Time Granularities and Ultimately Periodic Automata

Davide Bresolin, Angelo Montanari, and Gabriele Puppis

Dipartimento di Matematica e Informatica, Università di Udine
via delle Scienze 206, 33100 Udine, Italy
{bresolin,montana,puppis}@dimi.uniud.it

Abstract. The relevance of the problem of managing periodic phenomena is widely recognized in the area of knowledge representation and reasoning. One of the most effective attempts at dealing with this problem has been the addition of a notion of time granularity to knowledge representation systems. Different formalizations of such a notion have been proposed in the literature, following algebraic, logical, string-based, and automaton-based approaches. In this paper, we focus our attention on the automaton-based one, which allows one to represent a large class of granularities in a compact and suitable to algorithmic manipulation form. We further develop such an approach to make it possible to deal with (possibly infinite) sets of granularities instead of single ones. We define a new class of automata, called Ultimately Periodic Automata, we give a characterization of their expressiveness, and we show how they can be used to encode and to solve a number of fundamental problems, such as the membership problem, the equivalence problem, and the problem of granularity comparison. Moreover, we give an example of their application to a concrete problem taken from clinical medicine.

1 Introduction

The importance of managing periodic phenomena is widely recognized in a variety of applications in the areas of artificial intelligence and databases, including planning, natural language processing, temporal database inter-operability, data mining, and time management in workflow systems. One of the most effective attempts at dealing with this problem has been the addition of a notion of time granularity to knowledge and database systems. Different time granularities can be used to specify the occurrence times of different classes of events. For instance, the temporal characterizations of a flight departure, a business appointment, and a birthdate are usually given in terms of minutes, hours, and days, respectively. Furthermore, the ability of properly relating different time granularities is needed to process temporal information. As an example, when a computation involves pieces of information expressed at different time granularities, the system must integrate them in a principled way. Such an integration presupposes the formalization of the notion of granularity and the analysis of the relationships between different time granularities.

According to a commonly accepted perspective [1], any time granularity can be viewed as the partitioning of a given temporal domain in groups of elements, where each group is perceived as an indivisible unit (a granule). In particular, most granularities of interest are modeled as infinite sequences of granules that present a repeating pattern and, possibly, temporal gaps within and between granules. Even though conceptually clean, this point of view does not address the problem of finitely (and compactly) representing granularities to make it possible to deal with them in an effective (and efficient) way. In the literature, many different approaches to the management of time granularities have been proposed (we briefly survey them in Section 2). In this paper, we outline a general framework for time granularity that generalizes the automaton-based approach originally proposed by Dal Lago and Montanari in [6] by making it possible to deal with (possibly infinite) sets of granularities rather than single granularities. We give a characterization of ω-regular languages consisting of ultimately periodic words only, and we exploit such a characterization to define a proper subclass of Büchi automata, called Ultimately Periodic Automata (UPA), which includes all and only the Büchi automata that recognize ω-regular languages of ultimately periodic words. UPA allow one to encode single granularities, (possibly infinite) sets of granularities which have the same repeating pattern and different prefixes, and sets of granularities characterized by a finite set of non-equivalent patterns (the notion of equivalent patterns is given in Section 3), as well as any possible combination of them.

The rest of the paper is organized as follows. First, we briefly survey the most relevant formalisms for time granularity proposed in the literature. Next, we define UPA and we show how to use them to represent sets of periodical granularities. Noticeable properties of (the languages recognized by) UPA are then exploited to solve a number of basic problems about sets of time granularities. We focus our attention on the following problems: (i) *emptiness* (to decide whether a given set of granularities is empty); (ii) *membership* (to decide whether a granularity belongs to a given set of granularities); (iii) *equivalence* (to decide whether two representations define the same set of granularities); (iv) *minimization* (to compute compact representations of a given set of granularities); (v) *comparison of granularities* (for any pair of sets of granularities $\mathcal{G}, \mathcal{H}$, to decide whether there exist $G \in \mathcal{G}$ and $H \in \mathcal{H}$ such that $G \sim H$, where $\sim$ is one of the usual relations between granularities, e.g, partition, grouping, refinement, and aligned refinement [1]). Successively, we briefly analyze variants and extensions of UPA. Finally, we show how to apply the proposed framework to a real-world application taken from clinical medicine. We conclude the paper with a short discussion about achieved results and future research directions.

2 Formal Systems for Time Granularity

Different formal systems for time granularity have been proposed in the literature, following algebraic, logical, string-based, and automaton-based approaches [10]. The set-theoretic/algebraic approach, that subsumes well-known formalisms

developed in the areas of artificial intelligence and temporal databases, such as the temporal interval collection formalism [12] and the slice formalism [15], is described in detail in [1]. It assumes the temporal domain to be isomorphic to $\mathbb{N}$, and it defines every *time granularity* as a partition $G \subseteq 2^T$ of a set $T \subseteq \mathbb{N}$ such that for every pair of distinct sets $g, g' \in G$ (hereafter called *granules*), we have that either $\forall t \in g, t' \in g'(t < t')$ or $\forall t \in g, t' \in g'(t' < t)$. This definition captures both time granularities that cover the whole temporal domain, such as `Day`, `Week`, and `Month`, and time granularities with gaps within and between granules, like, for instance, `BusinessDay`, `BusinessWeek`, and `BusinessMonth`. Figure 1 depicts some of these granularities. For the sake of simplicity, we assume that both the first week and the first month start from the first day (such an assumption can be easily relaxed).

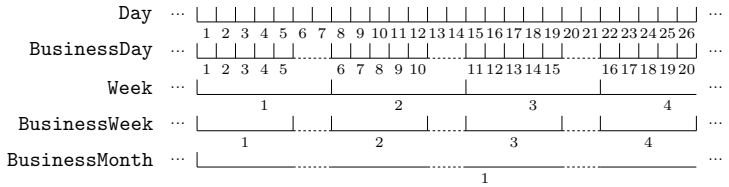

Fig. 1. Some examples of time granularities.

Various relations can be defined between pairs of granularities. Let us consider, for instance, the relations of grouping, refinement, partition, and aligned refinement (a large set of granularity relations is given in [1]). We have that a granularity G *groups into* (resp. *is refined by*) a granularity H if every granule of H is the union of some granules (resp. is contained in some granule) of G; moreover, a granularity G *partitions* a granularity H if G groups into H and G refines H; finally, a granularity G is an *aligned refinement* of H if, for every positive integer n, the n-th granule of G is included in the n-th granule of H. In the case of Figure 1, we have that `Day` groups into `BusinessMonth`, `BusinessDay` refines `Week`, `Day` partitions `Week`, and `BusinessWeek` is an aligned refinement of `Week`.

A symbolic representation of a significant class of time granularities has been obtained by means of the formalism of *Calendar Algebra* (CA) [16]. Such a formalism represents time granularities as expressions built up from a finite set of basic granularities through the application of suitable algebraic operators. For instance, the granularity `Week` can be generated by applying the operator $Group_7$ to the granularity `Day`. The operations of CA reflect the ways in which people define new granularities from existing ones, and thus CA turns out to be a fairly natural formalism. In despite of that, it suffers from some non-trivial drawbacks: it does not address in a satisfactory way some basic problems of obvious theoretical and practical importance, such as the equivalence problem, and it only partially works out, in a rather complex way, other relevant problems, such as the problem of granularity conversion [11].

A string-based model for time granularities has been proposed by Wijsen [18]. Infinite granularities are modeled as infinite words over an alphabet consisting of three symbols, namely, ■ (filler), □ (gap), and ≀ (separator), which are respectively used to denote time points covered by some granule, to denote time points not covered by any granule, and to delimit granules. Wijsen focuses his attention on the class of periodical granularities, that is, to granularities that, ultimately, periodically groups time points of the underlying temporal domain. Periodical granularities can be identified with ultimately periodic words, and they can be finitely represented by specifying a (possibly empty) prefix and a repeating pattern. As an example, the granularity `BusinessWeek` ■■■■■□□≀■■■■■□□≀... can be encoded by the empty prefix ε and the repeating pattern ■■■■■□□≀. In order to solve the equivalence problem, Wijsen defines a suitable *aligned form*, which forces separators to occur immediately after an occurrence of ■ (in such a case, one can encode each occurrence of the substring ■≀ by means of a single symbol ◀). The aligned form guarantees a one-to-one correspondence between strings and granularities, thus providing a straightforward solution to the equivalence problem.

The idea of viewing time granularities as ultimately periodic strings establishes a natural connection with the field of formal languages and automata. The basic idea underlying the automaton-based approach to time granularity is simple: we take an automaton $\mathcal{A}$ recognizing a *single* ultimately periodic word $u \in \{\square, \blacksquare, \blacktriangleleft\}^\omega$ and we say that $\mathcal{A}$ represents the granularity G if and only if u represents G. Such an automaton-based approach to time granularity has been proposed by Dal Lago and Montanari in [6], and later revisited by Dal Lago, Montanari, and Puppis in [7,8]. The resulting framework views granularities as strings generated by a specific class of automata, called Single-String Automata (SSA). In order to compactly encode the redundancies of the temporal structures, SSA are endowed with counters ranging over discrete finite domains (Extended SSA, ESSA for short). Properties of ESSA have been exploited to efficiently solve the equivalence and the granule conversion problems for single time granularities. Moreover, the relationships between ESSA and Calendar Algebra have been investigated in [7], where a number of algorithms that map Calendar Algebra expressions into automaton-based representations of time granularities are given. Such an encoding allows one to reduce problems about Calendar Algebra expressions to equivalent problems for ESSA. This suggests an alternative point of view on the automaton-based framework: besides a formalism for the direct specification of granularities, automata can be viewed as a low-level operational formalism into which high-level granularity specifications, such as those of Calendar Algebra, can be mapped.

The choice of Propositional Linear Temporal Logic (LTL for short) as a logical tool for granularity management has been advocated by Combi et al. [5]. Granularities are defined as models of LTL formulas, where suitable propositional symbols are used to mark the endpoints of granules. In this way, a large set of ω-regular granularities, such as, for instance, repeating patterns that can start at an arbitrary time point (unanchored granularities), can be captured.

Moreover, problems like checking the consistency of a granularity specification or the equivalence of two granularity expressions can be solved in a uniform way by reducing them to the validity problem for LTL, which is known to be in PSPACE. An extension of LTL that replaces propositional variables by first-order formulas defining integer constraints, e.g., $x \equiv_k y$, has been proposed by Demri [9]. The resulting logic, denoted by PLTL^{mod}(Past LTL with integer periodicity constraints), generalizes both the logical framework proposed by Combi et al. and Dal Lago and Montanari's automaton-based one, and it allows one to compactly define granularities as periodicity constraints. In particular, the author shows how to reduce the equivalence problem for ESSA to the model checking problem for PLTL^{mod}(-automata), which turns out to be in PSPACE, as for LTL.

3 A new class of automata

In this paper, we extend the automaton-based approach to deal with (possibly infinite) sets of periodical granularities. In [6], Dal Lago and Montanari show how single granularities can be modeled by SSA, that is, Büchi automata recognizing single ultimately periodic words. In the following, we identify a larger (proper) subclass of Büchi automata that recognizes languages only consisting of ultimately periodic words, and we show how to exploit them to efficiently deal with time granularities (proof details are given in [2]).

3.1 Ultimately Periodic Automata

We first show that ω-regular languages only consisting of ultimately periodic words (periodical granularities) can be represented via suitable Büchi automata, that we call Ultimately Periodic Automata. By definition, *ω-regular* languages are sets of infinite words recognized by Büchi automata. They can be expressed as finite unions of sets of the form $U \cdot V^\omega$, where U and V are regular languages of finite words. From this, it easily follows that any ω-regular language is non-empty iff it contains an ultimately periodic word. (An ultimately periodic word w is an infinite word of the form $u \cdot v^\omega$, where u and v ($\neq \varepsilon$) are finite words called the *prefix* and the *repeating pattern* of w, respectively.) From the closure properties of ω-regular languages, it follows that any ω-regular language L is uniquely identified by the set $UP(L)$ of its ultimately periodic words, that is, for every pair of ω-regular languages L, L', $L = L'$ iff $UP(L) = UP(L')$.

The following theorem provides a characterization of ω-regular languages of ultimately periodic words.

Theorem 1. *An ω-regular language L only consists of ultimately periodic words iff it is a finite union of sets $U \cdot \{v\}^\omega$, where $U \subseteq \Sigma^*$ is regular and v is a finite non-empty word.*

From Theorem 1, it follows that ω-regular languages of ultimately periodic words capture sets of granularities with possibly infinitely many prefixes, but with only a finite number of non-equivalent repeating patterns (we say that two

patterns v and v' are equivalent iff they can be obtained by rotating and/or repeating a finite word v'').

Theorem 1 yields a straightforward definition of the class of automata that captures all and only the ω-regular languages of ultimately periodic words.

Definition 1. *An* Ultimately Periodic Automaton *(UPA) is a Büchi automaton* $\mathcal{A} = (Q, q_0, \Delta, F)$ *such that, for every* $f \in F$*, the strongly connected component of* f *is either a single transient state or a single loop with no exiting transitions.*

Theorem 2. *UPA recognize all and only the* ω*-regular languages of ultimately periodic words.*

Theorem 2 can be reformulated by stating that UPA-recognizable languages are all and only the ω-regular languages L such that $L = UP(L)$.

Figure 2 depicts two UPA recognizing the languages $\{\square\}^* \cdot \{\blacksquare\blacktriangleleft\}^\omega$ and $\{\blacksquare\blacksquare\blacktriangleleft\}^\omega \cup \{\blacksquare\blacktriangleleft\}^\omega$, respectively. The former represents the unanchored granularity that groups days two by two, while the latter represents two granularities that respectively group days two by two and three by three.

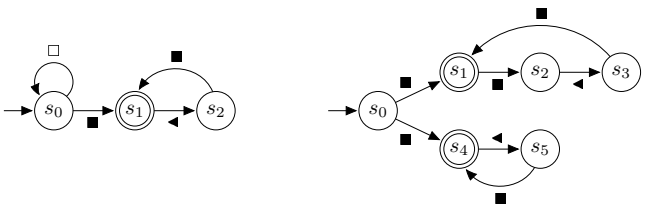

Fig. 2. Two examples of UPA.

By exploiting the same construction methods used in the case of Büchi automata, one proves that UPA are closed under union, intersection, and concatenation with regular languages. Furthermore, it is trivial to see that UPA satisfy a weak form of closure under ω-exponentiation, namely, for every finite non-empty word v, there is an UPA recognizing the language $\{v\}^\omega$. On the contrary, from Theorem 1 it easily follows that UPA are not closed under complementation. Consider, for instance, the empty language $\emptyset$. Its complement is the set of all ultimately periodic words, which is not an UPA-recognizable language (UPA-recognizable languages must encompass finitely many non-equivalent repeating patterns).

Theorem 3. *UPA are closed under intersection, union, and concatenation with regular languages, but they are not closed under complementation.*

UPA can be successfully exploited to efficiently solve a number of problems involving sets of granularities.

– **Emptiness**. The emptiness problem is solved in polynomial time by testing the existence of a loop involving some final state (henceforth *final loop*) reachable from the initial state.

– **Membership**. The membership problem consists in deciding whether an UPA $\mathcal{A}$ recognizes a given ultimately periodic word w. Given an UPA $\mathcal{B}$ recognizing the singleton $\{w\}$, one can decide in polynomial time whether $w \in \mathcal{L}(\mathcal{A})$ by testing the emptiness of the language recognized by the product automaton $\mathcal{A} \times \mathcal{B}$ over the alphabet $\{(\begin{smallmatrix}\square\\\square\end{smallmatrix}), (\begin{smallmatrix}\blacksquare\\\blacksquare\end{smallmatrix}), (\begin{smallmatrix}\blacktriangleleft\\\blacktriangleleft\end{smallmatrix})\}$.
– **Equivalence**. One can decide whether two given UPA $\mathcal{A}$ and $\mathcal{B}$ are equivalent by viewing them as generic Büchi automata, by computing their complements $\overline{\mathcal{A}}$ and $\overline{\mathcal{B}}$ (which are not necessarily UPA), and by testing the emptiness of both $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\overline{\mathcal{B}})$ and $\mathcal{L}(\mathcal{B}) \cap \mathcal{L}(\overline{\mathcal{A}})$. Here we provide an alternative, direct and more efficient method for deciding the equivalence problem. The solution exploits a suitable canonical form of UPA, which turns out to be unique up to isomorphisms. Such a form is obtained by a canonization algorithm that works as follows [2]:
  1. minimize the patterns of the recognized words and the final loops (using Paige-Tarjan-Bonic algorithm [17]);
  2. minimize the prefixes of the recognized words;
  3. compute the minimum *deterministic* automaton for the prefixes of the recognized words (using Brzozowski algorithm [3]);
  4. build the canonical form by adding the final loops to the minimum automaton for the prefixes.

  Brzozowski's algorithm (used at step 3) requires exponential time and space in the worst case, but it turned out to be faster than the other available algorithms in many experiments [4]. As a result, the proposed algorithm for testing the equivalence of UPA outperforms the alternative algorithm using Büchi automata.
– **Minimization**. The minimization problem consists in computing the most compact representations for a given set of granularities. Such a problem is somehow connected to the equivalence problem, since in many cases minimal automata turn out to be unique up to isomorphisms. In the case of UPA, the minimization problem is PSPACE-complete and it may yield different solutions. A minimal UPA can be obtained by simply replacing step 3 in the canonization algorithm with the computation of a minimal *non-deterministic* automaton for the prefixes of the recognized words (using the construction developed in [14]).
– **Comparison of granularities**. Usual relations between granularities can be checked by looking at their automaton-based representations. In particular, granularity comparison problems can be easily reduced to the emptiness problem for suitable product automata. As an example, let us consider the partition relation. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two UPA representing two sets of granularities $\mathcal{H}$ and $\mathcal{G}$, respectively. In order to check whether there exist two granularities $G \in \mathcal{G}$ and $H \in \mathcal{H}$ such that G *partitions* H, we first compute a product automaton $\mathcal{A}_3$ accepting all pairs of granularities that satisfy the 'partition' property, and then we test the emptiness of the recognized language. The automaton $\mathcal{A}_3$ is defined as follows:
  1. the set of states of $\mathcal{A}_3$ is $S_1 \times S_2 \times \{0, 1, 2\}$, where S_1 (resp. S_2) is the set of states of $\mathcal{A}_1$ (resp. $\mathcal{A}_2$);

2. the initial state of $\mathcal{A}_3$ is the tuple $(s_1, s_2, 0)$, where s_1 (resp. s_2) is the initial state of $\mathcal{A}_1$ (resp. $\mathcal{A}_2$);
3. the transition relation of $\mathcal{A}_3$ copies the transition relations of $\mathcal{A}_1$ and $\mathcal{A}_2$ in the first two components of states; it changes the third component from 0 to 1 when a final state of $\mathcal{A}_1$ occurs, from 1 to 2 when a final state of $\mathcal{A}_2$ occurs, and back to 0 immediately afterwards; finally, it constrains the recognized symbols to belong to the set $\{(\square\atop\square), (\blacksquare\atop\blacksquare), (\blacktriangleleft\atop\blacktriangleleft), (\blacksquare\atop\blacktriangleleft)\}$;
4. the final states of $\mathcal{A}_3$ are all and only the tuples of the form $(s_1, s_2, 2)$.

3.2 Relaxed UPA

In the following, we introduce a new class of automata which are as expressive as UPA, but produce more compact representations of granularities. UPA structure is well-suited for algorithmic manipulation. In particular, it allows one to solve the equivalence problem by taking advantage of a suitable canonical form. However, UPA may present redundancies in their structure, due to the presence of duplicated loops encoding the same pattern. As an example, consider the automata of Figure 3. They both recognize the language expressed by the ω-regular expression $\square\blacktriangleleft^{\omega} \cup \square\blacktriangleleft(\blacktriangleleft\blacktriangleleft)^*\square^{\omega}$, but the automaton to the right has less states than the one to the left. Unfortunately, it is not an UPA, because UPA do not allow transitions to exit from final loops.

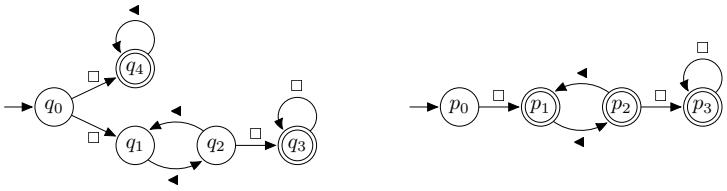

Fig. 3. UPA may present redundancies.

We define a new class of automata, that includes both automata of Figure 3, which only requires that, whenever an automaton leaves a final loop, it cannot reach it again.

Definition 2. *A* Relaxed UPA *(RUPA) is a Büchi automaton $\mathcal{A} = (Q, \Delta, q_0, F)$ such that, for every $f \in F$, the strongly connected component of f is either a single transient state or a single loop.*

The relation between UPA and RUPA is stated by the following theorem.

Theorem 4. *RUPA recognize all and only the UPA-recognizable languages.*

RUPA can be exploited to obtain more compact representations of granularities. To this end, one must take into consideration every strongly connected component S of an UPA and check whether it satisfies the following conditions:

1. S is a single loop that does not involve final states;

2. S encodes a single pattern v (up to rotations);
3. there exists a final loop C_f that encodes a pattern that is equivalent to v;
4. S and C_f have the same entering transitions.

If conditions 1-4 are satisfied, then S and C_f can be merged, preserving the recognized language. As an example, consider the UPA automaton to the left of Figure 3. The strongly connected component $\{q_1, q_2\}$ meets the above conditions: it is a single loop that does not involve final states, it encodes the single pattern ◀◀, which is equivalent to the pattern encoded by the final loop $\{q_4\}$, and $\{q_1, q_2\}$ and $\{q_4\}$ have the same entering transitions. Hence, q_4 can be removed from the automaton, provided that q_1 and q_2 become final states. The resulting RUPA automaton is exactly the one to the right of Figure 3.

Such a construction can be turned into an algorithm that transforms UPA into more compact RUPA by eliminating redundant final loops (if any). It first determines all the strongly connected components of UPA that meet conditions 1-4, and then it merges them. The algorithm turns out to be of polynomial time complexity. It must be noted that the algorithm transforms UPA into more compact, but not necessarily minimal, RUPA. The resulting RUPA are indeed not guaranteed to be of minimal size, even when the input UPA are minimal (and the minimization algorithm for UPA cannot be applied to RUPA).

3.3 Beyond (R)UPA

We conclude the section by briefly investigating possible extensions of (R)UPA. We have shown that the class of (R)UPA is the subclass of Büchi automata that recognize ω-regular languages of ultimately periodic words. As we shall illustrate in the next section, (R)UPA allow one to deal with meaningful real-world applications. Nevertheless, there are many sets of of periodical granularities which are not captured by (R)UPA.

In [2], we define a new class of automata, called *Three-Phase Automata* (3PA), which includes (R)UPA, that captures all and only the languages L for which there exists an ω-regular language L' such that $L = UP(L')$. This set of languages includes both ω-regular languages (the (R)UPA-recognizable languages) and non-ω-regular languages (the languages L such that $L = UP(L') \subset L'$). In particular, unlike (R)UPA, 3PA are able to capture sets of granularities featuring an infinite number of non-equivalent repeating patterns. Computations of 3PA consist in three steps: (i) the automaton guesses the prefix of an ultimately periodic word, then (ii) it guesses its repeating pattern and stores it in a queue, and finally (iii) it recognizes the stored pattern infinitely many times. 3PA are closed under union, intersection, concatenation with a regular language, and complementation. Moreover, it is not difficult to show that the solutions to the basic problems about sets of granularities given for (R)UPA can be generalized to 3PA.

There exist, however, noticeable sets of granularities featuring an infinite number of non-equivalent repeating patterns which are not 3PA-recognizable. This is the case, for instance, of the language $\{(\blacksquare^n \blacktriangleleft)^\omega | n \geq 0\}$ of all and only

the granularities that group days n by n, with $n > 0$. All 3PA that recognize these repeating patterns must indeed also recognize all, but finitely many, combinations of them. Such a distinctive property of all 3PA-recognizable languages is captured by the following theorem [2].

Theorem 5. *Let $L = UP(L')$ where L' is defined by the ω-regular expression $\bigcup_i U_i \cdot V_i^\omega$. For any i, if V_i includes (at least) two non-equivalent patterns v and v', then L includes all ultimately periodic words $u(v_o v_1 \dots v_n)^\omega$, for all $n \geq 0$, $u \in U_i$, and $v_i \in \{v, v'\}$.*

4 A Real-World Application

The need of dealing with sets of time granularities arises in several application domains. We focus our attention on the medical domain of heart transplant patients. Posttransplantation guidelines require outpatients to take drugs and to submit to periodical visits for life. These requirements are usually collected in formal protocols with schedules specifying the therapies and the frequency of the check-ups. We report an excerpt of the guidelines for an heart transplant patient reported in [13]. Depending on the physical conditions of the patient, the guidelines can require, together with other treatments, an estimation of the glomerular filtration rate (GFR) with one of the following schedules:

- 3 months and 12 months posttransplantation and every year thereafter;
- 3 months and 12 months posttransplantation and every 2 years thereafter.

These protocols involve the so-called *unanchored granularities*, to manage the various admissible starting points for the scheduled therapies (and/or check-ups), as well as *sets of granularities with different repeating patterns*, to capture the set of distinct periodicities of the scheduled therapies. The ability of dealing with sets of granularities, and not only with single granularities, is thus needed to reason about protocols and patient schedules. As an example, since different protocols can be specified for the same class of patients by different people/institutions, it is a critical problem to decide whether two protocols define the same set of therapies/granularities (equivalence problem). The decidability of this problem gives the possibility of choosing the most compact, or most suitable, representation for a given protocol. Another meaningful reasoning task is that of checking whether a given therapy/granularity assigned to a patient satisfies the prescribed protocol, that is, whether it belongs to the set of therapies/granularities of the protocol (granularity comparison problem).

Let us consider this latter problem. Consider the above given (sub)set of therapies/check-ups of a protocol for hearth transplant patients. Given an UPA $\mathcal{A}$ encoding (the granularities of) such a set of therapies/check-ups and an UPA $\mathcal{B}$ representing the single granularity of the specific therapy (up to a certain date), the granularity comparison problem can be decided by checking the existence of a word in $\mathcal{L}(\mathcal{A})$ that properly relates to the one contained in $\mathcal{L}(\mathcal{B})$. For the sake of simplicity, we consider months of 30 days and years of 365 days (relaxing

such a simplification is tedious, but trivial). The UPA $\mathcal{A}$ is depicted in Figure 4, where we use the shorthand $\circ \xrightarrow{a^n} \circ$ to denote a sequence of $n+1$ states and n a-labeled transitions.

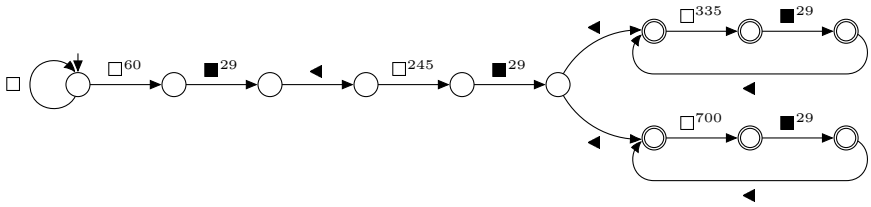

Fig. 4. The UPA-based specification of the protocol.

We model the granularity of the therapy assigned to the patient with a single ultimately periodic word v (equivalently, an SSA $\mathcal{B}$), where the occurrences of ◀ denote the days of the visits. We can check the consistency of the therapy with respect to the prescribed protocol by testing whether the granularity v is an *aligned refinement* of some granularity $u \in \mathcal{L}(\mathcal{A})$. Thus, the given consistency-checking problem can be seen as a particular case of granularity comparison problem. Given two words u and v that represent, respectively, granularities G and H, we have that H is an aligned refinement of G iff, for every $n \in \mathbb{N}^+$, $v[n] \in \{\blacksquare,\blacktriangleleft\}$ implies that $u[n] \in \{\blacksquare,\blacktriangleleft\}$ and that the words $v[1, n-1]$ and $u[1, n-1]$ encompass the same number of occurrences of ◀. Such a condition can be easily verified in polynomial time as follows. Given two UPA $\mathcal{A}$ and $\mathcal{B}$ representing two sets of granularities $\mathcal{G}$ and $\mathcal{H}$, (i) one constructs a product automaton $\mathcal{C}$ that accepts all pairs of granularities $G \in \mathcal{G}$ and $H \in \mathcal{H}$ such that H is an aligned-refinement of G, and then (ii) he/she tests the emptiness of the language recognized by $\mathcal{C}$.

As an example, consider the following instance of the temporal relation VISITS(PatientId, Date, Treatment).

PatientId	Date (MM/DD/YYYY)	Treatment
1001	02/10/2003	transplant
1001	04/26/2003	GFR
1002	06/07/2003	GFR
1001	06/08/2003	biopsy
1001	02/10/2004	GFR
1001	01/11/2005	GFR
1001	01/29/2006	GFR

By properly selecting records, we can build the granularity of GFR measurements for the patient identified by 1001. We represent this granularity as a single ultimately periodic word v (starting from 01/01/2003), in which the occurrences of ◀ denote the days of the visits. The UPA $\mathcal{B}$ recognizing v is depicted in Figure 5.

In order to check whether the granularity of GFR measurements for patient 1001 is an aligned refinement of some granularity in $\mathcal{L}(\mathcal{A})$, we must construct the

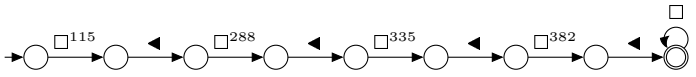

Fig. 5. The UPA representing GFR measurements for patient 1001.

product automaton for the relation of aligned refinement. Such an automaton recognizes the language

$$\left\{ \left(\begin{smallmatrix}\square\\\square\end{smallmatrix}\right)^{100}\left(\begin{smallmatrix}\blacksquare\\\square\end{smallmatrix}\right)^{15}\left(\begin{smallmatrix}\blacksquare\\\blacktriangleleft\end{smallmatrix}\right)\left(\begin{smallmatrix}\blacksquare\\\square\end{smallmatrix}\right)^{13}\left(\begin{smallmatrix}\blacktriangleleft\\\square\end{smallmatrix}\right)\left(\begin{smallmatrix}\square\\\square\end{smallmatrix}\right)^{245}\left(\begin{smallmatrix}\blacksquare\\\square\end{smallmatrix}\right)^{29}\left(\begin{smallmatrix}\blacktriangleleft\\\blacktriangleleft\end{smallmatrix}\right)\left(\begin{smallmatrix}\square\\\square\end{smallmatrix}\right)^{335}\left(\begin{smallmatrix}\blacksquare\\\blacktriangleleft\end{smallmatrix}\right)\left(\begin{smallmatrix}\blacksquare\\\square\end{smallmatrix}\right)^{28}\left(\begin{smallmatrix}\blacktriangleleft\\\square\end{smallmatrix}\right)\left(\begin{smallmatrix}\square\\\square\end{smallmatrix}\right)^{335}\cdot \cdot\left(\begin{smallmatrix}\blacksquare\\\square\end{smallmatrix}\right)^{18}\left(\begin{smallmatrix}\blacksquare\\\blacktriangleleft\end{smallmatrix}\right)\left(\begin{smallmatrix}\blacksquare\\\square\end{smallmatrix}\right)^{10}\left(\begin{smallmatrix}\blacktriangleleft\\\square\end{smallmatrix}\right)\left(\left(\begin{smallmatrix}\square\\\square\end{smallmatrix}\right)^{335}\left(\begin{smallmatrix}\blacksquare\\\square\end{smallmatrix}\right)^{29}\left(\begin{smallmatrix}\blacktriangleleft\\\square\end{smallmatrix}\right)\right)^{\omega} \right\}$$

over the alphabet $\{\left(\begin{smallmatrix}\square\\\square\end{smallmatrix}\right), \left(\begin{smallmatrix}\blacksquare\\\square\end{smallmatrix}\right), \left(\begin{smallmatrix}\blacktriangleleft\\\square\end{smallmatrix}\right), \left(\begin{smallmatrix}\blacksquare\\\blacksquare\end{smallmatrix}\right), \left(\begin{smallmatrix}\blacksquare\\\blacktriangleleft\end{smallmatrix}\right), \left(\begin{smallmatrix}\blacktriangleleft\\\blacksquare\end{smallmatrix}\right), \left(\begin{smallmatrix}\blacktriangleleft\\\blacktriangleleft\end{smallmatrix}\right)\}$. Since the resulting language is not empty, we can conclude that the therapy satisfies the prescribed protocol.

5 Discussion

In this paper, we developed an original automaton-based approach to the management of sets of granularities. We defined a new class of automata, called UPA, that allow one to represent sets of granularities having possibly infinitely many different prefixes and a finite number of non-equivalent repeating patterns. We showed how well-known results coming from automata theory can be exploited to solve a number of meaningful problems about sets of granularities. In particular, we provided effective solutions to the problems of emptiness, membership, equivalence, minimization, and comparison of granularities for UPA. Furthermore, we discussed variants and extensions of UPA (RUPA and 3PA) that increase compactness and expressiveness of granularity representations. Finally, we applied the proposed framework to a case study taken from the domain of clinical medicine. More specifically, we showed that UPA can be used to specify medical guidelines and to check whether concrete therapy plans conform to them.

As for open problems, on the one hand we are looking for larger classes of languages of ultimately periodic words that extends the class of 3PA, possibly preserving closure and (some) decidability properties. On the other hand, we are currently looking for the (proper) fragment of PLTL^{mod} defining the temporal logic counterpart of UPA as well as for the logical counterpart of 3PA. Pairing the logical formalism with the automaton-based one would allow us to use the former as a high-level interface for the specification of granularities and the latter as an internal formalism for efficiently reasoning about them.

References

1. C. Bettini, S. Jajodia, and X.S. Wang. *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Springer, July 2000.

2. D. Bresolin, A. Montanari, and G. Puppis. Time granularities and ultimately periodic automata. Technical Report 24, Dipartimento di Matematica e Informatica, Università di Udine, Italy, October 2003.
3. J.A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. *Mathematical Theory of Automata*, 12:529–561, 1962.
4. C. Campeanu, K. Culik II, K. Salomaa, and S. Yu. State complexity of basic operations on finite languages. In *4th International Workshop on Implementing Automata (WIA'99)*, volume 2214 of *LNCS*, pages 60–70. Springer, 2001.
5. C. Combi, M. Franceschet, and A. Peron. Representing and reasoning about temporal granularities. *Journal of Logic and Computation*, 14(1):51–77, 2004.
6. U. Dal Lago and A. Montanari. Calendars, time granularities, and automata. In *7th International Symposium on Spatial and Temporal Databases (SSTD)*, volume 2121 of *LNCS*, pages 279–298. Springer, 2001.
7. U. Dal Lago, A. Montanari, and G. Puppis. Time granularities, calendar algebra, and automata. Technical Report 4, Dipartimento di Matematica e Informatica, Università di Udine, Italy, February 2003.
8. U. Dal Lago, A. Montanari, and G. Puppis. Towards compact and tractable automaton-based representations of time granularity. In *8th Italian Conference on Theoretical Computer Science (ICTCS)*, volume 2841 of *LNCS*, pages 72–85. Springer, 2003.
9. S. Demri. LTL over integer periodicity constraints (extended abstract). In *Proceedings of the 7th Int. Conf. on Foundations of Software Science and Computation Structures (FOSSACS)*, volume 2987 of *Lecture Notes in Computer Science*, pages 121–135. Springer, April 2004.
10. J. Euzenat and A. Montanari. Time granularity. In M. Fisher, D. Gabbay, and L. Vila, editors, *Handbook of Temporal Reasoning in Artificial Intelligence*. Elsevier, 2004.
11. M. Franceschet and A. Montanari. Time granularities in databases, data mining, and temporal reasoning, by Claudio Bettini, Sushil Jajodia, and Sean X. Wang (book review). *The Computer Journal*, 45(6):683–685, 2002.
12. B. Leban, D. McDonald, and D. Foster. A representation for collections of temporal intervals. In *AAAI National Conference on Artificial Intelligence*, volume 1, pages 367–371. AAAI Press, 1986.
13. Loma Linda University Medical Center. Pediatric heart transplantation protocol, 2002.
14. O. Matz and A. Potthoff. Computing small nondeterministic automata. In *Proceedings of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, BRICS Notes Series, pages 74–88, 1995.
15. M. Niezette and J. Stevenne. An efficient symbolic representation of periodic time. In *International Conference on Information and Knowledge Management (CIKM)*, pages 161–168, Baltimore, MD, 1992. ACM Press.
16. P. Ning, S. Jajodia, and X.S. Wang. An algebraic representation of calendars. *Annals of Mathematics and Artificial Intelligence*, 36:5–38, 2002.
17. R. Paige, R.E. Tarjan, and R. Bonic. A linear time solution to the single function coarsest partition problem. *Theoretical Computer Science*, 40:67–84, 1985.
18. J. Wijsen. A string-based model for infinite granularities. In C. Bettini and A. Montanari, editors, *AAAI Workshop on Spatial and Temporal Granularities*, pages 9–16. AAAI Press, 2000.