y_1.\ \downarrow_0(x) = y_1 \wedge close\_ip1(y_1) \wedge \\ & \exists y_2.\ \downarrow_0(y_1, y_2) \wedge close\_cbp(y_2) \wedge \\ & \exists y_3.\ +1_{5secs}(y_1, y_3) \wedge close\_ip2(y_3),\end{aligned}$$

while its temporalized version is structured as follows:

$$\begin{aligned}\mathbf{G}(\quad & (\mathbf{EX_0}close\_pb \rightarrow \mathbf{EX_0}(\mathbf{EX_0}(close\_ip1 \wedge \mathbf{X_0}close\_cpb) \wedge \mathbf{EX_1}close\_ip2)) \wedge \\ & (\mathbf{EX_1}close\_pb \rightarrow \mathbf{EX_1}(\mathbf{EX_0}(close\_ip1 \wedge \mathbf{X_0}close\_cpb) \wedge \mathbf{EX_1}close\_ip2)) \wedge \\ & (\mathbf{EX_2}close\_pb \rightarrow \mathbf{EX_2}(\mathbf{EX_0}(close\_ip1 \wedge \mathbf{X_0}close\_cpb) \wedge \mathbf{EX_1}close\_ip2))).\end{aligned}$$

5 Conclusions and future work

In this paper, we provided the monadic second-order theories of DULSs and UULSs with expressively complete and elementarily decidable temporal logic counterparts. To this end, we defined temporalized automata, which can be seen as the automaton-theoretic counterpart of temporalized logics, and showed 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 exploited temporalized automata to successfully solve the problem of finding the temporal logic counterparts of the given theories of time granularity.

As a matter of fact, some forms of automaton combination, which differ from temporalization in various respects, have been proposed in the literature to increase the expressive power of temporal logics. As an example, extensions of PLTL with connectives defined by means of finite automata over ω-strings are investigated in (Vardi and Wolper 1994). To gain the expressive power of the full monadic second-order theory of $(\omega, <)$, Vardi and Wolper's Extended Temporal Logic (ETL) replaces the until operator of PLTL by an infinite bunch of automata connectives, that is, ETL allows formulas to occur as arguments of an automaton connective (as many formulas as the symbols of the automaton alphabet are). Given the well-known correspondence between formulas and automata, the application of automata connectives to formulas can be viewed as a form of automata combination. An extension of CTL* that substitutes ETL operators for PLTL ones is given in (Dam 1994). However, the switch from PLTL to ETL does not involve any change in the domain of interpretation (ω-structures in the first case, binary trees in the latter). On the contrary, in the case of temporalized automata/logics, component automata/temporal logics refer to different temporal structures, and thus their combination is paired with a combination of the underlying temporal structures.

We are developing our research on temporalized logics and automata for time granularity in various directions. First of all, we are trying to improve the complexity bound for the satisfiability problem for EQLTL(EQCTL$^*_{\mathrm{k}}$) over UULSs. Second, we are investigating the relationships between temporalized and classical automata. On the one hand, the languages recognized by temporalized automata are structurally different from those recognized by classical automata, e.g., `Büchi(Büchi)` automata recognize infinite strings of infinite strings. On the other hand, this fact does not imply that language problems for temporalized automata cannot be reduced to the corresponding problems for classical automata. As an example, the emptiness problem for `Büchi(Büchi)` automata can actually be reduced to the emptiness problem for `Büchi` automata. We are exploring the possibility of defining similar reductions for more complex temporalized automata. Finally, we are exploring the possibility of extending our correspondence results to other forms of logic combination, such as independent combination and join (Gabbay et al. 2003).

Acknowledgements

We would like to thank the anonymous reviewers for their comments and criticisms, that helped us to improve the paper, as well as Johan van Benthem, Ahmed Bouajjani, Marcelo Finger, Valentin Goranko, Maarten de Rijke, and Wolfgang Thomas for their positive feedback on the work reported in the paper. Thanks also to Pietro Corvaia, Alberto Policriti, and Franca Rinaldi for the useful discussions about the proof of Lemma 3.8.

References

Agrawal, R. and Srikant, R. 1995. Mining sequential patterns. In *Proceedings of the International Conference on Data Engineering*. IEEE Computer Society Press, 3–14.

Bettini, C., Brodsky, A., Jajodia, S., and Wang, X. S. 1997. Logical design for temporal databases with multiple granularities. *ACM Transactions on Database Systems 22,* 2, 115–170.

Bettini, C., Jajodia, S., Lin, J., and Wang, X. S. 1998. Discovering frequent event patterns with multiple granularities in time sequences. *IEEE Transactions on Knowledge and Data Engineering 10,* 2, 222–237.

Bettini, C., Jajodia, S., and Wang, X. 1996a. A general framework and reasoning models for time granularity. In *Proceedings of the International Workshop on Temporal Representation and Reasoning*. IEEE Computer Society Press, 104–111.

Bettini, C., Jajodia, S., and Wang, X. S. 1996b. Testing complex temporal relationships involving multiple granularities and its application to data mining. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. Vol. 15. ACM Press, 68–78.

Blackburn, P. and Bos, J. 2003. *Representation and Inference for Natural Language*. Studies in Logic, Language, and Information. CSLI Press. Forthcoming.

Chandra, R., Segev, A., and Stonebraker, M. 1994. Implementing calendars and temporal rules in next generation databases. In *Proceedings of the International Conference on Data Engineering*. IEEE Computer Society Press, Houston, TX, 264–273.

Ciapessoni, E., Corsetti, E., Montanari, A., and San Pietro, P. 1993. Embedding time granularity in a logical specification language for synchronous real-time systems. *Science of Computer Programming 20*, 141–171.

Combi, C., Franceschet, M., and Peron, A. 2002. A logical approach to represent and reason about calendars. In *Proceedings of the International Symposium on Temporal Representation and Reasoning.* IEEE Computer Society Press, 134–140.

Combi, C. and Pozzi, G. 2001. A temporal data model managing intervals with different granularities and indeterminacy from natural language sentences. *The VLDB Journal 9*, 294–311.

Corsetti, E., Crivelli, E., Mandrioli, D., Montanari, A., Morzenti, A., Pietro, P. S., and Ratto, E. 1991. Dealing with different time scales in formal specifications. In *Proceedings of the International Workshop on Software Specification and Design*. IEEE Computer Society Press, 92–101.

Corsetti, E., Montanari, A., and Ratto, E. 1991. Dealing with different time granularities in formal specifications of real-time systems. *The Journal of Real-Time Systems 3*, 191–215.

Courcoubetis, C., Vardi, M., Wolper, P., and Yannakakis, M. 1991. Memory efficient algorithms for the verification of temporal properties. In *Proceedings of Computer-Aided Verification*. Lecture Notes in Computer Science, vol. 531. Springer, Berlin, Germany, 233–242.

Cukierman, D. and Delgrande, J. 1998. Expressing time intervals and repetition within a formalization of calendars. *Computational Intelligence 14,* 4, 563–597.

Dam, M. 1994. CTL* and ECTL* as fragments of the modal μ-calculus. *Theoretical Computer Science 126*, 77–96.

Dreyer, W., Dittrich, A. K., and Schmidt, D. 1994. Research perspectives for time series management systems. *SIGMOD Record 23,* 1, 10–15.

Dyreson, C. E. and Snodgrass, R. T. 1995. Temporal granularity. In *The TSQL2 Temporal Query Language*, R. T. Snodgrass, Ed. Kluwer Academic Press, 347–385.

Emerson, E. 1990. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Vol. B*, J. van Leeuwen, Ed. Elsevier Science Publishers B.V., 995–1072.

Euzenat, J. 1995. An algebraic approach for granularity in qualitative space and time representation. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 894–900.

Fiadeiro, J. L. and Maibaum, T. 1994. Sometimes "tomorrow" is "sometime": Action refinement in a temporal logic of objects. In *Proceedings of the International Conference on Temporal Logic*. Lectures Notes on Artificial Intelligence, vol. 827. Springer, 48–66.

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

Foster, D., Leban, B., and McDonald, D. 1986. A representation for collections of temporal intervals. In *Proceedings of the National Conference on Artificial Intelligence*. 367–371.

Franceschet, M. 2002. Dividing and conquering the layered land. Ph.D. thesis, Department of Mathematics and Computer Science, University of Udine.

Franceschet, M. and Montanari, A. 2001a. A combined approach to temporal logics for time granularity. In *Proceedings of the International Workshop on Methods for Modalities (M4M)*.

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

Franceschet, M. and Montanari, A. 2002. Time granularities in databases, data mining, and temporal reasoning, by Bettini, Jajodia, and Wang (book review). *The Computer Journal 45,* 6, 683–685.

FRANCESCHET, M. AND MONTANARI, A. 2003. Branching within time: an expressively complete and elementarily decidable temporal logic for time granularity. *Research on Language and Computation 1,* 3-4, 229–263.

FRANCESCHET, M., MONTANARI, A., AND DE RIJKE, M. 2000. Model checking for combined logics. In *Proceedings of the International Conference on Temporal Logic*. 65–73.

FUM, D., GUIDA, G., MONTANARI, A., AND TASSO, C. 1989. Using levels and viewpoints in text representation. In *Proceedings of the International Conference on Artificial Intelligence and Information-Control Systems of Robots*. 37–44.

GABBAY, D., KURUCZ, A., WOLTER, F., AND ZAKHARYASCHEV, M. 2003. *Many-dimensional modal logics: theory and applications*. Studies in Logic and the Foundations of Mathematics, vol. 148. Elsevier.

GABBAY, D. M. AND DE RIJKE, M., Eds. 2000. *Frontiers of Combining Systems 2*. Studies in Logic and Computation, vol. 7. Research Studies Press/Wiley.

HAFER, T. AND THOMAS, W. 1987. Computation tree logic CTL* and path quantifiers in the monadic theory of the binary tree. In *Proceedings of the International Colloquium on Automata, Languages and Programming*. Lecture Notes in Computer Science, vol. 267. Springer, Karlsruhe, Germany, 269–279.

HALPERN, J. Y. AND SHOHAM, Y. 1991. A propositional modal logic of time intervals. *Journal of the ACM 38,* 4, 935–962.

IMMERMAN, N. AND KOZEN, D. 1989. Definability with bounded number of bound variables. *Information and Computation 83,* 2, 121–139.

JAJODIA, S., LITWIN, W., AND WIEDERHOLD, G. 1993. Integrating temporal data in a heterogeneous environment. In *Temporal Databases: Theory, Design and Implementation*, A. Tansel et al., Ed. Database Systems and Applications Series, Benjamin/Cummings Pub. Co., Redwood City, CA, 563–579.

JAJODIA, S., SUBRAHMANIAN, V. S., AND WANG, X. S. 1995. Temporal modules: An approach toward federated temporal databases. *Information Sciences 82*, 103–128.

JARD, C. AND JERON, T. 1989. On-line model checking for finite linear temporal logic specifications. In *Proceedings of the Workshop on Automatic Verification Methods for Finite State Systems*. Lecture Notes in Computer Science, vol. 407. Springer, 189–196.

KAMP, H. AND SCHIEHLEN, M. 2001. Temporal location in natural language. In *How to say when it happens*, H. Kamp and U. Reyle, Eds. Max Niemeyer Verlag.

KUPFERMAN, O., VARDI, M. Y., AND WOLPER, P. 2000. An automata-theoretic approach to branching-time model checking. *Journal of the ACM 47,* 2, 312–360.

LADKIN, P. 1987. The completeness of a natural system for reasoning with time intervals. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 462–467.

LAMPORT, L. 1985. On interprocess communication. Tech. Rep. 8, SRC, Palo Alto, CA.

MANNILA, H., TOIVONEN, H., AND VERKAMO, A. I. 1995. Discovering Frequent Episodes in Sequences. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. AAAI Press, Montreal, Canada.

MONTANARI, A. 1996. *Metric and Layered Temporal Logic for Time Granularity*. ILLC Dissertation Series 1996-02, ILLC, University of Amsterdam, The Netherlands.

MONTANARI, A., MAIM, E., CIAPESSONI, E., AND RATTO, E. 1992. Dealing with time granularity in the event calculus. In *Proceedings of the International Conference on fifth Generation Computer Systems*. 702–712.

MONTANARI, A. AND PERNICI, B. 1993. Temporal reasoning. In *Temporal Databases*, A. Tansel et al., Ed. Database Systems and Applications Series. Benjamin/Cummings Pub. Co., Redwood City, CA, 534–562.

MONTANARI, A., PERON, A., AND POLICRITI, A. 1999. Decidable theories of ω-layered metric temporal structures. *Logic Journal of the IGPL 7,* 1, 79–102.

MONTANARI, A., PERON, A., AND POLICRITI, A. 2000. The taming (timing) of the states. *Logic Journal of the IGPL 8,* 5, 681–699.

MONTANARI, A., PERON, A., AND POLICRITI, A. 2002. Extending Kamp's theorem to model time granularity. *Journal of Logic and Computation 12*, 641–677.

MONTANARI, A. AND POLICRITI, A. 1996. Decidability results for metric and layered temporal logics. *Notre Dame Journal of Formal Logic 37*, 260–282.

MONTANARI, A., SCIAVICCO, G., AND VITACOLONNA, N. 2002. Decidability of interval temporal logics over split-frames via granularity. In *Proceedings of the European Conference on Logic in Artificial Intelligence*. Lectures Notes on Artificial Intelligence, vol. 2424. Springer, Berlin, 259–270.

MOSZKOWSKI, B. 1983. Reasoning about digital circuits. Ph.D. thesis, Department of Computer Science, University of Stanford.

MOTA, E. AND ROBERTSON, D. 1996. Representing interaction of agents at different time granularities. In *Proceedings of the International Workshop on Temporal Representation and Reasoning*. IEEE Computer Society Press, 72–79.

NIEZETTE, M. AND STEVENNE, J. 1993. An efficient symbolic representation of periodic time. In *Proceeding of the International Conference on Information and Knowledge Management*. Lecture Notes in Computer Science, vol. 752. Springer, 161–168.

NING, P., JAJODIA, S., AND WANG, X. S. 2002. An algebraic representation of calendars. *Annals of Mathematics and Artificial Intelligence 36*, 5–38.

POESIO, M. AND BRACHMAN, R. J. 1991. Metric constraints for maintaining appointments: Dates and repeated activities. In *Proceedings of the National Conference on Artificial Intelligence*. MIT Press, 253–259.

SEGEV, A. AND CHANDRA, R. 1993. A data model for time-series analysis. In Advanced Database Systems. *Lecture Notes in Computer Science 759*, 191–212.

SHAHAR, Y. 1996. Dynamic temporal interpretation contexts for temporal abstraction. In *Proceedings of the International Workshop on Temporal Representation and Reasoning*. IEEE Computer Society Press, 64–71.

THOMAS, W. 1990. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Vol. B*, J. van Leeuwen, Ed. Elsevier Science Publishers, 133–191.

VARDI, M. Y. AND WOLPER, P. 1986. An automata-theoretic approach to automatic program verification. In *Proceedings of the Symposium on Logic in Computer Science*. IEEE Computer Society Press, Washington, D.C., USA, 332–345.

VARDI, M. Y. AND WOLPER, P. 1994. Reasoning about infinite computations. *Information and Computation 115,* 1, 1–37.

VENEMA, Y. 1991. A modal logic for chopping intervals. *Journal of Logic and Computation 1,* 4, 453–476.

WIJSEN, J. 1998. Reasoning about qualitative trends in databases. *Information Systems 23,* 7, 469–493.

WIJSEN, J. 1999. Temporal FDs on complex objects. *ACM Transactions on Database Systems 24,* 1, 127–176.

WOLPER, P. 1983. Temporal logic can be more expressive. *Information and Control 56,* 1/2, 72–99.

ZHOU, C. AND HANSEN, M. R. 1998. An adequate first order interval logic. In *Compositionality: the Significant Difference*, W. de Roever, H. Langmaak, and A. Pnueli, Eds. Lecture Notes in Computer Science, vol. 1536. 584–608.