On the Equivalence of Automaton-based Representations of Time Granularities

Ugo Dal Lago
Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy
dallago@cs.unibo.it

Angelo Montanari
Dipartimento di Matematica e Informatica, Università di Udine, Italy
montana@dimi.uniud.it

Gabriele Puppis
Dipartimento di Matematica e Informatica, Università di Udine, Italy
puppis@dimi.uniud.it

Abstract

A 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. In this paper we explore an automaton-based approach to the management of time granularity that compactly represents time granularities as single-string automata with counters, that is, Büchi automata, extended with counters, that accept a single infinite word. We focus our attention on the equivalence problem for the class of restricted labeled single-string automata (RLA for short). The equivalence problem for RLA is the problem of establishing whether two given RLA represent the same time granularity. The main contribution of the paper is the reduction of the (non-)equivalence problem for RLA to the satisfiability problem for linear diophantine equations with bounds on variables. Since the latter problem has been shown to be NP-complete, we have that the RLA equivalence problem is in co-NP.

1. Introduction

The notion of time granularity comes into play in a number of computer science scenarios, ranging from the specification and verification of timed workflow systems to the management of temporal constraints, from the design of temporal databases to temporal data mining applications.

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). Most granularities of practical interest are modeled as infinite sequences of time granules, which present a repeating pattern and, possibly, temporal gaps within and between granules. A number of different formalisms to finitely represent infinite time granularities have been proposed in the literature [11], based on algebraic [3, 15, 17, 18], logical [5, 10], string-based [20], and automaton-based [7, 4] approaches.

We restrict our attention to the automaton-based approach. First, we introduce single-string automata and we show that they are as expressive as Wijsen's string-based models. Next, we show how to extend single-string automata with counters to take advantage of regularities of modeled granularities. Besides making the structure of the automata more compact, this allow us to efficiently deal with those granularities which have a quasi-periodic structure. Single-string automata with counters are then used to provide an effective solution to the equivalence problem for granularity specifications. The decidability of such a problem implies the possibility of effectively testing the semantic equivalence of two different specifications, thus making it possible to use smaller, or more tractable, representations in place of bigger, or less tractable, ones.

The rest of the paper is organized as follows. In Section 2 we define the notion of time granularity and we briefly describe the string-based model of time granularities. In Section 3 we outline the distinctive features of the automaton-based approach, focusing our attention counters and multiple transitions, and we show how the fundamental problem of granularity equivalence can be formulated in terms of the proposed class of automata. In addition, we briefly analyze the relationships between the automaton-based approach and the logical one. In Section 4 we introduce Restricted Labeled single-string Automata and we provide a formal characterization of the words they recognize. Fi-

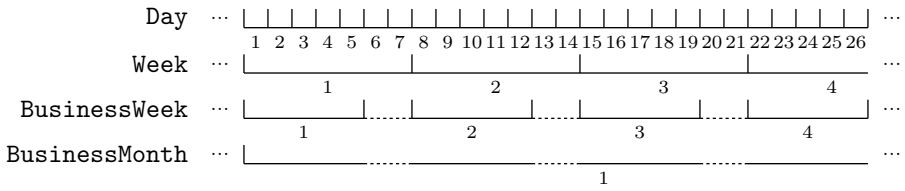

Figure 1. Some examples of time granularities.

nally, in Section 5 we show how the equivalence problem for RLA can be efficiently solved by reducing it to the satisfiability problem for linear diophantine equations with bounds on variables. The last section provides an assessment of the work and outlines future research directions.

2. A Framework for Time Granularity

The idea underlying all different notions of time granularity that have been proposed in the literature is that any time granularity can be viewed as a suitable partition of a fixed temporal domain. The temporal domain is usually assumed to be left-bounded and discrete (for instance, the linear order $(\mathbb{N}^+, <)$). As a matter of fact, one of the main motivations for such an assumption is the observation that most problems of practical interest involve granularities that are ultimately periodic with respect to a fixed bottom granularity and left-bounded (that is, they have an initial granule). It should be also noted that, by viewing $(\mathbb{Z}, <)$ as the disjoint union of $(\mathbb{N}^+, <)$ and its reverse order, it is not difficult to extend any given formalism from the case of left-bounded granularities to the case of bi-infinite granularities (the interested reader can read [19] for an application of such an idea in the field of formal languages and automata).

Definition 1. A *time granularity* is a partition G of a subset T of the temporal domain $(\mathbb{N}^+, <)$ such that for every pair of distinct sets $g, g' \in G$ (hereafter called *granules*), either $t < t'$ holds for all $t \in g, t' \in g'$ or $t' < t$ holds for all $t \in g, t' \in g'$.

The ordering on $\mathbb{N}^+$ induces an ordering on the set of granules of G: given $g, g' \in G$, $g < g'$ holds iff $t < t'$ holds for every $t \in g$, $t' \in g'$. Such an ordering naturally yields a labeling of granules: we say that x is the *label* of a granule $g \in G$, and we write $G(x) = g$, if g is the x-th element of G according to the induced order $<$. Note that Definition 1 captures both granularities that cover the whole temporal domain, such as `Day` and `Week`, and granularities with gaps within and between granules, like, for instance, `BusinessWeek` and `BusinessMonth` (see Figure 1).

Clearly, since the set of all structures that satisfy Definition 1 is uncountable, it is not possible to deal with all possible granularities by means of a finitary formalism. However, by restricting to those granularities that, ultimately, periodically group instants of the temporal domain (most granularity applications are only concerned with such a kind of structures), one can easily succeed in representing and manipulating them through finite objects.

In [20], Wijsen proposes a string-based framework for time granularities. 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. A typical example is the infinite (ultimately periodic) word ■■■■■□□≀■■■■■□□≀..., which represents the granularity `BusinessWeek` over the temporal domain of days. In order to guarantee a one-to-one correspondence between infinite strings and granularities, as well as to ease the treatment of the problems of granularity equivalence and granule conversion, Wijsen introduces an aligned form for string-based specifications of granularities. Such a form forces any separator ≀ to occur immediately after an occurrence of ■. As pointed out by Dal Lago and Montanari [7], if we encode each occurrence of the substring ■≀ by a single symbol ◀, we align the symbols of the string-based representation and the elements of the temporal domain, thus establishing a one-to-one correspondence between strings and granularities. Formally, we say that an infinite word $w \in \{■, □, ◀\}^\omega$ *represents* a granularity G if, for every $t, x \in \mathbb{N}^+$, we have $t \in G(x)$ iff $w[t] \neq □$ and the substring $w[1, t-1]$ contains exactly $x - 1$ occurrences of ◀. In the following, we shall adopt this simplified setting to represent granularities. In particular, we can identify ultimately periodical granularities with ultimately periodic words and we can finitely represent them by specifying their prefix and repeating pattern. Hence, any (finite or infinite) ultimately periodic time granularity can be modeled as an ordered pair (u, v) of finite words over the alphabet $\{■, □, ◀\}$, called *granspec*, where v differs from the empty string ε. As an example, the granularity `BusinessWeek` is represented by the granspec $(\varepsilon, ■■■■◀□□)$.

3. From Strings to Automata

The idea of viewing granularities as ultimately periodic words naturally connects time granularity to the fields of

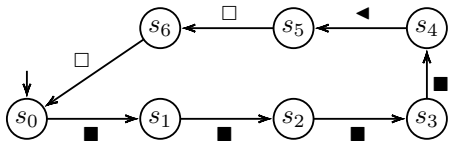

Figure 2. An SSA representing `BusinessWeek`.

formal languages and automata. An automaton-based approach to time granularity, which generalizes the string-based one in several respects, was originally proposed in [7]. The basic idea underlying the automaton-based approach to time granularity is the following one: we take a sequential Büchi automaton M recognizing a *single* infinite word $w \in \{\blacksquare, \square, \blacktriangleleft\}^\omega$ (hence the name single-string automaton) and we say that M represents the granularity G iff w represents G.

Definition 2. A *single-string automaton* (*SSA* for short) is a quadruple $M = (S, \Sigma, \delta, s_0)$, where

- S is a finite set of states,
- Σ is a finite alphabet (usually $\{\blacksquare, \square, \blacktriangleleft\}$),
- δ is a *total* transition function from S to $\Sigma \times S$,
- $s_0 \in S$ is the initial state.

The *run* of M is the (unique) pair $(\mathbf{s}, w) \in S^\omega \times \Sigma^\omega$ such that $\mathbf{s}[1] = s_0$ and, for every $i > 0$, $\delta\big(\mathbf{s}[i]\big) = \big(w[i], \mathbf{s}[i+1]\big)$. We say that w is the infinite word recognized by M if there exists $\mathbf{s} \in S^\omega$ such that the pair $(\mathbf{s}, w)$ is the unique run of M. It is immediate to see that single-string automata capture all and only the ultimately periodic granularities, namely, those granularities that can be represented by granspecs. Figure 2 depicts an SSA representing the granularity `BusinessWeek`.

The equivalence problem for SSA-based representations of time granularities trivially reduces to testing whether two given SSA recognize the same ultimately periodic word (i.e., automata equivalence problem).

Such a problem can be easily solved in *linear time* with respect to the number of states of the involved automata: given two SSA M and N recognizing two ultimately periodic words w_1 and w_2, (i) compute the minimum prefix u_1 (respectively, u_2) and the minimum repeating pattern v_1 (respectively, v_2) of w_1 (resp. w_2), and (ii) test whether $u_1 = u_2$ and $v_1 = v_2$ (notice that this holds iff $w_1 = w_2$, namely, M and N are equivalent SSA).

As regards the prefix of the ultimately periodic word recognized by an SSA, one can exploit the following property to test whether u is the minimum prefix of $w = u \cdot v^\omega$: u is the minimum prefix of $w = u \cdot v^\omega$ iff $u[|u|] \neq v[|v|]$ (if this is not the case, then consider the proper prefix $u[|u|-1]$ instead of u).

As for the repeating pattern of the ultimately periodic word recognized by an SSA, one can exploit Knuth-Morris-Pratt string matching algorithm [14] to compute the first non-trivial occurrence of v in $v \cdot v$ (if v occurs as a substring in $v \cdot v$ starting from the position $i > 1$, then $v[1, i-1]$ is the minimum repeating pattern of $u \cdot v^\omega$, for any finite word u).

These properties lead to a straightforward algorithm that tests the equivalence of two given single-string automata M and N in time $\mathcal{O}(|M| + |N|)$, where $|M|$ and $|N|$ denote the number of states of M and N, respectively.

3.1. Counters and Multiple Transitions

A major limitation of both string-based and automaton-based formalisms is that, whenever the granularity to be represented has a long prefix and/or a long repeating pattern, they produce lengthy representations. As an example, recall that leap years recur with exactly the same pattern every 400 years; then, it is easy to see that the size of any granspec/SSA representing years (or months) of the Gregorian Calendar in terms of days, must have size greater than 10^5. In such cases, computations on representations of time granularities may become rather expensive. In the following, we extend and refine the automaton-based approach by introducing counters in order to compactly encode redundancies of temporal structures. Precisely, we exploit the possibility of activating different transitions from the same (control) state and we rule them through guards envisaging the values of the counters.

Definition 3. An *extended single-string automaton* (*ESSA* for short) is a tuple $M = (S, I, \Sigma, \delta, \gamma, s_0, c_0)$, where

- S is a finite set of control states,
- I is a finite set of counters, whose *valuations* belong to the set $\mathcal{C}_I$ of functions from I to $\mathbb{N}$,
- Σ is a finite alphabet,
- δ is a *total* primary transition function from S to ${\mathcal{C}_I}^{\mathcal{C}_I} \times \Sigma \times S$,
- γ is a *partial* secondary transition function from S to $L_I \times {\mathcal{C}_I}^{\mathcal{C}_I} \times \Sigma \times S$, with L_I being a suitable logical language interpreted over $\mathbb{N}$ with free variables belonging to I,
- $s_0 \in S$ is the initial state,
- $c_0 \in \mathcal{C}_I$ is the initial valuation.

As in the case of single-string automata, the run of an ESSA is unique. In order to formally define it, we need to introduce the notion of configuration. A *configuration* for an ESSA M is a pair state-valuation (s, c), where $s \in S$ and $c \in \mathcal{C}_I$. The transitions of M are taken according to a total function $\Delta_M : S \times \mathcal{C}_I \to \Sigma \times S \times \mathcal{C}_I$ such that

i) if $\gamma(s) = (\varphi, \sigma, a, r)$ and c satisfies φ, then $\Delta_M(s, c) = (a, r, \sigma(c))$,

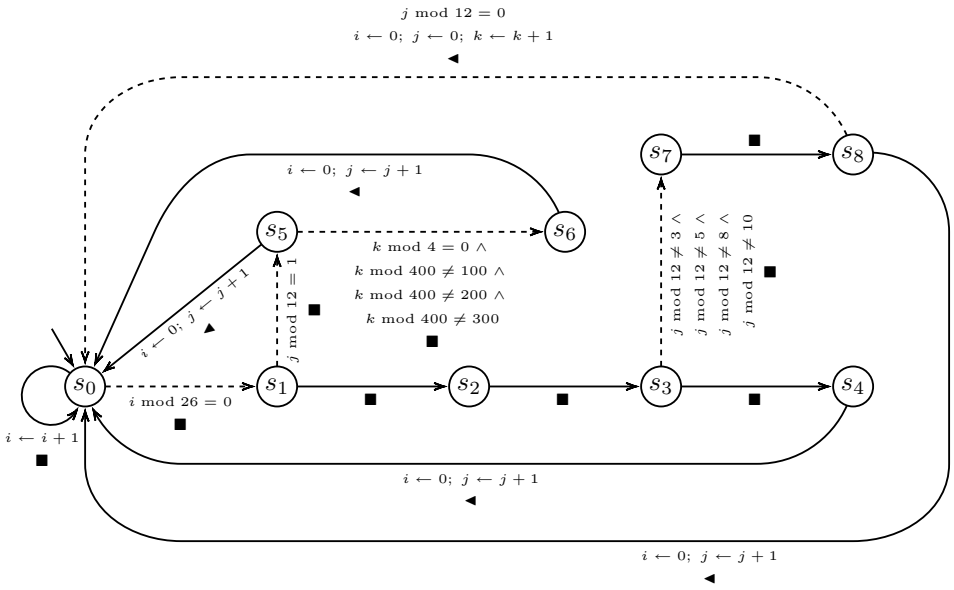

Figure 3. An ESSA representing `Month`.

ii) if $\gamma(s)$ is not defined or c does not satisfy the formula in the first component of $\gamma(s)$, then $\Delta_M(s,c) = (a, r, \sigma(c))$, where $\delta(s) = (\sigma, a, r)$.

Intuitively, a secondary transition of an ESSA is activated if its guard is satisfied by the valuation c; otherwise, a primary transition is activated. The (unique) *run* of an ESSA M is then defined as the triple $(\mathbf{s}, \mathbf{c}, w) \in S^\omega \times \mathcal{C}_I^\omega \times \Sigma^\omega$ such that (i) $\mathbf{s}[1] = s_0$, (ii) $\mathbf{c}[1] = c_0$, and (iii) $\Delta_M(\mathbf{s}[i], \mathbf{c}[i]) = (w[i], \mathbf{s}[i+1], \mathbf{c}[i+1])$ for all $i > 0$. Given an ESSA M and its (unique) run $(\mathbf{s}, \mathbf{c}, w)$, we say that w is the word recognized by M.

An interesting example of ESSA is given by Figure 3 which depicts an ESSA representing the granularity `Month` in terms of the granularity `Day`. The automaton uses three counters, i, j, and k, to store the index of the current day, current month, and current year, respectively. Each counter is initialized to 0, that is, the initial valuation c_0 is such that $c_0(i) = c_0(j) = c_0(k) = 0$. Control states are represented by circles, while transitions are represented by arrows annotated with the update operators. e.g., $j \leftarrow j + 1$, and the recognized symbol, e.g., ■). Primary and secondary transitions are identified by continuous and dashed arrows, respectively; secondary transitions have guards, e.g., $k \bmod 4 = 0$, which are specified as additional annotations of the corresponding dashed arrows.

From the above example, it is clear how ESSA can be exploited to compactly encode redundancies of temporal structures. However, the notion of ESSA is too general to be of practical interest: if we do not restrict the set of admissible formulas and update operators for primary and secondary transitions, several fundamental problems turn out to be undecidable. As an example, if we allow guards of the form $x = 0$ and update operators of the form $x \leftarrow x - 1$ and $x \leftarrow x + 1$, then the halting problem for Minsky (two-counters) machines [16] can be easily reduced to the equivalence problem for ESSA, thus showing that the latter problem is undecidable.

In [7], Dal Lago and Montanari suggest to

i) restrict to guards which are conjunctions of atomic formulas of the form $t_1 = t_2$ or $t_1 \neq t_2$, where both t_1 and t_2 are integer constants or terms of the form $i \bmod d$, with $i \in I$ and $d \in \mathbb{N}^+$;
ii) restrict to update operators which are functional compositions of the basic operators $i \leftarrow 0$ and $i \leftarrow i + 1$, where i ranges in I and the operator $i \leftarrow 0$ (respectively, $i \leftarrow i + 1$) maps a valuation c to the valuation $c[0/i]$ (respectively, $c[c(i) + 1/i]$), with $c[x/i]$ denoting the valuation such that $c[x/i](i) = x$ and $c[x/i](j) = c(j)$ for every $j \neq i$.

The resulting class of automata, called reducible extended single-string automata (shortly RESSA), is expressive enough to compactly encode granularities of practical interest and well behaved, namely, they guarantee decidability results for many relevant problems. As an example, the automaton in Figure 3 is a RESSA. Moreover, one can effectively map a RESSA to an equivalent SSA, thus proving that RESSA are as expressive as (but more compact than) SSA.

The equivalence between RESSA and SSA is obtained by defining, for any given RESSA M, an abstraction rela-

tion over the configurations of M, which turns out to be an equivalence of finite index compatible with the transition function Δ_M (see [2, 10] for similar constructions). Formally, we say that a relation $\sim$ over a (possibly infinite) set X is compatible with a function $f : X \rightarrow X$ iff, for every $x, x' \in X$, $x \sim x'$ implies $f(x) \sim f(x')$. If $M = (S, I, \Sigma, \delta, \gamma, s_0, c_0)$ is a RESSA and, for each counter $i \in I$, d_i is the least common multiple of all constants d that appear inside terms of the form $i \bmod d$ in $\gamma(S)$, then we can define the relation $\cong_M$ over the set $S \times \mathcal{C}_I$ of configurations of M in such a way that $(s, c) \cong_M (s', c')$ iff $s = s'$ and, for all $i \in I$, $c(i) = c'(i) \pmod{d_i}$. It is easy to show that $\cong_M$ is an equivalence of finite index which is compatible with the transition function Δ_M. As a matter of fact, according to the classification introduced by Henzinger and Majumdar [13], this means that RESSA belong to the first class of symbolic transition systems (i.e., infinite-state systems having finite bisimilarity quotients).

One can also prove that there is an exponential bound on the size of the SSA equivalent to a given RESSA. This shows that the equivalence problem for RESSA is in EXPTIME.

3.2. The logical counterpart of RESSA

In [10] Demri describes a logical framework that allows one to express, in a concise way, integer periodicity constraints over a linear temporal domain. The formalism is based on a fragment of Presburger linear temporal logic, denoted PLTL$^{\text{mod}}$. More precisely, the logical language is obtained by combining PLTL (i.e., linear temporal logic with past-time operators) with a suitable first-order constraint language IPC^{++}, whose formulas are built via standard Boolean connectives and existential quantifications, starting from basic atomic formulas of the form $x = d$, $x < d$, $x > d$, $x = y$, $x \equiv_k d$, and $x \equiv_k y + [d_1, d_2]$, where $x, y, ...$ are variables interpreted over $\mathbb{Z}$ and $d, k, d_1, d_2, ...$ are integer constants. Given a valuation $c : \{x, y, ...\} \rightarrow \mathbb{Z}$ for the variables $x, y, ...$, the semantics of an atomic formula is the obvious one:

- $c \vDash (x = d)$ iff $c(x) = d$,
- $c \vDash (x < d)$ iff $c(x) < d$,
- $c \vDash (x > d)$ iff $c(x) > d$,
- $c \vDash (x = y)$ iff $c(x) = c(y)$,
- $c \vDash (x \equiv_k d)$ iff $c(x) \equiv_k d$,
- $c \vDash (x \equiv_k y + [d_1, d_2])$ iff $c(x) \equiv_k c(y) + d$ for some $d_1 \leq d \leq d_2$.

The constraint language IPC^{++} is a strict fragment of Presburger arithmetic [12].

The language PLTL$^{\text{mod}}$ can be viewed as the temporalization (via PLTL) of IPC^{++}. Formally, let $\bigcirc^i x_j$ be the value of the variable x_j at the i-th successor of the current time point. PLTL$^{\text{mod}}$ formulas are PLTL formulas whose propositional letters are replaced with formulas of the form $\varphi[\bigcirc^{i_1} x_{j_1}, ..., \bigcirc^{i_k} x_{j_k}]$, which are obtained by substituting $\bigcirc^{i_l} x_{j_l}$ for all free occurrences of y_l in the IPC^{++}-formula $\varphi(y_1, ..., y_k)$, for $l = 1, \ldots, k$. A model of a PLTL$^{\text{mod}}$-formula is an infinite sequence of valuations, namely, a function of the form $\mathbf{c} : \mathbb{N} \times \{x, y, ...\} \rightarrow \mathbb{Z}$.

As an example, we show the encoding of some granularities of the Gregorian Calendar taken from [10]. These granularities are modeled as infinite sequences of valuations for the corresponding integer variables as follows:

- $sec \equiv_{60} 0 \wedge \Box(0 \leq sec < 60 \wedge \bigcirc sec \equiv_{60} sec + 1)$,
- $min \equiv_{60} 0 \wedge \Box(0 \leq min < 60 \wedge (sec = 59 \rightarrow \bigcirc min \equiv_{60} min + 1) \wedge (sec \neq 59 \rightarrow \bigcirc min = min))$,
- $hour \equiv_{24} 0 \wedge \Box(0 \leq hour < 24 \wedge (min = 59 \wedge sec = 59 \rightarrow \bigcirc hour \equiv_{24} hour + 1) \wedge (min \neq 59 \vee sec \neq 59 \rightarrow \bigcirc hour = hour))$,
- $day \equiv_7 0 \wedge \Box(0 \leq day < 7 \wedge (hour = 23 \wedge min = 59 \wedge sec = 59 \rightarrow \bigcirc day \equiv_7 day + 1) \wedge (hour \neq 23 \vee min \neq 59 \vee sec \neq 59 \rightarrow \bigcirc day = day))$,
- as for the granularities `Month` and `Year`, one can encode them by fixing some end dates far ahead in the time line (such an assumption is necessary since we cannot use constraints like $\bigcirc year = year + 1$ without incurring in undecidability [6, 10]).

Notice that no propositional variables appear in PLTL$^{\text{mod}}$. However, any propositional variable P can be easily encoded by an IPC^{++}-formula of the form $x_P = 1$, where x_P is a fresh variable associated with P.

Demri shows that, like plain LTL but unlike full Presburger LTL, PLTL$^{\text{mod}}$ enjoys a PSPACE-complete satisfiability problem [10]. Such a result is achieved by first defining suitable automaton-based representations for (abstracted) models of PLTL$^{\text{mod}}$-formulas and then by reducing the satisfiability problem to the emptiness problem for these automata.

In [10], Demri establishes an interesting connection between RESSA and linear temporal logics with integer periodicity constraints by reducing the equivalence problem for RESSA to a satisfiability problem for a suitable fragment of PLTL$^{\text{mod}}$. To this end, the guards associated with RESSA secondary transitions are rewritten as Boolean combinations of formulas like $x \equiv_k d$ and $\exists\, z.\ (x \equiv_k z \wedge y \equiv_{k'} z)$, and thus they belong to a (strict) fragment of IPC^{++}, denoted IPC* in [10].

Let PLTL* be the Presburger LTL fragment obtained by combining PLTL and IPC*. Demri shows that the equivalence problem for RESSA is reducible to the satisfiability

problem for PLTL*-formulas or, equivalently, to the emptiness problem for a suitable class of Büchi automata, where the input symbols are atomic IPC*-formulas. The size of the formulas corresponding to a given instance of the equivalence problem for RESSA is shown to be polynomially bounded with respect to the size of the automata, thus proving that the equivalence problem for RESSA is in PSPACE. Such a result improves the previously known EXPTIME upper bound given by Dal Lago and Montanari in [7].

Moreover, in [10] a reduction from the satisfiability problem for quantified boolean formulas to the equivalence problem for RESSA is also given, thus showing that the equivalence problem for RESSA is actually PSPACE-complete.

4. Restricted Labeled Single-string Automata

In this section we introduce a new class of automata, called restricted labeled single-string automata (RLA for short), which are an attempt to find a suitable trade-off between the handiness of SSA and the compactness of (reducible) ESSA [9]. RLA are similar to RESSA, since they exploit counters to compactly encode repeating patterns of time granularities. However, the distinctive feature of this class of automata lies in the structure of the transition functions, which is now more restricted. As an example, we define a uniform policy of counter update. By exploiting such restrictions, we will be able to devise improved algorithms for several problems on time granularities, including the equivalence one.

We now give an intuitive description of RLA structure and behavior. First of all, to simplify the notation and the formalization of properties, labels are moved from transitions to states. Moreover, the set of states is partitioned into two groups, respectively denoted by S_Σ and S_ε. S_Σ is the set of states where the labeling function is defined, while S_ε is the set of states where it is not defined. Furthermore, like in the case of ESSA, we distinguish between two kinds of transition, respectively called primary and secondary transitions. At any point of the computation, at most one (primary or secondary) transition is taken according to an appropriate rule envisaging the state at which the automaton lies and the value of the counter associated with that state. Primary transition functions can be defined in any state, while secondary transition functions are only defined in non-labeled states. A primary transition can be taken in a non-labeled state s only once the secondary transition associated with s has been consecutively taken $c_0(s)$ times, where $c_0(s)$ is the initial valuation for the counter associated with s.

Figure 4 depicts an RLA recognizing the word $(\blacktriangleleft\square^6)^\omega$, which represents the granularity `Monday` in terms of the granularity `Day`. States in S_Σ are represented by Σ-labeled

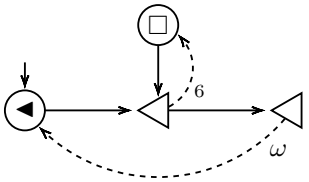

Figure 4. An RLA representing `Monday`.

circles, while states in S_ε are represented by triangles. Primary and secondary transitions are represented by continuous and dashed arrows, respectively. The (initial values of) counters are associated with states in S_ε (for the sake of readability, we depict them as labels of the secondary transitions exiting states in S_ε).

Definition 4. A *restricted labeled (single-string) automaton* (*RLA* for short) is a tuple $M = (S_\Sigma, S_\varepsilon, \Sigma, \Omega, \delta, \gamma, s_0, c_0)$, where

- S_Σ and S_ε are disjoint finite sets of (control) states (hereafter, we shall denote by S the set $S_\Sigma \cup S_\varepsilon$);
- Σ is a finite alphabet;
- $\Omega : S_\Sigma \rightarrow \Sigma$ is a *total* labeling function;
- $\delta : S \rightharpoonup S$ is a *partial* primary transition function whose transitive closure δ^+ is irreflexive (namely, it never happens that $(s, s) \in \delta^+$);
- $\gamma : S_\varepsilon \rightarrow S$ is a *total* secondary transition function such that for every $s \in S_\varepsilon$, $(\gamma(s), s) \in \delta^+$;
- $s_0 \in S$ is the initial state;
- $c_0 : S_\varepsilon \rightarrow \mathbb{N}^+ \cup \{\omega\}$ is the initial valuation.

Counters of RLA range over the natural numbers extended with a special value ω; they can be either set to their initial value or decremented (we tacitly assume that $n < \omega$ for all $n \in \mathbb{N}$ and $\omega - 1 = \omega$).

Let us denote by $\mathcal{C}_{S_\varepsilon}$ the set of all valuations of the form $c : S_\varepsilon \rightarrow (\mathbb{N} \cup \{\omega\})$ for the counters of an RLA $M = (S_\Sigma, S_\varepsilon, \Sigma, \Omega, \delta, \gamma, s_0, c_0)$. A *configuration* for M is a pair (s, c), where $s \in S$ and $c \in \mathcal{C}_{S_\varepsilon}$.

The transitions of M are taken according to a partial function $\Delta_M : S \times \mathcal{C}_{S_\varepsilon} \rightharpoonup S \times \mathcal{C}_{S_\varepsilon}$ satisfying the following conditions:

i) if $s \in S_\Sigma$ and $\delta(s)$ is defined, then $\Delta_M(s, c) = (\delta(s), c)$, namely, if the automaton lies in a labeled state and there is an exiting primary transition, then it takes the primary transition, which does not change the valuation,

ii) if $s \in S_\varepsilon$ and $c(s) > 0$, then $\Delta_M(s, c) = (\gamma(s), c')$, where $c' = c[c(s) - 1/s]$, namely, if the automaton lies in a non-labeled state whose counter has a positive value, then it takes the secondary transition and it decrements the counter by 1,

iii) if $s \in S_\varepsilon$, $c(s) = 0$, and $\delta(s)$ is defined, then $\Delta_M(s, c) = (\delta(s), c')$, where $c' = c[c_0(s)/s]$, namely,

if the automaton lies in a non-labeled state whose counter has value 0 and if there is an exiting primary transition, then it takes the primary transition and it re-initializes the counter,

iv) if none of the above conditions holds, then $\Delta_M(s,c)$ is undefined.

Notice that, since Δ_M may be not defined on some configurations, the run of an RLA may be finite.

The *run* of an RLA M is defined as follows. Let X^∞ be the set of all (finite and infinite) words over X, namely, $X^\infty = X^* \cup X^\omega$. The *run* of an RLA M is the pair $(\mathbf{s},\mathbf{c}) \in S^\infty \times \mathcal{C}^\infty_{S_\varepsilon}$ of maximum (possibly infinite) sequences of states and valuations such that (i) $\mathbf{s}[1] = s_0$, (ii) $\mathbf{c}[1] = c_0$, and (iii) $\Delta_M(\mathbf{s}[i],\mathbf{c}[i]) = (\mathbf{s}[i+1],\mathbf{c}[i+1])$ for all $1 \le i < |\mathbf{s}|$ $(= |\mathbf{c}|)$. Given the RLA M and its run $(\mathbf{s},\mathbf{c})$, one can extract a (finite or infinite) sequence of labeled states $\mathbf{s}_\Sigma \in S^\infty_\Sigma$ by discarding the valuations and the non-labeled states. Such a sequence is said to be the *labeled run* of M. We say that M recognizes the word w iff $w = \Omega(\mathbf{s}_\Sigma)$ (here Ω is extended from states to sequences of states in the natural way).

Notice that Definition 4 allows situations where states and transitions of an RLA form an unconnected (directed) graph. We can overcome these clumsy situations by discarding useless states and transitions. Since counters of reachable configurations range over finite domains, it is immediate to see that RLA recognize either finite or ultimately periodic words.

4.1. RLA-recognizable Words

The solution to the equivalence problem for RLA takes advantage of the following characterization of the words recognized by RLA, based on the notions of δ-*degree* and γ-*degree* of states.

The δ-*degree* of a state $s \in S$ is the (unique) natural number n such that $\delta^n(s)$ is defined, but $\delta^{n+1}(s)$ is not. For instance, the initial state of the automaton of Figure 4 has δ-degree 2. For each non-labeled state $s \in S_\varepsilon$, the γ-*degree* of s is the least $n \in \mathbb{N}$ such that $(\gamma(s), s) \in \delta^n$. For instance, the lower-middle state of the automaton of Figure 4 has γ-degree 1.

The notion of γ-*degree* can then be used to represent the nested structure of RLA transitions in terms of a binary relation Γ_M over the set S_ε defined as follows: $(s,r) \in \Gamma_M$ iff $s = \delta^i(\gamma(r))$, where i is less than the γ-degree of r. Note that the reflexive and transitive closure Γ^*_M is antisymmetric, namely, $(s,r) \in \Gamma^*_M$ and $(r,s) \in \Gamma^*_M$ imply $s = r$. Thus, Γ^*_M can be given the status of a well-founded partial order over the set of non-labeled states. Such a partial order immediately suggests an induction principle, called γ-*induction*, which we will extensively use in both definitions and proofs.

As an example, if we denote by s_0 the initial state of the RLA of Figure 4, by s_1 its successor, by s_2 the top-most state, and by s_3 the right-most state, we have that

- the δ-degree of s_0 (respectively, s_1, s_2, s_3) is 2 (respectively, 1, 2, 0),
- the γ-degree of s_1 is 1 and the γ-degree of s_3 is 2,
- $\Gamma_M = \{(s_1,s_3)\}$ and Γ^*_M consists the pair in Γ_M plus the pairs (s_1,s_1) and (s_3,s_3).

For every state s, let σ^M_s be the finite or infinite word inductively defined as follows:

- if $s \in S_\Sigma$, then $\sigma^M_s = \Omega(s)$,
- if $s \in S_\varepsilon$ and m is the γ-degree of s, then $\sigma^M_s = \left(\sigma^M_{\gamma(s)} \cdot \sigma^M_{\delta(\gamma(s))} \cdot \ldots \cdot \sigma^M_{\delta^{m-1}(\gamma(s))}\right)^{c_0(s)}$.

The well-definedness of σ^M_s directly follows from the principle of γ-induction.

RLA-recognizable words can be characterized as expressions like $(\blacksquare^4\blacktriangleleft\square^2)^\omega$ and $\blacksquare^6((\blacksquare^2\square)^2\square^2)^\omega$, which feature nested repetitions, as stated by the following proposition.

Proposition 1 (Dal Lago, Montanari, Puppis [9])**.** *The word recognized by an RLA $M = (S_\Sigma, S_\varepsilon, \Sigma, \Omega, \delta, \gamma, s_0, c_0)$ is of the form $\sigma^M_{s_0} \cdot \sigma^M_{\delta(s_0)} \cdot \ldots \cdot \sigma^M_{\delta^n(s_0)}$, where n is the δ-degree of s_0.*

4.2. A Complexity Measure for RLA

We now briefly describe a measure of complexity for RLA (for further details, we refer the reader to [9]). Besides the usual complexity measure based on the number of states of the automaton, there is another natural complexity measure which takes into account the nesting structure of RLA transition relations. Such a complexity measure plays a fundamental role in the analysis of main algorithms on RLA [9] and it will be used in the next section in the proof of one basic lemma.

For every state s of an RLA M and every integer n, let $\mathbf{C}^M_{s,n}$ be defined as follows (here we use double induction on s and n, where the ordering for the first, dominant, argument is induced by the relation Γ^*_M):

- if $n < 0$, then $\mathbf{C}^M_{s,n} = 0$;
- if $n \ge 0$, $s \in S_\Sigma$, and $\delta(s)$ is undefined, then $\mathbf{C}^M_{s,n} = 1$;
- if $n \ge 0$, $s \in S_\Sigma$, and $\delta(s)$ is defined, then $\mathbf{C}^M_{s,n} = 1 + \mathbf{C}^M_{\delta(s),n-1}$;
- if $n \ge 0$, $s \in S_\varepsilon$, m is the γ-degree of s, and $\delta(s)$ is undefined, then $\mathbf{C}^M_{s,n} = 1 + \mathbf{C}^M_{\gamma(s),m-1}$;

- if $n \geq 0$, $s \in S_\varepsilon$, m is the γ-degree of s, and $\delta(s)$ is defined, then $\mathbf{C}^M_{s,n} = 1 + max\big(\mathbf{C}^M_{\delta(s),n-1}, \mathbf{C}^M_{\gamma(s),m-1}\big)$.

The *complexity* $\|M\|$ of M is defined as $\|M\| = \mathbf{C}^M_{s_0,n}$, where s_0 is the initial state of M and n is the δ-degree of s_0. It is easy to show that $\|M\| \leq |M|^2$. As an example, the complexity of the automaton in Figure 4 is 6.

The running time of several algorithms operating on RLA-based representations of time granularities, e.g., granule conversion ones, can be expressed in terms of the complexities of the involved automata. This is the case, for instance, with simple algorithms that look for occurrences of particular symbols in the word recognized by a given RLA M, which require time $\mathcal{O}(\|M\|)$. In many cases, the running time of such algorithms turns out to be sub-linear with respect to the number of transitions to be taken to reach the addressed symbol occurrence, thus showing that algorithms working on RLA-based representations outperform those running on equivalent granspecs/SSA.

As an example, by exploiting the optimization algorithms described in [9], we can obtain an RLA representing the granularity `Month` in terms of days with 87 control states and complexity 14. Both these values are significantly less than the size of any equivalent granspec/SSA (see Section 3). Such an automaton is described by the following expression:

$$\Big(\Big(\blacksquare^2(\blacksquare^{28}\blacktriangleleft)^2\Big(\blacksquare^{30}((\blacktriangleleft\blacksquare^{29})^2\blacksquare)^2\blacktriangleleft\Big)^2 \Big(\blacksquare^3(\blacksquare^{27}\blacktriangleleft)^2\Big(\blacksquare^{30}((\blacktriangleleft\blacksquare^{29})^2\blacksquare)^2\blacktriangleleft\Big)^2\Big)^3\Big)^{25} \Big(\Big(\blacksquare^3(\blacksquare^{27}\blacktriangleleft)^2\Big(\blacksquare^{30}((\blacktriangleleft\blacksquare^{29})^2\blacksquare)^2\blacktriangleleft\Big)^2\Big)^4 \Big(\blacksquare^2(\blacksquare^{28}\blacktriangleleft)^2\Big(\blacksquare^{30}((\blacktriangleleft\blacksquare^{29})^2\blacksquare)^2\blacktriangleleft\Big)^2 \Big(\blacksquare^3(\blacksquare^{27}\blacktriangleleft)^2\Big(\blacksquare^{30}((\blacktriangleleft\blacksquare^{29})^2\blacksquare)^2\blacktriangleleft\Big)^2\Big)^3\Big)^{24}\Big)^3\Big)^\omega$$

The above arguments account for the compactness and tractableness of RLA compared to granspecs/SSA.

5. The Equivalence Problem for RLA

In this section we focus our attention on the equivalence problem for RLA-based representations of time granularities. As we previously pointed out, two single-string automata represent the same time granularity iff they accept the same ultimately periodic word. In Section 3 the equivalence problem for RESSA has been shown to be solvable in polynomial space with respect to the size of the input automata. Here we show that in the case of RLA we can devise a more efficient algorithm, which tests the (non-) equivalence of two given RLA in non-deterministic polynomial time. Our solution is based on a reduction of the non-equivalence problem to a number-theoretic problem, precisely, the problem of testing the satisfiability of linear diophantine equations, where variables are constrained by lower and upper bounds.

We start by giving some preliminary definitions. The operations of addition $+$ and multiplication $\cdot$ in $\mathbb{Z}$ can be naturally extended to the power-set $2^{\mathbb{Z}}$ as follows: if $S, T \subseteq \mathbb{Z}$, then $S + T = \{x + y \in \mathbb{Z} \ : \ x \in S, y \in T\}$ and $S \cdot T = \{x \cdot y \in \mathbb{Z} \ : \ x \in S, y \in T\}$. By a slight abuse of notation, we shall write expressions of the form $k \cdot S$, denoting the set $\{k \cdot x \ : \ x \in S\}$. Furthermore, we call *interval* any subset of $\mathbb{Z}$ of the form $[i, j] = \{x : i \leq x \leq j\}$, where $i \in \mathbb{Z} \cup \{-\omega\}$ and $j \in \mathbb{Z} \cup \{\omega\}$.

Intuitively, the idea underlying our solution to the equivalence problem for RLA is to represent the set of positions of all the occurrences of a labeled state in the labeled run of an RLA M by a union of m distinct sets of the form $k_1 I_1 + ... + k_n I_n$, where the values m and n are polynomially bounded with respect to the number of states of M and each I_j is a suitable interval of $\mathbb{Z}$.

Given two RLA M and N, one can decide whether M and N recognize the same infinite word by testing the emptiness of every set resulting from the intersection of two expressions E_1 and E_2, where E_1 represents the positions of the occurrences of an a-labeled state of M and E_2 represents the positions of the occurrences of a b-labeled state of N, with $a \neq b$.

The latter problem can then be reduced to the problem of testing the *non*-satisfiability of some linear diophantine equations with lower and upper bounds on the variables. Even though the satisfiability problem for linear diophantine equations with bounds on variables is known be NP-complete, several solutions that perform well in practice (even on equations with thousands of variables) have been proposed in literature (see, for instance, [1]).

The above argument shows that the equivalence problem for RLA is in co-NP. We provide no proof of the co-NP-hardness of the equivalence problem for RLA. As a matter of fact, we conjecture that the problem can be solved by a deterministic algorithm which takes polynomial time with respect to the size of the input automata. Unfortunately, at the moment we are only able to provide a non-deterministic algorithm for the non-equivalence problem of RLA.

The argument can be formalized as follows. Given an RLA $M = (S_\Sigma, S_\varepsilon, \Sigma, \Omega, \delta, \gamma, s_0, c_0)$, let us denote by $(\mathbf{s}, \mathbf{c})$ its (unique) run and by $\mathbf{s}_\Sigma$ the corresponding labeled run. Without loss of generality, we can temporarily assume that Ω is the identity function, which maps a labeled state s to itself (hence, we have $\Sigma = S_\Sigma$). Such an assumption allows us to think of σ^M_s, as defined in Section 4, as a se-

quence of labeled states, rather than a sequence of symbols. From Proposition 1, we immediately have

$$\mathbf{s}_\Sigma = \sigma^M_{s_0} \cdot \sigma^M_{\delta(s_0)} \cdot ... \cdot \sigma^M_{\delta^n(s_0)},$$

where n is the δ-degree of s_0. In addition, for every non-labeled state $s \in S_\varepsilon$, with γ-degree m, we set:

$$\rho^M_s = \sigma^M_{\gamma(s)} \cdot \sigma^M_{\delta(\gamma(s))} \cdot ... \cdot \sigma^M_{\delta^{m-1}(\gamma(s))}.$$

To keep track of the set of positions of any labeled state, we introduce the notion of (p, q-*succinct*) *linear progression*.

Definition 5. Given a set $P \subseteq \mathbb{Z}$ and two positive integers p, q, we say that P is a *p, q-succinct linear progression* if there exist $m \leq p$, $n_1, ..., n_m \leq q$, $k_{1,1}$, ..., k_{1,n_1}, ..., $k_{m,1}$, ..., $k_{m,n_m} \in \mathbb{Z}$, and some intervals $I_{1,1}$, ..., I_{1,n_1}, ..., $I_{m,1}$, ..., I_{m,n_m} such that $P = \bigcup_{1\leq i\leq m} \sum_{1\leq j\leq n_i} k_{i,j} I_{i,j}$.

For every $s \in S_\Sigma$, every $r \in S$, and every $n \in \mathbb{Z}$ less than or equal to the δ-degree of s, we denote by $P_{s,r,n}$ the set of positions of all the occurrences of s in the sequence $\sigma^M_r \cdot \sigma^M_{\delta(r)} \cdot ... \cdot \sigma^M_{\delta^n(r)}$. Clearly, the set of positions of all the occurrences of s in the labeled run $\mathbf{s}_\Sigma$ of M is $P_{s,s_0,n}$, where n is the δ-degree of s_0.

Now, by exploiting the definition of σ^M_r, we can easily verify the following recursive equations:

- $P_{s,r,n} = \emptyset$, if $n < 0$;
- $P_{s,r,n} = \emptyset$, if $n \geq 0$, $r \in S_\Sigma \setminus \{s\}$, and $\delta(r)$ is undefined;
- $P_{s,r,n} = \{1\}$, if $n \geq 0$, $r = s$, and $\delta(r)$ is undefined;
- $P_{s,r,n} = P_{s,\delta(r),n-1} + 1$, if $n \geq 0$, $r \in S_\Sigma \setminus \{s\}$, and $\delta(r)$ is defined;
- $P_{s,r,n} = \{1\} \cup \big(P_{s,\delta(r),n-1} + 1\big)$, if $n \geq 0$, $r = s$, and $\delta(r)$ is defined;
- $P_{s,r,n} = P_{s,\gamma(r),m-1} + |\rho^M_r| \cdot [0, c_0(r) - 1]$, if $n \geq 0$, $r \in S_\varepsilon$, m is the γ-degree of r, and $\delta(r)$ is undefined;
- $P_{s,r,n} = \big(P_{s,\gamma(r),m-1} + |\rho^M_r| \cdot [0, c_0(r) - 1]\big) \cup \big(P_{s,\delta(r),n-1} + |\sigma^M_r|\big)$, if $n \geq 0$, $r \in S_\varepsilon$, m is the γ-degree of r, and $\delta(r)$ is defined.

The above equations lead to a straightforward procedure $RLAPositions(M, s, r, n)$ that computes (a progression-based representation of) the set $P_{s,r,n}$ for the automaton M.

$RLAPositions(M, s, r, n)$

let $M = (S_\Sigma, S_\varepsilon, \Sigma, \Omega, \delta, \gamma, s_0, c_0)$
if $n < 0$ **then**
  return $\emptyset$
else if $r \in S_\Sigma$ **then**
  if $\delta(r) = \perp$ **then**
    if $r \neq s$ **then**
      return $\emptyset$
    else
      return $\{1\}$
    end if
  else
    $P \leftarrow 1 + RLAPositions(M, s, \delta(r), n-1)$
    if $r \neq s$ **then**
      return P
    else
      return $\{1\} \cup P$
    end if
  end if
else
  $m \leftarrow \gamma$-degree(r)
  $P \leftarrow RLAPositions(M, s, \gamma(r), m-1) + |\rho^M_r| * [0, c_0(r) - 1]$
  if $\delta(r) = \perp$ **then**
    return P
  else
    $Q \leftarrow RLAPositions(M, s, \delta(r), n-1) + |\sigma^M_r|$
    return $P \cup Q$
  end if
end if

The following lemma shows that the set $P_{s,r,n}$ is actually an $|S|^2, |S|^2$-succinct linear progression.

Lemma 2. *For every $s \in S_\Sigma$, every $r \in S$, and every $n \in \mathbb{Z}$ less than or equal to the δ-degree of r, $P_{s,r,n}$ is a $|S|^2, |S|^2$-succinct linear progression.*

Proof. First of all, note that the succinct linear progressions satisfy the following properties (compositionality):

- if P is a p, q-succinct linear progression, then $P + kI$ is a $p, q+1$-succinct linear progression for every integer k and every interval I,
- if P is a p, q-succinct linear progression and P' is a p', q'-succinct linear progression, then $P \cup P'$ is a $p + p', max(q, q')$-succinct linear progression.

Now, on the grounds of the recursive definition of $P_{s,r,n}$, for every pair of states s, r and every integer n, we can write

$$P_{s,r,n} = \bigcup_{1\leq i\leq m} \sum_{1\leq j\leq n_i} k_{i,j} I_{i,j}$$

for some integers $m, n_1, ..., n_m$ and then denote by $p_{r,n}$ (respectively, $q_{r,n}$) a suitable upper bound for m (respectively, for $n_1, ..., n_m$). Clearly, $P_{s,r,n}$ is a $p_{r,n}, q_{r,n}$-succinct linear progression, where $p_{r,n}$ and $q_{r,n}$ may depend on r and n,

but not on s. Moreover, by exploiting the compositionality of succinct linear progressions, we can assume that $p_{r,n}$ and $q_{r,n}$ satisfy the following recursive equations:

- $p_{r,n} = 0$, if $n < 0$;
- $p_{r,n} = 1$, if $n \geq 0$, $r \in S_\Sigma$, and $\delta(r)$ is undefined;
- $p_{r,n} = 1 + p_{\delta(r),n-1}$, if $n \geq 0$, $r \in S_\Sigma$, and $\delta(r)$ is defined;
- $p_{r,n} = p_{\gamma(r),m-1}$, if $n \geq 0$, $r \in S_\varepsilon$, m is the γ-degree of r, and $\delta(r)$ is undefined;
- $p_{r,n} = p_{\gamma(r),m-1} + p_{\delta(r),n-1}$, if $n \geq 0$, $r \in S_\varepsilon$, m is the γ-degree of r, and $\delta(r)$ is defined;
- $q_{r,n} = 0$, if $n < 0$;
- $q_{r,n} = 1$, if $n \geq 0$, $r \in S_\Sigma$, and $\delta(r)$ is undefined;
- $q_{r,n} = 1 + q_{\delta(r),n-1}$, if $n \geq 0$, $r \in S_\Sigma$, and $\delta(r)$ is defined;
- $q_{r,n} = 1 + q_{\gamma(r),m-1}$, if $n \geq 0$, $r \in S_\varepsilon$, m is the γ-degree of r, and $\delta(r)$ is undefined;
- $q_{r,n} = 1 + max(q_{\delta(r),n-1}, q_{\gamma(r),m-1})$, if $n \geq 0$, $r \in S_\varepsilon$, m is the γ-degree of r, and $\delta(r)$ is defined.

Finally, by exploiting double induction on r and n, it is easy to verify that

$$p_{r,n} \leq \left| \left\{ (t,t') \ : \ \begin{array}{c} t' = \delta^i(r),\ 0 \leq i \leq n, \\ t = t' \in S_\Sigma \ \vee \ (t,t') \in \Gamma^*_M \end{array} \right\} \right| \leq |S|^2$$

$$q_{r,n} = \mathbf{C}^M_{r,n} \leq |S|^2.$$

□

Consider now the generic case of an RLA whose labeling function Ω is not necessarily the identity function.

Proposition 3. *For every RLA $M = (S_\Sigma, S_\varepsilon, \Sigma, \Omega, \delta, \gamma, s_0, c_0)$ and every labeled state $s \in S_\Sigma$, the set of positions of all the occurrences of s in $\mathbf{s}_\Sigma$ is a $|S|^2, |S|^2$-succinct linear progression.*

Proof. Given an RLA $M = (S_\Sigma, S_\varepsilon, \Sigma, \Omega, \delta, \gamma, s_0, c_0)$, we define a new RLA $M' = (S_\Sigma, S_\varepsilon, S_\Sigma, \ \Omega', \delta, \gamma, s_0, c_0)$, where $\Omega'(s) = s$ for every $s \in S_\Sigma$. Clearly, the labeled run of M coincides with the labeled run of M'. Therefore, the claim trivially follows from Lemma 2. □

As a consequence of Proposition 3, we have that the set of positions of all the occurrences of a labeled state in the labeled run of an RLA can be effectively represented by a succinct linear progression P, where the number of its terms (i.e., sets of the form $k \cdot I$) is polynomially bounded by the number of control states of the automaton.

Moreover, the values defining each term of P of the form $k \cdot I$ can be represented using polynomial space with respect to the size of the automaton (here the size of the automaton comprises the number of the control states and the size of the initial valuation for the counters).

This basically means that the size of P is polynomially bounded by the size of the automaton. It also follows that $RLAPositions(M, s, s_0, n)$ takes polynomial time with respect to the size of the input.

We conclude the section by showing how to reduce the non-equivalence problem for RLA to the satisfiability problem for linear diophantine equations.

Theorem 4. *Two RLA $M = (S_\Sigma, S_\varepsilon, \Sigma, \Omega, \delta, \gamma, s_0, c_0)$ and $N = (S'_\Sigma, S'_\varepsilon, \Sigma, \Omega', \delta', \gamma', s'_0, c'_0)$ recognize two different words iff there exist a state $s \in S_\Sigma$ and a state $s' \in S'_\Sigma$, with $\Omega(s) \neq \Omega'(s')$, such that*

$$P_s \cap Q_{s'} \neq \emptyset,$$

where P_s (respectively, $Q_{s'}$) is the set of positions of all the occurrences of s (respectively, s') in the labeled run of M (resp. N).

Proof. If M and N recognize two different words, say w and w', we let i be the first position such that $w[i] \neq w'[i]$. We then define $s = \mathbf{s}_\Sigma[i]$ and $s' = \mathbf{s}'_\Sigma$. Clearly, $\Omega(s) \neq \Omega(s')$ holds and i belongs to both P_s and $Q_{s'}$. Conversely, if M and N recognize the same word w, then $w = \Omega(\mathbf{s}_\Sigma) = \Omega'(\mathbf{s}'_\Sigma)$, where $\mathbf{s}_\Sigma$ (respectively, $\mathbf{s}'_\Sigma$) is the labeled run of M (respectively, N). This implies that, for every $s \in S_\Sigma$ and $s' \in S'_\Sigma$, with $\Omega(s) \neq \Omega'(s')$, $P_s \cap Q_{s'} = \emptyset$ holds. □

Let P and Q be two linear progressions. If we write

$$P = \bigcup_{1 \leq i \leq m} \big(k_{i,1} I_{i,1} + ... + k_{i,n_i} I_{i,n_i}\big),$$

$$Q = \bigcup_{1 \leq i \leq m'} \big(h_{i,1} J_{i,1} + ... + h_{i,n'_i} J_{i,n'_i}\big),$$

then we have

$$P \cap Q \neq \emptyset$$

iff, for some $1 \leq i \leq m$ and $1 \leq i' \leq m'$, the following linear diophantine equation with bounds on variables is satisfiable:

$$\begin{cases} k_{i,1} x_{i,1} + ... + k_{i,n_i} x_{i,n_i} = h_{i',1} y_{i',1} + ... + h_{i',n_i} y_{i',n'_{i'}} \\ \forall\ 1 \leq j \leq n_i.\ min(I_{i,j}) \leq x_{i,j} \leq max(I_{i,j}) \\ \forall\ 1 \leq j \leq n'_{i'}.\ min(J_{i',j}) \leq y_{i',j} \leq max(J_{i',j}) \end{cases}$$

Checking the satisfiability of a generic linear diophantine equation with bounds on variables is known to be an NP-complete problem. As a matter of fact, as regards the NP-hardness, one can reduce the well-known *subset sum*

problem (i.e., given a finite set Z of integers, decide whether there exists a subset Z' of Z that exactly sums to 0) to the satisfiability problem for linear diophantine equations. More precisely, given a finite set $Z = \{k_1, ..., k_n\}$ of integers, we define the linear diophantine equation $k_1 z_1 + ... + k_n z_n = 0$, where each variable z_i can be either 0 or 1. It clearly follows that the equation is satisfiable iff Z is a positive instance of the subset sum problem.

Even though the satisfiability problem for linear diophantine equations with bounds on variables is NP-complete, several efficient algorithms, based on non-trivial properties of rings and lattices, have been proposed in the literature, e.g., [1]. These algorithms can solve (systems of) linear diophantine equations with thousands of variables in a reasonable time and thus they can be effectively exploited to test the emptiness of sets resulting from the intersection of two linear progressions.

The following (non-deterministic) algorithm solves the non-equivalence problem for RLA (namely, it has a computation that returns **true** iff the two input automata were not equivalent) by reducing it to the satisfiability problem for linear diophantine equations.

$\underline{RLANonEquivalence(M, N)}$
1: **let** $M = (S_\Sigma, S_\varepsilon, \Sigma, \Omega, \delta, \gamma, s_0, c_0)$
2: **let** $N = (S'_\Sigma, S'_\varepsilon, \Sigma, \Omega', \delta', \gamma', s'_0, c'_0)$
3: $n \leftarrow \delta$-degree(s_0)
4: **for all** $s \in S_\Sigma$ **do**
5: $\quad P[s] \leftarrow RLAPositions(M, s, s_0, n)$
6: **end for**
7: $n' \leftarrow \delta$-degree(s'_0)
8: **for all** $s' \in S'_\Sigma$ **do**
9: $\quad Q[s'] \leftarrow RLAPositions(N, s', s'_0, n')$
10: **end for**
11: **choose** $s \in S_\Sigma, s' \in S'_\Sigma$ **with** $\Omega(s) \neq \Omega'(s')$
12: **if** $P[s] \cap Q[s'] \neq \emptyset$ **then**
13: $\quad$ **return true**
14: **else**
15: $\quad$ **return false**
16: **end if**

6. Conclusions

In this paper, we considered the problem of modeling time granularities and that of testing the equivalence of their specifications. We first showed how to finitely represent ultimately periodic time granularities in terms of automata, starting from the most basic notion of automaton (single-string automaton) and then extending it with counters in order to compactly encode repetitions (extended single-string automaton and restricted labeled single-string automaton).

Then, we focused our attention on equivalence problems for automaton-based specifications of time granularities, proving that (i) the equivalence problem for single-string automata is decidable in linear time, (ii) the equivalence problem for reducible extended single-string automata is decidable in polynomial space and it is complete for such a class (this result is due to Demri [10]), and (iii) the (non-)equivalence problem for restricted labeled single-string automata is decidable in non-deterministic polynomial time.

As for the non-equivalence problem for restricted labeled single-string automata, we exactly showed that it can be reduced to the satisfiability problem for linear diophantine equations with bounds on variables. This latter problem is known to be NP-complete, which immediately provides an upper bound to the complexity of the original problem. However, it remains an open question to establish whether such an upper bound is optimal or not, that is, to establish whether the equivalence problem for restricted labeled single-string automata is co-NP-complete or not. It is conceivable that the equivalence problem for restricted labeled single-string automata may enjoy a deterministic polynomial-time solution, as happens, for instance, for a number of different problems over restricted labeled single-string automata, e.g., granule conversion problems and optimization problems [8, 9].

Acknowledgements

We would like to thank the anonymous referees for their useful remarks and constructive criticisms.

References

[1] K. Aardal, C. A. Hurkens, and A. Lenstra. Solving a system of linear diophantine equations with lower and upper bounds on the variables. *Mathematics of Operations Research*, 25(3):427–442, 2000.

[2] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

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

[4] D. Bresolin, A. Montanari, and G. Puppis. Time granularities and ultimately periodic automata. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 3229 of *LNCS*, pages 513–525. Springer, 2004.

[5] C. Combi, M. Franceschet, and A. Peron. Representing and reasoning about temporal granularities. *Journal of Logic and Computation*, 14:51–77, 2004.

[6] H. Comon and V. Cortier. Flatness is not a weakness. In *Proceedings of the 14th International Conference on Computer Science Logic (CSL)*, volume 1862 of *LNCS*, pages 262–276. Springer, 2000.

[7] U. Dal Lago and A. Montanari. Calendars, time granularities, and automata. In *Proceedings of the 7th International Symposium on Spatial and Temporal Databases (SSTD)*, volume 2121 of *LNCS*, pages 279–298. Springer, 2001.

[8] U. Dal Lago, A. Montanari, and G. Puppis. Towards compact and tractable automaton-based representations of time granularity. In *Proceedings of the 8th Italian Conference on Theoretical Computer Science (ICTCS)*, volume 2841 of *LNCS*, pages 72–85. Springer, 2003.

[9] U. Dal Lago, A. Montanari, and G. Puppis. Compact and tractable automaton-based representations for time granularities. *Theoretical Computer Science*, 373(1-2):115–141, 2007.

[10] S. Demri. LTL over integer periodicity constraints. In I. Walukiewicz, editor, *Proceedings of the 7th International Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, volume 2987 of *LNCS*, pages 121–135. Springer, 2004.

[11] J. Euzenat and A. Montanari. Time granularity. In M. Fisher, D. Gabbay, and L. Vila, editors, *Handbook of Temporal Reasoning in Artificial Intelligence*, pages 59–118. Elsevier, 2005.

[12] S. Ginsburg and E. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.

[13] T. Henzinger and R. Majumdar. A classification of symbolic transition systems. In *Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1770 of *LNCS*, pages 13–34. Springer, 2000.

[14] D. Knuth, J. Morris, and V. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6:323–350, 1977.

[15] B. Leban, D. McDonald, and D. Foster. A representation for collections of temporal intervals. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, volume 1, pages 367–371. AAAI Press, 1986.

[16] M. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

[17] M. Niezette and J. Stevenne. An efficient symbolic representation of periodic time. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 161–168, Baltimore, MD, 1992. ACM Press.

[18] P. Ning, S. Jajodia, and X. Wang. An algebraic representation of calendars. *Annals of Mathematics and Artificial Intelligence*, 36:5–38, 2002.

[19] D. Perrin and P. Schupp. Automata on integers, recurrence distinguishability, and the equivalence and decidability of monadic theories. In *Proceedings of the Symposium on Logic in Computer Science (LICS)*, pages 301–304. IEEE Computer Society, 1986.

[20] J. Wijsen. A string-based model for infinite granularities. In C. Bettini and A. Montanari, editors, *Proceedings of the AAAI Workshop on Spatial and Temporal Granularities*, pages 9–16. AAAI Press, 2000.