

On the Equivalence of Automaton-based Representations of Time Granularities

Ugo Dal Lago¹, Angelo Montanari², and Gabriele Puppis²

¹ Department of Computer Science
University of Bologna, Italy
dallago@cs.unibo.it

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

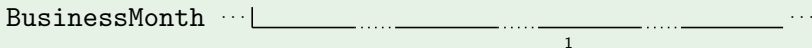
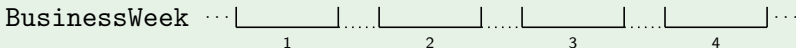
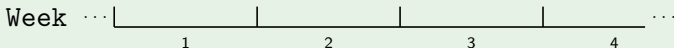
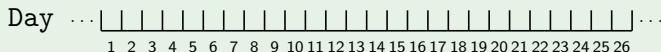
Let $(\mathbb{N}^+, <)$ be the underlying **temporal domain**.

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



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., $\text{Week} = \text{Group}_7(\text{Day})$ in the Calendar Algebra)



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

- **logical one:**

time granularities are defined by models of formulas in a suitable language (e.g., PLTL)



C. Combi, M. Franceschet, A. Peron. *Representing and Reasoning about Temporal Granularities*. *Journal of Logic and Computation*, 2004.

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*.

Proceedings of the AAI Workshop on Spatial and Temporal Granularities, 2000.

- **automaton-based one:**

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



[U. Dal Lago](#), [A. Montanari](#). *Calendars, Time Granularities, and Automata*. Proceedings of the 7th International Symposium on Spatial and Temporal Databases, 2001.

Basic ingredients of the string-based approach

- A fixed alphabet $\{\blacksquare, \square, \blacktriangleleft\}$, 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 $\{\blacksquare, \square, \blacktriangleleft\}$, namely, to **finite granularities** or **granularities that, ultimately, periodically group instants of the temporal domain**.

Example

The infinite word $\blacksquare\blacksquare\blacksquare\blacksquare\blacktriangleleft\square\square\blacksquare\blacksquare\blacksquare\blacksquare\blacktriangleleft\square\square\blacksquare\blacksquare\blacksquare\blacksquare\blacktriangleleft\square\square\dots$ 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 $(\varepsilon, \blacksquare\blacksquare\blacksquare\blacksquare\blacktriangleleft\square\square)$.

Connection between ultimately periodic words and automata:

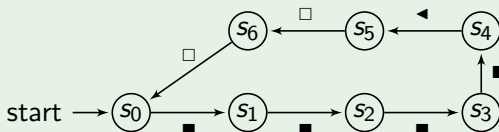
Proposition

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

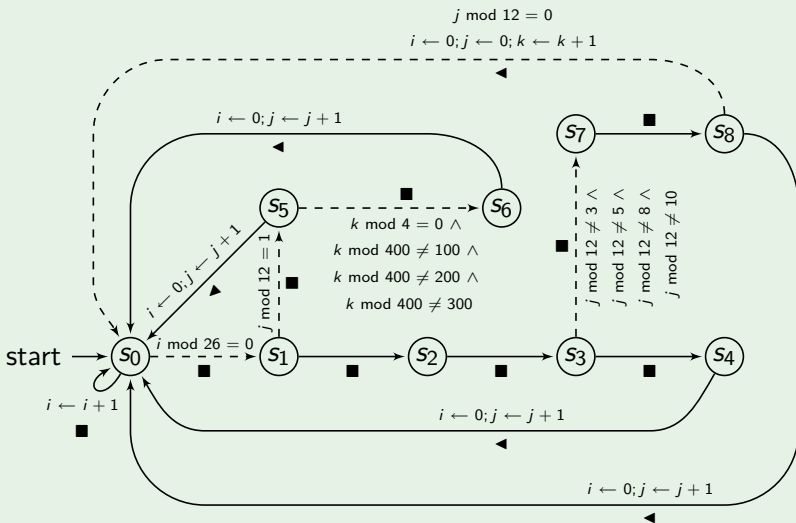
Representations based on Granspecs and SSA are *too large* with respect to inherently simple structure of granularities.

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



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),



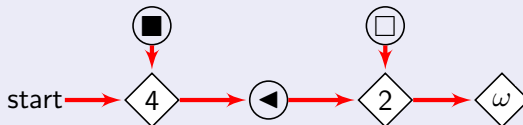
New problem

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**),



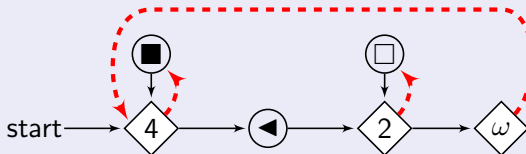
New problem

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,



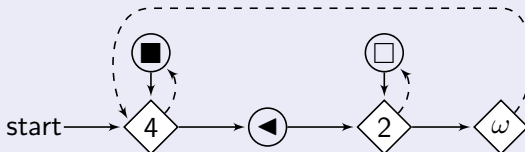
New problem

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).



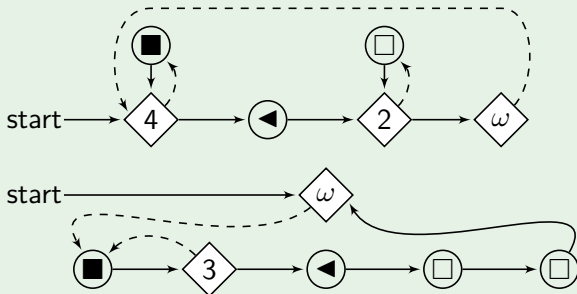
Definition

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

Examples

The two Granspecs $(\varepsilon, \blacksquare\blacksquare\blacksquare\blacksquare\blacktriangleleft\square\square)$ and $(\blacksquare\blacksquare, \blacksquare\blacksquare\blacktriangleleft\square\square\blacksquare\blacksquare\blacksquare\blacksquare\blacktriangleleft\square\square)$ 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).



S. Demri. *LTL Over Integer Periodicity Constraints.*
Proceedings of the 7th International Conference on
Foundations of Software Science and Computation
Structures, 2004.

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

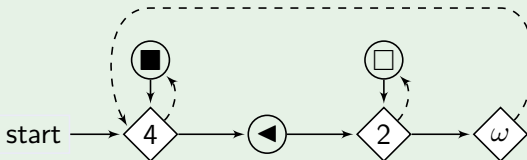
Definition

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

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

Example

Consider the following automaton:



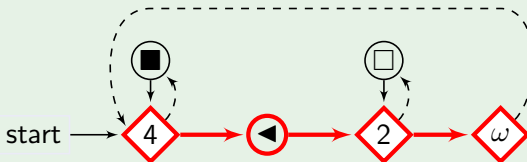
Definition

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:



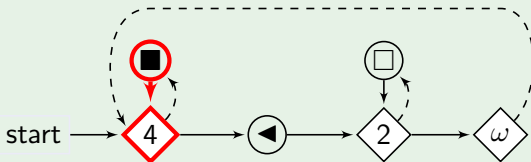
Definition

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:



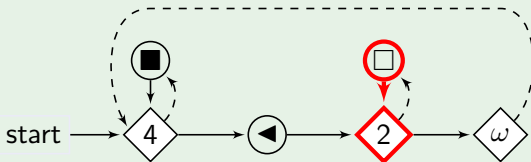
Definition

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

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

Example

Consider the following automaton:



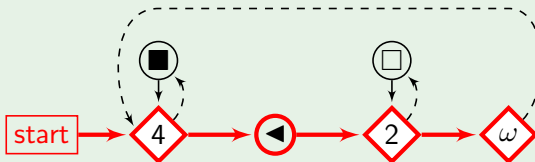
Definition

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:



Definition

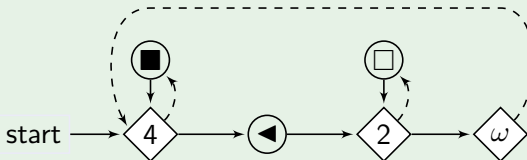
A **chain** is a path of primary transitions that goes

- 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:



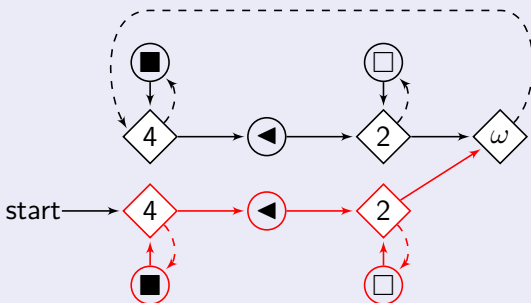
...it is easily seen to be sharing.

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 \mathcal{A} and \mathcal{B} are *not equivalent* iff there exist two distinct symbols a, b such that

$$\text{Occ}_{\mathcal{A}}(a) \cap \text{Occ}_{\mathcal{B}}(b) \neq \emptyset$$

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

Proposition

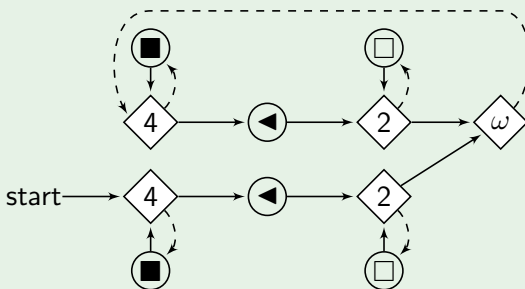
If \mathcal{A} is *non-sharing*, then the set $\text{Occ}_{\mathcal{A}}(a)$ can be presented as a **finite union of linear progressions** of the form

$$p_1 C_1 + \dots + p_n C_n$$

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 \mathcal{A} :



$$Occ_{\mathcal{A}}(\blacksquare) = \underset{\substack{\text{first} \\ \text{position}}}{1} + \underset{\substack{\text{loop} \\ \text{length}}}{1} \cdot \underset{\substack{\text{counter} \\ \text{interval}}}{[0, 3]}$$

$$\cup \underset{\substack{\text{first} \\ \text{position}}}{8} + \underset{\substack{1^{\text{st}} \text{ loop} \\ \text{length}}}{1} \cdot \underset{\substack{1^{\text{st}} \text{ counter} \\ \text{interval}}}{[0, 3]} + \underset{\substack{2^{\text{nd}} \text{ loop} \\ \text{length}}}{7} \cdot \underset{\substack{2^{\text{nd}} \text{ counter} \\ \text{interval}}}{[0, \omega[}$$

Fact

Testing the intersection of two linear progressions

$$p_1 C_1 + \dots + p_n C_n \quad \text{and} \quad q_1 D_1 + \dots + q_m D_m$$

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

$$p_1 x_1 + \dots + p_n x_n - q_1 y_1 - \dots - q_m y_m = 0$$

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.

⇒ *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.