On the Equivalence of Automaton-based Representations of Time Granularities

Ugo Dal Lago\textsuperscript{1}, Angelo Montanari\textsuperscript{2}, and Gabriele Puppis\textsuperscript{2}

\textsuperscript{1} Department of Computer Science
University of Bologna, Italy
dallago@cs.unibo.it

\textsuperscript{2} Department of Mathematics and Computer Science
University of Udine, Italy
\{angelo.montanari, gabriele.puppis\}@dimi.uniud.it
Motivations

- **relational databases**: to express temporal information at different time granularities, to relate different granules and convert associated data (queries)

- **artificial intelligence**: to reason about temporal relationships, e.g., to check consistency, validity, and equivalence of temporal constraints at different time granularities (temporal CSPs)

- **data mining**: to discover temporal relationships between collected events, to derive implicit information from such relationships.
Outline

- Introduction (time granularities and their representations)
- The automaton-based approach (Single-string automata)
- The equivalence problem
- The solution for RLA-based representations
Let \((\mathbb{N}^+, <)\) be the underlying **temporal domain**.

**Definition**

A **time granularity** \(G\) is a partition of a **subset** of \((\mathbb{N}^+, <)\) such that, for every pair of distinct sets \(g, g' \in G\) (called **granules**), one of the following two conditions holds:

1. \(g < g'\) (i.e., for all \(t \in g\) and for all \(t' \in g', t < t'\)),
2. \(g > g'\) (i.e., for all \(t \in g\) and for all \(t' \in g', t > t'\)).

**Examples**

| Day   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | ... |
|-------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|   |
| Week  |    |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
|       | 1 | 2 | 3 | 4 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| BusinessWeek |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
|       | 1 | 2 | 3 | 4 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |
| BusinessMonth |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 1  |
We cannot finitely represent *all granularities over an infinite domain* ⇒ we have to restrict ourselves to a proper subclass of structures.

**Possible approaches to time granularity representation**

- **algebraic one:**
  relationships between granularities are represented by algebraic terms built up from a finite set of operators (e.g., Week = \( \text{Group}_7(\text{Day}) \) in the Calendar Algebra)
  

- **logical one:**
  time granularities are defined by models of formulas in a suitable language (e.g., PLTL)
  
We cannot finitely represent *all granularities over an infinite domain* ⇒ we have to restrict ourselves to a proper subclass of structures.

### Possible approaches to time granularity representation

- **string-based one:**
  relationships between time points and granules are encoded by sequences of symbols from a given alphabet (e.g., Granspecs)

  *J. Wijsen. A String-based Model for Infinite Granularities.*

- **automaton-based one:**
  automata are used to encode string-based representations of time granularities (e.g., Single-string Automata)

Basic ingredients of the string-based approach

- A fixed alphabet \{■, □, ◀\}, where
  - ■ represents time points covered by some granule,
  - □ represents gaps within and between granules,
  - ◀ represents the last time point of each granule

- Restriction to ultimately periodic words over \{■, □, ◀\}, namely, to finite granularities or granularities that, ultimately, periodically group instants of the temporal domain.

Example

The infinite word ■■■■◀□□■■■■◀□□■■■■◀□□\ldots

represents the granularity BusinessWeek in terms of Day.

Such a string can be finitely presented by a Granspec, namely, a pair prefix-pattern, such as \(\varepsilon, ■■■■◀□□\).
Connection between ultimately periodic words and automata:

**Proposition**

Any ultimately periodic word is recognized by a **Single-string Automaton (SSA)**, namely, a Büchi automaton accepting a single infinite word.

**Corollary**

*Finite granularities and ultimately periodical granularities can be represented by Single-string Automata.*

An SSA representing **BusinessWeek**
Problem

Representations based on Granspecs and SSA are *too large* with respect to inherently simple structure of granularities.

Possible solution

Use counters and multiple transitions to compactly encode redundancies of time granularities:

- **counters** range over *discrete domains* (e.g., \( \mathbb{N} \)),
- **update operators** modify the values of the counters,
- **guards** rule the activation of **primary transitions** and **secondary transitions**
  (note: *only one transition is enabled at each step*).
An Extended SSA representing Month

\[ i \mod 26 = 0 \]
\[ j \mod 12 = 0 \]
\[ i \leftarrow 0; j \leftarrow 0; k \leftarrow k + 1 \]

\[ k \mod 4 = 0 \land \]
\[ k \mod 400 \neq 100 \land \]
\[ k \mod 400 \neq 200 \land \]
\[ k \mod 400 \neq 300 \]

\[ j \mod 12 \neq 3 \land \]
\[ j \mod 12 \neq 5 \land \]
\[ j \mod 12 \neq 8 \land \]
\[ j \mod 12 \neq 10 \]

\[ i \leftarrow i \land\]
\[ j \leftarrow j + 1 \]

\[ i \leftarrow i + 1 \land\]
\[ j \leftarrow j + 1 \]

\[ i \leftarrow 0; j \leftarrow j + 1 \]
New problem

Extended SSA *do not ease algorithmic manipulation.*

Solution (*Restricted Labeled Single-string Automata - RLA*)

One can introduce suitable restrictions:

- states can be **labeled** (namely, they recognize a symbol)
New problem

Extended SSA do not ease algorithmic manipulation.

Solution (Restricted Labeled Single-string Automata - RLA)

One can introduce suitable restrictions:

- states can be labeled (namely, they recognize a symbol) or non-labeled (in this case, they are assigned a counter),
New problem

Extended SSA do not ease algorithmic manipulation.

Solution (Restricted Labeled Single-string Automata - RLA)

One can introduce suitable restrictions:

- states can be **labeled** (namely, they recognize a symbol) or **non-labeled** (in this case, they are assigned a counter),
- the graph of primary transitions is **acyclic** (**forest graph**),

![Diagram](attachment:image.png)
New problem

Extended SSA *do not ease algorithmic manipulation.*

Solution (**Restricted Labeled Single-string Automata** - RLA)

One can introduce suitable restrictions:

- states can be **labeled** (namely, they recognize a symbol) or **non-labeled** (in this case, they are assigned a counter),
- the graph of primary transitions is **acyclic** (**forest graph**),
- secondary transitions depart from non-labeled states and form **back edges** in the forest graph,
New problem

Extended SSA *do not ease algorithmic manipulation.*

Solution *(Restricted Labeled Single-string Automata - RLA)*

One can introduce suitable restrictions:

- states can be **labeled** (namely, they recognize a symbol) or **non-labeled** (in this case, they are assigned a counter),
- the graph of primary transitions is **acyclic** *(forest graph)*,
- secondary transitions depart from non-labeled states and form **back edges** in the forest graph,
- **uniform policy** of counter update (decrement/reset).
**Definition**

The **equivalence problem** consists in deciding whether two given representations define *the same time granularity*.

**Examples**

The two Granspecs \((\varepsilon, \square\Box\Diamond\uparrow\Box\square)\) and \((\Box\Diamond, \Box\Diamond\uparrow\Box\Diamond\Box\Box\Box\Box\Box\Box\Box\Diamond\uparrow\Box\Box)\) are equivalent.

Similarly, the two Restricted Labeled SSA are equivalent:
As for string-based specification

The equivalence problem reduces to the *pattern matching problem* ⇒ the algorithm is linear in the size of the input Granspecs.

As for automaton-based representations

Trivial (but *inefficient*) solutions exist:

simply unfold the automata into equivalent Granspecs and then use pattern matching algorithms to test equivalence

⇒ *exponential complexity* for both Extended Single-String Automata and Restricted Labeled Single-String Automata.
As for Extended SSA

A better (but still rather inefficient) solution exists:

the equivalence problem is reduced to the *satisfiability problem for PLTL* (i.e., a temporalization of a fragment of Presburger logic)

⇒ the problem turns out to be in PSPACE

(*completeness* proved by using a reduction from the satisfiability problem for quantified boolean formulas).


In the following, we focus on a solution to the equivalence problem for Restricted Labeled Single-String Automata ...
Definition

A chain is a path of primary transitions that goes
- either from the target to the source of a secondary transition

Example

Consider the following automaton:
Definition

A **chain** is a path of primary transitions that goes

1. either *from the target to the source of a secondary transition*

Example

Consider the following automaton:
Definition

A **chain** is a path of primary transitions that goes

1. either *from the target to the source of a secondary transition*

Example

Consider the following automaton:
Definition

A **chain** is a path of primary transitions that goes

1. either *from the target to the source of a secondary transition*

Example

Consider the following automaton:
Definition

A **chain** is a path of primary transitions that goes

1. either *from the target to the source of a secondary transition*
2. or *from the entry point “start” to the deepest state.*

Example

Consider the following automaton:

![Automaton diagram](image-url)
Definition

A **chain** is a path of primary transitions that goes

1. either *from the target to the source of a secondary transition*
2. or *from the entry point “start” to the deepest state*.

An automaton is **sharing** if it contains some *overlapping chains*.

Example

Consider the following automaton:

...it is easily seen to be sharing.
**Lemma**

Any Restricted Labeled SSA can be transformed into an equivalent non-sharing automaton with (at most) a polynomial blowup of states.

**Proof idea**

Simply duplicate overlapping portions of chains:
Fact

Two Restricted Labeled SSA $A$ and $B$ are *not equivalent* iff there exist two distinct symbols $a$, $b$ such that

$$\text{Occ}_A(a) \cap \text{Occ}_B(b) \neq \emptyset$$

where $\text{Occ}_A(a)$ denotes the (possibly infinite) set of occurrence positions of $a$ in the word recognized by $A$.

Proposition

If $A$ is *non-sharing*, then the set $\text{Occ}_A(a)$ can be presented as a **finite union of linear progressions** of the form

$$p_1 C_1 + \ldots + p_n C_n$$

where $p_i \in \mathbb{N}^+$ and $C_i$ is an interval of $\mathbb{N}$ (the presentation uses only polynomial size w.r.t. the size of the automaton).
Example

Consider the non-sharing Restricted Labeled SSA $A$:

$$\text{Occ}_A(\blacksquare) = \begin{align*}
1 & \text{ first position} \\
1 & \text{ loop length} \\
\cdot & \text{ counter interval} \\
\cup & \begin{align*}
8 & \text{ first position} \\
1 & \text{ 1st loop length} \\
\cdot & \text{ 1st counter interval} \\
+ & \begin{align*}
7 & \text{ 2nd loop length} \\
\cdot & \text{ 2nd counter interval}
\end{align*}
\end{align*}
\end{align*}$$
Fact

Testing the intersection of two linear progressions

\[ p_1 C_1 + \ldots + p_n C_n \quad \text{and} \quad q_1 D_1 + \ldots + q_m D_m \]

is equivalent to the problem of testing the satisfiability of the linear diophantine equation

\[ p_1 x_1 + \ldots + p_n x_n - q_1 y_1 - \ldots - q_m y_m = 0 \]

over the bounded variables \( \min(C_i) \leq x_i \leq \max(C_i) \), \( \min(D_j) \leq y_j \leq \max(D_j) \).

Theorem

*The non-equivalence problem for Restricted Labeled SSA is reducible to the satisfiability problem for linear diophantine equations with bounds on variables.*

\[ \Rightarrow \quad \text{The RLA equivalence problem is in Co-NP.} \]
Open problem

Establish whether the non-equivalence problem for Restricted Labeled Single-string Automata is Co-NP-complete or not.

(Note: it is conceivable that the problem may enjoy a deterministic polynomial-time solution)

As a matter of fact, Restricted Labeled Single-string automata turned out to be well suited to algorithmic manipulation:

- polynomial-time algorithms for searching symbol occurrences in the word recognized by an RLA,
- polynomial-time algorithms that compute granule conversions between different time granularities w.r.t. to meaningful relationships (e.g., intersect, cover, covered by),
- polynomial-time algorithms that compute the most compact representation (or the most tractable representation) of a given string-based specification.