An algebraic theory of real-time formal languages

Catalin Dima

UNIVERSITÉ JOSEPH FOURIER - GRENOBLE 1
SCIENCES ET GEOGRAPHIE

THÈSE

pour obtenir le grade de
DOCTEUR de l'UNIVERSITÉ JOSEPH FOURIER
Spécialité: Informatique

présentée et soutenue publiquement par
M. Cătălin DIMA
le 11 décembre 2001

THÉORIE ALGÉBRIQUE DES LANGAGES FORMELS TEMPS RÉEL

Directeurs de thèse: Prof. Eugène Asarin, Dr. Oded Maler

Composition du jury :	Jean-Claude Fernandez	Président
	Paul Gastin	Rapporteur
	Nils Klarlund	Rapporteur
	P. S. Thiagarajan	Examinateur
	Pascal Weil	Examinateur
	Eugène Asarin	Directeur de Thèse
	Oded Maler	Directeur de Thèse

Thése preparée au sein du Laboratoire Vérimag

Lui Gabi si Iuliei, cu drag

Remerciements

Je remercie d'abord à Oded Maler et Eugène Asarin pour m'avoir donné la chance de finir mes études sous leur direction et soutenir ma thèse à Vérimag. Eugène a patiemment résisté aux tous mes essais échoués et ses critiques m'ont fait apprendre un nouveau sens de la recherche.

Je remercie à Joseph Sifakis pour m'avoir accueili pendant plus d'une année au sein du laboratoire Vérimag, en me donnant ainsi la possibilité de faire la recherche dans un milieu effervescent, entretenu par des chercheurs de haute qualité.

Je remercie aussi à Alain Girault et aux membres du groupe BIP de l'INRIA Rhône-Alpes, où j'ai été accueilli pendant une année en 2000. C'est eux qui on fait mon accomodation plus facile. Grâce a eux, j'ai pu poursuivre mes recherches pour la thèse, en parallèle avec mon travail dans le cadre du projet TOLÈRE.

Je remercie à Gheorghe Ştefănescu qui m'a toujours poussé à contacter diverses groupes de recherce et à qui je dois en fait mon arrivée en France.

Many thanks to the UNU/IIST, in particularly to Prof. Zhou Chaochen. It was there, during my fellowship at UNU/IIST from Febrary to August 1998, that my interest for the theory of real-time systems emerged. I also thank the TCS group at TIFR Mumbai for giving me the means to visit from September to December 1998. Xu Qiwen and Dang Van Hung, at UNU/IIST Macau, and Paritosh Pandya at TIFR Mumbai have helped me a lot in the early stages of the research for the thesis.

Merci à Liana et Marius pour leur chaleureuse amitié. Merci à Yasmina et Moez, qui sont des bons copains de bureau, et à Ana et Gerardo qui sont des bons copains tout court.

Merci finalement à tous les chercheurs du Vérimag pour cet environnement de bonne qualité qu'ils entretiennent.

Table of Contents

1. Introduction

Formal methods make themselves increasingly needed in a wide range of areas of computer science, from hardware specification and verification to the design and validation of computer systems. They are especially needed when critical properties of systems have to be insured. Within formal methods, the two main directions for producing evidence of the correct design of a system are the *model checking* approach and the *theorem proving* approach.

Within the model checking approach, systems are usually modeled as automata (very frequently with a finite state space) and properties are themselves specified in a descriptive language, such as logic or process algebra. For a large subclass of *safety* properties, the model-checking problem can be reformulated as the problem of checking whether the language of an automaton is empty.

Two basic features a specification language needs are sequentiality and parallelism, which in the automata model translate to concatenation, resp. intersection of automata. Whereas sequentiality interacts optimally with emptiness checking, parallelism brings in the well-known "state space explosion problem". Sequentiality is well studied and understood, while parallelism still raises problems at both theoretical and practical level. Regular expressions [Kle56] are among the most basic specification language. They model, however, only the sequential structure of systems. Still regular expressions are able to represent also parallel structure, due to the intersection construction and the celebrated Kleene theorem.

Timed systems and their automata model: timed automata

Timed systems (or real-time systems) are computer systems in which the components interact continuously with one another and with the environment, in order to provide a certain service in which time plays an important rôle. This rôle might be bounded response to some stimuli, limited duration of execution of tasks, and so on.

The now classical automata model for timed systems is the *timed automata* model [AD94]. Timed automata are finite automata enhanced with the possibility to record time passage, by means of real-valued clocks. Clocks evolve synchronously at rate 1, and transitions are taken when some simple arithmetic conditions on the clocks are met, and some transitions might reset some clocks to 0.

A wealth of algorithms ([Yov98, LPWY95] give surveys) and dedicated tools [BDM+98, LPY97] are now available for model-checking with timed automata. The main problem which limits the efficiency of any algorithm for model checking with timed automata is that the emptiness

problem for timed automata, though a decidable problem, has a very high complexity (PSPACE-complete), hence being even harder than model-checking for untimed systems.

On the specification side, several process algebras with time have been proposed in the beginning of the 90's [WY91, NSY93, BB91]. The semantics of these algebras rely upon timed automata.

Curiously, the search for *regular expressions* that allow specification of timing behaviors succeeds the concern for process algebras (though, in the untimed case, it is regular expressions that have preceded and issued process algebras [Mil80]). Only recently there have been issued several results for timed automata [ACM97, BP99], or for subclasses [Dim99b] or superclasses [BP01] of timed automata.

Timed regular expressions [ACM97] are a very convenient specification language for timed systems. They are regular expressions enhanced with the possibility to express intervals between two moments during the computation, by the use of interval-labeled parentheses. A left parenthesis corresponds to resetting a clock and a right parentheses, labeled with an interval I, corresponds to checking whether the clock value is in the interval I.

In spite of their elegance in use, timed regular expressions bear some expressiveness problems: intersection and renaming are essential in proving the reverse implication of the Kleene theorem for timed regular expressions and timed automata.

Subject and contributions of the thesis

In our thesis we study the relationship between timed automata and timed regular expressions.

In the first part, we study the simpler case of timed regular expressions in which we bound only state duration. The automata associated with this class behave like finite automata, most notably being closed under negation and algebraically definable via inverse monoid morphisms.

In the sequel we try to expand the technique developed in the first part for the whole class of timed automata. We start from the consideration that the parallel composition operation on automata destroys the sequential structure. Our idea is to drop the intersection operator from timed regular expressions by using *colored parentheses*, in which each color corresponds to one clock. The feature brought in by this idea is that the structuring of the specification would be preserved to a certain extent. Also renaming is no longer necessary. However this idea brings in some difficulties as well, mainly a different view of sequentialization.

In our calculus, an atomic regular expression contains parentheses of different colors. If we apply to such an atom a "color filter" which retains only parentheses of a certain color and deletes the other colors, we would get a timed regular expression of the form $E_1\langle E_2\rangle_I E_3$. Here E_1, E_2 and E_3 are untimed (i.e. nonparenthesized) regular expressions and I is some interval. The semantics of such an atom consists of signals in which a number of points have been distinguished: two points per each color, one "startpoint" for resetting the clock associated with the color, and one "endpoint" for checking the value of the clock.

This special form of atoms has nevertheless a huge expressive power in combination with the concatenation operation. This is a *partial* operation which allows two signals with distinguished points to be concatenated iff the distinguished endpoint for each color in the first signal matches the distinguished startpoint for the same color in the second signal.

In the second part of the thesis we study the algebraic structure of signals with distinguished points and the regular expressions with colored parentheses that represent sets of such signals. We prove that the emptiness problem is undecidable for regular expressions with colored parentheses, the problem lying in their untimed structure.

We then study this untimed structure by associating, for regular expressions with n colors, a class of finite automata with $2n$ accepting sets, which we call $2n$*-automata*. The idea is to have two accepting sets for each color: one for the startpoint associated with the color and one for endpoint for that color.

We show that the class of $2n$-automata is closed under union, intersection, concatenation and shuffle. The central theorem of this thesis is then that, under mild assumptions, $2n$-automata are also closed under star. On the other hand, we show that these automata can be used to represent timing information in the regular expressions. In other words, they can be used for representing constraints over the real domain. The idea is that each *run* in an automaton represents a *clock region*, in the sense of Alur and Dill [AD94].

We also show that the mild assumptions necessary for star closure are satisfied when modeling timed automata. As a consequence, we provide a method for checking whether the language denoted by a given regular expression with colored parentheses is empty.

As an auxiliary result, our technique allows the computation of *reachability relations* defined by timed automata. These are the relations on clock values defined by the behaviors of timed automata, such as starting from one state and reaching another. The computation of such relations is useful in verification, since the language accepted by a timed automaton is empty iff the reachability relation defined by initial and final states is the empty relation.

Summarizing, the main contributions of our thesis are the following:

- The presentation of a new class of regular expressions that generalize the timed regular expressions of [ACM97]. Our regular expressions do not need renaming or intersection and are more expressive than timed automata.
- The introduction of a new class of finite automata with $2n$ accepting states, automata that correspond to the atomic regular expressions that we utilize. The emptiness problem for these automata is decidable, but NP-complete.
- The translation of regular expressions into a class of finite automata with $2n$ accepting sets. This translation works for regular expressions bearing a certain *non-elasticity* property. This gives a method for checking for emptiness the semantics of a regular expression with the non-elasticity property.
- A new method for checking emptiness of a timed automaton by constructing the regular expression for it.

- A new method for computing the reachability relations defined by timed automata, based upon the same class of n-automata.
- A collateral result is that our finite automata give a new method for representing general clock constraints. In some cases, clock constraints are represented more compactly than with existing methods.

  The detailed study of a data structure that embodies this representation method is not the subject of this thesis.

Related work

As we have already mentioned, our study has started from the results in [ACM97, ACM01, Asa98] which introduce timed regular expressions, and [Her99], which shows the necessity of renaming in timed regular expressions. A different approach to regular expressions is given in [BP99, BP01]. The regular expressions in these studies do not need intersection or renaming, being based upon atoms of the type (a, C, X) for some symbol a, clock constraint C and reset set X. The paper [BP99] gives a variant of this, in which the reset sets are shifted to the concatenation symbols, hence having a whole range of *indexed concatenations*. However none of these papers study in detail the algebraic structure on which the semantics of regular expressions is based, and neither they give the possibility to lift the semantic operations of concatenation and star to syntactic operations on atoms.

In fact, it can be observed that, with the clock valuation semantics, if one tries to lift concatenation and star at a syntactic level then he would run into problems with the representation of the results. More specifically, timing in the atomic regular expressions of [BP99] or [BP01] is specified by clock constraints which utilize *nondiagonal* constraints of the type $x_i' - x_i = y_i' - y_i$ (this is an expression which says that the clocks x_i and y_i evolve synchronously). As a consequence, the "zones" in the $2n$-th dimensional space which satisfy such constraints are no longer unions of *clock regions* [AD94] and hence need representations for more general polyhedra. But it is known [Sor01] that general polyhedra-based representations are less efficient that representations which take advantage of the fact that the polyhedra in discussion are unions of regions.

The study whose results are perhaps the closest to our approach is Yan Jurski's PhD thesis [Jur99] (see also the journal version [CJ99]). In his thesis, Jurski proves that the reachability relation in timed automata is expressible in Pressburger arithmetic. The technique employed in his work can be characterized as a generalization of constraint graphs (which are the graphical representation of DBMs), with a construction of the "star" of a constraint graph. The problem with this approach is that constraint graphs cannot record "disjunctive information": they can only record conjunction of "diagonal" constraints over clocks, i.e., of the type $x_i - x_j \in I$. Whereas the star is naturally built as an infinite disjunction. Therefore Jurski needs to "flatten" each timed automaton, such that no nested loops be allowed, and only afterwards apply his star construction. On the contrary, our n-automata can record disjunctive information too (the set of accepting runs is a *union*) and therefore we may iterate the star closure theorem without any problem. Besides this,

our presentation allows not only expressing the timing behavior of timed automata, but also the representation of both the timing and untimed information in the same expression. Finally, Jurski's result is limited to timed automata *without diagonal constraints*, and it is not clear whether this restriction is essential or not.

We may also mention the approach on using Pressburger arithmetics and its decision procedures in systems with infinite state-space [Boi99, WB00]. [BC96] report on solving systems of linear equations over the integers by coding integer solutions into finite words and using finite automata to accept such words. The idea dates back to Büchi's work on *weak monadic second order theory of one succesor function* (WS1S) [Büc60], see also the two comprehensive handook articles [Tho90, Tho97] on the subject. We note here that our coding of integer solutions of constraints (which are nothing else but systems of linear equations) is different from the one used by [BC96, Boi99]. Our coding takes advantage of the *quasidiagonal* format of systems of linear equations which are associated with the clock constraints.

There are already several data structures that are used for reachability algorithms for timed automata [Tri98, LWYP99, ABK$^+$97, MLAH99], to cite a few only. Most of them are based upon the DBM technique, and DDDs are generalizations of BDDs [Bry86] for representing clock constraints. Our automata-based technique, with constraints regarded as runs in a finite automaton, is therefore new and might yield new data structures for reachability algorithms.

Let us also mention that there is a whole theory concerning constraint propagation [DMP91], which is an essential ingredient for the representation of reachable states [Tri98]. The most general way of looking at these is perhaps the $(\max, +)$-algebra [Gau99, GP97]. However $(\max, +)$-algebra does not deal with the possibility to chain timing constraints, that is, to specify algebraically the behavior of timed automata.

Another related work that we might mention here is the study [CG00] on employing periodic constraints in timed automata. Our presentation of timed automata requires that constraints use only intervals, but our theory of regular expressions allows the use of periodic constraints.

We finally mention the interest for Kleene theorems for subclasses of timed automata [Dim99b], or for superclasses [BP01].

Organization of the thesis

The thesis is divided in 10 chapters, including this introductory one and a conclusion chapter.

2. In the second chapter we give some basic properties of signals and timed languages. We prove here that the monoid of signals and the monoid of timed words are not "algebraically" related, and that the idea of producing timed languages via inverse morphisms from finite monoids issues only languages with no timing information.
3. In the third chapter we study the special class of one-clock timed automata, called *real-time automata*, in which the clock is reset at each transition. We show that language emptiness and universality are decidable, we give a "pumping lemma" characterization of the associated class

of languages, and show that, by utilizing a "stuttering-free concatenation" on the set of signals, we get exactly the class of languages accepted by real-time automata.

4. In the fourth chapter we review briefly the notion of timed automata and the possibility to have a Kleene theorem for them. The regular expressions we utilize here are taken from [BP99, BP01]. They involve clock constraints and resets, and therefore they are easily related to timed automata. However it is not the class of expressions which we aim to study, the reason (given in chapter 5 also) being that clock usage in expressions is specific to low-level specification languages, while regular expressions are meant to be a high-level specification language.
5. In the fifth chapter we review the timed regular expressions of [ACM97] and discuss their problems. We also present here our ideas for solving these problems - usage of colored parentheses and of a partial concatenation operation. This chapter is meant as an intuitive presentation of the problems we have sought to solve and the solution we have found. We also give here an undecidability result concerning timed regular expressions with negation.
6. In the sixth chapter we introduce and study our algebraic framework of signals with distinguished points. These signals are given a matricial presentation, mainly by similarity to Difference Bound Matrices [Bel57], which are, on their own, a subject of discussion in chapter 8. We define the partial concatenation operation on signals and establish some basic algebraic properties for it. Wee introduce concatenation by means of two "more basic" operations: *juxtaposition*, which can be thought as "conjugating" the two signals and *projection*, which can be thought as quantification. A first try to lift this operation at the specification level, that is, to provide a calculus with regular expressions with colored parentheses, is shown to fail, the reason being that projection is not compositional. More specifically, there is no way to define the projection operation on regular expressions, such that the semantics of the projection be the projection of the semantics. An equally worrying result is the undecidability of the emptiness problem for the general class of regular expressions with colored parentheses. This result follows by showing that the Post Correspondence Problem [Pos46] can be reduced to the emptiness problem for regular expressions with colored parentheses. Hence the undecidability problem is hidden in the untiming structure of the regular expressions.
7. In the seventh chapter we investigate the class of n-automata for their possibility to represent regular expressions with colored parentheses, but over a *discrete* time domain. The necessity of this study is emphasized by the undecidability result. We provide here a mild property – the *non-elasticity* property – that assures star closure of automata, and hence accomplish in part the task of representing regular expressions.
8. The eighth chapter is concerned with the generalization of these results for *continuous* time domains, in particular with the possibility to represent timing information in a continuous time domain with the automata defined in chapter 7. This approach is successful and provides the possibility to represent timing constraints in the continuous domain by n-automata.
9. In the ninth chapter we gather together all the results obtained so far in order to provide a compositional calculus with regular expressions with colored parentheses. In this chapter we also show that the non-elasticity property discovered in chapter 7 is satisfied by the regular

expressions which encode timed automata. We then provide a method for checking language emptiness in timed automata, by transformation to regular expressions.

Each chapter starts with a short presentation of the problems and solutions that are treated within it.

2. Signals and their basic properties

In this chapter we study some of the algebraic properties of signals and timed words.

Signals and timed words are the two alternative models for the behavior of timed systems. While signals put the accent on states in which the system is and on state durations, timed words put the accent on actions that a system is executing and on moments at which actions take place. We take the approach of [ACM01] and present these monoids as coproduct monoids – or, in an alternative terminology, as direct sums. This algebraic presentation makes some proofs more succinct.

Since the two notions, signals and timed words, try to model the same phenomena it is natural to search a connection between monoids of signals and monoids of timed words. In this chapter we prove that this connection is *not* of an algebraic nature: we prove that the monoid morphisms between the monoid of signals and the monoid of timed words are unable to relate state changes to actions. More formally, we prove that each signal is mapped, by such a monoid morphism, into a timed word *with no action*.

The second result of this chapter concerns the nonexistence of a Myhill-Nerode characterization of timed languages. We prove that any timed language (i.e. set of signals) that can be defined as the inverse image of a subset of a finite monoid, does not carry any timing information. That is, whenever a signal σ is in the language, any other signal with the same sequence of states (and with any other durations of these states) is in the language too. This property is based upon a lemma stating that there are at most two morphisms from the monoid of nonnegative reals to any finite monoid: the trivial morphism and, in the eventuality the target monoid has a "zero", the morphism which takes any positive number to this "zero". Hence the problem is traced to the "stuttering" structure of the real numbers, and we will see in the next chapter that, if we allow two signals to concatenate only when at the concatenation point they create a discontinuity, then the "inverse morphisms" approach will produce timed languages with nontrivial timing information.

This second result is proved using a "diagram-chasing" technique specific to category theory [Mac71]. This proof takes advantage of the algebraic presentation of signals and we believe it is drastically shorter than any other proof, that would need to mimic the uniqueness properties of the monoid of signals.

The chapter is organized as follows: the first section presents some basic properties about monoids, especially the construction of the coproduct (or direct sum) of a family of monoids. We also remind here the notion of Kleene algebra. We also incude in this section a short subsection recalling the definition of Kleene algebras. Then in the second section we recall the coproduct representation of signals and the fact that, similarly to languages of words, the powerset of signals

can be organized as a Kleene algebra. The third section presents also timed words as a coproduct monoid and gives the negative result about the monoids of signals and timed words. The fourth section is concerned with the other negative result, concerning the "lack of interest" of timed languages defined by inverse monoid morphisms.

2.1 Basic notions

In this section we remind the notion of coproduct (or direct sum) of a family of monoids, and the notion of Kleene algebra.

2.1.1 Coproduct monoids

Definition 2.1.1. *The **coproduct** of a family of monoids $(M_i, \cdot, e_i)_{i\in\mathcal{I}}$ is the monoid $(M, \cdot, e)$ defined as the quotient: $M = \left(\coprod_{i\in\mathcal{I}} M_i\right)^* \Big/ _{\sim}$ where*

- $\coprod$ *is the usual disjoint sum:* $\coprod_{i\in\mathcal{I}} M_i = \{(m_i, i) \mid i \in \mathcal{I}, m_i \in M_i\}$*;*
- $\left(\coprod_{i\in\mathcal{I}} M_i\right)^*$ *is the free monoid over* $\coprod_{i\in\mathcal{I}} M_i$ *(with concatenation denoted as juxtaposition and empty sequence as ε);*
- *and $\sim$ is the congruence on $\left(\coprod_{i\in\mathcal{I}} M_i\right)^*$ generated by the equations:*

$$(m_i, i)(m'_i, i) \sim (m_i \cdot m'_i, i) \qquad \forall m_i, m'_i \in M_i, i \in \mathcal{I} \tag{2.1}$$

$$(e_i, i) \sim (e_j, j) \qquad \forall i, j \in \mathcal{I} \tag{2.2}$$

$$(e_i, i) \sim \varepsilon \qquad \forall i, \in \mathcal{I} \tag{2.3}$$

We denote the coproduct of the family $(M_i, \cdot, e_i)_{i\in\mathcal{I}}$ as $\bigoplus_{i\in\mathcal{I}} M_i$. Note that the unit of $\bigoplus_{i\in\mathcal{I}} M_i$ is the class of ε. we denote this unit as e.

Observe that each element $\mathbf{m} \in \bigoplus_{i\in\mathcal{I}} M_i$ can be uniquely represented as a finite concatenation of "atomic" elements

$$\mathbf{m} = [m_{i_1}, i_1] \cdot \ldots \cdot [m_{i_k}, i_k] \tag{2.4}$$

in which $[m_{i_j}, i_k] \neq e$ for all $j \in [1 \ldots k]$.

Theorem 2.1.2. *The monoid $\bigoplus_{i\in\mathcal{I}} M_i$ has the following* universality property*: for any monoid $(M, \cdot, e)$ and family of morphisms $\phi_i : M_i \to M$, there exists a unique morphism $\phi : \bigoplus_{i\in\mathcal{I}} M_i \to M$ such that $\phi([m_i, i]) = \phi_i(m_i)$. This morphism is denoted $\langle\phi_i\rangle_{i\in\mathcal{I}}$ and is called the **coproduct** of the family $(\phi_i)_{i\in\mathcal{I}}$.*

This theorem is depicted in Figure 2.1. Here, ι_i denotes the inclusion morphism.

The construction of the coproduct morphism is the following: each element $\mathbf{m} \in \bigoplus_{i\in\mathcal{I}} M_i$ is decomposed as in Identity 2.4, $\mathbf{m} = [m_{i_1}, i_1] \cdot \ldots [m_{i_k}, i_k]$. We then put:

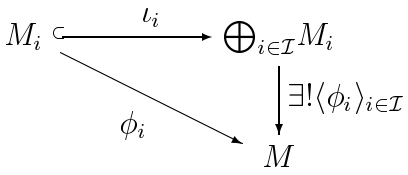


Fig. 2.1. The commutative diagram for Theorem 2.1.2.

$$\langle\phi_i\rangle_{i\in\mathcal{I}}(m) = \phi_{i_1}(m_{i_1}) \cdot \ldots \phi_{i_k}(m_{i_k})$$

When the family $\mathcal{I}$ is finite, say $\mathcal{I} = \{1, \ldots, n\}$, we denote the coproduct as $\bigoplus_{i=1}^{n} M_i$, and when the monoids M_i are identical (this implies that their operations and units are the same too) we denote it as $\bigoplus_{i\in\mathcal{I}} M$. Finally, when $\mathcal{I} = \{1, \ldots, n\}$ and $M = M_i$ for all $i \in \mathcal{I}$, we denote the coproduct as $\bigoplus_{i=1}^{n} M$.

2.1.2 Kleene algebras

We remind here one of the possible axiomatizations of Kleene algebras.

Definition 2.1.3. *A **Kleene algebra** is a structure $(A, +, \cdot, (\cdot)^*, 0, 1)$ which satisfies the following properties:*

1. *$\big(A, +, \cdot, 0, 1\big)$ is a semiring, that is:*
   - *$(A, +, 0)$ is an idempotent monoid;*
   - *$(A, \cdot, 1)$ is a monoid;*
   - *$\cdot$ distributes over $+$.*
2. *$(\cdot)^*$ satisfies the following equations [Con71, Koz94]:*

$$X \cdot Y \leq Y \Rightarrow X^* \cdot Y \leq Y \tag{2.5}$$
$$Y \cdot X \leq Y \Rightarrow Y \cdot X^* \leq Y \tag{2.6}$$
$$1 + X \cdot X^* \leq X^* \tag{2.7}$$
$$1 + X^* \cdot X \leq X^* \tag{2.8}$$

where $\leq$ is the partial order induced by the idempotent $+$ [Bir79], that is,

$$X \leq Y \text{ iff } X + Y = Y.$$

*A Kleene algebra is called **commutative** iff it satisfies the following identity [Con71]:*

$$X^* + Y^* = (X + Y)^* + (X^* \cup Y^*) \tag{2.9}$$

The classical example for Kleene algebras is the set of all languages over an alphabet Σ), $\big(\mathcal{P}(\Sigma^*), \cup, \cdot, (\cdot)^*, \emptyset, \{\varepsilon\}\big)$. An example of a commutative Kleene algebra is the Kleene algebra over a one-letter alphabet.

2.2 Signals

We denote $\mathbb{R}_{\leq 0}$, $\mathbb{R}_{\geq 0}$ and $\mathbb{R}_{>0}$ the sets of negative, nonnegative, resp. positive numbers, $\mathbb{Q}_{\geq 0}$ the set of nonnegative rational numbers, $\mathbb{Z}$ the set of integers and $\mathbb{N}$ the set of nonnegative integers (also called *naturals*). For each $n_1, n_2 \in \mathbb{Z}$, $[n_1 \dots n_2]$ denotes the *interval of integers* $\{n_1, n_1 + 1, \dots, n_2\}$, while $]2, 3]$ denotes the left-open, right-closed interval of reals whose limits are 2, resp. 3. $\mathbb{Q}Int$ denotes the set of intervals having bounds in $\mathbb{Q}_{\geq 0} \cup \{\infty\}$ and including the empty set, while $\mathbb{Z}Int$ denotes the set of intervals having bounds in $\mathbb{Z} \cup \{-\infty, \infty\}$ and including the empty set. An open interval is denoted as $]a, b[$, while a closed one is denoted as $[a, b]$. We will extensively use left-closed right-open intervals, which are thence denoted $[a, b[$.

For each function $f : \mathbb{R} \to A$, real number α and each $a \in A$, we say that *the left limit of f at α is a* and denote it $\lim_{t \nearrow \alpha} f = a$ iff the following property holds:

$$\text{there exists some } \varepsilon > 0 \text{ such that for all } t \in]\alpha - \varepsilon, \alpha[, f(t) = a$$

Right limits $\lim_{t \searrow \alpha} f = a$ can be defined similarly. This definition amounts to considering that the set A is equipped with the *discrete topology*.

A *left discontinuity* in f is some $\alpha \in \mathbb{R}$ for which $\lim_{t \nearrow \alpha} f(t) \neq f(\alpha) = \lim_{t \searrow \alpha} f(t)$. The discontinuity is *right* if we rather have that $\lim_{t \nearrow \alpha} f(t) = f(\alpha) \neq \lim_{t \searrow \alpha} f(t)$.

Definition 2.2.1. *A **signal** over a finite alphabet Σ is a function $\sigma : [0, \alpha[\to \Sigma$ where α is a nonnegative number, function which has finitely many discontinuities, all of them being left discontinuities.*

We denote $dom(\sigma)$ the domain of σ and $\ell(\sigma)$ its endpoint. $\ell(\sigma)$ is also called the *length* of σ.

Signals can be given graphical representation. For example, Figure 2.2 gives the graphical representation of the signal $\sigma_1 : [0, 6[\to \{a, b, c\}$ defined by:

$$\sigma_1(t) = \begin{cases} a & \text{iff } t \in [0, 1[\cup [\pi, 4.25[\\ b & \text{iff } t \in [1, \pi[\\ c & \text{iff } t \in [4.25, 6[\end{cases} \tag{2.10}$$

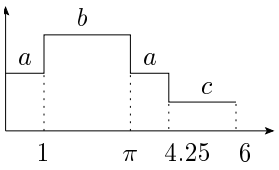


Fig. 2.2. A graphical representation of the signal defined in Identity 2.10.

$\mathsf{Sig}(\Sigma)$ denotes the set of signals over Σ. Note that there exists a unique signal with empty domain $\sigma_\epsilon : [0, 0[\to \Sigma$.

For $\sigma_1, \sigma_2 \in \mathsf{Sig}(\Sigma)$ with $dom(\sigma_i) = [0, e_i[$ $(i = 1, 2)$ define their *concatenation* $\sigma_1 \cdot \sigma_2 = \sigma$ as the signal with $dom(\sigma) = [0, e_1 + e_2[$ and such that

$$\sigma(t) = \begin{cases} \sigma_1(t) & \text{for } t \in [0, e_1[, \\ \sigma_2(t - e_1) & \text{for } t \in [e_1, e_1 + e_2[. \end{cases}$$

For example, the signal σ_1 in Figure 2.2 can be regarded as the concatenation of the two signals in Figure 2.3.

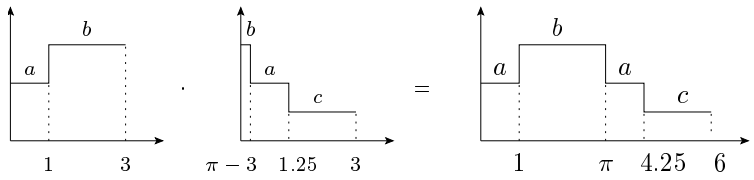


Fig. 2.3. An example of signal concatenation.

Proposition 2.2.2 ([ACM01]). *$(\mathsf{Sig}(\Sigma), \cdot, \sigma_\epsilon)$ is a a noncommutative monoid, called the **monoid of signals with concatenation**.*

Moreover $(\mathsf{Sig}(\Sigma), \cdot, \sigma_\epsilon)$ is isomorphic to the coproduct $\bigoplus_{a\in\Sigma} \mathbb{R}_{\geq 0}$ of $card(\Sigma)$ copies of the monoid of nonnegative reals $(\mathbb{R}_{\geq 0}, +, 0)$.

Proof. It is clear that the domain of each signal σ splits into finitely many intervals $[e_{i-1}, e_i[$ $(i \in [1 \dots n])$ on which σ is constant. Therefore we may identify σ with the formal concatenation:

$$\phi(\sigma) = a_1^{t_1} a_2^{t_2} \dots a_n^{t_n} \text{ where } t_i = e_i - e_{i-1} \text{ and } a_i = \sigma(t)\ \forall t \in [e_{i-1}, e_i[\tag{2.11}$$

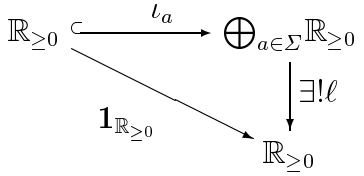
If we add $\phi(\sigma_\epsilon) = 0$ we obtain the isomorphism $\phi : \mathsf{Sig}(\Sigma) \to \bigoplus_{a\in\Sigma} \mathbb{R}_{\geq 0}$. □

As we can see, this proposition gives a more "friendly" presentation of signals [ACM97]: the signal presented in Figure 2.2 is represented also by the following element of $\bigoplus_{a\in\Sigma} \mathbb{R}_{\geq 0}$:

$$\sigma_1 = a^1 b^{\pi-1} a^{4.25-\pi} c^{1.75}$$

Observe also that the empty signal σ_ϵ is denoted by the empty sequence ε and that $a^0 = \varepsilon = \sigma_\epsilon$ for any $a \in \Sigma$. We will utilize both notations ε and σ_ϵ for the unit of concatenation in $\mathsf{Sig}(\Sigma)$.

Proposition 2.2.2 allows us to define the length of a signal as a monoid morphism induced by the coproduct property: denote first $\iota_a : \mathbb{R}_{\geq 0} \to \mathsf{Sig}(\Sigma)$ the coproduct inclusion for each copy of $\mathbb{R}_{\geq 0}$. Then ℓ is the unique morphism $\ell : \mathsf{Sig}(\Sigma) \to \mathbb{R}_{\geq 0}$ defined by the following diagram:



For each symbol $a \in \Sigma$ we denote $\mathsf{Sig}_a(\Sigma)$ the set of signals whose value is constantly a:

$$\mathsf{Sig}_a(\Sigma) := \iota_a(\mathbb{R}_{\geq 0}) = \{a^t \mid t \in \mathbb{R}_{\geq 0}\}$$

2.2.1 Timed languages: basic properties

For any set X, the set of subsets of X is denoted as $\mathcal{P}(X)$.

Subsets of $\mathsf{Sig}(\Sigma)$ are called **timed languages**. Signal concatenation can be naturally extended to timed languages:

$$L_1 \cdot L_2 = \{\sigma_1\sigma_2 \mid \sigma_1 \in L_1, \sigma_2 \in L_2\}$$

and gives rise to star:

$$L^* = \bigcup_{n\in\mathbb{N}} L^n$$

where $L^0 = \{\sigma_\epsilon\}$ and $L^{n+1} = L^n \cdot L$.

Proposition 2.2.3. $\big(\mathcal{P}(\mathsf{Sig}(\Sigma)), \cup, \cdot, (\cdot)^*, \emptyset, \{\sigma_\epsilon\}\big)$ *is a Kleene algebra.*

Proof. All properties follow by transporting the proof that $\big(\mathcal{P}(\Sigma^*), \cup, \cdot(\cdot)^*, \emptyset, \{\varepsilon\}\big)$ is a Kleene algebra [Sal66, Koz94]. For example, the implication 2.5 follows by proving, by induction on $n \in \mathbb{N}$, that $X \cdot Y \subseteq Y$ implies $X^n \cdot Y \subseteq Y$. □

2.3 Timed words

Given a set of symbols Σ^* and a word $w \in \Sigma^*$, the *length* of w, denoted $|w|$, is the number of symbols in w. It can be regarded as the unique morphism $|\cdot| : (\Sigma^*, \cdot, \varepsilon) \to (\mathbb{N}, +, 0)$ determined by $|a| = 1$ for all $a \in \Sigma$. Remind that $(\Sigma^*, \cdot, \varepsilon)$ is the free monoid generated by Σ.

Definition 2.3.1. *A **timed word** over the alphabet Σ is a pair $\omega = (w, \tau)$ consisting of a word $w \in \Sigma^*$, and a function $\tau : [1 \ldots |w| + 1] \to \mathbb{R}_{\geq 0}$.*

The word w is called the *sequence of actions* in ω and the function τ is called the *sequence of time labels* in ω. The *length* of a timed word is simply the length of the sequence of actions in it, and is denoted $|(w, \tau)|$.

We denote $\mathsf{TW}(\Sigma)$ the set of timed words over Σ. Subsets of it are called *timed word languages*.

On $\mathsf{TW}(\Sigma)$ we can define a *concatenation* operation as an extension of the concatenation on words: given two timed words (w_1, τ_1) and (w_2, τ_2) define $(w_1, \tau_1) \cdot (w_2, \tau_2) = (w_1w_2, \tau)$ where

$$\tau(i) = \begin{cases} \tau_1(i) & \text{iff } i \in [1 \ldots |w_1|] \\ \tau_1(|w_1| + 1) + \tau_2(1) & \text{iff } i = w_1 + 1 \\ \tau_2(i - |w_1|) & \text{iff } |w_1| + 2 \leq i \leq |w_1| + |w_2| + 1 \end{cases}$$

Note again that there exists a unique timed word $\overline{\epsilon} = (\varepsilon, \mathbf{0})$ whose sequence of time labels is the function which maps the unique element in its domain to 0.

Proposition 2.3.2. *$(\mathsf{TW}(\Sigma), \cdot, \overline{\epsilon})$ is a noncommutative monoid.*

Moreover $(\mathsf{TW}(\Sigma), \cdot, \overline{\epsilon})$ is isomorphic to the coproduct monoid $\mathbb{R}_{\geq 0} \oplus \Sigma^*$, where $(\mathbb{R}_{\geq 0}, +, 0)$ is the monoid of nonnegative reals.

Proof. The isomorphism is simply the "rearrangement" of each timed word as follows:

$$\phi\big((w,\tau)\big) = \tau(1) \cdot w_1 \cdot \ldots \cdot w_{|w|} \cdot \tau(|w|+1) \qquad \square$$

It follows that $|\cdot| : \big(\mathsf{TW}(\Sigma), \cdot, \overline{\varepsilon}\big) \to (N, +, 0)$ is a monoid morphism.

2.3.1 Relating the monoids of signals and of timed words

For the sequel we fix two alphabets Σ, Ω. We call a monoid morphism $\phi : \mathsf{Sig}(\Sigma) \to \mathsf{TW}(\Omega)$ as *trivial* if for each signal $\sigma \in \mathsf{Sig}(\Sigma)$

$$\phi(\sigma) = \big(\varepsilon, \alpha\big) \text{ for some } \alpha \in \mathbb{R}_{\geq 0}.$$

Observe that a morphism is trivial iff $|\phi(\sigma)| = 0$ for all $\sigma \in \mathsf{Sig}(\Sigma)$.

Proposition 2.3.3. *Any monoid morphism from $\mathsf{Sig}(\Sigma)$ to $\mathsf{TW}(\Omega)$ is a trivial morphism in the above sense.*

Proof. Assume $\phi : \mathsf{Sig}(\Sigma) \to \mathsf{TW}(\Omega)$ is a monoid morphism. We first prove that for each $a \in \Sigma$, $\phi\big|_{Sig_a(\Sigma)}$ is a trivial morphism. Then the result would follows since each signal is a concatenation of constant signals.

Observe first that, for each two nonnegative reals $\alpha, \beta \in \mathbb{R}_{\geq 0}$,

$$\text{if } \alpha \leq \beta \text{ then } |\phi(a^\alpha)| \leq |\phi(a^\beta)| \tag{2.12}$$

Here we have used the fact that the length of a timed word $|\cdot|$ is a monoid morphism. This comes as $a^\alpha \cdot a^{\beta-\alpha} = a^\beta$ and hence, if we apply the composition of morphisms $\psi = |\cdot| \circ \phi$ to a^β we get

$$|\phi(a^\beta)| = |\phi(a^\alpha)| + |\phi(a^{\beta-\alpha})| \geq |\phi(a^\alpha)|$$

This implies that there is a countable partition $(\mathsf{Sig}_a^n)_{n\in\mathbb{N}}$ of $Sig_a(\Sigma)$ such that $\mathsf{Sig}_a^n(\Sigma) =$ the set of constant a-signals which are mapped by the morphism $|\cdot| \circ \phi$ to the integer n:

$$\mathsf{Sig}_a^n(\Sigma) = \{a^\alpha \mid |\phi(a^\alpha)| = n\}$$

Let us denote n_0 the *first* integer with the property that $\mathsf{Sig}_a^{n_0}(\Sigma)$ has a nonempty interior. Such a number must exist since, for each $n \in \mathbb{N}$, if there exist some $\alpha_1, \alpha_2 \in \mathsf{Sig}_a^n(\Sigma)$ with $\alpha_1 < \alpha_2$ then $[\alpha_1, \alpha_2] \subseteq \mathsf{Sig}_a^n(\Sigma)$ due to implication 2.12. As a consequence, $\mathsf{Sig}_a^{n_0}(\Sigma) \supseteq]0, \alpha[$ for some $\alpha \in \mathbb{R}_{\geq 0}$.

But then, for each $\beta \in]0, \alpha[$, $\beta/2 \in]0, \alpha[$ too and hence:

$$n_0 = |\phi(a^\beta)| = |\phi(a^{\beta/2}a^{\beta/2})| = |\phi(a^{\beta/2} \cdot \phi(a^{\beta/2})| = |\phi(a^{\beta/2})| + |\phi(a^{\beta/2})| = n_0 + n_0$$

It follows that $n_0 = 0$.

But then, for each $\beta \in \mathbb{R}_{\geq 0}$, if we denote $k = \lceil \frac{\beta}{\alpha} \rceil$ then $\frac{\beta}{k} < \alpha$ and hence $|\phi(\frac{\beta}{k})| = 0$. It follows that

$$|\phi(a^\beta)| = |\phi(a^{\frac{\beta}{k} \cdot k})| = k|\phi(a^{\frac{\beta}{k}})| = 0$$

This proves our claim that $|\cdot| \circ \phi$ maps all signals from $Sig_a(\Sigma)$ to 0. But every signal $\sigma \in \mathsf{Sig}(\Sigma)$ is a concatenation of constant signals $\sigma = a_1^{t_1} \cdot \ldots \cdot a_p^{t_p}$ where $a_i \in \Sigma$. Hence we have that for all $\sigma \in \mathsf{Sig}(\Sigma)$

$$|\phi(\sigma)| = |\phi(a_1^{t_1}) \cdot \ldots \cdot \phi(a_p^{t_p})| = 0 \qquad \square$$

Remark 2.3.4. In general, trivial morphisms from $\mathsf{Sig}(\Sigma)$ to $\mathsf{TW}(\Omega)$ are linear on each submonoid $\mathsf{Sig}(\{a_i\})$ of $\mathsf{Sig}(\Sigma)$, but the slopes might be different. Hence we might add to the above proof the observation that, on each $\mathsf{Sig}(\{a_i\})$ there exists some $K \in \mathbb{R}_{\geq 0}$ such that $\phi(a^t) = Kt$. Consequently, we may conclude that all morphisms from $\mathsf{Sig}(\Sigma)$ to $\mathsf{TW}(\Omega)$ are "piecewise-linear", that is, there exist $K_1, \ldots, K_m$ such that

$$\phi(a_1^{t_1} \ldots a_m^{t_m}) = K_1 t_1 + \ldots K_m t_m$$

where $m = card(\Sigma)$.

We have another option for relating signals and timed words: to join the two structures into a single one, i.e. define signals with actions or timed words with states:

Definition 2.3.5. *The monoid of **signals with actions** over the set of states Ω and the set of actions Σ is the coproduct $\bigoplus_{a \in \Omega} \mathbb{R}_{\geq 0} \oplus \Sigma^*$ of $card(\Omega)$ copies of the monoid of nonnegative reals $(\mathbb{R}_{\geq 0}, +, 0)$ with the free monoid $(\Sigma^*, \cdot, \varepsilon)$.*

Hence now the monoids of signals and of timed words can be regarded as "particularizations" (or *projections*, in the algebraic sense) of the monoid of signals with actions. However we will not utilize this notion since it will make all derived notions and proofs unnecessarily more complicated.

2.4 Timed regular languages defined by inverse monoid morphisms

We start this section by reminding the way regular languages are related to monoid morphisms [Eil74].

Definition 2.4.1. *Given $(M, \cdot, e)$ a (possibly infinite) monoid, an M-**automaton** is a tuple $\mathcal{A} = (Q, M, \delta, q_0, Q_f)$ where $q_0 \in Q$, $Q_f \subseteq Q$ and $\delta : Q \times M \to Q$ has the property that for all $q \in Q$, $m_1, m_2 \in M$:*

$$\delta(q, m_1 \cdot m_2) = \delta(\delta(q, m_1), m_2) \quad \textit{and} \quad \delta(q, e) = q \tag{2.13}$$

$\mathcal{A}$ is called **finite** whenever Q is finite. The **accepted language** of $\mathcal{A}$ is $L(\mathcal{A}) = \{m \in M \mid \delta(q_0, m) \in Q_f\}$. It is also assumed that all states $q \in Q$ in an M-automaton are *accessible*, i.e. there exists some $m \in M$ such that $\delta(q_0, m) = q$.

Definition 2.4.2. *Given a monoid M, the family of M-**regular languages**, denoted $\mathsf{Reg}(M)$, is the family of subsets $L \subseteq M$ which are the accepted language of some* finite *M-automaton.*

Theorem 2.4.3 ([Eil74]). *$\mathsf{Reg}(M)$ coincides with the family of subsets $L \subseteq M$ for which there exists some finite monoid $(F, \circ, e')$, some surjective monoid morphism $\phi : M \to F$ and some subset $F' \subseteq F$ such that $L = \phi^{-1}(F')$.*

2.4.1 Essentially untimed regular languages

The **untiming** of a signal is the sequence of symbols that appear in it. Observe that in the untiming of a signal, two consecutive symbols are distinct. Hence the untiming application (actually, morphism) is not surjective.

We will take advantage of the algebraic definition of $\mathsf{Sig}(\Sigma)$ and define the untiming as a coproduct of morphisms. We define first the monoid $SF(\Sigma)$ of **stuttering-free words**, that is, the set of words in which no two consecutive letters are equal. This monoid is endowed with a concatenation operation that "fuses" two identical letters. For example,

$$aba \cdot ac = abac$$

We may define $SF(\Sigma)$ in two ways: as a coproduct monoid and as a quotient monoid:

The *coproduct* definition is the following: for each $a \in \Sigma$ consider the monoid $(S_a = \{\varepsilon, a\}, \cdot, \varepsilon)$ where $a \cdot a = a$ (i.e. a is idempotent). Define then $SF(\Sigma^*)$ as the coproduct of the family monoids $(S_a, \cdot, \varepsilon)_{a \in \Sigma}$,

$$SF(\Sigma) = \bigoplus_{a \in \Sigma} S_a$$

and denote $\kappa_a : S_a \hookrightarrow SF(\Sigma)$ the inclusion morphism which defines this coproduct.

The *quotient* presentation of $SF(\Sigma)$ is the following: consider the relation $\rho \subseteq \Sigma \times \Sigma$ defined by

$$\rho = \{(aa, a) \mid a \in \Sigma\}$$

This relation can be uniquely extended to a congruence on Σ^* as follows:

$$\tilde{\rho} = \{(w_1 aaw_2, w_1 aw_2), (w_1 aw_2, w_1 aaw_2) \mid w_1, w_2 \in \Sigma^*, a \in \Sigma\}$$

Then $SF(\Sigma)$ is isomorphic to the quotient of the Σ^* by the congruence $\tilde{\rho}$. That is, elements of $SF(\Sigma)$ can be thought as equivalence classes w.r.t. $\tilde{\rho}$. We denote $\overbrace{\cdot}$: $\Sigma^* \to \Sigma^*/\tilde{\rho}$ the canonical projection. For example, $\overbrace{aabcbba} = abcba$.

Concatenation of stuttering-free words, i.e. the *internal* operation on $SF(\Sigma)$ that makes it a monoid, can then be defined as follows: for each $w, w' \in SF(\Sigma)$, w not ending in a and w' not beginning in b,

$$wa \cdot bw' = \begin{cases} wabw' & \text{iff } a \neq b \\ waw' & \text{iff } a = b \end{cases}$$

We may now proceed to the algebraic definition of untiming: consider the following monoid morphism:

$$\rho_a : (\mathbb{R}_{\geq 0}, +, 0) \to (S_a, \cdot, \varepsilon), \quad \rho_a(t) = \begin{cases} a & \text{iff } t \neq 0 \\ \varepsilon & \text{iff } t = 0 \end{cases}$$

The following diagram defines, by theorem 2.1.2, the **untiming morphism** $\mathcal{U}$:

$$\begin{array}{ccc} \mathbb{R}_{\geq 0} & \xrightarrow{\iota_a} & \mathsf{Sig}(\Sigma) = \bigoplus_{a\in\Sigma} \mathbb{R}_{\geq 0} \\ \Big\downarrow{\scriptstyle \rho_a} & & \Big\downarrow{\scriptstyle \exists! \mathcal{U} = \langle \kappa_a \circ \rho_a \rangle_{a\in\Sigma}} \\ S_a & \xrightarrow{\kappa_a} & SF(\Sigma) \end{array}$$

Definition 2.4.4. *A timed language $L \subseteq \mathsf{Sig}(\Sigma)$ is called **essentially untimed** iff there exists some set $L' \subseteq SF(\Sigma)$ (i.e.* a language in the classical sense*) such that $L = \mathcal{U}^{-1}(L')$. The class of **essentially untimed regular languages** consists of essentially untimed languages that are inverse images of regular languages in $SF(\Sigma)$.*

Observe that $SF(\Sigma)$-regular languages in the sense of definition 2.4.2 are in fact regular languages in the classical sense [HU92] with the restriction that in each word no two consecutive letters are equal – and this restriction means intersection with a regular set.

2.4.2 Syntactic monoids on $(\mathsf{Sig}(\Sigma), \cdot, \sigma_\epsilon)$ are not interesting

In this section we show that $\mathsf{Sig}(\Sigma)$-regular languages are essentially untimed. Remind first that a **zero element** in a monoid $(M, \cdot, e)$ is an element $\uparrow \in M$ which satisfies

$$\forall x \in M,\ \uparrow \cdot x = x \cdot \uparrow = \uparrow$$

Our result relies on the following lemma:

Lemma 2.4.5. *Suppose $f : (\mathbb{R}_{\geq 0}, +, 0) \to (M, \cdot, e)$ is a surjective monoid morphism and $(M, \cdot, e)$ is a finite monoid. Then*

- *either f is the trivial morphism $f(x) = e$, $\forall x \in \mathbb{R}_{\geq 0}$ (and hence $M = \{e\}$);*
- *or $M = \{e, \uparrow\}$ where $\uparrow$ is a zero element and f maps any $x \neq 0$ to $\uparrow$ and 0 to e.*

Proof. We note first that the surjectivity of f implies that $\cdot$ is commutative. We may also prove by induction on k that for all $x \in \mathbb{R}_{\geq 0}$ and for all $k \in \mathbb{N}$, $\quad f(k \cdot x) = f(x)^k$.

Let's prove first the following:

Claim. If M contains only idempotents then it has at most two elements, and one of these elements is a zero element (in fact *the* zero element, since it is unique).

Proof. Suppose $M = \{m_1, \ldots, m_n\}$. Then $m = m_1 \cdot \ldots \cdot m_n$ is the zero element because

$$\begin{aligned} m \cdot m_i &= m_1 \cdot \ldots \cdot m_n \cdot m_i = m_1 \cdot \ldots \cdot m_i \cdot m_i \cdot \ldots \cdot m_n \\ &= m_1 \cdot \ldots \cdot m_i \cdot \ldots \cdot m_n = m \end{aligned}$$

where we have used the commutativity of $\cdot$.

Two cases arise then: the first is when $m = e$, the unit of M. But the unit can be a zero element only if the monoid has only one element, the unit:

$$e = e \cdot m_i \text{ (as } e \text{ is the zero element)} = m_i \text{ (as } e \text{ is the unit)}$$

The second case is $m \neq e$. In this case there must exist $\beta \neq 0$ such that $f(\beta) = m$.

Take then any $\alpha > 0$. Since $f(\alpha)$ is idempotent we have that $f(\alpha) = f(k \cdot \alpha)$ for any $k \in \mathbb{N}_{\geq 1}$. If we take then $k \geq \lceil \frac{\beta}{\alpha} \rceil$ we obtain $k\alpha \geq \beta$ and hence $k\alpha - \beta \geq 0$, hence $f(k\alpha - \beta)$ is defined.

But then, as $m = f(\beta)$ is the zero element we have that $f(\beta) = f(\beta) \cdot f(k\alpha - \beta)$. Therefore

$$m = f(\beta) = f(\beta) \cdot f(k\alpha - \beta) = f(\beta + k\alpha - \beta) = f(k\alpha) = f(\alpha)$$

which proves that any positive real α is mapped to the zero element. □

So what is left to prove is that if f is surjective then M contains only idempotents. Take a positive real $\alpha \geq 0$ and denote $G(\alpha)$ the image under f of the submonoid $\alpha\mathbb{N}$ of *multiples* of α:

$$G(\alpha) = \big\{ f(n\alpha) \mid n \in \mathbb{N} \big\}$$

It is clear that $G(2^p\alpha) \subseteq G(2^{p+1}\alpha)$ for any $p \in \mathbb{Z}$. But since M is finite, there must exist some $p_0 \in \mathbb{Z}$ such that $G(2^{p_0}\alpha) = G(2^{p_0+1}\alpha)$. Let us then denote $\beta = \min(\alpha, 2^{p_0}\alpha)$, hence $G(\beta) = G\Big(\frac{\beta}{2}\Big)$.

We may prove that $G(\beta)$ is *cyclic*, that is, if we denote $k = card G(\beta)$ then

$$G(\beta) = \big\{ f(i\beta) \mid i \in [1 \ldots k] \big\} = \big\{ f(0), f(\beta), \ldots, f(k\beta) \big\} \tag{2.14}$$

$$f((k+1)\beta) = f(\beta) \tag{2.15}$$

To this end, note that if there exist $i \in [1 \ldots k]$, $j \in \mathbb{N}$ with $i \neq j$ and such that $f(i\beta) = f(j\beta)$ then we would have that $f((i+n)\beta) = f((j+n)\beta)$ for all $n \in \mathbb{N}$. But this would imply that

$$G(\beta) = \big\{ f(0), \ldots, f((j-1)\beta) \big\}$$

and hence $cardG(\beta) \leq j-1$, which implies that $j \geq k+1$. This shows that Identity 2.14 holds.

For showing that Identity 2.15 holds too, suppose that $f((k+1)\beta) = f(i\beta)$ for some $i \in [1 \dots k]$. By a simple induction argument, we may prove that, for all $n \in \mathbb{N}$,

$$f\big((k+1+n(k+1-i))\beta\big) = f\big((i+n(k+1-i))\beta\big) \tag{2.16}$$

On the other hand, since $f\Big(\frac{\beta}{2}\Big) \in G(\beta)$, we must have some $j \in [0 \dots k]$ such that $f\Big(\frac{\beta}{2}\Big) = f(j\beta)$ too. Therefore,

$$f(\beta) = f\Big(\frac{\beta}{2}\Big) \cdot f\Big(\frac{\beta}{2}\Big) = f(j\beta) \cdot f(j\beta) = f(2j\beta)$$

It follows that $2j \geq k+1$. By recursively appying the Identity 2.16 we further get that:

$$\begin{aligned} f(2j\beta) = f\big((2j-(k+1-i))\beta\big) = f\big((2j-2(k+1-i))\beta\big) = \dots \\ = f\bigg(\Big(2j-\Big(\Big\lfloor\frac{2j-k}{k+1-i}\Big\rfloor+1\Big)(k+1-i)\Big)\beta\bigg) \end{aligned} \tag{2.17}$$

and $2j-\Big(\Big\lfloor\frac{2j-k}{k+1-i}\Big\rfloor+1\Big)(k+1-i) \in [i \dots k]$.

But this rewrites to the fact that $f(\beta) = f(l\beta)$ for some $l \in [i \dots k]$, which would be impossible, by Identity 2.14, unless $i = l = 1$. Hence Identity 2.15 holds too.

But then, starting from Identity 2.17 and replacing i with 1 we may further conclude that

$$f(\beta) = f(2j\beta) = f\bigg(\Big(2j-\Big(\Big\lfloor\frac{2j-k}{k}\Big\rfloor+1\Big)\cdot k\Big)\beta\bigg) = f\bigg(\Big(2j-\Big\lfloor\frac{2j}{k}\Big\rfloor\cdot k\Big)\beta\bigg)$$

with $2j-\Big\lfloor\frac{2j}{k}\Big\rfloor k \in [1 \dots k]$. This also implies, by Identity 2.14, that

$$2j-\Big\lfloor\frac{2j}{k}\Big\rfloor k = 1$$

As a consequence, $k \mid 2j-1$, fact which, corroborated with the hypothesis that $j \in [1 \dots k]$, implies that $k = 2j-1$.

Therefore,

$$f\Big(\frac{\beta}{2}\Big) = f((k+1)\beta) = f(\beta) \text{ (by Identity 2.15)}$$

or, by multiplying by 2,

$$f(\beta) = f(2\beta) = f(\beta)^2.$$

Hence $f(\beta)$ is an idempotent. By an easy induction we may show then that $f(2^p\beta)$ is an idempotent too, which implies that $f(\alpha) = f(2^{p_0}\beta)$ is an idempotent. □

Observe that it was essential to utilize the fact that $f\left(\frac{\beta}{2}\right) \in G(\beta)$ to show that $G(\beta)$ is cyclic.

The difference between this lemma and Proposition 2.3.3 is that the monoid of natural numbers is an ordered monoid, in which the order is compatible with the monoidal structure. On the contrary, in a finite monoid there is no compatible ordering for free. This is why the two proofs are different, with the proof of Lemma 2.4.5 much more involved than the proof of Proposition 2.3.3.

Theorem 2.4.6. *The family of* $\mathsf{Sig}(\Sigma)$*-regular languages equals the family of essentially untimed regular languages.*

Proof. This is a corollary of theorem 2.4.3 and the above lemma. The right-to-left inclusion follows by an easy argument: if K is a regular language in $SF(\Sigma)$, then we have some surjective morphism $\phi : SF(\Sigma) \to M$ where $(M, \cdot, e)$ is a finite monoid, and some subset $F \subseteq M$ such that $K = \phi^{-1}(F)$. But then $\phi \circ \mathcal{U} : \mathsf{Sig}(\Sigma) \to M$ is a surjective monoid morphism too and defining $L = \mathcal{U}^{-1}(K) = (\phi \circ \mathcal{U})^{-1}(F)$ we get that L is a regular language in $(\mathsf{Sig}(\Sigma), \cdot, \sigma_\epsilon)$.

For the reverse inclusion, suppose we have some finite monoid $(M, \cdot, e)$ and some surjective morphism $\phi : \mathsf{Sig}(\Sigma) \to M$. We look for a decomposition of ϕ in which the untimed morphism $\mathcal{U}$ occurs.

Remember that, for any $a \in \Sigma$, $\iota_a : \mathbb{R}_{\geq 0} \to \mathsf{Sig}(\Sigma)$ is the inclusion morphism for the coproduct property. It follows that $\phi \circ \iota_a : \mathbb{R}_{\geq 0} \to M$ is a monoid morphism. This implies, by the above lemma, that the image of this morphism, $Im(\phi \circ \iota_a)$, is a submonoid of M having at most two elements, $M = \{e, \uparrow\}$ with $\uparrow \cdot \uparrow = \uparrow$.

But the monoid $(S_a = \{\varepsilon, a\}, \cdot, \varepsilon)$ which was used to define $\mathcal{U}$ on page 27 is isomorphic to $Im(\phi \circ \iota_a)$, since $aa = a$ in S_a. Define then the morphism $j_a : S_a \to M$ as

$$j_a(\varepsilon) = e \text{ and } j_a(a) = \phi(\iota_a(1)) = \phi(\iota_a(t)) \text{ for all } t \neq 0$$

That is, j_a is either the isomorphism from S_a to M or the trivial morphism.

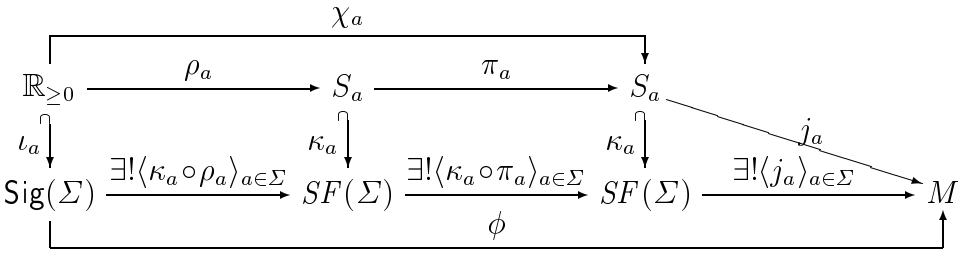
Define also the morphisms $\chi_a : \mathbb{R}_{\geq 0} \to S_a$ and $\pi_a : S_a \to S_a$ as follows:

$$\chi_a(0) = \varepsilon \text{ and for all } \alpha \neq 0, \quad \chi_a(t) = \begin{cases} a & \text{iff } \phi(\iota_a(t)) \neq e \\ \varepsilon & \text{otherwise} \end{cases}$$

$$\pi_a(\varepsilon) = \varepsilon \qquad \pi_a(a) = \chi_a(1) = \chi_a(t) \text{ for any } t \neq 0$$

Then $\phi \circ \iota_a = j_a \circ \chi_a$ and $\chi_a = \pi_a \circ \rho_a$.

Now we are ready to chase the following diagram, in which all the squares and triangles commute:



The upper triangle and the outer square commute as shown above. The inner squares and the right triangle commute by just the coproduct property. The outcome of this diagram is the commutativity of the bottom square, i.e.

$$\phi = \langle j_a \rangle_{a\in\Sigma} \circ \langle \kappa_a \circ \pi_a \rangle_{a\in\Sigma} \circ \langle \kappa_a \circ \rho_a \rangle_{a\in\Sigma} = \langle j_a \rangle_{a\in\Sigma} \circ \langle \kappa_a \circ \pi_a \rangle_{a\in\Sigma} \circ \mathcal{U} \tag{2.18}$$

since $\mathcal{U} = \langle \kappa_a \circ \rho_a \rangle_{a\in\Sigma}$.

This commutativity follows by the coproduct property: both the left-hand side and the right-hand side, when composed with ι_a, give $j_a \circ \chi_a$. Therefore, both must be equal to the unique coproduct $\langle j_a \circ \chi_a \rangle_{a\in\Sigma}$. Hence we have managed to show that the untiming morphism $\mathcal{U} = \langle \kappa_a \circ \rho_a \rangle_{a\in\Sigma}$ occurs in some decomposition of ϕ.

So suppose we have some $\mathsf{Sig}(\Sigma)$-regular language L, witnessed by the subset $F \subseteq M$, i.e. $L = \phi^{-1}(F)$. Consider then the $SF(\Sigma)$-regular language $K = \left(\langle j_a \rangle_{a\in\Sigma} \circ \langle \kappa_a \circ \pi_a \rangle_{a\in\Sigma}\right)^{-1}(F)$. Then, using the decomposition 2.18 we get

$$L = \phi^{-1}(F) = \left(\langle j_a \rangle_{a\in\Sigma} \circ \langle \kappa_a \circ \pi_a \rangle_{a\in\Sigma} \circ \mathcal{U}\right)^{-1}(F) \;=\; \mathcal{U}^{-1}(K)$$

Hence L is indeed essentially untimed. □

3. Real-time automata

In this chapter we study a class of automata which seems to be the largest extension from finite automata still carrying the decidability of both the emptiness and the universality problems, a Pumping Lemma and, moreover, a Kleene theorem in which the semantics of the associated regular expressions is based upon a *total concatenation operation*. The automata we study, called Real-Time Automata (RTA), can be regarded as timed automata with a single clock which is reset at each transition, and they appeared (in a slightly different version) in connection with the so-called Simple Duration Calculus [HJ96].

However, even at this lowest level of introduction of timing constraints into finite automata we find that complementation raises a specific problem: the subset construction can be adapted to handle timing constraints, but it works only if the automata are *stuttering-free*, i.e., two states labeled with the same symbol are not connected by any transition. We also find out that *language determinism*, i.e., the property that each signal is associated with a unique run that starts in an initial state, cannot be captured by local properties like state-determinism or stuttering-freeness: stuttering-free state-deterministic RTA are less expressive than RTA.

We solve this problem by introducing the Kleene algebra of sets of real numbers. The rôle of concatenation from Kleene algebras of languages is taken here by addition of sets of real numbers. This operation models the process of removing one stuttering transition by "fusing" the adjacent states. We then study the sub-Kleene algebra generated by finite unions of intervals with rational bounds and prove a normal form theorem for this subalgebra, result which is based on properties of integer division and roughly says that elements in this subalgebra are "ultimately periodic". This result is not a corollary of the normal form for regular languages over a one letter alphabet because our Kleene algebra has two generators whose generated subalgebras are not disjoint but which cannot generate one another.

We also show here that the class of languages recognizable by real-time automata can be given an "algebraic" characterization, that is, can be presented as inverse morphic images of subsets in finite monoids, but *the monoid of signals needs to be redefined*: the concatenation has to be a "stuttering-free" concatenation, allowing two signals to concatenate only if they produce a discontinuity at the concatenation point.

The chapter runs as follows: in the next section we remind what RTA are, their associated regular expressions and the problem raised by their complementation. In the second section we introduce the Kleene algebra of sets of nonnegative reals and prove the normal form theorem. The third section contains the constructions that accomplish stuttering elimination and determinization

and the fourth is reserved for a pumping lemma and expressiveness issues concerning real-time automata. Finally, the fifth section is concerned with the "algebraic" presentation of the class of languages accepted by real-time automata.

This chapter contains the results from [Dim00a, Dim01b]

3.1 Real-time automata and their regular expressions

Real-time automata are state-labeled timed automata [ACM97] with a single clock which is reset at each transitions.

3.1.1 Real-time automata defined

Definition 3.1.1. *A **real-time automaton** (RTA) is a tuple $\mathcal{A} = (Q, \Sigma, \lambda, \iota, \delta, Q_0, Q_f)$ where Q is the (finite) set of* states*, Σ is the (finite nonempty)* alphabet*, $\delta \subseteq Q \times Q$ is the* transition relation*, $Q_0, Q_f \subseteq Q$ are the sets of* initial*, resp.* final *states, $\lambda : Q \to \Sigma$ is the* state labeling function *and $\iota : Q \to \mathbb{Q}Int$ is the* time labeling function*.*

We also call the pair $\big(\lambda(q), \iota(q)\big)$ the **label** of the state q.

RTA work over signals: a **run** of length $n \geq 1$ is a sequence of states $(q_i)_{i\in[1\ldots n]}$ connected by δ, i.e., $(q_i, q_{i+1}) \in \delta, \forall i \in [1 \ldots n-1]$. The run is **associated with** a signal σ iff there exists a decomposition

$$\sigma = \lambda(q_1)^{t_1} \cdot \ldots \cdot \lambda(q_n)^{t_n}$$

such that $t_i \in \iota(q_i)$ for all $i \in [1 \ldots n]$. Or, in other words, iff there exist some sequence of "splitting" points $0 = e_1 \leq \ldots \leq e_{n+1} = \ell(\sigma)$ such that $e_{i+1} - e_i \in \iota(q_i)$ and $\sigma(t) = \lambda(q_i)$ for all $t \in [e_i, e_{i+1}[$ and all $i \in [1 \ldots n]$. Observe that the "splitting" points must contain all the discontinuities in the signal but this inclusion might be strict, case in which we say that the run is *stuttering*.

A run is **accepting** if it starts in Q_0 and ends in Q_f. When a signal σ is associated with some accepting run we say that σ is **accepted** by $\mathcal{A}$. The **language** of $\mathcal{A}$ is the set of signals associated with some accepting run of $\mathcal{A}$ and is denoted $L(\mathcal{A})$. Two RTA are **equivalent** if they have the same language. If we denote the class of all RTA whose alphabet is Σ as RTA(Σ), then we may define the class of **real-time recognizable languages** over Σ as

$$TRec(\Sigma) = \{L \in \mathsf{Sig}(\Sigma) \mid \exists \mathcal{A} \in \text{RTA}(\Sigma) \text{ s.t. } L(\mathcal{A}) = L\}$$

As an example, the automaton in Figure 3.1 accepts the signal $\sigma = a^{2.5}b^{6.5}$ The accepting run which is associated with this signal is (q, r, s, t) and the splitting points are $e_1 = 0$, $e_2 = 2.5$, $e_3 = e_4 = 6.5$ and $e_5 = 9$. Note that the run is stuttering.

A real-time automaton whose alphabet is Σ is, from a syntactic point of view, a "finite presentation" of a classical automaton over the (uncountable) set of symbols $\Sigma \times \mathbb{R}_{\geq 0}$, where a state q

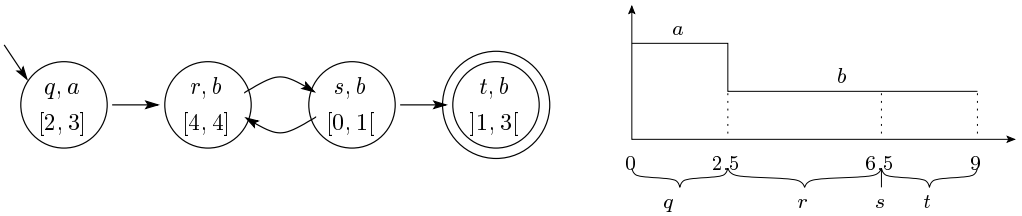


Fig. 3.1. An example of a real-time automaton and a signal accepted by it.

labeled (a, I) embodies a whole family of states labeled with (a, α) for all $\alpha \in I$. However the comparison stops at this syntactic level since semantically $\Sigma \times \mathbb{R}_{\geq 0}$ comes with a structure which is unavailable for the alphabet of a classical automaton: according to this structure, we may "fuse" two symbols sharing the same state label. It is this structure that allows the acceptance of the signal σ in the above example by splitting the symbol $b^{6.5}$ into three symbols b^4, b^0 and $b^{2.5}$.

We end this subsection with the following adaptation of the decidability of the emptiness problem for finite automata.

Proposition 3.1.2. *The emptiness problem for RTA is decidable.*

The proof relies on the algorithm for computing the sets of accessible states and then checking whether a final state is accessible, which can be done in linear time w.r.t. $card(Q) \cdot card(\delta)$.

Real-time automata can also be defined such that their accepted language would consist of timed words instead of signals. Most notably stuttering steps would translate into epsilon-transitions in the timed words setting. The whole contents of this chapter can then be translated to such automata without very much difficulty.

3.1.2 Regular expressions and the Kleene theorem

We have observed that labels in RTA are finite presentations of sets of symbols from $\Sigma \times \mathbb{R}_{\geq 0}$. This observation can be further extended to considering regular expressions over $\Sigma_{\mathbb{Q}Int} := \Sigma \times \mathbb{Q}Int$ with the aim of obtaining a Kleene theorem:

Definition 3.1.3. *Consider* $\mathsf{Rat}(\Sigma_{\mathbb{Q}Int})$*, the set of rational (or regular) expressions over* $\Sigma_{\mathbb{Q}Int}$*, i.e., defined by the following grammar*

$$E ::= 0 \mid \mathbf{1} \mid a_I \mid E + E \mid E \cdot E \mid E^*$$

where the atoms a_I *are symbols from* $\Sigma_{\mathbb{Q}Int}$*.*

Rational expressions in $\mathsf{Rat}(\Sigma_{\mathbb{Q}Int})$ *will be mainly called* ***real-time regular expressions****.*

There are two types of semantics for real-time regular expressions: the first one, called henceforth **abstract**, is the classic semantics in terms of words over the set of symbols $\Sigma_{\mathbb{Q}Int}$ and is denoted $|\cdot|$. For this semantics, $|0|$ is the empty set and $|\mathbf{1}|$ is the set containing the empty word over $\Sigma_{\mathbb{Q}Int}$, word which is denoted $\mathbf{1}$ too.

The second semantics, called the **real-time semantics** or simply **the semantics**, is in terms of signals and is denoted $\|\cdot\|$:

$$\|a_I\| = \{\sigma \in \mathsf{Sig}(\Sigma) \mid \exists t \in I \text{ such that } \sigma = a^t\} \qquad \|E^*\| = \|E\|^*$$
$$\|E+F\| = \|E\| \cup \|F\| \qquad \|0\| = \emptyset$$
$$\|E \cdot F\| = \|E\| \cdot \|F\| \qquad \|\mathbf{1}\| = \{\sigma_\epsilon\}$$

Note also that $\|a_{[0,0]}\| = \{\sigma_\epsilon\}$ for any $a \in \Sigma$.

The following straightforward property relates the two types of semantics:

Proposition 3.1.4. *For each real-time regular expression $E \in \mathsf{Rat}(\Sigma_{\mathbb{Q}Int})$,*

$$\|E\| = \bigcup \{\|w\| \mid w \in |E|\}$$

We define the class of **real-time regular languages** over Σ as

$$TReg(\Sigma) = \{L \in \mathsf{Sig}(\Sigma) \mid \exists E \in \mathsf{Rat}(\Sigma_{\mathbb{Q}Int}) \text{ such that } \|E\| = L\}$$

Theorem 3.1.5 (Kleene theorem for RTA). $TRec(\Sigma) = TReg(\Sigma)$.

The Kleene theorem would follow almost immediately from proposition 3.1.4 and the classical Kleene theorem [HU92] if we would have transition-labeled real-time automata rather than state-labeled. Since this kind of automata will further show useful, we define them here and provide the straightforward translations from and to RTA:

Definition 3.1.6. *A **transition-labeled RTA** (*t-RTA*) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, Q_0, Q_f)$ where Q, Σ, Q_0 and Q_f have the same names and properties as in RTA and the transition relation δ satisfies $\delta \subseteq Q \times \Sigma \times \mathbb{Q}Int \times Q$ with $card(\delta) < \infty$.*

Transitions of the form $(q, a, I, r) \in \delta$ will be called *a-transitions*.

Since a transition-labeled RTA is a finite automaton over a finite subset of $\Sigma_{\mathbb{Q}Int}$, we may speak of its language in the classical sense, as the set of words over $\Sigma_{\mathbb{Q}Int}$ which are concatenations of the labels of some accepting run. Let's call this the **abstract language** and denote it as $L_{abs}(\mathcal{A})$. The **real-time language accepted** by $\mathcal{A}$, or simply **the language** of $\mathcal{A}$, denoted $L(\mathcal{A})$, is then the union of the semantics of each word in $L_{abs}(\mathcal{A})$, with this abstract word viewed as a regular expression over $\Sigma_{\mathbb{Q}Int}$, that is,

$$L(\mathcal{A}) = \bigcup \{\|w\| \mid w \in L_{abs}(\mathcal{A})\} \tag{3.1}$$

The translations between RTA and transition-labeled RTA are the usual transformations of a state-labeled automaton into a transition-labeled one and back, with a special case when the empty signal is accepted by the t-RTA:

- Given some RTA $\mathcal{A} = (Q, \Sigma, \lambda, \iota, \delta, Q_0, Q_f)$, a transition-labeled RTA with the same language is $\mathcal{B} = (Q \cup \{t_0\}, \Sigma, \theta, \{t_0\}, Q_f)$ where $t_0 \notin Q$ and

$$\theta = \big\{\big(q, \lambda(r), \iota(r), r\big) \mid (q,r) \in \delta\big\} \cup \big\{\big(t_0, \lambda(q), \iota(q), q\big) \mid q \in Q_0\big\}$$

- For the reverse, given some transition-labeled RTA $\mathcal{B} = (Q, \Sigma, \theta, Q_0, Q_f)$, a RTA whose language is $L(\mathcal{B})$ is $\mathcal{A} = (\theta \cup \{q_\varepsilon\}, \Sigma, \delta, \lambda, \iota, T_0, T_f)$ where
  – for each RTA state $(q, a, I, r) \in \theta$, $\lambda\big((q, a, I, r)\big) = a$ and $\iota\big((q, a, I, r)\big) = I$;
  – $\lambda(q_\varepsilon) = a$ for some $a \in \Sigma$ (assumed nonempty) and $\iota(q_\varepsilon) = [0, 0]$;
  – $T_0 = \{(q, a, I, r) \mid q \in Q_0\} \cup \{q_\varepsilon\}$;
  – $T_f = \{(q, a, I, r) \mid r \in Q_f\} \cup \{q_\varepsilon \mid \sigma_\epsilon \in L(\mathcal{B})\}$;
  – $\delta = \big\{\big((q, a, I, r), (r, b, J, s)\big) \mid (q, a, I, r), (r, b, J, s) \in \theta\big\}$

Hence, when $\sigma_\epsilon \in L(\mathcal{B})$ we must add to $\mathcal{A}$ an initial and final state for accepting σ_ϵ. Note that this state is neither the source nor the target of any transition.

Proof (of the Kleene theorem 3.1.5). The proof is then the following: we have already seen that each RTA $\mathcal{A}$ is equivalent to some t-RTA $\mathcal{B}$. Then, by applying the classical Kleene theorem we get a regular expression $E \in \mathsf{Rat}(\Sigma_{\mathbb{Q}Int})$, that is, a real-time regular expression, whose abstract semantics equals $L_{abs}(\mathcal{B})$. Then, by combining properties 3.1.4 and Identity 3.1 we obtain that the (timed) semantics of E and the language of $\mathcal{A}$ are equal. The reverse implication is similar. □

We end this subsection with a procedure for removing the zeroes from the time labels in RTA which is a straightforward adaptation of the epsilon-elimination procedure for finite automata [HU92]: First observe that transitions of the kind $(q, a, [0, 0], r)$ play the rôle of ε-transitions in finite automata. Consequently, in each t-RTA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, Q_f)$ we split each transition (q, a, I, r) with $0 \in I$ in two parts, the first being $\big(q, a, I \setminus \{0\}, r\big)$ and the second $\big(q, a, [0, 0], r\big)$. We further define, for each state q in $\mathcal{A}$,

$$\varepsilon(q) = \{q' \in Q \mid \text{ there exists a run } \big((q_{i-1}, a_i, [0, 0], q_i)\big)_{i \in [1 \ldots n]} \text{ with } q = q_0, q' = q_n\}$$

Then, the t-RTA $\overline{\mathcal{A}} = (\overline{Q}, \Sigma, \overline{\delta}, Q_0, \overline{Q}_f)$ in which

$$\begin{aligned} \overline{\delta} &= \big\{(q, a, I \setminus \{0\}, r) \mid \exists (q, a, I, s) \in \delta \text{ and } r \in \varepsilon(s)\big\} \\ \overline{Q}_f &= Q_f \cup \{q \in Q \mid Q_f \cap \varepsilon(q) \neq \emptyset\} \end{aligned}$$

is equivalent to $\mathcal{A}$. Note that when translating transition-labeled RTA without zero labels into state-labeled RTA, we will get the special initial state q_ε whose time label is $[0, 0]$, needed for not loosing the empty signal from the accepted language.

All the above observations can be gathered together in the following:

Proposition 3.1.7. *Each transition-labeled RTA is equivalent to some t-RTA in which the interval labels of the transitions do not contain* 0.

Each RTA is equivalent to some RTA in which there exists a single state whose interval label contains 0*, the label of this state is actually* $[0, 0]$ *and no transition enters or leaves this state.*

3.1.3 The problem of complementation of real-time automata

The usual way of showing that a class of automata is closed under complementation is to prove that the automata can be transformed such that for each entry there exists a unique run that starts in an initial state, for then complementation would be accomplished simply by complementing the set of final states. The notions that assure the *uniqueness* of the run for RTA are state-determinism combined with stuttering freeness:

Definition 3.1.8. *A RTA $\mathcal{A}$ is **language deterministic** if each signal in $L(\mathcal{A})$ is associated with a unique run that starts in an initial state.*

*$\mathcal{A}$ is **stuttering-free** if*

- *there exists a state $q_\varepsilon \in Q_0$ which is not connected to any other state and whose time label is $\iota(q_\varepsilon) = [0, 0]$;*
- *the time labels of all the other states do not contain 0;*
- *for each transition $(q, r) \in \delta$, $\lambda(q) \neq \lambda(r)$.*

*$\mathcal{A}$ is **state-deterministic** if initial states have disjoint (state- or interval-)labels and transitions starting in the same state have disjoint (state- or interval-)labels too:*

Whenever $r \neq s$ and either $r, s \in Q_0$ or $(q, r), (q, s) \in \delta$ then either $\lambda(r) \neq \lambda(s)$ or $\iota(r) \cap \iota(s) = \emptyset$.

*$\mathcal{A}$ is called **deterministic** iff it is both state-deterministic and stuttering-free.*

The translations of these notions for transition-labeled RTA are the following:

Definition 3.1.9. *A t-RTA $\mathcal{A}$ is **transition-deterministic** if it has a single initial state and for each state $q \in Q$ and symbol $a \in \Sigma$, if two distinct a-transitions leave q then their time labels are disjoint, i.e.,*

If $(q, a, I, r), (q, a, J, s) \in \delta$ and $I \cap J \neq \emptyset$ then $I = J$ and $r = s$.

*$\mathcal{A}$ is **stuttering-free** if the time labels of the transitions do not contain zero and there are no two distinct adjacent transitions labeled with the same symbol, i.e., if $(q, a, I, r), (r, b, J, s) \in \delta$ then $a \neq b$.*

*$\mathcal{A}$ is **deterministic** if it is state-deterministic and stuttering-free.*

Proposition 3.1.10. *The translations between RTA and t-RTA provided in section 3.1.2 are such that*

- *state determinism in RTA is translated to transition determinism in t-RTA and vice-versa and*
- *stuttering-freeness in RTA is translated to stuttering-freeness in t-RTA and vice-versa.*

It is clear that determinism implies language determinism. On the other hand, state-determinism without stuttering freeness does not imply language determinism, due to the nondeterministic nature of choosing the stuttering steps. But a more important observation is that stuttering-free RTA are *strictly less expressive* than general RTA: consider the language of constant signals with integer

length $L_{\mathbb{N}} = \{a^n \mid n \in \mathbb{N}\}$, which is accepted by the RTA in the Figure 3.2. Observe that this RTA is language deterministic.

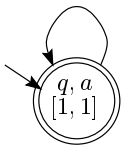


Fig. 3.2. An RTA for the language $L_{\mathbb{N}}$.

Proposition 3.1.11. *$L_{\mathbb{N}}$ cannot be accepted by any stuttering-free RTA.*

Proof. The proof is based on the intuition that a stuttering-free RTA for $L_{\mathbb{N}}$ would need an infinite number of states:

Suppose we had a stuttering-free automaton $\mathcal{A} = (Q, \Sigma, \delta, \lambda, \iota, T_0, T_f)$ which would recognize $L_{\mathbb{N}}$. We may consider $\Sigma = \{a\}$ since any state with other state-label cannot be in an accepting run. Then, since the automaton is stuttering-free, $\delta = \emptyset$. Hence the number of accepting runs in $\mathcal{A}$ equals the number of states that are both initial and final. Denote then μ the max in $\mathbb{R}_{\geq 0} \cup \{\infty\}$ of the time labels of these initial and final states. But then both the assumption $\mu = \infty$ and $\mu < \infty$ lead to a contradiction:

If $\mu = \infty$ then for some state $q \in Q_0 \cap Q_f$ we have that $\iota(q) = \langle l, \infty[$ where '$\langle$' is any left parenthesis. Then any constant signal $\sigma = a^t$ with $t \in \langle l, \infty[$ would be accepted by $\mathcal{A}$, contradicting the assumption that only signals with natural endpoints are accepted.

On the other hand, if $\mu < \infty$ then any constant signal $\sigma = a^n$ with $n \in \mathbb{N}$, $n > \mu$, is not accepted by $\mathcal{A}$, again a contradiction. □

This proof can be easily adapted for showing that state-clock automata [RS97] cannot accept the following language:

$$\|[b]_{]0,\infty[}| \cdot L_{\mathbb{N}} \cdot |[b]_{]0,\infty[}\| = \left\{b^{t_1} a^k b^{t_2} \mid k \in \mathbb{N}, t_1, t_2 \in \mathbb{R}_{\geq 0}\right\}$$

A similar property can be shown for event-clock automata, but in which stuttering is replaced by ε-steps.

Despite Proposition 3.1.11, there is no problem in building a RTA for the complement of $L_{\mathbb{N}}$: it is the RTA in Figure 3.3 below.

Figure 3.4 below gives an example of how to transform some stuttering RTA into a stuttering-free RTA.

Hence we discover the need of computing the "sum" of two intervals and the "star" of some interval, or, in a more formal setting, the need for defining some operations that satisfy the following relations suggested by Figures 3.3 and 3.4:

$$\mathbb{R} \setminus \{1\}^* = \{1\}^* +]0, 1[\quad \text{and} \quad [2, 3]^* = \{0\} \cup [2, 3] \cup [4, \infty[$$

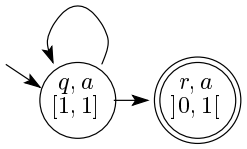


Fig. 3.3. A RTA for the language $\mathsf{Sig}(\{a\}) \setminus L_{\mathbb{N}}$.

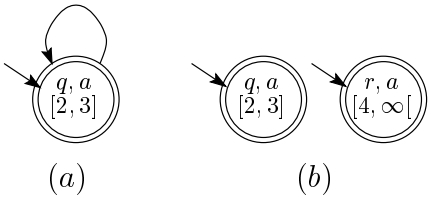


Fig. 3.4. The stuttering RTA at (a) is equivalent to the stuttering-free one at (c).

3.2 The Kleene algebra of sets of real numbers

The powerset of the nonnegative numbers $\mathcal{P}(\mathbb{R}_{\geq 0})$ is naturally endowed with a concatenation operation: it is addition extended to sets of reals:

$$X + Y = \{x + y \mid x \in X, y \in Y\} \text{ for all } X, Y \subseteq \mathbb{R}$$

whose unit is the set $\mathbf{0} = \{0\}$.

We can also define a star operation via the usual least fixpoint construction

$$X^* = \bigcup_{n \in \mathbb{N}} nX$$

where the multiples of X are defined as usual: $0X = \mathbf{0}$ and $(n+1)X = nX + X$.

The following theorem can be easily verified:

Theorem 3.2.1. *The structure* $\mathcal{P}(\mathbb{R}_{\geq 0}) = \big(\mathcal{P}(\mathbb{R}_{\geq 0}), \cup, +, (\cdot)^*, \emptyset, \mathbf{0}\big)$ *is a* ***commutative Kleene algebra*** *(see Definition 2.1.3).*

Because a complement operation is available, $\neg X = \mathbb{R}_{\geq 0} \setminus X$, we actually get a **commutative complemented Kleene algebra**, i.e., a boolean algebra which is also a commutative Kleene algebra.

Note also that summation with singletons distributes over intersection:

$$\{x\} + \big(Y \cap Z\big) = \big(\{x\} + Y\big) \cap \big(\{x\} + Z\big) \tag{3.2}$$

but distributivity of summation over intersection is not valid in general as the following example shows:

$$\begin{aligned} \big(]2,3[+]4,5[\big) \cap \big(]2,3[+]5,6[\big) &=]7,8[\\ ]2,3[+ \big(]4,5[\cap]5,6[\big) &= \emptyset \end{aligned}$$

3.2.1 Normal forms

Denote $\mathcal{K}(\mathbb{Q}Int)$ the sub-(commutative complemented Kleene) algebra generated by $\mathbb{Q}Int$ in $\mathcal{P}(\mathbb{R}_{\geq 0})$, that is the family of sets which can be obtained from intervals of $\mathbb{Q}Int$ by applying union, summation, star and complementation.

Definition 3.2.2. *A set $X \in \mathcal{K}(\mathbb{Q}Int)$ can be **written in normal form** if there exist X_1, X_2 two finite unions of rational intervals and $k \in \mathbb{Q}_{\geq 0}$, $N \in \mathbb{N}$ such that*

$$X = X_1 \cup (X_2 + \{k\}^*) \tag{3.3}$$

$$\textit{and} \qquad X_1 \subseteq [0, Nk[, X_2 \subseteq [Nk, (N+1)k[\tag{3.4}$$

*We call N the **bound** of the normal form.*

We will work with normal forms in which X_1 and X_2 are unions of *disjoint intervals*. It is straightforward how to transform some normal form such that this property holds.

Normal forms are not unique: for the normal form in the definition and some $p \in \mathbb{N}$, the following expression:

$$X = \Big(X_1 \cup \big(X_2 + \{0, k, 2k, \ldots, (p-1)k\}\big)\Big) \cup \big(X_2 + \{pk\} + \{k\}^*\big)$$

is a normal form too, but with bound $N + p$.

Any finite union of rational intervals X can be put into normal form: when X is bounded from above by M then $X = X \cup \big(\emptyset + \{1\}^*\big)$ is a normal form with bound $\lceil M \rceil$. When X is unbounded, suppose $X = X_1 \cup [M, \infty[$ is some decomposition of it into disjoint intervals, where $M \notin \mathbb{N}$. Then

$$X = \Big(X_1 \cup \big[M, \lceil M \rceil\big[\Big) \cup \Big(\big[\lceil M \rceil, \lceil M \rceil + 1\big[+ \{1\}^*\Big)$$

is a normal form with bound $\lceil M \rceil$.

Proposition 3.2.3. *For each set X written into normal form as $X = X_1 \cup \big(X_2 + \{k\}^*\big)$, $X = \emptyset$ iff both X_1 and X_2 are empty.*

This property, though trivial, has its own importance since we will use normal forms as time labels in automata and we still want to have a decidable emptiness problem for the resulting automata.

Sometimes, after the application of different operations to normal forms we might not be able to get directly a normal form; instead, we might get a **weak normal form**, which is a decomposition as in Identity 3.3 but without the additional requirement 3.4 on the existence of the bound. As an example we have the following:

- $X = \big(]2, 3[\cup]4, 6[\big) \cup \Big(\big([6, 7[\cup [8, 9[\big) + \{3\}^*\Big)$ is written in normal form with bound 2 since we have $X_1 =]2, 3[\cup]4, 6[\subseteq [0, 6[$, $X_2 = [6, 7[\cup]8, 9[\subseteq [6, 9[$.
- $Y = \big(]2, 3[\cup]4, 7[\big) \cup \big([5, 7] + \{3\}^*\big)$ is a weak normal form which is not a normal form: there is no $N \in \mathbb{N}$ such that $[5, 7] \subseteq \big[3N, 3(N+1)\big[$ and $]2, 3[\cup]4, 6[\subseteq [0, 3N[$.

However both expand to the same set:

$$X = Y =]2,3[\cup]4,7] \cup \bigcup_{n\geq 3} [3n-1, 3n+1]$$

Lemma 3.2.4. *Weak normal forms can be transformed into normal forms.*

Proof. Take $X = X_1 \cup (X_2 + \{k\}^*)$ some weak normal form and define $M = \sup(\sup X_1, \sup X_2)$. Two cases arise:

1. $M = \infty$. This means that there exists some $L \in \mathbb{R}_{\geq 0}$ such that $\langle L, \infty[\subseteq X$, where '$\langle$' denotes some left parenthesis. Define then $n = \left\lfloor \frac{L}{k} \right\rfloor$, hence $nk \leq L < (n+1)k$. Then for each $i \geq n+1$, $X_2 + \{ik\} \subseteq \left[(n+1)k, \infty\right[\subseteq \langle L, \infty[$. It follows that X is a finite union of intervals

$$X = X_1 \cup \left(\bigcup_{i=0}^{n} \Big(X_2 + \{ik\} \Big) \right) \cup \langle L, \infty[$$

   and thus we know how to transform it into normal form.
2. $M < \infty$. Define then $n = \left\lfloor \frac{M}{k} \right\rfloor + 1$, hence $(n-1)k \leq M < nk$. Define further

$$\begin{aligned} Z_1 &= X_1 \cup \left[\left(\bigcup_{i=0}^{n-1} \Big(X_2 + \{ik\} \Big) \right) \cap [0, nk[\right] \\ Z_2 &= \left(\bigcup_{i=1}^{n} \Big(X_2 + \{ik\} \Big) \right) \cap \left[nk, (n+1)k\right[\end{aligned}$$

   We claim that $X = Z_1 \cup \big(Z_2 + \{k\}^*\big)$ which is a normal form with bound n.

To prove this, observe first that for each $i > j$, $i, j \in \mathbb{N}$,

$$\big(X_2 + \{ik\}\big) \cap [0, jk[= \emptyset$$

Moreover, distributivity of summation of singletons over intersection (property 3.2) implies that, for each $j \in \mathbb{N}$,

$$\big(X_2 + \{jk\}\big) \cap \left[(n+j)k, \infty\right[= \big(X_2 \cap [nk, \infty[\big) + \{jk\} = \emptyset$$

due to the fact that $X_2 \subseteq [0, M] \subset [0, nk[$. This also implies that, for each $i \leq j$, $i, j \in \mathbb{N}$,

$$\big(X_2 + \{ik\}\big) \cap \left[(n+j)k, \infty\right[= \emptyset$$

Therefore, by the same distributivity property 3.2 we get

$$\begin{aligned}\big(X_2+\{k\}^*\big)\cap\big[(n+j)k,(n+j+1)k\big[&= \\ &= \Big(\bigcup_{i=j+1}^{n+j}\big(X_2+\{ik\}\big)\Big)\cap[(n+j)k,(n+j+1)k[= \\ &= \Big(\bigcup_{i=1}^{n}\big(X_2+\{ik\}\big)\Big)\cap[nk,(n+1)k[+\{jk\}\end{aligned}$$

and further

$$\begin{aligned}X &= X_1\cup\big(X_2+\{k\}^*\big)\\ &= X_1\cup\Big(\big(X_2+\{k\}^*\big)\cap\Big([0,nk[\cup\bigcup_{j\geq 0}[(n+j)k,(n+j+1)k[\Big)\Big)\\ &= X_1\cup\Big(\bigcup_{i=0}^{n-1}\Big(X_2+\{ik\}\Big)\cap[0,nk[\Big)\cup\\ &\qquad\bigcup_{j\geq 0}\Big(\Big(\bigcup_{i=1}^{n}\Big(X_2+\{ik\}\Big)\cap[nk,(n+1)k[\Big)+\{jk\}\Big)\\ &= Z_1\cup\big(Z_2+\{k\}^*\big)\end{aligned}$$

□

3.2.2 A normal form theorem

The key result for normal forms is the following:

Theorem 3.2.5. *Each $X\in\mathcal{K}(\mathbb{Q}Int)$ can be written in normal form.*

Proof. We must show that the result of any operation applied to some normal forms can be put into normal form. We first list some useful identities valid in $\mathcal{P}(\mathbb{R})$ [Con71]:

$$X^{**} = X^* \tag{3.5}$$
$$(X\cup Y)^* = X^*+Y^* \tag{3.6}$$
$$(X^*+Y)^* = \{0\}\cup(X^*+Y^*+Y) \tag{3.7}$$

We employ the notations $lcm(p,q)$ and $gcd(p,q)$ where $p,q\in\mathbb{Q}_{\geq 0}$ as the generalization of lcm and gcd from integers. The formal definitions are:

$$\begin{aligned}lcm(p,q) &= \min\{r\in\mathbb{Q}_{\geq 0}\mid\exists l,m\in\mathbb{N}\text{ such that }lp=r=mq\}\\ gcd(p,q) &= \frac{pq}{lcm(p,q)}\end{aligned}$$

We also use the following *ultimately periodicity property*:

For each n distinct positive rationals $a_i \in \mathbb{Q}_{\geq 0}$ ($i \in [1 \ldots n]$) we have that $\{a_1, \ldots, a_n\}^*$ is *ultimately periodic*, i.e., there exist some *finite* set of rationals B and some rationals $q, r \in \mathbb{Q}_{\geq 0}$ such that

$$\{a_1, \ldots, a_n\}^* = B \cup \big(\{q\} + \{r\}^*\big) \tag{3.8}$$

This property can be seen as an equivalent form of the normal form theorem for regular languages over a one letter alphabet. For a direct proof note first that, given two rationals $p, q \in \mathbb{Q}_{\geq 0}$,

$$\{p, q\}^* = B \cup \Big(\big\{lcm(p, q)\big\} + \big\{\gcd(p, q)\big\}^*\Big)$$

where $B = \big\{\alpha \in \mathbb{Q}_{\geq 0} \mid \alpha < lcm(p, q), \alpha = lp + mq, l, m \in \mathbb{N}\big\} = \{p, q\}^* \cap \big[0, lcm(p, q)\big[$. The property 3.8 follows from this by induction upon the number of elements in the starred set.

Fix now two normal forms $X = X_1 \cup (X_2 + \{k\}^*)$ with bound M and $Y = Y_1 \cup (Y_2 + \{l\}^*)$ with bound N and denote $m = lcm(k, l)$. We then get the following form for $X \cup Y$:

$$X_1 \cup Y_1 \cup \left(\left(\bigcup_{i=0}^{m/k-1} \Big(X_2 + \{ik\}\Big) \cup \bigcup_{i=0}^{m/l-1} \Big(Y_2 + \{il\}\Big)\right) + \{m\}^*\right)$$

This is a weak normal form and Lemma 3.2.4 shows how to transform it into normal form.

For $X + Y$, distributivity of $+$ over $\cup$ transforms it into:

$$\big(X_1 + Y_1\big) \cup \big(X_1 + Y_2 + \{l\}^*\big) \cup \big(X_2 + Y_1 + \{k\}^*\big) \cup \big(X_2 + Y_2 + \{k\}^* + \{l\}^*\big)$$

An instantiation of identity 3.6 gives $\{k\}^* + \{l\}^* = \{k, l\}^*$. The ultimately periodicity property 3.8 gives a normal form for this set and thence we have above a union of weak normal forms which we already know how to bring to normal form.

For X^* we have two cases. The first one occurs when one of X_1 and X_2 contains a nonpoint interval. Then *the set X^* is a finite union of rational intervals*, so we know how to bring it into normal form.

To prove this claim, note that for each nonpoint interval, e.g., $]a, b]$ (that is $b - a > 0$), denoting $m_0 = \left\lceil \dfrac{a}{b-a} \right\rceil$, we have that $]a, b]^* = \mathbf{0} \cup \bigcup_{i=1}^{m_0-1}]ia, ib] \cup]m_0 a, \infty[$ since the choice of m_0 assures that $(m_0 + 1)a < m_0 b$. Hence from the m_0-th iteration the intervals start to overlap.

The second case for X^* is when both X_1 and X_2 consist of point intervals. Applying identity 3.6 we get $X^* = X_1^* + \big(X_2 + \{k\}^*\big)^*$. Then by the ultimately periodicity property 3.8 X_1 can be written into normal form, so we may concentrate on $\big(X_2 + \{k\}^*\big)^*$.

By identities 3.7 and 3.6 we further get

$$\big(X_2 + \{k\}^*\big)^* = \{0\} \cup \big(X_2 + X_2^* + \{k\}^*\big) \;=\; \{0\} \cup \Big(X_2 + \big(X_2 \cup \{k\}\big)^*\Big)$$

Finally the ultimately periodicity property 3.8 tells us that $\big(X_2 \cup \{k\}\big)^*$ can be put into normal form and therefore this case reduces to a summation of normal forms.

For $\neg X$ the strength of the normal form, i.e., the additional requirement on the existence of the bound N helps us giving directly the result: it is

$$\neg X = \Big(\neg X_1 \cap [0, Nk[\Big) \cup \Big(\big(\neg X_2 \cap [Nk, (N+1)k[\big) + \{k\}^*\Big)$$

and the bound of this normal form is N too. □

In [CG00] it is proved that the set of finite unions of n-dimensional normal forms in which $X_1 = \emptyset$ forms a boolean algebra. The essential novelty in our result is the closure of 1-dimensional normal forms under star.

Though the proof of Theorem 3.2.5 is based on the same technique that gives the normal form of regular languages over a one-letter alphabet, it cannot be a simple corollary of that result: even if we restrict our attention to the algebra generated by intervals with natural bounds, denote it $\mathbb{N}Int$, we find *two* generators: the point set $\{1\}$ and the nonpoint interval $]0, 1[$. Neither of them may generate the other: $\{1\}$ generates just sets with isolated points or complements of such sets (i.e., countable or co-countable sets), while $]0, 1[$ generates just finite unions of intervals (it cannot generate $]0, 1[+\{1\}^*$).

One might also think that the result follows from Eilenberg's theory of automata with multiplicities [Eil74]. But this is not the case either since in that work star is defined via some formal power series and one cannot prove, unless defining some suitable equivalence on power series, that e.g. $[0, 1[^* = [0, \infty[$.

Finally note the interesting relation which holds between the two generators of $\mathbb{N}Int$, showing they are not independent:

$$]0, 1[^* = (\mathbf{0} \cup]0, 1[) + \{1\}^* \tag{3.9}$$

3.2.3 Matrices of normal forms

At the end of this section we make a brief excursion into matrix theory. We construct, as in [Koz94] the Kleene algebra of $n \times n$ matrices over $\mathcal{P}(\mathbb{R}_{\geq 0})$ whose operations are the matrix extensions of the operations in the Kleene algebra $\mathcal{P}(\mathbb{R}_{\geq 0})$:

$$(A \cup B)_{ij} = A_{ij} \cup B_{ij} \qquad (A + B)_{ij} = \bigcup_{k=1}^{n} (A_{ik} + B_{kj})$$

$$A^* = \bigcup_{n \in \mathbb{N}} nA$$

where $0A = I_n$ and $(n+1)A = nA + A$, I_n denoting the unit for matrix summation, i.e.

$$(I_n)_{ij} = \begin{cases} \{0\} & \text{iff } i = j \\ \emptyset & \text{iff } i \neq j \end{cases}$$

If we write in detail the components of A^* we have:

$$(A^*)_{ij} = \bigcup \left\{ A_{i\,i_1} + A_{i_1 i_2} + \ldots + A_{i_m j} \mid i_1, \ldots, i_m \in [1 \ldots n], m \in \mathbb{N} \right\} \cup \{0 \mid i = j\} \tag{3.10}$$

The star of a matrix A can be computed by the following well-known Floyd-Warshall-Kleene algorithm [Con71, Eil74]: we recursively define a sequence of $n+1$ matrices $A(k)$ ($0 \leq k \leq n$) with

$$\begin{aligned} A(0) &= A \cup I_n \\ A(k)_{ij} &= A(k-1)_{ij} \cup \left(A(k-1)_{ik} + \left(A(k-1)_{kk} \right)^* + A(k-1)_{kj} \right) \end{aligned} \tag{3.11}$$

Proposition 3.2.6 ([Eil74]). *$A(n) = A^*$ for any matrix over $\mathcal{P}(\mathbb{R}_{\geq 0})$.*

The classical proof may run as follows: one proves first that $(nA)_{ik} + (mA)_{kj} \subseteq ((n+m)A)_{ij}$. This implies that $(A^*)_{ik} + \left((A^*)_{kk}\right)^* + (A^*)_{kj} \subseteq (A^*)_{ij}$. Then one shows that $A(k)_{ij} \subseteq (A^*)_{ij}$ by induction on k and hence get the left-to-right inclusion.

The right-to-left inclusion follows by proving that

$$A(k)_{ij} = \bigcup \left\{ A_{i\,i_1} + A_{i_1 i_2} + \ldots + A_{i_m j} \mid i_1, \ldots, i_m \in [1 \ldots k], m \in \mathbb{N} \right\} \cup \{0 \mid i = j\}$$

by induction on k.

Corollary 3.2.7. *If A is a matrix of normal forms then A^* can be transformed into a matrix of normal forms too.*

Corollary 3.2.8. *For each matrix of normal forms A if for all indices $i \neq j$ we have that $0 \notin A_{ij}$ then for all indices $i \neq j$, $0 \notin (A^*)_{ij}$.*

Proof. This is a corollary of relation 3.10: for any $i_1, \ldots, i_m \in [1 \ldots n]$ consider the sum $A_{i\,i_1} + A_{i_1 i_2} + \ldots + A_{i_m j}$. As we assumed $i \neq j$ we must have some $p \in [1 \ldots m]$ such that $i_p \neq i_{p+1}$. Thence $0 \notin A_{i_p i_{p+1}}$ and therefore the sum itself does not contain 0. By identity 3.10, it follows that $0 \notin (A^*)_{ij}$. □

Note however that for any i, $(A^*)_{ii}$ will always contain 0.

3.3 Determinization and complementation of RTA

The above theory suggests that "periodic" constraints may replace intervals in RTA:

Definition 3.3.1. *An **augmented real-time automaton** is a tuple $\mathcal{A} = (Q, \Sigma, \delta, \lambda, \iota, Q_0, Q_f)$ where Q, Σ, Q_0, Q_f, δ and λ are the same as in RTA while $\iota : Q \to \mathcal{K}(\mathbb{Q}Int)$ (actually ι gives a normal form).*

Augmented RTA work similarly to RTA: *runs* have the same definition and a signal σ is *associated with* a run $(q_i)_{i \in [1 \ldots n]}$ iff $\sigma = \lambda(q_1)^{t_1} \cdot \ldots \cdot \lambda(q_n)^{t_n}$ The emptiness problem is again decidable

in linear time w.r.t. $card(Q)$. Note that we need a preliminary step in which states q whose interval label *denotes the empty set* are removed. It is here where we need Proposition 3.2.3.

The different notions of determinism remain unchanged for augmented RTA; hence we will speak of state-deterministic augmented RTA and stuttering-free augmented RTA in the sense of Definition 3.1.8.

We also have a transition-labeled version of augmented RTA, called in the sequel **augmented t-RTA**, which are tuples $\mathcal{B} = (Q, \Sigma, \delta, Q_0, Q_f)$ like t-RTA, the difference being that the transition relation is time-labeled with normal forms instead of just intervals: $\delta \subseteq Q \times \Sigma \times \mathcal{K}(\mathbb{Q}Int) \times Q$. The different notions of determinism in Definition 3.1.9 are the same for augmented t-RTA, the translations between RTA and t-RTA and back from subsection 3.1.2 work with augmented automata too and Proposition 3.1.10 is valid for augmented automata.

The following theorem says that we do not increase the expressive power of RTA if we use normal forms instead of mere intervals:

Theorem 3.3.2. *$TReg(\Sigma)$ equals the class of languages accepted by augmented RTA.*

The proof is very close to the one of Theorem 3.1.5 and is based on the following property of regular expressions:

$$\|[a]_{]a,b[\cup(]c,d[+\{k\}^*)}\| = \|[a]_{]a,b[} + [a]_{]c,d[} \cdot [a]^*_{\{k\}}\|$$

Of course, we also have to redefine regular expressions allowing normal forms as indices for atoms.

The first step in determinization is the achievement of stuttering-freeness and the proof runs smoother for *augmented t-RTA*:

Theorem 3.3.3. *Each augmented t-RTA is equivalent to some stuttering-free augmented t-RTA.*

Proof. As a preliminary step, in the given augmented t-RTA we remove zeroes from the time labels by applying Proposition 3.1.7, slightly modified for handling normal forms instead of mere intervals. We also assume that all transitions with empty time label have been removed.

We achieve stuttering freeness by removing all stuttering a-transitions for some $a \in \Sigma$, and then repeating this for all the other letters in Σ. The idea is to find, for each pair of states (q_1, q_2) the set of durations of signals that are associated with runs starting in q_1, ending in q_2 and containing only a-transitions. For this we need to recursively add all the intervals of the transitions that may lie on such a run. This is the place where we apply the normal form theorem 3.2.5 and the algorithm for computing the star of a matrix of sets of positive numbers. The formalization is the following:

Start with some augmented t-RTA $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_f)$ and number its states as $Q = \{q_1, \ldots, q_p\}$. Construct a matrix A whose elements are the interval labels of the a-labeled states:

$$A_{ij} = \begin{cases} X & \text{iff } (q_i, a, X, q_j) \in \delta \\ \emptyset & \text{otherwise} \end{cases}$$

Then $(A^*)_{ij}$ consists of the lengths of signals associated with runs starting in q_i, ending in q_j and consisting of a-transitions only. More formally,

$$(A^*)_{jk} = \bigcup \{X_1+\ldots+X_m \mid (r_{i-1}, a, X_i, r_i)_{i\in[1\ldots n]} \text{ is a run in } \mathcal{A} \text{ and } r_0 = q_j, r_n = q_k\} \cup \{0 \mid j = k\} \quad (3.12)$$

This fact is a corollary of identity 3.10.

Computation of A^* is done by the Floyd-Warshall-Kleene algorithm (3.11). Note here the importance of Corollary 3.2.7: the elements of A^* are still normal forms, hence they may be used for labeling some new transitions of an augmented RTA. Hence, while non-a-transitions will be preserved, the *nonempty* components of A^* will replace all a-transitions: their time label will be $(A^*)_{ij}$ and they will be connected only to states from which no other a-transition is issued.

Formally, consider a disjoint copy of Q, $Q' = \{q'_i \mid q_i \in Q\}$; the primed states will be reached exactly after an a-transition. Build then $\mathcal{B} = (Q \cup Q', \Sigma, \overline{\delta}, Q_0, Q_f \cup Q'_f)$ where Q'_f is the set of copies of final states and

$$\overline{\delta} = \{(q, b, X, r), (q', b, X, r) \mid b \neq a, (q, b, X, r) \in \delta\} \cup \left\{\left(q_i, a, (A^*)_{ij} \setminus \{0\}, q'_j\right) \mid \left(A^*\right)_{ij} \setminus \{0\} \neq \emptyset\right\}$$

The need for removing zero from the new transitions comes from the fact that we do not want to add stuttering steps involving the other symbols from Σ.

The equivalence of $\mathcal{A}$ and $\mathcal{B}$ follows from the observation that a run $\left((r_{i-1}, a_i, X_i, r_i)\right)_{i\in[1\ldots n]}$ in $\mathcal{A}$ associated with some signal σ, can be transformed into a run in $\mathcal{B}$ for σ by replacing all maximal sequences of a-transitions with the appropriate a-transition time-labeled from A^* and by priming the state that follows this transition.

Observe that, by construction, no two a-transitions are directly connected. On the other hand, all non-a-transitions involving nonprimed states are just copied, hence no stuttering transitions are added on these states. Finally, the primed states are not involved in any stuttering transitions since they are targets of a-transitions and sources of non-a-transitions. This shows that by recursively applying this construction for all letters in Σ we end with a stuttering-free augmented RTA. □

The number of states in the final t-RTA is $2^{card(\Sigma)} \cdot card(Q)$, since at each step the states are at most duplicated. Concerning the number of transitions, note that, for each $a \in \Sigma$, at each step the number of a-transitions is either doubled (if a is not chosen at that moment for stuttering elimination) or squared. Since there is a single step in which the number of a-transitions is squared, an upper bound for the number of a-transitions would be $2^{2\cdot card(\Sigma)} \cdot m_a^2$, where m_a is the initial number of a-transitions. Note that the earlier we choose to eliminate the stuttering a-transitions, the smaller the number of a-transitions we obtain. This is because squaring would apply to a smaller number of transitions.

The last step in the determinization process is the achievement of determinism in stuttering-free automata. This time, the construction works smoother for *state-labeled* automata:

Theorem 3.3.4. *Each stuttering-free augmented RTA is equivalent to some deterministic augmented RTA.*

Proof. Note that, as we work with state-labeled RTA, the given stuttering-free RTA has the special initial state q_ε whose time-label is $[0,0]$ and which is not connected to any other state.

Start with a stuttering-free augmented RTA $\mathcal{B} = \big(Q \cup \{q_\varepsilon\}, \Sigma, \delta, \lambda, \iota, Q_0 \cup \{q_\varepsilon\}, Q_f\big)$ with $q_\varepsilon \notin Q$. For some subset of states $S \subseteq Q$ we write $\lambda(S) = a$ as a shortcut for saying that all states in S are identically labeled with a.

If $\mathcal{B}$ were untimed, the states of the deterministic automaton would have been identically state-labeled subsets of Q and we would draw a transition from some S_1 with $\lambda(S_1) = a$ to some S_2 with $\lambda(S_2) = b$ iff $S_2 = \{r \in Q \mid \exists q \in S_1 \text{ s.t. } (q,r) \in \delta\}$. Taking into account the time labels is done by splitting S_2 into several "smaller" states, each one with its distinct time label, such that their time give a partition of $\mathbb{R}_{>0}$.

To each $U \subseteq Q$ with $\lambda(U) = a$ we associate the set of time labels appearing in U:

$$Tl(U) = \{X \in \mathcal{K}(\mathbb{Q}Int) \mid \exists q \in U \text{ s.t. } \iota(q) = X\}$$

Let $\mathcal{R}$ denote the set of triples $[S, S', a]$ where $a \in \Sigma$ and $S' \subseteq S \subseteq Q$ with $\lambda(S) = a$. Define then $\overline{\lambda}\big([S,S',a]\big) = a$ and

$$\overline{\iota}\big([S,S',a]\big) = \mathbb{R}_{>0} \cap \Big(\bigcap Tl(S')\Big) \cap \neg \Big(\bigcup Tl(S \setminus S')\Big)$$

where the usual conventions $\bigcap \emptyset = \mathbb{R}_{\geq 0}$ and $\bigcup \emptyset = \emptyset$ apply. Intuitively the control passes through $[S,S',a]$ iff in $\mathcal{B}$ the control may pass through some state in S' but not through any of the states in $S \setminus S'$. We put $\mathbb{R}_{>0}$ in front of $\overline{\iota}([S,S',a])$ because otherwise we would lose stuttering-freeness. Also note that it is *here* where we need the result that normal forms are closed under complementation, because we need to put $\overline{\iota}([S,S',a])$ into normal form and $\overline{\iota}([S,S',a])$ contains complementation.

Hence we build $\mathcal{C} = (\mathcal{R} \cup \{q_\varepsilon\}, \Sigma, \tilde{\delta}, \tilde{\lambda}, \tilde{\iota}, \mathcal{R}_0, \mathcal{R}_f)$ in which

- $\tilde{\delta}$ consists of transitions going from each $[S,S',a] \in \mathcal{R}$ to each tuple $[U,U',b]$ defined by

$$U = \big\{q \in Q \mid \exists r \in S' \text{ s.t. } (r,q) \in \delta \text{ and } \lambda(q) = b\big\} \quad \text{and} \quad U' \subseteq U.$$

  Case $U' = \emptyset$ stands for the situation when the length of the current state in the signal is not in any of the sets from $Tl(U)$. Note how states $[\emptyset,\emptyset,a]$ time-labeled with $\mathbb{R}_{>0}$ play the role of the trap states in finite automata.
- initial and final states are

$$\begin{aligned} \mathcal{R}_0 &= \Big\{[S,S',a] \in \overline{Q} \mid S = \big\{q \in Q_0 \mid \lambda(q) = a\big\}\Big\} \cup \{q_\varepsilon\} \\ \mathcal{R}_f &= \big\{[S,S',a] \in \overline{Q} \mid S' \cap Q_f \neq \emptyset\big\} \cup \big\{q_\varepsilon \mid \sigma_\epsilon \in L(\mathcal{B})\big\} \end{aligned}$$

The proof that $\mathcal{C}$ is equivalent to $\mathcal{B}$ proceeds by induction on the number of discontinuities in a signal. The construction assures that, at each discontinuity, exactly one state can be chosen such that the control goes to that state. □

The complexity of this construction is exponential in the number of states: by denoting $n = card(Q)$, observe first that the number of states $[S, S', a]$ where $card(S) = k$ is at most $2^k \cdot \left(\begin{array}{c} n \\ k \end{array} \right)$ (at most due to the fact that some sets of states S might not be consistently state-labeled). Therefore the cardinality of $\mathcal{R}$ is at most

$$\sum_{k=0}^{n} \left(2^k \cdot \left(\begin{array}{c} n \\ k \end{array} \right) \right) = 3^n$$

Theorem 3.3.5. *$TRec(\Sigma)$ is closed under complementation.*

The universality problem for $TRec(\Sigma)$ is decidable.

Proof. This is a corollary of Theorem 3.3.4 and Proposition 3.1.2. The important property provided by the construction of the deterministic augmented RTA $\mathcal{C}$ in this theorem is that each signal (including the empty signal!) *is associated with a unique run that starts in* T_0. Hence the augmented RTA that accepts $Sig(\Sigma) \setminus L(\mathcal{C})$ is obtained by complementing the set of final states of $\mathcal{C}$. □

Let us finally underline the need for theorem 3.2.5 in determinization: in our construction, we actually build an automaton whose time labels are in fact *extended regular expressions (i.e., using complementation) over intervals*. In the absence of theorem 3.2.5, such an automaton would not be an augmented RTA any longer and we would be in no position to decide whether, after complementing the set of final states, the resulting automaton would still be an augmented RTA. This would make questionable the decidability of the universality problem.

It is actually this problem what stops the application of the determinization construction for RTA whose time labels lie in a class larger than $\mathbb{Q}Int$ in which comparison of the time bounds is effective - for example, the class of intervals whose bounds are algebraic numbers. If this class of intervals is chosen for time labels, it is unclear whether the universality problem remains decidable.

3.4 The Pumping Lemma and expressiveness issues

Lemma 3.4.1 (Pumping Lemma). *If a language L is accepted by a RTA then there exists $N \in \mathbb{N}$ such that each signal σ having at least N discontinuities can be factored into three signals $\sigma = \sigma_1 \cdot \sigma_2 \cdot \sigma_3$, such that σ_2 contains at least one discontinuity and for any $n \in \mathbb{N}$ we have $\sigma_1 \cdot \sigma_2^n \cdot \sigma_3 \in L$.*

Proof. The proof of this lemma is almost the same as in the untimed case, the difference lying in the reference to discontinuities. Take $\mathcal{A} = (Q, \Sigma, \delta, \lambda, \iota, Q_0, Q_f)$ a *stuttering-free* augmented RTA accepting L and define $N = card(Q) + 1$. It is clear that each signal $\sigma \in L$ having N discontinuities must be accepted by some run having exactly $N + 1$ states, hence one of the state must be repeated throughout the run. Since we assumed that $\mathcal{A}$ is stuttering-free we cannot have self loops at the repeated state. Hence the part of the run which can be repeated must contain at least two distinctly state-labeled states and therefore σ_2 must contain a discontinuity. □

Proposition 3.4.2. *The language* $L_{nonreg} = \{\sigma \in \mathsf{Sig}(\{a,b\}) \mid \ell(\sigma) \in [1,3]\}$ *is not real-time regular.*

Proof. Supposing L_{nonreg} is real-time regular, we may pick up a signal $\sigma \in L_{nonreg}$ such that its number of discontinuities is more than the natural number N provided by the Pumping Lemma. An example is the signal

$$\sigma = \underbrace{a^{\frac{k}{N}} b^{\frac{k}{N}} \cdot \ldots \cdot a^{\frac{k}{N}} a^{\frac{k}{N}}}_{N\text{times}}$$

that is, for each $k \in [1 \ldots N]$, $\sigma(t) = a$ for $t \in \left[\frac{2k-2}{N}, \frac{2k-1}{N}\right)$ and $\sigma(t) = b$ for $t \in \left[\frac{2k-1}{N}, \frac{2k}{N}\right)$.

Then by the Pumping Lemma σ can be factored as $\sigma = \sigma_1 \cdot \sigma_2 \cdot \sigma_3$ such that σ_2 has at least a discontinuity, (and hence $\ell(\sigma_2) > 0$) and $\sigma_1 \cdot \sigma_2^n \cdot \sigma_3 \in L_{nonreg}$ for any $n \in \mathbb{N}$. But then $n \cdot \ell(\sigma_2) \leq 3$ for all $n \in \mathbb{N}$, which is in obvious contradiction with $\ell(\sigma_2) > 0$. □

It is easy to build a state-labeled timed automaton [ACM97] with a single clock accepting L_{nonreg}. Note also that the untiming of this language $\mathcal{U}(L_{nonreg})$ is a regular (untimed) language.

3.5 Stuttering-free concatenation

Theorem 2.4.6 seems to provide a disappointing result concerning the possibility to have some results on syntactic monoids for real-time languages. But it this is not the case: we just have to shift our attention to other monoidal structures on the set of signals.

A "quasimonoidal" structure on $\mathsf{Sig}(\Sigma)$ arises if we consider a *partial* concatenation operation $\odot$ as follows: for each *nonempty* signal $\sigma : [0, \alpha[\to \Sigma$ ($\alpha \neq 0$) we define the *last* symbol occurring in σ as $\mathsf{last}(\sigma) = \lim_{t \nearrow \alpha} \sigma(t)$. Alternatively, $\mathsf{last}(\sigma)$ is the last letter in $\mathcal{U}(\sigma)$. The partial operation, called **stuttering-free concatenation**, is defined as follows: for each $\sigma, \tau \in \mathsf{Sig}(\Sigma) \setminus \{\sigma_\epsilon\}$

$$\sigma \odot \tau = \begin{cases} \sigma \cdot \tau & \text{, iff } \mathsf{last}(\sigma) = \tau(0) \\ \text{undefined} & \text{, iff } \mathsf{last}(\sigma) \neq \tau(0) \end{cases} \tag{3.13}$$

Further, for any $\sigma \in \mathsf{Sig}(\Sigma)$, put $\sigma_\epsilon \odot \sigma = \sigma = \sigma \odot \sigma_\epsilon$

We may easily extend this operation to a total one, by augmenting $\mathsf{Sig}(\Sigma)$ with a fresh symbol $\uparrow$, standing for "undefined", which becomes a "zero element":

$$\forall \sigma \in \mathsf{Sig}(\Sigma),\ \uparrow \odot \sigma = \sigma \odot \uparrow\ = \uparrow$$

Of course then instead of having $\sigma \odot \tau =$ undefined we put $\sigma \odot \tau = \uparrow$.

Proposition 3.5.1. $(\mathsf{Sig}^{\uparrow}(\Sigma), \odot, \sigma_\epsilon)$ *is a monoid, where* $\mathsf{Sig}^{\uparrow}(\Sigma) = \mathsf{Sig}(\Sigma) \cup \{\uparrow\}$.

Hence the whole theory of regularity from Chapter 2 applies. It should be noted that, though we have augmented $\mathsf{Sig}(\Sigma)$ with the "undefined" element we may still define regular subsets of $\mathsf{Sig}(\Sigma)$ as those that are regular in $\mathsf{Sig}^{\uparrow}(\Sigma)$. The question is whether in this case we will not get again "uninteresting" regular languages. We will show that this is not the case by relating $\mathsf{Sig}^{\uparrow}(\Sigma)$-regular languages with languages accepted by real-time automata.

3.5.1 Syntactic monoids for stuttering-free concatenation and real-time automata

In this subsection we prove that real-time regular languages can be characterized by inverse monoid morphisms whose domain is $\mathsf{Sig}^{\uparrow}(\Sigma)$. Unfortunately the mechanism of finite monoid recognizability does not give in general *finite representations* of the generated class of timed languages. This means that, by inverse monoid morphisms, we may obtain languages in which the timing information might not necessarily consists of time intervals, like in real-time regular languages.

Take, for example, the language $L_{dirichlet} \subseteq \mathsf{Sig}(\{a\})$ consisting of signals whose length is a rational number

$$L_{dirichlet} = \{a^t \mid t \in \mathbb{Q}_{\geq 0}\}$$

This language can be given as the inverse morphic image of the subset $\{e, a\}$ in the monoid $M_3 = \{e, a, \uparrow\}$ in which $aa = \uparrow$, under the morphism

$$\phi : \mathsf{Sig}^{\uparrow}(\Sigma) \to M_3,$$
$$\phi(a^t) = \begin{cases} a & \text{iff } t \in \mathbb{Q}_{>0} \\ \uparrow & \text{iff } t \notin \mathbb{Q}_{>0} \end{cases}$$
$$\phi(\uparrow) = \uparrow$$

because

$$\phi(a^t \odot a^{t'}) = \phi(\uparrow) = \phi(a^t)\phi(a^{t'})$$

We interpret this as the fact that monoid recognizability and finite generation of timed languages are "orthogonal" properties[1].

To cope with this problem, we will utilize here **extended RTAs** here, which are tuples $\mathcal{A} = (Q, \Sigma, \delta, Q_0, Q_f)$ like RTAs, but in which the only constraint on δ is that it gives a finite set of tuples $\delta \subseteq Q \times \mathcal{P}(\mathbb{R}_{\geq 0}) \times Q$, that is, each tuple (q, X, r) might contain *any* subset of reals $X \subseteq \mathbb{R}_{\geq 0}$. For this class of automata, all the closure results, including complementation, hold. The only property that is not valid is their decidability.

Theorem 3.5.2. *Given some RTA $\mathcal{A}$, $L(\mathcal{A})$ is a $\mathsf{Sig}^{\uparrow}(\Sigma)$-regular language. The following reverse also holds: for each $Reg(\mathsf{Sig}^{\uparrow}(\Sigma))$-regular language L, $L \setminus \{\uparrow\}$ is accepted by some extended RTA.*

Proof. We will utilize here transition-labeled RTA.

Assume $\mathcal{B}$ is some t-RTA accepting L, with $\sigma_\epsilon \notin L$. Using the Theorems 3.3.3 and 3.3.4, we get a deterministic t-RTA $\mathcal{C} = (Q, \Sigma, \theta, q_0, Q_f)$ with the same language as $\mathcal{B}$. Then define the $\mathsf{Sig}^{\uparrow}(\Sigma)$-automaton $\mathcal{A} = (Q \cup \{\uparrow\}, \mathsf{Sig}^{\uparrow}(\Sigma), \delta, q_0, Q_f)$ as follows: for each constant signal a^t, with $a \in \Sigma$ and $t \in \mathbb{R}_{\geq 0}$, define

[1] Observe that the algebraic characterization in [BPT01] also is insensitive to finite presentation of each set in the finite decomposition of ${\mathbb{R}_{\geq 0}}^n$.

$$\delta(q,a^t)=\begin{cases} r & \text{iff } \exists X\subseteq\mathbb{R}_{>0} \text{ s.t. } (q,a,X,r)\in\theta \text{ and } t\in X \\ \uparrow & \text{otherwise}\end{cases}$$

This definition makes δ a *partial function*, i.e., correctly defined: we cannot have $r_1,r_2\in Q$ with $(q,a,X_1,r_1),(q,a,X_2,q_2)\in\theta$ for some X_1,X_2, both containing t since we would contradict transition determinism in $\mathcal{B}$.

Then extend δ on *all* signals using again the decomposition property 2.11: for each $\sigma=a_1^{t_1}\odot\ldots\odot a_n^{t_n}$ we put $\delta(q,\sigma)=\delta\big(\delta(\ldots\delta(q,a_1^{t_1}),\ldots),a_n^{t_n}\big)$. Finally put $\delta(q,\sigma_\epsilon)=q$ and $\delta(q,\uparrow)=\uparrow$. By now δ is a total function.

The equality $L(\mathcal{A})=L(\mathcal{C})$ follows from the deterministic character of $\mathcal{C}$. The left-to-right inclusion $L(\mathcal{A})\subseteq L(\mathcal{C})$ is straightforward. For the right-to-left inclusion, observe that, by determinism of $\mathcal{C}$ we have that $\sigma\in L(\mathcal{C})$ iff there exists a *unique* accepting run $\big((q_{i-1},a_i,X_i,q_i)\big)_{i\in[1\ldots n]}$ associated with σ. As an outcome of stuttering freeness we have $a_i\neq a_{i+1}$ for all $i\in[1\ldots n-1]$. This implies that for the (unique!) decomposition $\sigma=a_1^{t_1}\odot\ldots\odot a_n^{t_n}$, we must have $t_i\in X_i$ and $\delta(q_{i-1},a_i^{t_i})=q_i$ and therefore

$$\delta(q_0,\sigma)=\delta(q_0,a_1^{t_1}\odot\ldots\odot a_n^{t_n})=\delta\big(\delta(\ldots\delta(q_0,a_1^{t_1}),\ldots),a_n^{t_n}\big)=q_n\in Q_f \tag{3.14}$$

and hence $\sigma\in L(\mathcal{A})$.

For the reverse implication, take some set $L\in Reg(\mathsf{Sig}^{\uparrow}(\Sigma))$, hence there exists some $\mathsf{Sig}^{\uparrow}(\Sigma)$-automaton $\mathcal{A}=(Q,\mathsf{Sig}^{\uparrow}(\Sigma),\delta,q_0,Q_f)$ such that $L=L(\mathcal{A})$.

For each $q,r\in Q$ and $a\in\Sigma$ define $X(a,q,r)\subseteq\mathbb{R}_{>0}$ as the set of lengths of a-signals which lead from q to r:

$$X(a,q,r)=\{t\mid\delta(q,a^t)=r\}$$

Define the (extended) t-RTA $\mathcal{B}=(Q\times\Sigma\cup\{q_0\},\theta,q_0,Q_f\times\Sigma)$ where

$$\begin{aligned}\theta \;=\; & \big\{\big((q,a),b,X(b,q,r),(r,b)\big)\mid a,b\in\Sigma,a\neq b,q,r\in Q\big\}\cup \\ & \cup\big\{\big(q_0,a,X(a,q_0,q),(q,a)\big)\mid a\in\Sigma,q\in Q\big\}\end{aligned}$$

(If $\sigma_\epsilon\in L$ then just add q_0 to the set of final states). Note that in $\mathcal{B}$ no two transitions with the same Σ-label are consecutive, i.e. $\mathcal{B}$ is a stuttering-free extended t-RTA.

The equality $L(\mathcal{B})=L(\mathcal{A})$ follows by the decomposition property 2.11 and the stuttering-freeness of $\mathcal{A}$ which assures that, when a signal is associated with some run in $\mathcal{A}$, the decomposition points that witness this must be exactly the discontinuity points within the signal. □

Hence $\mathsf{Sig}^{\uparrow}(\Sigma)$-regular languages are "more interesting" than $\mathsf{Sig}(\Sigma)$-regular languages, since the class of languages accepted by RTA contains nontrivial examples with timing information.

At the end of this chapter we will give a simple property which argues our view of monoid recognizability being orthogonal to finite generation.

Denote first $Sig_{eff}(\Sigma)$ the class of signals whose discontinuities occur only at rational points and whose endpoints are rational too. We call a $\mathsf{Sig}^{\uparrow}(\Sigma)$-automaton **effective** if there exists some

algorithm for deciding, for each signal $\sigma \in Sig_{\mathit{eff}}(\Sigma)$ and states q, r, whether $\delta(q, \sigma) = r$. On the other hand, we call a RTA **effective** if for each time label X in $\mathcal{A}$, the set $X \cap \mathbb{Q}_{\geq 0}$ is a recursive set.

Proposition 3.5.3. *The translations provided by Theorem 3.5.2 are such that effective* $\mathsf{Sig}^{\uparrow}(\Sigma)$*-automata are translated into effective deterministic RTA and vice-versa.*

Proof. The first implication is straightforward, since the algorithm for deciding whether $p \in X$, for each time label X and rational p, is a particularization of the algorithm provided by the given effective $\mathsf{Sig}^{\uparrow}(\Sigma)$-automaton. For the reverse we will consider the following algorithm: for each signal $\sigma \in Sig_{\mathit{eff}}(\Sigma)$ and pair of states $q, r \in Q$, consider all the paths in the RTA which lead from q to r, whose number of transitions equals the number of symbols of $\mathcal{U}(\sigma)$ (the untiming of σ) and are such that the i-th transition is labeled with the i-th symbol in $\mathcal{U}(\sigma)$. This set is finite for each signal due to stuttering-freeness of the RTA. Then, for each such run, using the algorithm provided by the given RTA, check whether the length of the i-th constant component of the signal is in the time label of the i-th transition within the run. □

We have no answer to the questions whether the other constructions in this chapter (concatenation, star closure, complementation) preserve effectiveness.

4. Timed automata

This chapter gives an outlook of semantics of timed automata [AD94] and of the clock regular expressions [BP99, BP01] that can be associated with them. We remind the Kleene theorem which connects them, and provide an alternative proof of this theorem for regular expressions that use indexed concatenations, theorem first proved in [BP99]. We also remind an alternative semantics for timed automata, semantics which makes reference to *reset points* rather than clock values, like in [BJLWY98]. The clock valuation semantics and the reset clock semantics are interchangeable, but we will see in the next chapters that the latter works better for timed regular expressions.

The chapter runs as follows: the first section presents clock valuations and clock constraints. In the second we remind the semantics of timed automata as timed transition systems, and show how this semantics can be transformed into a compositional one, such that clock regular expressions be equivalent to timed automata We also present here the alternative proof of the Kleene theorem for clock regular expressions with indexed concatenation, proof which is based upon the possibility to define classical regular expressions with indexed concatenation. The final section presents the reset time semantics for timed automata.

4.1 Clocks and clock constraints

Throughout this and the subsequent chapters, Ξ will denote the countable set of symbols $\Xi = \{x_1, x_2, \dots\}$ while Ξ_n will denote the first n symbols from Ξ, $\Xi_n = \{x_1, \dots, x_n\}$. We name the symbols from Ξ as **clocks** as they will be used to remember the time passage in the class of automata under study here. From these symbols we construct logical formulas which will be used to express constraints on clocks values which are to be satisfied at different moments while processing the signal.

An **atomic clock constraint** over Ξ_n is a formula of the following type:

- $x_i \in U$, for some $i \in [n]$ and nonnegative interval $U \in \mathbb{N}Int$;
- $x_i - x_j \in U$ for some $i, j \in [n], i \neq j$ and interval $U \in \mathbb{Z}Int$.

Observe that we allow also comparisons of clocks w.r.t. *negative* intervals.

An **elementary clock constraint** over Ξ_n is conjunction of the form

$$\bigwedge_{i=1}^{n}(x_i \in U_i) \wedge \bigwedge_{i,j\in[1\dots n],i\neq j} (x_i - x_j \in U_{ij}) \tag{4.1}$$

where each conjunct is an atomic clock constraint. A **clock constraint** over Ξ_n is any boolean combination of atomic clock constraints. The set of clock constraints over the set of clocks Ξ_n is denoted $\mathcal{C}(\Xi_n)$, while the set of elementary clock constraints over Ξ_n is denoted $\mathcal{EC}(\Xi_n)$.

A **clock valuation** is a function $v : \Xi \to \mathbb{R}_{\geq 0}$. Usually we are interested only in the values associated with the first n clocks, that is, in the restriction $v|_{\Xi_n} : \Xi_n \to \mathbb{R}_{\geq 0}$, which we denote v too.

Clock valuations can be extended to *interpretations of clock constraints* in the well-known way:

- First, each atomic clock constraint $x_i \in U$ is interpreted by "replacing" the clock x_i with its value $v(x_i)$ and then computing the truth value of the resulting formula, where the symbol $\in$ is interpreted as membership.
- Then the boolean operations are applied to the resulting truth values to get the truth value of the whole interpreted formula.

We denote $v \models C$ if the interpretation of C induced by v is the truth value "true".

We will identify a clock valuation $v : \Xi_n \to \mathbb{R}_{\geq 0}$ with an n-dimensional point $v \in \mathbb{R}^n_{\geq 0}$. Therefore we may import different operations on n-dimensional points to clock valuations. The two operations we use in the sequel are

1. Addition with a nonnegative integer: given $v \in \mathbb{R}^n_{\geq 0}$ and $t \in \mathbb{R}_{\geq 0}$, we denote $v + t$ the clock valuation defined by $(v + t)(x_n) = v(x_n) + t$.
2. Resetting the set of clocks in $X \subseteq [1 \dots n]$, or, equivalently, projection onto a subspace S of $\mathbb{R}_{\geq 0}$ defined by the equations $S = \{x_i = 0 \mid i \in X\}$: given $v \in \mathbb{R}^n_{\geq 0}$, we denote $v[X := 0]$ the clock valuation given by

$$v[X := 0](x_i) = \begin{cases} 0 & \text{iff } i \in X \\ v(x_i) & \text{otherwise} \end{cases}$$

4.2 Timed automata and their clock valuation semantics

In the sequel we fix a set of symbols Σ and a positive integer $n \in \mathbb{N}$.

Definition 4.2.1. *A **timed automaton** with n clocks is a tuple $\mathcal{A} = (Q, \delta, \lambda, Q_0, Q_f)$ where, Q denotes the (finite) set of states, δ denotes the transition relation*

$$\delta \subseteq Q \times \mathcal{EC}(\Xi_n) \times \mathcal{P}(\Xi_n) \times Q \text{ with } card(\delta) < \infty$$

λ denotes the state labeling function $\lambda : Q \to \Sigma$, and $Q_0, Q_f \subseteq Q$ are the sets of initial, resp. final states.

The classical way to give semantics to each timed automaton is to build a *timed transition system* first from the specified automaton, then to consider the set of *accepting runs* in this transition system, and finally to concatenate the labels of all transitions in each such run in order to get

the set of signals accepted by the given timed automaton. Hence, given a timed automaton $\mathcal{A} = (Q, \delta, \lambda, Q_0, Q_f)$, we associate a transition system whose set of configurations is the uncountable set of tuples (q, v) comprising a state $q \in Q$ and a clock valuation $v \in \mathbb{R}^n_{\geq 0}$ and whose transitions are classified as *instantaneous* or *timed*, as they are produced either by a transition in δ or by the passage of time while resting in a state q. Formally, the *timed transition system* associated with $\mathcal{A}$ is $\mathcal{T}(\mathcal{A}) = (Q \times \mathbb{R}^n_{\geq 0}, \theta, Q_0 \times \{\mathbf{0}_n\}, Q_f \times \mathbb{R}^n_{\geq 0})$ where

$$\theta = \big\{ \big((q, v), \varepsilon, (q', v')\big) \mid \exists (q, C, X, q') \in \delta \text{ such that } v \models C \text{ and } v' = v[X := 0] \big\} \cup \tag{4.2}$$

$$\cup \big\{ \big((q, v), a^t, (q, v + t)\big) \mid v \in \mathbb{R}^n_{\geq 0}, a = \lambda(q) \big\} \tag{4.3}$$

We call transitions of the form 4.2 as *instantaneous transitions* while those of the form 4.3 are called *timed transitions*.

In this transition system, the set of **runs** is the set of sequences $\big((q_{i-1}, v_{i-1}), z_i, (q_i, v_i)\big)_{i \in [1..k]}$ with $\big((q_{i-1}, v_{i-1}), z_i, (q_i, v_i)\big) \in \theta$ for all $i \in [1..k]$. An **accepting run** is a run in which $q_0 \in Q_0$, $v_0 = \mathbf{0}_n$ and $q_k \in Q_f$. The **language accepted by** $\mathcal{A}$ is then the set of concatenations of labels of transitions of each accepting run:

$$L(\mathcal{A}) = \big\{ z_1 \cdot z_2 \cdot \ldots \cdot z_k \mid \big((q_{i-1}, v_{i-1}), z_i, (q_i, v_i)\big)_{i \in [1..k]} \text{ is an accepting run } \big\}$$

We also say that the signal $z_1 \cdot \ldots \cdot z_k$ *is associated with* the run $\big((q_{i-1}, v_{i-1}), z_i, (q_i, v_i)\big)_{i \in [1..k]}$.

This semantics has the drawback of being unstructured and hence noncompositional. To make it compositional, we observe first that we may consider only runs in which instantaneous and timed transitions from $\mathcal{T}(\mathcal{A})$ alternate. More formally:

1. Suppose that in some run we have two consecutive instantaneous transitions, $\big(\_, \varepsilon, (q', v')\big)$ and $\big((q', v'), \varepsilon, \_\big)$. Then insert $((q, v), a^0, (q, v))$, where a is an arbitrary letter, in between them. Do this for all such consecutive occurrences.
2. Suppose now we are given a run in which two timed transitions $\big((q, v), a^t, (q, v + t)\big)$ and $\big((q, v+t), b^{t'}, (q, (v+t)+t')\big)$ are consecutive. Since λ is a function, we necessarily have $a = b$. Then replace these two transitions with a single timed transition $\big((q, v), a^{t+t'}, (q, v + (t + t'))\big)$.

We may therefore join together timed transitions with instantaneous transitions and "forget" the in-between configuration. Hence, we transform the transition system into the following: $\mathcal{T}'(\mathcal{A}) = (Q \times \mathbb{R}^n_{\geq 0}, \theta', Q_0 \times \{\mathbf{0}_n\}, Q_f \times \mathbb{R}^n_{\geq 0})$ where

$$\theta' = \big\{ \big((q, v), a^t, (q', v')\big) \mid \exists (q, C, X, q') \in \delta \text{ such that } v + t \models C, v' = v[X := 0] \text{ and } \lambda(q) = a \big\}$$

For such a transition system, a run is defined as a sequence $\big((q_{i-1}, v_{i-1}), a_i^{t_i}, (q_i, v_i)\big)_{i \in [1..k]}$ in which $\big((q_{i-1}, v_{i-1}), a_i^{t_i}, (q_i, v_i)\big) \in \theta'$ for all $i \in [1..k]$. Accepting runs have the same defining requirements as in the timed transition system $\mathcal{T}(\mathcal{A})$, and the language accepted by $\mathcal{A}$ still consists of the concatenation of the labels on each accepting run, that is,

$$L(\mathcal{A}) = \left\{ a_1^{t_1} \cdot \ldots \cdot a_k{}^{t_k} \mid \left((q_{i-1}, v_{i-1}), a_i^{t_i}, (q_i, v_i)\right)_{i\in[1..k]} \text{ is an accepting run in } \mathcal{T}'(\mathcal{A}) \right\}$$

Further we may split θ' into a union of $card(\delta)$ sets, one set for each tuple $(q, C, X, q') \in \delta$:

$$\theta(q, X, C, q') = \left\{ \left((q, v), a^t, (q', v')\right) \mid v + t \models C, v' = v[X := 0] \text{ and } \lambda(q) = a \right\}$$

Then we "classify" accepting runs according to the sequences of tuples of transitions in δ which are employed. That is, we define a *δ-sequence* as a sequence of tuples $\rho = \left(q_{i-1}, C_i, Y_i, q_i\right)_{i\in[1..k]}$, each tuple belonging to δ. We also define the set of runs *subsumed* by the δ-sequence $\rho = \left(q_{i-1}, C_i, Y_i, q_i\right)_{i\in[1..k]}$ as follows:

$$S(\rho) = \left\{ \left((q_{i-1}, v_{i-1}), a_i^{t_i}, (q_i, v_i)\right)_{i\in[1..k]} \mid \text{for each } i \in [1..k], \right.$$
$$\left. \left((q_{i-1}, v_{i-1}), a_i^{t_i}, (q_i, v_i)\right) \in \theta(q_{i-1}, C_i, Y_i, q_i) \right\}$$

This set naturally provides a set of signals which are *associated* to ρ:

$$L(\rho) = \left\{ a_1^{t_1} \cdot \ldots \cdot a_k^{t_k} \mid \left((q_{i-1}, v_{i-1}), a_i^{t_i}, (q_i, v_i)\right)_{i\in[1..k]} \text{ is a run subsumed by } \rho \right\} \tag{4.4}$$

In order to define accepting δ-sequences we must observe that these have to insure that all the subsumed runs must begin with the zero clock valuation $\mathbf{0}_n \in \mathbb{R}^n_{\geq 0}$. Therefore an accepting δ-sequence must not only start in an initial state and end in a final state, but also contain an initial tuple whose constraint imposes that all clocks are zero.

Formally, we define an *accepting δ-sequence* as a sequence $\rho = \left(q_{i-1}, C_i, Y_i, q_i\right)_{i\in[1..k]}$ in which $\left(q_{i-1}, C_i, Y_i, q_i\right)_{i\in[2..k]}$ is a δ-sequence, $q_0 = q_1 \in Q_0$, $C_1 = \bigwedge_{i=1}^{n}(x_i = 0)$, $Y_1 = \emptyset$, and $q_k \in Q_f$. This amounts to adding all transitions $(q_0, \bigwedge_{i=1}^{n}(x_i = 0), \emptyset, q_0)$ to θ' and requiring that all accepting runs start with one of these.

As a consequence, we get that

$$L(\mathcal{A}) = \bigcup \left\{ L(\rho) \mid \rho \text{ is an accepting } \delta\text{-sequence in } \mathcal{A} \right\}$$

Let us now observe that we have, in some sense, already separated an abstract level, in which runs have exactly the classical meaning as sequences of transitions in an automaton, and a "semantic" level, in which each abstract run is interpreted as some set of signals.

A new step consists of hiding away from states: note that, when building the set $L(\rho)$, the information regarding states is used only for retrieving the symbols that compose the associated signal. Hence we may build *abstract runs* $\overline{\rho} = (a_i, C_i, Y_i)_{i\in[1..k]}$ for which, if we consistently add states into tuples, we get a δ-sequence. And further build the set $S(\overline{\rho})$ of runs in $\mathcal{T}'(\mathcal{A})$ which are subsumed by $\overline{\rho}$ and the set $L(\overline{\rho})$ of timed words associated with $\overline{\rho}$. This is nothing but the spirit of the Kleene theorem.

More formally, an **abstract run** is a sequence $\overline{\rho} = (a_i, C_i, Y_i)_{i\in[1..k]}$ for which there exists a sequence of states $(q_i)_{i\in[0..k]}$ such that $\big(q_{i-1}, C_i, Y_i, q_i\big)_{i\in[1..k]}$ is a δ-sequence and for each $i \in [1..n-1]$, $\lambda(q_{i-1}) = a_i$. The two sets $S(\overline{\rho})$ and $L(\overline{\rho})$ are then built as follows:

$$\begin{aligned} S(\overline{\rho}) =& \big\{ \big((v_{i-1}, a_i^{t_i}, v_i)\big)_{i\in[1..k]} \mid \exists (q_i)_{i\in[0..k]} \text{ such that} \\ & \qquad \big((q_{i-1}, v_{i-1}), a_i^{t_i}, (q_i, v_i)\big) \in \theta(q_{i-1}, C_i, Y_i, q_i) \text{ for all } i \in [1\ldots k] \big\} \\ L(\overline{\rho}) =& \big\{ a_1^{t_1} \ldots \cdot a_k^{t_k} \mid \big((v_{i-1}, a_i^{t_i}, v_i)\big)_{i\in[1..k]} \in S(\overline{\rho}) \big\} \end{aligned}$$

Again it is easy to see that

$$L(\mathcal{A}) = \bigcup \big\{ L(\overline{\rho}) \mid \rho \text{ is an accepting abstract run in } \mathcal{A} \big\}$$

Moreover, the Kleene theorem for finite automata assures us that the set of abstract accepting runs can be generated by some regular expression over atoms of the type $(a, C, X) \subseteq \Sigma \times \mathcal{EC}(\Xi_n) \times \mathcal{P}(\Xi_n)$.

Now we observe that the sets $S(\overline{\rho})$ can be subject of a *concatenation* operation by matching on the clock valuation "in the middle". That is, given two abstract runs $\overline{\rho} = (a_i, C_i, Y_i)_{i\in[1..k]}$ and $\overline{\rho}' = (b_i, C_i', Y_i')_{i\in[1..k']}$,

$$\begin{aligned} S(\overline{\rho}) \cdot S(\overline{\rho}') = \big\{ \big((v_{i-1}, c_i^{t_i}, v_i)\big)_{i\in[1..k+k']} \mid & \big\{ \big((v_{i-1}, c_i^{t_i}, v_i)\big)_{i\in[1..k]} \in S(\overline{\rho}) \text{ and} \\ & \big((v_{i-1}, c_i^{t_i}, v_i)\big)_{i\in[k+1..k+k']} \in S(\overline{\rho}') \big\} \end{aligned}$$

That is, we have that $c_i = a_i$ for $i \in [1..k]$ and $c_i = b_{i-k}$ for $i \in [k+1..k+k']$.

And the final observation is that the intermediary clock valuations in each run belonging to some $S(\overline{\rho})$, are useless, both for concatenation purposes and when constructing the set $L(\overline{\rho})$ of signals associated with $\overline{\rho}$. We mean that we may consider only tuples (v, σ, v') consisting of a signal $\sigma \in \mathsf{Sig}(\Sigma)$ and two clock valuations $v, v' \in \mathbb{R}^n_{\geq 0}$, tuples which are called **signals with clock valuations**. Then the set $S(\overline{\rho})$ may be replaced by the following:

$$\begin{aligned} \overline{S}(\overline{\rho}) = \big\{ (v, a_1^{t_1} \cdot \ldots \cdot a_k^{t_k}, v') \mid & \exists (v_i)_{i\in[0..k]}, \exists (q_i)_{i\in[0..k]} \text{ such that } v_0 = v, v_k = v' \text{ and} \\ & \big((q_{i-1}, v_{i-1}), a_i^{t_i}, (q_i, v_i)\big) \in \theta(q_{i-1}, C_i, Y_i, q_i) \big\} \end{aligned}$$

while the set $L(\overline{\rho})$ could be described as:

$$L(\overline{\rho}) = \big\{ a_1^{t_1} \cdot \ldots \cdot a_k^{t_k} \mid (v, a_1^{t_1} \cdot \ldots \cdot a_k^{t_k}, v') \in S(\overline{\rho}) \big\}$$

Clearly, the concatenation operation on sets $S(\overline{\rho})$ could be easily "adapted" to the sets $\overline{S}(\overline{\rho})$.

We may summarize the above not completely formal discussion as follows: we define first the set of **signals with clock valuations**

$$\mathsf{Sigclk}(\Sigma) = \big\{ (v, \sigma, v') \mid \sigma \in \mathsf{Sig}(\Sigma), v, v' \in \mathbb{R}^n_{\geq 0} \big\}$$

We then define a partial operation of concatenation on $\mathsf{Sigclk}(\Sigma)$ as follows: for all pairs of signals with clock valuations $(v_1, \sigma_1, v_1'), (v_2, \sigma_2, v_2') \in \mathsf{Sigclk}(\Sigma)$,

$$(v_1, \sigma_1, v_1') \cdot (v_2, \sigma_2, v_2') = \begin{cases} (v_1, \sigma_1 \cdot \sigma_2, v_2') & \text{iff } v_1' = v_2 \\ \uparrow & \text{otherwise} \end{cases}$$

We further extend this partial operation to a *total operation* on subsets of $\mathsf{Sigclk}(\Sigma)$ by putting, for each $S_1, S_2 \subseteq \mathsf{Sigclk}(\Sigma)$,

$$S_1 \cdot S_2 = \big\{\alpha \cdot \beta \mid \alpha \in S_1, \beta \in S_2 \text{ and } \alpha \cdot \beta \neq \uparrow\big\}$$

whose unit is the set

$$S_\varepsilon = \big\{(v, \varepsilon, v) \mid v \in \mathbb{R}^n_{\geq 0}\big\}.$$

By the usual least fipoint construction, we then get the *star* operation: for each $S \in \mathsf{Sig}(\Sigma)$,

$$S^* = \bigcup_{n \in \mathbb{N}} S^n$$

where $S^0 = S_\varepsilon$ and $S^{n+1} = S^n \cdot S$ for all $n \in \mathbb{N}$.

Definition 4.2.2. *The set of n-**clocked regular expressions** as the language generated by the following grammar:*

$$E ::= (a, C, X) \mid E + E \mid E \cdot E \mid E^* \mid \varepsilon \mid 0 \tag{4.5}$$

where $C \in \mathcal{EC}(\Xi_n)$, $a \in \Sigma$ and $X \subseteq \Xi_n$.

We denote the set of n-clocked regular expressions as $\mathsf{CReg}_n(\Sigma)$.

The **semantics** of n-clocked regular expressions is an application $\|\cdot\| : \mathsf{CReg}_n(\Sigma) \to \mathcal{P}(\mathsf{Sigclk}(\Sigma))$ inductively defined as follows:

$$\begin{aligned} \|0\| &= \emptyset \\ \|\varepsilon\| &= S_\varepsilon \\ \|(a, C, X)\| &= \big\{(v, a^t, v') \mid v + t \models C \text{ and } v' = (v + t)[X := 0]\big\} \\ \|E_1 + E_2\| &= \|E_1\| \cup \|E_2\| \\ \|E_1 \cdot E_2\| &= \|E_1\| \cdot \|E_2\| \\ \|E^*\| &= \|E\|^* \end{aligned}$$

Besides this, each n-clocked regular expression is endowed with an *abstract semantics*, which is the classical semantics as a set of words over $\mathcal{EC}(\Xi_n) \times \Sigma \times \mathcal{P}(\Xi_n)$. We denote this abstract semantics as $|\cdot| : \mathsf{CReg}_n(\Sigma) \to \big(\mathcal{EC}(\Xi_n) \times \Sigma \times \mathcal{P}(\Xi_n)\big)^*$.

The following property, similar to the Proposition 3.1.4 and relates the abstract semantic of a regular expression to its semantics in terms of signals with clock valuations:

Proposition 4.2.3. *For each n-clocked regular expression E,*

$$\|E\| = \left\{ \|\omega\| \mid \omega \in |E| \right\}$$

in which $|E|$ is the abstract semantics of E.

Proof. By easy structural induction over the n-clocked regular expression.

The following straightforward property shows how to associate signals with clock valuations to δ-sequences:

Proposition 4.2.4. *Given some timed automaton $\mathcal{A} = (Q, \delta, \lambda, Q_0, Q_f)$ consider some δ-sequence $\left(q_{i-1}, C_i, X_i, q_i\right)_{i \in [1 \ldots k]}$ and denote $E(\rho)$ the following n-clocked regular expression:*

$$E(\rho) = (a, \bigwedge_{i=1}^{n} x_i = 0, \emptyset), (\lambda(q_1), C_1, X_1) \cdot \ldots \cdot (\lambda(q_{k-1}), C_{k-1}, X_{k-1}) \tag{4.6}$$

Then

$$L(\rho) = \|E(\rho)\|$$

in which $L(\rho)$ is the language associated with a run, defined in Identity 4.4 above.

Define also the family $\mathsf{TRec}_n(\Sigma)$ as the family of timed languages which are accepted by some timed automaton and $\mathsf{TRat}_n(\Sigma)$ as the family of timed languages which are the semantics of n-clocked regular expressions of the form $(a, \bigwedge_{i=1}^{n} x_i = 0, \emptyset) \cdot E$ where $E \in \mathsf{CReg}_n(\Sigma)$.

Theorem 4.2.5 (Kleene theorem for timed automata, [BP01]). *The classes of timed languages $\mathsf{TRec}_n(\Sigma)$ and $\mathsf{TRat}_n(\Sigma)$ are equal, and the equality is effective.*

Proof. Corollary of the classical Kleene theorem and the Lemma 4.2.3.

4.2.1 A Kleene theorem with indexed concatenation

In [BP99], another Kleene theorem is presented in the framework of *indexed* concatenations and stars. There is a natural question concerning the connections between this result and the above Kleene theorem 4.2.5. We show here that these results are intimately related and it is still the classical Kleene theorem which can be put at the basis of both. This proof can be seen as a rearrangement of the proof in [BP99].

In the cited paper, the semantics of timed automata is given in terms of *constrained generators*: a **constrained generator** is a pair $(\mathcal{G}, \Lambda)$ consisting of two mappings $\mathcal{G} : \mathbb{R}^n_{\geq 0} \to \mathcal{P}(\mathsf{Sig}(\Sigma))$ and $\Lambda : \mathbb{R}^n_{\geq 0} \times \mathsf{Sig}(\Sigma) \to \mathcal{P}(\mathbb{R}^n_{\geq 0})$, with the further requirement that for each $u \in \mathsf{Sig}(\Sigma)$ and $v \in \mathbb{R}^n_{\geq 0}$, $u \in \mathcal{G}(v)$ iff $\Lambda(v, u) \neq \emptyset$.

The aim is to associate, to each sequence of transitions[1], τ, a pair $\left(\mathcal{G}, \Lambda\right)$ that gives the following information:

[1] Actually, to each *set of sequences*, see the definition of n-clocked regular expressions with indexed concatenation.

- For each $v \in \mathbb{R}^n_{\geq 0}$, $\mathcal{G}(v)$ gives the set of signals that describe the behavior of the timed automaton through the sequence of transitions τ, if the automaton starts with the clock valuation v and τ can be performed starting with v.
- For each $v \in \mathbb{R}^n_{\geq 0}$ and $\sigma \in \mathsf{Sig}(\Sigma)$, $\Lambda(v, \sigma)$ gives the possible clock valuations in which the timed automaton might arrive after performing the sequence of transitions τ, provided it starts with the clock valuation v and is able to "parse" the signal σ.

More specifically, with each clock constraint $C \in \mathcal{EC}(\Xi_n)$ and symbol $a \in \Sigma$, the following atomic constrained generators is associated:

$$\mathcal{G}_{(a,C)}(v) = \{a^t \mid v + t \models C\}$$
$$\Lambda_{(a,C)}(v, a^t) = \{v + t \mid v + t \models C\}$$

The idea is then to build regular expressions over such atoms, and the full expressivity is acquired only if concatenation might *reset* some clocks. This feature is brought in by defining *indexed concatenations* as follows: given two constrained generators $(\mathcal{G}_1, \Lambda_1)$ and $(\mathcal{G}_2, \Lambda_2)$, and some subset $X \subseteq \Xi_n$, the *X-indexed concatenation* of $(\mathcal{G}_1\Lambda_1)$ with $(\mathcal{G}_2, \Lambda_2)$ is the constrained generator denoted as $(\mathcal{G}, \Lambda) = (\mathcal{G}_1, \Lambda_1) \odot_X (\mathcal{G}_2, \Lambda_2)$ and defined as:

$$\mathcal{G}(v) = \{\sigma_1 \cdot \sigma_2 \mid \sigma_1 \in \mathcal{G}_1(v) \text{ and there exists some } v' \in \Lambda_1(v, \sigma_1) \text{ such that } \sigma_2 \in \mathcal{G}_2(v'[X := 0])\}$$
$$\Lambda(v, \sigma) = \bigcup \{\Lambda_2(v'[X := 0], \sigma_2) \mid \sigma = \sigma_1 \cdot \sigma_2 \text{ for some } \sigma_1 \in \mathcal{G}_1(v) \text{ and } v' \in \Lambda_1(v, \sigma_1)\}$$

Each of these indexed concatenations induce naturally an *indexed iteration*, denoted $(\cdot)^{\oplus_X}$ and defined as follows:

$$(\mathcal{G}, \Lambda)^{\oplus_X} = \bigcup_{i \geq 1} (\mathcal{G}, \Lambda)^{i \odot_X}$$

where

$$(\mathcal{G}, \Lambda)^{1 \odot_X} = (\mathcal{G}, \Lambda)$$
$$(\mathcal{G}, \Lambda)^{(i+1) \odot_X} = (\mathcal{G}, \Lambda)^{i \odot_X} \odot_X (\mathcal{G}, \Lambda).$$

Observe that $(\mathcal{G}, \Lambda)^{\oplus X}$ does not "contain" the zero iteration.

Let us also denote $(\mathcal{G}_\varepsilon, \Lambda_X)$ the following constrained generator:

$$\mathcal{G}_\varepsilon(v) = \{\varepsilon\} \tag{4.7}$$
$$\Lambda_X(v, \varepsilon) = \{v[X := 0]\} \tag{4.8}$$

The set of n-**clocked regular expressions with indexed concatenation** is the set $\mathsf{IReg}(\Sigma)$ defined by the following grammar:

$$E ::= 0 \mid \varepsilon \mid (a, C) \mid E + E \mid E \odot_X E \mid E^{\oplus_X}$$

where $C \in \mathcal{EC}(\Xi_n)$, $a \in \Sigma$ and $X \subseteq \Xi_n$.

Their semantics is given by the following rules:

$$\begin{aligned} \|0\| &= \emptyset & \|E_1 + E_2\| &= \|E_1\| \cup \|E_2\| \\ \|\varepsilon\| &= \{(\mathcal{G}_\varepsilon, \Lambda_X)\} & \|E_1 \odot_X E_2\| &= \|E_1\| \odot_X \|E_2\| \\ \|(a, C)\| &= \{(\mathcal{G}_{(a,C)}, \Lambda_{(a,C)})\} & \|E^{\oplus X}\| &= \|E\|^{\oplus X} \end{aligned}$$

We will show here that there exists a straightforward bidirectional translation between clocked regular expressions with indexed concatenation and n-clocked regular expressions, translation which works by regarding constrained generators as sets of signals with clock valuations and vice-versa. This translation relies on a simple property of untimed languages which deals with indexed concatenations, property which we will state and prove here. In other words, we show that the Kleene theorem from [BP99] is a corollary of the Kleene theorem for finite automata too.

In order to relate the constrained generators semantics with the signals with clock valuations semantics, let us consider, for each $X \subseteq \Xi_n$, the atomic n-clock expression

$$\varepsilon_X = (a, \bigwedge_{i=1}^{n} x_i = 0, X)$$

whose semantics is

$$\|\varepsilon_X\| = \{(v, \varepsilon, v[X := 0]) \mid v \in \mathbb{R}^n_{\geq 0}\}$$

Then each n-clocked atom (a, C, X) can be decomposed as follows:

$$\|(a, C, X)\| = \|(a, C, \emptyset)\| \cdot \|\varepsilon_X\|$$

The next observation to be made is that, for any $C \in \mathcal{EC}(\Xi_n)$ and $a \in \Sigma$, $\|(a, C, \emptyset)\|$ is the *graph* of the function[2] $\Lambda_{(a,C)} : \mathbb{R}^n_{\geq 0} \times \mathsf{Sig} \to \mathcal{P}(\mathbb{R}^n_{\geq 0})$ from the constrained generator $(\mathcal{G}_{(a,C)}, \Lambda_{(a,C)})$ – that is, the set $\{(v, \sigma, v') \mid \Lambda(v, \sigma) = v'\}$. Moreover, $\mathcal{G}_{(a,C)}$ gives the "domain" of $\Lambda_{(a,C)}$, that is, the set of tuples $(v, \sigma) \in \mathbb{R}^n_{\geq 0} \times \mathsf{Sig}(\Sigma)$ for which $\Lambda(v, \sigma) \neq \emptyset$. This observation can be easily generalized as follows:

> Each set of signals with clock valuations $S \subseteq \mathsf{Sigclk}(\Sigma)$ is the graph of the second component of a constrained generator $(\mathcal{G}, \Lambda)$.

In particular, the sets ε_X are the graphs of the constrained generator $(\mathcal{G}_\varepsilon, \Lambda_X)$ defined in the Identities 4.7 and 4.8.

Two other important observations to be made are that concatenation of subsets of $\mathsf{Sigclk}(\Sigma)$ corresponds to the $\emptyset$-indexed concatenation of constrained generators, and that nonemptyset-indexed concatenation of constrained generators can be reduced to $\emptyset$-indexed concatenation by the aid of the constrained generators $(\mathcal{G}_\varepsilon, \Lambda_X)$:

[2] In fact, this is even a partial function with values in $\mathbb{R}^n_{\geq 0}$.

Proposition 4.2.6. *Given two sets of signals with clock valuations S_1 and S_2 which are the graphs of the constrained generators $(\mathcal{G}_1, \Lambda_1)$, resp. $(\mathcal{G}_2, \Lambda_2)$, the set $S = S_1 \cdot S_2$ is the graph of the constrained generator $(\mathcal{G}_1, \Lambda_1) \odot_\emptyset (\mathcal{G}_2, \Lambda_2)$.*

Moreover,

$$(\mathcal{G}_1, \Lambda_1) \odot_X (\mathcal{G}_2, \Lambda_2) = (\mathcal{G}_1, \Lambda_1) \odot_\emptyset (\mathcal{G}_\varepsilon, \Lambda_X) \odot_\emptyset (\mathcal{G}_2, \Lambda_2)$$

Proof. By straightforward verification.

By now, we may state the following:

Lemma 4.2.7. *Given $k+1$ constraints $C_i \in \mathcal{EC}(\Xi_n)$ ($i \in [0..k]$), $k+1$ symbols $a_i \in \Sigma$ ($i \in [0..k]$) and k subsets $Y_i \subseteq \Xi_n$ ($i \in [1..k]$), consider the n-clocked regular expression (without indexed concatenation)*

$$E = (a_1, C_1, \emptyset) \cdot \varepsilon_{Y_1} \cdot (a_2, C_2, \emptyset) \cdot \ldots \cdot \varepsilon_{Y_k} \cdot (a_{k+1}, C_{k+1}, \emptyset)$$

and the n-clocked regular expression with indexed concatenation

$$F = (a_1, C_1) \odot_{Y_1} \ldots \odot_{Y_k} (a_{k+1}, C_{k+1})$$

Then $\|E\|$ is the graph of the second component of $\|F\|$.

Proof. By induction on k, using Proposition 4.2.6 for the induction step.

To complete the claimed connection we introduce regular expressions with indexed concatenations for untimed languages and prove a simple property concerning the translation from expressions with indexed concatenations into classical regular expressions:

Definition 4.2.8. *The set of **regular expressions** over Σ with **indexed concatenations** from Ω is defined as follows:*

$$E ::= 0 \mid \varepsilon \mid a \mid E + E \mid E \odot_x E \mid E^{\oplus_x}$$

where $a \in \Sigma$ and $x \in \Omega \cup \{\varepsilon\}$.

The semantics of these expressions is in terms of languages over $(\Sigma \cup \Omega)^*$ as follows:

$$\begin{aligned} |0| &= \emptyset & |E_1 + E_2| &= |E_1| \cup |E_2| \\ |\varepsilon| &= \{\varepsilon\} & |E_1 \odot_x E_2| &= |E_1| \cdot \{x\} \cdot |E_2| \\ |a| &= \{a\} & |E^{\oplus_x}| &= \big(|E| \cdot \{x\}\big)^* \cdot |E| \end{aligned}$$

Lemma 4.2.9. *The set of languages which are the semantics of a regular expression with indexed concatenation from Ω equals the set of regular languages over $\Sigma \cup \Omega$.*

Proof. The direct inclusion is a straightforward consequence of the semantics of regular expressions with indexed concatenation.

The inverse inclusion follows by induction upon the structure of the (classical) regular expression: the base cases 0, ε and $a \in \Sigma$ are trivial, while for $x \in \Omega$ we consider the expression $\varepsilon \odot_x \varepsilon$.

For the induction step, suppose that we have two classical regular expressions E_1 and E_2 and that we have built regular expressions with indexed concatenation $\overline{E}_1$ and $\overline{E}_2$ such that $|E_1| = |\overline{E}_1|$ and $|E_2| = |\overline{E}_2|$. Then

$$\begin{aligned} |E_1 + E_2| &= |\overline{E}_1 + \overline{E}_2| \\ |E_1 \cdot E_2| &= |\overline{E}_1 \odot_\varepsilon \overline{E}_2| \\ |E_1^*| &= |\varepsilon + \overline{E}_1^{\oplus_\varepsilon}| \end{aligned}$$

□

As a corollary we have

Theorem 4.2.10. *The classes* $\mathsf{TRec}(\Sigma)$ *and* $\mathsf{TRat}(\Sigma)$ *are equal to the class of timed languages which are the semantics of some clocked regular expression with indexed concatenation of the form* $\left(a, \bigwedge_{i=1}^{n} x_i = 0\right) \cdot E$*, where* $E \in \mathsf{IReg}_n(\Sigma)$.

Proof. This is a corollary of Proposition 4.2.6 and Lemma 4.2.9: for the direct inclusion, we transform each n-clocked regular expression E into a regular expression over $\tilde{\Sigma} = \mathcal{EC}(\Xi_n) \times \Sigma$ with indexed concatenation over $\Omega = \left\{\varepsilon_X \mid X \subseteq \Xi_n\right\}$, denote it $\tilde{E}$. At this point we introduce the *timed* semantics of such a regular expression as the union of the timed semantics of all the words in its abstract semantics over $(\tilde{\Sigma} \cup \Omega)^*$.

Now, we only have to replace operations of the form $\odot_{\varepsilon_X}$ with $\odot_X$ and $\oplus_{\varepsilon_X}$ with $\oplus_X$ and observe that, for any clocked regular expression with indexed concatenation E',

$$\|E'\| = \bigcup \left\{\|W\| \mid W \in |E'|\right\}$$

where $|E'|$ is the semantics of E' as a regular expression over $\tilde{\Sigma} \cup \Omega$. This property, corroborated with Proposition 4.2.6, assures us that the timed semantics of $\tilde{E}$ (with the replacements of $\odot_{\varepsilon_X}$ with $\odot_X$) equals the timed semantics of E.

The reverse inclusion follows by the same argument. □

4.3 Reset time semantics for timed automata

In this section we will show another semantics that can be given to timed automata, semantics originally proposed in [BJLWY98]. The idea is to record the reset time for each clock, and the current time. In other words, we only make a change of variables, from clock values to reset times, change of variables defined as follows: for each clock $x_i \in \Xi_n$,

$$v(x_i) = t - r(x_i) \text{ where } t \text{ is the "current time point".}$$

Though this semantics is almost the same as the clock valuation semantics, it has certain features that will help us develop our theory concerning reachability. The regular expressions we utilize here are n-clocked regular expressions defined in 4.5, we will only provide a different semantics for them in terms of *signals with reset times*.

Definition 4.3.1. *A **signal with reset times** is a tuple* $(t_1, \dots, t_n, t, \sigma, t'_1, \dots, t'_n, t')$ *where* $\sigma \in \mathsf{Sig}$ *and* $t_i, t'_i, t, t' \in \mathbb{R}_{\geq 0}$ *for each* $i \in [1 \dots n]$.

The intuition is that the real t_i represents some moment before a chain of transitions when the clock x_i was reset, t is the "initial" moment, t' is the moment when the last transition in the chain is taken, and t'_i represents the last reset time for the clock x_i before the moment t'. The set of signals with reset times is denoted $\mathsf{Sigreset}(\Sigma)$.

Similarly to signals with clock valuations, signals with reset times can be concatenated if and only if the intermediary time points match. More formally, given two signals with reset times $\xi = (t_1, \dots, t_n, t, \sigma, t'_1, \dots, t'_n, t')$ and $\xi' = (u_1, \dots, u_n, u, \sigma', u'_1, \dots, u'_n, u')$, the concatenation $\xi'' = \xi \cdot \xi'$ is defined as follows:

$$\xi'' = \begin{cases} (t_1, \dots, t_n, t, \sigma \cdot \sigma', u'_1, \dots, u'_n, u') & \text{iff for all } i \in [1..n], t'_i = u_i \text{ and } t' = u \\ \uparrow & \text{otherwise} \end{cases}$$

This concatenation operation is extended, as usual, to *sets* of signals with reset times: for each pair of sets $S_1, S_2 \subseteq \mathsf{Sigreset}(\Sigma)$,

$$S_1 \cdot S_2 = \{\xi_1 \cdot \xi_2 \mid \xi_1 \in S_1, \xi_2 \in S_2 \text{ and } \xi_1 \cdot \xi_2 \neq \uparrow\} \tag{4.9}$$

which is a total operation on $\mathsf{Sigreset}(\Sigma)$ whose unit is the set of signals with reset times

$$S_\varepsilon = \{(t_1, \dots, t_n, t, \varepsilon, t_1, \dots, t_n, t) \mid t, t_i \in \mathbb{R}_{\geq 0}\}$$

Again as usual, concatenation on sets gives rise to the star operation

$$S^* = \bigcup_{i \in \mathbb{N}} S^i$$

where $S^0 = S_\varepsilon$ and $S^{i+1} = S^i \cdot S$ for all $i \in \mathbb{N}$.

The configurations of the timed transition system for the reset time semantics are tuples comprising a state and $n+1$ positive numbers, the first n representing the reset time for each clock and the last recording the current moment. That is, the timed transition system is the tuple $\mathcal{T} = \left(Q \times {\mathbb{R}_{\geq 0}}^{n+1}, \theta, Q_0 \times \mathbf{0}_{n+1}, Q_f \times {\mathbb{R}_{\geq 0}}^{n+1}\right)$ where:

$$\begin{aligned} \theta = \{&((q, t_1, \dots, t_n, t), a^{t'}, (q', t'_1, \dots, t'_n, t'')) \mid \exists (q, C, X, q') \in \delta \text{ such that} \\ &\lambda(q) = a, t'' = t + t', t'_i = t'' \text{ for all } i \in X, t'_i = t_i \text{ for all } i \notin X \text{ and } v \models C \\ &\text{where } v \text{ is the clock valuation defined by } v(x_i) = t'' - t_i \text{ for all } i \in [1..n]\} \end{aligned}$$

By an argument similar to the one given in section 4.2.1 we may transform this semantics into a compositional one by giving reset time semantics to each δ-sequence in the timed automaton. This compositional reset time semantics is built using the following basic rule:

$$\begin{aligned}\|(a,C,X)\| = \big\{((t_1,\ldots,t_n,t),a^{t'},(t'_1,\ldots,t'_n,t'')) \mid\ & t''=t+t', t'_i=t'' \text{ for all } i\in X,\\ & t'_i=t_i \text{ for all } i\notin X \text{ and } v\models C \text{ where } v \text{ is the clock valuation defined by}\\ & v(x_i)=t''-t_i \text{ for all } i\in[1..n]\big\}\end{aligned} \tag{4.10}$$

and the "compositionality" rule 4.9. Hence each δ-sequence $\rho=(q_{i-1},C_i,Y_i,q_i)_{i\in[1..k]}$ in $\mathcal{A}$ gives rise to the following word over $\Sigma\times\mathcal{EC}(\Xi_n)\times\mathcal{P}(\Xi_n)$

$$w(\rho)=(a_1,C_1,X_1)\cdot\ldots\cdot(a_k,C_k,X_k) \text{ where } a_i=\lambda(q_{i-1})$$

Then the language of the given timed automaton is:

$$L(\mathcal{A})=\bigcup\big\{\big\|w\big((\varepsilon,\bigwedge_{i=1}^{n}x_i=0,\emptyset)\cdot\rho\big)\big\| \mid \rho \text{ is an accepting run in } \mathcal{A}\big\}$$

The semantics of n-clock regular expressions may be given similarly by the usual rules which allow commuting semantics with union, concatenation and star, that is

$$\begin{aligned}\|E_1+E_2\| &= \|E_1\|\cup\|E_2\|\\ \|E_1\cdot E_2\| &= \|E_1\|\cdot\|E_2\|\\ \|E^*\| &= \|E\|^*\end{aligned}$$

provided that the atoms have the semantics given in 4.10 above and the expressions 0 and ε have the following semantics:

$$\|0\|=\emptyset,\quad \|\varepsilon\|=S_\varepsilon$$

5. Timed regular expressions

In this section we investigate the possibility to define some timed regular expressions that do not use clocks and clock constraints. The reason for searching such expressions is, at a first sight, esthetic, since the clocked regular expressions are harder to write. But this reason hides a more profound one: namely that, at the specification level, properties refer to (i.e., bind) state durations, or intervals separating two actions, or delays. Clock manipulation might be regarded as a "low-level" language, like automata, whereas regular expressions are intended to be a "high-level" language easy to handle.

There exists a "high-level" approach to regular expressions that has preceded the clocked regular expressions: it is the *timed regular expressions* of [ACM97]. These expressions do not use clocks, they only provide time binding by the use of some interval-indexed parentheses. For example, the timed regular expression $a\langle bc\rangle_1$ specifies the set of signals in which an a-state with an arbitrary duration is followed by a b-state and then by a c-state, the overall duration of the b and c states being equal to 1.

Though giving a neat specification language, timed regular expressions hide some mathematical problems, connected to the density of the set of real numbers. Namely, and contrary to classical regular expressions, they are not closed under intersection, and hence this operation must be put between the basic operations such that the generative power be reasonable. Even with intersection they still show less expressive power than timed automata, and another operation is needed then: renaming.

This chapter recalls these problems and discusses one possible solution to them. This solution is the use of *colored* parentheses. As simple it seems, this solution shows itself some hurdles: first, the language of "colored" and balanced parentheses is not a context-free language, hence it might raise difficult problems concerning parsing and translating. The solution we find is to consider a different concatenation operation, that allows two expressions with colored parentheses to concatenate on "matching" parentheses. But the algebraic bases for this interpretation must be laid, and the subsequent chapters are concerned with this task.

The chapter is more of a "hand-waiving" style, presenting more intuition and discussions than formalization. It runs as follows: the first section recalls the definition of timed regular expressions and their relationship to timed automata. We also show here some peculiarities of interpreting a timed regular expression without parentheses as a classical regular expression. In other words, we investigate succinctly the effect of the untiming morphism at the regular expression level. The second section presents an undecidability result concerning the extension of timed regular expressions

with negation. This is a rather expected result, however it does not follow from the undecidability of the universality problem for timed automata, due to the "incomplete" Kleene connection between timed automata and timed regular expressions. The third section is a short abstract of the results that connect timed regular expressions with timed automata. The last section discusses the problems and the possible solutions for the generalization of timed regular expressions with colored parentheses. This section is informal and will be developed in the following chapters.

5.1 Basic properties of timed regular expressions

Definition 5.1.1 ([ACM97]). *The set of **timed regular expressions** is given by the following grammar:*

$$E ::= 0 \mid \varepsilon \mid a \mid E + E \mid E \wedge E \mid E \cdot E \mid E^* \mid \langle E \rangle_I$$

where a is any symbol in Σ and I is any positive interval.

The semantics of timed regular expressions is, of course, in terms of signals. The idea is that the angle brackets bind the duration of signals:

$$\|a\| = \{a^t \mid t \in \mathbb{R}_{>0}\} \qquad \|E_1 + E_2\| = \|E_1\| + \|E_2\|$$
$$\|E_1 \wedge \|E_2\| = \|E_1\| \cap \|E_2\| \qquad \|E_1 \cdot E_2\| = \|E_1\| \cdot \|E_2\|$$
$$\|E^*\| = \|E\|^* \qquad \|\langle E \rangle\|_I = \{\sigma \in \|E\| \mid \ell(\sigma) \in I\}$$

There is an alternative way of generalizing from real-time regular expressions: namely allow atoms of the type $[A]_I$ for any set $A \subseteq \Sigma$. The semantics of such an atom would be the following:

$$[A]_I = \big\{\sigma \in \mathsf{Sig}(\Sigma) \mid \sigma(t) \in A \text{ for all } t \in dom(\sigma)\big\}$$

Of course, we need conjunction in both cases. Then we may replace, in an "inside-out" manner, each expression of the type $\langle E \rangle_I$ with $E \wedge [\Sigma]_I$.

5.1.1 Timed regular expressions without brackets

Timed regular expressions *without brackets* can be given also an untimed semantics, that is, in terms of words over Σ. We would expect that this semantics be related to the untiming morphism $\mathcal{U}$. More formally, if we denote $U(E)$ the classical regular expression which we associate to the timed regular expression[1] E, then we would like to have

$$|U(E)| = \mathcal{U}(\|E\|) \tag{5.1}$$

where $|U(E)|$ is the set of words defined by the classical regular expression $U(E)$.

[1] That is, $U(E) = E$ when E contains no parentheses

But there is a small problem: the semantics of the timed regular expression aa is equal to the semantics of the timed regular expression a, fact which does not hold when the two expressions are viewed as (classical) regular expressions. In other words, this "brute" transformation of timed regular expressions without brackets into classical regular expressions is not compatible with the untiming morphism:

$$|aa \cap a| = \emptyset \neq \{a\} = \mathcal{U}(\|aa\|) \cap \mathcal{U}(\|a\|)$$

We recall then that the untiming morphism is in fact a morphism whose target is the monoid of stuttering-free words, endowed with the stuttering-free concatenation. This implies that if we want Identity 5.1 to hold, we need to consider a different semantics for classical regular expressions: namely, to interpret each classical regular expression into elements of $SF(\Sigma)$ and to interpret regular expression concatenation as *stuttering-free concatenation*, see page 27. This is not a nice solution since it requires a reconsideration of the theory of finite automata and regular expressions for the special monoid $SF(\Sigma)$.

There is yet another solution which does not induce this reconsideration: recall that $SF(\Sigma)$ is also representable as the quotient $\Sigma^*/\tilde{\rho}$ (Chapter 2, page 27), where $\tilde{\rho}$ is generated by the relation $(aa, a) \in \rho$ for all $a \in \Sigma$. We may then consider the closure under $\tilde{\rho}$ for the semantics of the regular expression. In other words, the timed regular expression a, when interpreted as a classical regular expression, would have the semantics $\{a^n \mid n \in \mathbb{N}_{\geq 1}\} = |aa^*|$.

Syntactically, this can be done as follows: given a timed regular expression without braces, we replace each symbol a with the classical regular expression aa^*. Denote U this syntactic operation. Of course, a formal definition of $U(E)$ would be done by structural induction on the timed regular expression E. Hence U commutes with all operations – summation, conjunction, concatenation, star.

This solution implies a weaker version of Identity 5.1: remind that $\overbrace{\cdot}^{} : \Sigma^* \to SF(\Sigma)$ denotes the canonical projection induced by the congruence $\tilde{\rho}$. Its action consists of transforming each sequence of identical symbols into one symbol, e.g., $\overbrace{abbabcca}^{} = ababca$. Then

$$\overbrace{|U(E)|}^{} = \mathcal{U}(\|E\|) \tag{5.2}$$

One may think that this property also holds for the "brute" transformation $U(E) = E$. But this is not true, especially due to the use of *conjunction* in timed regular expressions:

$$\overbrace{|aa \wedge a|}^{} = \emptyset \neq \{a\} = \mathcal{U}(\|aa \wedge a\|)$$

The translation of timed regular expressions without brackets into classical regular expressions will be instrumental in Chapter 9.

5.2 Undecidability of the language emptiness problem for extended timed regular expressions

As for the case of classical regular expressions, we may extend the grammar by allowing the use of *negation*. The resulting expressions will be called *extended timed regular expressions,* by similarity with extended classical regular expressions which utilize negation. The generating grammar for this class of expressions is the following:

$$E ::= a \mid E + E \mid E \wedge E \mid E \cdot E \mid E^* \mid \langle E \rangle_I \mid \neg E$$

and the semantics for the negation is, naturally, based upon set complementation:

$$\|\neg E\| = \mathsf{Sig}(\Sigma) \setminus \|E\|$$

In this section we show that the emptiness problem for the semantics of extended timed regular expressions is undecidable. The technique we use is drawn from one of the undecidability results concerning Duration Calculus [ZCHS93], namely the undecidability of the fragment that allow $\ell = a$ formulas. We prove the result by showing that the halting problem for two-counter machines [HU92] is reducible to our emptiness problem.

We mention that this negative result does not follow from the undecidability of the universality problem for timed automata [AD94], since the Kleene theorem relating timed automata and timed regular expressions involves renaming.

A 2-counter machine $\mathcal{C} = (Q, q_0, T)$ consists of a set of locations Q, an initial location q_0 and a set of transitions T which are tuples (q_1, s, t, x, y, q_2) where $s, t \in \{? = 0, ? \neq 0\}$ and $x, y \in \{-1, 0, 1\}$. The meaning is the following:

- The machine works on two counters x and y which can hold arbitrarily large, nonnegative values, and which may be checked and/or modified by each transition.
- A transition in which s is '$? = 0$' is taken iff the first counter is zero, and similarly a transition in which t is '$? = 0$' is taken iff the second counter is zero.
- Taking a transition in which $x = +1$ increases the first counter by one. When $x = 0$ the first counter is not changed, while for $x = -1$ the first counter is decreased by one, if its value is positive, and is left unchanged otherwise. Similarly for the values of y and the second counter.

It is additionally required that the machine be *deterministic* in the following sense: for each state $q \in Q$ and preconditions $s, t \in \{? = 0, ? \neq 0\}$, at most one transition can be enabled in q by the preconditions s and t:

$$1 \geq card\big\{(q, s, t, x, y, q') \mid q' \in Q, x, y \in \{-1, 0, 1\}\big\}$$

We may see the states of the 2-counter machine as labels of the statements of a program containing test conditions over each counter and increments and/or decrements of each counter.

A *configuration* of a 2-counter machine is then a triple (q, x, y) containing a location and the values of each counter. A *run* of the 2-counter machine is a (finite or infinite) sequence of configurations connected by transitions which starts with both tapes holding 0.

The *halting problem* for 2-counter machines is the problem of whether a given 2-counter machine has a finite run. Here, the 2-counter machine is an input to the problem.

Theorem 5.2.1 ([HU92]). *The halting problem for 2-counter machines is undecidable.*

Theorem 5.2.2. *The language emptiness problem for extended timed regular expressions is undecidable.*

Proof. We will prove that the emptiness problem for extended timed regular expressions is *many-one reducible* (in the sense of [HU92]) to the halting problem for 2-counter machines:

Start with a 2-counter machine $\mathcal{C} = (Q, q_0, T)$ and suppose it has a finite run, $\big((q_i, m_i, n_i)\big)_{i\in[1\dots N]}$. We associate to this run a family of signals over the set of symbols $\Sigma = Q \cup \{a, b, c, d\}$ where $a, b, c, d \notin Q$. The association will be such that, for each signal in this family and each $i \in [1 \dots N]$, the interval $[i, i+1[$ of the signal consists of a first part in which the signal is constantly equal to q_i and then a sequence of $2m_i + 1$ discontinuities where the signal jumps from a to b and from b to a, and another sequence of $2n_i + 1$-discontinuities where the signal jumps from c to d and from d to c.

Formally, we say that the signal σ **encodes within the interval** $[k, k+1[$ the configuration (q, m, n) if

$$\sigma\big|_{[k,k+1[} = q^{t_0} a^{t_1} b^{t_2} \cdots a^{t_{2m+1}} b^{t_{2m+2}} c^{t_{2m+3}} d^{t_{2m+4}} \cdots c^{t_{2m+2n+3}} d^{t_{2m+2n+4}}$$

with $t_i > 0$ for all $i \in [0 \dots 2m + 2n + 4]$.

We then say that the signal σ **encodes the run** $\big(q_i, n_i, m_i\big)_{i\in[1\dots N]}$ if it encodes the configuration (q_i, m_i, n_i) within the interval $[i, i+1[$ for each $i \in [1 \dots N]$.

We aim at building an extended timed regular expression that accepts only signals which are associated with the run. This expression must therefore specify the initial configuration of the run and each of the transitions. The initial configuration is specified by an expression which says that the first interval of each signal encodes the first configuration of the 2-counter machine. Then, each transition is simulated by an extended timed regular expression which accepts some signal iff, whenever in some interval $[k, k+1[$ the signal encodes some configuration in which the transition is enabled, then in the subsequent interval $[k+1, k+2[$ the signal encodes the configuration which results by taking the respective transition. Then, the expression that simulates the run is the intersection of all these expressions.

The initial configuration is encoded by the extended timed regular expression

$$Init = (\langle q_0 \cdot a \cdot b \cdot c \cdot d \rangle_1) \cdot \Sigma$$

Then each transition $\tau = (q, s, t, x, y, r)$ is simulated by a regular expression which we will denote $tr(\tau)$ and build in the sequel. There can be $2 \times 2 \times 3 \times 3 = 36$ types of transitions: 2 due to

the two different modes of checking the contents of a tape and 3 due to different ways the contents of each tape can be updated.

We give as an example the expression that simulates a transition of the type $(q, ? = 0, ? \neq 0, +1, -1, r)$. Our aim is that any signal $\sigma \in \|tr(\tau)\|$ has the property that if

$$\sigma\big|_{[k,k+1[} = q^{t_0}a^{t_1}b^{t_2}c^{t_3}d^{t_4}\dots c^{t_{2n+1}}d^{t_{2n+2}}c^{t_{2n+3}}d^{t_{2n+4}}$$

with $n \geq 1$ then

$$\sigma\big|_{[k+1,k+2[} = r^{t_0}a^{u}b^{v}a^{w}b^{t_2}c^{t_3}d^{t_4}\dots c^{t_{2n+1}}d^{t_{2n+2}+t_{2n+3}+t_{2n+4}}$$

where $u + v + w = t_1$.

The expression $tr(\tau)$ is a conjunction of the following three subexpressions:

- the subexpression saying that if the signal encodes, within some interval $[k, k+1[$, a configuration in which τ is enabled then the interval $[k+1, k+2[$ of the signal starts with state r:

$$\neg\Big(\Sigma^* \cdot \langle q \cdot a \cdot b \cdot c \cdot d \cdot c \cdot \Sigma^*\rangle_1) \cdot (\Sigma \setminus \{r\}) \cdot \Sigma^*\Big)$$

- a conjunction of two expressions saying that if the interval $[k, k+1[$ encodes a configuration in which transition τ is enabled then in $[k+1, k+2[$ the state r is followed by states a, b, a and b in this order such that the length of the last b-state equals the length of the only b within the interval $[k, k+1[$ while the length of the aba-signal equal the length of the only a state in the interval $[k, k+1[$:

$$\begin{aligned}&\neg\Big(\Sigma^* \cdot q \cdot a \cdot (a+b) \cdot \langle b \cdot c \cdot d \cdot c \cdot d \cdot (c \cdot d)^* \cdot r \cdot (a+b)^*\rangle_1 \cdot (\Sigma \setminus \{b\}) \cdot \Sigma^*\Big) \wedge \\ &\neg\Big(\Sigma^* \cdot q \cdot a \cdot \langle b \cdot c \cdot d \cdot c \cdot d \cdot (c \cdot d)^* \cdot r \cdot \neg(a \cdot b \cdot a)\rangle_1) \cdot \Sigma^*\Big)\end{aligned}$$

- a conjunction of three expressions saying that if the interval $[k, k+1[$ encodes a configuration where transition τ can be taken and there are $n+1$ c-states and $n+1$ d-states within this interval then in the interval $[k+1, k+2[$ there have to be n states c and n states d such that the length of the i-th c-state within $[k+1, k+2[$ is equal to the length of the i-th c-state within $[k, k+1[$ for all $i \in [n]$, the length of the i-th d-state within $[k+1, k+2[$ is equal to the length of the i-th c-state within $[k, k+1[$ for all $i \in [n-1]$ and finally the length of the last d-state within $[k+1, k+2[$ equals the sum of the lengths of the $n+1$-th c-state, the n-th and the $n+1$-th d-state within $[k, k+1[$:

$$\begin{aligned}&\neg\Big(\Sigma^* \cdot q \cdot a \cdot b \cdot (c \cdot d)^* \cdot c \cdot \langle c \cdot d \cdot (c \cdot d)^* \cdot c \cdot d \cdot r \cdot (a+b+c+d)^*\rangle_1) \cdot (\Sigma \setminus \{c\}) \cdot \Sigma^*\Big) \wedge \\ &\neg\Big(\Sigma^* \cdot q \cdot a \cdot b \cdot (c \cdot d)^* \cdot c \cdot d \cdot \langle d \cdot (c \cdot d)^* \cdot c \cdot d \cdot r \cdot (a+b+c+d)^*\rangle_1) \cdot (\Sigma \setminus \{d\}) \cdot \Sigma^*\Big) \wedge \\ &\neg\Big(\Sigma^* \cdot q \cdot a \cdot b \cdot (c \cdot d)^* \cdot c \cdot d \cdot c \cdot \langle c \cdot d \cdot r \cdot (a+b+c+d)^* \cdot c\rangle_1) \cdot (\Sigma \setminus \{d\}) \cdot \Sigma^*\Big) \wedge \\ &\neg\Big(\Sigma^* \cdot q \cdot a \cdot b \cdot (c \cdot d)^* \cdot (c \cdot d)^2 \cdot \langle d \cdot r \cdot (a+b+c+d)^* \cdot c\rangle_1) \cdot (\Sigma \setminus \{d\}) \cdot \Sigma^*\Big)\end{aligned}$$

The specification of the remaining 35 types of transitions can be done similarly.

Then the (finite) run of the 2-counter machine, if it exists, is simulated by the following extended timed regular expression:

$$E_{\mathcal{C}} = Init \wedge \bigwedge_{\tau \in T} tr(\tau) \qquad \square$$

5.3 Relating timed regular expressions and timed automata

Timed regular expressions are a nice specification language, but they carry some expressivity problems. The following theorem and the discussion after it shows them:

Theorem 5.3.1 ([ACM97]). *The class of timed languages accepted by timed automata equals the class of timed languages accepted by timed regular expressions with intersection and renaming.*

Here renaming refers to signals: formally, given two sets Σ and Ω and a mapping $f : \Sigma \to \Omega$, this can be extended canonically to a monoid morphism $f^\flat \mathsf{Sig}(\Sigma) \to \mathsf{Sig}(\Omega)$. The morphism $f^\flat$ simply replaces symbols from Σ with symbols from Ω in each signal. For example, for the function $f : \{a, b\} \to \{c, d\}$ given by $f(a) = c$ and $f(b) = d$, $f^\flat(a^2 b^{1.5}) = c^2 d^{1.5}$. Renamings are not necessarily bijective, and it is this feature that is essential in the Kleene theorem relating timed regular expressions with timed automata.

The direct inclusion follows by showing first that automata with a single clock can be embedded into timed regular expressions without intersection, and then by decomposing a timed automaton with n clocks into n automata with a single clock, building the timed regular expressions for each timed automaton and intersecting the results. The timed regular expression for each one-clock automaton will specify the *runs* rather than the signals accepted by it, that is, the one-clock automaton is considered to work over signals whose symbols are exactly the states of the automaton. This is why, after building the intersection, one needs to apply the renaming that associates to each state in the timed automaton, its label.

The reverse inclusion follows by proving that the usual union/intersection/concatenation/star constructions can be generalized to timed automata.

It was also shown in [ACM97] that intersection is necessary for representing timed automata. Their example is the timed regular expression $a\langle bc\rangle_1 \wedge \langle ab\rangle_1 c$, which cannot be expressed without conjunction. The timed language accepted by this expression is:

$$L_0 = \{a^\alpha b^\beta c^\gamma \mid \alpha + \beta = 1, \beta + \gamma = 1\} \tag{5.3}$$

Moreover, in [Her99] it was shown that renaming also is necessary. An example of timed automaton whose language cannot be represented by regular expressions without renaming is presented in Figure 5.1 (modification from [Her99]).

The language of this automaton equals the renaming $x \mapsto b$ applied to the semantics of the timed regular expression $\big((xa)^*\langle b(ax)^*\rangle_1 \wedge \langle (xa)^* b\rangle_1 (ax)^*\big)$.

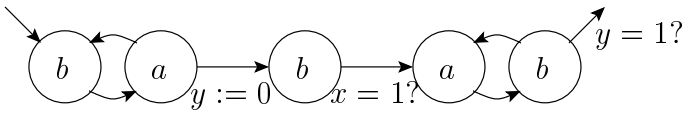


Fig. 5.1. A timed automaton that is not equivalent to any timed regular expression, even with intersection.

5.4 Colored parentheses: basic ideas and problems

Let us first start with an observation with a "language theoretical" flavor: timed regular languages can be redefined with *matching* parentheses, that is, by putting interval indices on each *left* parenthesis as well. Hence, instead of $\langle ab\rangle_1$ we would have $\langle_1 ab\rangle_1$, and a construction like $\langle_2 ab\rangle_1$ would not be a timed regular expression. What we would get this way is the *Dyck language* over the set

$$P = \left\{ `\langle_I`, `\rangle_I` \mid I \in \mathbb{Q}Int \right\}$$

The generating grammar will be almost the same:

$$E ::= a \mid E + E \mid E \wedge E \mid E \cdot E \mid E^* \mid \langle_I E\rangle_I$$

where $a \in \Sigma$ and $I \in \mathbb{Q}Int$. Let us use the notation Δ_Ω for the Dyck language over a (possibly infinite) set of symbols Ω.

The big problem with timed regular expressions is that they cannot be "interleaved", due to their context-freeness. But it is exactly interleaving what is necessary for specifying the language L_0 in 5.3 without intersection!

Our idea is to use *colored parentheses*: for example, L_0 would be specified by the following timed regular expression with colored parentheses:

$$\langle_1^{\text{blue}} a \, \langle_1^{\text{red}} b\rangle_1^{\text{blue}} \, c\rangle_1^{\text{red}}$$

But this idea poses some big language-theoretic problem: if we want to specify also cyclic behaviors, we fall into non-context-free specification languages! Consider just a cyclic behavior of the kind

$$\langle_1^{\text{blue}} a \, \langle_1^{\text{red}} b\rangle_1^{\text{blue}} \langle_1^{\text{blue}} a\rangle_1^{\text{red}} \langle_1^{\text{red}} b\rangle_1^{\text{blue}} \ldots$$

with arbitrarily many repetitions. Specifying the union of all such timed regular expressions is not easy: if we try

$$\left(\langle_1^{\text{blue}} a\rangle_1^{\text{red}} \langle_1^{\text{red}} b\rangle_1^{\text{blue}}\right)^* \tag{5.4}$$

then the first red parenthesis is a right parenthesis! This implies both syntactic and semantic problems:

- At the syntactic level, if we accept expressions like 5.4, then what to do with expressions like

$$\left(a\,\langle_I^{\text{red}} b\right)^*?$$

- If such expressions are also acceptable, then how to interpret them? Intuitively, such an expression resets a *potentially infinite number of clocks*, hence we might run into trouble with decidability.
- Even when such expressions are to be rejected – by some non-context-free rules – how to prove then a Kleene theorem? Its proof for the timed regular expressions of [ACM97] is essentially based upon the context-free presentation of expressions.

To put all the above considerations into a more formal and "language-theoretic" framework, we have to consider n (infinite!) sets of matching parentheses, each indexed with an interval:

$$P_i = \left\{ `\langle_I^i`, `\rangle_I^i` \mid I \in \mathbb{Q}Int \right\} \text{ for all } i \in [1 \ldots n]$$

We may then define n *deletion morphisms*, $(e_i)_{i\in[1\ldots n]}$, each e_i deleting all parentheses not in P_i: these morphisms are the canonical extensions of the following deletion functions:

$$p_i : \bigcup_{i=1}^{n} P_i \to P_i \cup \{\varepsilon\}, p_i(a) = \begin{cases} a & \text{iff } a \in P_i \\ \varepsilon & \text{otherwise} \end{cases}$$

$$e_i : \left(\bigcup_{i=1}^{n} P_i\right)^* \to (P_i)^*, e_i(a_1 \ldots a_n) = p_i(a_1) \ldots p_i(a_n)$$

Then define the language of *correctly matching parentheses* over $\bigcup_{i=1}^{n} P_i$,

$$L_{\text{par}} = \left\{ w \in \left(\bigcup_{i=1}^{n} P_i\right)^* \mid \text{ for each } i \in [1 \ldots n], e_i(w) \in \Delta_{P_i} \right\}$$

This language is unfortunately context-sensitive for $n \geq 2$: just consider the intersection of L_{par} with $\left(\langle_1^1\right)^* \left(\langle_1^2\right)^* \left(\rangle_1^1\right)^* \left(\rangle_1^2\right))^*$, which gives a language of the form $\{a^k b^l c^k d^l \mid k, l \in \mathbb{N}\}$ which is an easy prey to the Bar-Hillel (pumping) lemma for context-free languages [HU92].

Let us mention, at the end of these considerations, that the language L_{par} can be generated by matricial grammars [DP89], or with the so-called *contextual grammars with distributed catenation and shuffle* [KMM97].

5.4.1 Changing the concatenation

Here we come with the idea to use a different concatenation operation: the expression

$$\langle_1^{\text{blue}} a\, \langle_1^{\text{red}} b\rangle_1^{\text{blue}} \langle_1^{\text{blue}} a\rangle_1^{\text{red}} \langle_1^{\text{red}} b\rangle_1^{\text{blue}}\, a\rangle_1^{\text{red}}$$

could be represented by an expression like

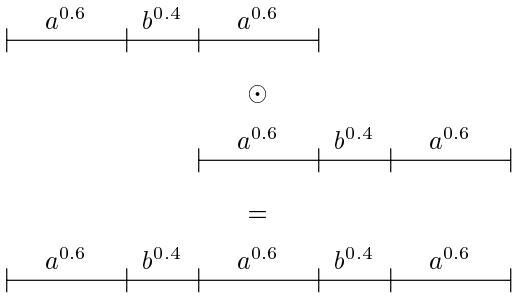


Fig. 5.2. An example of "overlapping" concatenation.

$$\langle_1^{\text{blue}} a \langle_1^{\text{red}} b \rangle_1^{\text{blue}} a \rangle_1^{\text{red}} \odot \langle_1^{\text{blue}} a \langle_1^{\text{red}} b \rangle_1^{\text{blue}} a \rangle_1^{\text{red}}$$

At the semantic level, concatenation would require two signals to *match* on their a-parts, as depicted in Figure 5.2:

The question is then how to identify the subsignals on the right and on the left that must match. Our idea is to use some distinguished points in each signal, like some markers for the moments when each parenthesis opens or closes. If we order these points such that the left point for the i-th color is t_i and the right point for the i-th color is t_{n+i} then we may represent the above concatenation like in Figure 5.3.

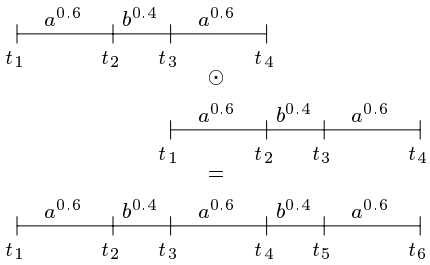


Fig. 5.3. An example of concatenation with distinguished points.

Of course, the result has more distinguished points than the operands, so we would need also a convention how to index them. But this very fact to increase the number of distinguished points creates some problems when trying to define star: namely we need to manage with unbounded numbers of points and to rearrange the indices after each concatenation etc.

The issue from this is to observe that, once the time point t_3 was concatenated, it has played its role, since the right parenthesis it represents has found a matching left parenthesis. Then we may simply forget it: the result of concatenation from Figure 5.3 would no longer have six distinguished points, but four. Hence we need two operations: a "juxtaposition" operation that "fuses" two signals with distinguished points along a certain subset of points, and a "projection" operation, that forgets the points that have "actively" participated to the concatenation.

Let us see now how this idea works for timed automata too.

5.4.2 The "overlapping" concatenation for timed automata

In the previous chapter we have introduced the reset time semantics which associates to each run in the timed automaton a signal with reset times $\xi = (t_1, \dots, t_n, t, \sigma, t'_1, \dots, t'_n, t')$ where t_i, t'_i, t, t' are nonnegative reals and $\sigma \in \mathsf{Sig}(\Sigma)$ is a signal. ξ can be concatenated to another signal with reset times $\xi' = (u_1, \dots, u_n, u, \sigma', u'_1, \dots, u'_n, u')$ iff the last $n+1$ components of the left operand match the first $n+1$ components of the right operand.

Observe first that, when ξ and ξ' above can be concatenated, the signals with reset times which can be obtained from ξ and ξ' by translating all the reals by some constant α can be concatenated too. That is, the following two signals with reset times can be concatenated:

$$\xi'' = (t_1 + \alpha, \dots, t_n + \alpha, t + \alpha, \sigma, t'_1 + \alpha, \dots, t'_n + \alpha, t' + \alpha)$$
$$\xi''' = (u_1 + \alpha, \dots, u_n + \alpha, u + \alpha, \sigma', u'_1 + \alpha, \dots, u'_n + \alpha, u' + \alpha)$$

This means that the only useful timing information in each signal with reset times is the set of differences between the components. We may then define an equivalence relation on $\mathsf{Sigreset}(\Sigma)$ which relates each pair of signals which "differ by a constant". Hence, $\xi_1 \simeq \xi_2$ if there exists $\alpha \in \mathbb{R}_{\geq 0}$ such that

$$\xi_1 = (t_1, \dots, t_n, t, \sigma, t'_1, \dots, t'_n, t')$$
$$\xi_2 = (t_1 + \alpha, \dots, t_n + \alpha, t + \alpha, \sigma, t'_1 + \alpha, \dots, t'_n + \alpha, t' + \alpha)$$

Though this equivalence is not a congruence, our observation above shows that it satisfies the following property:

If $\xi_1 \simeq \xi_2$ and $\xi_1 \cdot \xi_3 \neq \uparrow$ then there exists $\xi_4 \simeq \xi_3$ such that $\xi_2 \cdot \xi_4 \neq \uparrow$ and the reverse.

In other words, it is a bisimulation w.r.t. concatenation.

Hence in the equivalence class of ξ_1 and ξ_2 the timing information refers only to the *differences* between the reset points and therefore can be represented by an *antisymmetric matrix* $A \in \mathcal{M}_{2n+2}(\mathbb{R})$, (i.e., with $A_{ij} = -A_{ji}$ for all $i, j \in [1 \dots 2n+2]$) having the property that for each $i, j \in [1 \dots n]$:

$$\begin{aligned}
A_{ij} &= t_j - t_i & \qquad A_{n+i+1,n+j+1} &= t'_i - t'_j \\
A_{n+1,j} &= t_j - t & \qquad A_{2n+2,n+1+j} &= t'_j - t' \\
A_{i,n+1} &= t - t_i & \qquad A_{n+1+i,2n+2} &= t' - t'_i \\
A_{i,n+1+j} &= t'_j - t_i & \qquad A_{n+1,n+1+j} &= t'_j - t \\
A_{2n+2,j} &= t_j - t' & \qquad A_{n+1,2n+2} &= t' - t
\end{aligned}$$

This way a signal with reset times can be represented as a tuple $\xi = (A, \alpha, \sigma)$ consisting of an antisymmetric matrix $A \in \mathcal{M}_{2n+2}(\mathbb{R})$, a real number α which represents the "offset" of ξ w.r.t. some representative in its equivalence class, and a signal σ.

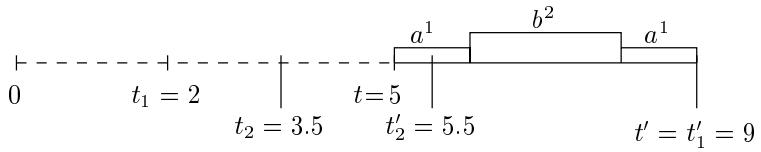


Fig. 5.4. A graphical representation of the signal with reset times$(2, 3.5, 5, a^1b^2a^1, 9, 5.5, 9)$.

On the other hand, signals with reset times may be graphically represented by putting all the timing and signal information on the time axis:

Observe that this representation is asymmetric: there is no information regarding the signal that has passed in between the first reset time t_1 and the "start of observation" t.

For a more "symmetric" graphical representation we might use objects like in the Figure 5.5.

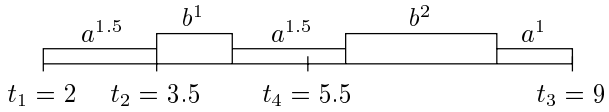


Fig. 5.5. A signal with "symmetric" reset time information.

We will call the $\simeq$ class of an object like in Figure 5.5 as an 4-*signal*. This presentation provides information about what happened since the first reset time in consideration: the piece of signal $\sigma_{14} = a^{1.5}b^1a^1$ tells the "history" between the time point $t_1 = 2$ and the time point $t_4 = 5.5$.

For a more algebraic setting, the 4-signal in Figure 5.5 can be represented as a 4×4 *matrix* whose (i, j) entry records the piece of signal in between the i-th and the j-th time point. In order to distinguish the case when $t_i > t_j$ from the case when $t_i < t_j$ we may employ *antisignals*, or signals in which time flows in the opposite direction, or, moreover, signals which are "read" in the reverse order. Intuitively, the antisignal corresponding to the signal $a^\alpha b^\beta$ should be $b^{-\beta}a^{-\alpha}$. The matrix representing the 4-signal in Figure 5.5 is the following:

$$A = \begin{pmatrix} \varepsilon & a^{1.5} & a^{1.5}b^1a^{1.5}b^2a^1 & a^{1.5}b^1a^1 \\ a^{-1.5} & \varepsilon & b^1a^{1.5}b^2a^1 & b^1a^1 \\ a^{-1}b^{-2}a^{-1.5}b^{-1}a^{-1.5} & a^{-1}b^{-2}a^{-1.5}b^{-1} & \varepsilon & a^{-1}b^{-2}a^{-1} \\ a^{-1}b^{-1}a^{-1.5} & a^{-1}b^{-1} & a^{0.5}b^2a^1 & \varepsilon \end{pmatrix}$$

Note that in the matrix A we have that, for each $i, j, k \in [1 \dots 8]$, $A_{ij}A_{jk} = A_{ik}$. We will extensively use this "triangle identity" in the subsequent chapters.

Then, if we generalize concatenation of signals with reset times to 4-signals we get exactly the overlapping concatenation we have defined in the previous subsection. A graphical representation of concatenation is given in Figure 5.6.

The necessary condition for the two 4-signals in Figure 5.6 to concatenate can be put also in terms of matrix representation: suppose that we have the following block-decomposition of the

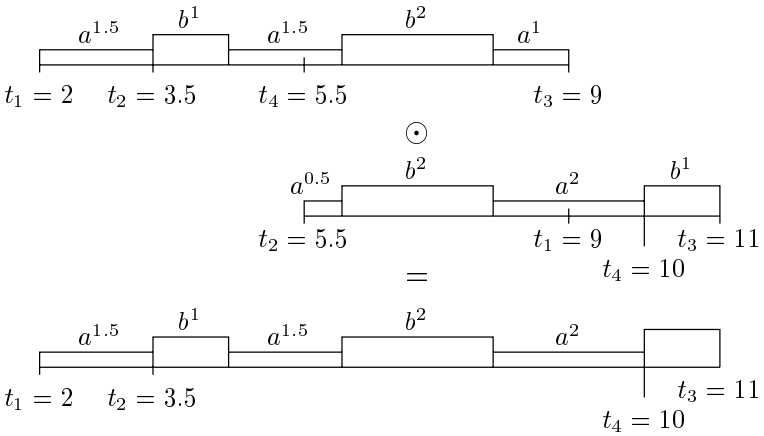


Fig. 5.6. Concatenation of two 4-signals.

matrices that represent each 4-signal in Figure 5.6:

$$A = \left(\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right), \qquad A' = \left(\begin{array}{c|c} A'_1 & A'_2 \\ \hline A'_3 & A'_4 \end{array}\right)$$

with $A_1, \ldots, A_4, A'_1, \ldots, A'_4$ being antisymmetric matrices in $\mathcal{M}_2(\mathbb{R}_{\geq 0})$.

Then A can be concatenated to A' iff $A_4 = A'_1$. The result will be the matrix

$$A'' = \left(\begin{array}{c|c} A_1 & B \\ \hline -B^t & A'_4 \end{array}\right)$$

where $B_{ij} = (A_2)_{ik}(A'_2)_{kj}$ for all $i, j \in [1 \ldots 4]$ and some $k \in [1 \ldots 4]$. Here B^t is the transpose of the matrix B.

The next chapter discusses this formalization, and in particular the regular expressions which work over $2n$-signals and their relationship to timed automata. Our choice for a matricial presentation of $2n$-signals, which could be thought as holding a lot of "redundant" information due to the triangle identity, comes not only from an aim to "algebraize" everything, but also because this presentation is closely related to a certain data structure which is aimed at representing timing information: the *Difference Bound Matrices* (DBMs) [Bel57]. We will take full advantage of the intimate relationship between DBMs and our matricial presentation of n-signals.

6. Matrices of signals

In this chapter we provide the basic algebraic properties of the overlapping concatenation. We base our definition of concatenation on two other operations: projection, which "forgets" certain rows and columns in a matrix, and juxtaposition, which "fuses" two matrices along a certain submatrix. This choice is again not only algebraic, but also emphasizes special properties of each of these two basic operations in different contexts.

We then define a class of regular expressions whose semantics is based on $2n$-signals (parity is needed for concatenation). The atoms of these expressions are matrices whose components are timed regular expressions – we call them $2n$-regsignals. These are in fact our algebraization of regular expressions with colored parentheses. We also show here that timed automata can be simulated by regular expressions over $2n$-signals.

We then make a first try to lift concatenation at the specification level, that is, we try to provide a compositional calculus with regular expressions, in order to be able to check whether a regular expression has a nonempty semantics. But we discover very quickly that no compositional concatenation operation can be defined on $2n$-regsignals, and the problem lies in the noncompositionality of projection. This means that we cannot have the wished calculus of emptiness for free. On the contrary, we show that juxtaposition can be lifted to a compositional operation on $2n$-regsignals.

We also discover an equally serious problem, namely that the emptiness problem for our regular expressions is undecidable. This result follows by encoding any instance of the Post Correspondence Problem into a regular expression of a very simple form: a star of a sum of $2n$-regsignals. The problem lies therefore in the untimed structure of the expressions, and we leave this problem for study in the next chapter.

This chapter is organized as follows: in the first section we present the definition of n-signals. They are in fact presented as a particular case of n-*dominoes*, which are matrices whose components are a mixture of signals and antisignals. In the second section we give the definition of the projection, juxtaposition and concatenation operations, and provide some algebraic properties relating them. In a short third section we present the notion of n-signal language and the more general notion of n-domino language and show that these languages form a Kleene algebra w.r.t. the concatenation inherited from $2n$-signals and the star operation which is induced by concatenation. The fourth section presents the notions of regsignals and regminoes, which are "compact", single-matricial representations of n-signal languages, respectively n-domino languages. We show here the possibility to define a compositional juxtaposition and the impossibility to define a compositional projection on regsignals (regminoes). We also define here the class of regular expressions

whose atoms are regsignals (resp. regminoes). In the fifth section we give the translation of timed automata semantics into regular expressions over regsignals, or, in other words, we give an n-signal-semantics of timed automata. And in the sixth and last section we give the undecidability result concerning the emptiness problem for regular expressions over regsignals.

6.1 n-dominoes and n-signals

As pointed out at the end of Chapter 5, matricial presentations of n-signals require working not only with signals, but also with *antisignals*, that is, sequences of the type $a_1^{-t_1}a_2^{-t_2}\dots a_k^{-t_k}$ where $t_1,\dots,t_k \in \mathbb{R}_{\geq 0}$. Intuitively, such a sequence denotes the history of states and their duration, and hence $a^{-t}\cdot a^t = \varepsilon$. Formally, we replace the set of signals $\mathsf{Sig}(\Sigma)$ (which is the coproduct of $card(\Sigma)$ copies of the monoid $(\mathbb{R}_{\geq 0},+,0)$) with the coproduct of $card(\Sigma)$ copies of the *group* $(\mathbb{R},+,0,-)$. We denote this coproduct group by $\mathsf{BiSig}(\Sigma)$. Of course, in $\mathsf{BiSig}(\Sigma)$ we would also have "words" containing positive and negative powers like a^2b^{-3} and such words are counterintuitive in the "timed world", but we are forced to use them as they naturally occur by concatenation of signals and antisignals. In algebraic terms, working with mixed words like a^2b^{-3} is a must since the union of the set of signals and the set of antisignals does not have a nice algebraic structure – it is not stable w.r.t. concatenation.

We denote the inverse operation in $\mathsf{BiSig}(\Sigma)$ as $(\cdot)^{-1}$. Hence $(a^t)^{-1} = a^{-t}$ and $\left(a^tb^u\right)^{-1} = b^{-u}a^{-t}$. For a timed language $L \subseteq \mathsf{Sig}(\Sigma)$ we denote L^{-1} the set of inverses of signals in L. The set of antisignals over Σ is denoted $\mathsf{Sig}(\Sigma^{-1})$ and does not contain, by definition, the empty signal σ_ϵ.

Definition 6.1.1. *An n-**domino** over Σ is a matrix $w = (w_{ij})_{i,j\in[1\dots n]}$ of elements from $\mathsf{BiSig}(\Sigma)$ with the following property:*

$$w_{ij}\cdot w_{jk} = w_{ik} \text{ for each } i,j,k \in [1\dots n] \tag{6.1}$$

*When $w_{ij} \in \mathsf{Sig}(\Sigma)\cup\mathsf{Sig}(\Sigma^{-1})$ for all $i,j\in[1\dots n]$ we say that w is an n-**signal** over Σ.*

Identity 6.1 will be referred to as the *triangle identity*.

We denote by $D_n(\Sigma)$ the class of n-dominoes over Σ and by $\mathsf{Sig}_n(\Sigma)$ the class of n-signals over Σ. Observe that in an n-domino w we have that for all $i,j\in[1\dots n]$, $w_{ii}=\varepsilon$ and $w_{ij}=w_{ji}^{-1}$.

The above definition does not faithfully formalize the drawings we have made in the previous chapter since in those drawings we have also associated a real number, t_i, to each entry in the matrix. But this difference is inessential since, given any real number α, we may associate it to the first index, that is, put $t_1=\alpha$, and then build the sequence of t_is by putting $t_i = t_1+\ell(w_{1i})$. By abusing notation, we will still draw the n-signals as in the first section, with the t_is in place.

Remark 6.1.2. For every n-signal there exists some ordering on $[1\dots n]$, say $i_1 \prec \dots \prec i_n$ (or, equivalently, a bijection $\varphi:[1\dots n]\to[1\dots n]$, with $\varphi(j)=i_j$) such that $w_{i_k,i_{k+1}} \in \mathsf{Sig}(\Sigma)$ for all $k\in[1\dots n-1]$. When $\prec$ satisfies this property, we say that it is an *ordering compatible with* w. The ordering is unique when $w_{ij}\neq\varepsilon$ for all $i\neq j$.

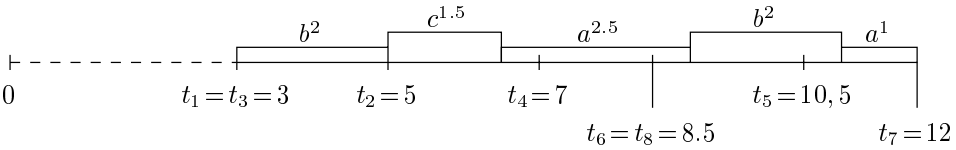


Fig. 6.1. An example of a 8-signal.

For example, for the 8-signal presented in Figure 6.1 we may have the ordering $1 \prec 3 \prec 2 \prec 4 \prec 6 \prec 8 \prec 5 \prec 7$, as it follows by reading the time points from left to right. Observe also that this ordering is not unique, the ordering $3 \prec 1 \prec 2 \prec 4 \prec 8 \prec 6 \prec 5 \prec 7$ being also compatible with the 8-signal in the figure.

Remark 6.1.3. Let us observe that, in order to give an $2n$-domino w we may specify only the first n components w_{ij} with $i, j \in [1 \dots n]$ and the "pseudodiagonal" components $w_{i,n+i}$ with $i \in [1 \dots n]$, or, similarly, the *last* n components w_{ij} with $i, j \in [n+1 \dots 2n]$ and the pseudodiagonal components. This follows since the remaining components can be defined by the aid of the triangle identity 6.1. For example, if we have specified the first n components and the pseudodiagonal components, the remaining components to be specified are w_{ij} with $(i,j) \in \big([n+1 \dots 2n] \times [n+1 \dots 2n]\big) \cup \big([n+1 \dots 2n] \times [1 \dots n]\big) \cup \big([1 \dots n] \times [n+1 \dots 2n]\big)$. These can be recovered as follows:

- For $i \in [1 \dots n]$ and $j \in [n+1 \dots 2n]$, $w_{ij} = w_{i,j-n} \cdot w_{j-n,j}$ and $w_{ji} = w_{ij}^{-1}$.
- For $i, j \in [n+1 \dots 2n]$, $w_{ij} = (w_{i,i-n})^{-1} \cdot w_{i-n,j-n} \cdot w_{j-n,j}$.

6.1.1 n-dominoes over a one-letter alphabet

The class of n-dominoes over a one-letter alphabet form a special class, due to the fact that any concatenation of a signal and an antisignal is a signal or an antisignal – Therefore, any n-domino is also an n-signal.

In fact, instead of working with n-signals we might employ n-tuples of reals – that is, instead of working with a n-signal $w \in \mathsf{Sig}_n(\Sigma)$, we might work with an n-tuple $(t_1, \dots, t_n) \in \mathbb{R}^n$ for which $w_{ij} = a^{t_j - t_i}$. This observation was already made at the end of last chapter.

Then the projection/juxtaposition/concatenation operations are straightforward operations on tuples:

1. Given an n-tuple $\overline{t} \in \mathbb{R}^n_{\geq 0}$ and $X \subseteq [1 \dots n]$, the X-**projection** of $\overline{t}$ is the tuple denoted $\overline{t}|_X$ with the property that
$$\overline{t}|_X = (t_{i_1}, \dots, t_{i_k})$$
where $X = \{i_1, \dots, i_k\}$, $i_j < i_{j+1}$.
2. Given $m, n, p \in \mathbb{N}$ with $p \leq \min(m, n)$, the p-**juxtaposition** of an m-tuple $\overline{t} = (t_1, \dots t_m) \in \mathbb{R}^m_{\geq 0}$ with an n-tuple $\overline{u} = (u_1, \dots, u_n) \in \mathbb{R}^n_{\geq 0}$ is defined iff $t_{m-p+i} = u_i$ for all $i \in [1 \dots p]$ and is the $(m+n-p)$-tuple $\overline{v} = (z_1, \dots, z_{m+n-p})$ with

$$v_i = \begin{cases} t_i & \text{for } i \in [1 \dots m] \\ u_i & \text{for } i \in [m+1 \dots m+n-p] \end{cases}$$

We denote $\overline{z} = \overline{x} \square_p \overline{y}$.

3. Given two $2n$-tuples $\overline{t}, \overline{u} \in \mathbb{R}^{2n}_{\geq 0}$, their **concatenation** is the $2n$-tuple

$$\overline{t} \odot \overline{u} = \left(\overline{t} \square_n \overline{u}\right)\Big|_{[n+1 \dots 2n]}$$

The operations we will build on n-dominoes are then "matricial" translations of these three operations.

One question can then be raised here: why haven't we adapted this n-tuple approach to signals, and use the more cumbersome matricial presentation for n-signals? Different reasons for our choice will be given at different moments throughout this chapter, especially when defining the three operations on n-signals. These reasons are, to a certain extent, related to the fact that the set of signals and antisignals is not closed under concatenation.

6.2 Operations on n-dominoes

In this section we introduce several operations on n-dominoes, operations which aim at modeling the concatenation on signals with distinguished time points from the introduction of the chapter. We start with presenting some notations.

Given a natural $n \in \mathbb{N}$ and a set $X \subseteq [1 \dots n]$ we denote $l_X : [1 \dots n] \to [1 \dots card(X)]$ the surjection defined by

$$l_X(i) = card\{j \in X \mid j \leq i\} \tag{6.2}$$

Observe that the restriction of l_X to X is a bijection. We denote the inverse of this bijection by l_X^{-1}. Hence $l_X^{-1} : [1 \dots card(X)] \to X$.

Observe that, when $X = A \cup B$ with $A < B$, that is $x < y$ for all $x \in A$ and $y \in B$, we have:

$$l_{A \cup B}(i) = \begin{cases} l_A(i) & \text{iff } i \in A \\ card(A) + l_B(i) & \text{iff } i \in B \end{cases} \tag{6.3}$$

6.2.1 Projection

A useful operation is *projection*, which cuts some of the rows and columns of the matrix, such that the remaining matrix be still a square matrix.

Definition 6.2.1. *Given an n-domino $w \in D_n(\Sigma)$ the X-**projection** of w is denoted $w\big|_X$ and is defined as:*

$$\left(w\big|_X\right)_{ij} = w_{l_X^{-1}(i) l_X^{-1}(j)} \text{ for all } i, j \in [1 \dots card(X)] \tag{6.4}$$

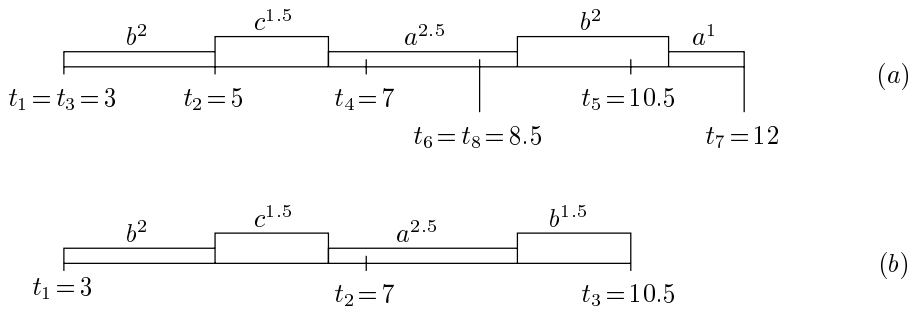


Fig. 6.2. The projection of the 8-signal at (a) onto the set $X = \{1, 4, 5\}$ gives the 3-signal at (b).

It is clear that the X-projection of an n-signal is a $card(X)$-signal. An example of projection is given in Figure 6.2.

Proposition 6.2.2. *For each $w \in D_n(\Sigma)$, $X \subseteq [1 \dots n]$ with $card(X) = p$ and $Y \subseteq [1 \dots p]$,*

$$w\big|_X\big|_Y = w\big|_{l_X^{-1}(Y)} \tag{6.5}$$

Proof. The components of the left-hand side in identity 6.5 are:

$$\big(w\big|_X\big|_Y\big)_{ij} = \big(w\big|_X\big)_{l_Y^{-1}(i),l_Y^{-1}(j)} = w_{l_X^{-1}(l_Y^{-1}(i)),l_X^{-1}(l_Y^{-1}(j))}$$

The identity follows if we prove $l_Y \circ l_X = l_{l_X^{-1}(Y)} : l_X^{-1}(Y) \to [1 \dots card(Y)]$ and this can be showed as follows:

$$\begin{aligned} l_Y(l_X(i)) &= card\big\{j \in Y \mid j \le l_X(i)\big\} = card\big\{l_X^{-1}(j) \in l_X^{-1}(Y) \mid l_X^{-1}(j) \le i\big\} \\ &= card\big\{k \in l_X^{-1}(Y) \mid k \le i\big\} = l_{l_X^{-1}(Y)}(i) \end{aligned}$$

We have applied here the fact that $l_X : X \to [1 \dots p]$ is a strictly increasing bijection.

Proposition 6.2.3. *For each $w, w' \in D_n(\Sigma)$ and $X, Y \subseteq [1 \dots n]$ with $X \cap Y \neq \emptyset$, $X \cup Y = [1 \dots n]$, if $w\big|_X = w'\big|_X$ and $w\big|_Y = w'\big|_Y$ then $w = w'$.*

In general, given $X_1, X_2, \dots, X_k \subseteq [1 \dots n]$ such that $X_1 \cup \dots \cup X_k = [1 \dots n]$ and $X_i \cap X_{i+1} \neq \emptyset$ for all $i \le k - 1$, if $w\big|_{X_i} = w'\big|_{X_i}$ for all $i \le k$ then $w = w'$.

Proof. We only need to prove that $w_{ij} = w'_{ij}$ for $i \in X$ and $j \in Y$, since the other cases hold by hypothesis or by symmetry: take some $k \in X \cap Y$, which must exist since $X \cap Y \neq \emptyset$. Since $i, k \in X$, we have $w_{ik} = w'_{ik}$. Similarly, $k, j \in Y$ implies $w_{kj} = w'_{kj}$. Therefore

$$w_{ij} = w_{ik}w_{kj} = w'_{ik}w'_{kj} = w'_{ij}$$

The second property follows by showing, by induction on i, that $w\big|_{X_1 \cup \dots \cup X_i} = w'\big|_{X_1 \cup \dots \cup X_i}$. □

6.2.2 Juxtaposition

The *juxtaposition* operation joins two matrices along a common "submatrix", such that both matrices can be found in the result:

Definition 6.2.4. *Given an m-domino $w \in D_m(\Sigma)$, an n-domino $w' \in D_n(\Sigma)$ and an integer $p \leq \min(m,n)$, the p-**indexed juxtaposition** of w and w' is defined if and only if $w\big|_{[m-p+1\ldots m]} = w'\big|_{[1\ldots p]}$, is denoted $w\square_p w'$ and is the $(m+n-p)$-domino $w'' \in D_{m+n-p}$ with the property that*

$$w''\big|_{[1\ldots m]} = w \text{ and } w''\big|_{[m-p+1\ldots m+n-p]} = w' \tag{6.6}$$

Note that Proposition 6.2.3 assures the uniqueness of the $(m+n-p)$-domino $w'' = w\square_p w'$.

The explicit construction of $w\square_p w'$ is the following:

$$\left(w\square_p w'\right)_{ij} = \begin{cases} w_{ij} & \text{iff } i,j \in [1\ldots m] \\ w'_{i-m+p,j-m+p} & \text{iff } i,j \in [m-p+1\ldots m+n-p] \\ w_{ik}\cdot w'_{k-m+p,j-m+p} & \text{iff } i\in[1\ldots m], j\in[m-p+1\ldots m+n-p], k\in[m-p\ldots m] \end{cases} \tag{6.7}$$

The components w_{ij} with $i \in [m-p+1\ldots m+n-p]$ and $j \in [1\ldots m]$ can be recovered as $(w_{ji})^{-1}$ from the third line in the definition. Note again that the definition of w''_{ij} for the case when $i \in [1\ldots m], j \in [m-p+1\ldots m+n-p]$ is independent of the choice of $k \in [1\ldots m]$.

An example of juxtaposition is depicted in Figure 6.3.

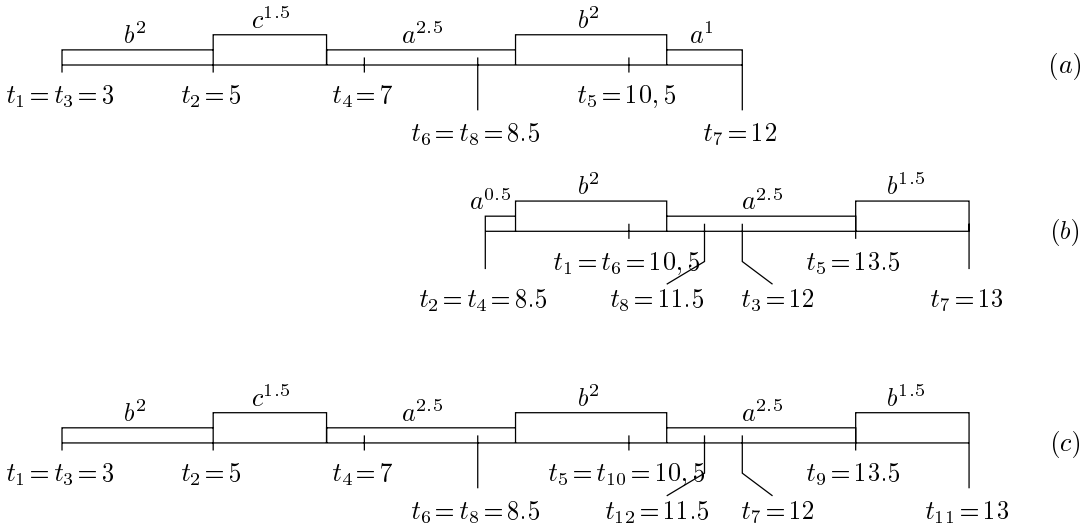
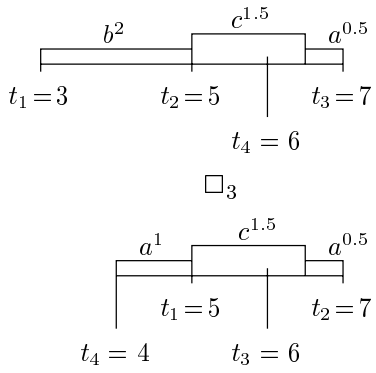


Fig. 6.3. The 4-juxtaposition of the 8-signals at (a) and (b) gives the 12-signal at (c).

Remark 6.2.5. Observe that, for each $w \in D_n(\Sigma)$ and $p,q \leq m$ such that $p+q \geq m+1$, $w\big|_{[1\ldots p]}\square_{p+q-m} w\big|_{[m-q+1\ldots m]} = w$.

This result is a direct corollary of property 6.2.3.

Remark 6.2.6. The p-juxtaposition of a m-signal with a n-signal does not yield a $(m+n-p)$-signals in general. As an example, the 3-juxtaposition of the two 4-signals in Figure 6.4 cannot yield a 5-signal since, intuitively, the point t_4 of the second 4-signal does not correctly fit in between the first two points t_1 and t_2 of the first 4-signal. This rewrites as the fact that the component $w_{15} = b^2a^{-1}$ of the result is neither a word nor an antiword.



$$w = \begin{pmatrix} \varepsilon & b^2 & \varepsilon & b^2c^{1.5}a^{0.5} & b^2a^{-1} \\ b^{-2} & \varepsilon & b^{-2} & c^{1.5}a^{0.5} & a^{-1} \\ \varepsilon & b^2 & \varepsilon & b^2c^{1.5}a^{0.5} & a^{-0.5}c^{-1.5}a^{-1} \\ a^{-0.5}c^{-1.5}b^{-2} & a^{-0.5}c^{-1.5} & a^{-0.5}c^{-1.5}b^{-2} & \varepsilon & c^{-1}a^{-1} \\ a^1b^{-2} & a^1 & a^1c^{1.5}a^{0.5} & a^1c^1 & \varepsilon \end{pmatrix}$$

Fig. 6.4. The 3-juxtaposition of the two 4-signals in this figure does not yield a 5-signal. The result is the 5-domino depicted below them.

This is an outcome of the fact that $\mathsf{Sig}(\Sigma) \cup \mathsf{Sig}(\Sigma^{-1})$ is not closed under concatenation, and argues in favor of our need of introducing n-dominoes as the basis of the study.

A sufficient condition for the p-juxtaposition of a m-signal w with a n-signal w' to be a $(m+n-p)$-signal is the conjunction of the following properties:

1. For each $i \in [1 \ldots m-p], j \in [m-p+1 \ldots m]$, $w_{ij} \in \mathsf{Sig}(\Sigma)$.
2. For each $i \in [1 \ldots p], j \in [p+1 \ldots n]$, $w'_{ij} \in \mathsf{Sig}(\Sigma)$.

This follows since, under this condition, the third line in the detailed definition of $w \square_p w'$ (Definition 6.7) would yield only elements of $\mathsf{Sig}(\Sigma)$.

This problem with concatenation which is not an internal operation on n-signals is one of our reasons for using the matricial presentation of n-signals, and of defining them as special cases of n-dominoes. Had we worked only with signals with distinguished points, we would have had to use the above sufficient condition for correctly defining the juxtaposition. But, as we will see later,

this condition is not satisfied by the n-signals which are issued by the n-signals semantics of timed automata, as we will see in a further section in this chapter. Hence, our calculus with n-signals would have been less expressive than timed automata.

6.2.3 Properties of juxtaposition

Proposition 6.2.7. *1. For each $w \in D_m(\Sigma)$, $w' \in D_n(\Sigma)$ given $X \subseteq [1 \dots m]$ with $card(X) = p$, given $Y \subseteq [1 \dots n]$ with $card(Y) = q$, and given $r \leq \min(p, q)$ such that and $[m - r + 1 \dots m] \subseteq X$ and $[1 \dots r] \subseteq Y$, we have that*

$$w\big|_X \Box_r w'\big|_Y = (w \Box_r w')\big|_{X \cup (Y+m-r)}. \tag{6.8}$$

2. For each $w \in D_m(\Sigma)$, $w' \in D_n(\Sigma)$ and $w'' \in D_p(\Sigma)$ and for each $q \leq \min(m, n)$ and $r \leq \min(n, p)$,

$$(w \Box_q w') \Box_r w'' = w \Box_q (w' \Box_r w'') \tag{6.9}$$

Proof. Both properties will be proved with the aid of Propositions 6.2.2 and 6.2.3:

For the first identity, denote first $p = card(X)$ and $q = card(Y)$. Let us observe first that the left-hand side $w\big|_X \Box_r w'\big|_Y$ is defined iff the right-hand side $(w \Box_r w')\big|_{X \cup (Y+m-r)}$ is:

$$\begin{aligned} w\big|_X\big|_{[p-r+1 \dots p]} &= w\big|_{l_X^{-1}([p-r+1 \dots p])} && \text{(by identity 6.5)} \\ &= w\big|_{[m-r+1 \dots m]} \end{aligned}$$

since, by hypothesis, $[m - r + 1 \dots m] \subseteq X \subseteq [1 \dots m]$ and hence $l_X([m - r + 1 \dots m]) = [p - r + 1 \dots p]$, and, respectively,

$$\begin{aligned} w'\big|_Y\big|_{[1 \dots r]} &= w\big|_{l_Y^{-1}([1 \dots r])} && \text{(by identity 6.5)} \\ &= w\big|_{[1 \dots r]} \end{aligned}$$

since, again by hypothesis, $[1 \dots r] \subseteq Y \subseteq [1 \dots n]$ and hence $l_Y([1 \dots r]) = [1 \dots r]$.

We will prove then that the projections of both sides of identity 6.5 onto $[1 \dots p]$ and $[p - r + 1 \dots p + q - r]$ are respectively equal. The projections of the left-hand side of Identity 6.8 are, by definition of $\Box$, the following:

$$\begin{aligned} \big(w\big|_X \Box_r w'\big|_Y\big)\big|_{[1 \dots p]} &= w\big|_X \\ \big(w\big|_X \Box_r w'\big|_Y\big)\big|_{[p-r+1 \dots p+q-r]} &= w'\big|_Y \end{aligned}$$

For the projections of the right-hand side of Identity 6.8 we will apply the identity 6.5. First, the projection onto $[1 \dots p]$ gives:

$$
\begin{aligned}
(w\Box_r w')\big|_{X\cup(Y+m-r)}\ \big|_{[1\ldots p]} &= (w\Box w')\big|_{l^{-1}_{X\cup(Y+m-r)}([1\ldots p])} && \text{(by identity 6.5)}\\
&= (w\Box_r w')\big|_{l_X^{-1}([1\ldots p])} && \text{(by property 6.3 of } l_X)\\
&= (w\Box_r w')\big|_X && \text{(since } l_X(X)=[1\ldots p])\\
&= (w\Box_r w')\big|_{l^{-1}_{[1\ldots m]}(X)} && \text{(since } X\subseteq[1\ldots m])\\
&= (w\Box_R w')\big|_{[1\ldots m]}\ \big|_X && \text{(by identity 6.5)}\\
&= w\big|_X
\end{aligned}
$$

Before computing the projection onto $[p-r\ldots p+q-r]$ we observe that $l_{X\cup(Y+m-r)}\big(X\setminus(Y+m-r)\big)=[1\ldots p-r]$. Therefore:

$$
\begin{aligned}
&(w\Box_r w')\big|_{X\cup(Y+m-r)}\ \big|_{[p-r\ldots p+q-r]} =\\
&\quad= (w\Box_r w')\big|_{l^{-1}_{X\cup(Y+m-r)}([p-r\ldots p+q-r])} && \text{(by identity 6.5)}\\
&\quad= (w\Box_r w')\big|_{l^{-1}_{Y+m-r}([1\ldots q])} && \text{(by the above observation)}\\
&\quad= (w\Box_r w')\big|_{Y+m-r} && \text{(since } l_{Y+m-r}(Y+m-r)=[1\ldots q])\\
&\quad= (w\Box_r w')\big|_{l^{-1}_{[m-r+1\ldots m+n-r]}(Y)} && \text{(since } Y+m-r\subseteq[m-r+1\ldots m+n-r])\\
&\quad= (w\Box_r w')\big|_{[m-r+1\ldots m+n-r]}\ \big|_Y && \text{(by identity 6.5)}\\
&\quad= w'\big|_Y
\end{aligned}
$$

For proving the Identity 6.9, let us observe first that the right-hand side is defined iff the left-hand side is defined, and both iff $w\Box_q w'$ and $w'\Box_r w''$ are defined, since:

$$
\begin{aligned}
(w\Box_q w')\big|_{[m+n-q-r+1\ldots m+n-q]} &= (w\Box_q w')\big|_{l^{-1}_{[m-q+1\ldots m+n-q]}([n-r+1\ldots n])}\\
&\quad\text{(since } l_{[m-q+1\ldots m+n-q]}([m+n-q-r+1\ldots m+n-q])=[n-r+1\ldots n]])\\
&= (w\Box_q w')\big|_{[m-q+1\ldots m+n-q]}\ \big|_{[n-r+1\ldots n]} && \text{(by identity 6.5)}\\
&= w'\big|_{[n-r+1\ldots n]} && \text{(by definition of } \Box_q)
\end{aligned}
$$

$$
\begin{aligned}
(w'\Box_r w'')\big|_{[1\ldots q]} &= (w'\Box_r w'')\big|_{l^{-1}_{[1\ldots n]}([1\ldots q])} && \text{(since } l_{[1\ldots n]}([1\ldots q])=[1\ldots q]])\\
&= (w\Box_q w')\big|_{[1\ldots n]}\ \big|_{[1\ldots q]} && \text{(by identity 6.5)}\\
&= w'\big|_{[1\ldots q]} && \text{(by definition of } \Box_q)
\end{aligned}
$$

Hence

- $w\big|_{[m-q+1\ldots m]} = w'\big|_{[1\ldots q]}$ if and only if $w\big|_{[m-q+1\ldots m]} = (w'\Box_r w'')\big|_{[1\ldots q]}$.
- $w'\big|_{[n-r+1\ldots n]} = w''\big|_{[1\ldots r]}$ if and only if $(w\Box_q w')\big|_{[m+n-q-r+1\ldots m+n-q]} = w''\big|_{[1\ldots r]}$.

We will then show that the projections of both sides of this identity onto the sets $[1 \dots m]$, $[m-q+1 \dots m+n-q]$ and $[m+n-q-r+1 \dots m+n+p-q-r]$ are respectively equal. To this end, observe first that

$$l_{[1\dots m+n-r]}([1 \dots m]) = [1 \dots m] \tag{6.10}$$

$$l_{[1\dots m+n-q]}([m-q+1 \dots m+n-q]) = [m-q+1 \dots m+n-q] \tag{6.11}$$

Then, the three projections of the left-hand side of Identity 6.9 can be rewritten as follows:

$$\begin{aligned}
\big((w \Box_q w') \Box_r w''\big)\big|_{[1\dots m]} &= \big((w \Box_q w') \Box_r w''\big)\big|_{l^{-1}_{[1\dots m+n-q]}([1\dots m])} && \text{(by observation 6.10)}\\
&= \big((w \Box_q w') \Box_r w''\big)\big|_{[1\dots m+n-q]}\big|_{[1\dots m]} && \text{(by identity 6.5)}\\
&= (w \Box_q w')\big|_{[1\dots m]} && \text{(by definition of } \Box_r)\\
&= w && \text{(by definition of } \Box_q)
\end{aligned}$$

$$\begin{aligned}
\big((w \Box_q w') \Box_r w''\big)\big|_{[m-q+1\dots m+n-q]} &= \\
&= \big((w \Box_q w') \Box_r w''\big)\big|_{l^{-1}_{[1\dots m+n-q]}([m-q+1\dots m+n-q])} && \text{(by observation 6.11)}\\
&= \big((w \Box_q w') \Box_r w''\big)\big|_{[1\dots m+n-q]}\big|_{[m-q+1\dots m+n-q]} && \text{(by identity 6.5)}\\
&= (w \Box_q w')\big|_{[m-q+1\dots m+n-q]} && \text{(by definition of } \Box_r)\\
&= w' && \text{(by definition of } \Box_q)\\
\big((w \Box_q w') \Box_r w''\big)\big|_{[m+n-q-r+1\dots m+n+p-q-r]} &= \\
&= w'' && \text{(by definition of } \Box_q)
\end{aligned}$$

Before computing the projections of the right-hand side of Identity 6.9, note the following properties:

$$l_{[m-q+1\dots m+n+p-q-r]}([m-q+1 \dots m+n-q]) = [1 \dots n] \tag{6.12}$$

$$l_{[m-q+1\dots m+n+p-q-r]}([m+n-q-r+1 \dots m+n+p-q-r]) = [n-r+1 \dots n+p-r] \tag{6.13}$$

The projections of the right-hand side of Identity 6.9 are the following:

$$\begin{aligned}
\big(w \Box_q (w' \Box_r w'')\big)\big|_{[1\dots m]} &= w && \text{(by definition of } \Box_q)\\
\big(w \Box_q (w' \Box_r w'')\big)\big|_{[m-q+1\dots m+n-q]} &= \\
&= \big(w \Box_q (w' \Box_r w'')\big)\big|_{l^{-1}_{[m-q+1\dots m+n+p-q-r]}([1\dots n])} && \text{(by observation 6.12)}\\
&= \big(w \Box_q (w' \Box_r w'')\big)\big|_{[m-q+1\dots m+n+p-q-r]}\big|_{[1\dots n]} && \text{(by identity 6.5)}\\
&= (w' \Box_r w'')\big|_{[1\dots n]} && \text{(by definition of } \Box_q)\\
&= w' && \text{(by definition of } \Box_r)
\end{aligned}$$

$$
\begin{aligned}
\big(w\Box_q(w'\Box_r w'')\big)\big|_{[m+n-q-r+1\ldots m+n+p-q-r]} &= \\
&= \big(w\Box_q(w'\Box_r w'')\big)\big|^{-1}_{[m-q+1\ldots m+n+p-q-r]}([n-r+1\ldots n+p-r]) && \text{(by observation 6.13)}\\
&= \big(w\Box_q(w'\Box_r w'')\big)\big|_{[m-q+1\ldots m+n+p-q-r]}\big|_{[n-r+1\ldots n+p-r]} && \text{(by identity 6.5)}\\
&= (w'\Box_r w'')\big|_{[n-r+1\ldots n+p-r]} && \text{(by definition of } \Box_q)\\
&= w'' && \text{(by definition of } \Box_r)
\end{aligned}
$$

Hence, an application of Proposition 6.2.3 ends our proof. □

At the end of this subsection we prove a property which relates orderings compatible with 4-signals to juxtaposition and will be used in the proof of the undecidability theorem.

6.2.4 Concatenation

The concatenation of two $2n$-dominoes $w, w' \in D_{2n}(\Sigma)$ is defined as the n-juxtaposition of w and w' followed by the projection onto the first and the last n components.

Definition 6.2.8. *Given two $2n$-dominoes $w, w' \in D_{2n}(\Sigma)$, the **concatenation** of w and w' is denoted as $w \odot w'$ and defined as*

$$w \odot w' = \big(w\Box_n w'\big)\big|_{[1\ldots n]\cup[2n+1\ldots 3n]} \tag{6.14}$$

In detail, $w \odot w'$ is defined iff $w\big|_{[n+1\ldots 2n]} = w'\big|_{[1\ldots n]}$ and in this case we have

$$(w \odot w')_{ij} = \begin{cases} w_{ij} & \text{iff } i,j \in [1\ldots n] \\ w'_{ij} & \text{iff } i,j \in [n+1\ldots 2n] \\ w_{ik}\cdot w'_{k-n,j} & \text{iff } i\in[1\ldots n], j\in[n+1\ldots 2n] \text{ and } k\in[n+1\ldots 2n] \end{cases} \tag{6.15}$$

The components w_{ij} with $i \in [n+1\ldots 2n]$ and $j \in [1\ldots n]$ can be recovered as $(w_{ji})^{-1}$ from the third line in the definition. An example of concatenation is given in Figure 6.5 below. The reader may observe now that the concatenation of the two 8-signals from Figure 6.5, is the projection of their 4-juxtaposition, as given in Figure 6.3, onto the set $[1\ldots 4]\cup[9\ldots 12]$.

Remark 6.2.9. As an outcome of the remark 6.2.6, the composition of two $2n$-signals might not be a $2n$-signal in general.

Proposition 6.2.10. *Composition is associative, has no unit but each $2n$-domino has a left and a right unit and a left and right inverse w.r.t. this unit:*

1. *For each triplet of $2n$-dominoes $w, w', w'' \in D_{2n}(\Sigma)$*

$$(w \odot w') \odot w'' = w \odot (w' \odot w'') \tag{6.16}$$

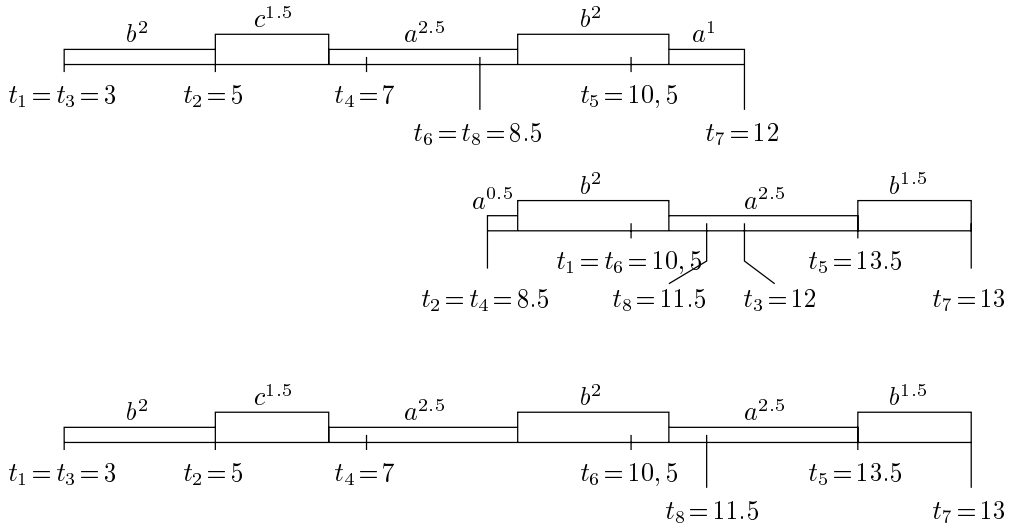


Fig. 6.5. Concatenation of two 8-signals.

2. *For each* $w \in D_{2n}(\Sigma)$*, denote* $\mathbf{1}^l_w$*, resp.* $\mathbf{1}^r_w$ *the* $2n$*-dominoes defined as follows: for each* $i, j \in [1 \dots n]$*,*

$$(\mathbf{1}^l_w)_{ij} = (\mathbf{1}^l_w)_{n+i,j} = (\mathbf{1}^l_w)_{i,n+j} = (\mathbf{1}^l_w)_{n+i,n+j} = w_{ij} \tag{6.17}$$
$$(\mathbf{1}^r_w)_{ij} = (\mathbf{1}^r_w)_{n+i,j} = (\mathbf{1}^r_w)_{i,n+j} = (\mathbf{1}^r_w)_{n+i,n+j} = w_{n+i,n+j} \tag{6.18}$$

Observe that $(\mathbf{1}^l_w)_{i,n+i} = (\mathbf{1}^r_w)_{i,n+i} = \varepsilon$*, for any* $i \in [1 \dots n]$*.*
Then

$$\mathbf{1}^l_w \odot w = w \odot \mathbf{1}^r_w = w \tag{6.19}$$

3. *In the same setting, define* $\tilde{w} \in D_{2n}(\Sigma)$ *as follows:*

$$\tilde{w}_{ij} = \begin{cases} w_{i+n,j+n} & \textit{iff } i, j \in [1 \dots n] \\ w_{i+n,j-n} & \textit{iff } i \in [1 \dots n], j \in [n+1 \dots 2n] \\ w_{i-n,j+n} & \textit{iff } i \in [n+1 \dots 2n], j \in [1 \dots n] \\ w_{i-n,j-n} & \textit{iff } i, j \in [n+1 \dots 2n] \end{cases} \tag{6.20}$$

Then

$$w \odot \tilde{w} = \mathbf{1}^l_w \textit{ and } \tilde{w} \odot w = \mathbf{1}^r_w \tag{6.21}$$

Proof. The associativity property follows from the definition of composition and the associativity of juxtaposition. Let us observe first that $(w \odot w') \odot w''$ is defined iff $w \odot (w' \odot w'')$ is defined, and both are defined iff $w \odot w'$ and $w' \odot w''$ are defined. This follows directly from the proof of associativity of $\Box$. Then:

$$
\begin{aligned}
(w \odot w') \odot w'' &= \big((w \square_n w')\big|_{[1\ldots n]\cup[2n+1\ldots 3n]} \square_n w''\big)\big|_{[1\ldots n]\cup[2n+1\ldots 3n]} \\
&= \big((w \square_n w') \square_n w''\big)\big|_{[1\ldots n]\cup[2n+1\ldots 3n]\cup[3n+1\ldots 4n]}\big|_{[1\ldots n]\cup[2n+1\ldots 3n]} && \text{(by identity 6.8)} \\
&= \big((w \square_n w') \square_n w''\big)\big|_{l^{-1}_{[1\ldots n]\cup[2n+1\ldots 4n]}([1\ldots n]\cup[2n+1\ldots 3n])} && \text{(by identity 6.5)} \\
&= \big((w \square_n w') \square_n w''\big)\big|_{[1\ldots n]\cup[3n+1\ldots 4n]} \\
&\text{(since } l_{[1\ldots n]\cup[2n+1\ldots 4n]}([1\ldots n]\cup[3n+1\ldots 4n]) = ([1\ldots n]\cup[2n+1\ldots 3n])\text{)}
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
w \odot (w' \odot w'') &= \big(w \square_n (w' \square_n w'')\big|_{[1\ldots n]\cup[2n+1\ldots 3n]}\big)\big|_{[1\ldots n]\cup[2n+1\ldots 3n]} \\
&= \big(w \square_n (w' \square_n w'')\big)\big|_{[1\ldots n]\cup[n+1\ldots 2n]\cup[3n+1\ldots 4n]}\big|_{[1\ldots n]\cup[2n+1\ldots 3n]} && \text{(by identity 6.8)} \\
&= \big(w \square_n (w' \square_n w'')\big)\big|_{l^{-1}_{[1\ldots 2n]\cup[3n+1\ldots 4n]}([1\ldots n]\cup[2n+1\ldots 3n])} && \text{(by identity 6.5)} \\
&= \big(w \square_n (w' \square_n w'')\big)\big|_{[1\ldots n]\cup[3n+1\ldots 4n]} \\
&\text{(since } l_{[1\ldots 2n]\cup[3n+1\ldots 4n]}([1\ldots n]\cup[3n+1\ldots 4n]) = ([1\ldots n]\cup[2n+1\ldots 3n])\text{)}
\end{aligned}
$$

For the second property, observe first that $\mathbf{1}^l_w$ and w can be concatenated (in this order) since by definition

$$\mathbf{1}^l_w\big|_{[n+1\ldots 2n]} = w\big|_{[1\ldots n]}$$

Then

$$
\begin{aligned}
&(\mathbf{1}^l_w \square_n w)_{ij} = \\
&\quad = \begin{cases} (\mathbf{1}^l_w)_{ij} & \text{for } i,j \in [1\ldots n] \\ w_{ij} & \text{for } i,j \in [n+1\ldots 2n] \\ (\mathbf{1}^l_w)_{ik}\cdot w_{k-n,j} & \text{for } i \in [1\ldots n], j \in [n+1\ldots 2n] \text{ and some } k \in [n+1\ldots 2n] \end{cases} \\
&\quad = \begin{cases} w_{ij} & \text{for } i,j \in [n+1\ldots 2n] \text{ or } i,j \in [n+1\ldots 2n] \\ (\mathbf{1}^l_w)_{i,n+i}\cdot w_{ij} & \text{for } i \in [1\ldots n] \text{ and } j \in [n+1\ldots 2n] \end{cases} \\
&\quad = w_{ij}
\end{aligned}
$$

The proof is similar for the right unit $\mathbf{1}^r_w$.

Finally, for proving that $\tilde{w}$ is the inverse of w w.r.t. concatenation we only must observe that, for any $i,j \in [1\ldots n]$ we have

$$
\begin{aligned}
\big(w \odot \tilde{w}\big)_{ij} &= \big(w \odot \tilde{w}\big)_{n+i,n+j} = w_{ij} \\
\big(w \odot \tilde{w}\big)_{i,n+j} &= w_{i,n+i}\cdot \tilde{w}_{i,n+j} = w_{i,n+i}\cdot w_{n+i,j} = w_{ij}
\end{aligned}
$$

□

6.3 n-domino languages

Having defined n-dominoes and their operations, we may turn our attention to sets of n-dominoes now. These sets will be called n-**domino languages**, respectively n-**signal languages** when they consist of n-signals only. We denote the family of n-domino languages as $\mathcal{DL}_n(\Sigma)$ and the family of n-signal languages as $\mathcal{WL}_n(\Sigma)$.

All the operations built so far extend naturally to n-domino languages. We will only write here the extension of the concatenations to $2n$-domino languages, since this gives rise to a star operation: given two $2n$-domino languages $L, L' \subseteq D_{2n}(\Sigma)$, the concatenation of L and L' is the $2n$-domino language

$$L \odot L' = \{w \odot w' \mid w \in L, w' \in L'\}.$$

Let us consider the following language:

$$\mathbf{1}_{2n} = \left\{w \in D_{2n}(\Sigma) \mid \forall i \in [1 \dots n], w_{i,n+i} = \varepsilon\right\} = \left\{\mathbf{1}_w^l \mid w \in D_{2n}(\Sigma)\right\} \tag{6.22}$$

Proposition 6.3.1. *$\mathbf{1}_{2n}$ is the unit for concatenation on sets, hence $\left(\mathcal{P}(D_{2n}(\Sigma)), \odot, \mathbf{1}_{2n}\right)$ is a monoid.*

Concatenation gives rise to a *star* operation: for each $2n$-domino language $L \subseteq D_{2n}(\Sigma)$, the **star** of L is defined as:

$$L^{\circledast} = \bigcup_{k \geq 0} L^{k\odot}$$

where $L^{0\odot} = \mathbf{1}_{2n}$ and $L^{(k+1)\odot} = L^{k\odot} \odot L$ for all $k \in \mathbb{N}$.

Proposition 6.3.2. *The structure $\left(\mathcal{P}(D_{2n}(\Sigma)), \cup, \odot, \circledast, \emptyset, \mathbf{1}_{2n}\right)$ is a Kleene algebra.*

We will also use the *positive star* operation $\oplus$, defined as the positive iterations of the given language:

$$L^{\oplus} = \bigcup_{k \geq 1} L^{k\odot}$$

A nice property relating projection and juxtaposition is the following:

Proposition 6.3.3. *Given an m-domino language L, an n-domino language L' and a positive number $p \leq \min(m, n)$, denote $\overline{L}$ the set of $(m + n - p)$-dominoes which project onto elements of L and $\overline{L}'$ the set of $(m + n - p)$-dominoes which project onto elements of L', that is:*

$$\overline{L} = \left\{w \in D_{m+n-p}(\Sigma) \mid w\big|_{[1\dots m]} \in L\right\}$$
$$\overline{L}' = \left\{w \in D_{m+n-p}(\Sigma) \mid w\big|_{[m-p+1\dots m+n-p]} \in L'\right\}$$

Then

$$L \square_p L' = \overline{L} \cap \overline{L}' \tag{6.23}$$

Proof. By straightforward verification:

$$\begin{aligned} L\Box_p L' &= \left\{w\Box_p w' \mid w \in L, w' \in L'\right\} \\ &= \left\{w \in D_{m+n-p}(\Sigma) \mid w\big|_{[1\ldots n]} \in L, w\big|_{[m-p+1\ldots m+n-p]} \in L'\right\} \\ &= \left\{w \in D_{m+n-p}(\Sigma) \mid w\big|_{[1\ldots n]} \in L\right\} \cap \left\{w \in D_{m+n-p}(\Sigma) \mid w\big|_{[m-p+1\ldots m+n-p]} \in L'\right\} \\ &= \overline{L} \cap \overline{L}' \end{aligned}$$

□

6.4 Regminoes, regsignals, and regular expressions over them

Going back to the "colored parentheses" idea, we would like to decompose each object of the form

$$\left\langle^{\text{blue}} a \left\langle^{\text{red}} b\right\rangle_1^{\text{blue}} \left\langle^{\text{blue}} a\right\rangle_1^{\text{red}} \left\langle^{\text{red}} b\right\rangle_1^{\text{blue}} a\right\rangle_1^{\text{red}}$$

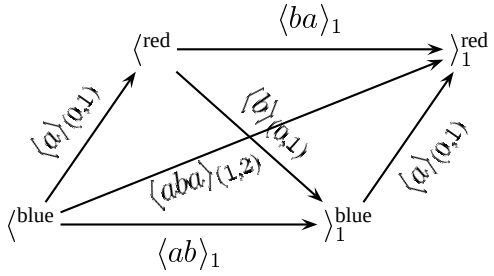
into a concatenation of the kind

$$\left\langle^{\text{blue}} a \left\langle^{\text{red}} b\right\rangle_1^{\text{blue}} a\right\rangle_1^{\text{red}} \odot \left\langle^{\text{blue}} a \left\langle^{\text{red}} b\right\rangle_1^{\text{blue}} a\right\rangle_1^{\text{red}}$$

Consequently, we would like to base our regular expressions with colored parentheses on atoms like $\left\langle^{\text{blue}} a \left\langle^{\text{red}} b\right\rangle_1^{\text{blue}} a\right\rangle_1^{\text{red}}$, that is, in which, if we apply a "color filter" for the any of the colors, we would get a timed regular expression of the kind $E_1 \cdot \langle E_2\rangle_I E_3$, in which E_1, E_2 and E_3 are regular expressions without timing parentheses.

This rough idea still needs some refinement: observe that, in an atom of the kind $\left\langle^{\text{blue}} a \left\langle^{\text{red}} b\right\rangle_1^{\text{blue}} a\right\rangle_1^{\text{red}}$, there exist some implicit timing constraints limiting the duration of each state: neither of the two states a or the state b may last more than 1 time unit. A graphical presentation of the resulting object is the following:

$\left\langle^{\text{blue}} a \left\langle^{\text{red}} b\right\rangle_1^{\text{blue}} a\right\rangle_1^{\text{red}}$



We will put all this information in a matricial presentation: we define *regminoes* as matrices of timed regular expressions, whose semantics consists of sets of dominoes. These are the atoms of our calculus of regular expressions with colored parentheses:

Definition 6.4.1. *An n-**regmino** is a matrix $R = (R_{ij})_{i,j\in[1\ldots n]}$ whose components are timed regular expressions: for each $i, j \in [1\ldots n]$, R_{ij} is a timed regular expression over $\Sigma \cup \Sigma^{-1}$.*

*An n-**regsignal** is an n-regmino R for which, for each $i, j \in [1\ldots n]$, $R_{ij} = \langle E\rangle_I + \langle E'\rangle_I$ for some untimed regular expression E over Σ, some untimed regular expression E' over Σ^{-1} and some interval $I \in \mathbb{Q}Int$.*

The set of n-regminoes is denoted $\mathcal{RD}_n(\Sigma)$ while the set of n-regsignals is denoted $\mathcal{RS}ig_n(\Sigma)$.

The **semantics** of n-regminoes is the mapping $\|\cdot\| : \mathcal{RD}_n(\Sigma) \to \mathcal{P}(D_n(\Sigma))$ defined as follows: for each $R \in \mathcal{RD}_n(\Sigma)$,

$$\|R\| = \{w \in D_n(\Sigma) \mid w_{ij} \in \|R_{ij}\| \text{ for all } i, j \in [1 \dots n]\} \tag{6.24}$$

Remark 6.4.2. Observe that the semantics of an n-regsignal contains only n-signals.

Figure 6.6 gives an example of a 4-regsignal and a 4-signal in its semantics.

$$R = \begin{pmatrix} \varepsilon & \langle b \rangle_{\{2\}} & \langle b + \varepsilon \rangle_{[0,1]} & \langle b(ca)^* \rangle_{[3,5]} \\ \langle b^{-1} \rangle_{\{-2\}} & \varepsilon & \langle b^{-1} \rangle_{[-3,-1]} & \langle (a + \varepsilon)ca \rangle_{[2,3]} \\ \langle b + \varepsilon \rangle_{[-1,0]} & \langle b \rangle_{[1,3]} & \varepsilon & \langle bac + a^{-1}b^{-1} \rangle_{[-2,4]} \\ \langle (a^{-1}c^{-1})^* b^{-1} \rangle_{[-5,-3]} & \langle a^{-1}c^{-1}a^{-1} \rangle_{[-3,-2]} & \langle c^{-1}a^{-1}b^{-1} + ba \rangle_{[-4,2]} & \varepsilon \end{pmatrix}$$

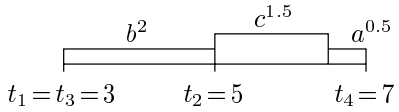


Fig. 6.6. A 4-regsignal and a 4-signal in its semantics.

We have used here the expression $\langle bac + a^{-1}b^{-1} \rangle_{[-2,4]}$ as a shortcut for $\langle bac \rangle_{[-2,4]} + \langle a^{-1}b^{-1} \rangle_{[-2,4]}$. Observe also that, since signals cannot have negative length and antisignals cannot have positive length, we may further replace this timed regular expression with $\langle bac \rangle_{[0,4]} + \langle a^{-1}b^{-1} \rangle_{[-2,0]}$.

For each n-regsignal $R \in \mathcal{RS}ig_n(\Sigma)$ we will denote $R_{ij} = R^+_{ij} + R^-_{ij}$ with R^+_{ij} being the "positive part" and R^-_{ij} the "negative part" of the timed regular expression, that is:

$$R^+_{ij} = \langle E \rangle_I \text{ with } E \text{ regular expression over } \Sigma^* \text{ and } I \subseteq [0, \infty)$$
$$R^-_{ij} = \langle E \rangle_I \text{ with } E \text{ regular expression over } (\Sigma^{-1})^* \text{ and } I \subseteq (-\infty, 0]$$

6.4.1 Projection and juxtaposition on n-regsignals

We have seen that the domino operations can be naturally extended to languages. We may ask then whether there is a way to *represent* the results of each operation, when applied to languages which are representable by regminoes. We show here that, for juxtaposition and intersection, such a representation can be found, but not for union and projection, and, hence, neither for concatenation and star.

We present these results in an algebraic setting, that is, we define juxtaposition and intersection *on regminoes* and prove that they are *compositional*. On the other hand, we show that the natural candidate for the X-projection operation on regminoes - the operation which removes the rows and columns not in X - is *not* compositional.

Definition 6.4.3. *Given $R \in \mathcal{RD}_n(\Sigma)$ and $X \subseteq [1 \dots n]$, the X-**projection** of R is the $(card(X))$-regmino denoted $R|_X$ and defined as follows:*

$$\left(R\big|_X\right)_{ij} = R_{l_X^{-1}(i)l_X^{-1}(j)} \text{ for all } i,j \in [1\ldots card(X)]$$

*Given $R_1 \in \mathcal{RD}_m(\Sigma)$, $R_2 \in \mathcal{RD}_n(\Sigma)$, and $p \leq \min(m,n)$ a nonnegative integer, the p-**juxtaposition** of R_1 with R_2 is the $(m+n-p)$-regmino $R \in \mathcal{RD}_{m+n-p}(\Sigma)$ denoted $R = R_1 \square_p R_2$ and defined as follows:*

$$\left(R_1 \square_p R_2\right)_{ij} =$$

$$= \begin{cases} (R_1)_{ij} & \text{iff } i \in [1\ldots m-p], j \in [1\ldots m] \\ & \text{or } i \in [1\ldots m], j \in [1\ldots m-p] \\ (R_2)_{i-m+p,j-m+p} & \text{iff } i \in [m-p+1\ldots m+n-p], j \in [m+1\ldots m+n-p] \\ & \text{or } i \in [m+1\ldots m+n-p],\ j \in [m-p+1\ldots m+n-p] \\ (R_1)_{ij} \cap (R_2)_{i-m+p,j-m+p} & \text{iff } i,j \in [m-p+1\ldots m] \\ \bigcap\limits_{k=m-p+1}^{m} (R_1)_{ik}(R_2)_{k-m+p,j-m+p} & \text{iff } i \in [1\ldots m-p], j \in [m+1\ldots m+n-p] \\ \bigcap\limits_{k=m-p+1}^{m} (R_2)_{i-m+p,k-m+p}(R_1)_{kj} & \text{iff } j \in [1\ldots m-p], i \in [m+1\ldots m+n-p] \end{cases}$$

Unfortunately projection is not a good syntactic operation since it does not commute with semantics of regminoes: we might have an n-regmino with an empty semantics whose projection onto some subset has a nonempty semantics. For example, consider the following 4-regsignal over a one-letter alphabet $\Sigma = \{a\}$ (in fact, a matrix whose entries are sets of reals):

$$R = \begin{pmatrix} \varepsilon & a^1 + a^2 & a^2 + a^5 & a^6 + a^8 \\ a^{-2} + a^{-1} & \varepsilon & a^1 + a^3 & a^4 + a^7 \\ a^{-5} + a^{-2} & a^{-3} + a^{-1} & \varepsilon & a^3 + a^4 \\ a^{-8} + a^{-6} & a^{-7} + a^{-4} & a^{-4} + a^{-3} & \varepsilon \end{pmatrix} \tag{6.25}$$

This 4-regsignal has an empty semantics: if we construct all the integer-valued matrices A whose components belong to the respective components of R, that is, with $A_{ij} \in R_{ij}$ for all $i,j \in [1\ldots 4]$, we observe that none is a 4-signal as none satisfies the triangle identity. However the projection of R onto the set $X = \{1,3\}$ gives the following 2-regsignal:

$$R\big|_{\{1,3\}} = \begin{pmatrix} \varepsilon & \{a^1, a^2\} \\ \{a^{-2}, a^{-1}\} & \varepsilon \end{pmatrix}$$

whose semantics is nonempty, since:

$$w = \begin{pmatrix} \varepsilon & a^1 \\ a^{-1} & \varepsilon \end{pmatrix} \in \left\|R\big|_{\{1,3\}}\right\| \cap \mathsf{Sig}_2(\{a\})$$

In general, we only have the inclusion

$$\big\|R\big|_X\big\| \supset \{w\big|_X \mid w \in \|R\|\} \tag{6.26}$$

for each $R \in \mathcal{RD}(\Sigma)$ and $X \subseteq [1 \ldots n]$.

Contrary to projection, *indexed juxtaposition* is compositional w.r.t. semantics:

Proposition 6.4.4. *The following property holds for any $R_1 \in \mathcal{RD}_m(\Sigma)$, $R_2 \in \mathcal{RD}_n(\Sigma)$, and $p \leq \min(m,n)$:*

$$\|R_1 \Box_p R_2\| = \|R_1\| \Box_p \|R_2\| \tag{6.27}$$

Proof. The property follows by easy verification: for the direct inclusion observe that, if $w \in \|R_1 \Box_p R_2\|$ then $w\big|_{[1\ldots m]} \in \|R_1\|$ and $w\big|_{[m-p+1\ldots m+n-p]} \in \|R_2\|$. But $w = w\big|_{[1\ldots m]} \Box_p w\big|_{[m-p+1\ldots m+n-p]}$, hence $w \in \|R_1\| \Box_p \|R_2\|$.

For the inverse inclusion we just have to observe that, if we are given $w_1 \in \|R_1\|$ and $w_2 \in \|R_2\|$ such that $w_1 \Box_p w_2$ is defined, then for all $i \in [1 \ldots m-p]$, $j \in [m+1 \ldots m+n-p]$ and $k \in [m-p+1 \ldots m]$, $(w_1 \Box w_2)_{ij} = (w_1)_{ik} \cdot (w_2)_{k-m+p,j-m+p}$ and this implies that

$$(w_1 \Box w_2)_{ij} \in \bigcap_{k=m-p+1}^{m} \|(R_1)_{ik}\| \cdot \|(R_2)_{k-m+p,j-m+p}\| = \|R_1 \Box_p R_2\|_{ij} \qquad \Box$$

An operation which is available *for n-regsignals only* is *intersection*:

Definition 6.4.5. *Given two n-regsignals $R_1, R_2 \in \mathcal{RS}ig_n(\Sigma)$ the **intersection** of R_1 and R_2 is the n-regsignal R with*

$$R_{ij} = (R_1)_{ij} \cap (R_2)_{ij} \text{ for all } i,j \in [1 \ldots n] \tag{6.28}$$

We denote then $R = R_1 \cap R_2$.

Remark 6.4.6. Of course, to actually obtain n-regsignals we need to transform the intersection in each component into a regular expression. Observe that it is essential that both operands are n-regsignals since then each component can be still written in the form $\langle E \rangle_I + \langle E' \rangle_{I'}$ with E an untimed regular expression over Σ and E' an untimed regular expression over Σ^{-1}. Here I is intended to be a nonnegative interval and I' a nonpositive interval.

On the contrary, the intersection of the semantics of two n-regminoes might not be representable as an n-regmino. This follows even for $n = 1$ by the nonclosure of timed regular expressions under intersection [ACM97, Her99].

Proposition 6.4.7. *For each pair of n-regsignals $R_1, R_2 \in \mathcal{RS}ig_n(\Sigma)$ we have*

$$\|R_1 \cap R_2\| = \|R_1\| \cap \|R_2\|$$

We may then to alternatively define p-juxtaposition $R_1 \Box_p R_2$ by the aid of projection and intersection as follows: suppose $R_1 \mathcal{RS}ig_m(\Sigma)$ and $R_2 \in \mathcal{RS}ig_n(\Sigma)$. Consider then the following two $(m+n-p)$-regsignals which, intuitively, extend R_1, resp. R_2:

$$(\overline{R}_1)_{ij} = \begin{cases} (R_1)_{ij} & \text{iff } i,j \in [1 \ldots m] \\ \Sigma^* + (\Sigma^{-1})^* & \text{otherwise} \end{cases}$$

$$(\overline{R}_2)_{ij} = \begin{cases} (R_2)_{i-m+p,j-m+p} & \text{iff } i,j \in [m-p+1 \ldots m+n-p] \\ \Sigma^* + (\Sigma^{-1})^* & \text{otherwise} \end{cases}$$

Observe that $\|\overline{R}_1\|\big|_{[1\ldots m]} = \|R_1\|$ and $\|\overline{R}_2\|\big|_{[m-p+1\ldots m+n-p]} = \|R_2\|$.
Then

$$\|R_1 \square_p R_2\| = \|\overline{R}_1 \cap \overline{R}_2\| \tag{6.29}$$

since we have:

$$\begin{aligned} \|R_1 \square_p R_2\| &= \|R_1\| \square_p \|R_2\| && \text{by proposition 6.4.4} \\ &= \|\overline{R}_1\| \cap \|\overline{R}_2\| && \text{by Proposition 6.3.3} \\ &= \|\overline{R}_1 \cap \overline{R}_2\| && \text{by Proposition 6.4.7} \end{aligned}$$

6.4.2 $2n$-domino regular expressions and $2n$-signal regular expressions

We may push the theory further by defining regular expressions whose atoms are regminoes.

Definition 6.4.8. *The class of $2n$-**domino regular expressions** is generated by the grammar:*

$$E ::= R \mid E + E \mid E \odot E \mid E^{\circledast} \tag{6.30}$$

*where R is a $2n$-regmino. When the atoms in a $2n$-domino regular expression E are all $2n$-regsignals we say that E is a $2n$-**signal regular expression**.*

We denote by $\mathsf{RegD}_{2n}(\Sigma)$ the class of $2n$-domino regular expressions over Σ and by $\mathsf{RegSig}_{2n}(\Sigma)$ the subclass of $2n$-signal regular expressions over Σ.

The semantics of a $2n$-domino regular expression is in terms of $2n$-dominoes and uses the indexed concatenations and stars:

$$\begin{aligned} \|E + E'\| &= \|E\| \cup \|E'\| \\ \|E \odot E'\| &= \|E\| \odot \|E'\| \\ \|E^{\circledast}\| &= \|E\|^{\circledast} \end{aligned}$$

Observe that the definition $\|E \odot E'\| = \|E\| \odot \|E'\|$ does not contradict the fact that the semantics of n-regminoes is noncompositional w.r.t. concatenation. The left-hand side of $\odot$ is an abstract operation on *regular expressions*, that is, its result is a regular expression, and not a regmino.

We would like to define a specific semantics of $2n$-signal regular expressions in terms of $2n$-signal languages. But $2n$-signal regular expressions cause a special problem due to the fact that

$\mathsf{Sig}_{2n}(\Sigma)$ is not closed under concatenation. We then need to restrict the semantics of each $2n$-signal regular expression to its intersection with $\mathsf{Sig}_{2n}(\Sigma)$. We denote the $2n$-signal language-semantics of a $2n$-signal regular expression as $\|\cdot\|_s$. Hence, for each $E \in \mathsf{RegSig}_{2n}(\Sigma)$,

$$\|E\|_s = \|E\| \cap \mathsf{Sig}_{2n}(\Sigma) \tag{6.31}$$

Observe that, due to Remark 6.4.2 we have that $\|R\|_s = \|R\|$ for each n-regsignal R.

Remark 6.4.9. Occasionally, we will also speak of **n-signal regular expressions**. These are nothing else but formal sums of n-regsignals, since for n odd no concatenation operation is available. This notion is useful as n-regsignals are not closed under summation.

We may also define a class of automata equivalent to $2n$-domino regular expressions, equivalence which follows via the Kleene theorem and the compositionality of $2n$-domino regular expression semantics. We call them *$2n$-regsignal automata*. We will only provide, in Figure 6.7, an example of such an automaton, the general definition being easily deducible.

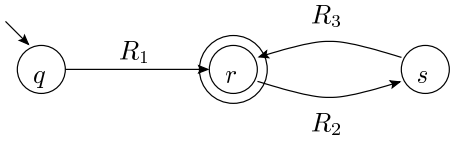


Fig. 6.7. An example of a $2n$-regsignal automata that corresponds to the $2n$-domino regular expressions $R_1 \odot (R_2 \odot R_3)^{\circledast}$, for any $2n$-regsignals R_1, R_2, R_3.

6.5 $2n$-signal regular expressions and timed automata

We have started the study of n-signals with the aim of modeling timed languages. This section provides the formalization of this modeling, namely the way the language of an n-clock timed automaton can be presented by some $2n$-signal regular expression.

In the introduction to this chapter we have intuitively presented the way to encode a signal with reset times into a $2n$-signal. The decoding of a $2n$-signal into a signal with reset times works as follows: we simply need to consider the component with the largest length in the matrix, and then distinguish some points in it, according to the timing constraints. This idea can be generalized to a definition of the *timed language* associated with a $2n$-signal regular expression, as the set of largest components that occur in some $2n$-signal which belongs to the semantics of the given $2n$-signal regular expression. More formally:

Definition 6.5.1. *Given a $2n$-signal regular expression $E \in \mathsf{RegSig}_{2n}(\Sigma)$, the **timed language associated with** E consists of the following set of signals:*

$$L(E) = \big\{\sigma \mid \exists w \in \|E\|_s, \exists i,j \in [1\ldots n], \exists \prec \text{ an ordering compatible with } w \text{ such that } w_{ij} = \sigma \text{ and } \forall k \in [1\ldots n], i \prec k \prec j\big\} \tag{6.32}$$

The following theorem formalizes the intuition that n-clocked timed automata can be presented as $2n$-signal regular expressions:

Theorem 6.5.2. *The class of languages accepted by timed automata with n clocks is included in the class of timed languages associated with some $(2n+2)$-signal regular expression.*

Proof. We will actually prove that the semantics of each n-clocked timed automaton *in which each transition resets at least one clock* can be associated with a $2n$-signal regular expression. The "$+2$" increment in the theorem statement comes from an augmentation of the number of clocks by one which is reset on each transition. Throughout this proof we will consider the reset time semantics of timed automata.

So take a timed automaton with n clocks, $\mathcal{A} = (Q, \delta, \lambda, Q_0, Q_f)$, in which each transition resets at least one clock. We code each transition $\tau = q \xrightarrow{(C,X)} r$ in which $X \neq \emptyset$ and $\lambda(q) = a$ into a $2n$-regsignal $R(\tau)$ as follows: suppose that the constraint in the atom is

$$C = \big(\bigwedge_{i \in [1..n]} x_i \in I_i \big) \wedge \big(\bigwedge_{i,j \in [1..n],\ i \neq j} x_i - x_j \in J_{ij} \big)$$

Then the components of the $2n$-regsignal $R(\tau)$ are:

$$R(\tau)_{ij} = \begin{cases} \langle \Sigma^* \cup (\Sigma^{-1})^* \rangle_{J_{ij}} & \text{for } i,j \in [1 \ldots n] \\ \langle \Sigma^* \cup (\Sigma^{-1})^* \rangle_{J_{kl}} & \text{for } i = n+k, j = n+l, k, l \in \overline{X} \\ \varepsilon & \text{for } i = n+k, j = n+l, k, l \in X \\ \langle \Sigma^* \cup (\Sigma^{-1})^* \rangle_{J_{ik}} & \text{for } i \in [1 \ldots n], j = n+k, k \in \overline{X}, i \neq k \\ \langle \Sigma^* \cup (\Sigma^{-1})^* \rangle_{J_{kj}} & \text{for } j \in [1 \ldots n], i = n+k, k \in \overline{X}, j \neq k \\ \varepsilon & \text{for } j = n+i, i \in \overline{X} \text{ or } i = n+j, j \in \overline{X} \\ \langle \Sigma^* \cdot a \rangle_{I_i} & \text{for } i \in [1 \ldots n], j = n+k, k \in X \\ & \quad \text{or } i = n+l, j = n+k, k \in X, l \in \overline{X} \\ \langle a^{-1} \cdot (\Sigma^{-1})^* \rangle_{(-I_i)} & \text{for } j \in [1 \ldots n], i = n+k, k \in X \\ & \quad \text{or } i = n+k, j = n+l, k \in X, l \in \overline{X} \end{cases}$$

Here $-I = \{-\alpha \mid \alpha \in I\}$.

Observe the utility of having $X \neq \emptyset$, since we may then code the subconstraint $x_i \in I_i$ by a comparison on the duration between the last reset point for clock x_i and the reset point of any clock in X.

Consider also the matrix $\mathcal{E}(2n)$ whose all entries are ε, $\mathcal{E}^{2n}_{ij} = \varepsilon$ for all $i, j \in [1 \ldots 2n]$. The meaning of this matrix is the following: when this matrix is concatenated to the left of a regular expression, all the starting points of the result are the same.

We then build a $2n$-regsignal automaton $\mathcal{B} = (Q \cup \{q_0\}, \theta, \{q_0\}, Q_f)$ in which

$$\theta = \big\{ q \xrightarrow{R(\tau)} r \mid \tau = q \xrightarrow{(C,X)} r \in \delta \big\} \cup \big\{ q_0 \xrightarrow{\mathcal{E}(2n)} q \mid q \in Q_0 \big\}$$

To prove that this construction is correct, observe first that each accepting run in $\mathcal{A}$ can be uniquely transformed into an accepting run in $\mathcal{B}$ that starts in $\{q_0\}$ by just appending some transition labeled with the $2n$-regsignal $\mathcal{E}(2n)$, and vice-versa.

Consider then a run $\rho = (q_i)_{i\in[1\ldots k]}$ with $\tau_i = q_i \xrightarrow{C_i,X_i} q_{i+1}$ for all $i \in [1\ldots k-1]$. We may associate to this run the following regsignal:

$$R(\rho) = R(\tau_1) \odot \ldots \odot R(\tau_{k-1})$$

Consider now the "word-like" n-clocked expression $E(\rho)$ associated with the run ρ, as defined in Identity 4.6 on page 61.

$$E(\rho) = (a, \bigwedge_{i=1}^{n} x_i = 0, \emptyset), (\lambda(q_1), C_1, X_1) \cdot \ldots \cdot (\lambda(q_{k-1}), C_{k-1}, X_{k-1})$$

Then, if a signal with reset times $\gamma = (t_1, \ldots, t_n, t, \sigma, t'_1, \ldots, t'_n, t')$ is in the (reset time) semantics of $E(\rho)$ then $t_i = t_j = t = 0$ and the signal with reset times can be represented by the following $2n$-signal:

$$\big(w(\gamma)\big)_{ij} = \begin{cases} \varepsilon & \text{for all } i,j \in [1\ldots n] \\ \sigma\big|_{[t_i,t'_k)} & \text{for } j = n+k, i,k \in [1\ldots n] \end{cases}$$

(of course, the whole $2n$-signal results with the aid of the triangle identity).

It is clear that there exists a bijection between the set of signals with reset times with $t_1 = \ldots = t_n = t = 0$ and $t'_i \geq 0$ and the set of $2n$-signals $w \in \mathsf{Sig}_{2n}(\Sigma)$ with $w_{ij} = \varepsilon$ for all $i,j \in [1\ldots n]$. We may apply this bijection to the n-clocked semantics of $E(\rho)$ and hence get a $2n$-*signal semantics* for it.

The proof ends if we show that, for each run ρ, this $2n$-signal semantics for $E(\rho)$ equals the semantics of the $2n$-signal regular expression $\mathcal{E}(2n) \odot R(\rho)$. This will be proved by induction on the length of the run.

For zero-length runs the proof is trivial. Let us suppose then that we have proved the property for all runs of length up to $k-1$ and take some run of length k, say $\rho = (q_i)_{i\in[1\ldots k+1]}$, with $\tau_i = q_i \xrightarrow{(C_i,X_i)} q_{i+1} \in \delta$ for all $i \in [1\ldots k]$. Denote also $\rho' = (q_i)_{i\in[1\ldots k]}$, the run reduced to the first $k-1$ steps. Hence we have

$$E(\rho) = E(\rho') \cdot (a_{k+1}, C_{k+1}, X_{k+1})$$

Then each signal with reset times in the reset semantics of $\mathcal{E}(\rho)$ can be decomposed as

$$(0, \ldots, 0, \sigma, t_1, \ldots, t_n, t) \cdot (t_1, \ldots, t_n, t, a_{k+1}, t'_1, \ldots, t'_n, t')$$

where $\gamma = (0, \ldots, 0, \sigma, t_1, \ldots, t_n, t) \in \|E(\rho)\|$ and $\gamma' = (t_1, \ldots, t_n, t, a_{k+1}, t'_1, \ldots, t'_n, t') \in \|(a_{k+1}, C_{k+1}, X_{k+1})\|$.

We may build then the $2n$-signal $w(\gamma)$ associated with γ, which, by induction hypothesis, is in the semantics of $\mathcal{E}(2n) \odot R(\rho')$. We further consider the following $2n$-signal which, intuitively, is associated with γ':

$$w'_{ij} = \begin{cases} w(\gamma)_{n+i,n+j} & \text{iff } i,j \in [1 \dots n] \\ w(\gamma)_{n+i,j} \cdot a_{k+1} & \text{iff } i \in [1 \dots n], j \in [n+1 \dots 2n] \end{cases} \tag{6.33}$$

the rest being derivable by the triangle identity.

Observe that $w(\gamma) \odot w'$ is defined and the result is

$$w(\gamma) \odot w' = w(\gamma \cdot \gamma')$$

which proves in fact that the $2n$-signal semantics of $E(\rho)$ is included in the semantics of $\mathcal{E}(2n) \odot R(\rho)$.

For the reverse inclusion we proceed by mirroring the above argument: for the induction step, consider a $2n$-signal $z \in \|\mathcal{E}(2n) \odot R(\rho)\| = \|\mathcal{E}(2n) \odot R(\rho') \odot R(a_{k+1}, C_{k+1}, X_{k+1})\|$. Hence, $z = w \odot w'$ with $w \in \|\mathcal{E}(2n) \odot R(\rho)\|$ and $w' \in \|R(a_{k+1}, C_{k+1}, X_{k+1})\|$. By the induction hypothesis, w is in the $2n$-signal semantics of $E(\rho')$, hence $w = w(\gamma)$ for some signal with reset times $\gamma = (0, \dots, 0, \sigma, t_1, \dots, t_n, t) \in \|E(\rho')\|$. We may then build a signal with reset times from the information provided by w' as follows: $\gamma' = (t_1, \dots, t_n, t, a_{k+1}, t'_1, \dots, t'_n, t')$ where

$$\begin{aligned} t'_i &= t_i + \ell(w'_{i,n+i}) \text{ for each } i \in [1 \dots n] \\ t' &= t'_i \text{ for some } i \in X_{k+1} \end{aligned}$$

But $\gamma \cdot \gamma'$ is defined and produces the signal with reset times $\gamma'' = (0, \dots, 0, \sigma, t'_1, \dots, t'_n, t') \in \|E(\rho)\|$ which clearly has the property that $w(\gamma'') = z$. □

6.6 The emptiness problem for $2n$-signal regular expressions is undecidable

In this section we show that our regular expressions over regsignals have an undecidable emptiness problem, hence being more expressive than timed automata. In particular, we show here how to encode each instance of the Post Correspondence Problem into a $2n$-signal regular expression. Interestingly, the problem comes from the "untimed" part, the time playing no role in this result.

We remind here briefly the *Post Correspondence Problem* [Pos46] and the result concerning its undecidability:

Definition 6.6.1. *A **PCP instance** consists of a finite list $\big((u_i, v_i)\big)_{i \in [1 \dots p]}$ of pairs of words, $u_i, v_i \in \Sigma^*$. A **solution** of this instance consists of a finite list of indices $(i_j)_{j \in [1 \dots p]}$ such that*

$$u_{i_1} u_{i_2} \dots u_{i_p} = v_{i_1} v_{i_2} \dots v_{i_p}$$

The *Post Correspondence Problem* is the problem of checking whether a given PCP instance has a solution.

Theorem 6.6.2 ([Pos46, HU92]). *The Post Correspondence Problem is undecidable.*

Theorem 6.6.3. *The emptiness problem for $2n$-signal regular expressions is undecidable.*

Proof. We encode each PCP instance into a 4-signal regular expression. Hence, supposing we are given the instance $\big((x_i, y_i)\big)_{i\in[1\dots p]}$, we associate to each PCP-domino (x_i, y_i) the following 4-regsignal:

$$R_i = \begin{pmatrix} \varepsilon & \Sigma^* + (\Sigma^{-1})^* & x_i & \Sigma^* + (\Sigma^{-1})^* \\ \Sigma^* + (\Sigma^{-1})^* & \varepsilon & \Sigma^* + (\Sigma^{-1})^* & y_i \\ x_i^{-1} & \Sigma^* + (\Sigma^{-1})^* & \varepsilon & \Sigma^* + (\Sigma^{-1})^* \\ \Sigma^* + (\Sigma^{-1})^* & y_i^{-1} & \Sigma^* + (\Sigma^{-1})^* & \varepsilon \end{pmatrix} \tag{6.34}$$

Then, by using the 4-regsignal $\mathcal{E}(4)$ defined in the proof of Theorem 6.5.2,

$$\big\|\mathcal{E}(4) \odot \big(\sum_{i=1}^{p} R_i\big)^{\circledast} \odot \mathcal{E}(4)\big\| \neq \emptyset$$

iff the given PCP has a nontrivial solution.

To observe this, consider first a 4-signal $w \in \big\|\mathcal{E}(4) \odot \big(\sum_{i=1}^{p} R_i\big)^{\circledast} \odot \mathcal{E}(4)\big\|$. By definition, we have $w = w_0 \odot w_1 \odot \dots \odot w_k \odot w_{k+1}$ with $w_0, w_{k+1} \in \|\mathcal{E}(4)\|$ and $w_j \in \|R_{l_j}\|$ for all $j \in [1\dots k]$ and $l_j \in [1\dots p]$. This implies that $(w_j)_{13} = u_{l_j}$ and $(w_j)_{24} = v_{l_j}$ for all $j \in [1\dots k]$, and, by construction of $\mathcal{E}(4)$, that $(w_0)_{12} = (w_{k+1})_{34} = \varepsilon$.

We first need to prove that $w_0 \odot \dots \odot w_{k+1}$ is a 4-signal. The following proposition will help us:

Proposition 6.6.4. *Given two 4-signals $z, z' \in \mathsf{Sig}_4(\Sigma)$, suppose that $z_{13}, z_{24}, z'_{13}, z'_{24}$ are not antisignals, that is, $z_{13}, z_{24}, z'_{13}, z'_{24} \in \mathsf{Sig}(\Sigma)$. Suppose also that $z\Box_2 z'$ is defined. Then $z\Box_2 z'$ is a 6-signal.*

Proof. The proof of this property is done by case study on the possible orderings which are compatible with z, respectively with z'.

For each of the two 4-signals, they are 6 cases that can occur, under the assumption that the $(1,3)$-components and the $(2,4)$-components are not antisignals, three when $3 \prec 4$:

$$(a) 1 \prec 2 \prec 3 \prec 4, \qquad (b) 2 \prec 1 \prec 3 \prec 4, \qquad (c) 1 \prec 3 \prec 2 \prec 4$$

and the 3 symmetric cases in which $4 \prec 3$. Since not all combinations are possible due to the need to have $z\Box_2 z'$ defined, the correct combinations are as much as 18. Let us denote $\prec$ the ordering compatible with z and $\prec'$ the ordering compatible with z' and suppose that $3 \prec 4$, hence $1 \prec' 2$ due to correct juxtaposition.

The only problematic components of $z\Box_2 z'$ are $(z\Box_2 z')_{16}$ and $(z\Box_2 z')_{25}$. To see that the other components are indeed signals or antisignals, observe that $(z\Box_2 z')_{15} = z_{13}z'_{13} \in \mathsf{Sig}(\Sigma)$ and $(z\Box_2 z')_{26} = z_{24}z'_{24} \in \mathsf{Sig}(\Sigma)$ by hypothesis.

1. For the six cases in which $2 \prec' 3$ and $2 \prec' 4$, that is, when $z'_{23}, z'_{24} \in \mathsf{Sig}(\Sigma)$, we have that

$$(z \square_2 z')_{16} = z_{13} z'_{14} = z_{13} z'_{12} z'_{24} \in \mathsf{Sig}(\Sigma)$$
$$(z \square_2 z')_{25} = z_{24} z'_{23} \in \mathsf{Sig}(\Sigma)$$

   By similarity, the other two cases in which $1 \prec 3$ and $2 \prec 3$ are also solved.
2. For the last remaining case, when $1 \prec 3 \prec 2 \prec 4$ and $1 \prec' 3 \prec' 2 \prec' 4$, observe first that we get $z'_{14} \in \mathsf{Sig}(\Sigma)$, and therefore

$$(z \square_2 z')_{16} = z_{13} z'_{14} \in \mathsf{Sig}(\Sigma)$$

   we observe that

$$z_{32} z_{24} = z'_{13} z'_{32}$$

   But the four factors of this identity are signals, and signals have the following *equidivisibility property*:

$$\text{there exists } \sigma \in \mathsf{Sig}(\Sigma) \text{ such that } \begin{cases} \text{either} & z_{32} = z'_{13}\sigma \text{ and } z'_{32} = \sigma z_{24} \\ \text{or} & z_{24} = \sigma z'_{32} \text{ and } z'_{13} = z_{32}\sigma \end{cases}$$

   Graphically, the two possibilities are depicted in Figure 6.8

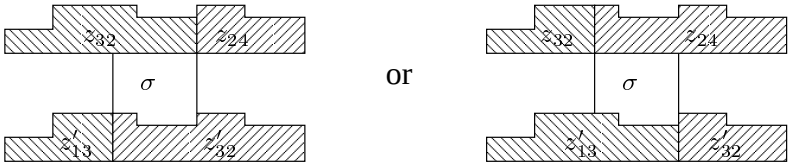


Fig. 6.8. The equidivisibility property.

   Let us consider the first variant, that is, $z_{32} = z'_{13}\sigma$ and $z'_{32} = \sigma z_{24}$. It follows that:

$$(z \square_2 z')_{25} = z_{23} z'_{13} = \sigma^{-1} (z'_{13})^{-1} z'_{13} = \sigma^{-1} \in \mathsf{Sig}(\Sigma^{-1})$$

   We have already seen that $(z \square_2 z')_{16} \in \mathsf{Sig}(\Sigma)$, hence none of the components of $z \square_2 z'$ is a mixture of signals and antisignals, which means that $z \square_2 z' \in \mathsf{Sig}_4(\Sigma)$. (Observe that, in this case, on $z \square_2 z'$, we may choose the compatible ordering $1 \prec'' 3 \prec'' 4 \prec'' 5 \prec'' 2 \prec'' 6$.)
   The same result follows if we choose the second variant, that is $z_{24} = \sigma z'_{32}$ and $z'_{13} = z_{32}\sigma$. □

Proof (of Theorem 6.6.3, continued). We may prove, by induction on j and by means of Proposition 6.6.4, that $w_0 \odot \ldots \odot w_j$ is a 4-signal.

On the other hand, if we explicitely build w from $w_0, \ldots, w_{k+1}$ we get that

$$w_{13} = (w_0)_{13} \cdot (w_1)_{13} \cdot \ldots \cdot (w_k)_{13} \cdot (w_{k+1})_{13} = u_{l_1} u_{l_2} \ldots u_{l_k}$$
$$w_{24} = (w_0)_{24} \cdot (w_1)_{24} \cdot \ldots \cdot (w_k)_{24} \cdot (w_{k+1})_{24} = v_{l_1} v_{l_2} \ldots v_{l_k}$$

Then, the triangle identity 6.1 implies that $w_{23} = w_{21} w_{13} = w_{24} w_{43}$ and therefore $w_{13} = w_{24}$, fact which assures that $\big(l_j\big)_{j \in [1 \ldots k]}$ is a solution of the given PCP instance.

For the reverse implication, suppose now that the PCP instance $(u_i, v_i)_{i \in [1 \ldots p]}$ has a solution $\big(l_j\big)_{j \in [1 \ldots k]}$. Let us denote, for simplicity, for each $j \in [1 \ldots k]$, $\tilde{u}_{l_j} = u_{l_j} \cdot \ldots \cdot u_{l_k}$ and $\tilde{v}_{l_j} = v_{l_j} \cdot \ldots \cdot v_{l_k}$.

We build a sequence $(w_j)_{j \in 1 \ldots k]}$ of 4-signals which, intuitively, record the positioning of the j-th domino in the chain of concatenations. Formally:

$$w_j = \begin{pmatrix} \varepsilon & \tilde{u}_{l_j} \cdot (\tilde{v}_{l_j}^{-1}) & u_{l_j} & \tilde{u}_{l_j} \cdot (\tilde{v}_{l_{j+1}}^{-1}) \\ \big(\tilde{u}_{l_j} \cdot (\tilde{v}_{l_j}^{-1})\big)^{-1} & \varepsilon & \tilde{u}_{l_{j+1}} \cdot (\tilde{v}_{l_j}^{-1}) & v_{l_j} \\ u_{l_j}^{-1} & \big(\tilde{u}_{l_j} \cdot (\tilde{v}_{l_{j+1}}^{-1})\big) - 1 & \varepsilon & \tilde{u}_{l_{j+1}} \cdot (\tilde{v}_{l_{j+1}}^{-1}) \\ \big(\tilde{u}_{l_j} \cdot (\tilde{v}_{l_{j+1}}^{-1})\big)^{-1} & v_{l_j}^{-1} & \big(\tilde{u}_{l_{j+1}} \cdot (\tilde{v}_{l_{j+1}}^{-1})\big)^{-1} & \varepsilon \end{pmatrix}$$

For example, $(w_j)_{23}$ holds the word or antiword that lies in between the occurrence of u_{l_j} and v_{l_j}.

The fact that $\big(l_j\big)_{i \in [1 \ldots k]}$ is a solution implies that, for each $i \in [1 \ldots k]$, $\tilde{u}_{l_j} \cdot (\tilde{v}_{l_j}^{-1})$ is either a word or an antiword. Hence $w_j \in \|R_j\|$.

It is then easy to check by induction that $w_j\big|_{\{3,4\}} = w_{j+1}\big|_{\{1,2\}}$ and that the concatenation $w_1 \odot \ldots \odot w_k$ is a 4-signal. The proof is accomplished if we observe that $(w_1)_{12} = (w_k)_{34} = \varepsilon$, hence we may concatenate at left and right with the matrix $\mathcal{E}(4)$, viewed this time as a 4-signal, to get that

$$\mathcal{E}(4) \odot w_1 \odot \ldots \odot w_k \odot \mathcal{E}(4) \in \Big\| \mathcal{E}(4) \odot \big(\sum_{i=1}^{p} R_i \big)^{\circledast} \odot \mathcal{E}(4) \Big\| \qquad \square$$

Note that the problems concerning the semantics of $2n$-signal regular expressions, problems due to nonclosure of $\mathsf{Sig}_{2n}(\Sigma)$ under concatenation, are harmless for the proof of this theorem.

Corollary 6.6.5. *$2n$-signal regular expressions are strictly more expressive than timed automata.*

Throughout the following chapters we will search for the following two things:

- A subclass of $2n$-signal regular expressions having a decidable emptiness problem and the same expressive power as timed automata and
- A discrete representation of this class, which allows manipulating only untimed $2n$-signals.

The search will proceed hand-in-hand, since the discrete representation for the subclass will eventually lead to the decision procedure.

7. n-words and their automata

A closer look at Theorem 6.6.3 shows that time plays no rôle in the undecidability of the emptiness problem. It is only the untimed structure of n-dominoes that gives the possibility to encode PCP instances into 4-domino regular expressions. We therefore need to study in deeper detail this untimed structure, that is, untimed n-dominoes and untimed n-signals. We will call the latter as n-*words*. Actually, the whole theory of juxtaposition and concatenation might have been introduced on untimed dominoes and n-words, but we have preferred introducing it for signals in order to justify its utility for the study of timed automata.

We investigate in this chapter a class of finite automata that is naturally associated with these n-words. We will call these automata as n-*automata*. The idea is to have n accepting sets, such that a run accepts an n-word iff it passes through an accepting set exactly when it crosses one of the distinguished points in the n-word. Of course, the accepting sets are indexed, such that when crossing the distinguished point i, the i-th accepting set is reached. In the matrix presentation of n-words, this is rephrased as follows: the run in between the moment of passing through the i-th accepting set and the moment of passing through the j-th accepting set is labeled with w_{ij}. Or, in the case w_{ij} is an *antiword*, its inverse w_{ij}^{-1} labels the run between the moment of passing through the j-th accepting set and the moment of passing through the i-th accepting set.

We show here that n-automata are as expressive as sums of n-*regwords* (that is, sums of untimed n-regsignals) and that they are closed under concatenation. We also show that they have a decidable emptiness problem, though with a high complexity solution (in the NP class [GJ79]).

This allows us to identify better what harms the emptiness problem for $2n$-word regular expressions: it is the star operation in combination with the *elasticity* of 4-regwords that represent each PCP domino. By elasticity we name the property that, for some 4-regword which represents a PCP domino, allows the two words in the domino to be arbitrarily far away from one other. Our idea is then to forbid this elasticity both at the untimed and timed level and to show that, when simulating timed automata with $2n$-signal regular expressions, we obtain non-elastic $2n$-regsignals too.

Non-elasticity does not prove to be a nice algebraic property since it is not closed under concatenation. But our search is for a property that assures decidability rather than for a class of $2n$-word regular expressions which is decidable, since such a property can be checked on different classes of algebraically closed classes of $2n$-signal languages.

One question might be asked here: why do we "complicate" our life and use a fussy class of automata and not work with classical finite automata and the intersection construction? But in fact, our class of automata is nothing else but a compact representation of an "asynchronous"

composition of finite automata. Then, in a certain sense, our non-elasticity property requires a bound on the asynchronicity in order for the emptiness problem to become decidable. Even more, our class of automata will be able to represent also timing constraints *over the continuous time domain*, as we will see in the next chapter.

This chapter runs as follows: in the first section, we define n-words, n-regwords and the regular expressions over $2n$-regwords, and show that all the algebraic properties of n-signals and n-regsignals from Chapter 6 hold for n-words and n-regwords. The second section contains the definition of n-automata and their basic closure properties, must notably the closure under projection and juxtaposition. We also show here that the emptiness problem for n-automata is decidable and that n-automata are equivalent to n-regwords. The third section serves for the introduction of the non-elasticity property and some basic observations on it. The fourth section contains the main result of this chapter, the star closure property of $2n$-automata whose accepted languages have the property that all their powers are non-elastic $2n$-word languages.

7.1 n-words

n-words can be thought as words with distinguished points, similarly to n-signals. Hence, when presenting words with distinguished points, we must employ *antiwords*, that is, words over the set of symbols $\Sigma^{-1} = \{a^{-1} \mid a \in \Sigma\}$. Algebraically, we work on the *free group* generated by the set of symbols Σ, which is nothing else but the set $(\Sigma \cup \Sigma^{-1})^*$, endowed with a concatenation operation which "cancels" inverse letters.

Definition 7.1.1. *An **untimed** n-**domino** is a matrix $w = (w_{ij})_{i,j\in[1\dots n]}$ of elements from $(\Sigma\cup\Sigma^{-1})^*$ which satisfies the triangle identity 6.1, that is,*

$$\text{for all } i,j,k \in [1\dots n], w_{ij}w_{jk} = w_{ik}$$

*When $w_{ij} \in \Sigma^* \cup (\Sigma^{-1})^*$ we say that w is an n-**word** over Σ.*

A graphical representation of a 3-word is given in Figure 7.1:

$$W = \begin{pmatrix} \varepsilon & a & abab \\ a^{-1} & \varepsilon & bab \\ b^{-1}a^{-1}b^{-1}a^{-1} & b^{-1}a^{-1}b^{-1} & \varepsilon \end{pmatrix} =$$

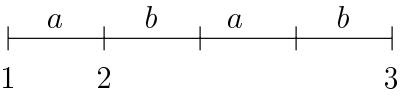


Fig. 7.1. A 3-word and its graphical representation.

The whole theory of projection/juxtaposition/concatenation will be used directly for n-words ($2n$-words where needed) without rephrasing, as it can be easily adapted to the untimed structure. We translate in this introduction all the notations and results for the untimed case:

The set of n-words is denoted $\mathcal{WD}_n(\Sigma)$ while the set of untimed n-dominoes over Σ is denoted $\mathcal{UD}_n(\Sigma)$. Note that juxtaposition of n-words does not necessarily yield n-words.

An n-**word language** is any subset of $\mathcal{WD}_n(\Sigma)$. Similarly to $2n$-signal languages, the set of $2n$-word languages can be given a Kleene algebra structure with the concatenation inherited from $2n$-words and the resulting star operation.

An n-**regword** R is then a $n \times n$ matrix whose entries are (untimed) regular expressions over $\Sigma \cup \Sigma^{-1}$. The set of n-regwords is denoted $\mathcal{RW}_n(\Sigma)$.

The **semantics** of an n-regword consists of untimed n-dominoes with the property that $w_{ij} \in R_{ij}$ for each $i, j \in [1 \dots n]$ and is denoted $\|R\|$:

$$\|R\| = \left\{ w \in \mathcal{UD}_n(\Sigma) \mid w_{ij} \in |R_{ij}| \text{ for all } i, j \in [1 \dots n] \right\}$$

Remind that $|E|$ denotes the semantics of the classical regular expression E.

Similarly to n-regsignals, n-regwords semantics is not compositional w.r.t. projection but is compositional w.r.t. juxtaposition.

For each n-regword $R \in \mathcal{WD}_n(\Sigma)$ we will denote R^+_{ij} the "positive part" of the (i,j)-component of R and R^-_{ij} the "negative part" of the (i,j)-component of R, that is,

$$R_{ij} = R^+_{ij} + R^-_{ij} \text{ with } R^+_{ij} = R_{ij} \cap \Sigma^*, R^-_{ij} = R_{ij} \cap (\Sigma^{-1})^*$$

In fact, we will utilize this decomposition for both R^+_{ij} and R^-_{ij} being regular expressions that denote respectively $R_{ij} \cap \Sigma^*$ and $R_{ij} \cap (\Sigma^{-1})^*$.

The set of $2n$-**word regular expressions** is defined by the following grammar:

$$E ::= R \mid E + E \mid E \odot E \mid E^{\circledast} \tag{7.1}$$

where R is any $2n$-regword. Their semantics is based upon the $2n$-word language operations as usually:

$$\begin{aligned} \|E + E'\| &= \|E\| \cup \|E'\| \\ \|E \odot E'\| &= \|E\| \odot \|E'\| \\ \|E^{\circledast}\| &= \|E\|^{\circledast} \end{aligned}$$

All the properties that hold for n-regminoes and n-regsignals will also hold for untimed n-regminoes and n-regwords.

Remark 7.1.2. Proposition 6.4.7 and identity 6.29 hold for untimed regminoes, since any intersection of untimed regular expressions over $\Sigma \cup \Sigma^{-1}$ can be transformed into a regular expression over $\Sigma \cup \Sigma^{-1}$.

7.2 n-automata

We define here a class of finite automata that are equivalent to n-regwords. The idea is to generalize from finite automata by utilizing n sets of accepting states and requiring that the accepting runs

pass at least once through each of these sets. The class can be generalized to support also untimed n-regminoes but we will not present this generalization.

Definition 7.2.1. *An n-**automaton** over an alphabet Σ is a tuple $\mathcal{A} = (Q, \delta, Q_1, \dots, Q_n)$ in which Q is the finite set of states, $\delta \subseteq Q \times \Sigma \times Q$ is the transition function, and for each $i \in [1 \dots n]$, $Q_i \subseteq Q$ is the set of accepting states for index i.*

A *run* in such an automaton is simply a sequence of transitions $(q_j, a_j, q_{j+1})_{i\in[1\dots k]}$ with $(q_j, a_j, q_{j+1}) \in \delta$ for all $i \in [1 \dots k]$. We also have word-labeled transitions, as in finite automata: $q \xrightarrow{w} q'$ if there exists a sequence of transitions from q to q' whose concatenation of labels gives w. For any run $\rho = (q_j, a_j, q_{j+1})_{i\in[1\dots k]}$ and two indices $i_1, i_2 \in [1 \dots k+1]$, we denote $\mathsf{word}(\rho, i_1, i_2)$ the word or antiword which labels the transitions in between the i_1-th state and the i_2-th state in the run:

$$\mathsf{word}(\rho, i_1, i_2) = \begin{cases} a_{i_1} a_{i_1+1} \dots a_{i_2-1} & \text{iff } i_1 \le i_2 \\ a_{i_2-1}^{-1} a_{i_2-2}^{-1} \dots a_{i_1}^{-1} & \text{iff } i_1 > i_2 \end{cases} \tag{7.2}$$

By mirroring this definition we also get antiword-labeled transitions: for $w \in (\Sigma^{-1})^*$, $q \xrightarrow{w} q'$ if $q' \xrightarrow{w^{-1}} q$.

An **accepting run** is a run $\rho = (q_j, a_j, q_{j+1})_{j\in[1\dots k]}$ that passes through each accepting set, i.e.,

$$\text{for each } i \in [1 \dots n],\ \{q_1, \dots, q_k\} \cap Q_i \neq \emptyset$$

Given an accepting run $\rho = (q_j, a_j, q_{j+1})_{j\in[1\dots k]}$ and a set of n indices within this run, $\boldsymbol{l} = (l_i)_{i\in[1\dots n]}$ with $l_i \in [1 \dots k]$, such that $q_{l_i} \in Q_i$ for all $i \in [1 \dots n]$, an n-word $w \in \mathcal{WD}_n(\Sigma)$ is said to be **accepted** by the run ρ and the index sequence $\boldsymbol{l}$ iff

$$\text{for each } i, j \in [1 \dots n],\ q_{l_i} \xrightarrow{w_{ij}} q_{l_j},\ \text{that is, } w_{ij} = \begin{cases} a_{l_i} a_{l_i+1} \dots a_{l_j-1} & \text{iff } l_i \le l_j \\ a_{l_i-1}^{-1} a_{l_i-2}^{-1} \dots a_{l_j}^{-1} & \text{iff } l_i > l_j \end{cases}$$

We say that the sequence of indices $\boldsymbol{l}$ **witnesses** the acceptance of the n-word by the run ρ.

A first example is provided in the Figures 7.2 and 7.3:

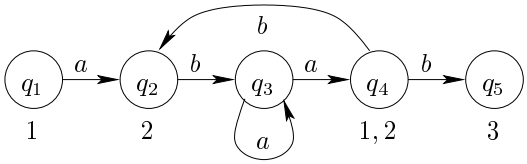


Fig. 7.2. An example of a 3-automaton. The accepting sets are $Q_1 = \{q_1, q_4\}$, $Q_2 = \{q_2, q_4\}$ and $Q_3 = \{q_5\}$.

The 3-automaton in Figure 7.2 accepts the 3-word in Figure 7.3 (a): the associated run is $\big((q_1, a, q_2), (q_2, b, q_3), (q_3, a, q_4), (q_4, b, q_5)\big)$ and the witnessing sequence $(1, 2, 5)$. Note that the same run, but with the witnessing sequence $(4, 2, 5)$ accepts the 3-word in Figure 7.3 (b).

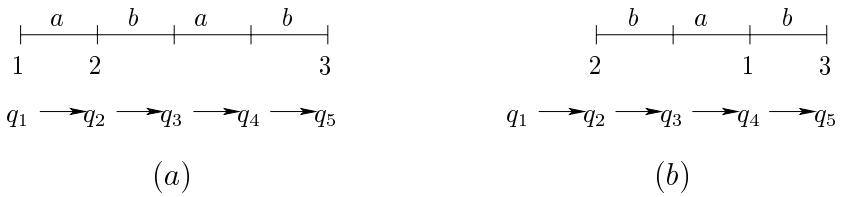


Fig. 7.3. Two 3-words accepted by the automaton in Figure 7.2. The accepting runs are depicted below each word. The witnessing sequences can be retrieved by identifying the indices in the run which correspond to the distinguished positions inside the 3-words.

Remark 7.2.2. Observe that a run might be longer in both directions than the word actually accepted by it. For example, the 3-word in Figure 7.3 (b) might be accepted by a shorter run, namely $\big((q_2, b, q_3), (q_3, a, q_4), (q_4, b, q_5)\big)$ in combination with the witnessing sequence $(3, 1, 4)$.

Hence in any n-automaton we may consider only runs that start and end in some accepting set, in pair with witnessing sequences that contain the index 1 and the final index in the run.

Remark 7.2.3. Observe also that the witnessing sequence does not necessarily capture *all* the moments when the accepting run passes through an accepting set. In Figure 7.3, the two runs pass twice through Q_2, but only once this pass is really needed and used.

There exists an alternative way of accepting n-words: we may define an accepting run as a sequence of tuples $\rho = \big((X_i, q_i, X_i'), a_i, (X_{i+1}, q_{i+1}, X_{i+1}')\big)_{i\in[1\ldots k]}$ with the following properties:

- $(q_i, a_i, q_{i+1}) \in \delta$.
- For each $i \in [1 \ldots k-1]$, $X_i' = X_{i+1}$.
- For each $i \in [1 \ldots k]$, if $j \in X_i' \setminus X_i$ then $q_i \in Q_j$.
- $X_1 = \emptyset$ and $X_{k+1} = [1 \ldots n]$.

The components X_i record the "history" of passing through accepting sets up to the i-th state while the X_i' components also take into account the i-th state. The translation from the "witnessing" presentation to the "history" presentation is straightforward: given an accepting run ρ and a witnessing sequence $(l_i)_{i\in[1\ldots k]}$, we construct the "history" components as $X_i = \{l_j \mid j < i\}$ and $X_i' = \{l_j \mid j \leq i\}$. For the reverse, given an accepting run in the "history" presentation $\rho = \big((X_i, q_i, X_i'), a_i, (X_{i+1}, a_{i+1}, X_{i+1})\big)_{i\in[1\ldots k]}$ we associate the witnessing sequence $(l_i)_{i\in[1\ldots n]}$ in which $l_i = j$ iff $j \in X_i' \setminus X_i$.

Definition 7.2.4. *The n-**word language accepted by** $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of n-words accepted by some accepting run in $\mathcal{A}$, together with some witnessing set of indices.*

*The class of **regular** n-**languages** consists of the family of n-word languages which can be accepted by some n-automaton.*

7.2.1 The emptiness problem for n-automata

Proposition 7.2.5. *The problem of checking whether the language of an n-automaton is nonempty is an NP-complete problem.*

Proof. Let us first show that the non-emptiness problem is NP-easy. To this end, consider the following "nondeterministic algorithm" (in the sense of [HU92]) that associates to each state $q \in Q$ an index set $X \subseteq [1 \dots n]$ with the property that there exists a run that starts anywhere, ends in q and passes through each of the accepting sets whose indices are in X:

- pick $q \in Q$;
- put $X := \{i \in [1 \dots n] \mid q \in Q_i\}$;
- put $S := \emptyset$ - put $T := \{(q, X)\}$;
- `while` $X \neq [1 \dots n]$ `and` $S \neq T$ `do`
  - put $S := T$;
  - pick $(q, X) \in T$;
  - pick $a \in \Sigma$;
  - pick $r \in Q$;
  - `if` $(q, a, r) \notin \delta$ `then stop`;
  - compute the new $X := X \cup \{i \in [1 \dots n] \mid r \in Q_i\}$;
  - compute the new $T := \big(T \setminus \big(\{r\} \times \mathcal{P}([1 \dots n])\big)\big) \cup \{(r, X)\}$;

  `endwhile`;

`if` $X = [1 \dots n]$ then `write(''nonempty'')`
`else write(''empty'')`.

This nondeterministic algorithm runs in polynomial time and linear space in the size of the given n-automaton. It is clear that, if there exists an accepting run, then one of the choices in this algorithm will find it.

For the NP-hardness part, we show that the Hamiltonian Path Problem (HP) can be polynomially reduced to checking emptiness of an n-automaton. Remind that the *Hamiltonian Path Problem* [GJ79] is the problem whether a given directed graph contains a path which visits each node exactly once.

The polynomial reduction of HP to the emptiness problem for n-automata is the following: given a graph $G = (V, E)$ with $V = \{v_1, \dots, v_k\}$, we construct the following k-automaton: $\mathcal{A} = (V \times [1 \dots k], \delta, Q_1, \dots, Q_k)$ where

$$\delta = \big\{(v, j) \to (v', j + 1) \mid (v, v') \in E, j \in [1 \dots k - 1]\big\}$$
$$Q_i = \big\{(v_i, j) \mid j \in [1 \dots k]\big\}$$

We then have that each accepting run through $\mathcal{A}$ corresponds to a Hamiltonian path in G and vice-versa. This follows since an accepting run must have k nodes as it visits each accepting set at least once and each visit increases the "level" of the node by one, and also the number of sets visited is less or equal than the number of nodes in the run. □

We present here an algorithm that is an adaptation of the Floyd-Warshall-Kleene algorithm, hence containing $O\big(card(Q)^3\big)$ *iterations*, but each iteration might take exponential time since it involves operations on possibly exponentially many *subsets* of $[1 \dots n]$. It associates to each pair

(q, r) of states in the given n-automaton, a set of subsets Ξ_{qr} of $[1 \dots n]$. The set Ξ_{qr} has the property that, for each $X \in \Xi_{qr}$ there exists a run from q to r that passes through each Q_i for each $i \in X$. Once the matrix Ξ is constructed, the answer is "YES" if and only if there exists one component (q, r) with $[1 \dots n] \in \Xi_{qr}$.

For the computation of Ξ, we suppose an ordering of Q is given, say $Q = \{q_1, \dots, q_p\}$ with $p \in \mathbb{N}$. A special operation on $\mathcal{P}\big(\mathcal{P}([1 \dots n])\big)$, denoted $\otimes$, is used. This operation works as follows: given $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{P}([1 \dots n])$,

$$\mathcal{X} \otimes \mathcal{Y} = \big\{ Z_1 \cup Z_2 \mid Z_1 \in \mathcal{X}, Z_2 \in \mathcal{Y} \big\}$$

Remark 7.2.6. $\otimes$ is associative.

The algorithm works by constructing a sequence of matrices $(\Xi_k)_{k \in [0 \dots p]}$ with

$$(\Xi_0)_{ij} = \begin{cases} \{X\} & \text{iff } i = j \text{ and for all } l \in X, q_i \in Q_l \\ \{\emptyset\} & \text{otherwise} \end{cases}$$

$$(\Xi_{k+1})_{ij} = (\Xi_k)_{i,k+1} \otimes (\Xi_k)_{k+1,k+1} \otimes (\Xi_k)_{k+1,j}$$

Proposition 7.2.7. *For each $i, j \in [1 \dots n]$ and $k \in [0 \dots p]$, $X \in (\Xi_k)_{ij}$ iff there exists a run that starts in q_i, ends in q_j, whose intermediary states are labeled with indices less than or equal to k, and which passes through all accepting sets Q_l for each $l \in X$.*

Proof. By induction on k.

For $k = 0$ the proof is trivial. Suppose we have proved the result for k. Then, for each $i, j \in [1 \dots n]$ and $X \in (\Xi_{k+1})_{ij}$, by associativity we have that $X = Z_1 \cup Z_2 \cup Z_3$ with $Z_1 \in (\Xi_k)_{i,k+1}, Z_2 \in (\Xi_k)_{k+1,k+1}$ and $Z_3 \in (\Xi_k)_{k+1,j}$. By induction hypothesis we get the following three runs:

- A run $\rho_1 = (r_h^1, a_h^1, r_{h+1}^1)_{h \in [1 \dots m_1]}$ with
  1. $r_1^1 = q_i$, $r_{m_1+1}^1 = q_{k+1}$;
  2. $r_h^1 \in \{q_1, \dots, q_k\}$ for all $h \in [2 \dots m_1]$;
  3. For each $l \in Z_1$ there exists $h \in [1 \dots m_1]$ such that $r_h^1 \in Q_l$.
- A run $\rho_2 = (r_h^2, a_h^2, r_{h+1}^2)_{h \in [1 \dots m_2]}$ with
  1. $r_1^2 = q_{k+1}$, $r_{m_2+1}^2 = q_{k+1}$;
  2. $r_h^2 \in \{q_1, \dots, q_k\}$ for all $h \in [2 \dots m_2]$.
  3. For each $l \in Z_2$ there exists $h \in [1 \dots m_2]$ such that $r_h^2 \in Q_l$.
- A run $\rho_3 = (r_h^3, a_h^3, r_{h+1}^3)_{h \in [1 \dots m_3]}$ with
  1. $r_1^3 = q_{k+1}$, $r_{m_3+1}^3 = q_j$;
  2. $r_h^3 \in \{q_1, \dots, q_k\}$ for all $h \in [2 \dots m_3]$;
  3. For each $l \in Z_3$ there exists $h \in [1 \dots m_3]$ such that $r_h^3 \in Q_l$.

But then, by concatenating these three runs we obtain the run

$$\rho = \Big((r_h^1, a_h^1, r_{h+1}^1)_{h \in [1 \dots m_1]}, (r_h^2, a_h^2, r_{h+1}^2)_{h \in [1 \dots m_2]}, (r_h^3, a_h^3, r_{h+1}^3)_{h \in [1 \dots m_3]} \Big)$$

which verifies the claimed property. The inverse implication follows similarly. □

Corollary 7.2.8. *$L(\mathcal{A}) \neq \emptyset$ iff there exist $i, j \in [1 \dots p]$ such that $[1 \dots n] \in (\Xi_p)_{ij}$.*

The advantage of this algorithm is that the sets Ξ_{ij} can be represented as BDDs, since they give an "and-or" information concerning the runs that connect q_i to q_j.

The search for a component that contains $[1 \dots n]$ in the matrix Ξ might prove a lengthy process, even if we restrict ourselves to only the union $\bigcup_{i=1}^{n} Q_i$. But, with a simple trick, we may only need to check a single component: we append to the set of states Q two special states, denoted q_* and q_{**}. q_* is used for looping *before* any run and q_{**} for looping *after* any run. We will call the state q_* as the *source state* and the state q_{**} as the *sink state*.

Formally, we transform $\mathcal{A}$ into the n-automaton $\tilde{\mathcal{A}} = (\tilde{Q}, \tilde{\delta}, Q_1, \dots, Q_n)$ where:

$$\tilde{Q} = (Q \cup \{q_*, q_{**}\})$$
$$\tilde{\delta} = \delta \cup \left\{ q_* \xrightarrow{a} q_*, q_* \xrightarrow{a} q, q_{**} \xrightarrow{a} q_{**}, q \xrightarrow{a} q_{**} \mid q \in Q_i, i \in [1 \dots n], a \in \Sigma \right\}$$

Definition 7.2.9. *The automaton $\tilde{\mathcal{A}}$ is called the **completion** of $\mathcal{A}$.*

As a consequence, once we have constructed the matrix Ξ for $\tilde{\mathcal{A}}$, we only need to check whether the component corresponding to (q_*, q_{**}) contains $[1 \dots n]$.

7.2.2 ε-transitions in n-automata

The class of n-automata was defined without allowing ε-transitions. However in the sequel we will sometimes need them in order to make simpler constructions of n-automata.

An **n-automaton with ε-transitions** is a tuple $\mathcal{A} = (Q, \delta, Q_1, \dots, Q_n)$ in which $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$. The notions of run, accepting run and n-word accepted by an accepting run are the same as for "ordinary" n-automata.

The elimination of ε-transitions proceeds, like for finite automata, by computing the reflexive-transitive closure of the relation $\xrightarrow{\varepsilon}$ on Q. There is however a problem specific to n-automata, the recomputation of accepting sets. Remind that, in the process of removing ε-transitions from finite automata, a state is declared as "accepting" (i.e. final) iff it reaches, after finitely many ε-transitions, a final state. This cannot be the case for n-automata, as we may see from the example in Figure 7.4.

The 4-automaton in Figure 7.4 (b) is obtained by removing all the ε-transitions from the 4-automaton in Figure 7.4 (a) with the usual technique, that is, by putting $q \xrightarrow{x} r$ in the new automaton iff there exist states q', r' such that $q(\xrightarrow{\varepsilon})^* q' \xrightarrow{x} r'(\xrightarrow{\varepsilon})^* r$. We have denoted here $(\xrightarrow{\varepsilon})^*$ the reflexive and transitive closure of the relation $\xrightarrow{\varepsilon}$on Q.

But we need to redefine also the accepting states, and besides the choice $Q_1 = \{q_1\}$ and $Q_4 = \{q_4\}$ there is no way for redefining the accepting sets that renders the resulting 4-automaton at (b) equivalent to the one at (a). The reason for this deadlock is that, if we choose q_2 to be in both Q_2 and Q_3 then the resulting 4-automaton would accept the 4-word represented in Figure 7.5 (a), which is not accepted by the 4-automaton in Figure 7.4 (a). The same situation occurs if we choose

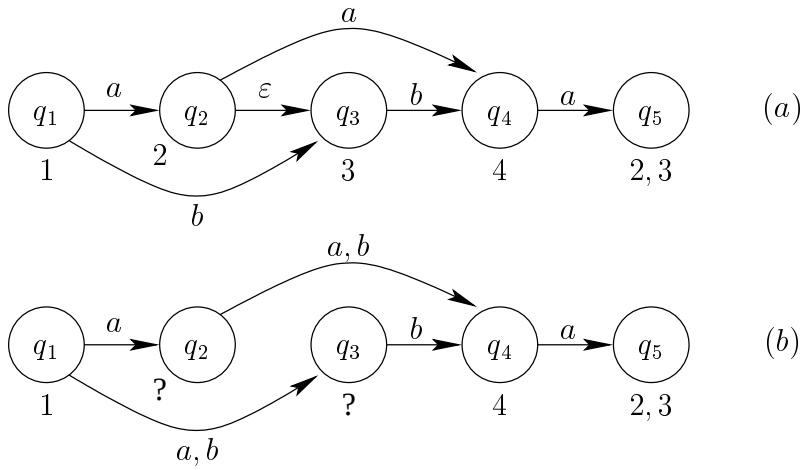


Fig. 7.4. The "brute force" removal of ε-transitions from the 4-automaton at (a) is drawn at (b). In this 4-automaton there is no way to establish the accepting sets to which q_2 and q_3 must belong.

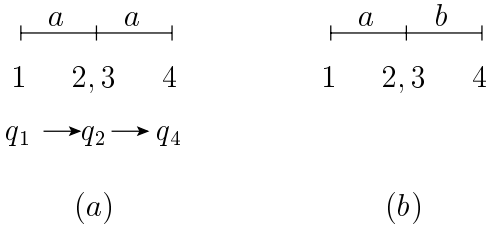


Fig. 7.5. Two 4-words for exemplifying the peculiarities of removing ε-transitions in n-automata.

q_3 to be in both Q_2 and Q_3, whereas if we leave the accepting sets unchanged then the resulting automaton would not accept the 4-word depicted in Figure 7.5 (b).

The solution is to replicate each state that takes part into a sequence of ε-transitions, according to the number of distinct runs with ε-transitions that pass through it. For our 4-automaton in Figure 7.4 the solution is the 4-automaton in Figure 7.6.

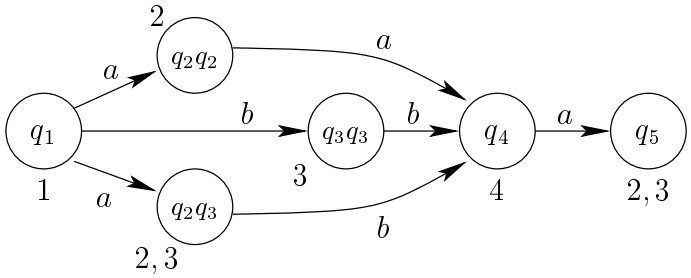


Fig. 7.6. A 4-automaton without ε-transitions equivalent to the 4-automaton in Figure 7.4.

In general, suppose we are given an n-automaton with ε-transitions $\mathcal{A} = (Q, \delta, Q_1, \ldots, Q_n)$. The states of the n-automaton without ε-transitions are pairs (q, q', X) consisting of two states in

Q and an index set $X \subseteq [1 \dots n]$. The idea is to encode in each such state an ε-run that starts in q, ends in q' and passes through all the accepting sets whose indices are in X.

Formally, the following automaton without ε-transitions can be showed equivalent to $\mathcal{A}$:

$$\begin{aligned}
\mathcal{B} &= \big(Q', \theta, Q'_1, \dots, Q'_n\big) \text{ where} \\
Q' &= \big\{(q, q', X) \mid \exists \rho = (q_i)_{i\in[1\dots k]} \text{ such that } q_1 = q, q_k = q' \text{ and } q_i \xrightarrow{\varepsilon} q_{i+1} \text{ for all } i \in [1 \dots k-1] \\
&\qquad\qquad \text{and for all } j \in X, \text{ there exists some } i \in [1 \dots n] \text{ with } q_i \in Q_j \big\} \\
\theta &= \big\{(q, q', X) \xrightarrow{a} (r, r', Y) \mid q' \xrightarrow{a} r\big\} \\
Q'_j &= \big\{(q, q', X) \mid j \in X\big\}, \qquad \text{for all } j \in [1 \dots n]
\end{aligned}$$

7.2.3 Basic operations with n-automata

Proposition 7.2.10. *The class of regular n-languages is closed under union and intersection.*

Proof. Both results are straightforward generalizations of the closure results for regular languages. We will provide only the proof for intersection:

Given two n-automata $\mathcal{A} = (Q, \delta, Q_1, \dots, Q_n)$ and $\mathcal{A}' = (Q'\delta', Q'_1, \dots, Q'_n)$, the n-automaton that accepts $L(\mathcal{A}) \cap L(\mathcal{A}')$ is $\mathcal{A}_\cap = (Q \times Q', \delta_\cap, Q_1 \times Q'_1, \dots, Q_n \times Q'_n)$ where

$$\delta_\cap = \big\{(q, q') \xrightarrow{a} (r, r') \mid q \xrightarrow{a} q' \in \delta \text{ and } r \xrightarrow{a} r' \in \delta'\big\} \qquad \square$$

Remark 7.2.11. Observe that the resulting automaton is a completion since the pair state (q_*, q_*) plays the rôle of q_* and the pair state (q_{**}, q_{**}) works as q_{**}.

Proposition 7.2.12. *Given an n-automaton $\mathcal{A} = (Q, \delta, Q_1, \dots, Q_n)$ and a subset $J \subseteq [1 \dots n]$ with $card(J) = p$, then the p-word language $L(\mathcal{A})\big|_J$ can be recognized by some p-automaton.*

Proof. The first step is to take the completion $\tilde{\mathcal{A}}$ of $\mathcal{A}$. Then we transform this automaton by remembering in each state the set of indices from $\overline{J}$ which "can be visited" through a run that starts in q_*.

Formally, suppose that $J = \{i_1, \dots, i_p\}$ is the presentation of J in increasing order, that is, $i_k = l_J^{-1}(k)$. We then construct the p-automaton $\mathcal{B} = (\tilde{Q} \times \mathcal{P}(\neg J), \overline{\delta}, Q'_1, \dots, Q'_{n-p})$ where $\neg J$ denotes the complement of J and

$$\begin{aligned}
\overline{\delta} &= \big\{(q, X) \xrightarrow{a} (r, Y) \mid q \xrightarrow{a} r \in \tilde{\delta}, X \subseteq Y \subseteq \neg J \text{ and } r \in Q_j \text{ for all } j \in Y \setminus X\big\} \\
Q'_k &= Q_{i_k} \times \mathcal{P}(\neg J), \text{ or, in other words, } Q'_k = Q_{l_J^{-1}(k)} \times \mathcal{P}(\neg J)
\end{aligned}$$

This automaton is not yet the desired one since there is no guarantee that in an accepting run in $\mathcal{B}$, the index component of each state records exactly the set of indices from $\neg J$ which have been visited. But we will take advantage of the existence of the completion states q_* and q_{**} in the following way: we restrict the set of states to those that are *reachable* from $(q_*, \emptyset)$ and *coreachable* from (q_{**}, J). Denote $\overline{Q}$ the resulting set of states and $\overline{\mathcal{B}}$ the restricted p-automaton. We claim that, with this restriction, only p-words in $L(\mathcal{A})\big|_J$ are accepted.

To this end, observe that an accepting run in $\neg\mathcal{B}$ must necessarily be extensible to an accepting run that starts in $q_* \times \emptyset$ and ends in $q_{**} \times \neg J$. But the construction of the transition function assures then that, for all $i \in J$ this run must pass through a state belonging to $Q_i \times \mathcal{P}(\neg J)$. Since it is an accepting run, it also passes through each Q_i for all $i \in J$. Therefore, if we forget the set component of each state we get an accepting run in $\tilde{\mathcal{A}}$. Hence the p-word associated with the run in $\overline{\mathcal{B}}$ is the J-projection of the n-word associated with the run in $\tilde{\mathcal{A}}$. The reverse proof follows similarly. □

Proposition 7.2.13. *Given an m-automaton $\mathcal{A} = (Q, \delta, Q_1, \ldots, Q_m)$, an n-automaton $\mathcal{B} = (Q', \delta', Q'_1, \ldots, Q'_n)$ and a number $p \leq \min(m, n)$, the p-juxtaposition of $L(\mathcal{A})$ with $L(\mathcal{B})$ is accepted by some $(m + n - p)$-automaton.*

Proof. The essential tool used here is the relationship between juxtaposition, extension and projection on n-word languages given by Identity 6.23. Hence we build the $(m+n-p)$-automaton which accepts $L(\mathcal{A})\square_p L(\mathcal{B})$ as an intersection of an extension of $\mathcal{A}$ with an extension of $\mathcal{B}$. The idea is to use the completed automata $\tilde{\mathcal{A}}$, resp. $\tilde{\mathcal{B}}$, transformed into $(m + n - p)$-automata by adding new accepting components. The new components will simply contain all the states in the automata.

Formally, we transform $\mathcal{A}$ into the $(m + n - p)$-automaton $\overline{\mathcal{A}} = (Q_*, \delta_*, Q_1, \ldots, Q_{m+n-p})$ where:

$$\begin{aligned} Q_* &= (Q \cup \{q_*, q_{**}\}) \\ \delta_* &= \delta \cup \left\{q_* \xrightarrow{a} q_*, q_* \xrightarrow{a} q, q_{**} \xrightarrow{a} q_{**}, q \xrightarrow{a} q_{**} \mid q \in Q_i, i \in [1 \ldots m]\right\} \\ Q_k &= Q_* \text{ for each } k \in [m + 1 \ldots m + n - p]. \end{aligned}$$

Similarly we extend $\mathcal{B}$ into $\overline{\mathcal{B}} = (Q'_*, \delta'_*, S_1, \ldots, S_{m+n-p})$ where:

$$\begin{aligned} Q'_* &= (Q' \cup \{q_*, q_{**}\}) \\ \delta'_* &= \delta' \cup \left\{q_* \xrightarrow{a} q_*, q_* \xrightarrow{a} q, q_{**} \xrightarrow{a} q_{**}, q \xrightarrow{a} q_{**} \mid q \in Q_i, i \in [1 \ldots m]\right\} \\ S_k &= \begin{cases} Q'_* & \text{for each } k \in [1 \ldots m - p] \\ Q'_{k-m+p} & \text{for each } k \in [m - p + 1 \ldots m + n - p]. \end{cases} \end{aligned}$$

Observe that

$$\begin{aligned} L(\overline{\mathcal{A}}) &= \left\{w \in \mathcal{WD}_{m+n-p}(\Sigma) \mid w\big|_{[1\ldots m]} \in L(\mathcal{A})\right\} \\ L(\overline{\mathcal{B}}) &= \left\{w \in \mathcal{WD}_{m+n-p}(\Sigma) \mid w\big|_{[m-p+1\ldots m+n-p]} \in L(\mathcal{B})\right\}. \end{aligned}$$

Then identity 6.23 implies that

$$L(\mathcal{A})\square_p L(\mathcal{B}) = L(\overline{\mathcal{A}}) \cap L(\overline{\mathcal{B}}).$$ □

7.2.4 Relationship with n-regwords

Once at this point, we may ask what is the relationship between n-regwords and n-automata. The following theorem gives this relationship:

Theorem 7.2.14. *The class of n-word languages accepted by n-automata equals the class of n-word languages which are the semantics of a sum of n-regwords.*

Proof. The left-to-right inclusion works by the aid of the classical Kleene theorem as follows: first, we decompose each n-automaton into several smaller ones, in which each accepting set is a singleton. Hence, the language of the n-automaton $\mathcal{A} = (Q, \delta, Q_1, \ldots, Q_n)$ equals the union of the languages of all the automata $\mathcal{B} = (Q, \delta, \{q_1\}, \ldots, \{q_n\})$ with $q_i \in Q_i$ for all $i \in [1 \ldots n]$. It there remains to prove the inclusion for such automata.

So consider that for the given automaton $card(Q_i) = 1$, i.e., $Q_i = \{q_i\}$ for all $i \in [1 \ldots n]$. For each $i, j \in [1 \ldots n]$, denote $\mathcal{A}_{ij}$ the finite automaton whose transition function is δ and whose initial, resp. final states are q_i, resp. q_j, that is, $\mathcal{A}_{ij} = (Q, \Sigma, \delta, q_i, \{q_j\})$. This automaton constrains all the n-words whose (i, j)-component is *positive* (i.e. in Σ^*). The constraint for the n-words whose (i, j)-component is *negative* (i.e. in $(\Sigma^{-1})^*$) is provided by the *inverted* language of $\mathcal{A}_{ji}$, $L(\mathcal{A}_{ji})^{-1} \subseteq (\Sigma^{-1})^*$.

Let us denote then by E_{ij} the regular expression over Σ equivalent (by the Kleene theorem) to $\mathcal{A}_{ij}$. Construct the n-regword R whose components are

$$R_{ij} = E_{ij} + E_{ji}^{-1} \tag{7.3}$$

We have denoted here by E^{-1} the expression obtained from E by replacing each letter in Σ with its inverse.

We claim that $L(\mathcal{A}) = \|R\|$. The inclusion $\subseteq$ is assured by construction. For the reverse we will essentially use the triangular identity characterizing n-words:

Consider $w \in \|R\|$ and let $\prec$ be an ordering on $[1 \ldots n]$ which is compatible with w, that is, if $i \prec j$ then $w_{ij} \in \Sigma^*$. We construct a run in $\mathcal{A}$ for w, run which passes through the accepting states in the order indicated by $\prec$. The run is constructed inductively on the order $\prec$ as follows: denote i_k the k-th index in the order $\prec$, for $k \in [1 \ldots n]$.

For $k = 2$ we have $w_{i_1,i_2} \in |E_{i_1 i_2}| = L(\mathcal{A}_{i_1,i_2})$ and hence we get a run in $\mathcal{A}$ that starts in q_{i_1} and ends in q_{i_2}.

Suppose we have built a run, up to k, that passes through $q_{i_1}, \ldots, q_{i_k}$ in this order. To extend this run we consider first a run ρ associated with $w_{i_k,i_{k+1}}$ in $\mathcal{A}_{i_k,i_{k+1}}$. This run exists since, by hypothesis, $w_{i_k i_{k+1}} \in |E_{i_k i_{k+1}}| = L(\mathcal{A}_{i_k i_{k+1}})$. Moreover this run starts in q_{i_k} and ends in $q_{i_{k+1}}$. Then we just append this run to the one we have built so far.

The fact that this concatenation is consistent with the other constraints imposed on the word follows by the triangle identity: for each $l < k + 1$ the fraction of the run that is associated with w_{i_l,i_k} concatenated to the run associated with $w_{i_k,i_{k+1}}$ in $\mathcal{A}_{i_k,i_{k+1}}$ is an accepting run associated with $w_{i_l,i_{k+1}}$ in $\mathcal{A}_{i_l,i_{k+1}}$. Observe also that the compatibility of the ordering $\prec$ assures that the concatenation $w_{i_l,i_k} \cdot w_{i_k,i_{k+1}}$ gives a word and not a mixture of letters and antiletters.

The embedding of n-regwords into n-automata works by an argument similar to the intersection argument. Hence, given an n-regword $R \in \mathcal{WD}_n(\Sigma)$, for each pair $(i,j) \in \big([1\ldots n]\times[1\ldots n]\big)$ we consider the finite automaton which is associated with the regular expression R^+_{ij} by the classical Kleene theorem, be it

$$\mathcal{A}^+_{ij} = (Q^+(i,j), \Sigma, \delta^+(i,j), Q^+_0(i,j), Q^+_f(i,j)).$$

Consider also the finite automaton which is associated with R^-_{ij}, be it

$$\mathcal{A}^-_{ij} = (Q^-(i,j), \Sigma^{-1}, \delta^-(i,j), Q^-_0(i,j), Q^-_f(i,j)).$$

Observe that $L(\mathcal{A}_{ij}) \subseteq (\Sigma^{-1})^*$.

Build then the automaton $\mathcal{A}_{ij}$ as the union of $\mathcal{A}^+_{ij}$ and the *inverse* of $\mathcal{A}^-_{ij}$. The inverse of $\mathcal{A}^-_{ij}$ is $\big(Q^-(i,j), \big(\delta^-(i,j)\big)^{-1}, Q^-_f(i,j), Q^-_0(i,j)\big)$ where

$$\big(\delta^-(i,j)\big)^{-1} = \{q \xrightarrow{a} r \mid r \xrightarrow{a^{-1}} q \in \delta^-(i,j)\}$$

We denote $\mathcal{A}_{ij} = (Q(i,j), \delta(i,j), Q_0(i,j), Q_f(i,j))$.

We transform $\mathcal{A}_{ij}$ into an n-automaton by the same technique used in the proof of Proposition 7.2.13. That is, we do the following steps:

- Append two new states q_* and q_{**} to $Q(i,j)$ and put $q_* \xrightarrow{a} q_*, q_* \xrightarrow{a} q$ for each $q \in Q_0(i,j)$ and $a \in \Sigma$, and $q_{**} \xrightarrow{a} q_{**}, q \xrightarrow{a} q_{**}$ for each $q \in Q_f(i,j)$ and $a \in \Sigma$. We denote $Q_*(i,j) = Q \cup \{q_*, q_{**}\}$.
- For each $k \in [1\ldots n] \setminus \{i,j\}$ we put $Q_k = Q_*(i,j)$ and $Q_i = Q_0(i,j), Q_j = Q_f(i,j)$.

Denote the resulting automaton as $\mathcal{A}^*_{ij}$.

Build then the intersection of all $\mathcal{A}^*_{ij}$ for all $i,j \in [1\ldots n]$ by a straightforward generalization of the intersection construction from Proposition 7.2.10. An easy check shows that the resulting n-automaton accepts indeed $\|R\|$. □

7.3 Non-elasticity

In our search of a decidable class of $2n$-word regular expressions we may recall that, when we have constructed the $2n$-signal semantics to timed automata in 6.5.2, we have produced only signals with reset times in which for each $i \in [1\ldots n]$, either $t'_i = t_i$, or, for each $j \in [1\ldots n]$, $t'_i > t_j$. Or, in $2n$-signal format, for each $i \in [1\ldots n]$, either $w_{i,n+i} = \varepsilon$, or, for each $j \in [1\ldots n]$, $w_{j,n+i} \in \mathsf{Sig}(\Sigma)$.

In the sequel we will focus on the following weaker property (written here for n-words):

(N) For each $i,j \in [1\ldots n]$, one of the following requirements holds

(N1) $w_{i,n+i} = \varepsilon$;

(N2) $w_{j,n+j} = \varepsilon$;

(N3) $w_{i,n+j} \in \Sigma^*$ and $w_{j,n+i} \in \Sigma^*$.

Equivalently, this property says that for each $i, j \in [1 \dots n]$, if $w_{i,n+i} \neq \varepsilon$ and $w_{j,n+j} \neq \varepsilon$ then $w_{i,n+j} \in \Sigma^*$ and $w_{j,n+i} \in \Sigma^*$. We prefer the above formulation since we will make some reference to $2n$-words that have only property *(N3)*.

Observe that, for PCP dominoes, this property forbids the situation in which one of the words ends before the other begins. If we recall the proof of the undecidability of PCP, we may observe that the simulation of a Turing Machine by a PCP instance requires copying the contents of certain cells, and this procedure needs "elastic" dominoes in which the relative distance between the two words composing a domino can be arbitrarily large.

We will show that this property assures the star closure of $2n$-automata. As a consequence, this property assures the decidability of the emptiness problem for $2n$-word regular expressions.

Definition 7.3.1. *A $2n$-word w is called **non-elastic** if the property* (N) *holds. If for each $i, j \in [1 \dots n]$ only the property* (N3) *holds then we say that w is **strictly non-elastic**.*

*For each $2n$-word regular expression E, the **non-elastic semantics** of E, denoted $|E|_n$ consists of the non-elastic $2n$-words in its semantics.*

*For each $2n$-automaton $\mathcal{A}$, the **non-elastic language** of $\mathcal{A}$ denoted $L_n(\mathcal{A})$, consists of the non-elastic $2n$-words in its language.*

The set of indices $i \in [1 \dots n]$ which satisfy property *(N3)* for a $2n$-word w is called the set of *strictly non-elastic* indices for w. Examples of elastic, non-elastic and strictly non-elastic 4-words are given in Figure 7.7.

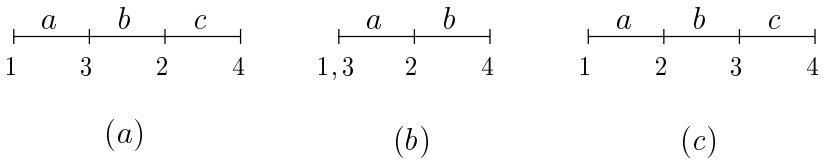


Fig. 7.7. (a) A elastic 4-word. (b) A non-elastic 4-word. (c) A strictly non-elastic 4-word.

For a more general scheme of the positioning of distinguished points in non-elastic or strictly non-elastic $2n$-words, see Figure 7.8. This figure also presents intuitively the "interface" part of and the "contribution" part of the non-elastic $2n$-word.

Note that in a strictly non-elastic $2n$-word the interface part is empty.

Remark 7.3.2. Observe that, for any non-elastic $2n$-word w, $w_{i,n+i} \in \Sigma^*$. Moreover, if w is strictly non-elastic and $w_{i,n+i} = w_{j,n+j} = \varepsilon$ then $w_{i,n+j} = w_{j,n+i} = \varepsilon$.

Let us see now how non-elasticity works in $2n$-automata: consider an $2n$-automaton $\mathcal{A}$, an accepting run $\rho = (q_i, a_i, q_{i+1})_{i \in [1 \dots m]}$ in $\mathcal{A}$, a $2n$-word $w \in \mathcal{WD}_{2n}(\Sigma)$ and a sequence of indices $\boldsymbol{l} = (l_i)_{i \in [1 \dots 2n]}$ witnessing the acceptance of w by ρ. If w is non-elastic then the witnessing sequence bears the following property:

For each $i, j \in [1 \dots n]$,

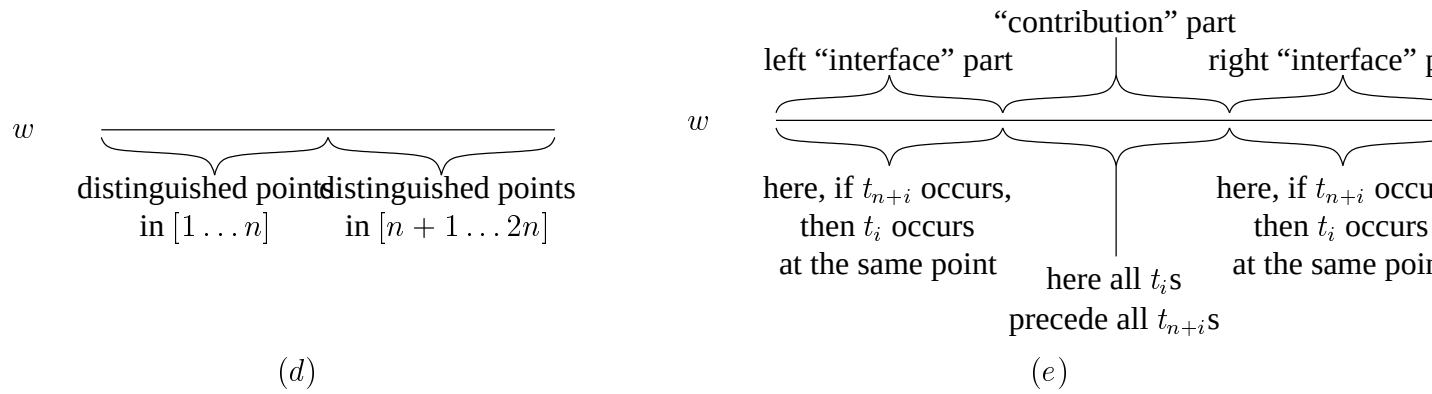


Fig. 7.8. Positioning of distinguished points in (a) a strictly non-elastic $2n$-word and (b) in a non-elastic n-word.

1. $l_i \leq l_{n+i}$;
2. If $l_i \neq l_{n+i}$ and $l_j \neq l_{n+j}$ then $l_i \leq l_{n+j}$ and $l_j \leq l_{n+i}$.

We will call the pair $(\rho, \boldsymbol{l})$ with the above properties a *non-elastic pair*. It is clear that, in any $2n$-automata, non-elastic pairs are associated only to non-elastic $2n$-words accepted by the automaton.

The following property shows the way of constructing the non-elastic semantics for $2n$-regwords:

Proposition 7.3.3. *Start with some $2n$-regword R. Denote $\mathcal{X}$ the set of subsets $X \subseteq [1 \dots n]$ which satisfy the property that $\varepsilon \in R_{i,n+i}$ for all $i \in X$. For each $X \in \mathcal{X}$, denote $R(X)$ the following $2n$-regword:*

$$R(X)_{ij} = \begin{cases} R_{ij} & \text{for } i,j \in [1 \dots n] \text{ or } i,j \in [n+1 \dots 2n] \\ \varepsilon & \text{for } j = n+i, i \in X \text{ or } i = n+j, j \in X \\ R_{ij} & \text{for } j = n+k, k \in X, i \in [1 \dots n] \text{ or } i = n+k, k \in X, j \in [1 \dots n] \\ R^+_{ij} & \text{for } i \in \neg X, j = n+k, k \in \neg X \\ R^-_{ij} & \text{for } j \in \neg X, i = n+k, k \in \neg X \end{cases} \quad (7.4)$$

Then the non-elastic semantics of R equals the semantics of the $2n$-word regular expression $\sum_{X \in \mathcal{X}} R(X)$.

Proof. The property follows by double inclusion. One direction is straightforward, since clearly the semantics of each $R(X)$ is non-elastic and included in the semantics of R.

The reverse follows again easily since each non-elastic $2n$-word w in the semantics of R also belongs to the semantics of $R(X)$, where X is the *complement* of the set of strictly non-elastic indices in w. □

Unfortunately non-elasticity is not preserved by concatenation, as the example in the Figure 7.9 shows.

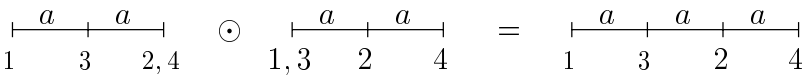


Fig. 7.9. The concatenation of two non-elastic 4-words does not necessarily yield a non-elastic 4-word.

As a consequence, the undecidability theorem 6.6.3 can be proved even for non-elastic 4-words, since we may decompose the 4-regwords corresponding to each domino in a PCP instance into a concatenation of non-elastic 4-words.

We might be tempted to restrict $\odot$ to a "more partial" concatenation, let's call it *non-elastic concatenation* and denote it $\odot_r$, that would produce only non-elastic $2n$-words. But observe that this non-elastic concatenation is nonassociative, as it is exemplified in Figure 7.10.

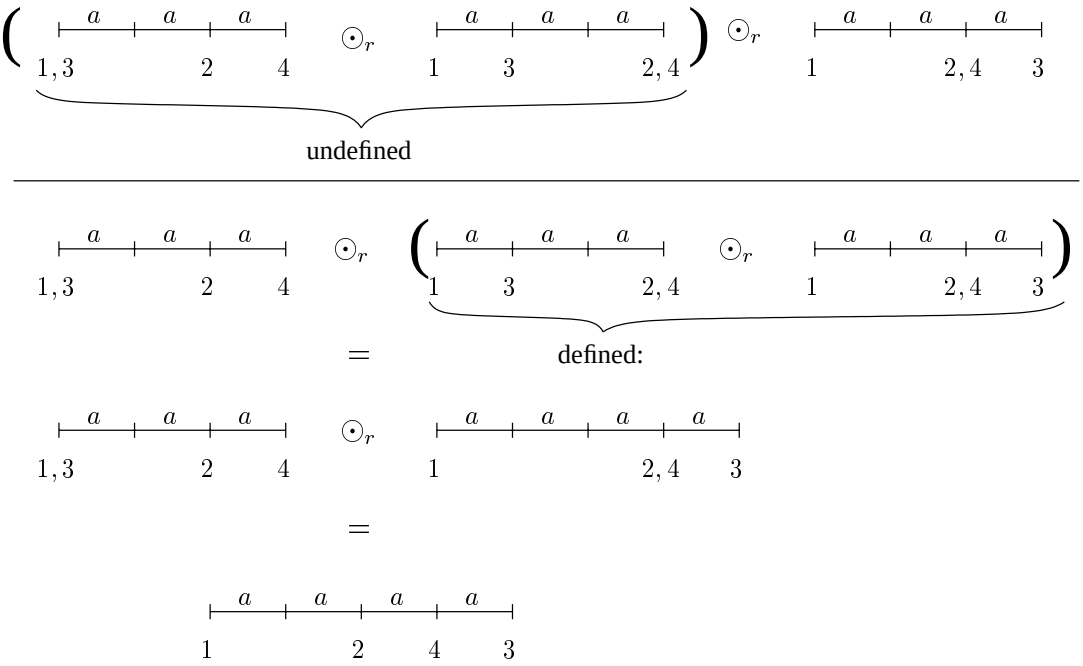


Fig. 7.10. An example of nonassociativity of the concatenation $\odot_r$: the first parenthesis layout leads to undefined, since the concatenation of the two 4-words in the parenthesis leads to a non-elastic 4-word. Of contrary, the second parenthesis layout gives a non-elastic 4-word.

This would make doubtful the possibility to construct the associated *non-elastic star* since the powers L^i of a $2n$-word language $L \subseteq W_{2n}(\Sigma)$ might not be uniquely defined.

Hence, non-elasticity is not a good algebraic property and one might be tempted to search for other properties. But our aim in this chapter is not to find algebraic structures that are "good" w.r.t. decidability. We only want to isolate some property that assures decidability and then, in a subsequent chapter, to show that particular structures, associated to special classes of $2n$-words, bear these properties and hence have decidable emptiness problems.

We will show in the next section that non-elasticity, when carefully handled, leads indeed to decidability. Careful handling means the following two conditions:

1. If we intend to concatenate two non-elastic $2n$-word languages, we need to check first that only non-elastic $2n$-words are produced.
2. If we intend to build the star of a non-elastic $2n$-word language, we need to check first that, by concatenating the given non-elastic $2n$-word language with itself an arbitrary number of times, we get only non-elastic $2n$-words.

Observe that, in a certain sense, the second condition says that the *non-elastic star* is built in a "canonical" manner, since it assures that all the non-elastic concatenations on which it relies are associative.

7.4 The non-elastic star closure theorem

We start this section by noting that, as a combination of the juxtaposition and the projection constructions, the family of $2n$-word languages accepted by $2n$-automata is closed under concatenation. It is clear that if require the given $2n$-automata to accept only non-elastic $2n$-words such that the concatenation of the two languages produces only non-elastic $2n$-words, we still get the same construction. We provide here a direct concatenation construction as we will get this way some intuition for the main theorem of this section, the star closure theorem.

Take two $2n$-automata $\mathcal{A} = (Q, \delta, Q_1, \ldots, Q_{2n})$ and $\mathcal{B} = (Q', \delta', Q'_1, \ldots, Q'_{2n})$, both of which accept only non-elastic $2n$-words. Suppose also that $L(\mathcal{A}) \odot L(\mathcal{B})$ is composed of non-elastic $2n$-words. Our aim is to build a $2n$-automaton for this set, and we proceed as follows:

If both automata accepted only *strictly non-elastic* $2n$-words then the idea would be the following: we start $\mathcal{A}$ on the "NW quarter" of the given $2n$-word (that is, on $w\big|_{[1\ldots n]}$) and check whether on this section we pass through all the accepting states $Q_1, \ldots, Q_n$. Then continue until we reach an accepting state between $Q_{n+1}, \ldots, Q_{2n}$. (the assumption that w is non-elastic implies that the first such moment succeeds all the moments of passing through $Q_1, \ldots, Q_n$). At this moment we start $\mathcal{B}$ and synchronously fire transitions from both automata.

From now on, $\mathcal{A}$ must pass through some accepting set Q_{n+i} ($i \in [1 \ldots n]$) *synchronously* with a passage of $\mathcal{B}$ through the accepting set Q'_i. We record all indices i that have observed such a "synchronous passage" into some set X. Once this set equals $[1 \ldots n]$, we are sure we have identified in our given $2n$-word a section which is accepted by $\mathcal{A}$, and we may now proceed with finalizing the run in $\mathcal{B}$.

Formally, the $2n$-automaton for $L(\mathcal{A}) \odot L(\mathcal{B})$ would be

$$\mathcal{C} = \Big(Q \cup Q' \cup \big(Q \times Q' \times \mathcal{P}([1 \dots n])\big), \delta'', Q_1, \dots, Q_n, Q'_{n+1}, \dots Q'_{2n}\Big) \text{ with}$$

$$\delta'' = \delta \cup \delta' \cup \big\{(q, q', [1 \dots n]) \xrightarrow{\varepsilon} q' \mid q \in Q, q' \in Q'\big\} \cup$$
$$\big\{q \xrightarrow{\varepsilon} (q, q', I) \mid q \in Q_{n+i}, q' \in Q'_i \text{ for all } i \in I \subseteq [1 \dots n]\big\} \cup$$
$$\big\{(q, q', I) \xrightarrow{a} (r, r', J) \mid q \xrightarrow{a} q' \in \delta, r \xrightarrow{a} r' \in \delta', I \subseteq J \subseteq [1 \dots n]$$
$$\text{and for all } i \in J \setminus I, r \in Q_{n+i} \text{ and } r' \in Q'_i\big\}$$

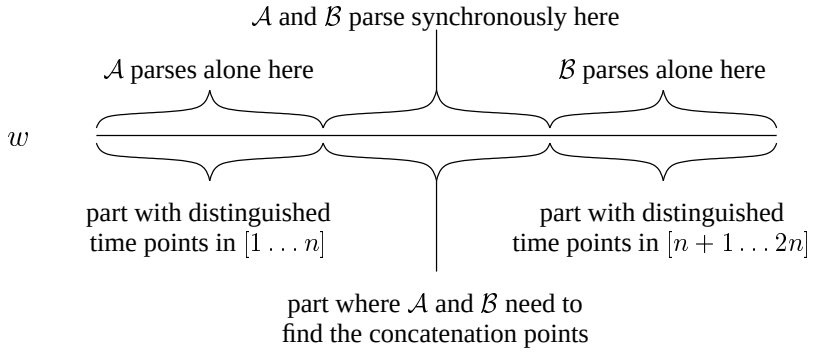


Fig. 7.11. Graphical exemplification of the concatenation construction.

Observe that an accepting run assures, by construction, that all the sets $Q_{n+i} \times Q'_i$ are visited "in between" the last moment when an accepting set Q_j with $j \in [1 \dots n]$ is visited and the first moment when an accepting set Q'_{n+k} with $k \in [1 \dots n]$ is visited. This property is consistent with the hypothesis that all $2n$-words in $L(\mathcal{A})$ and $L(\mathcal{B})$ are strictly non-elastic.

The conditions *(N1)* and *(N2)* pose specific problems since we might need to "start" the automaton $\mathcal{B}$ "before" the automaton $\mathcal{A}$ has visited all the accepting sets Q_j with $j \in [1 \dots n]$. But the idea is the same, namely to "synchronize" the two automata as follows:

> Each time $\mathcal{A}$ passes through the accepting set Q_{n+i}, $\mathcal{B}$ must pass through the accepting set Q'_i and viceversa.

Accepting states would then be simply tuples (q, q', X) in which q is a state in $\mathcal{A}$, q' is a state in $\mathcal{B}$ and X is the set recording synchronous passages.

At this point, one problem might arise, a problem which we have observed also on the projection construction: to actually be sure that the X component has recorded all the synchronous passages, we need to start with an empty X and finish with an $X = [1 \dots n]$. We do this by working with *completed* automata, and then reducing the state space of the resulting $2n$-automaton to the states reachable from $(q_*, q'_*, \emptyset)$ (which acts as a source state) and coreachable from $(q_{**}, q'_{**}, [1 \dots n])$ (which acts as a sink state).

Formally, we build first $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{B}}$, the completions of $\mathcal{A}$ and $\mathcal{B}$. We denote q_* and q'_* the source states of each automaton, respectively q_{**} and q'_{**} their sink states. We then define the following $2n$-automaton:

$$\begin{aligned}
\mathcal{C} &= \big(Q \times Q' \times \mathcal{P}([1 \dots n]), \theta, S_1, \dots, S_{2n}\big) \text{ with} \\
\theta &= \big\{(q, q', X) \xrightarrow{a} (r, r', Y) \mid q \xrightarrow{a} r \in \tilde{\delta}, q' \xrightarrow{a} r' \in \tilde{\delta}', X \subseteq Y \subseteq [1 \dots n], \\
&\qquad \text{and for all } i \in Y \setminus X, r \in Q_{n+i} \text{ and } r' \in Q'_i\big\} \\
S_i &= Q_i \times Q' \times \mathcal{P}([1 \dots n]) \text{ for all } i \in [1 \dots n] \\
S_{n+i} &= Q \times Q'_{n+i} \times \mathcal{P}([1 \dots n]) \text{ for all } i \in [1 \dots n]
\end{aligned}$$

Finally drop all the states of $\mathcal{C}$ that are not reachable from $(q_*, q'_*, \emptyset)$ or not coreachable from $(q_{**}, q'_{**}, [1 \dots n])$, that is, consider the automaton

$$\begin{aligned}
\mathcal{D} &= \big(\overline{Q}, \theta \cap (\overline{Q} \times \Sigma \times \overline{Q}), S_1 \cap \overline{Q}, \dots, S_{2n} \cap \overline{Q}\big) \text{ where} \\
\overline{Q} &= \big\{\overline{q} \in Q \times Q' \times \mathcal{P}([1 \dots n]) \mid \exists w, w' \in \Sigma^* \text{ such that} \\
&\qquad (q_*, q'_*, \emptyset) \xrightarrow{w} \overline{q} \xrightarrow{w'} (q_{**}, q'_{**}, [1 \dots n])\big\}
\end{aligned}$$

Proposition 7.4.1. $L(\mathcal{D}) = L(\mathcal{A}) \odot L(\mathcal{B})$.

Proof. For the right-to-left inclusion, take $w \in L(\mathcal{A})$ and $w' \in L(\mathcal{B})$ such that $w \odot w'$ is defined. Both $2n$-words come with an accepting run, be they $\rho = (q_i, a_i, q_{i+1})_{i \in [1 \dots k-1]}$ for w, respectively $\rho' = (q'_j, a'_j, q'_{j+1})_{j \in [1 \dots k'-1]}$ for w', and with two witnessing sets of indices, $\boldsymbol{i} = (i_l)_{l \in [1 \dots 2n]}$, respectively $\boldsymbol{j} = (j_l)_{l \in [1 \dots 2n]}$, such that

$$\begin{aligned}
&q_{i_l} \in Q_l \text{ and } w_{lm} = \mathsf{word}(\rho, i_l, i_m) \text{ for all } l, m \in [1 \dots 2n] \\
&q'_{j_l} \in Q'_l \text{ and } w'_{lm} = \mathsf{word}(\rho', j_l, j_m) \text{ for all } l, m \in [1 \dots 2n]
\end{aligned}$$

Observe first that the assumption that $w \odot w'$ is defined implies that for each $l, m \in [1 \dots n]$, $w_{l+n,m+n} = w'_{lm}$. Hence the piece of run from ρ that lies in between $q_{i_{l+n}}$ and $q_{i_{m+n}}$ has the same length as the piece of run from ρ' that lies in between q'_{j_l} and q'_{j_m}. More formally, $\mathsf{word}(\rho, i_{n+l}, i_{n+m}) = \mathsf{word}(\rho', j_l, j_m)$. This has the following important consequence:

$$\text{for each } l, m \in [1 \dots n], \quad i_{m+n} - i_{l+n} = j_m - j_l \tag{7.5}$$

Our first aim is then to extend the two runs such that they have equal length and the moment when the first run passes through Q_{n+l} be the same as the moment when the second run passes through Q'_l. This extension will be accomplished by adding loops in q_*, q'_*, q_{**} and/or q'_{**}.

We may assume, according to Remark 7.2.2, that both runs start and end in some accepting set and both witnessing sequences contain the first and the last index in the respective run, that is,

- $1 = i_{l_0}, k = i_{m_0}$ for some $l_0, m_0 \in [1 \dots 2n]$,
- $1 = j_{l'_0}, k' = j_{m'_0}$ for some $l'_0, m'_0 \in [1 \dots 2n]$.

Moreover, by the non-elastic assumption we may consider that $l_0, l'_0 \in [1 \dots n]$ and $m_0, m'_0 \in [n+1 \dots 2n]$.

On the other hand, by Identity 7.5 we have that $i_{m_0} - i_{l'_0+n} = j_{m_0-n} - j_{l'_0}$. It follows that $i_{l'_0+n}$ is the earliest moment at which ρ passes through some accepting set Q_{n+h} for some $h \in [1 \dots n]$,

and j_{m_0-n} is the latest moment at which ρ' passes through some accepting set $Q'_{h'}$ for some $h' \in [1 \dots n]$. That is, if we denote

$$\underline{\imath} = \min\{i_l \mid l \in [n+1 \dots 2n]\} \qquad \underline{\jmath} = \min\{j_l \mid l \in [1 \dots n]\}$$
$$\overline{\imath} = \max\{i_l \mid l \in [n+1 \dots 2n]\} \qquad \overline{\jmath} = \max\{j_l \mid l \in [1 \dots n]\}$$

then

$$\overline{\imath} = k = i_{m_0} \qquad \underline{\jmath} = 1 = j_{l'_0}$$
$$\underline{\imath} = i_{l'_0+n} \qquad \overline{\jmath} = j_{m_0-n}$$

Hence, due to Identity 7.5,

$$\overline{\imath} - \underline{\imath} = \overline{\jmath} - \underline{\jmath}. \tag{7.6}$$

On the other hand, from the equality $\mathsf{word}(\rho, i_{n+l}, i_{n+m}) = \mathsf{word}(\rho', j_l, j_m)$ we msut also have that, for each $h \in [\underline{\imath}, \overline{\imath}]$,

$$a_h = a'_{h-\underline{\imath}+1} \tag{7.7}$$

We then extend ρ by adding $k' - \overline{\jmath} + 1$ replicas of q_{**} at the end and one replica of q_* at the beginning. Similarly, we extend ρ' by adding $\underline{\imath}$ replicas of q'_* at the beginning and one replica of q'_{**} at the end. Observe that the two runs would have the same length since

$$k + k' - \overline{\jmath} + 2 = \overline{\imath} + k' - \overline{\jmath} + \underline{\jmath} + 1 = \overline{\imath} + k' - \overline{\imath} + \underline{\imath} + 1 = k' + \underline{\imath} + 1$$

More formally, denoting $\overline{k} = k' + 1 + \underline{\imath}$, we consider the runs $\overline{\rho} = (r_i, b_i, r_{i+1})_{i \in [1 \dots \overline{k}-1]}$ and $\overline{\rho}' = (r'_i, b_i, r'_{i+1})_{i \in [1 \dots \overline{k}-1]}$ with

$$r_i = \begin{cases} q_* & \text{for } i = 1 \\ q_{i-1} & \text{for } i \in [2 \dots k+1] \\ q_{**} & \text{for } i \in [k+2 \dots \overline{k}] \end{cases} \qquad r'_i = \begin{cases} q'_* & \text{for } i \in [1 \dots \underline{\imath}] \\ q'_{i-\underline{\imath}} & \text{for } i \in [\underline{\imath}+1 \dots \overline{k}-1] \\ q'_{**} & \text{for } i = \overline{k} \end{cases}$$

$$b_i = \begin{cases} a_{i-1} & \text{iff } i \in [2 \dots k+1] \\ a'_{i-\underline{\imath}} & \text{iff } i \in [\underline{\imath}+1 \dots \underline{\imath}+k'] \end{cases}$$

The property 7.7 assures that the b_i's can be uniquely chosen for all $i \in [\underline{\imath}+1 \dots k+1]$.

Observe that these two extended runs bear the desired property: the moment when the first run passes through Q_{n+l} is the same as the moment when the second run passes through Q'_l, or, more formally,

$$\text{for any } l \in [1 \dots n], r_i = q_{i_{l+n}} \text{ iff } r'_i = q'_{j_l}. \tag{7.8}$$

We finally construct the $\mathcal{C}$ run $\chi = \big((r_i, r'_i, I_i), b_i, (r_{i+1}, r'_{i+1}, I_{i+1})\big)_{i\in[1\ldots\overline{k}-1]}$ in which $I_1 = \emptyset$ and

$$I_{i+1} = I_i \cup \big\{l \in [1\ldots n] \mid r_i = q_{i_{l+n}}\big\}$$

Property 7.8 assures that χ is indeed a run in $\mathcal{C}$, because it implies the following fact: if $I_{i+1} \neq I_i$ then for all $l \in I_{i+1} \setminus I_i$, $r_i \in Q_{n+l}$ and $r'_i \in Q'_l$. Observe also that, for each $i \in [1\ldots\overline{k}]$, I_i records all the indices $l \in [1\ldots n]$ with the property that the run has passed through $Q_{l+n} \times Q'_l$. Moreover, since the run contains only states that are accessible from $(q_*, q'_*, \emptyset)$ and coaccessible from $(q_{**}, q'_{**}, [1\ldots n])$, it is actually a run in $\mathcal{D}$.

We consider then the family of indices $\boldsymbol{p} = (p_l)_{l\in[1\ldots 2n]}$, defined as follows:

$$p_l = \begin{cases} i_l + 1 & \text{for } l \in [1\ldots n] \\ j_l + \underline{\imath} & \text{for } l \in [n+1\ldots 2n] \end{cases}$$

It remains to prove that the pair $(\chi, \boldsymbol{p})$ is associated with $w_1 \odot w_2$, that is, that $\boldsymbol{p}$ witnesses the acceptance of $w \odot w'$ by χ:

1. For $l, m \in [1\ldots n]$, $(w \odot w')_{lm} = w_{lm}$, $r_{p_l} = r_{i_l+1} = q_{i_l}$ and $r_{p_m} = r_{i_m+1} = q_{i_m}$. Since we have that $q_{i_l} \xrightarrow{w_{lm}} q_{i_m}$ we get then

$$(r_{p_l}, r'_{p_l}, I_{p_l}) \xrightarrow{(w\odot w')_{lm}} (r_{p_m}, r'_{p_m}, I_{p_m})$$

2. For $l, m \in [n+1\ldots 2n]$, $(w \odot w')_{lm} = w'_{lm}$, $r_{p_l} = r'_{j_l+\underline{\imath}} = q'_{j_l}$ and $r_{p_m} = r'_{j_m+\underline{\imath}} = q'_{j_m}$. Since we have that $q'_{i_l} \xrightarrow{w'_{lm}} q'_{i_m}$ we get then

$$(r_{p_l}, r'_{p_l}, I_{p_l}) \xrightarrow{(w\odot w')_{lm}} (r_{p_m}, r'_{p_m}, I_{p_m})$$

3. For $l \in [1\ldots n]$ and $m \in [n+1\ldots 2n]$ we have to decompose $(w \odot w')_{lm}$ into a concatenation of two words or two antiwords. We then have the following subcases:
   - For $w'_{m-n,m} = \varepsilon$ we have $(w \odot w')_{lm} = w_{lm}$ and hence fall in the first case above.
   - For $w_{l,l+n} = \varepsilon$ we have $(w \odot w')_{lm} = w'_{lm}$ and hence fall in the second case above.
   - If $w_{l,l+n} \neq \varepsilon$ and $w'_{m-n,m} \neq \varepsilon$ then, by the non-elasticity assumption,

$$(w \odot w')_{lm} = w_{l,l+n} \cdot w'_{lm} \in \Sigma^*.$$

   On the other hand, by Identity 7.5 we have $i_{l+n} - i_{l'_0+n} = j_l - j_{l'_0}$, that is, $i_{l+n} + 1 = j_l + \underline{\imath}$. Therefore, similarly to the above cases,

$$(r_{i_l+1}, r'_{i_l+1}, I_{i_l+1}) \xrightarrow{w_{l,l+n}} (r_{i_{l+n}+1}, r'_{i_{l+n}+1}, I_{i_{l+n}+1}) = (r_{j_l+\underline{\imath}}, r'_{j_l+\underline{\imath}}, I_{j_l+\underline{\imath}}) \xrightarrow{w'_{lm}} (r_{j_m+\underline{\imath}}, r'_{j_m+\underline{\imath}}, I_{j_m+\underline{\imath}})$$

   which assures that $(r_{p_l}, r'_{p_l}, I_{p_l}) \xrightarrow{(w\odot w')_{lm}} (r_{p_m}, r'_{p_m}, I_{p_m})$ in this case too.

For the reverse inclusion, take some accepting run $\rho = \big((r_i, r'_i, I_i), b_i, (r_{i+1}, r'_{i+1}, I_{i+1})\big)_{i\in[1\ldots k-1]}$ in $\mathcal{D}$ with $(r_1, r'_1, I_1) = (q_*, q'_*, \emptyset)$ and $(r_k, r'_k, I_k) = (q_{**}, q'_{**}, [1\ldots n])$ and fix some family of indices $\boldsymbol{l} = (l_i)_{i\in[1\ldots 2n]}$ such that $(r_{l_i}, r'_{l_i}, I_{l_i}) \in S_i$ for all $i \in [1\ldots 2n]$. Suppose also that the associated word is w, that is,

$$\mathsf{word}(\rho, l_i, l_j) = w_{ij} \text{ for all } i, j \in [1\ldots 2n]$$

Let us first observe that $I_k = [1\ldots n]$ implies that for each $j \in [1\ldots n]$ there exists some index $p_j \in [1\ldots k]$ such that $j \in I_{p_j} \setminus I_{p_j-1}$. By construction, this implies that $r_{p_j} \in Q_{j+n}$ and $r'_{p_j} \in Q'_j$.

But then the sequence $\rho_1 = (r_i, b_i, r_{i+1})_{i\in[1\ldots k-1]}$ is an accepting run in $\tilde{\mathcal{A}}$: consider the $2n$-word z accepted by this run and the sequence of indices $((l_i)_{i\in[1\ldots n]}, (p_j)_{j\in[1\ldots n]})$, that is, bearing the following properties:

$$\begin{aligned} \mathsf{word}(\rho_1, r_{l_i}, r_{l_j}) &= z_{ij} & \mathsf{word}(\rho_1, r_{p_i}, r_{p_j}) &= z_{n+i,n+j} \\ \mathsf{word}(\rho_1, r_{l_i}, r_{p_j}) &= z_{i,n+j} & \mathsf{word}(\rho_1, r_{p_i}, r_{l_j}) &= z_{n+i,j} \end{aligned}$$

As a consequence, $z\big|_{[1\ldots n]} = w\big|_{[1\ldots n]}$.

Similarly, the sequence $\rho_2 = (r'_i, b_i, r'_{i+1})_{i\in[1\ldots k-1]}$ is an accepting run in $\tilde{\mathcal{A}}$ and we may construct the $2n$-word z' accepted by this run and the sequence of indices $((p_j)_{j\in[1\ldots n]}, (l_i)_{i\in[n+1\ldots 2n]})$, that is

$$\begin{aligned} \mathsf{word}(\rho_2, r'_{l_i}, r'_{l_j}) &= z'_{ij} & \mathsf{word}(\rho_2, r'_{p_i}, r'_{p_j}) &= z'_{n+i,n+j} \\ \mathsf{word}(\rho_1, r'_{l_i}, r'_{p_j}) &= z'_{i,n+j} & \mathsf{word}(\rho_1, r'_{p_i}, r'_{l_j}) &= z'_{n+i,j} \end{aligned}$$

with the corollary that $z'\big|_{[n+1\ldots 2n]} = w'\big|_{[n+1\ldots 2n]}$.

But the above relations also imply that, for all $i, j \in [1\ldots n]$, $z_{n+i,n+j} = z'_{ij}$, hence $z\big|_{[n+1\ldots 2n]} = z'\big|_{[1\ldots n]}$. If we corroborate this with the observations that $z\big|_{[1\ldots n]} = w\big|_{[1\ldots n]}$ and $z'\big|_{[n+1\ldots 2n]} = w'\big|_{[n+1\ldots 2n]}$, we get that

$$w = z \odot z' \in L(\mathcal{A}) \odot L(\mathcal{B}) \qquad \square$$

Remark 7.4.2.

1. Reachability plays an essential rôle in the proof of the reverse inclusion. Without the hypothesis that the run of $\mathcal{D}$ starts in $(q_*, q'_*, \emptyset)$ and ends in $(q_{**}, q'_{**}, [1\ldots n])$ we would not be able to split this run into accepting runs of $\mathcal{A}$ and $\mathcal{B}$. Specifically, the existence of the family of indices $(p_i)_{i\in[1\ldots n]}$ (which helped constructing the two $2n$-words z and z') could not be assured.
2. The set of indices p_i cannot be placed anywhere in the run ρ, since, by hypothesis, $\mathcal{A}$ and $\mathcal{B}$ are non-elastic. Concretely, if an index p_i precedes an index l_j with $j \in [1\ldots n]$ then $z_{i,n+j}$ is an antiword, and this may happen iff $z_{i,i+n} = \varepsilon$. Hence we must have $l_j = p_i$.

The main result of this chapter is the following:

Theorem 7.4.3. *Given a $2n$-automaton $\mathcal{A}$, suppose that, for any $k \in \mathbb{N}$, $L(\mathcal{A})^{k\odot}$ is a non-elastic $2n$-word language, or, equivalently, that $L(\mathcal{A})^{\circledast}$ is a non-elastic $2n$-word language. Then $L(\mathcal{A})^{\circledast}$ is accepted by a $2n$-automaton.*

Proof. We will construct actually the automaton for $L^{\circledast\geq 2} := \bigcup_{k\geq 2} L(\mathcal{A})^{k\odot}$ since the automaton for $L(\mathcal{A})^{\circledast}$ follows by applying the union construction. The technique draws much from the concatenation construction for non-elastic $2n$-automata. We will first explain the construction for a $2n$-automaton that accepts only strictly non-elastic $2n$-words, and then generalize this construction to arbitrary non-elastic $2n$-automata.

The idea for the simpler case of strictly non-elastic $2n$-automata is to use two component states as in the proof for concatenation, but each time one component has completed an accepting run in $\mathcal{A}$, a new component starts checking for an accepting run in $\mathcal{A}$. That is, at each moment during the parse of the given $2n$-word we have one or two copies of $\mathcal{A}$ working synchronously, and on the sections where two copies are working, a passage of the first copy through Q_{n+i} must be simultaneous with a passage of the second copy through Q_i. Also, each such index i that witnesses a synchronous passage must be recorded in a set I. (Of course, there might be passages through $Q_{n+i} \times Q_i$ that are ignored.) We may consider the points where these "synchronous passages" happen as the "concatenation points" between two factors of the given $2n$-word.

When the index set equals $[1 \dots n]$, the first copy has identified a section of the $2n$-word which is accepted by $\mathcal{A}$, hence it stops. The second copy must continue its search since it has only passed through the accepting sets $Q_1, \dots, Q_n$. Next time it reaches an accepting set from $Q_{n+1}, \dots, Q_{2n}$, a new copy of $\mathcal{A}$ is restarted and proceeds synchronously with the old copy as described above. The whole process is stopped with a choice not to start a new copy of $\mathcal{A}$ after a passage of the active copy through one of $Q_{n+1}, \dots, Q_{2n}$, hence leaving to this active copy only the task of finishing an accepting run in $\mathcal{A}$.

The formalization of this construction is the following: start with a strictly non-elastic $2n$-automaton $\mathcal{A} = (Q, \delta, Q_1, \dots, Q_{2n})$. The automaton (with ε-transitions) accepting $L^{\circledast\geq 2}$ would be then:

$$
\begin{aligned}
\mathcal{B} &= \Big(Q \cup \big(Q \times Q \times \mathcal{P}([1 \dots n])\big), \theta, Q_1, \dots, Q_{2n}\Big) \text{ where} \\
\theta &= \delta \cup \big\{(q, q', [1 \dots n]) \xrightarrow{\varepsilon} q' \mid q, q' \in Q\big\} \cup \\
&\qquad \big\{q \xrightarrow{\varepsilon} (q, q', X) \mid \text{ for all } i \in X, q \in Q_{n+i} \text{ and } q' \in Q_i\big\} \cup \\
&\qquad \big\{(q, r, X) \xrightarrow{a} (q', r', Y) \mid q, q', r, r' \in Q, X \subseteq Y \subseteq [1 \dots n] \text{ and} \\
&\qquad\qquad \text{for all } i \in Y \setminus X, q' \in Q_{n+i} \text{ and } r' \in Q_i\big\}
\end{aligned}
$$

A graphical presentation of the way $\mathcal{B}$ parses a strictly non-elastic$2n$-word is given in Figure 7.12.

Observe that strict non-elasticity assures the fact that at each moment, only two copies of $\mathcal{A}$ are necessary, because it is not possible that more than two $2n$-words overlap on the same part.

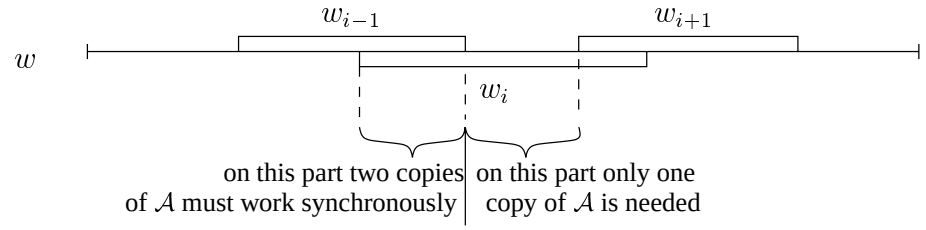


Fig. 7.12. A graphical presentation of the way $\mathcal{B}$ parses a strictly non-elastic $2n$-word. We also suggest here that no three consecutive factors in a decomposition of w may overlap.

This is a very important property that is not valid for elastic concatenations – for example, when concatenating dominoes of a PCP instance.

In order to cope with properties *(N1)* and *(N2)* we need to take into account the fact that more than two copies of $\mathcal{A}$ might have to be initiated when the active copy passes through some accepting set Q_{n+i}. Intuitively, when concatenating (non-strict) non-elastic $2n$-words, interface parts of the factors overlap and hence at each point we might have more than two copies of $\mathcal{A}$ that need to work synchronously.

Suppose we have to parse a non-elastic $2n$-word w whose decomposition is

$$w = w_0 \odot \ldots \odot w_l \odot w_{l+1} \odot \ldots \odot w_m \odot \ldots \odot w_p, \text{ with } w_j \in L(\mathcal{A}) \forall j \in [0 \ldots p] \qquad (7.9)$$

Two important observations are to be made here: the first is that on each part of w no more than two factors may overlap on their contribution parts – and, as a consequence, these two factors must be successive factors, i.e., w_k and w_{k+1} for some $k \in [0 \ldots p-1]$.

The second observation is related to the interface parts of the factors: on each part of w (viewed as a word with distinguished points), several factors overlap on their left interface part – be they $w_k, \ldots, w_l$, with $0 \leq k < l \leq p$. We may consider that the part starts at the leftmost symbol from w – that is, that it is a prefix part of w. The number of factors which overlap on their left interface part is no longer uniformly bounded, as it was the case with the contribution part. But they bear an important property: if in the considered part of w, w_l contains some distinguished point t_i for some $i \in [1 \ldots n]$, then w_{l-1} must have its distinguished point t_{i+n} on the same position – and, as w_{l-1} is on its left interface part, its distinguished point t_i must also be on the same position. Inductively, we obtain that all the factors $w_k, \ldots, w_{l-1}$ must have their distinguished points t_i and t_{i+n} on the same position.

This property is very important, since it implies that any run in $\mathcal{A}$ which is associated to w_k during this part of w, that is, which passes through all the accepting sets Q_i at the distinguished point t_i for w_k, *is also a run associated to* w_l! Of course, this does not mean that the same run will be extensible to an accepting run for the whole w_l – maybe the run for w_k will eventually lead to a deadlock when trying to associate it to w_l. But this property says that *we do not need to memorize the whole sequence of states in the run*, we only need to *memorize the set of states in which the automaton $\mathcal{A}$ might be on a run which is associated to this prefix part of* w. And this imposes a

uniform bound on the memory that is needed for a device that accepts $L(\mathcal{A})^{\circledast \geq 2}$: this bound is proportional to the cardinality of Q. Hence, at least intuitively, a finite automaton would suffice.

We will now introduce several notations and ideas for our construction. Denote first

$$\mathcal{X} = \{X \subseteq [1 \dots 2n] \mid \forall i \in [1 \dots n], i + n \in X \Rightarrow i \in X\} \tag{7.10}$$

$$\mathcal{Q} = \{(X, q, X') \mid X, X' \in \mathcal{X}, X \subseteq X' \text{ and for all } i \in X' \setminus X \text{ we have } q \in Q_i\} \tag{7.11}$$

The states of our "starred" automaton will be quadruples (S, α, β, T) with $S, T \subseteq \mathcal{Q}$ and $\alpha, \beta \in \mathcal{Q} \cup \{\emptyset\}$. α is called the *left active* component, β is the *right active* component while S is the *history component* and T is the *prophecy* component.

The pair (α, β) plays the same role as the pair of states in the construction for the strictly non-elastic case[1] for strictly non-elastic $2n$-words as α must pass through an accepting set Q_{n+i} at the same time β passes through the accepting set Q_i for the same $i \in [1 \dots n]$. We say here that $\alpha = (X, q, X')$ *is passing through* some Q_i iff

The prophecy component's utility is then the following: it provides the bounded memory which is needed for parsing the left interface parts of the factors. Symmetrically, the history component is utilized for parsing the right interface parts of the factors.

7.11

Formally, the states are of the following three forms:

1. $(\emptyset, \emptyset, (X, q, X'), T)$ where $(X, q, X') \in \mathcal{Q}$ and $T \subseteq \mathcal{Q}$, with the property that, for all $(Y, r, Y') \in T$,
   a) $X \cap [n + 1 \dots 2n] \supseteq (Y \cap [1 \dots n]) + n$.
   b) $X' \cap [n + 1 \dots 2n] \supseteq (Y' \cap [1 \dots n]) + n$.
   c) $(Y' \setminus Y) \cap [n + 1 \dots 2n] \subseteq ((Y' \setminus Y) \cap [1 \dots n]) + n \subseteq (X' \setminus X) \cap [n + 1 \dots 2n]$.

   These states are used during the search for the first set of concatenation points, the ones that "separate" w_0 from w_1. The requirements intuitively say that the tuples in the prophecy component cannot consider passing through some accepting set Q_i unless the right active component is passing through the accepting set Q_{n+i} for the same i.
2. $(S, (X, q, X'), (Y, r, Y'), T)$ with $(X, q, X'), (Y, r, Y') \in \mathcal{Q}$ and $S, T \subseteq \mathcal{Q}$, with the following properties:
   a) $X \cap [n + 1 \dots 2n] = (Y \cap [1 \dots n]) + n$.
   b) $X' \cap [n + 1 \dots 2n] = (Y' \cap [1 \dots n]) + n$.
   c) For each $(U, s, U') \in S$,
      i. $U \cap [n + 1 \dots 2n] \supseteq (X \cap [1 \dots n]) + n$.
      ii. $U' \cap [n + 1 \dots 2n] \supseteq (X' \cap [1 \dots n]) + n$.
      iii. $(U' \setminus U) \cap [1 \dots n] \subseteq ((U' \setminus U) \cap [n + 1 \dots 2n]) - n \subseteq (X' \setminus X) \cap [1 \dots n]$.
   d) For each $(V, t, V') \in T$,

[1] Note that, in fact, we have a difference with the construction in the strictly non-elastic case since both the left and the right active components record *all* the indices of the accepting sets throughout which they pass, that is, their memory needs to reach the set $[1 \dots 2n]$ before considering they have accomplished their duty of tracking an accepting run in $\mathcal{A}$.

i. $Y \cap [n+1 \dots 2n] \supseteq (V \cap [1 \dots n]) + n$.
ii. $Y' \cap [n+1 \dots 2n] \supseteq (V' \cap [1 \dots n]) + n$.
iii. $(V' \setminus V) \cap [n+1 \dots 2n] \subseteq ((V' \setminus V) \cap [1 \dots n]) + n \subseteq (Y' \setminus Y) \cap [n+1 \dots 2n]$.

These states are used when trying to find concatenation points in between the w_{j-1} and w_j for all $j \in [2 \dots p-1]$.

3. $(S, (X, q, X'), \emptyset, \emptyset)$ with $(X, q, X') \in \mathcal{Q}$, $S \subseteq \mathcal{Q}$, with the property that for all $(Y, r, Y') \in S$,
   a) $U \cap [n+1 \dots 2n] \supseteq (X \cap [1 \dots n]) + n$.
   b) $U' \cap [n+1 \dots 2n] \supseteq (X' \cap [1 \dots n]) + n$.
   c) $(U' \setminus U) \cap [1 \dots n] \subseteq ((U' \setminus U) \cap [n+1 \dots 2n]) - n \subseteq (X' \setminus X) \cap [1 \dots n]$.

   These states are used when trying to find concatenation points for the last concatenation $w_{p-1} \odot w_p$.

The transitions are the following:

1. $(\emptyset, \emptyset, (X, q, X'), T) \xrightarrow{a} (\emptyset, \emptyset, (X', q', X''), T')$ iff
   a) $q \xrightarrow{a} q'$;
   b) For all $(V', t', V'') \in T'$ there exists $(V, t, V') \in T$ such that $t \xrightarrow{a} t'$.

   These transitions are used during the first search for "concatenation points". The above requirements, corroborated with the consistency requirements on hex-uples, say that, if the active component considers passing through some accepting set Q_{n+i} then each tuple in the prophecy component must also consider passing through the accepting set Q_i.
2. $(S, (X, q, X'), (Y, r, Y'), T) \xrightarrow{a} (S', (X', q', X''), (Y', r', Y''), T')$ iff
   a) $q \xrightarrow{a} q'$, $r \xrightarrow{a} r'$;
   b) For all $(U, s, U') \in S$ there exists $(U,' s', U'') \in S'$ such that $s \xrightarrow{a} s'$;
   c) For all $(V', t', V'') \in T'$ there exists $(V, t, V') \in T$ such that $t \xrightarrow{a} t'$.

   This is the general pattern for the evolution of all the copies of $\mathcal{A}$ during their search for "concatenation points".
3. $(S, (X, q, X'), \emptyset, \emptyset) \xrightarrow{a} (S', (X', q', X''), \emptyset, \emptyset)$ iff
   a) $q \xrightarrow{a} q'$;
   b) For all $(U, s, U') \in S$ there exists $(U', s', U'') \in S'$ such that $s \xrightarrow{a} s'$.

   These transitions are used after finding the last "concatenation points", that is, while parsing the last factor w_p of w.
4. $(S, (X, q, X'), (Y, r, Y'), T) \xrightarrow{\varepsilon} (S', (Y, r, Y'), (Z, s, Z'), T')$ iff
   a) There exists $X'' \in \mathcal{X}$ such that $(X', q, X'') \in S$;
   b) There exists $Y \in \mathcal{X}$ such that $(Y, r, Y') \in T$;
   c) For each $(Z, s, Z') \in S$ there exists $Z'' \in \mathcal{X}$ such that $(Z', s, Z'') \in S$.
   d) For each $(Z', s, Z'') \in T'$ there exists $Z \in \mathcal{X}$ such that $(Z, s, Z') \in T$.

   This transition is taken upon the decision that the current left active component needs to postpone its search for concatenation points – because it has arrived at the "right interface" of the current factor.
5. $(\emptyset, \emptyset, (X, q, X'), T) \xrightarrow{\varepsilon} (\emptyset, (X', q, X''), (Y', r, Y''), T')$ iff

a) There exists $Y \in \mathcal{X}$ such that $(Y, r, Y') \in T$;
b) For each $(Z', s, Z'') \in T'$ there exists $Z \in \mathcal{X}$ such that $(Z, s, Z') \in T$.

Transitions of this type are taken upon decision to activate a second copy of $\mathcal{A}$ in order to check the first set of "concatenation points".

6. $(S, (X, q, X'), (Y, r, Y'), \emptyset) \xrightarrow{\varepsilon} (S', (Y, r, Y'), \emptyset, \emptyset)$ iff
   a) There exists $X'' \in \mathcal{X}$ such that $(X', q, X'') \in S$;
   b) For each $(Z, s, Z') \in S$ there exists $Z'' \in \mathcal{X}$ such that $(Z', s, Z'') \in S$.

   This transition is taken upon decision that the latest set of "concatenation points" should be the last one to be checked.

The accepting sets are, for all $i \in [1 \dots n]$,

$$U_i = \big\{(S, (X, q, X'), (Y, r, Y'), T) \mid i \in X \setminus X' \text{ and for all } (Z, s, Z') \in S, i \in Z' \setminus Z\big\} \cup \big\{(\emptyset, \emptyset, (X, q, X'), T) \mid i \in X' \setminus X\big\} \tag{7.12}$$

$$U_{n+i} = \big\{(S, (X, q, X'), (Y, r, Y'), T) \mid i+n \in Y' \setminus Y \text{ and for all } (Z, s, Z') \in T, i+n \in Z' \setminus Z\big\} \cup \big\{(S, (X, q, X'), \emptyset, \emptyset) \mid n + i \in X' \setminus X\big\} \tag{7.13}$$

Finally, we restrict the state space only to states *reachable* from $\Big(\emptyset, \emptyset, (\emptyset, q_*, \emptyset), \big\{(\emptyset, q_*, \emptyset)\big\}\Big)$ and *coreachable* from $\Big(\big\{([1 \dots 2n], q_{**}, [1 \dots 2n])\big\}, ([1 \dots 2n], q_{**}, [1 \dots 2n]), \emptyset, \emptyset\Big)$. Let us denote $\mathcal{D} = (Q_{\circledast}, \delta_{\circledast}, U_1, \dots, U_{2n})$ the automaton build above.

Claim. $L(\mathcal{A})^{\circledast \geq 2} \supseteq L(\mathcal{D})$.

Proof (of the Claim). Consider some accepting run in $\mathcal{A}$,

$$\rho = \big((S_i, \alpha_i, \beta_i, T_i), a_i, (S_{i+1}, \alpha_{i+1}, \beta_{i+1}, T_{i+1})\big)_{i \in [1 \dots m-1]}$$

with

$$(S_1, \alpha_1, \beta_1, T_1) = \Big(\emptyset, \emptyset, (\emptyset, q_*, \emptyset), \big\{(\emptyset, q_*, \emptyset)\big\}\Big)$$
$$(S_m, \alpha_m, \beta_m, T_m) = \Big(\big\{([1 \dots 2n], q_{**}, [1 \dots 2n])\big\}, ([1 \dots 2n], q_{**}, [1 \dots 2n]), \emptyset, \emptyset\Big)$$

Consider also some non-elastic $2n$-word $w \in \mathcal{WD}_{2n}(\Sigma)$ and a sequence $(p_i)_{i \in [1 \dots 2n]}$ with $p_i \in [1 \dots m]$, sequence which witnesses the acceptance of w by ρ, that is,

- $(S_{p_i}, \alpha_{p_i}, \beta_{p_i}, T_{p_i}) \in U_i$ for all $i \in [1 \dots 2n]$, and
- $(S_{p_i}, \alpha_{p_i}, \beta_{p_i}, T_{p_i}) \xrightarrow{w_{ij}} (S_{p_j}, \alpha_{p_j}, \beta_{p_j}, T_{p_j})$ for all $i, j \in [1 \dots 2n]$.

Since the first state in the run is $\Big(\emptyset, \emptyset, (\emptyset, q_*, \emptyset), \big\{(\emptyset, q_*, \emptyset)\big\}\Big)$ and the last state in the run is $\Big(\big\{([1 \dots 2n], q_{**}, [1 \dots 2n])\big\}, ([1 \dots 2n], q_{**}, [1 \dots 2n]), \emptyset, \emptyset\Big)$, the run must contain some ε-transitions which shift tuples from the right active to the left active component or make tuples "arise" or "disappear" in the left and right active components. More specifically, we may find an

increasing sequence of indices $(k_j)_{j\in[1\dots p]}$ such that the k_j-th transition in ρ is of type 4,5 or 6. At a closer look, we note that the k_1-th transition must be of the type 4, the k_p-th transition must be of the type 6 and the transitions $k_2, \dots, k_{p-1}$ must be of the type 5, and also that $p \geq 3$.

More formally:

$$\begin{aligned}
S_{k_1} &= \alpha_{k_1} = S_{k_1+1} = \emptyset \\
\alpha_{k_1+1} &= \beta_{k_1} = (X_{k_1}, q_{k_1}, X'_{k_1}) \\
\beta_{k_1+1} &= (X_{k_1+1}, q_{k_1+1}, X'_{k_1+1}) \\
T_{k_1} &\supseteq \big\{(Y, r, Y') \mid \exists Y'' \in \mathcal{X} \text{ such that } (Y', r, Y'') \in T_{k_1+1} \text{ or } (Y', r, Y'') = \beta_{k_1+1}\big\} \\
T_{k_p} &= \beta_{k_p+1} = T_{k_1+1} = \emptyset \\
\alpha_{k_p+1} &= \beta_{k_p} = (X_{k_p+1}, q_{k_p+1}, X'_{k_p+1}) \\
\alpha_{k_p} &= (X_{k_p}, q_{k_p}, X'_{k_p}) \\
S_{k_p+1} &\supseteq \big\{(Y', r, Y'') \mid \exists Y \in \mathcal{X} \text{ such that } (Y, r, Y') \in S_{k_p} \text{ or } (Y, r, Y') = \alpha_{k_p}\big\} \\
\alpha_{k_j+1} &= \beta_{k_j} \\
S_{k_j+1} &\supseteq \big\{(Y', r, Y'') \mid \exists Y \in \mathcal{X} \text{ such that } (Y, r, Y') \in S_{k_j} \text{ or } (Y, r, Y') = \alpha_{k_j}\big\} \\
T_{k_j} &\supseteq \big\{(Y, r, Y') \mid \exists Y'' \in \mathcal{X} \text{ such that } (Y', r, Y'') \in T_{k_j+1} \text{ or } (Y', r, Y'') = \beta_{k_j+1}\big\}
\end{aligned}$$

Our aim is to show that w, the $2n$-word associated with ρ, can be factored into p $2n$-words, $w = w_1 \odot \dots \odot w_p$ with $w_j \in L(\mathcal{A})$ for all $j \in [1 \dots p]$

Let us denote also

$$\begin{aligned}
\alpha_i &= (X_i, q_i, X'_i), \text{ for all } i \in [k_1 + 1 \dots m] \\
\beta_i &= (Y_i, r_i, Y'_i), \text{ for all } i \in [1 \dots k_p]
\end{aligned}$$

Observe first that the following concatenation of transitions:

$$\rho_1 = \big((r_i, a_i, r_{i+1})_{i\in[1\dots k_1-1]}, (r_{k_1}, \varepsilon, q_{k_1+1}), (q_i, a_i, q_{i+1})_{i\in[k_1+1\dots k_2-1]}\big)$$

is a run in $\mathcal{A}$, since the k_1-th moment corresponds to the shift of the right active component into the left active component.

Moreover, the sequence $\big((Y'_i)_{i\in[1\dots k_1]}, (X'_i)_{i\in[k_1+1\dots k_2]}\big)$ records the indices of accepting states of $\mathcal{A}$ through which ρ_1 has passed. Since it is possible that $X'_{k_2} \neq [1 \dots 2n]$, we cannot say that this run is accepting. We would therefore like to extend it to an accepting run, by carefully extracting more information from the history components S_i with $i > k_2$:

We extend ρ_1 to a run $\overline{\rho}_1 = \big((Z_i^1, s_i^1, \overline{Z}_i^1), a_i, (Z_i^1, s_i^1, \overline{Z}_i^1)\big)_{i\in[1\dots m]}$ by induction by choosing, for each $i \geq k_2$, a tuple $(Z_i^1, s_i^1, \overline{Z}_i^1) \in S_i$ such that $s_i^1 \xrightarrow{a_i} s_{i+1}^1$ in $\mathcal{A}$. The fact that $\overline{\rho}_1$ extends ρ_1 means that

$$Z_i^1 = \begin{cases} Y_i & \text{for } i \in [1 \dots k_1] \\ X_i & \text{for } i \in [k_1+1 \dots k_2] \end{cases} \qquad \overline{Z}_i^1 = \begin{cases} Y_i' & \text{for } i \in [1 \dots k_1] \\ X_i' & \text{for } i \in [k_1+1 \dots k_2] \end{cases}$$

$$s_i = \begin{cases} r_i & \text{iff } i \in [1 \dots k_1] \\ q_i & \text{iff } i \in [k_1+1 \dots k_2] \end{cases}$$

The possibility to choose at each step a state s_{i+1}^1 is assured by the requirements 2.b, 3.b, 4.c, and/or 6.b, that transitions in $\mathcal{D}$ must obey, according to the situation in which the i-th tuple $(S_i, \alpha_i, \beta_i, T_i)$ falls and also according to the label a_i. Consequently we have that $Z_{i+1}^1 = \overline{Z}_i^1$ for all $i \in [1 \dots m]$.

Observe that $Z_1 = \overline{Z}_1 = \emptyset$ and the last tuple in $\overline{\rho}_1$ must belong to S_m and hence must be $([1 \dots 2n], q_{**}, [1 \dots 2n])$, therefore $Z_m = \overline{Z}_m = [1 \dots 2n]$. Moreover, for each $j \in \overline{Z}_i^1 \setminus Z_i^1$ we must have, by construction, that $s_i^1 \in Q_j$. Hence $\overline{\rho}_1$ is an accepting run in $\mathcal{A}$ in the "history" presentation, run that starts in q_*, ends in q_{**}.

Therefore, if we associate to this run the sequence of indices $\boldsymbol{l}^1 = (l_u^1)_{u \in [1 \dots 2n]}$ with $l_u^1 = i$ iff $u \in \overline{Z}_i^1 \setminus Z_i^1$, then this sequence witnesses that $\overline{\rho}_1$ accepts some $2n$-word w_1. Observe that $(\overline{\rho}_1, \boldsymbol{l}^1)$ is a non-elastic pair, hence w_1 is a non-elastic $2n$-word. Our intuition is that w_1 is the ***first*** factor of w in its decomposition into $2n$-words from $L(\mathcal{A})$.

The remaining factors can be recovered, together with their accepting runs, by generalizing the above argument: for each $j \in [1 \dots p]$ the concatenation

$$\rho_j = \left((r_i, a_i, r_{i+1})_{i \in [k_{j-1}+1 \dots k_j - 1]}, (r_{k_j}, \varepsilon, q_{k_j+1}), (q_i, a_i, q_{i+1})_{i \in [k_j+1 \dots k_{j+1}-1]}\right)$$

is a run in $\mathcal{A}$ that we intuit to correspond to a part of an accepting run associated with a $2n$-word w_j; we have denoted here $k_0 = 0$ and $k_{p+1} = m$. We extend this run in both directions to an accepting run $\overline{\rho}_j$ which starts in q_* and ends in q_{**} as follows:

- For the part of the run in between $k_{j-1}+1$ and k_j we put $s_i^j = r_i$, $Z_i^j = Y_i$ and $\overline{Z}_i^j = Y_i'$, while
- For the part of the run in between $k_j + 1$ and k_{j+1} we put $s_i^j = q_i$, $Z_i^j = X_i$ and $\overline{Z}_i^j = X_i'$.
- Suppose we have build the run from $i+1$ to k_{j+1}, for some. $i \leq k_j$. We then choose a tuple $(Z_i^j, s_i^j, \overline{Z}_i^j) \in T_i$ such that, $s_i^j \xrightarrow{a_i} s_{i+1}^j$ in $\mathcal{A}$.
  The availability of this choice follows from the requirements 1.b, 2.c, 4.d, respectively 5.b from the definition of the transition function of $\mathcal{D}$.
- Suppose also that we have build the run from k_{j-1} to $i-1$ for some $i \geq k_{j+1}+1$ We then choose a tuple $(Z_i^j, s_i^j, \overline{Z}_i^j) \in S_i$ such that if x denotes the $(i-1)$-th transition in ρ, then then $s_{i-1} \xrightarrow{x} s_i$ in $\mathcal{A}$.
  The possibility to choose is assured by the requirements 2.b, 3.b, 4.c, respectively 6.b from the definition of the transition function in $\mathcal{D}$.

Also observe that in each such run, for each $i \in [1 \dots p]$, we have that $Z_{i+1}^j = \overline{Z}_i^j$ as assured by the definition of the transition function $\delta_{\circledast}$. We call this property the *Consecutiveness Property*.

Finally, we associate to each run $\overline{\rho}_j$ the sequence of integers

$$\boldsymbol{l}^j = (l^j_u)_{u\in[1\ldots 2n]} \text{ in which } l^j_u = i \text{ iff } u \in \overline{Z}^j_i \setminus Z^j_i.$$

This sequence witnesses that $\overline{\rho}_j$ accepts some $2n$-word w_j in $\mathcal{A}$, which is actually a non-elastic $2n$-word.

Though we have identified these accepting runs in $\mathcal{A}$, we still need to prove that w_j correctly concatenates to w_{j+1} and that the result of concatenating all w_j is w. To this end we prove the following property:

(*)	For all $j \in [1\ldots p-1]$ and for all $i \in [1\ldots m]$,
	$Z^j_i \cap [n+1\ldots 2n] = (Z^{j+1}_i \cap [1\ldots n]) + n$ and
	$\overline{Z}^j_i \cap [n+1\ldots 2n] = (\overline{Z}^{j+1}_i \cap [1\ldots n]) + n$.

In other words, we prove that $\overline{\rho}_j$ passes through some accepting set Q_{n+i} at the same moment when $\overline{\rho}_{j+1}$ passes through the accepting set Q_i for the same $i \in [1\ldots n]$. It is clear that this requirement is sufficient for proving that w_j and w_{j+1} correctly concatenate.

For proving the desired property (*), let us observe first that for all $i \in [k_j+1\ldots k_{j+1}]$, the run $\overline{\rho}_j$ "gives" the left active component and $\overline{\rho}_{j+1}$ "gives" the right active component of ρ. That is, $(Z^j_i, s^j_i, \overline{Z}^j_i) = (X_i, q_i, X'_i)$ and $(Z^{j+1}_i, s^{j+1}_i, \overline{Z}^{j+1}_i) = (Y_i, r_i, Y'_i)$. But then, by construction (requirements 2.a and 2.b in the definition of $Q_\circledast$), we have $X_i \cap [n+1\ldots 2n] = Y_i \cap [1\ldots n] + n$ and $X'_i \cap [n+1\ldots 2n] = Y'_i \cap [1\ldots n] + n$. This implies that our property (*) holds over the interval $[k_j+1\ldots k_{j+1}]$. Observe also that the interval $[k_j+1\ldots k_{j+1}]$ is nonempty since the sequence $(k_j)_{j\in[1\ldots p]}$ is strictly increasing.

Consider now the interval $[k_{j-1}+1\ldots k_j]$. In this interval, $\overline{\rho}_j$ is the right active component while $\overline{\rho}_{j+1}$ is included in the prophecy component. That is, $(Z^j_i, s^j_i, \overline{Z}^j_i) = (Y_i, r_i, Y'_i)$ and $(Z^{j+1}_i, s^{j+1}_i, \overline{Z}^{j+1}_i) \in T_i$. Note that half of the property (*) holds for $i = k_j$: by construction, we have $\overline{Z}^j_{k_j} = Z^j_{k_j+1}$ and $\overline{Z}^{j+1}_{k_j} = Z^{j+1}_{k_j+1}$. On the other hand, the property (*) holds for k_j+1 as proved above, hence we have $Z^j_{k_j+1} \cap [n+1\ldots 2n] = (Z^{j+1}_{k_j+1} \cap [1\ldots n]) + n$. Therefore, by the Consecutiveness Property, $\overline{Z}^j_{k_j} \cap [n+1\ldots 2n] = (\overline{Z}^{j+1}_{k_j} \cap [1\ldots n]) + n$.

We will then prove the property (*) by a "decreasing induction" argument: suppose that $\overline{Z}^j_i \cap [n+1\ldots 2n] = (\overline{Z}^{j+1}_i \cap [1\ldots n]) + n$ for some $i \in [k_{j-1}+1\ldots k_j]$. Since $(Z^{j+1}_{i-1}, s^{j+1}_{i-1}, \overline{Z}^{j+1}_{i-1}) \in T_{i-1}$ and $(Z^j_{i-1}, s^j_{i-1}, \overline{Z}^j_{i-1})$ is the right active component, we must have by construction (requirements 2.c.i and 2.c.iii in the definition of $Q_\circledast$) that

$$Z^j_{i-1} \cap [n+1\ldots 2n] \supseteq (Z^{j+1}_{i-1} \cap [1\ldots n]) + n \tag{7.14}$$

$$\big((\overline{Z}^{j+1}_{i-1} \setminus Z^{j+1}_{i-1}) \cap [n+1\ldots 2n]\big) + n \subseteq (\overline{Z}^j_{i-1} \setminus Z^j_{i-1}) \cap [n+1\ldots 2n] \tag{7.15}$$

We then have the following sequence of identities:

$$\begin{aligned}
Z^j_{i-1} \cap [n+1 \ldots 2n] &= \big(\overline{Z}^j_{i-1} \cap [n+1 \ldots 2n]\big) \setminus \big((\overline{Z}^j_{i-1} \setminus Z^j_{i-1}) \cap [n+1 \ldots 2n]\big) \\
&= \big((\overline{Z}^{j+1}_{i-1} \cap [1 \ldots n]) + n\big) \setminus \big((\overline{Z}^j_{i-1} \setminus Z^j_{i-1}) \cap [n+1 \ldots 2n]\big) \\
&\qquad \text{(by assumption)} \\
&\subseteq \big((\overline{Z}^{j+1}_{i-1} \cap [1 \ldots n]) + n\big) \setminus \big((\overline{Z}^{j+1}_{i-1} \setminus Z^{j+1}_{i-1}) \cap [1 \ldots n]\big) + n\big) \\
&\qquad \text{(by inclusion 7.15)} \\
&= (Z^{j+1}_{i-1} \cap [1 \ldots n]) + n \\
&\subseteq Z^j_{i-1} \cap [n+1 \ldots 2n] \qquad (7.16)
\end{aligned}$$

From identities 7.14 and 7.16 we get by double inclusion that $Z^j_{i-1} \cap [n+1 \ldots 2n] = (Z^{j+1}_{i-1} \cap [1 \ldots n]) + n$, that is, the other half of property (*) holds for $i-1$.

Let us now consider the intervals $[k_v + 1 \ldots k_{v+1}]$ for $v \leq j-2$. Within these intervals, both $\overline{\rho}_j$ and $\overline{\rho}_{j+1}$ take part into the prophecy component, i.e., $(Z^j_i, s^{j+1}_i, \overline{Z}^j_i), (Z^{j+1}_i, s^{j+1}_i, \overline{Z}^{j+1}_i) \in T_i$. Again as above, the validity of the property (*) within the interval $[k_{v+1} + 1 \ldots k_{v+2}]$ and the Consecutiveness Property assures that half of the property (*) holds for $i = k_v$, namely $\overline{Z}^j_{k_v} \cap [n + 1 \ldots 2n] = (\overline{Z}^{j+1}_{k_v} \cap [1 \ldots n]) + n$. We will prove by decreasing induction that it holds within all the interval $[k_v + 1 \ldots k_{v+1}]$.

Let us provide first the properties that connect the right active component of the $(i-1)$-th tuple in the run ρ with the tuples of the runs $\overline{\rho}_j$ and $\overline{\rho}_{j+1}$, as implied by the definition of $Q_{\circledast}$:

$$Y_{i-1} \cap [n+1 \ldots 2n] \supseteq (Z^j_{i-1} \cap [1 \ldots n]) + n \qquad (7.17)$$

$$Y_{i-1} \cap [n+1 \ldots 2n] \supseteq (Z^{j+1}_{i-1} \cap [1 \ldots n]) + n \qquad (7.18)$$

$$Y'_{i-1} \cap [n+1 \ldots 2n] \supseteq (\overline{Z}^j_{i-1} \cap [1 \ldots n]) + n \qquad (7.19)$$

$$Y'_{i-1} \cap [n+1 \ldots 2n] \supseteq (\overline{Z}^{j+1}_{i-1} \cap [1 \ldots n]) + n \qquad (7.20)$$

$$\big((\overline{Z}^j_{i-1} \setminus Z^j_{i-1}) \cap [1 \ldots n]\big) + n \subseteq (Y'_{i-1} \setminus Y_{i-1}) \cap [n+1 \ldots 2n] \qquad (7.21)$$

$$\big((\overline{Z}^{j+1}_{i-1} \setminus Z^{j+1}_{i-1}) \cap [1 \ldots n]\big) + n \subseteq (Y'_{i-1} \setminus Y_{i-1}) \cap [n+1 \ldots 2n] \qquad (7.22)$$

Suppose that $Z^j_i \cap [n+1 \ldots 2n] = (Z^{j+1}_i \cap [1 \ldots n]) + n$, hence, by the Consecutiveness Property, $\overline{Z}^j_{i-1} \cap [n+1 \ldots 2n] = (\overline{Z}^{j+1}_{i-1} \cap [1 \ldots n]) + n$. Denote further $V = (Z^j_{i-1} \cap [n+1 \ldots 2n]) \setminus \big((Z^{j+1}_{i-1} \cap [1 \ldots n]) + n\big)$. Since $Z^j_{i-1} \in \mathcal{X}$, we have that $Z^j_{i-1} \cap [n+1 \ldots 2n] \subseteq (Z^j_{i-1} \cap [1 \ldots n]) + n$. Hence, by inclusion 7.17,

$$V \subseteq (Z^j_{i-1} \cap [1 \ldots n]) + n \subseteq Y_{i-1} \cap [n+1 \ldots 2n]$$

On the other hand,

$$\begin{aligned}
V &\subseteq \big((Z^j_{i-1} \cap [1 \ldots n]) + n\big) \setminus \big((Z^{j+1}_{i-1} \cap [1 \ldots n]) + n\big) && \text{(since } Z^j_{i-1} \in \mathcal{X}\text{)} \\
&= \big((Z^j_{i-1} \setminus Z^{j+1}_{i-1}) \cap [1 \ldots n]\big) + n \\
&\subseteq \big((\overline{Z}^j_{i-1} \setminus Z^{j+1}_{i-1}) \cap [1 \ldots n]\big) + n \\
&= \big((\overline{Z}^{j+1}_{i-1} \setminus Z^{j+1}_{i-1}) \cap [1 \ldots n]\big) + n && \text{(by induction hypothesis)} \\
&\subseteq (Y'_{i-1} \setminus Y_{i-1}) \cap [n+1 \ldots 2n] && \text{(by inclusion 7.22)}
\end{aligned}$$

Hence, in order to avoid the contradiction with $V \subseteq Y_{i-1} \cap [n+1 \ldots 2n]$ we must have $V = \emptyset$. Similarly, if we denote $V' = \big((Z_{i-1}^{j+1} \cap [1 \ldots n]) + n\big) \setminus (Z_{i-1}^{j} \cap [n+1 \ldots 2n])$, we get that

$$\begin{aligned}
V' &\subseteq Y_{i-1} \cap [n+1 \ldots 2n] \\
V' &\subseteq \big((\overline{Z}_{i-1}^{j+1} \cap [1 \ldots n]) + n\big) \setminus (Z_{i-1}^{j} \cap [n+1 \ldots 2n]) \\
&= \big((\overline{Z}_{i-1}^{j+1} \cap [1 \ldots n]) + n\big) \setminus \big((Z_{i-1}^{j} \cap [1 \ldots n]) + n\big) \\
&= \big((\overline{Z}_{i-1}^{j+1} \setminus Z_{i-1}^{j}) \cap [1 \ldots n]\big) + n \\
&= \big((\overline{Z}_{i-1}^{j} \setminus Z_{i-1}^{j}) \cap [1 \ldots n]\big) + n \\
&\subseteq (Y'_{i-1} \setminus Y_{i-1}) \cap [n+1 \ldots 2n]
\end{aligned}$$

which implies $V' = \emptyset$ too. Hence, $Z_{i-1}^{j} \cap [n+1 \ldots 2n] = (Z_{i-1}^{j+1} \cap [1 \ldots n]) + n$ and this proves that property (*) holds within the interval $[1 \ldots k_{j+1}]$.

By mirroring the above arguments, we may show that property (*) also holds within the interval $[k_{j+1}+1 \ldots m]$. We only show the argument for the intervals $[k_v+1 \ldots k_{v+1}]$ for $v \geq j+2$:

In this interval, both $\overline{\rho}_j$ and $\overline{\rho}_{j+1}$ take part in the history component. Since the property holds for $i = k_v$, we have $\overline{Z}_{k_v}^{j} \cap [n+1 \ldots 2n] = (\overline{Z}_{k_v}^{j+1} \cap [1 \ldots n]) + n$. But since $\overline{Z}_{k_v}^{j} = Z_{k_v+1}^{j}$ and $\overline{Z}_{k_v}^{j+1} = Z_{k_v+1}^{j+1}$ we also have that the "first half" of the property (*) holds for $i = k_v + 1$, that is, $Z_{k_v+1}^{j} \cap [n+1 \ldots 2n] = (Z_{k_v+1}^{j+1} \cap [1 \ldots n]) + n$. We will prove by increasing induction that it holds for all $i \in [k_v+1 \ldots k_{v+1}]$.

Suppose that $\overline{Z}_{i}^{j} \cap [n+1 \ldots 2n] = (\overline{Z}_{i}^{j+1} \cap [1 \ldots n]) + n$. Since $Z_{i+1}^{j} = \overline{Z}_{i}^{j}$ and $Z_{i+1}^{j+1} = \overline{Z}_{i}^{j+1}$ we then get that $Z_{i+1}^{j} \cap [n+1 \ldots 2n] = (Z_{i+1}^{j+1} \cap [1 \ldots n]) + n$, so it only remains to check the other half of the property (*).

Denote $V = (\overline{Z}_{i+1}^{j} \cap [n+1 \ldots 2n]) \setminus \big((\overline{Z}_{i+1}^{j+1} \cap [1 \ldots n]) + n\big)$. We then have:

$$\begin{aligned}
V &\subseteq (\overline{Z}_{i+1}^{j} \cap [n+1 \ldots 2n]) \setminus \big((Z_{i+1}^{j+1} \cap [1 \ldots n]) + n\big) && \text{(since } \overline{Z}_{i+1}^{j+1} \supseteq Z_{i+1}^{j+1}\text{)} \\
&= (\overline{Z}_{i+1}^{j} \cap [n+1 \ldots 2n]) \setminus (Z_{i+1}^{j} \cap [n+1 \ldots 2n]) && \text{(by induction hypothesis)} \\
&= (\overline{Z}_{i+1}^{j} \setminus Z_{i+1}^{j}) \cap [n+1 \ldots 2n]) \\
&\subseteq \big((X'_{i+1} \setminus X_{i+1}) \cap [1 \ldots n]\big) + n && \text{(by requirement 2.c.iii)} \\
&\subseteq (X'_{i+1} \cap [1 \ldots n]) + n \\
&\subseteq \overline{Z}_{i+1}^{j+1} \cap [n+1 \ldots 2n] && \text{(by requirement 2.c.ii)} \\
&\subseteq (\overline{Z}_{i+1}^{j+1} \cap [1 \ldots n]) + n && \text{(since } \overline{Z}_{i+1}^{j+1} \in \mathcal{X}\text{)}
\end{aligned}$$

(references are to requirements in the definition of $Q_{\circledast}$) which proves that $V = \emptyset$ in order to avoid contradiction between the first and the last inclusion.

On the other hand, if we denote $V' = \big((\overline{Z}_{i+1}^{j+1} \cap [1 \ldots n]) + n\big) \setminus (\overline{Z}_{i+1}^{j} \cap [n+1 \ldots 2n])$ we would have the following sequence of inclusions:

$$\begin{aligned}
V' &\subseteq \big((\overline{Z}^{j+1}_{i+1} \cap [1 \dots n]) + n\big) \setminus (Z^{j}_{i+1} \cap [n+1 \dots 2n]) \\
&= \big((\overline{Z}^{j+1}_{i+1} \cap [1 \dots n]) + n\big) \setminus \big((Z^{j+1}_{i+1} \cap [1 \dots n]) + n\big) \\
&= \big((\overline{Z}^{j+1}_{i+1} \setminus Z^{j+1}_{i+1}) \cap [1 \dots n]\big) + n \\
&\subseteq \big((X'_{i+1} \setminus X_{i+1}) \cap [1 \dots n]\big) + n && \text{(by requirement 2.c.iii)} \\
&\subseteq (X'_{i+1} \cap [1 \dots n]) + n \\
&\subseteq \overline{Z}^{j}_{i+1} \cap [n+1 \dots 2n] && \text{(by requirement 2.c.ii)}
\end{aligned}$$

Hence again $V' = \emptyset$ in order to avoid the contradiction between the first and the last inclusion.

It remains to prove that $l^1_i = p_i$ and $l^k_{n+i} = p_{n+i}$ for all $i \in [1 \dots n]$, that is, that the concatenation $w_1 \odot \dots \odot w_k$ really gives w. But this property is evidently true, since, by construction of the accepting sets in $\mathcal{D}$ (Identities 7.12 and 7.13), regardless of the component to which $(Z^1_{p_i}, q^1_{p_i}, \overline{Z}^1_{p_i})$ belongs (but note it may belong only to the right or left active component or to the history component), we have that $i \in \overline{Z}^1_{p_i} \setminus Z^1_{p_i}$, and similarly for p_{n+i}.

This ends our proof of our first Claim. □

Claim. $L(\mathcal{A})^{\circledast \geq 2} \subseteq L(\mathcal{D})$

Proof. Start with a concatenation $w = w_1 \odot \dots \odot w_p$ with $w_i \in L(\mathcal{A})$ for all $i \in [1 \dots p]$, and consider p accepting runs in the completed $2n$-automaton $\tilde{\mathcal{A}}$, one for each w_i, together with their witnessing sequences of indices:

$$\rho_i = (q^i_j, a^i_j, q^i_{j+1})_{j \in m_i - 1} \text{ with witnessing index sequence } (l^i_k)_{k \in [1 \dots 2n]}$$

We assume that each run starts in q_* and ends in q_{**}.

Transform these runs into runs in the "history" presentations, that is, denote X^i_j the set of indices of the accepting states which were visited by each run ρ_i *just before* the j-th step and by $\overline{X}^i_j$ the set of indices of accepting states visited by ρ_i *up to* the j-th step, and also denote Δ^i_j their difference:

$$\begin{aligned}
X^i_j &= \big\{u \in [1 \dots 2n] \mid \exists v \in [1 \dots j-1] \text{ such that } l^i_u = v\big\} \\
\overline{X}^i_j &= \big\{u \in [1 \dots 2n] \mid \exists v \in [1 \dots j] \text{ such that } l^i_u = v\big\} \\
\Delta^i_j &= \overline{X}^i_j \setminus X^i_j.
\end{aligned}$$

Note first that, due to the fact that w_i and w_{i+1} correctly concatenate, we have that for all $u, v \in [1 \dots n]$ and for all $j_1, j_2 \in [1 \dots m_i]$, $j_3, j_4 \in [1 \dots m_{i+1}]$ if $u \in \Delta^i_{j_1}, v \in \Delta^i_{j_2}$ and $u + n \in \Delta^{i+1}_{j_3}, v + n \in \Delta^{i+1}_{j_4}$ then $j_1 - j_2 = j_3 - j_4$.

By a trick similar to the right-to-left proof for concatenation, we bring all runs to equal length such that when ρ_i passes through Q_{n+j}, ρ_{i+1} passes through Q_j and vice-versa (we call this property the *Synchronicity Property*)

To this end, we extend each run ρ_i to a run $\tilde{\rho}_i$ of length m (the same for all i-s) by adding loops in q_* and/or q_{**}, then suitably redefine the indices l^i_j and the index sets X^i_j, $\overline{X}^i_j$ and Δ^i_j such that

for all $i \in [1 \dots p-1]$ we have $\Delta_j^i \cap [n+1 \dots 2n] = (\Delta_j^{i+1} \cap [1 \dots n]) + n$. This can be done as follows:

Take some $u \in [1 \dots n]$. Then, for each $i \in [1 \dots p]$ define $\gamma_i \in [1 \dots m_i]$ as the *unique* index for which $u \in \Delta_{\gamma_i}^i$. Similarly, define $\overline{\gamma}_i \in [1 \dots m_i]$ as the *unique* index for which $u+n \in \Delta_{\overline{\gamma}_i}^i$. Define further $\delta_i = \overline{\gamma}_i - \gamma_{i+1}$.

This integer gives the number of loops in q_* that must be appended to ρ_{i+1} such that it is brought to the same length as ρ_i.

This follows because, if the runs would have the Synchronicity Property then γ_{i+1} must equal $\overline{\gamma}_i$, regardless of the choice of the index $u \in [1 \dots n]$.

Then, for each $i \in [2 \dots p]$ we append, at the beginning of ρ_i, $\delta_1 + \ldots + \delta_i$ copies of the state q_*. Also, the indices that witness the acceptance of w_i are then shifted by $\sum_{j=1}^{i-1} \delta^j$, that is,

$$\tilde{l}_k^i = l_k^i + \sum_{j=1}^{i-1} \delta^j.$$

It is routine to check then that, after suitably redefining the index sets X_j^i, $\overline{X}_j^i$ and Δ_j^i, we have that

$$\forall i \in [1 \dots p-1], \forall j \in [1 \dots m], \Delta_j^i \cap [n+1 \dots 2n] = (\Delta_j^{i+1} \cap [1 \dots n]) + n \tag{7.23}$$

However the runs do not have the same length yet. To bring them to the same length, define first $m = \max\left\{m_i + \sum_{j=1}^{i-1} \delta^j \mid i \in [1 \dots p]\right\}$, and then append, at the end of each run ρ_i, $m - m_i$ copies of the state q_{**}, and we are done.

We suppose from now on that the runs ρ_i have equal length m and that their associated index sets satisfy condition 7.23. Observe that, by the hypothesis that all $2n$-words w_i correctly concatenate, we may choose the runs ρ_i such that *all the labels of the transitions are the same* (a similar property was obtained for concatenation), hence we may consider that there exist $a_1 \dots a_m \in \Sigma$ such that for all $i \in [1 \dots p]$,

$$\rho_i = (q_j^i, a_j, q_{j+1}^i)_{j \in m-1}$$

Let us show now some properties of the index sets X_j^i, $\overline{X}_j^i$ and Δ_j^i:

(X1) For each $i \in [1 \dots p]$ and $j \in [1 \dots m]$, $X_j^i, \overline{X}_j^i \in \mathcal{X}$.

(X2) For each $i \in [1 \dots p-1]$ and $j \in [1 \dots m]$, $X_j^i \cap [n+1 \dots 2n] = (X_j^{i+1} \cap [1 \dots n]) + n$ and $\overline{X}_j^i \cap [n+1 \dots 2n] = (\overline{X}_j^{i+1} \cap [1 \dots n]) + n$

(X3) For each $i \in [1 \dots p-1]$ and $j \in [1 \dots m]$, $X_j^i \subseteq X_j^{i+1}$ and $\overline{X}_j^i \subseteq \overline{X}_j^{i+1}$.

(X4) For each $1 \leq i < i' \leq p$ and $j \in [1 \dots m]$, $X_j^i \cap [n+1 \dots 2n] \supseteq (X_j^{i'} \cap [1 \dots n]) + n$ and $\overline{X}_j^i \cap [n+1 \dots 2n] \supseteq (\overline{X}_j^{i'} \cap [1 \dots n]) + n$.

Property *(X1)* follows due to the fact that $\overline{X}_j^i = X_j^i \cup \Delta_j^i$, while property *(X2)* follows by induction on j:

$$\begin{aligned} X^i_{j+1} \cap [n+1 \ldots 2n] &= (X^i_j \cap [n+1 \ldots 2n]) \cup (\Delta^i_{j+1} \cap [n+1 \ldots 2n]) \\ &= \big((X^{i+1}_j \cap [1 \ldots n]) + n\big) \cup \big((\Delta^{i+1}_{j+1} \cap [1 \ldots n]) + n\big) \\ &= (X^{i+1}_{j+1} \cap [1 \ldots n]) + n \end{aligned}$$

Property *(X3)* is a consequence of the first two properties, while property *(X4)* follows by induction on $i' - i$, since the first three properties imply that

$$\begin{aligned} (X^{i'}_j \cap [1 \ldots n]) + n = \overline{X}^{i'-1}_j \cap [n+1 \ldots 2n] \subseteq (X^{i'-1}_j \cap [1 \ldots n]) + n = \ldots \\ \ldots \subseteq (X^{i+1}_j \cap [1 \ldots n]) + n = X^i_j \cap [n+1 \ldots 2n] \end{aligned}$$

The idea that guides us in building the $\mathcal{D}$ run for w is that the union of the history, left active, right active and prophecy components at each step j in ρ must be the set $\{(X^i_j, q^i_j, \overline{X}^i_j) \mid i \in [1 \ldots p]\}$. The problem is to correctly choose the left and the right active components at each step, and to check the additional constraints on the states of this run, and it is here where the non-elastic assumption plays its role.

Clearly, the order in which the runs ρ_i "become" left active components is the order of concatenation. Or, in other words, if $(X^i_j, q^i_j, \overline{X}^i_j)$ is the left active component at step j, then the right active component is $(X^{i+1}_j, q^{i+1}_j, \overline{X}^{i+1}_j)$, the prophecy component is $\big\{(X^{i'}_j, q^{i'}_j, \overline{X}^{i'}_j) \mid i' > i+1\big\}$ and the history component is $\big\{(X^{i'}_j, q^{i'}_j, \overline{X}^{i'}_j) \mid i' < i\big\}$.

The choice of the moments at which a run ρ_i passes from the history component into the right active component, then into the left active component and finally into the history component need not be unique. But the bottom line is that ρ_i cannot "sleep" all the time it parses the contribution part of w_i This translates to the fact that ρ_i cannot be in the history component at the end of $(w_i)_{u,n+u}$ and cannot be in the prophecy component before the beginning of $(w_i)_{u,n+u}$.

The non-elasticity property intervenes then in the fact that, if we have decided to shift ρ_i, say from the prophecy into the history component, then we will never "regret" this decision, that is, we will never need ρ_i in the prophecy component back.

This means that each run ρ_i will be, at its turn, in the prophecy component, then a left active component, then a right active component, and finally in the history component. Observe then that each time we shift the left active component into the right active component we must employ an ε-transition. This means that the run ρ would have length $m+p-m$, since it simulates each transition in all the runs ρ_i and p since there must be p ε-transitions for shifting left active into right active components.

Our choice for the shifting moments is a "lazy" one: a run ρ_i is moved from its place only when there is a run $\rho_{i'}$ with $i' > i$ that needs to be "pulled out" of the history component because it is about to finish its parsing of the contribution part of $w_{i'}$ – it needs to be "waked up before it's too late".

Formally, the construction runs by induction as follows: the first tuple of in the run is

$$\big(\emptyset, \emptyset, (X^1_1, q_*, \overline{X}^1_1), \big\{(X^i_1, q_*, \overline{X}^i_1) \mid i \geq 2\big\}\big)$$

(observe that in fact $X_1^i = \overline{X}_1^i = \emptyset$ for all $i \in [1 \dots]$ since $q_* \notin Q_u$ for any $u \in [1 \dots 2n]$).

Assume that we have built the run up to the $(k-1)$-th tuple. Denote the run built so far as θ_{k-1} and its last tuple as $\tau_{k-1} = (S_l, \alpha_l, \beta_l, T_l)_{l \in [1 \dots k-1]}$. For the induction, we also assume that

- either $\beta_{k-1} = (X_{j-1}^i, q_{j-1}^i, \overline{X}_{j-1}^i)$ with $k = j + i$
- or $\beta_{k-1} = \emptyset$ and then $\alpha_{k-1} = (X_{j-1}^p, q_{j-1}^p, \overline{X}_{j-1}^p)$ with $k = j + p$.

These properties hold for $k = 1$, hence the induction has indeed a base case.

We then have the following cases, triggering specific ways to extend the run θ_{k-1}:

1. If $\beta_{k-1} = (X_{j-1}^i, q_{j-1}^i, \overline{X}_{j-1}^i)$ and for all $i' > i$, $\Delta_j^{i'} \cap [n+1 \dots 2n] \subseteq (\Delta_j^{i'} \cap [1 \dots n]) + n$ (that is, we have not reached the end of some component $(w_{i'})_{u,n+u}$) then we extend the run with the tuple

$$\big((S_{k-1}, \alpha_{k-1}, \beta_{k-1}, T_{k-1}), a_{k-1-i}, (S_k, \alpha_k, \beta_k, T_k)\big)$$

in which

$$\begin{aligned}
S_k &= \big\{(X_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' < i-1\big\} \\
\alpha_k &= \begin{cases} \emptyset & \text{iff } i = 1 \\ (X_j^{i-1}, q_j^{i-1}, \overline{X}_j^{i-1}) & \text{iff } i \geq 2 \end{cases} \\
\beta_k &= (X_j^i, q_j^i, \overline{X}_j^i) \\
T_k &= \big\{(X_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' > i\big\}
\end{aligned}$$

2. If $\beta_{k-1} = (X_{j-1}^i, q_{j-1}^i, \overline{X}_{j-1}^i)$ and there exists some $i' > i$ for which $\big(\Delta_j^{i'} \cap [n+1 \dots 2n]\big) \setminus \big((\Delta_j^{i'} \cap [1 \dots n]) + n\big) \neq \emptyset$ then let

$$\overline{\imath} = \max\big\{i' \geq i \mid \big(\Delta_j^{i'} \cap [n+1 \dots 2n]\big) \setminus \big((\Delta_j^{i'} \cap [1 \dots n]) + n\big) \neq \emptyset\big\}$$

Observe that in this case the tuple $(X_j^{\overline{\imath}}, q_j^{\overline{\imath}}, \overline{X}_j^{\overline{\imath}})$ cannot belong to the history component since it would contradict the requirements 1.c or 2.c.iii, according to whether $\alpha_{k-1} = \emptyset$ or not.

We then append $\overline{\imath} - i + 2$ tuples as follows:

a) The first tuple to be appended is

$$\big((S_{k-1}, \alpha_{k-1}, \beta_{k-1}, T_{k-1}), \varepsilon, (S_k, \alpha_k, \beta_k, T_k)\big)$$

and has the following components:

$$\begin{aligned}
S_k &= \big\{(X_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' \leq i-1\big\} \\
\alpha_k &= (X_j^i, q_j^i, \overline{X}_j^i) \\
\beta_k &= \big(X_j^{i+1}, q_j^{i+1}, X_j^{i+1} \cup (\Delta_j^{i+1} \setminus [n+1 \dots 2n])\big) \\
T_k &= \big\{(X_j^{i'}, q_j^{i'}, X_j^{i'}) \mid i' > i+1\big\}
\end{aligned}$$

Observe that we do not change the third component in the tuples belonging to the prophecy component. Also observe that $\beta_k \neq \emptyset$ because $i < \overline{\imath} \leq p$.

b) For each $l \in [1 \ldots \overline{\iota} - i - 1]$, the l-th appended tuple is:

$$\big((S_{k+l-1}, \alpha_{k+l-1}, \beta_{k+l-1}, T_{k+l-1}), \varepsilon, (S_{k+l}, \alpha_{k+l}, \beta_{k+l}, T_{k+l})\big)$$

in which

$$\begin{aligned}
S_{k+l} &= \big\{(\overline{X}_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' < i + l - 1\big\} \\
\alpha_{k+l} &= \big(X_j^{i+l-1} \cup (\Delta_j^{i+l-1} \setminus [n+1 \ldots 2n]), q_j^{i+l-1}, \overline{X}_j^{i+l-1}\big) \\
\beta_{k+l} &= \big(X_j^{i+l}, q_j^{i+l}, X_j^{i+l} \cup (\Delta_j^{i+l} \setminus [n+1 \ldots 2n])\big) \\
T_{k+l} &= \big\{(X_j^{i'}, q_j^{i'}, X_j^{i'}) \mid i' > i + l\big\}
\end{aligned}$$

These are the transitions that "pull" the $(i+l)$-th tuple from the prophecy component into the right active component. This operation is accompanied by the modification of the index set of the $(i+l)$-th tuple, but the index sets of all tuples in the prophecy component are left unchanged.

c) For $l = \overline{\iota} - i$, the $\overline{\iota} - i$ appended tuple is:

$$\big((S_{k+\overline{\iota}-i-1}, \alpha_{k+\overline{\iota}-i-1}, \beta_{k+\overline{\iota}-i-1}, T_{k+\overline{\iota}-i-1}), a_{k-1}, (S_{k+\overline{\iota}-i}, \alpha_{k+\overline{\iota}-i}, \beta_{k+\overline{\iota}-i}, T_{k+\overline{\iota}-i})\big)$$

with

$$\begin{aligned}
S_{k+\overline{\iota}-i} &= \big\{(\overline{X}_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' < \overline{\iota} - 1\big\} \\
\alpha_{k+\overline{\iota}-i} &= \big(X_j^{\overline{\iota}} \cup \Delta_j^{\overline{\iota}} \setminus [n+1 \ldots 2n]), q_j^{\overline{\iota}}, \overline{X}_j^{\overline{\iota}}\big) \\
\beta_{k+\overline{\iota}-i} &= \begin{cases} \emptyset & \text{iff } \overline{\iota} = p \\ (X_j^{\overline{\iota}+1}, q_j^{\overline{\iota}+1}, \overline{X}_j^{\overline{\iota}+1}) & \text{iff } \overline{\iota} \leq p-1 \end{cases} \\
T_{k+\overline{\iota}-i} &= \big\{(X_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' > \overline{\iota} + 1\big\}
\end{aligned}$$

Hence, once the $\overline{\iota}$-th tuple is pulled out from the prophecy component, we also change the index sets of all tuples remaining in the prophecy component. This is possible since, by choice of $\overline{\iota}$, we will prove that all the tuples in $T_{k+\overline{\iota}-i}$ do not contradict requirements 1.c and 2.c.iii.

3. If $\beta_{k-1} = \emptyset$, which can only happen when $\alpha_{k-1} = (X_{j-1}^p, q_{j-1}^p, \overline{X}_{j-1}^p)$, we append to the run the following tuple:

$$\big((S_{k-1}, \alpha_{k-1}, \beta_{k-1}, T_{k-1}), a_{k-p-1}, (S_k, \alpha_k, \beta_k, T_k)\big)$$

in which

$$\begin{aligned}
S_k &= \big\{(X_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' < p\big\} \\
\alpha_k &= (X_j^p, q_j^p, \overline{X}_j^p) \\
\beta_k &= \emptyset \\
T_k &= \emptyset
\end{aligned}$$

Observe that, after case 2, we already get at stage $k+\overline{\imath}-i+1$ since we have appended $\overline{\imath}-i+1$ tuples to θ_{k-1}. We denote θ the run obtained after stage $k=m+p$.

It remains to prove that the appended tuples are indeed states from $Q_{\circledast}$. We will only prove the validity of requirements 2 (a, b, c.i, c.ii, c.iii, d.i, d.ii, d.iii) since the other requirements can be regarded as special cases of requirements 2(a ... d.iii). Consequently, we will only study the cases 1 and 2 in the construction of θ. Let us show first that the properties *(X1)-(X4)* imply that the newly appended tuples satisfy requirements 2.a, 2.b, 2.c.i, 2.c.ii, 2.d.i, 2.d.ii, respective of the case the tuple falls in.

In case 1, requirements 2.a and 2.b are restatements of property *(X2)*, while requirements 2.c.i, 2.c.ii, 2.d.i, and 2.d.ii are restatements of property *(X4)*.

For the case 2, subcase a, that is, for the tuple $(S_k,\alpha_k,\beta_k,T_k)$, requirement 2.a results directly from *(X2)*. On the other hand, we have that

$$\begin{aligned}\big(X_j^{i+1}\cup(\Delta_j^{i+1}\setminus[n+1\ldots 2n])\big)\cap[1\ldots n] &= (X_j^{i+1}\cap[1\ldots n])\cup(\Delta_j^{i+1}\cap[1\ldots n])\\ &= \overline{X}_j^{i+1}\cap[1\ldots n]\end{aligned}$$

hence requirement 2.b is implied by *(X2)*. Next, properties 2.c.i, 2.c.ii and 2.d.i are trivially implied by *(X4)*. Finally, due to *(X4)* again, we observe that for all $i'>i+1$,

$$\big(X_j^{i+1}\cup(\Delta_j^{i+1}\setminus[n+1\ldots 2n])\big)\cap[n+1\ldots 2n]\supseteq X_j^{i+1}\cap[n+1\ldots 2n]\supseteq X_j^{i'}$$

hence requirement 2.d.ii holds also for the case 2, subcase a.

For the case 2, subcase b, that is, for each tuple $(S_{k+l},\alpha_{k+l},\beta_{k+l},T_{k+l})$ with $l\in[1\ldots\overline{\imath}-i-1]$, the requirements 2.a and 2.b can be similarly shown to derive from to *(X2)*. For requirement 2.c.i observe that for all $i'<i+l-1$ we have by *(X4)*

$$\overline{X}_j^{i'}\supseteq\overline{X}_j^{i+l-1}\supseteq X_j^{i+l-1}\cup(\Delta_j^{i+l-1}\setminus[n+1\ldots 2n])$$

while requirement 2.c.ii is directly implied by *(X4)*. Then, requirement 2.d.i is a direct consequence of *(X4)* while for 2.d.ii we have that for all $i'>i+l$,

$$X_j^{i+l}\cup(\Delta_j^{i+l}\setminus[n+1\ldots 2n])\cap[1\ldots n]=\overline{X}_j^{i+l}\cap[1\ldots n]\supseteq\overline{X}_j^{i'}\supseteq X_j^{i'}$$

Finally, in case 2, subcase c, that is, for the tuple $(S_{k+\overline{\imath}-i},\alpha_{k+\overline{\imath}-i},\beta_{k+\overline{\imath}-i},T_{k+\overline{\imath}-i})$, the proof that all requirements 2.a, 2.b, 2.c.i, 2.c.ii, 2.d.i, and 2.d.ii hold is very similar to the other cases.

Consider now the requirement 2.d.iii. For the case 1, requirement 2.d.iii can be proved as follows: if we suppose that for all $i'>i$, $\Delta_{j-1}^{i'}\cap[n+1\ldots 2n]\subseteq(\Delta_{j-1}^{i'}\cap[1\ldots n])+n$ then we can show that for all $i'>i$,

$$\Delta_j^i\cap[n+1\ldots 2n]\supseteq(\Delta_j^{i'}\cap[1\ldots n])+n. \tag{7.24}$$

We can prove this inclusion as follows: for each $i'>i$, consider some index $u+n\in\big((\Delta_j^{i'}\cap[1\ldots n])+n\big)$. Identity 7.23 says that $(\Delta_j^{i'}\cap[1\ldots n])+n=\Delta_j^{i'-1}\cap[n+1\ldots 2n]$, hence $u+n\in$

$(\Delta_j^{i'-1} \cap [n+1 \dots 2n])$. On the other hand, the hypothesis of case 1 says that $\Delta_j^{i'-1} \cap [n+1 \dots 2n] \subseteq \Delta_j^i \cap [n+1 \dots 2n]$, and hence $u+n \in \Delta_j^i \cap [n+1 \dots 2n]$, which shows that our inclusion 7.24 holds.

For the case 2 subcase a, consider the tuple $(S_k, \alpha_k, \beta_k, T_k)$ which is the first to be appended to θ_{k-1}. The requirement 2.d.iii for this tuple is the following: for all $i' > i$,

$$\begin{aligned}(X_j^{i'} \setminus X_j^{i'}) \cap [n+1 \dots 2n] &\subseteq \big((X_j^{i'} \setminus X_j^{i'}) \cap [1 \dots n]\big) + n \\ &\subseteq \Big(\big(X_j^{i+1} \cup (\Delta_j^{i+1} \setminus [n+1 \dots 2n])\big) \setminus X_j^{i+1}\Big) \cap [n+1 \dots 2n]\end{aligned}$$

But this property holds trivially since all the three sets involved in this chain of inclusions are empty.

For case 2, subcase b, consider each tuple $(S_{k+l}, \alpha_{k+l}, \beta_{k+l}, T_{k+l})$ with $l \in [1 \dots \overline{\iota} - i + 1]$, But for the requirement 2.d.iii to hold for this tuple, we must have that for all $i' > i + l$,

$$\begin{aligned}(X_j^{i'} \setminus X_j^{i'}) \cap [n+1 \dots 2n] &\subseteq \big((X_j^{i'} \setminus X_j^{i'}) \cap [1 \dots n]\big) + n \\ &\subseteq \Big(\big(X_j^{i+l} \cup (\Delta_j^{i+l} \setminus [n+1 \dots 2n])\big) \setminus X_j^{i+l}\Big) \cap [n+1 \dots 2n]\end{aligned}$$

which again holds trivially since all sets are empty.

Finally, in case 2, subcase c, requirement 2.d.iii for tuple $(S_{k+\overline{\iota}-i}, \alpha_{k+\overline{\iota}-i}, \beta_{k+\overline{\iota}-i}, T_{k+\overline{\iota}-i})$ says that, for all $i' > \overline{\iota} + 1$,

$$\Delta_j^{i'} \cap [n+1 \dots 2n] \subseteq (\Delta_j^{i'} \cap [1 \dots n]) + n \subseteq \Delta_j^{\overline{\iota}+1} \cap [n+1 \dots 2n]$$

First observe that, by choice of $\overline{\iota}$, for all $i' > \overline{\iota}$ the first inclusion holds. We then only have to prove that the second inclusion holds too.

Suppose this does not hold for some $i' > \overline{\iota}+1$. We show that this would be in contradiction with the choice of $\overline{\iota}$: take some $u \in [1 \dots n]$ with $u+n \in \big((\Delta_j^{i'} \cap [1 \dots n]) + n\big) \setminus (\Delta_j^{\overline{\iota}+1} \cap [n+1 \dots 2n])$. Then $u+n \in \overline{X}_j^{i'}$, and by *(X3)*, $u+n \in \overline{X}_j^{\overline{\iota}+1}$. Since $u+n \notin \Delta_j^{\overline{\iota}+1}$ we must have $u+n \in X_j^{\overline{\iota}+1}$. Again by *(X3)*, we get that $u+n \in X_j^i$, hence $u+n \in \overline{X}_j^i$ and further $u+n \notin \Delta_j^i$. But then, by gathering all the information we obtained on $u+n$ we get:

$$u+n \in \big((\Delta_j^{i'} \cap [1 \dots n]) + n\big) \setminus (\Delta_j^i \cap [n+1 \dots 2n]) \tag{7.25}$$

This is in contradiction with the choice of $\overline{\iota}$, since the greatest integer in $[1 \dots p]$ which verifies property 7.25 is $\overline{\iota}$, and we have assumed $i' > \overline{\iota} + 1$.

Let us finally check requirement 2.c.iii. First, we observe that for the case 2, subcases b and c, requirement 2.c.iii holds trivially since in the respective chain of inclusions the two sets are empty. (The proof of this observation is similar to the proof that requirement 2.d.iii holds in cases 2, subcases b and c.)

For checking case 1 and case 2, subcase a, we will first show that the first part of the inclusion from requirement 2.c.iii holds, that is,

$$\text{for all } i' < i, (\Delta_j^{i'} \cap [1 \dots n]) + n \subseteq \Delta_j^{i'} \cap [n+1 \dots 2n]. \tag{7.26}$$

This property will be proved by contradiction, with essential use of the non-elasticity assumption:

Suppose that there exists some $i' < i$ and some $u_0 \in [1 \dots n]$ such that $u_0 \in \Delta_j^{i'} \cap [1 \dots n] \setminus (\Delta_j^{i'} \cap [n+1 \dots 2n]) - n$. The first observation to be made is that since i' is in the history component at $j-1$, there must exist a moment $j' < j$ at which i' had to be pulled out from the prophecy component because for some $i' < i_2$, we had that $\Delta_{j'}^{i_2} \cap [n+1 \dots 2n] \not\subseteq (\Delta_{j'}^{i_2} \cap [1 \dots n]) + n$. Pick up then some $v_0 \in [1 \dots n]$ such that $v_0 + n \in (\Delta_{j'}^{i_2} \cap [n+1 \dots 2n]) \setminus \big((\Delta_{j'}^{i_2} \cap [1 \dots n]) + n\big)$. Hence $v_0 + n \in \overline{X}_{j'}^{i_2}$, and since $\overline{X}_{j'}^{i_2} in \mathcal{X}$ we have that $v_0 \in \overline{X}_{j'}^{i_2}$, and further $v_0 \in X_{j'}^{i_2}$.

Let us further observe that $u_0 + n \notin X_j^{i'}$ and hence, by *(X3)* $u_0 + n \notin X_j^{i_2}$ Moreover, we have that $v_0 \in X_{j'}^{i_2}$ which by *(X3)* gives that $v_0 \in X_{j'}^{i'}$. This means that there exists $j_1 > j$ such that $u_0 + n \in \Delta_{j_1}^{i_2}$ and there exists $j_2 < j'$ such that $v_0 \in \Delta_{j_2}^{i'}$.

We then need the following property:

> (W) For each $1 \leq k < k' \leq p$ and each $l, l' \in [1 \dots m]$, suppose $u \in \Delta_l^k$ and $v + n \in \Delta_{l'}^{k'}$ for some $u, v \in [1 \dots n]$. Denote also a_t the label of the t-th transition in any of the runs ρ_k or $\rho_{k'}$. Then,
> - If $l = l'$ then $(w_k \odot \dots \odot w_{k'})_{u,v+n} = \varepsilon$.
> - If $l < l'$ then $(w_k \odot \dots \odot w_{k'})_{u,v+n} = a_l a_{l+1} \dots a_{l'-1}$.
> - If $l > l'$ then $(w_k \odot \dots \odot w_{k'})_{u,v+n} = a_{l'-1}^{-1} \dots a_{l+1}^{-1} a_l^{-1}$.

Proof (of property (W)*).* The property (W) can be proved by induction on $k' \geq k$: for $k = k'$ it holds straightforwardly since it says that the concatenation of the labels of the transitions in between the moment ρ_k passes through Q_u (i.e., l) and the moment it passes through Q_{v+n} (i.e., l') equals $w_{u,v+n}$.

Suppose then it holds for k', and we want to prove it for $k'+1$. Since $v + n \in \Delta_{l'}^{k'+1}$ it follows that there exists some $l_1 \leq l'$ such that $v \in \Delta_{l_1}^{k'+1}$. And further, by 7.23, that $v + n \in \Delta_{l_1}^{k'}$. We thence have 12 cases concerning the relative position of l_1, l and l'. We will only study three of them and discuss the similarities with the other cases.

- Suppose $l < l_1 < l'$. Then, by the induction hypothesis for k, k', l and l_1, we have $(w_k \odot \dots \odot w_{k'-1})_{u,v+n} = a_l a_{l+1} \dots a_{l_1-1}$. On the other hand, $(w_{k'})_{v,v+n} = a_{l_1} \dots a_{l'-1}$. Therefore,

$$\begin{aligned}(w_k \odot \dots \odot w_{k'+1})_{u,v+n} &= (w_k \odot \dots \odot w_{k'})_{u,v+n} \cdot (w_{k'+1})_{v,v+n} \\ &= a_l a_{l+1} \dots a_{l_1-1} \cdot a_{l_1} \dots a_{l'-1} \\ &= a_l a_{l+1} \dots a_{l'-1}\end{aligned}$$

  since the label of the t-th transition in any of the p runs is the same. Hence property (W) holds for $k'+1$. This proof works also for all the cases in which l_1 is in between l and l'.
- Suppose $l' < l_1 < l$. Then, by the induction hypothesis for k, k', l and l_1, we have $(w_k \odot \dots \odot w_{k'-1})_{u,v+n} = a_{l-1}^{-1} \dots a_{l_1}^{-1}$. On the other side, $(w_{k'})_{v,v+n} = a_{l_1-1}^{-1} \dots a_{l'}^{-1}$. Therefore,

$$\begin{aligned}(w_k \odot \ldots \odot w_{k'+1})_{u,v+n} &= (w_k \odot \ldots \odot w_{k'})_{u,v+n} \cdot (w_{k'+1})_{v,v+n} \\ &= a_{l-1}^{-1} \ldots a_{l_1}^{-1} \cdot a_{l_1-1} \ldots a_{l'-1} \\ &= a_{l-1}^{-1} \ldots a_{l'-1}\end{aligned}$$

hence property (W) holds for $k'+1$ also in this case. This proof works also for all the cases in which l_1 is in between l' and l.

- Suppose $l_1 < l < l'$. In this case we have, by the induction hypothesis, that $(w_k \odot \ldots \odot w_{k'-1})_{u,v+n} = a_{l-1}^{-1} \ldots a_{l_1}^{-1}$ and $(w_{k'})_{v,v+n} = a_{l_1} \ldots a_{l'-1}$. But

$$\begin{aligned}(w_k \odot \ldots \odot w_{k'})_{u,v+n} &= (w_k \odot \ldots \odot w_{k'-1})_{u,v+n} \cdot (w_{k'})_{v,v+n} \\ &= a_{l-1}^{-1} \ldots a_{l_1}^{-1} \cdot a_{l_1} \ldots a_{l'-1} \\ &= a_l a_{l+1} \ldots a_{l'-1}\end{aligned}$$

Hence property (W) holds for $k'+1$ in this last case. A similar proof can be produced in the case $l_1 < l' < l$. □

We may now particularize the property (W) as follows: we put $u := u_0$, $v := v_0$, $k := i'$, $k' := i_2$, $l := j_1$, $l' := j_2$ and get that

- $(w_{i'} \odot \ldots \odot w_{i_2})_{u,u+n} \neq \varepsilon$;
- $(w_{i'} \odot \ldots \odot w_{i_2})_{v,v+n} \neq \varepsilon$;
- $(w_{i'} \odot \ldots \odot w_{i_2})_{u,v+n}$ is an antiword;

facts which clearly contradict the non-elasticity assumption on $w_{i'} \odot \ldots \odot w_{i_2}$. Hence our assumption on the nonvalidity of Inclusion 7.26 is itself false.

Let us then observe that, based on the validity of the first half of requirement 2.c.iii for all $i' < i$ and on Identity 7.23, the last half of requirement 2.c.iii can be proved by induction as follows:

$$\begin{aligned}\Delta_j^{i'} \cap [n+1 \ldots 2n] &= (\Delta_j^{i'+1} \cap [1 \ldots n]) + n && \text{(by Identity 7.23)} \\ &\subseteq \Delta_j^{i'+1} \cap [n+1 \ldots 2n] && \text{(by Inclusion 7.26)} \\ &\vdots \\ &\subseteq \Delta_j^{i-1} \cap [n+1 \ldots 2n] \\ &= (\Delta_j^{i} \cap [1 \ldots n]) + n && \text{(by Identity 7.23)}\end{aligned}$$

It follows that $\theta = \theta_{m+p}$ is indeed a run in $\mathcal{D}$. We need now to show it is an accepting run, and then associate a sequence of indices $(t_u)_{u \in [1 \ldots 2n]}$ such that the t_u-th state in this run is in U_u and such that, for $u \in [1 \ldots n]$, the passage through U_u of θ be synchronous with the passage of ρ_1 through Q_u and the passage of θ through U_{n+u} be synchronous with the passage of ρ_p through Q_{n+u}.

Remind that l_u^1 represents the moment ρ_1 passes through Q_u, i.e., $u \in \Delta_{l_u^1}^1$. Consider then the index $k \in [1 \ldots m+p]$ which denotes the moment in the run θ which corresponds to l_u^1. That is, $k = l_u^1 + i$ for some $i \in [0 \ldots p]$, and further:

- If $i = 0$ then $\alpha_k = \emptyset$ and $\beta_k = (X^1_{l^1_u}, q^1_{l^1_u}, Z)$. Note that we may have either $Z = \overline{X}^1_{l^1_u}$ or $Z = X^1_{l^1_u} \cup (\Delta^1_{l^1_u} \setminus [n+1 \dots 2n])$, according to whether the $(l^1_u - 1)$-th transition in θ_{m+p} falls in case 1 or case 2, subcase a from the construction of θ.
- Otherwise, $\alpha_k = (X^{i-1}_{l^1_u}, q^{i-1}_{l^1_u}, \overline{X}^{i-1}_{l^1_u})$ and hence $(X^1_{l^1_u}, q^1_{l^1_u}, \overline{X}^1_{l^1_u}) \in S_k \cup \{\alpha_k\}$.

In the first case, it is clear that $(S_k, \alpha_k, \beta_k, T_k) = (\emptyset, \emptyset, (X^1_{l^1_u}, q^1_{l^1_u}, Z), T_k) \in U_u$ because by construction we have $u \in Z \setminus X^1_{l^1_u}$, in any of the cases Z falls in.

In the second case, if $i = 2$ then $\alpha_k = (X^1_{l^1_u}, q^1_{l^1_u}, \overline{X}^1_{l^1_u})$, $S_k = \emptyset$ and we get again that $(S_k, \alpha_k, \beta_k, T_k) \in U_u$. For $i \geq 3$ we must have that $(X^1_{l^1_u}, q^1_{l^1_u}, \overline{X}^1_{l^1_u}) \in S_k$. Let us observe that requirement 2.c.iii in the construction of $Q_\circledast$ says that $\Delta^1_{l^1_u} \cap [1 \dots n] \subseteq \Delta^{i-1}_{l^1_u}$, and hence $u \in \Delta^{i-1}_{l^1_u}$.

Suppose then that $(S_k, \alpha_k, \beta_k, T_k) \notin U_u$. This implies that there exists some $i' < i-1, i' > 1$ such that $u \notin \Delta^{i'}_{l^1_u}$. Then there must exist some $i_1 < i'$ such that $u \in \Delta^{i_1}_{l^1_u}$ and $u + n \notin \Delta^{i_1}_{l^1_u}$. But this is in contradiction with the requirement 2.c.iii, first inclusion. Hence the assumption is false, that is, $(S_k, \alpha_k, \beta_k, T_k) \in U_u$.

We may therefore define $t_u = l^1_u + i$. We also define $t_{n+u} = l^p_{n+u} + i$, where $l^p_{n+u} + i$ is the index in the run θ_{m+p} which corresponds to the $(l^p_{n+u} - 1)$-th transition in each of the runs ρ_i, $i \in [1 \dots p]$. Similarly to the proof for t_u, we may get that $(S_{t_{n+u}}, \alpha_{t_{n+u}}, \beta_{t_{n+u}}, T_{t_{n+u}}) \in U_{n+u}$.

It remains just to observe that, by property (W), for each $u, v \in [1 \dots n]$, the word or antiword that labels the transitions in between the t_u-th state and the t_{n+v}-th state equals the word or antiword that labels the transitions in between the l^1_u-th state and the l^p_{n+v}-th state of any of the runs ρ_i, that is, equals $(w_1 \odot \dots \odot w_p)_{u,n+v}$. Similarly, by concatenating the labels of the piece of run from θ_{m+p} that lies in between the t_u-th state and the t_v-th state (eventually in reversed order, if $t_u > t_v$) we get $(w_1)_{uv}$, and similarly for t_{u+n}, t_{v+n} and $(w_p)_{u+n,v+n}$. Hence, the run θ_{m+p} accepts indeed $w_1 \odot \dots \odot w_p$.

This ends our proof of the second claim, and, consequently, the proof of Theorem 7.4.3. □

8. Representing timing information with n-words

Up to this moment we have investigated only the possibility to use $2n$-automata for representing the discrete information in $2n$-signal regular expressions. In this chapter we investigate the possibility of representing also the *timing* information in $2n$-signal regular expressions. By the timing information we mean the set of tuples representing the duration of each $2n$-signal in the semantics of a $2n$-signal regular expression $E \in \mathsf{RegSig}_{2n}$, that is, the set

$$\big\{\ell(\sigma) \mid \sigma \in \|E\|\big\}$$

Here $\ell(\sigma)$ is the extension of the length morphism to n-signals, that is, a n-signal over a one-letter alphabet $\{a\}$ with $(\ell(\sigma))_{ij} = a^{\ell(\sigma_{ij})}$. Or, in other words, a $n \times n$ matrix $\alpha \in \mathcal{M}at_n(\mathbb{R})$ which satisfies the triangle identity $\alpha_{ij} + \alpha_{jk} = \alpha_{ik}$.

Let us note that, for the $2n$-signal-semantics of timed automata, our aim translates to the construction of the *reachability relation* on clocks, that is, the dependence between clock values when starting in initial states and the clock values with which final states may be reached.

This problem is nothing else but the problem of representing *timing constraints* over a set of real variables (or clocks). *Conjunctive* timing constraints are usually represented as *difference bound matrices* (DBMs, see [Bel57, Yov98]). These can be thought as $n \times n$ matrices over intervals (nonnecessarily positive), $D \in \mathsf{Mat}_{n\times n}(\mathbb{Z}Int)$ such that $D_{ij} = -D_{ji}$.

The idea is that each variable is associated with an index in the index set of the matrix, and each difference $x_i - x_j \in I$ puts the interval I in the (j, i)-component (note that the indices have swapped their places). To represent single-variable constraints like $x \in [3, 4]$, a special variable x_0 is appended, whose value is considered always 0, such that constraints like $x \in [3, 4]$ may be written as $x - x_0 \in [3, 4]$.

For example, the two-variable constraint $x \in [3, 4] \wedge y \in [1, 2] \wedge x - y = 2$ is represented by the following 3×3 matrix over intervals:

$$D = \begin{pmatrix} 0 & [3,4] & [1,2] \\ [-4,-3] & 0 & -2 \\ [-2,-1] & 2 & 0 \end{pmatrix} \tag{8.1}$$

Actually, to avoid redundant representation of the same interval, the important information is kept as follows: if $D_{ij} =]a, b[$, the (i, j)-th component will be $(b, \text{`}<\text{'})$ while the $(j, i)$-th component will be $(-a, \text{`}<\text{'})$. In other words, instead of the matrix of intervals D one keeps a matrix P of pairs (α, μ) where α is a real number and μ is a relation symbol $\mu \in \{\text{`}<\text{'}, \text{`}\leq\text{'}, \text{`}=\text{'}\}$, such that

$$P_{ij} = (\sup D_{ij}, \mu) \text{ where } \mu = \begin{cases} \text{`}=\text{'} & \text{iff } D_{ij} \text{ is a point interval} \\ \text{`}\leq\text{'} & \text{iff } \max D_{ij} \text{ exists and } \max D_{ij} \neq \inf D_{ij} \\ \text{`}<\text{'} & \text{otherwise} \end{cases}$$

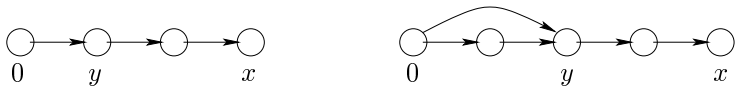
We will however utilize the "redundant" matrix of intervals D in order not to complicate certain proofs.

If we think of the set of interpretations that validate the constraint $x \in [3, 4] \wedge y \in [1, 2] \wedge x - y = 2$, then this set has more than one representation. The reason is that the first constraint can be deduced from the other two by arithmetic manipulation. However the representation given in 8.1 can be thought as the "canonical" one, since it is "closed" under these arithmetical manipulations – even more, it is "minimal", that is, no other representation can be found in which some of the intervals are smaller than the ones in D.

As already said, a DBM can only represent *conjunctions of atomic constraints*, therefore general atomic constraints (i.e. containing disjunction also) require using sets of DBMs. Our aim is to show that n-automata over *one-letter alphabets* can be used also for representing *arbitrary* clock constraints. We will start by showing, as a corollary of the previous chapter, how to represent timing constraints *over the discrete time domain* $\mathbb{Z}$:

The constraint $x = 3$ can be represented by a finite automaton (i.e. a 2-automaton) with four states and three transitions in a chain. The idea is that the constraint $x = 3$ is satisfied by the set of integers $\{3\}$, which is a regular language over the one-symbol set $\{1\}$ – simply because $3 = 1 + 1 + 1$.

To represent a two-clock constraint $x = 3 \wedge y = 1$ one might transform the above 2-automaton into the 3-automaton depicted in Figure 8.1 at (a). What we have done is to identify, during the run that accepts 3, a point in which $y = 1$ is satisfied. Note that we have an implicit subconstraint[1] $x - y = 2$, which is represented by the two arrows in between the state labeled y and the state labeled x. A "nonpoint" constraint, $x \in [3, 4] \wedge y \in [1, 2] \wedge x - y = 2$, is depicted at (b) (remind that we work with discrete clocks for the moment):



(a) (b)

Fig. 8.1. Two n-automata representations of timing constraints over a discrete domain: (a): $x = 3 \wedge y = 1$ and (b): $x \in [3, 4] \wedge y \in [1, 2] \wedge x - y = 2$.

Of course, to actually have 3-automata we would put Q_1 as the state labeled 0, Q_2 as the state labeled x and Q_3 as the state labeled y.

[1] Which would result after bringing the constraint to the "normal form" [Yov98].

A more complicated constraint is depicted in Figure8.2.

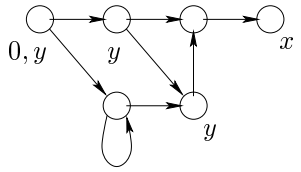


Fig. 8.2. An n-automata representation of the following timing constraint: $\big(y \in [2, \infty[\wedge x - y = 2\big) \vee \big(y \in \{0\} \wedge x \in [4, \infty[\big) \vee \big(y \in [0, 2] \wedge x = 4\big) \vee \big(y \in [0, 1] \wedge x = 3\big)$.

Of course, our approach makes constraint representation more sensible to the numbers used in the constraints than other representations (e.g., unions of DBMs). But observe that the constraint in Figure 8.2 uses only 5 states, instead of the 4 DBMs necessary. Also the *clock difference diagrams* approach [LWYP99] does not provide a better representation for this constraint, since the intervals used in each atomic constraint are distinct. Hence the n-automata representation of clock constraints might be better in some cases.

So far, so good with the intuition about discrete timing, but how to export it to the *continuous timing*? Here we will get aid from the notion of *clock regions* [AD94]. A *region* is a special kind of DBM, in which each interval is either a point or an open unit length interval. For example,

$$R = \begin{pmatrix} 0 &]2,3[&]1,2[\\ ]-3,-2[& 0 &]-1,0[\\ ]-2,-1[&]0,1[& 0 \end{pmatrix} \tag{8.2}$$

Of course, this is not exactly the classical definition of a region: they were originally defined as sets of points $x \in \mathbb{R}^n$ which have the same integral parts and the same ordering between the fractional parts [AD94]. For our example, the graphical representation of the region R defined in Identity 8.2 (considering that this region represents in fact the constraint $x \in]2,3[\wedge y \in]1,2[\wedge x - y \in]0,1[$) is depicted in Figure 8.3:

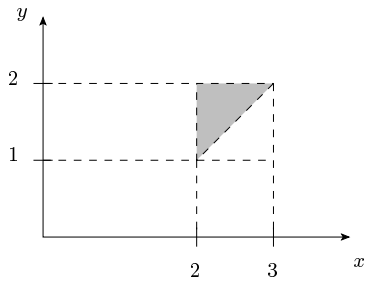


Fig. 8.3. The graphical representation of the region defined in 8.2 is the interior of the shadowed triangle.

It is clear that any DBM whose intervals have integer bounds can be decomposed into a (finite or infinite) set of regions.

Now the key idea is to observe that the 3-regword of all endpoints of the intervals of a region *has a nonempty semantics*: for example, our region R described in Identity 8.2 defines the following 3-regword whose components are the bounds of the intervals of R:

$$\begin{pmatrix} 0 & \{2,3\} & \{1,2\} \\ \{-3,-2\} & 0 & \{-1,0\} \\ \{-2,-1\} & \{-1,0\} & 0 \end{pmatrix} \tag{8.3}$$

Here the sets represent regular expressions over the symbol set $\{1\}$: for example[2], $\{2,3\} = 1 + 1 + 1 \cup 1 + 1$. In other words, we use here the *unary* encoding of integers.

Then, if from the upper triangular part we keep the upper bounds and from the lower triangular part we keep the lower bounds we get the following 3-word:

$$w = \begin{pmatrix} 0 & 3 & 2 \\ -3 & 0 & -1 \\ -2 & 1 & 0 \end{pmatrix}$$

Note that this 3-word is (represented by the) lower vertex of the shadowed triangle in Figure 8.3.

We then just have to add the information about what kind of bound is each component in w. This information is given by a 3×3 matrix M whose entries are relation symbols from the set $\{`<`, `=`, `>`\}$. Hence, we put $M_{ij} = `<`$ iff w_{ij} is the supremum of R_{ij} but does not belong to R_{ij}, we put $M_{ij} = `=`$ if $w_{ij} = \max R_{ij}$ and $M_{ij} = `>`$ iff $w_{ij} = \inf R_{ij}$ and $w_{ij} \notin R_{ij}$.

For our example, the following pair represents the region in 8.2:

$$\left(\begin{pmatrix} 0 & 3 & 2 \\ -3 & 0 & -1 \\ -2 & 1 & 0 \end{pmatrix} , \begin{pmatrix} `=` & `<` & `<` \\ `>` & `=` & `>` \\ `>` & `<` & `=` \end{pmatrix} \right)$$

This representation of the region in 8.2 is not unique: the following pair is also a representation of it:

$$\left(\begin{pmatrix} 0 & 2 & 2 \\ -2 & 0 & 0 \\ -2 & 0 & 0 \end{pmatrix} , \begin{pmatrix} `=` & `>` & `<` \\ `<` & `=` & `<` \\ `>` & `>` & `=` \end{pmatrix} \right)$$

Note that the 3-word in it is the upper left vertex of the triangle in Figure 8.3.

It is then clear that not all the matrices having lower or upper bounds from the intervals involved in 8.2 are 3-words, since some do not verify the triangle identity: they are 2^6 such matrices but only 3 vertices of the region!

[2] We have preferred this set notation instead of the regular expression notation due to the ambiguous overloading of summation it would imply.

There is yet one more thing to observe: the matrices of relation symbols have themselves a sort of "triangle" property. due to the fact that they have to represent correct regions. We will see later the exact formulation of this property, but let's see an example of an incorrect matrix:

$$M = \begin{pmatrix} `=` & `>` & `<` \\ `<` & `=` & `>` \\ `>` & `<` & `=` \end{pmatrix}$$

Intuitively, M is incorrect because the cycle $(M_{12}, M_{23}, M_{31}) = (`>`, `>`, `>`)$ is inconsistent with the component M_{11}: the cycle requires that M_{11} be $`>`$, which is unimaginable for a diagonal component which must always be zero!

Once we are convinced regions may be represented this way, we only have to think that n-automata can be adapted to accept pairs consisting of a n-word over a one-letter alphabet and a matrix of relational symbols. There are two ways: either put the relational matrices into states, or put them into transitions. The two ways are completely interchangeable, as are state-labeled or transition-labeled finite automata. Then what we need to assure is that in an accepting run all states or all transitions are labeled with the same relational matrix.

Yet we are not through with the problems: concatenation is a clear operation on $2n$-words, but how do we generalize it to DBMs/regions/pairs like the above? In fact, we need to define a concatenation operation on regions, concatenation that would be compositional w.r.t. the "semantic" concatenation on $2n$-signals.

If we regard the problem from the logical point of view, when we want to concatenate two DBMs which represent two constraints C_1, C_2 over $2n$ variables, what we need is to take their conjunction $C_1 \wedge C_2$, then to identify the last n variables of C_1 with the first n variables of C_2, and finally to project the result over these variables "in the middle": denoting $x_1, \ldots, x_{2n}$ the variables in C_1 and $y_1, \ldots, y_{2n}$ the variables in C_2, we need

$$C_1 \odot C_2 = \exists x_{n+1} \ldots \exists x_{2n} \Big(C_1 \wedge C_2[x_{n+1}/y_1, \ldots, x_{2n}/y_n] \wedge \bigwedge_{i=1}^{n} x_{n+i} = y_i \Big)$$

in which the notation $C[x/y]$ stands for the syntactic replacement of variable y with x in C.

But if we proceed by pure arithmetic tools to compute the concatenation of two regions we will find out that *it might not be a region* but a more general DBM: even for 2-dimensional DBMs, that is, constraints over a single clock, we have

$$\big(x \in]1, 2[\big) \odot \big(x \in]2, 3[\big) = \big(x \in]3, 5[\big) \tag{8.4}$$

And here is an example concerning 4-regions:

$$\begin{pmatrix} 0 &]0,1[&]0,1[&]1,2[\\ ]-1,0[& 0 &]0,1[&]0,1[\\ ]-1,0[&]-1,0[& 0 &]0,1[\\ ]-2,-1[&]-1,0[&]-1,0[& 0 \end{pmatrix} \odot \begin{pmatrix} 0 &]0,1[&]0,1[&]1,2[\\ ]-1,0[& 0 &]0,1[&]0,1[\\ ]-1,0[&]-1,0[& 0 &]0,1[\\ ]-2,-1[&]-1,0[&]-1,0[& 0 \end{pmatrix} =$$

$$= \begin{pmatrix} 0 &]0,1[&]0,2[&]1,3[\\ ]-1,0[& 0 &]0,2[&]0,2[\\ ]-2,0[&]-2,0[& 0 &]0,1[\\ ]-3,-1[&]-2,0[&]-1,0[& 0 \end{pmatrix} \quad (8.5)$$

The issue from this is to *decompose* the result into regions. That is, we will define the concatenation of two regions as a *set* of regions. For the simple example described by Formula 8.4 we put

$$x \in]1,2[\odot x \in]2,3[= \big(x \in]3,4[\big) \vee \big(x = 4\big) \vee \big(x \in]4,5[\big)$$

The advantage is that, contrary to constraints, which, after each conjunction, need to be brought to normal forms, region concatenation gives directly normal forms.

Since we represent regions by pairs (w, M) consisting of a $2n$-word over $\{1\}$ and a $2n$-relation M, we may wonder how this concatenation can be implemented over such items, and also why it gives sets of regions rather than mere regions, since $2n$-word concatenation is not a set-based partial operation. The answer relies on the need of a concatenation operation on $2n$-relations, which itself returns *sets* of $2n$-relations. For our simple example 8.4 of 2-regions, we have

$$(2, \text{`}<\text{'}) \odot (2, \text{`}>\text{'}) = \big\{(4, \text{`}<\text{'}), (4, \text{`}=\text{'}), (4, \text{`}>\text{'})\big\}$$

because

$$(\text{`}<\text{'}) \odot (\text{`}>\text{'}) = \big\{(\text{`}<\text{'}), (\text{`}=\text{'}), (\text{`}>\text{'})\big\}$$

We might also observe that some pairs representing regions may fail to concatenate, even when the represented regions concatenate: for example, the following pairs cannot concatenate because the 4-words in it cannot, though they represent the concatenation of the 4-regions in 8.5:

$$\left(\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & 0 \end{pmatrix}, \begin{pmatrix} \text{`}=\text{'} & \text{`}>\text{'} & \text{`}>\text{'} & \text{`}>\text{'} \\ \text{`}<\text{'} & \text{`}=\text{'} & \text{`}>\text{'} & \text{`}<\text{'} \\ \text{`}<\text{'} & \text{`}<\text{'} & \text{`}=\text{'} & \text{`}<\text{'} \\ \text{`}<\text{'} & \text{`}>\text{'} & \text{`}>\text{'} & \text{`}=\text{'} \end{pmatrix}\right) \odot$$

$$\odot \left(\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & 0 \end{pmatrix}, \begin{pmatrix} \text{`}=\text{'} & \text{`}<\text{'} & \text{`}<\text{'} & \text{`}<\text{'} \\ \text{`}>\text{'} & \text{`}=\text{'} & \text{`}<\text{'} & \text{`}>\text{'} \\ \text{`}>\text{'} & \text{`}>\text{'} & \text{`}=\text{'} & \text{`}>\text{'} \\ \text{`}>\text{'} & \text{`}<\text{'} & \text{`}<\text{'} & \text{`}=\text{'} \end{pmatrix}\right) = \uparrow$$

But this does not mean that no pair of representations of the 4-regions from Identity 8.5 can be concatenated: we just have to be more careful when choosing the representations. For example, the two representations in Identity 8.6 below can be concatenated, since the lower right corners of the first representation equal the upper left corners of the second representation. The result is obtained by first concatenating the 4-words in each representation, then concatenating the two 4-relation matrices. We present the result as a cartesian product between the resulting 4-word and a set of 4-relation matrices, just in order to save space.

$$\left(\left(\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ -1 & -1 & -1 & 0 \end{pmatrix}, \begin{pmatrix} `=` & `>` & `>` & `>` \\ `<` & `=` & `>` & `<` \\ `<` & `<` & `=` & `<` \\ `<` & `>` & `>` & `=` \end{pmatrix}\right) \odot \right.$$
$$\left. \odot \left(\begin{pmatrix} 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} `=` & `<` & `<` & `>` \\ `>` & `=` & `>` & `>` \\ `>` & `>` & `=` & `>` \\ `<` & `<` & `<` & `=` \end{pmatrix}\right)\right) =$$
$$= \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ -1 & -1 & 0 & 0 \\ -1 & -1 & 0 & 0 \end{pmatrix} \times \left\{ \begin{pmatrix} `=` & `>` & `>` & `>` \\ `<` & `=` & `<` & `>` \\ `<` & `>` & `=` & `>` \\ `<` & `<` & `<` & `=` \end{pmatrix}, \right.$$
$$\left. \begin{pmatrix} `=` & `>` & `>` & `>` \\ `<` & `=` & `=` & `>` \\ `<` & `=` & `=` & `>` \\ `<` & `<` & `<` & `=` \end{pmatrix}, \begin{pmatrix} `=` & `>` & `>` & `>` \\ `<` & `=` & `>` & `>` \\ `<` & `<` & `=` & `>` \\ `<` & `<` & `<` & `=` \end{pmatrix} \right\} \quad (8.6)$$

The $(1,3)$ component of the resulting 4-relation matrices is fixed, because it must be consistent with the fact that the $(1,4)$ component in the left operand is a ‘ $>$ ‘ and the $(2,3)$ component in the second operand is a ‘ $>$ ‘ too. Similarly, the components $(1,4)$ and $(2,4)$ are uniquely generated. However, the $(2,3)$ component is not uniquely generated, and this is why the above concatenation produces three representations.

Clearly, the above result is not the representation of all regions that are included in the resulting DBM as given in Identity 8.5. But if we try all the combinations of representations of the two factor regions and join together all the results, we will obtain the expected decomposition of the DBM into regions.

By summarizing, in order to get a correct representation of the concatenation of regions by pairs (w, M), we need to work with *sets* of such pairs, and to assure that, in each such set, together with a pair (w, M) representing a region R, all the pairs that represent this region are contained.

We will develop in this chapter the theory of DBMs, regions and n-word representations for regions. Part of the chapter is a restatement of some well-known properties concerning DBM normalization and/or constraint propagation [Gau99, DMP91, vH89]. But the bulk of it is new, and

concerns the restatement of the properties of concatenation and star on $2n$-signals and $2n$-words, as given in Chapter 6, for regions and $2n$-word representations of regions.

A permanent concern in this chapter is again the compositionality of the projection, juxtaposition, resp. concatenation operations. This is quite normal, since we try to define syntactic operations on representations of sets of $2n$-signals, and we already know from Chapter 6 that, for $2n$-regminoes, such compositional operations are not possible (excepting juxtaposition). The key property that makes projection compositional is *convexity* of intervals.

This chapter runs as follows: the first section resumes some well-known properties concerning DBM normalization. In the second section we introduce our concatenation operation on regions and prove its compositionality. The third section serves for introducing the n-word representations for regions and for defining concatenation on them. In the fourth section we generalize n-automata to a class that works on n-word representations and prove that this class enjoys the same properties as n-automata. Most notably, we show that the non-elastic star closure theorem also holds for automata that work on non-elastic $2n$-word representations.

Let us note that n-automata representing clock constraints are different from region automata in the sense of Alur and Dill [AD94]: in n-automata, each region is represented by a *run*, while in region automata, each region is a state.

8.1 Difference bound matrices

Traditionally, n-regsignals over a one-letter alphabet $R \in \mathcal{RS}ig_n(\{a\})$ bearing the property

$$R_{ij} \in \mathbb{Z}Int \text{ for all } i,j \in [1 \ldots n]$$

are called **difference bound matrices**, or n-DBMs. By generalizing, we call any n-regsignal over a one-letter alphabet as an **extended difference bound matrix**, or n-EDBM. Observe that an EDBM is in fact a matrix whose components are *regular expressions over intervals*, in the sense used in Chapter 3. For example, the following matrix is a 2-EDBM and not a 2-DBM:

$$A = \begin{pmatrix} 0 & \{3\}^* \\ \{-3\}^* & 0 \end{pmatrix}$$

When speaking of regular expressions over intervals, we have in mind the theory developed in Section 3.2. Which, of course, needs to be extended over the whole Kleene algebra of intervals with integer bounds $\mathcal{K}(\mathbb{Z}Int)$ (in Section 3.2 we have studied only intervals with *natural* bounds).

The class of n-DBMs is denoted $\mathcal{D}\mathsf{bm}_n$ while the class of n-EDBMs is denoted $\mathcal{E}\mathsf{dbm}_n$.

The **semantics** of an n-EDBM $D \in \mathcal{E}\mathsf{dbm}_n$ is formally defined as follows:

$$\|D\| = \left\{\sigma \in \mathsf{Sig}_n(\{a\}) \mid \text{ for all } i,j \in [1 \ldots n] \text{ there exists } \alpha \in D_{ij} \text{ such that } \sigma_{ij} = a^\alpha\right\}$$

We present, in this section, several properties concerning DBMs, most of them well-known in the domain of $(\max, +)$-algebras [Gau99, GP97, Gau92]:

Proposition 8.1.1 ([Gau99]). *Given an n-DBM $D \in \mathcal{D}\mathsf{bm}_n$, $\|D\| \neq \emptyset$ if and only if for each cycle $(i_j)_{j\in[1\ldots k+1]}$ with $i_j \in [1\ldots n]$ and $i_{k+1} = i_1$, we have that*

$$0 \in \sum_{j=1}^{k} D_{i_j i_{j+1}} \tag{8.7}$$

Proof. We may observe first that, given an n-signal $\sigma \in \mathsf{Sig}_n(\{a\})$, for each cycle $(i_j)_{j\in[1\ldots k+1]}$ with $i_j \in [1\ldots n]$ and $i_{k+1} = i_1$, $\sum_{j=1}^{k} \sigma_{i_j i_{j+1}} = \sigma_\epsilon = a^0$. This property follows by induction on k from the triangle identity 6.1. We then only have to observe that for each $\sigma \in \|D\|$, $\sum_{j=1}^{k} \sigma_{i_j i_{j+1}} \in \sum_{j=1}^{k} D_{i_j i_{j+1}}$. □

Definition 8.1.2. *An n-DBM $D \in \mathcal{D}\mathsf{bm}_n$ is said to be **in normal form** iff the following two properties hold:*

1. *For each $i \in [1\ldots k]$, $D_{ii} = \{0\}$.*
2. *For each $i, j, k \in [1\ldots k]$,*

$$D_{ik} \subseteq D_{ij} + D_{jk} \tag{8.8}$$

We will refer to the property 8.8 as the **triangle inclusion**, by similarity to the triangle identity for n-dominoes. The set of n-DBMs in normal form is denoted $\mathcal{D}\mathsf{nf}_n$.

Remark 8.1.3. The triangle inclusion implies that a DBM in normal form $D \in \mathcal{D}\mathsf{nf}_n$ is antisymmetric, that is, it has the property that $D_{ij} = -D_{ji}$ for all $i, j \in [1\ldots n]$.

Proposition 8.1.4. *Any n-DBM in normal form has a nonempty semantics.*

Proof. We first check the hypotheses of Proposition 8.1.1 for cycles of length 3. To this end, take three indices $i, j, k \in [1\ldots n]$. Two cases occur:

1. D_{ik} is a point interval, $D_{ik} = \{\alpha\}$ for some $\alpha \in \mathbb{Z}$. Then

   $$D_{ij} + D_{jk} + D_{ki} \supseteq D_{ik} + D_{ki} = \{\alpha\} + \{-\alpha\} = \{0\}$$

   hence Proposition 8.1.1 holds.
2. D_{ik} has a nonempty interior. Denote then $\alpha = \inf D_{ik}$ and $\beta = \sup D_{ik}$. Clearly $\alpha < \beta$. Then

   $$D_{ij} + D_{jk} + D_{ki} \supseteq D_{ik} + D_{ki} \supseteq]\alpha, \beta[+]-\beta, -\alpha[=]\alpha-\beta, \beta-\alpha[\ni 0$$

   due to the property $\alpha < \beta$.

On the other hand, for any cycle $(i_j)_{j\in[1\ldots k]}$ with $i_{k+1} = i_1$, the triangle inclusion 8.8 can be repeatedly used to prove that

$$\sum_{j=1}^{k} D_{i_j i_{j+1}} \supseteq D_{i_1 i_3} + \sum_{j=3}^{k} D_{i_j i_{j+1}} \supseteq \ldots \supseteq D_{i_1 i_{k-1}} + D_{i_{k-1} i_k} + D_{i_k i_1}$$

But we already know that $0 \in D_{i_1 i_{k-1}} + D_{i_{k-1} i_k} + D_{i_k i_1}$ from the first part of the proof. Therefore, this chain of inclusions implies that property 8.7 must also hold for the whole cycle $(i_j)_{j\in[1\ldots k]}$. □

Observe that, in the above proof, we have used the fact that D_{ij} is an interval. Hence the property does not hold for EDBMs. In fact, the example 6.25 from Chapter 6, page 99 is an example of an EDBM which is in normal form, but has an empty semantics:

$$R = \begin{pmatrix} 0 & \{1,2\} & \{2,5\} & \{6,8\} \\ \{-2,-1\} & 0 & \{1,3\} & \{4,7\} \\ \{-5,-2\} & \{-3,-1\} & 0 & \{3,4\} \\ \{-8,-6\} & \{-7,-4\} & \{-4,-3\} & 0 \end{pmatrix}$$

As a corollary, an n-DBM is equivalent to an n-DBM in normal form (is *normalizable*) iff its semantics is nonempty.

The following result shows that, unlike the case of n-regsignals, projection is compositional for DBMs in normal form. Remind that projection of an n-regsignal $R \in \mathcal{RS}ig_n(\Sigma)$ onto some set $X \subseteq [1\ldots n]$ was defined in Chapter 6, Definition 6.4.3 as the matrix resulting after deleting the rows and columns of R whose indices are not in X.

Proposition 8.1.5. *For each n-DBM in normal form D and $X \subseteq [1\ldots n]$, $D\big|_X$ is an n-DBM in normal form too and $\|D\big|_X\| = \|D\|\big|_X$.*

Proof. The first part of the property is straightforward. We prove the second part by induction on the size of $n - card(X)$, and the proof of the induction step relies on interval convexity.

Take some $(n+1)$-DBM in normal form $D \in \mathcal{D}\mathsf{nf}_n$ and consider its projection $D\big|_{[1\ldots n]}$. Take further some n-signal $\sigma \in \|D\big|_{[1\ldots n]}\|$. We will prove that this signal can be (perhaps not uniquely) extended to a $(n+1)$-signal $\sigma' \in \|D\|$.

Fix some real $\alpha_1 \in \mathbb{R}$ and denote $\alpha_i = \alpha_1 + \sigma_{1i}$, for all $i \in [2\ldots n]$. Observe then that

$$\alpha_j - \alpha_i = \sigma_{1j} - \sigma_{1i} = \sigma_{1j} + \sigma_{i1} = \sigma_{ij} = \sigma'_{ij}$$

by straightforward applications of the triangle identity 6.1.

Let us observe that, if we find a real α_{n+1} such that

$$\alpha_{n+1} - \alpha_i \in D_{i,n+1} \text{ for all } i \in [1\ldots n] \tag{8.9}$$

then we are done, because the matrix σ' defined by

$$\sigma'_{ij} = \begin{cases} \sigma_{ij} & \text{iff } i,j \in [1 \ldots n] \\ \alpha_{n+1} - \alpha_i & \text{iff } j = n+1, i \in [1 \ldots n+1] \\ \alpha_j - \alpha_{n+1} & \text{iff } i = n+1, j \in [1 \ldots n+1] \end{cases}$$

would be an $(n+1)$-signal, as the triangle identity 6.1 can be easily checked on σ', e.g.:

$$\sigma'_{i,n+1} + \sigma'_{n+1,j} = \alpha_{n+1} - \alpha_i + \alpha_j - \alpha_{n+1} = \alpha_j - \alpha_i = \sigma_{ij}$$

and further, σ' would be in the semantics of D, since $\sigma'_{i,n+1} = \alpha_{n+1} - \alpha_i \in D_{i,n+1}$ by construction.

For proving property 8.9 let us first prove that, for all $i, j \in [1 \ldots n]$,

$$(\alpha_i + D_{i,n+1}) \cap (\alpha_j + D_{j,n+1}) \neq \emptyset \tag{8.10}$$

To this end observe that, by the triangle inclusion 8.8,

$$\alpha_i + D_{i,n+1} - \alpha_j - D_{j,n+1} = \alpha_i - \alpha_j + D_{i,n+1} + D_{n+1,j} \supseteq \sigma_{ji} + D_{ij}$$

and since $\sigma_{ji} = -\sigma_{ij} \in -D_{ij}$ it follows that $0 \in \sigma_{ji} + D_{ij}$. Hence

$$0 \in \alpha_i + D_{i,n+1} - \alpha_j - D_{j,n+1}$$

which is equivalent to property 8.10.

But then, due to convexity,

$$\bigcap_{i=1}^{n} \left(\alpha_i + D_{i,n+1}\right) \neq \emptyset$$

fact which shows the existence of a real α_{n+1} that satisfies property 8.9. □

The following property says that the adjective "normal form" is correctly chosen to characterize the property:

Proposition 8.1.6. *For any two n-DBMs in normal form, $D, D' \in \mathcal{D}\mathsf{nf}_n$ with $D_{ij} \subseteq D'_{ij}$ for all $i, j \in [1 \ldots n]$ and $D_{ij} \neq D'_{ij}$ for some $i, j \in [1 \ldots n]$ we have that*

$$\|D\| \subsetneq \|D'\|$$

Proof. This property is a corollary of Proposition 8.1.5: take some n-DBM in normal form $D \in \mathcal{D}\mathsf{nf}_n$. The property is trivial if all components are point intervals, because choosing D' as in the statement would lead to having $D'_{ij} = \emptyset$ for some $i, j \in [1 \ldots n]$.

Suppose then that D_{ij} has a nonempty interior for some $i, j \in [1 \ldots n]$, say $D_{ij} =]\alpha, \beta[$ (the cases with other parentheses are treated similarly). Suppose also we have some other n-DBM in normal form D' with $D'_{ij} \subsetneq D_{ij}$. Take then $\gamma \in D_{ij} \setminus D'_{ij}$. Note that γ can be regarded as a 2-signal.

But then, by the construction in Proposition 8.1.5, this 2-signal can be inductively extended to an n-signal, denote it $\overline{\gamma}$, which belongs to the semantics of D. This ends the proof, since $\overline{\gamma} \notin \|D'\|$ due to the fact that $\overline{\gamma}_{ij} = \gamma \notin D'_{ij}$. □

For each sets of real numbers $A \subseteq \mathbb{R}$, denote $[A]$ the convex closure of A, that is:

$$[A] = \{\lambda x + (1-\lambda)y \mid x, y \in A, \lambda \in [0,1]\} \tag{8.11}$$

Observe that the convex closure commutes with summation: for any two sets $A, B \subseteq \mathbb{R}$,

$$[A] + [B] = [A + B] \tag{8.12}$$

Proposition 8.1.7. *Given two n-DBMs in normal form D, D' the following n-DBM is also in normal form:*

$$[D \cup D']_{ij} = [D_{ij} \cup D'_{ij}]$$

More generally, for any set of n-DBMs in normal form $\mathcal{D} = (D_i)_{i\in\mathcal{I}}$ (nonnecessarily finite or countable), the following n-DBM is in normal form:

$$[\mathcal{D}]_{ij} = \Big[\bigcup_{D\in\mathcal{D}} D_{ij}\Big]$$

Proof. By verification of the triangle inclusion 8.8: given a triplet $i, j, k \in [1 \dots n]$,

$$\begin{aligned}
[D \cup D']_{ij} + [D \cup D']_{jk} &= [D_{ij} \cup D'_{ij}] + [D_{jk} \cup D'_{jk}] \\
&= \big[(D_{ij} \cup D'_{ij}) + (D_{jk} \cup D'_{jk})\big] \\
&= \big[(D_{ij} + D_{jk}) \cup (D_{ij} + D'_{jk}) \cup (D'_{ij} + D_{jk}) \cup (D'_{ij} + D'_{jk})\big] \\
&\supseteq \big[(D_{ij} + D_{jk}) \cup (D'_{ij} + D'_{jk})\big] \\
&\supseteq \big[D_{ik} \cup D'_{ik}\big] \qquad \text{by assumption that } D \text{ is in normal form} \\
&= [D \cup D']_{ik}
\end{aligned}$$

The generalization to arbitrary families of n-DBMs in normal form follows along the same lines, due to the distributivity of summation over union. Note that we rely on the fact that the convex closure of any set of real numbers is an interval. □

Propositions 8.1.6 and 8.1.7 allow us to define the *normalization* of a DBM D *with nonempty semantics*: it is the largest DBM D' (with respect to inclusion) satisfying the triangle inclusion and bearing the property that $D'_{ij} \subseteq D_{ij}$ for all $i, j \in [1 \dots n]$. That is, D' is the normalization of D iff D' is in normal form, $\|D'\| \subseteq \|D\|$ and for all D'' with $\|D''\| \subseteq \|D\|$ we have $\|D''\| \subseteq \|D'\|$.

Proposition 8.1.8. *The normalization of each DBM is unique.*

If D' is the normalization of D then $\|D'\| = \|D\|$.

Proof. The first part of the statement follows by Proposition 8.1.7, since this proposition says that the set of DBMs in normal form whose semantics is included in $\|D\|$ forms a complete superior lattice [Bir79], which hence has a supremum.

The second part can be proved by contradiction: suppose that $\|D\| \neq \|D'\|$, hence we may pick a n-signal in $\|D\| \setminus \|D'\|$ and produce the convex closure $[D \cup \sigma]$, which would be, by Proposition 8.1.7, a n-DBM in normal form strictly larger than D', hence contradicting the assumption. □

To bring an n-DBM D with the property 8.7 into normal form, we may utilize another form of the Floyd-Warshall-Kleene algorithm: namely, we build the sequence of n-DBMs $(\Delta_k)_{k\in[0\ldots n]}$ inductively as follows [Gau99]:

$$\begin{aligned} \Delta_0 &= D \\ (\Delta_{k+1})_{ij} &= (\Delta_{k-1})_{ij} \cap \big((\Delta_{k-1})_{ik} + (\Delta_{k-1})_{kj} \qquad \text{for each } i,j \in [1\ldots n] \end{aligned}$$

The correctness of this algorithm is the subject of the following proposition. Similar results can be found in [Gau99, Tri98][3].

Proposition 8.1.9. *Provided D has a nonempty semantics, Δ_n is the normalization of the DBM D.*

Proof. Observe first that the nonemptiness hypothesis on $\|D\|$ implies that, for each $i,j \in [1\ldots n]$,

$$\begin{aligned} \bigcap \Big\{ \sum_{l=1}^{p-1} D_{i_l i_{l+1}} \mid i_l \in [1\ldots n], p\in\mathbb{N}, i_1 = i, i_p = j \Big\} = \\ = \bigcap \Big\{ \sum_{l=1}^{p-1} D_{i_l i_{l+1}} \mid i_l \in [1\ldots n], \forall l_1, l_2 \in [1\ldots p], i_{l_1} \neq i_{l_2}, i_1 = i, i_p = j, p\in\mathbb{N} \Big\} \end{aligned} \quad (8.13)$$

This follows because, if we have $i_{l_1} = i_{l_2}$ then $0 \in D_{i_{l_1},i_{l_1+1}} + \ldots + D_{i_{l_2-1},i_{l_2}}$ and hence

$$\begin{aligned} D_{i_1 i_2} + \ldots D_{i_{l_1-1},i_{l_1}} + D_{i_{l_1},i_{l_1+1}} + \ldots + D_{i_{l_2-1},i_{l_2}} + D_{i_{l_2},i_{l_2+1}} + \ldots + D_{i_{p-1} i_l} \supseteq \\ \supseteq D_{i_1 i_2} + \ldots D_{i_{l_1-1},i_{l_1}} + D_{i_{l_2},i_{l_2+1}} + \ldots + D_{i_{p-1} i_l} \end{aligned}$$

and this implies that, in the intersection in 8.13, sums in which an index is repeated are "useless".

Let us denote $\overline{D}$ the set defined in Identity 8.13. Then $\|\overline{D}\| = \|D\|$ due to the fact that all n-signals in $\|D\|$ must obey the triangle identity. It only remains to show that $\overline{D}$ is in normal form. To this end, observe that, for each $i,j \in [1\ldots n]$,

$$\begin{aligned} \overline{D}_{ij} + \overline{D}_{jk} = \bigcap \Big\{ \sum_{l=1}^{p-1} D_{i_l i_{l+1}} \mid i_l \in [1\ldots n], \forall l_1, l_2 \in [1\ldots p], i_{l_1} \neq i_{l_2}, i_1 = i, i_p = j, p\in\mathbb{N} \Big\} + \\ + \bigcap \Big\{ \sum_{l=1}^{q-1} D_{j_l j_{l+1}} \mid j_l \in [1\ldots n], \forall l_1, l_2 \in [1\ldots q], j_{l_1} \neq j_{l_2}, j_1 = j, j_q = k, q\in\mathbb{N} \Big\} \end{aligned} \quad (8.14)$$

Remind now that the distributivity of intersection over summation

$$A \cap (B + C) = (A \cap B) + (A \cap C)$$

holds iff in the right hand side the intersections are nonempty. This is our case since we have assumed that the semantics of D is nonempty and hence each intersection in Identity 8.14 must be nonempty. Hence we may apply this distributivity (in the reverse direction) to get:

[3] Such results are an essential tool in the computation of the set of reachable states in timed automata.

$$\overline{D}_{ij} + \overline{D}_{jk} = \bigcap \Big\{ \sum_{l=1}^{p-1} D_{i_l i_{l+1}} + \sum_{l=1}^{q-1} D_{j_l j_{l+1}} \mid i_l \in [1 \dots n], \forall l_1, l_2 \in [1 \dots p], i_{l_1} \neq i_{l_2}, i_1 = i, \\ i_p = j, p \in \mathbb{N}, j_l \in [1 \dots n], \forall l_1, l_2 \in [1 \dots q], j_{l_1} \neq j_{l_2}, j_1 = j, j_q = k, q \in \mathbb{N} \Big\}$$

and then observe that from the two sequences of indices we may construct a single sequence that starts in i and ends in k, which means that the above intersection is included in the left-hand side of identity 8.13, written for the indices i and k. □

8.2 Regions

In this chapter we will be interested in a special class of DBMs in normal form, namely DBMs in which each interval is either a point interval or a unit interval. The reason to do this is that we want to represent (E)DBMs as sets of regions.

We will call such DBMs as *regions*, due to their close connection to the regions in timed automata [AD94].

Definition 8.2.1. *An n-DBM $D \in \mathcal{D}\mathsf{bm}_n$ is called an n-**region** if it has a nonempty semantics and, for each $i, j \in [1 \dots n]$,*

- *Either D_{ij} is a point interval $D_{ij} = \{\alpha\}$ for some $\alpha \in \mathbb{Z}$,*
- *Or D_{ij} is an open interval of unit length $D_{ij} =]\beta, \beta+1[$ for some $\beta \in \mathbb{Z}$.*

The set of regions is denoted $\mathcal{R}\mathsf{egn}_n$.

Proposition 8.2.2. *Each region is in normal form.*

Proof. Take some region $D \in \mathcal{R}\mathsf{egn}_n$ and suppose it is not in normal form, that is, there exist some $i, j, k \in [1 \dots n]$ such that $D_{ij} \not\subseteq D_{ik} + D_{kj}$. We will show then that it does not verify the nonemptiness property 8.7. We have the following cases to analyze:

1. All three intervals D_{ij}, D_{ik} and D_{kj} are point intervals, say $D_{ij} = \{\alpha\}$, $D_{ik} = \{\beta\}$ and $D_{kj} = \{\gamma\}$. Then it follows that $\alpha \neq \beta + \gamma$. But this implies that

   $$D_{ij} + D_{jk} + D_{ki} = \{\alpha - \beta - \gamma\} \neq \{0\}$$

   which is in contradiction with property 8.7.
2. $D_{ij} =]\alpha, \alpha+1[$, $D_{ik} =]\beta, \beta+1[$ and $D_{kj} = \{\gamma\}$. Then the assumption that $D_{ij} \not\subseteq D_{ik} + D_{kj}$ rewrites to $]\alpha, \alpha+1[\not\subseteq]\beta+\gamma, \beta+\gamma+1[$. Hence we must have either $\alpha \geq \beta + \gamma + 1$ or $\alpha + 1 \leq \beta + \gamma$.
   It then follows that

   $$D_{ij} + D_{jk} + D_{ki} =]\alpha-\gamma-\beta-1, \alpha-\gamma-\beta[\not\ni 0$$

   The case when D_{jk} is a point interval and D_{ik} is an open unit interval is similar.

3. $D_{ij} =]\alpha, \alpha{+}1[$, $D_{ik} =]\beta, \beta{+}1[$ and $D_{kj} =]\gamma, \gamma{+}1[$. Then the assumption that $D_{ij} \not\subseteq D_{ik} + D_{kj}$ rewrites to $]\alpha, \alpha{+}1[\not\subseteq]\beta{+}\gamma, \beta{+}\gamma{+}2[$. Hence we must have either $\alpha \geq \beta{+}\gamma{+}2$ or $\alpha{+}1 \leq \beta{+}\gamma$. But again this implies that $0 \notin D_{ij} + D_{jk} + D_{ki}$, as it can be easily seen by verification. □

In the introduction to this subsection we have talked about decomposition of each DBM into a union of regions. The formalization of this is given by the "inclusion" relation, ' $\sqsubseteq$ ' $\subseteq \mathcal{R}\mathsf{egn}_n \times \mathcal{E}\mathsf{dbm}_n$, defined as follows

$$R \sqsubseteq D \text{ iff } R_{ij} \subseteq D_{ij} \text{ for all } i, j \in [1 \dots n]$$

If $R \sqsubseteq D$ for some region R and EDBM D, then we say that R is *included in* D, or that D *includes* R. We also denote $D \sqsupseteq R$ in this case.

Of course, $\sqsubseteq$ can be defined as a relation on EDBMs, but we utilize it only on $\mathcal{R}\mathsf{egn}_n \times \mathcal{E}\mathsf{dbm}_n$. For each $D \in \mathcal{E}\mathsf{dbm}_n$ we also denote $(\sqsupseteq D)$ the set of regions which are included in D.

The following property shows that, by replacing an EDBM D with the set of regions which are included in D we lose nothing w.r.t. semantics:

Proposition 8.2.3. *For each n-EDBM $D \in \mathcal{E}\mathsf{dbm}_n$,*

$$\|D\| = \bigcup \{ \|R\| \mid R \sqsubseteq D \}$$

Proof. The inverse inclusion is straightforward. For the direct inclusion, take some n-signal $\sigma \in \|D\|$. Define then the following DBM:

$$R_{ij} = \begin{cases} \{\sigma_{ij}\} & \text{iff } \sigma_{ij} \in \mathbb{N} \\ \big(\lfloor \sigma_{ij} \rfloor, \lceil \sigma_{ij} \rceil\big) & \text{iff } \sigma_{ij} \notin \mathbb{N} \end{cases}$$

By definition $R \ni \sigma$, hence R has a nonempty semantics, hence it is a region. On the other hand, for each $i, j \in [1 \dots n]$, $\sigma_{ij} \in D_{ij}$ by assumption. But then the following two cases arise, according to whether σ_{ij} is integer or not:

- $\sigma_{ij} \in \mathbb{Z}$. Then clearly $R_{ij} \subseteq D_{ij}$.
- $\sigma_{ij} \notin \mathbb{Z}$. But D_{ij} has integer bounds, hence either $\lfloor \sigma_{ij} \rfloor \in D_{ij}$ or $\lfloor \sigma_{ij} \rfloor$ is the lower bound of D_{ij}. Similarly, either $\lceil \sigma_{ij} \rceil \in D_{ij}$ or $\lceil \sigma_{ij} \rceil$ is the upper bound of D_{ij}. Hence in this case too $R_{ij} \subseteq D_{ij}$. □

Another essential property of regions is the fact that region representation of a set of n-signals is unique:

Proposition 8.2.4. *For each pair of n-regions $R, R' \in \mathcal{R}\mathsf{egn}_n$, if $\|R\| \cap \|R'\| \neq \emptyset$ then $R = R'$.*

Proof. The hypothesis implies that, for each $i, j \in [1 \dots n]$, $R_{ij} \cap R'_{ij} \neq \emptyset$. But as the intervals R_{ij} and R'_{ij} are either point intervals or open intervals of unit length, $R_{ij} \cap R'_{ij} \neq \emptyset$ is equivalent to $R_{ij} = R'_{ij}$. □

Corollary 8.2.5. *For each two sets of regions $\mathcal{R}_1, \mathcal{R}_2 \subseteq \mathcal{R}\mathsf{egn}_n$, $\|\mathcal{R}_1\| \subseteq \|\mathcal{R}_2\|$ if and only if $\mathcal{R}_1 \subseteq \mathcal{R}_2$.*

Proof. The inverse inclusion is straightforward. For the direct inclusion, take some region $R \in \mathcal{R}_1$, hence $\|R\| \subseteq \|\mathcal{R}_1\|$ which implies that $\|R\| \subseteq \|\mathcal{R}_2\|$. But then, for each $\sigma \in R$ there must exist a region $R' \in \mathcal{R}_2$ with $\sigma \in R'$. But $\sigma \in \|R\| \cap \|R'\|$, which implies that $R = R'$. □

8.2.1 Juxtaposition and concatenation on regions

The representation of EDBMs by regions would not be satisfactory if we would not have a compositional concatenation on these representations: compositionality is essentially needed for further representing EDBMs – and regular expressions over them – with the aid of n-automata.

In Chapter 6, Definition 6.4.3 we have introduced a juxtaposition operation on regsignals, hence we may think we only need consider its particularization to regions. Since projection is compositional on regions, we would have the desired compositional concatenation. However, with the definition from Chapter 6, juxtaposition is not an internal operation on regions, that is, the result might not necessarily be a region, but rather a DBM. An example was provided in the introductory part of this chapter.

The issue from this situation is to define juxtaposition as an operation that associates, to each pair of regions, a *set* of regions, namely those regions which are included in the DBM constructed by Definition 6.4.3. For avoiding working with that heavy definition, but also due to the compositionality of projection, we may define way more simpler the juxtaposition of regions as follows:

Definition 8.2.6. *Given two regions $R_1 \in \mathcal{R}\mathsf{egn}_m$ and $R_2 \in \mathcal{R}\mathsf{egn}_n$ and some integer $p \leq \min(m, n)$, suppose that*

$$R_1\big|_{[n-p+1\ldots n]} = R_2\big|_{[1\ldots p]} \tag{8.15}$$

*The **region p-juxtaposition** of R_1 and R_2 is the following set of $(m+n-p)$-regions:*

$$R_1 \square_p R_2 = \left\{ R \in \mathcal{R}\mathsf{egn}_{m+n-p} \mid R\big|_{[1\ldots m]} = R_1, R\big|_{[n-p+1\ldots m+n-p]} = R_2 \right\} \tag{8.16}$$

If $R_1\big|_{[n-p+1\ldots n]} \neq R_2\big|_{[1\ldots p]}$ then we put $R_1 \square_p R_2 = \emptyset$.

In the sequel, the juxtaposition operation from Definition 6.4.3, with regions as arguments, will be called as *DBM juxtaposition*.

The next concern is to check that all the (signal-)juxtaposition properties, as stated in Chapter 6, hold for region juxtaposition too.

Proposition 8.2.7. *1. Region juxtaposition is compositional: for each $R_1 \in \mathcal{R}\mathsf{egn}_m$ and $R_2 \in \mathcal{R}\mathsf{egn}_n$ and $p \leq \min(m, n)$,*

$$\|R_1 \square_p R_2\| = \|R_1\| \square_p \|R_2\| \tag{8.17}$$

2. *For each* $R \in \mathcal{R}\mathsf{egn}_m$, $R' \in \mathcal{R}\mathsf{egn}_n$, *for each* $X \subseteq [1 \dots m]$ *with* $card(X) = p$, *for each* $Y \subseteq [1 \dots n]$ *with* $card(Y) = q$ *and given* $r \leq \min(p, q)$, *suppose that* $[m - r + 1 \dots m] \subseteq X$ *and* $[1 \dots r] \subseteq Y$. *Then we have that*

$$R\big|_X \Box_r R'\big|_Y = (R \Box_r R')\big|_{X \cup (Y+m-r)}. \tag{8.18}$$

3. *Region juxtaposition is associative: for each* $R_1 \in \mathcal{R}\mathsf{egn}_m$, $R_2 \in \mathcal{R}\mathsf{egn}_n$ *and* $R_3 \in \mathcal{R}\mathsf{egn}_p$, *and for each* $k \leq \min(m, n)$ *and* $l \leq \min(n, p)$,

$$(R_1 \Box_k R_2) \Box_l R_3 = R_1 \Box_k (R_2 \Box_l R_3) \tag{8.19}$$

Proof. The first property follows due to the compositionality of projection: suppose that requirement 8.15 holds. Then:

$$\begin{aligned}
\|R_1 \Box_p R_2\| &= \big\{\sigma \in R \mid R\big|_{[1\dots m]} = R_1, R\big|_{[n-p+1\dots m+n-p]} = R_2\big\} \\
&= \big\{\sigma \in R \mid \sigma\big|_{[1\dots m]} \in \|R_1\|, \sigma\big|_{[n-p+1\dots m+n-p]} \in \|R_2\|\big\} \\
&= \|R_1\| \Box_p \|R_2\|.
\end{aligned}$$

In the case requirement 8.15 does not hold, there must exist some indices $i, j \in [1 \dots p]$ such that $(R_1)_{i+m-p,j+m-p} \neq (R_2)_{ij}$. But this implies that $(R_1)_{i+m-p,j+m-p} \cap (R_2)_{ij} = \emptyset$, fact which assures that for any two n-signals $\sigma_1 \in \|R_1\|$ and $\sigma_2 \in \|R_2\|$ we must have $(\sigma_1)_{i+m-p,j+m-p} \neq \sigma_{ij}$. Therefore, the sets $\|R_1\|$ and $\|R_2\|$ cannot be p-juxtaposed, i.e. $\|R_1\| \Box_p \|R_2\| = \emptyset$. As a consequence, $\|R_1 \Box_p R_2\| = \emptyset = \|R_1\| \Box_p \|R_2\|$.

The other two properties can be proved with the aid of the compositionality of juxtaposition and projection, by using Corollary 8.2.5 and Proposition 6.2.7:

$$\begin{aligned}
\|R\big|_X \Box_r R'\big|_Y\| &= \|R_1\|\big|_X \Box_r \|R_2\|\big|_Y && \text{by compositionality of juxtaposition} \\
&= (\|R\| \Box_r \|R\|')\big|_{X \cup (Y+m-r)} && \text{by Proposition 6.2.7} \\
&= \|(R \Box_r R')\|\big|_{X \cup (Y+m-r)} && \text{by compositionality of projection} \\
\|(R_1 \Box_k R_2) \Box_l \|R_3\| &= \big(\|R_1\| \Box_k \|R_2\|\big) \Box_l \|R_3\| \\
&= \|R_1\| \Box_k \big(\|R_2\| \Box_l R_3\big) && \text{by associativity of } \Box_p \\
&= \|R_1 \Box_k (R_2 \Box_l R_3)\| && \Box
\end{aligned}$$

Consequently we may define the desired compositional concatenation on $2n$-regions:

Definition 8.2.8. *Given* $R_1, R_2 \in \mathcal{R}\mathsf{egn}_{2n}$, *the* $2n$-***region-concatenation*** *of* R_1 *and* R_2 *is the following set of* $2n$*-regions:*

$$R_1 \odot R_2 = \big\{R\big|_{[1\dots n]\cup[2n+1\dots 3n]} \mid R \in R_1 \Box_n R_2\big\} \tag{8.20}$$

The following proposition shows that region concatenation shares almost the same properties as $2n$-signal concatenation:

Proposition 8.2.9. *1. Region concatenation is compositional: for each two $2n$-regions, $R_1, R_2 \in \mathcal{R}\mathsf{egn}_{2n}$,*

$$\|R_1 \odot R_2\| = \|R_1\| \odot \|R_2\| \tag{8.21}$$

Observe that here we have implicitly extended the semantics application $\|\cdot\|$ to sets *of regions.*

2. Region concatenation is associative: for each triplet of $2n$-regions $R_1, R_2, R_3 \in \mathcal{R}\mathsf{egn}_{2n}$,

$$R_1 \odot (R_2 \odot R_3) = (R_1 \odot R_2) \odot R_3 \tag{8.22}$$

3. Each region has a left and a right unit: for each $R \in \mathcal{R}\mathsf{egn}_{2n}$ there exist $1^l_R, 1^r_R \in \mathcal{R}\mathsf{egn}_{2n}$ such that

$$1^l_R \odot R = R \odot 1^r_R = \{R\} \tag{8.23}$$

The definitions of the left and right units are the following:

$$\begin{aligned}(1^l_R)_{ij} &= (1^l_R)_{n+i,j} = (1^l_R)_{i,n+j} = (1^l_R)_{n+i,n+j} = R_{ij}\\ (1^r_R)_{ij} &= (1^l_R)_{n+i,j} = (1^l_R)_{i,n+j} = (1^l_R)_{n+i,n+j} = R_{n+i,n+j}\end{aligned}$$

4. For each region $R \in \mathcal{R}\mathsf{egn}_{2n}$ there exists a weak inverse *$\tilde{R}$ with the property that*

$$1^l_R \in R \odot \tilde{R}, \qquad 1^r_R \in \tilde{R} \odot R \tag{8.24}$$

The definition of the weak inverse is the following:

$$\tilde{R}_{ij} = \begin{cases} R_{i+n,j+n} & \textit{iff } i,j \in [1 \ldots n] \\ R_{i+n,j-n} & \textit{iff } i \in [1 \ldots n], j \in [n+1 \ldots 2n] \\ R_{i-n,j+n} & \textit{iff } i \in [n+1 \ldots 2n], j \in [1 \ldots n] \\ R_{i-n,j-n} & \textit{iff } i,j \in [n+1 \ldots 2n] \end{cases} \tag{8.25}$$

Proof. The first property is a straightforward corollary of the compositionality of juxtaposition and projection, while the second property follows from the associativity of region juxtaposition and its proof runs exactly as the proof of the associativity of $2n$-signal composition – see Proposition 6.2.10.

The other two properties have more specific proofs, compared to their "relatives" form Proposition 6.2.10, and this is due to the particularity of region juxtaposition. Still both of them rely essentially on compositionality.

For proving property 3, observe first that each $2n$-signal $\tau \in 1^r_R$ is a unit of the form 1^r_θ for some $\theta \in \mathsf{Sig}_{2n}(\{a\})$. On the other hand, for any $2n$-signal $\sigma \in \mathsf{Sig}_{2n}(\Sigma)$, if $\sigma \odot 1^r_\theta$ is defined then $1^r_\theta\big|_{[1\ldots n]} = \sigma\big|_{[n+1\ldots 2n]}$. But this implies that $1^r_\theta = 1^r_\sigma$ and therefore

$$\begin{aligned}\|R \odot 1^r_R\| &= \{\sigma \odot \tau \mid \sigma \in \|R\|, \tau \in \|1^r_R\|\} \\ &= \{\sigma \odot 1^r_\theta \mid \sigma \in \|R\|, 1^r_\theta\big|_{[1\ldots n]} = \sigma\big|_{[n+1\ldots 2n]}\} \\ &= \{\sigma \odot 1^r_\sigma \mid \sigma \in \|R\|\} \\ &= \|R\|\end{aligned}$$

The equality $R \odot 1^r_R = \{R\}$ follows then by the uniqueness property 8.2.5. The validity of the identity regarding the right unit can be established similarly.

The proof of the last property proceeds along similar lines: for each $\sigma \in R$ we have that $\tilde{\sigma} \in \|\tilde{R}\|$ and $\sigma \odot \tilde{\sigma} = 1^l_\sigma$. Hence $1^l_\sigma \in \|R \odot \tilde{R}\|$, which implies that $\|1^l_R\| \subseteq \|R \odot \tilde{R}\|$. And here we apply Corollary 8.2.5 to get that $1^l_R \in R \odot \tilde{R}$. □

Let us observe here that only the weak inverse property can be obtained, that is, we cannot have equality in Identity 8.24, since in general the set $R \odot \tilde{R}$ might have cardinality greater than 1.

As usual, any operation defined on elements of a certain type can be easily extended to sets of elements. Therefore we also dispose of the following compositional concatenation on *sets of regions*: for each pair of sets of regions $\mathcal{R}_1, \mathcal{R}_2 \subseteq \mathcal{R}\mathsf{egn}_{2n}$,

$$\mathcal{R}_1 \odot \mathcal{R}_2 = \{R_1 \odot R_2 \mid R_1 \in \mathcal{R}_1, R_2 \in \mathcal{R}_2\}$$

Let us also denote the set of all left and right units for $2n$-region concatenation as $\mathbf{1}_{2n}$:

$$\mathbf{1}_{2n} = \{R \in \mathcal{R}\mathsf{egn}_{2n} \mid \forall i \in X, R_{i,n+i} = 0\} = \{1^l_R \mid R \in \mathcal{R}\mathsf{egn}_{2n}\} \tag{8.26}$$

It is easy to see that $\mathbf{1}_{2n}$ is a unit for concatenation on sets of regions. Once in the possession of this unit we may define the *star* operation $\circledast$ on sets of regions as follows: for each $\mathcal{R} \subseteq \mathcal{R}\mathsf{egn}_{2n}$

$$\mathcal{R}^{\circledast} = \bigcup_{k \in \mathbb{N}} \mathcal{R}^{k\odot}$$

where $\mathcal{R}^{0\odot} = \mathbf{1}_{2n}$ and $\mathcal{R}^{(k+1)\odot} = \mathcal{R}^{k\odot} \odot \mathcal{R}$ for all $k \in \mathbb{N}$.

Proposition 8.2.10. *For any sets of regions $\mathcal{R}, \mathcal{R}' \subseteq \mathcal{R}\mathsf{egn}_{2n}$,*

$$\|\mathcal{R} \odot \mathcal{R}'\| = \|\mathcal{R}\| \odot \|\mathcal{R}'\| \tag{8.27}$$

$$\|\mathcal{R}^{\circledast}\| = \|\mathcal{R}\|^{\circledast} \tag{8.28}$$

Consequently, $\big(\mathcal{P}(\mathcal{R}\mathsf{egn}_{2n}), \cup, \emptyset, \odot, \mathbf{1}_{2n}, (\cdot)^{\circledast}\big)$ is a Kleene algebra.

8.3 Representing DBMs with the aid of n-words and n-relations

In this section we formalize the possibility to represent sets of n-regions with pairs consisting of a n-word and a matrix of relational symbols.

8.3.1 n-relations

Definition 8.3.1. *An n-**relation** is an $n \times n$ matrix M over the set of relation symbols $\Gamma = \{<, =, >\}$ satisfying the following property (called in the sequel as the **consistency property**):*

There exists no cycle in the matrix $(i_j)_{j\in[1\ldots k+1]}$ with $i_j \in [1 \ldots n]$, $k \geq 1$ and $i_{k+1} = i_1$ such that

- *For all $j \in [1 \ldots k]$, $M_{i_j,i_{j+1}} \in \{`<`, `=`\}$;*
- *For some $j \in [1 \ldots k]$, $M_{i_j,i_{j+1}} = `<`$.*

We denote the set of n-relations as Γ_n.

Observe that any n-relation M is an *antisymmetric matrix*, that is:

- $M_{ii} = `=`$ and
- If $M_{ij} = `=`$ then $M_{ji} = `=`$;
- If $M_{ij} = `<`$ then $M_{ji} = `>`$;
- If $M_{ij} = `>`$ then $M_{ji} = `<`$.

An alternative definition of n-relations is the following: for each sequence of indices $\mathbf{m} = (m_r)_{r\in[1\ldots p]}$ with $m_{p+1} = m_1$, denote

$$A(\mathbf{m}) = \{r \in [1 \ldots p] \mid M_{m_r m_{r+1}} = `=`\}$$
$$B(\mathbf{m}) = \{r \in [1 \ldots p] \mid M_{m_r m_{r+1}} = `<`\}$$
$$C(\mathbf{m}) = \{r \in [1 \ldots p] \mid M_{m_r m_{r+1}} = `>`\}$$

Then an n-relation is an $n \times n$ matrix over Γ bearing the property that

$$\text{If } card(A(\mathbf{m})) \leq p-1 \text{ then both } card(B(\mathbf{m})) \geq 1 \text{ and } card(C(\mathbf{m})) \geq 1. \tag{8.29}$$

Let us also provide a simple way for checking whether a matrix $M \in \mathcal{M}_{n\times n}(\Gamma)$ is an n-relation:

Proposition 8.3.2. *M is an n-relation if and only if it is antisymmetric and the consistency property holds for all cycles of length equal to 3.*

Proof. The reverse implication can be proved as follows: take some arbitrary sequence of indices $\mathbf{m} = (m_r)_{r\in[1\ldots p]}$ with $m_{p+1} = m_1$. Suppose that property 8.29 is false for this sequence. Of course, the antisymmetry and the hypothesis imply that $p \geq 4$. We will show that the same property is false for a shorter sequence, fact which, by induction, would imply that property 8.29 would be false for a sequence of length 3.

To this end, assume w.l.o.g. that $card(C(\mathbf{m})) = 0$, hence we have $card(A(\mathbf{m}))+card(B(\mathbf{m})) = n$. We have 4 cases to study, according to whether $M_{m_1m_2}$ is $`<`$ or $`=`$, respectively to whether $M_{m_2m_3}$ is $`<`$ or $`=`$.

Consider first the case $M_{m_1m_2} = M_{m_2m_3} = `<`$. Since by hypothesis, the sequence (m_1, m_2, m_3) satisfies property 8.29, we must have $M_{m_3m_1} = `>`$, hence, by antisymmetry,

$M_{m_1m_3} =$ ‘ $<$ ‘. But then the sequence $\mathbf{m}' = (m_1, m_3, \ldots, m_p)$ does *not* satisfy property 8.29, because we have replaced two ‘ $<$ ‘ signs by with one ‘ $<$ ‘ sign, hence

$$A(\mathbf{m}') = A(\mathbf{m}), \quad B(\mathbf{m}') = B(\mathbf{m}) - 1, \quad C(\mathbf{m}') = C(\mathbf{m}) = 0$$

Similarly, if $M_{m_1m_2} =$ ‘ $<$ ‘ and $M_{m_2m_3} =$ ‘ $=$ ‘ then $M_{m_1m_3} =$ ‘ $>$ ‘ and again the sequence $\mathbf{m}'$ does not satisfy property 8.29. The other cases are treated similarly. □

Remark 8.3.3. By Proposition 8.3.3, we have that in an n-relation M the length 3 cycles may only be of the types (‘ $=$ ‘,‘ $=$ ‘,‘ $=$ ‘), (‘ $<$ ‘,‘ $=$ ‘,‘ $>$ ‘), (‘ $=$ ‘,‘ $<$ ‘,‘ $>$ ‘), (‘ $<$ ‘,‘ $<$ ‘,‘ $>$ ‘) and their circular permutations.

8.3.2 Operations on n-relations

Since our aim is to represent sets of regions (and thus EDBMs) with the aid of n-words and n-relations, we need to provide an algebraic calculus of composition and star on $2n$-relations too. This is the issue of this subsection.

The first operation to be defined is projection: for each $M \in \Gamma_n$ and $X \subseteq [1 \ldots n]$, the X-**projection** of M is the $card(X)$-relation resulting by deleting from M the rows and columns that are not in X.

Definition 8.3.4. *Given an m-relation $M_1 \in \Gamma_m$, an n-relation $M_2 \in \Gamma_n$ and a positive integer $p \leq \min(m, n)$, if*

$$M_1\big|_{[m-p+1\ldots m]} = M_2\big|_{[1\ldots p]} \tag{8.30}$$

*then the p-**juxtaposition** of M_1 with M_2 is the following set of $(m + n - p)$-relations:*

$$M_1 \square_p M_2 = \left\{ M \in \Gamma_{m+n-p} \mid M\big|_{[1\ldots m]} = M_1, M\big|_{[m-p+1\ldots m+n-p]} = M_2 \right\} \tag{8.31}$$

If requirement 8.30 is not satisfied then we put $M_1 \square_p M_2 = \emptyset$.

The following proposition shows that relation juxtaposition enjoys all the properties of the juxtaposition operations seen so far:

Proposition 8.3.5. *1. Given an m-relation $M_1 \in \Gamma_m$, an n-relation $M_2 \in \Gamma_n$ and a positive integer $p \leq \min(m, n)$, $M_1 \square_p M_2 \neq \emptyset$ if and only if M_1 and M_2 verify the requirement 8.30.*

2. *For each $M_1 \in \mathcal{R}\mathsf{egn}_m$, $M_2 \in \mathcal{R}\mathsf{egn}_n$, for each $X \subseteq [1 \ldots m]$ with $card(X) = p$, for each $Y \subseteq [1 \ldots n]$ with $card(Y) = q$ and each $r \leq \min(p, q)$, suppose that $[m - r + 1 \ldots m] \subseteq X$ and $[1 \ldots r] \subseteq Y$. Then we have that*

$$M_1\big|_X \square_r M_2\big|_Y = (M_1 \square_r M_2)\big|_{X \cup (Y+m-r)}. \tag{8.32}$$

3. *Relation juxtaposition is associative: for each $M_1 \in \Gamma_m$, $M_2 \in \Gamma_n$ and $M_3 \in \Gamma_p$, and for each $k \leq \min(m, n)$ and $l \leq \min(n, p)$,*

$$(M_1 \square_k M_2) \square_l M_3 = M_1 \square_k (M_2 \square_l M_3) \tag{8.33}$$

Proof. Concerning the first property, note first that we are interested in proving the reverse implication, since the direct implication is contained in the definition of juxtaposition. Hence consider two relations M_1 and M_2 satisfying the hypotheses. In order to build a $(m+n-p)$-relation from them, we must first see what could be the possible choices for the components (i,j) in the new relation, where $i \in [1 \dots m-r+1]$ and $j \in [m+1 \dots m+n-p]$.

So let's consider, for each such pair of indices $(i,j) \in [1 \dots m-r+1] \times [m+1 \dots m+n-p]$, the following set of pairs of relation symbols:

$$\Delta_{ij} = \left\{ \left((M_1)_{ik}, (M_2)_{k-m+p,j} \right) \mid k \in [m-p+1 \dots m] \right\}$$

Let us then call pairs that are different from ($`<`$,$`>`$) or ($`>`$,$`<`$) as *critical*. We would like to prove that, once some critical pair is in Δ_{ij}, all the other critical pairs from Δ_{ij} do not "contradict" it. There could be several "contradictory" pairs that may occur: ($`<`$,$`<`$) and ($`>`$,$`>`$), or ($`<`$,$`=`$) and ($`>`$,$`=`$), or ($`=`$,$`=`$) and ($`>`$,$`>`$), or ($`<`$,$`<`$) and ($`>`$,$`>`$), or their symmetrics/cyclic permutations. We will prove the impossibility of occurrence only for the first case, the other proofs being similar.

Suppose the contrary, hence there exists $k,l \in [m-p+1 \dots m]$ such that $(M_1)_{ik} = `<`$, $(M_2)_{k-m+p,j} = `<`$, and $(M_1)_{il} = `>`$, $(M_2)_{l-m+p,j} = `>`$. But then, by the consistency requirement, $(M_1)_{kl} = `>`$ and $(M_2)_{k-m+p,l} = `<`$, which is in contradiction with the assumption 8.30. Hence, the two critical pairs cannot occur both in Δ_{ij}.

It follows that, if $((M_1)_{ik}, (M_2)_{k-m+p,j})$ is a critical pair, then *there is only one choice* of a third relation symbol M_{ji} such that the triplet $((M_1)_{ik}, (M_2)_{k-m+p,j}, M_{ji})$ satisfies the consistency requirement for cycles of length 3. Moreover, this choice is the same for two critical pairs that are "noncontradictory".

As a consequence, an $(m+n-p)$-relation in $M_1 \square_p M_2$ can be constructed the following way:

- For each $i,j \in [1 \dots m]$, $M_{ij} = (M_1)_{ij}$.
- For each $i,j \in [m-p+1 \dots m+n-p]$, $M_{ij} = (M_2)_{i-m+p,j-m+p}$.
- For each $i \in [1 \dots m-p]$ and $j \in [m+1 \dots m+n-p]$, M_{ij} is a choice which is consistent with any (i.e. all) of the critical pairs in Δ_{ij}. Also M_{ji} is the reverse of M_{ij}.

Observe that the pairs ($`<`$,$`>`$) and ($`>`$,$`<`$) are noncritical since they are consistent with *any* choice of a third relation symbol. That is, if a set Δ_{ij} contains only noncritical pairs then the choice of M_{ij} can be any relation symbol.

This observation ends the proof of the first property. However we will observe that the same proof could be employed for establishing the truth of the following two related properties:

(A) Given an m-relation $M_1 \in \Gamma_m$, an n-relation $M_2 \in \Gamma_n$, a positive integer $p \leq \min(m,n)$ and a set $X \subseteq [1 \dots m]$ with $card(X) = p$, if $M_1\big|_X = M_2\big|_{[1\dots p]}$ then there exists some $M \in \Gamma_{m+n-p}$ such that $M\big|_{[1\dots m]} = M_1$ and $M\big|_{[m-p+1\dots m+n-p]} = M_2$.

(B) With the same hypotheses as above, and considering also a set $Y \subseteq [1 \dots n]$ with $card(Y) = p$, if $M_1\big|_{[m-p+1\dots m]} = M_2\big|_Y$ then there exists some $M \in \Gamma_{m+n-p}$ such that $M\big|_{[1\dots m]} = M_1$ and $M\big|_{[m-p+1\dots m+n-p]} = M_2$.

In fact, these properties can be regarded as connected to a definition of a *set-indexed juxtaposition*, something of the type $M_1 \square_{X,p} M_2$, respectively $M_1 \square_{p,Y} M_2$.

For proving the second property, start with $M_1 \in \Gamma_m$, $M_2 \in \Gamma_n$, and with sets X, Y and integers p, q, r as in the statement of the property. The right-to-left inclusion of Identity 8.32 is straightforward since for each $(m+n-p)$-relation $M \in (M_1 \square_p M_2)\big|_{X \cup (Y+m-r)}$ we have that

$$M\big|_{[1\ldots k]} \in (M_1 \square_p M_2)\big|_{X\cup(Y+m-r)}\big|_{[1\ldots p]} = (M_1 \square_p M_2)\big|_X = \{M_1\big|_X\}$$
$$M\big|_{[p-r+1\ldots p+q-r]} \in (M_1 \square_p M_2)\big|_{X\cup(Y+m-r)}\big|_{[p-r+1\ldots p+q-r]} = (M_1 \square_p M_2)\big|_{Y+m-r} = \{M_2\big|_Y\}$$

For the left-to-right inclusion we will essentially rely on the two properties (A) and (B): take a $(p+q-r)$-relation $M \in M_1\big|_X \square_r M_2\big|_Y$, that is, $M\big|_{[1\ldots p]} = M_1\big|_X$ and $M\big|_{[p-r+1\ldots p+q-r]} = M_2\big|_Y$.

From $M\big|_{[1\ldots p]} = M_1\big|_X$ we conclude, by means of property (A), that there exists an $(m+q-r)$-relation $\overline{M}_1 \in \Gamma_{m+q-r}$ such that

$$\overline{M}_1\big|_{[1\ldots m]} = M_1 \text{ and } \overline{M}_1\big|_{[m-p+1\ldots m+q-r]} = M$$

From $M\big|_{[p-r+1\ldots p+q-r]} = M_2\big|_Y$ we deduce, by applying property (B), that there exists a $(n+p-r)$-relation $\overline{M}_2 \in \Gamma_{m+q-r}$ such that

$$\overline{M}_2\big|_{[1\ldots p+q-r]} = M \text{ and } \overline{M}_2\big|_{[p-r+1\ldots n+p-r]} = M_2$$

But these two choices imply that

$$\overline{M}_1\big|_{[m-p+1\ldots m+q-r]} = \overline{M}_2\big|_{[1\ldots p+q-r]}$$

hence there must further exist $\overline{M} \in \Gamma_{m+n-r}$ such that

$$\overline{M}\big|_{[1\ldots m+q-r]} = \overline{M}_1 \text{ and } \overline{M}\big|_{[m-p+1\ldots m+n-r]} = \overline{M}_2$$

We then only have to observe that this $(m+n-r)$-relation belongs to $M_1 \square_p M_2$ because:

$$\overline{M}\big|_{[1\ldots m]} = \overline{M}\big|_{[1\ldots m+q-r]}\big|_{[1\ldots m]} = \overline{M}_1\big|_{[1\ldots m]} = M_1$$
$$\overline{M}\big|_{[m-r+1\ldots m+n-r]} = \overline{M}\big|_{[m-p+1\ldots m+n-r]}\big|_{[p-r+1\ldots n+p-r]} = \overline{M}_2\big|_{[p+r-1\ldots n+p-r]} = M_2$$

Finally, the proof of the third property follows by easy verification:

$$
\begin{aligned}
(M_1 \square_k M_2) \square_l M_3 &= \big\{ M \in \Gamma_{m+n+p-k-l} \mid M\big|_{[1\ldots m+n-k]} \in M_1 \square_k M_2 \\
&\qquad \text{and } M\big|_{[m+n-k-l+1\ldots m+n+p-k-l]} = M_3 \big\} \\
&= \big\{ M \in \Gamma_{m+n+p-k-l} \mid M\big|_{[1\ldots m+n-k]}\big|_{[1\ldots m]} = M_1, \\
&\qquad M\big|_{[1\ldots m+n-k]}\big|_{[m-k+1\ldots m+n-k]} = M_2, \\
&\qquad \text{and } M\big|_{[m+n-k-l+1\ldots m+n+p-k-l]} = M_3 \big\} \\
&= \big\{ M \in \Gamma_{m+n+p-k-l} \mid M\big|_{[1\ldots m]} = M_1, M\big|_{[m-k+1\ldots m+n-k]} = M_2, \\
&\qquad \text{and } M\big|_{[m+n-k-l+1\ldots m+n+p-k-l]} = M_3 \big\} \\
M_1 \square_k (M_2 \square_l M_3) &= \big\{ M \in \Gamma_{m+n+p-k-l} \mid M\big|_{[1\ldots m]} = M_1 \\
&\qquad \text{and } M\big|_{[m-k+1\ldots m+n+p-k-l]} \in M_2 \square_l M_3 \big\} \\
&= \big\{ M \in \Gamma_{m+n+p-k-l} \mid M\big|_{[1\ldots m]} = M_1, M\big|_{[m-k+1\ldots m+n+p-k-l]}\big|_{[1\ldots n]} = M_2, \\
&\qquad \text{and } M\big|_{[m-k+1\ldots m+n+p-k-l]}\big|_{[n-l+1\ldots n+p-l]} = M_3 \big\} \\
&= \big\{ M \in \Gamma_{m+n+p-k-l} \mid M\big|_{[1\ldots m]} = M_1, M\big|_{[m-k+1\ldots m+n-k]} = M_2, \\
&\qquad \text{and } M\big|_{[m+n-k-l+1\ldots m+n+p-k-l]} = M_3 \big\} \qquad \square
\end{aligned}
$$

Having these properties, we proceed further to defining concatenation and star:

Definition 8.3.6. *Given two $2n$-relation $M_1, M_2 \in \Gamma_{2n}$, their **concatenation** is defined as follows:*

$$M_1 \odot M_2 = \big(M_1 \square_n M_2\big)\big|_{[1\ldots n]\cup[2n+1\ldots 3n]} \tag{8.34}$$

The good properties of concatenation are the following:

Proposition 8.3.7. *1. $2n$-relation concatenation is associative: for each triplet of $2n$-relations $M_1, M_2, M_3 \in \Gamma_{2n}$,*

$$M_1 \odot (M_2 \odot M_3) = (M_1 \odot M_2) \odot M_3 \tag{8.35}$$

2. Each $2n$-relation has a left and a right unit: for each $M \in \Gamma_{2n}$ there exist $1^l_M, 1^r_M \in \Gamma_{2n}$ such that

$$1^l_M \odot M = M \odot 1^r_M = M \tag{8.36}$$

The definitions of the left and right pseudounits are the following:

$$
\begin{aligned}
(1^l_M)_{ij} &= (1^l_M)_{n+i,j} = (1^l_M)_{i,n+j} = (1^l_M)_{n+i,n+j} = M_{ij} \\
(1^r_M)_{ij} &= (1^r_M)_{n+i,j} = (1^r_M)_{i,n+j} = (1^r_M)_{n+i,n+j} = M_{n+i,n+j}
\end{aligned}
$$

Observe that, similarly to all units for the concatenations encountered so far, $(1^l_M)_{ii} = (1^r_M)_{ii} = \text{`}=\text{'}$ for all $i \in [1 \ldots 2n]$.

3. *For each $2n$-relation $M \in \Gamma_{2n}$ there exists a* weak inverse $\tilde{M}$ *with the property that*

$$1^l_M \in M \odot \tilde{M}, \qquad 1^r_M \in \tilde{M} \odot M \tag{8.37}$$

The definition of the weak inverse is the following:

$$\tilde{M}_{ij} = \begin{cases} M_{i+n,j+n} & \textit{iff } i,j \in [1 \dots n] \\ M_{i+n,j-n} & \textit{iff } i \in [1 \dots n], j \in [n+1 \dots 2n] \\ M_{i-n,j+n} & \textit{iff } i \in [n+1 \dots 2n], j \in [1 \dots n] \\ M_{i-n,j-n} & \textit{iff } i,j \in [n+1 \dots 2n] \end{cases} \tag{8.38}$$

Proof. The proof of this proposition is similar to the proof of Proposition 8.2.9. We will only prove the first property, whose proof relies on Proposition 8.3.5:

$$\begin{aligned}
(M_1 \odot M_2) \odot M_3 &= \big((M_1 \odot M_2) \square_n M_3\big)\big|_{[1\dots n]\cup[2n+1\dots 3n]} \\
&= \big((M_1 \square_n M_2)\big|_{[1\dots n]\cup[2n+1\dots 3n]} \square_n M_3\big)\big|_{[1\dots n]\cup[2n+1\dots 3n]} \\
&= \big(M_1 \square_n M_2 \square_n M_3\big)\big|_{[1\dots n]\cup[2n+1\dots 3n]\cup[3n+1\dots 4n]}\big|_{[1\dots n]\cup[2n+1\dots 3n]} \\
&= \big(M_1 \square_n M_2 \square_n M_3\big)\big|_{[1\dots n]\cup[3n+1\dots 4n]} \\
M_1 \odot (M_2 \odot M_3) &= \big((M_1 \square_n (M_2 \odot M_3)\big)\big|_{[1\dots n]\cup[2n+1\dots 3n]} \\
&= \big(M_1 \square_n (M_2 \square_n M_3)\big|_{[1\dots n]\cup[2n+1\dots 3n]}\big)\big|_{[1\dots n]\cup[2n+1\dots 3n]} \\
&= \big(M_1 \square_n M_2 \square_n M_3\big)\big|_{[1\dots n]\cup[n+1\dots 2n]\cup[3n+1\dots 4n]}\big|_{[1\dots n]\cup[2n+1\dots 3n]} \\
&= \big(M_1 \square_n M_2 \square_n M_3\big)\big|_{[1\dots n]\cup[3n+1\dots 4n]}
\end{aligned}$$

□

The powerset $2n$-relations becomes then a monoid with concatenation and with the following unit:

$$\mathbf{1}^{\Gamma}_{2n} = \big\{ M \in \Gamma_{2n} \mid M_{i,n+i} = \text{`} = \text{`} \big\}$$

Let us finally introduce star on *sets* of $2n$-relations: given a set of $2n$-relations $\mathcal{M} \subseteq \Gamma_{2n}$, the **star** of $\mathcal{M}$ is defined as:

$$\mathcal{M}^{\circledast} = \bigcup_{k \geq 0} \mathcal{M}^{k\odot} \tag{8.39}$$

where $\mathcal{M}^{0\odot} = \mathbf{1}^{\Gamma}_{2n}$ and $\mathcal{M}^{(k+1)\odot} = \mathcal{M}^{k\odot} \odot \mathcal{M}$ for each $k \in \mathbb{N}$.

8.3.3 n-word representations

Definition 8.3.8. *A tuple $(W, M) \in \mathcal{WR}_n \times \Gamma_n$ is called a n-**word representation**.*

*The n-region $R \in \mathcal{R}\mathsf{egn}_n$ **represented** by the tuple (W, M) is defined as follows: for all $i, j \in [1 \dots n]$,*

$$R_{ij} = \begin{cases} \{W_{ij}\} & \textit{iff } M_{ij} = \text{`}=\text{`} \\ ]W_{ij}-1, W_{ij}[& \textit{iff } M_{ij} = \text{`}<\text{`} \\ ]W_{ij}, W_{ij}+1[& \textit{iff } M_{ij} = \text{`}>\text{`} \end{cases} \tag{8.40}$$

The region represented by the tuple (W, M) is denoted $[W, M]$. That is, we define a mapping $[\cdot] : \mathcal{WR}_n \times \Gamma_n \to \mathcal{R}\mathsf{egn}_n$, called *representation*, which associates to each n-word representation the region which is represented by it.

The above definition should be incorrect unless we prove the following:

Proposition 8.3.9. *For each $(W, M) \in \mathcal{WR}_n \times \Gamma_n$, $[W, M]$ has a nonempty semantics.*

Proof. The proof idea is to check that $[W, M]$ is in normal form. By Remark 8.3.3, we have to check four (representative) cases on M. We check here only two, the (‘ = ‘,‘ = ‘,‘ = ‘) case and the (‘ < ‘,‘ < ‘,‘ > ‘) case, the proof in the other cases being similar.

1. Take some $i, j, k \in [1 \ldots n]$ and suppose that $M_{ij} =$ ‘ = ‘, $M_{jk} =$ ‘ = ‘ and $M_{ki} =$ ‘ = ‘. Then, by the triangle identity

$$[W, M]_{ij} + [W, M]_{jk} = \{W_{ij}\} + \{W_{jk}\} = \{W_{ij} + W_{jk}\} = \{W_{ik}\} = [W, M]_{ik}$$

2. Take some $i, j, k \in [1 \ldots n]$ and suppose that $M_{ij} =$ ‘ < ‘, $M_{jk} =$ ‘ < ‘ and $M_{ki} =$ ‘ < ‘. Then, again by the triangle identity

$$\begin{aligned} [W, M]_{ij} + [W, M]_{jk} &=]W_{ij}-1, W_{ij}[+]W_{jk}-1, W_{jk}[=]W_{ij}+W_{jk}-2, W_{ij}+W_{jk}[\\ &=]W_{ik}-2, W_{ik}[\qquad \text{while} \\ [W, M]_{ik} &=]W_{ik}-1, W_{ik}[\end{aligned}$$

because, by construction, $[W, M]_{ki} =]W_{ki}, W_{ki}+1[$ and $W_{ki} = -W_{ik}$. Hence, we get that $[W, M]_{ik} \subseteq [W, M]_{ij} + [W, M]_{jk}$ in this case too. □

The mapping $[\cdot]$ defines an equivalence relation on $\mathcal{WR}_n \times \Gamma_n$ (the *kernel* of $[\cdot]$, in category theoretic terms) denoted in the sequel $\equiv_n$ and defined as follows:

$$(W, M) \equiv_n (W', M') \text{ iff } [W, M] = [W', M'] \tag{8.41}$$

The following proposition can be used to prove that each n-region has at least one n-word representation:

Proposition 8.3.10. *Given some n-word representation $(W, M) \in \mathcal{WR}_n \times \Gamma_n$, suppose that there exists some region $R \in \mathcal{R}\mathsf{egn}_{n+1}$ such that $R|_{[1\ldots n]} = [W, M]$. Then there exists a $(n+1)$-word representation of R, $(W', M') \in \mathcal{WR}_{n+1} \times \Gamma_{n+1}$, such that $(W', M')|_{[1\ldots n]} = (W, M)$.*

Proof. We first “close” each open interval in R, that is, compute the DBM $\overline{R} \in \mathcal{D}\mathsf{bm}_{n+1}$ with

$$\overline{R}_{ij} = \begin{cases} [\alpha, \alpha+1] & \text{iff } R_{ij} =]\alpha, \alpha+1[\\ \{\alpha\} & \text{iff } R_{ij} = \{\alpha\} \end{cases}$$

It is easy to see that $\overline{R}$ is still in normal form since the triangle inclusion is preserved by taking closures of the sets involved in it.

We now simulate in part the proof of Proposition 8.1.5, namely choose a set of integers α_i such that $\alpha_j - \alpha_i = W_{ij}$ (for simplicity we put $\alpha_i = W_{1i}$) and then prove that $\bigcup_{i=1}^{n} \left(\alpha_i + R_{i,n+1}\right) \neq \emptyset$. Then we observe that this intersection is a closed unit length interval with integer bounds, since each factor of the intersection is. Thence we may choose one of the bounds of this interval, denote it K, and build from it a $(n+1)$-word $W' \in \mathcal{WD}_{n+1}$ which extends W:

$$\begin{aligned} W'_{ij} &= W_{ij} \text{ for all } i,j \in [1 \dots n] \\ W'_{i,n+1} &= W_{i1} + K \text{ for all } i \in [1 \dots n] \\ W'_{n+1,i} &= -W_{i,n+1} \text{ for all } i \in [1 \dots n] \end{aligned}$$

Similarly to the proof of Proposition 8.1.5 we get that W' is a $(n+1)$-word (in fact a $(n+1)$-signal) which belongs to $\|\overline{R}\|$. We now need to choose some relation symbols that extend M to some $(n+1)$-relation M' such that $[W', M'] = R$. This extension is done as follows:

$$M'_{i,n+1} = \begin{cases} \text{`} = \text{`} & \text{iff } R_{i,n+1} = \{W_{i,n+1}\} \\ \text{`} < \text{`} & \text{iff } R_{i,n+1} =]W_{i,n+1} - 1, W_{i,n+1}[\\ \text{`} > \text{`} & \text{iff } R_{i,n+1} =]W_{i,n+1}, W_{i,n+1} + 1[\end{cases}$$

and of course $M'\big|_{[1\dots n]} = M$.

Let us show that this is a $(n+1)$-relation, i.e., that it is consistent. Since $M'\big|_{[1\dots n]} = M$ we only need to check cycles of length 3 in which $n+1$ participates.

Suppose such a cycle is inconsistent, say $M_{ij} = \text{`} < \text{`}, M_{j,n+1} = \text{`} < \text{`}$ and $M_{n+1,i} = \text{`} < \text{`}$. But then $R_{ij} =]W_{ij} - 1, W_{ij}[$, $R_{j,n+1} =]W_{j,n+1} - 1, W_{j,n+1}[$, $R_{n+1,i} =]W_{n+1,i} - 1, W_{n+1,i}[$. It follows that

$$R_{ij} + R_{j,n+1} + R_{n+1,i} =]W_{ij} + W_{j,n+1} + W_{n+1,i} - 3, W_{ij} + W_{j,n+1} + W_{n+1,i}[=]-3, 0[$$

which obviously contradicts the assumption that R has a nonempty semantics, that is, the Identity 8.7 The other cases of inconsistent cycles are treated similarly.

Hence M' is indeed a $(n+1)$-relation. As a consequence, (W', M') is one of the $(n+1)$-word representations for R. □

The following corollary says that representation of sets of n-regions with the aid of n-word representations is complete:

Corollary 8.3.11. *For each region $R \in \mathcal{R}\mathsf{egn}_n$ there exists an n-word representation of it.*

Proof. We only have to apply Proposition 8.3.10 iteratively, starting with a representation of $R\big|_{\{1\}}$, that is, with $[0, \text{`} = \text{`}]$. □

8.3.4 Operations on n-word representations

Our quest on providing representations of EDBM demands now to extend projection, juxtaposition, concatenation and positive star from n-words and n-relations to n-word representations. The extensions are then, as expected, the following:

$$(W, M)\big|_X = (W\big|_X, M\big|_X) \tag{8.42}$$

$$(W, M)\Box_p(W', M') = \big\{W\Box_p W', M'') \mid M'' \in M\Box_p M'\big\} \tag{8.43}$$

$$(W, M) \odot (W', M') = \big((W, M)\Box_n(W', M')\big)\big|_{[1\ldots n]\cup[2n+1\ldots 3n]} \tag{8.44}$$

$$\mathcal{W}^{\circledast} = \bigcup_{k\geq 0} \mathcal{W}^{k\odot} \tag{8.45}$$

where $\mathcal{W} \subseteq \mathcal{WR}_{2n} \times \Gamma_{2n}$.

The main concern is then to show that the n-word representation operations correctly simulate the n-signal/region operations, that is, to show they are compositional, because this would assure us of the usefulness of n-word representations. The following proposition paves the way of proving this compositionality:

Proposition 8.3.12. *1. For each* $(W, M) \in \mathcal{WR}_n \times \Gamma_n$*,*

$$[W\big|_X, M\big|_X] = [W, M]\big|_X \tag{8.46}$$

2. For each m*-word representation* $(W_1, M_1) \in \mathcal{WR}_m \times \Gamma_m$*, each* n*-word representation* $(W_2, M_2) \in \mathcal{WR}_n \times \Gamma_n$ *and each integer* $p \leq \min(m, n)$*,*

$$[W_1, M_1]\Box_p[W_2, M_2] = \big\{[W_1'\Box_p W_2', M] \mid (W_1', M_1') \equiv (W_1, M_1), \\ (W_2', M_2') \equiv (W_2, M_2) \text{ and } M \in M_1'\Box_p M_2'\big\} \tag{8.47}$$

3. For each pair of $2n$*-word representations* $(W_1, M_2), (W_2, M_2) \in \mathcal{WR}_{2n} \times \Gamma_{2n}$*,*

$$[W_1, M_1] \odot [W_2, M_2] = \big\{[W_1' \odot W_2', M] \mid \exists(W_1', M_1') \equiv (W_1, M_1), \\ \exists(W_2', M_2') \equiv (W_2, M_2) \text{ and } M \in M_1' \odot M_2'\big\} \tag{8.48}$$

Proof. The first property follows by easy verification and we skip its proof.

For the second property, the inverse inclusion is straightforward: given any $(m + n - p)$-region $R \in \big\{[W_1'\Box_p W_2', M] \mid (W_1', M_1') \equiv (W_1, M_1), (W_2', M_2') \equiv (W_2, M_2)$ and $M \in M_1'\Box_p M_2'\big\}$, we have, by the first property 8.46, that $R\big|_{[1\ldots m]} = [W_1, M_1]$ and $R\big|_{[m-p+1\ldots m+n-p]} = [W_2, M_2]$. But this implies that $R \in [W_1, M_1]\Box_p[W_2, M_2]$ by definition of $\Box_p$ on regions.

For the left-to-right inclusion we rely upon Proposition 8.3.10 in the following way: take some $R \in [W_1, M_1]\Box_p[W_2, M_2]$. Take some representation of $R\big|_{[m-p+1\ldots m]}$, say $[W_3, M_3] = R\big|_{[m-p+1\ldots m]}$. Then, using Proposition 8.3.10, extend this representation to two other representations: one equivalent to (W_1, M_1) and the other equivalent to (W_2, M_2). Denote these two representations as $(\overline{W}_1, \overline{M}_1)$, respectively $(\overline{W}_2, \overline{M}_2)$.

Observe that, since both these representations are extensions of (W_3, M_3) we have that

$$\overline{W}_1\big|_{[m-p+1\ldots m]} = W_3 = \overline{W}_2\big|_{[1\ldots p]}$$
$$\overline{M}_1\big|_{[m-p+1\ldots m]} = M_3 = \overline{M}_2\big|_{[1\ldots p]}$$

But thence $\overline{W}_1 \Box_p \overline{W}_2$ is defined, and it remains to choose an extension $\overline{M}$ of $\overline{M}_1$ and $\overline{M}_2$ such that $[\overline{W}_1 \Box_p \overline{W}_2, \overline{M}] = R$. This extension is the following: for each $i \in [1 \ldots m-p]$ and $j \in [m+1 \ldots m+n-p]$,

$$\overline{M}_{ij} = \begin{cases} \text{`}=\text{'} & \text{iff } R_{ij} = \big\{(\overline{W}_1 \Box_p \overline{W}_2)_{ij}\big\} \\ \text{`}<\text{'} & \text{iff } R_{ij} = \big((\overline{W}_1 \Box_p \overline{W}_2)_{ij} - 1, (\overline{W}_1 \Box_p \overline{W}_2)_{ij}\big) \\ \text{`}>\text{'} & \text{iff } R_{ij} = \big((\overline{W}_1 \Box_p \overline{W}_2)_{ij}, (\overline{W}_1 \Box_p \overline{W}_2)_{ij} + 1\big) \end{cases} \tag{8.49}$$

Of course, $\overline{M}\big|_{[1\ldots m]} = \overline{M}_1$ and $\overline{M}\big|_{[m-p+1\ldots m+n-p]} = \overline{M}_2$.

It is routine to check that only these three conditions on R_{ij} really hold and that $\overline{M}$ is consistent. And, by construction, we have $[\overline{W}_1 \Box_p \overline{W}_2, \overline{M}] = R$.

The last two properties are easy corollaries of the property 8.47. □

Remark 8.3.13. Observe that the identity

$$[W_1, M_1] \Box_p [W_2, M_2] = \big\{[W_1 \Box_p W_2, M] \mid M \in M_1 \Box_p M_2\big\}$$

is not valid in general since it might be possible that $[W_1, M_1]\big|_{[n+1\ldots 2n]} = [W_2, M_2]\big|_{[1\ldots n]}$ but $W_1\big|_{[n+1\ldots 2n]} \neq W_2\big|_{[1\ldots n]}$ just because the same region might have different representations.

An example of this mismatch is provided in the introductory part of this chapter.

This observation raises the problem whether we may correctly represent region concatenation with $2n$-word representation concatenation. The idea that helps us overcome this problem is that, for each pair of $2n$-regions which correctly concatenate, there must exist a pair of $2n$-word representations which correctly concatenate, and hence represents the concatenation of the two given regions. In other words, we will be interested in composing *sets of* $2n$*-word representations* which bear the property that *all* the $2n$-word representations associated with a certain region are in the set. The formalization of this idea is the following notion of *convexity*:

Definition 8.3.14. *A set of n-word representations $\mathcal{N} \subseteq \mathcal{WR}_n \times \Gamma_n$ is called* ***convex*** *if it is saturated by the equivalence relation $\equiv_n$, that is, if the following property holds:*

For each set of n-word representations $\mathcal{W} \subseteq \mathcal{WR}_n \times \Gamma_n$ and each n-word representation $(W, M) \in \mathcal{N}$, if $[W, M] = [W', M']$ for some $(W', R') \in \mathcal{WR}_n \times \Gamma_n$ then also $(W', M') \in \mathcal{N}$.

In the sequel, for each set of n-word representations $\mathcal{W} \subseteq \mathcal{WR}_n \times \Gamma_n$, we denote $[\mathcal{W}]$ as the set of regions which are represented by some element of $\mathcal{W}$:

$$[\mathcal{W}] = \big\{[W, M] \mid (W, M) \in \mathcal{W}\big\}$$

Proposition 8.3.15. *1. For each convex set of n-word representations $\mathcal{W} \subseteq \mathcal{WR}_n \times \Gamma_n$ and each $X \subseteq [1 \ldots n]$, $\mathcal{W}\big|_X$ is also a convex set of $card(X)$-word representations.*

2. For each two convex sets of word representations $\mathcal{W}_1 \subseteq \mathcal{WR}_m \times \Gamma_m$, $\mathcal{W}_2 \subseteq \mathcal{WR}_n \times \Gamma_n$, and integer $p \leq \min(m, n)$, $\mathcal{W}_1 \square_p \mathcal{W}_2$ is a convex set of $(m + n - p)$-word representations and

$$[\mathcal{W}_1 \square_p \mathcal{W}_2] = [\mathcal{W}_1] \square_p [\mathcal{W}_2] \tag{8.50}$$

3. For each two convex sets of $2n$-word representations $\mathcal{W}_1, \mathcal{W}_2 \subseteq \mathcal{WR}_{2n} \times \Gamma_{2n}$, $\mathcal{W}_1 \odot \mathcal{W}_2$ is a convex set and

$$[\mathcal{W}_1 \odot \mathcal{W}_2] = [\mathcal{W}_1] \odot [\mathcal{W}_2]$$

4. For each convex set of $2n$-word representations $\mathcal{W} \subseteq \mathcal{WR}_{2n} \times \Gamma_{2n}$, $\mathcal{W}^{\circledast}$ is convex and

$$[\mathcal{W}^{\circledast}] = [\mathcal{W}]^{\circledast}$$

Proof. All the properties rely on Proposition 8.3.10. For the first property, observe that, if $(W, M) \in \mathcal{W}$ and $[W', M'] = [W, M]$ then there must exist $R \in [\mathcal{W}]$ such that $R\big|_X = [W, M] = [W', M']$. But then we may recursively apply Proposition 8.3.10 to extend (W', M') to a n-region (W'', M'') that is, with $(W'', M'')\big|_X = (W', M')$, such that $[W'', M''] = R$. But since $R \in [\mathcal{W}]$, by convexity of $\mathcal{W}$ it follows that $(W'', M'') \in \mathcal{W}$. Fact which implies that $(W', M') \in \mathcal{W}\big|_X$.

For the second property, observe that identity 8.47 from Proposition 8.3.12 gives the left-to-right inclusion:

$$\begin{aligned}
[\mathcal{W}_1 \square_p \mathcal{W}_2] &= \bigcup \big\{ [W_1 \square_p W_2, M] \mid \exists M_1 \in \Gamma_m, M_2 \in \Gamma_n \text{ such that } (W_1, M_1) \in \mathcal{W}_1, \\
&\qquad (W_2, M_2) \in \mathcal{W}_2 \text{ and } M\big|_{[1\ldots m]} = M_1, M\big|_{[m-p+1\ldots m+n-p]} = M_2 \big\} \\
&\subseteq \bigcup \big\{ [W_1' \square_p W_2', M] \mid \exists (W_1, M_1) \in \mathcal{W}_1, (W_2, M_2) \in \mathcal{W}_2 \text{ with} \\
&\qquad (W_1', M_1') \equiv (W_1, M_1), (W_2', M_2') \equiv (W_2, M_2) \text{ and } M \in M_1 \square_p M_2 \big\} \\
&= \bigcup \big\{ [W_1, M_1] \square_p [W_2, M_2] \mid (W_1, M_1) \in \mathcal{W}_1, (W_2, M_2) \in \mathcal{W}_2 \big\} \\
&= [\mathcal{W}_1] \square_p [\mathcal{W}_2]
\end{aligned}$$

For the reverse identity, suppose we have some region $R \in [\mathcal{W}_1] \square_p [\mathcal{W}_2]$. Hence $R\big|_{[1\ldots m]} \in [\mathcal{W}_1]$ and $R\big|_{[m-p+1\ldots m+n-p]} \in [\mathcal{W}_2]$. It follows that there exist $(W_1, M_1) \in \mathcal{W}_1$ with $R\big|_{[1\ldots m]} = [W_1, M_1]$.

Consider now $(W_1, M_1)\big|_{[m-p+1\ldots m]}$. We have that

$$\big[(W_1, M_1)\big|_{[m-p+1\ldots m]}\big] = R\big|_{[m-p+1\ldots m]}$$

hence, by Proposition 8.3.10 we may extend this p-word representation to an n-word representation that represents $R\big|_{[m-p+1\ldots m+n-p]}$, say

$$(W_3, M_3)\big|_{[1\ldots p]} = (W_1, M_1)\big|_{[m-p+1\ldots m]} \text{ and} \tag{8.51}$$

$$[W_3, M_3] = R\big|_{[m-p+1\ldots m+n-p]} \tag{8.52}$$

Observe that Identity 8.52 implies that $[W_3, M_3] \in [\mathcal{W}_2]$, hence, by convexity of $\mathcal{W}_2$ it follows that $(W_3, M_3) \in \mathcal{W}_2$. On the other hand, Identity 8.51 says that W_1 and W_3 can be p-juxtaposed and so can M_1 and M_3. Hence $(W_1 \square_p W_3, M_1 \square_p M_3)$ is nonempty and $[W_1 \square_p W_3, M_1 \square_p M_3] \subseteq [\mathcal{W}_1 \square_p \mathcal{W}_2]$.

It remains just to pick some $(m+n-p)$-relation $M \in M_1 \square_p M_3$ such that $R \in [W_1 \square_p W_3, M]$, and the choice is the same as for the M defined in 8.49 in the proof of Proposition 8.3.12. Also it is easy to observe that the convexity of both $\mathcal{W}_1$ and $\mathcal{W}_2$ implies the convexity of $\mathcal{W}_1 \square_p \mathcal{W}_2$.

The proof of the last two properties is a straightforward corollary of the first two. □

8.4 n-region automata

Definition 8.4.1. *An n-**region automaton** is a tuple $\mathcal{A} = (Q, \delta, Q_1, \ldots Q_n, \nu)$ in which all but the last components form an n-automaton $\mathcal{A}^\flat = (Q, \delta, Q_1, \ldots, Q_n)$ over the one-letter alphabet $\Sigma = \{1\}$ while $\nu : Q \to \Gamma_n$ is the* n-relation *labeling function, associating a n-relation to each state.*

The n-automaton $\mathcal{A}^\flat = (Q, \delta, Q_1, \ldots, Q_n)$ is called the *underlying* n-automaton of $\mathcal{A}$.

n-region automata are intended to represent EDBMs by means of n-word representations: a **run** in an n-region automaton is a sequence of transitions in δ which match on intermediary states, with the additional property that *all states in the run are labeled with the same n-relation*. Because only one symbol can label any transition, we will represent each run as the sequence of states in the run, $\rho = (q_i)_{i \in [1 \ldots k]}$ and denote $\nu(\rho)$ the n-relation which identically labels all the states q_i in ρ.

A run $\rho = (q_j)_{j \in [1 \ldots k]}$ is **accepting** if it passes through each accepting set, that is, if for each $i \in [1 \ldots n]$ there exists some $j \in [1 \ldots k]$ such that $q_j \in Q_i$. Given an-word representation $(W, M) \in \mathcal{WR}_n \times \Gamma_n$, a run $\rho = (q_j)_{j \in [1 \ldots k]}$ and a sequence of indices of states in the run $\boldsymbol{l} = (l_i)_{i \in [1 \ldots n]}$ with the property that $q_{l_i} \in Q_i$, we say that the n-word representation (W, M) is **accepted** by $(\rho, \boldsymbol{l})$ if W is accepted by the underlying n-automaton $\mathcal{A}^\flat$ and $M = \nu(\rho)$. Similarly to n-automata, we call the sequence $\boldsymbol{l}$ as the sequence **witnessing** the acceptance of (W, M) by ρ.

Remark 8.4.2. Since in each n-region automaton we are interested only in runs in which states are labeled with the same n-relation, we will consider only n-region automata in which the transition function is consistent with the n-relation labeling ν, that is, in which whenever $q \xrightarrow{a} r$ for some $a \in \Sigma$ then $\nu(q) = \nu(r)$.

To each n-region automaton we will associate three languages:

- The n-**word representation language accepted by** $\mathcal{A}$, denoted $L_{rep}(\mathcal{A})$, consists of the n-word representations accepted by some tuple $(\rho, \boldsymbol{l})$ as above.
- The **region language** of $\mathcal{A}$ is the set of regions which are represented by some n-word representation in $L_{rep}(\mathcal{A})$, and is denoted $L_{rgn}(\mathcal{A})$:

$$L_{rgn}(\mathcal{A}) = \big\{ [W, M] \mid (W, M) \in L_{rep}(\mathcal{A}) \big\} \tag{8.53}$$

- Finally, the n-**signal language** of $\mathcal{A}$ is the union of the semantics of the regions in the region language of $\mathcal{A}$,

$$L_{sig}(\mathcal{A}) = \bigcup \{\|R\| \mid R \in L_{rgn}(\mathcal{A})\}$$

Remark 8.4.3. Observe that, for each two n-region automata $\mathcal{A}$ and $\mathcal{B}$, $L_{rgn}(\mathcal{A}) = L_{rgn}(\mathcal{B})$ iff $L_{sig}(\mathcal{A}) = L_{sig}(\mathcal{B})$, but it might be possible that $L_{rgn}(\mathcal{A}) = L_{rgn}(\mathcal{B})$ and $L_{rep}(\mathcal{A}) \neq L_{rep}(\mathcal{B})$, due to the possibility to represent the same n-region by different n-word representations. But if the two n-word representation languages are convex, then we also have $L_{rgn}(\mathcal{A}) = L_{rgn}(\mathcal{B})$ iff $L_{rep}(\mathcal{A}) = L_{rep}(\mathcal{B})$.

Definition 8.4.4. *An n-region automaton is called **convex** if its n-word representation language is convex.*

With this definition, the following chain of equivalences is valid for convex n-region automata:

$$L_{rep}(\mathcal{A}) = L_{rep}(\mathcal{B}) \text{ iff } L_{rgn}(\mathcal{A}) = L_{rgn}(\mathcal{B}) \text{ iff } L_{sig}(\mathcal{A}) = L_{sig}(\mathcal{B}) \tag{8.54}$$

Hence, when we will need to prove the equality of the languages of two convex n-region automata we will only need to prove the equality of their n-word representation language.

8.4.1 Basic closure properties for n-region automaton

Throughout this section we prove that the operations on n (or $2n$)-automata can be extended to operations on n-region automata. This subsection is just a restatement for EDBMs and n-region automata of the results contained in Chapter 7. We start by the translation of Proposition 7.2.10:

Proposition 8.4.5. *The class of n-signal languages accepted by n-region automata is closed under union and intersection. Moreover, if $\mathcal{A}$ and $\mathcal{B}$ are two convex n-region automata, then one can build convex n-region automaton for $L_{sig}(\mathcal{A}) \cup L_{sig}(\mathcal{B})$ and $L_{sig}(\mathcal{A}) \cap L_{sig}(\mathcal{B})$.*

Proof. The constructions are straightforward adaptations from Proposition 7.2.10. The convexity property follows due to the fact that intersection and union of saturated sets give saturated sets. □

Theorem 8.4.6. *The class of n-signal languages which are the n-signal language of some n-region automaton equals the class of n-signal languages which are the semantics of a sum of n-EDBMs.*

Note that the result refers to *extended DBMs*. It is clear that, in general, n-region automata are more expressive than sums of mere DBMs.

Proof. The proof of the direct inclusion is very similar with the proof of Theorem 7.2.14.

Consider all tuples of accepting states $(q_1, \ldots, q_n)$ with $q_i \in Q_i$ and such that all the states in the tuple are labeled with the same n-relation M. For each such tuple and n-relation M, we construct the n-region automaton in which only the states labeled with M are present and in which all Q_is are singleton sets $Q_i = \{q_i\}$. Denote this reduced automaton $\mathcal{A}(q_1, \ldots, q_n, M)$.

Then, for each $i, j \in [1 \dots n]$ we build the regular expression E^+_{ij} which denotes the set of positive integers that are the length of a path from q_i to q_j in $\mathcal{A}(q_1, \dots, q_n, M)$. Since we speak about positive integers, we may put E^+_{ij} in the form

$$E^+_{ij} = A \cup (B + \{c\}^*)$$

where A, B are finite sets of integers[4] and $c \in \mathbb{N}$.

Then, from E^+_{ij} we build a regular expression *over intervals* – that is, an n-EDBM – as follows:

1. If $M_{ij} =$ ' $=$ ' then clearly
$$D^+_{ij} = E^+_{ij} = A \cup (B + \{c\}^*).$$
2. If $M_{ij} =$ ' $<$ ' then observe that, intuitively, each $\alpha \in A$ is an *upper bound* for a n-region which is accepted by $\mathcal{A}$ along a path from q_i to q_j. Hence we put
$$D^+_{ij} = \bigcup \{\,]\alpha{-}1, \alpha[\mid \alpha \in A \} \cup \Big(\bigcup \{\,]\beta{-}1, \beta[\mid \beta \in B \} + \{c\}^* \Big)$$
3. If $M_{ij} =$ ' $>$ ' then we put
$$D^+_{ij} = \bigcup \{\,]\alpha, \alpha{+}1[\mid \alpha \in A \} \cup \Big(\bigcup \{\,]\beta, \beta{+}1[\mid \beta \in B \} + \{c\}^* \Big)$$

Finally, from all these regular expressions over nonnegative intervals we build the n-EDBM $D \in \mathcal{E}\mathsf{dbm}_n$ defined by:

$$D_{ij} = D^+_{ij} + (-D^+_{ji})$$

where $-D$ denotes the regular expression over real intervals which results by changing every bound into its opposite; for example, for the case 3 above,

$$-D^+_{ji} = \bigcup \{\,]{-}\alpha{-}1, -\alpha[\mid \alpha \in A \} \cup \Big(\bigcup \{\,]{-}\beta{-}1, -\beta[\mid \beta \in B \} + \{-c\}^* \Big)$$

It is then easy to check that the semantics of the sum of all n-EDBMs built for each tuple $(q_1, \dots, q_n)$ and each n-relation M equals the n-signal language of $\mathcal{A}$.

The reverse inclusion can be proved by induction on n as follows: in the base case, we code each regular expression over real intervals into a 2-region automaton. The idea is to decompose each regular expression over intervals

$$R = A \cup (B + \{c\}^*) \tag{8.55}$$

into a union of two regular expressions, one containing only point intervals and the other containing only open intervals of unit length. Hence the basic case reduces to the following constructions:

[4] Remind that we use ' $\cup$ ' for denoting union and ' $+$ ' for denoting concatenation for regular expressions over intervals.

1. Suppose that $A, B \subseteq \mathbb{N}$. Denote $m_A = \max A$ and $m_B = \max B$. Then the 2-region automaton equivalent to R is:

$$\begin{aligned}
\mathcal{C} &= \big(Q, \delta, Q_1, Q_2, \nu\big) \text{ where} \\
Q &= ([0 \ldots m_A] \times \{1\}) \cup ([0 \ldots m_B] \times \{2\}) \cup ([1 \ldots c] \times \{3\}) \\
\delta &= \big\{((\alpha-1,1),(\alpha,1)) \mid \alpha \in [1 \ldots m_A]\big\} \cup \big\{((\alpha-1,2),(\alpha,2)) \mid \alpha \in [1 \ldots m_B]\big\} \cup \\
&\quad \big\{((\alpha-1,3),(\alpha,3)) \mid \alpha \in [1 \ldots c]\big\} \cup \big\{((c,3),(1,3))\big\} \cup \big\{((b,2),(1,3)) \mid b \in B\big\} \\
Q_1 &= \{(0,1),(0,2)\} \\
Q_2 &= (A \times \{1\}) \cup (B \times \{2\}) \cup \{(c,3)\} \\
\nu(q) &= \text{`} = \text{`} \text{ for all } q \in Q
\end{aligned}$$

   Observe that $\mathcal{C}$ is a convex 2-region automaton.
2. Suppose there exist two strictly increasing sequences of integers $(\alpha_i)_{i \in [1 \ldots card(A)]}$ and $(\beta_i)_{i \in [1 \ldots card(B)]}$ such that:

$$A = \big\{\,]\alpha_i, \alpha_i + 1[\mid i \in card(A)\big\} \text{ and } B = \big\{\,]\beta_i, \beta_i + 1[\mid i \in card(B)\big\}$$

   Then the 2-region automaton equivalent to R is

$$\begin{aligned}
\mathcal{C} &= \big(Q, \delta, Q_1, Q_2, \nu\big) \text{ where} \\
Q &= \big([0 \ldots \alpha_{card(A)} + 1] \times \{1\} \times \{\text{`} <\text{`}, \text{`} >\text{`}\}\big) \cup \\
&\quad \big([0 \ldots \beta_{card(B)} + 1] \times \{2\} \times \{\text{`} <\text{`}, \text{`} >\text{`}\}\big) \cup \\
&\quad \big([1 \ldots c] \times \{3\} \times \{\text{`} <\text{`}, \text{`} >\text{`}\}\big) \\
\delta &= \big\{((x-1,1,s),(x,1,s)) \mid x \in [1 \ldots \alpha_{card(A)} + 1], s \in \{\text{`} <\text{`}, \text{`} >\text{`}\}\big\} \cup \\
&\quad \big\{((x-1,2,s),(x,2,s)) \mid x \in [1 \ldots \beta_{card(B)} + 1], s \in \{\text{`} <\text{`}, \text{`} >\text{`}\}\big\} \cup \\
&\quad \big\{((x-1,3,s),(x,3,s)) \mid x \in [1 \ldots c], s \in \{\text{`} <\text{`}, \text{`} >\text{`}\}\big\} \cup \\
&\quad \big\{((\beta_i, 2, \text{`} >\text{`}), (1, 3, \text{`} >\text{`})) \mid i \in [1 \ldots card(B)]\big\} \cup \\
&\quad \big\{((\beta_i + 1, 2, \text{`} <\text{`}), (1, 3, \text{`} <\text{`})) \mid i \in [1 \ldots card(B)]\big\} \\
Q_1 &= \{(0,1,s),(0,2,s) \mid s \in \{\text{`} <\text{`}, \text{`} >\text{`}\}\} \\
Q_2 &= \big\{(\alpha_i, 1, \text{`} >\text{`}), (\alpha_i + 1, 1, \text{`} <\text{`}) \mid i \in [1 \ldots card(A)]\big\} \cup \\
&\quad \big\{(\beta_i, 1, \text{`} >\text{`}), (\beta_i + 1, 1, \text{`} <\text{`}) \mid i \in [1 \ldots card(B)]\big\} \cup \\
&\quad (\{c\} \times \{3\} \times \{\text{`} <\text{`}, \text{`} >\text{`}\}) \\
\nu(k, l, \text{`} <\text{`}) &= \text{`} <\text{`} \text{ and } \nu(k, l, \text{`} >\text{`}) = \text{`} >\text{`} \text{ for all } k, l
\end{aligned}$$

   Observe again that $\mathcal{C}$ is convex.

For the induction step we rely upon the following property:

Proposition 8.4.7. *Given an n-region automaton $\mathcal{A}$, there exists an $(n+1)$-region automaton with the property that*

$$L_{rgn}(\mathcal{B}) = \big\{ R \in \mathcal{R}\mathsf{egn}_{(n+1)} \mid R\big|_{[1\ldots n]} \in L_{rgn}(\mathcal{A}) \big\} \tag{8.56}$$

or, equivalently, by Proposition 8.1.5, $L_{sig}(\mathcal{B}) = \big\{ \sigma \in \mathsf{Sig}_{n+1}(\{a\}) \mid \sigma\big|_{[1\ldots n]} \in L_{sig}(\mathcal{A}) \big\}$.

Moreover, if $\mathcal{A}$ is convex, then $\mathcal{B}$ can be chosen to be convex too.

Proof. Denote the given n-region automaton $\mathcal{A} = (Q, \delta, Q_1, \ldots, Q_n, \nu)$. We first build an extension of the automaton, by appending states similar to q_* and q_{**} from Definition 7.2.9 of the completion of an n-automaton. We will actually append $2 \cdot card(\Gamma_n)$ states, namely the union $Q_* \cup Q_{**}$ where

$$Q_* = \big\{ (q_*, M) \mid M \in \Gamma_n \big\} \text{ and } Q_{**} \big\{ (q_{**}, M) \mid M \in \Gamma_n \big\}$$

due to the need to have a state q_* and a state q_{**} labeled with each n-relation. These states will be connected to the others as follows:

$$\begin{aligned} \hat{\delta} = \delta \cup \big\{ ((q_*, M), (q_*, M)), ((q_{**}, M), (q_{**}, M)) \mid M \in \Gamma_n \big\} \cup \\ \big\{ ((q_*, M), q'), (q', (q_{**}, M)) \mid M \in \Gamma_n \text{ and there exists } i \in [1 \ldots n] \text{ such that } q' \in Q_i \big\} \end{aligned}$$

Thus we get an n-region automaton, which we call the *completion* of $\mathcal{A}$ and denote $\hat{\mathcal{A}}$. In this automaton, any accepting run can be extended to a run that starts in Q_* and end in Q_{**}.

We further transform this automaton into an $(n+1)$-region automaton by putting any state in the $(n+1)$-th accepting set and by augmenting all n-relation labels to $(n+1)$-relation labels. The resulting automaton is

$$\begin{aligned} \mathcal{B} &= (Q', \delta', Q'_1, \ldots, Q'_n, Q'_{n+1}, \nu') \text{ with } Q' = Q \cup Q_* \cup Q_{**} \text{ and} \\ Q' &= \big\{ (q, M) \mid q \in Q, M \in \Gamma_{n+1} \text{ and } M\big|_{[1\ldots n]} = \nu(q) \big\} \\ \delta' &= \big\{ ((q, M), (q', M)) \mid (q, q') \in \hat{\delta} \big\} \\ Q'_i &= \big\{ (q, M) \in Q' \mid q \in Q_i \big\} \text{ for } i \in [1 \ldots n] \\ Q'_{n+1} &= Q' \\ \nu(q, M) &= M \text{ for all } (q, M) \in Q' \end{aligned}$$

By a straightforward adaptation of the proof for Proposition 7.2.12 we get that

$$L_{rep}(\mathcal{B}) = \big\{ (W, M) \in \mathcal{W}\mathbb{R}_{n+1} \times \Gamma_{n+1} \mid (W\big|_{[1\ldots n]}, M\big|_{[1\ldots n]}) \in L_{rep}(\mathcal{A}) \big\} \tag{8.57}$$

Observe that this property is equivalent to the fact that $L_{rep}(\mathcal{B})\big|_{[1\ldots n]} = L_{rep}(\mathcal{A})$.

The convexity of $\mathcal{B}$ follows easily from the above property: suppose that $[W, M] = [W', M']$ and $[W, M] \in L(\mathcal{B})$. By assumption, we then have

$$[W\big|_{[1\ldots n]}, M\big|_{[1\ldots n]}] = [W, M]\big|_{[1\ldots n]} = [W', M']\big|_{[1\ldots n]} = [W'\big|_{[1\ldots n]}, M'\big|_{[1\ldots n]}]$$

Further, $[W, M] \in L(\mathcal{B})$ implies that $[W\big|_{[1\ldots n]}, M\big|_{[1\ldots n]}] \in L_{rgn}(\mathcal{A})$. From these and from the convexity of $\mathcal{A}$ we get that $(W'\big|_{[1\ldots n]}, M'\big|_{[1\ldots n]}) \in L_{rep}(\mathcal{A})$, which, by identity 8.57, is equivalent to $(W', M') \in L_{rep}(\mathcal{B})$. □

Proof (of Theorem 8.4.6, continued). For each EDBM $R \in \mathcal{E}\mathsf{dbm}_n$ and each $1 \leq i < j \leq n$, we construct the 2-region automaton equivalent to R_{ij}. Then we extend this automaton to an n-region automaton, by recursively applying Proposition 8.4.7. Finally, we intersect the $n(n-1)/2$ automata to get the n-region automaton equivalent to R.

If we are given a finite sum of EDBMs, we utilize the above construction for each term of the sum, and then apply the union construction from Proposition 8.4.5. □

8.4.2 Non-elasticity for $2n$-DBMs

To further adapt the results on concatenation and star closure from Chapter 7, we need to transport non-elasticity for $2n$-regsignals/$2n$-regions/$2n$-word representations and to relate these properties to one another.

Definition 8.4.8. *A $2n$-signal $\sigma \in \mathsf{Sig}_{2n}(\Sigma)$ is called **non-elastic** if the following property holds:*

(NS) *For each $i, j \in [1 \dots n]$, if $\sigma_{i,n+i} \neq \varepsilon$ and $\sigma_{j,n+j} \neq \varepsilon$ then $\ell(\sigma_{i,n+j}) \geq 0$ and $\ell(\sigma_{j,n+i}) \geq 0$.*

*A $2n$-signal language is called **non-elastic** if each $2n$-signal in it is non-elastic.*
*A $2n$-region automaton is called **non-elastic** if its $2n$-signal language is non-elastic.*

As we intend to represent n-signal languages by sets of n-regions, and further by sets of n-word representations, we need to transport the notion of non-elasticity from $2n$-signals to $2n$-regions and to $2n$-word representations in a consistent way. Moreover, we expect that the notion of non-elasticity of $2n$-word representations rely on the notion of non-elasticity of $2n$-words, similar to Definition 7.3.1.

Definition 8.4.9. *A $2n$-DBM $D \in \mathcal{D}\mathsf{bm}_{2n}$ is called **non-elastic** iff the following property holds:*

(ND) *For each $i, j \in [1 \dots n]$, if $D_{i,n+i} \setminus \{0\} \neq \emptyset$, $D_{j,n+j} \setminus \{0\} \neq \emptyset$ then $D_{i,n+j} \subseteq [0, \infty[$ and $D_{j,n+i} \subseteq [0, \infty[$.*

Proposition 8.4.10. *For each $2n$-DBM in normal form $D \in \mathcal{D}\mathsf{nf}_{2n}$, D is non-elastic iff $\|D\|$ is non-elastic.*

Proof. For proving the first property, observe first that, if $\|D\|$ contains a $2n$-signal which is elastic then D itself must not be non-elastic. For the other implication suppose D is elastic, hence there exists a pair of indices $i_0, j_0 \in [1 \dots n]$ such that $D_{i_0,n+i_0} \setminus \{0\} \neq \emptyset$, $D_{j_0,n+j_0} \neq \emptyset$ but $D_{i_0,n+j_0} \not\subseteq [0, \infty[$ or $D_{j_0,n+i_0} \not\subseteq [0, \infty[$. Suppose also, for the sake of contradiction, that $\|D\|$ is non-elastic.

Let us first observe that $D_{i_0,n+i_0} \subseteq [0, \infty[$ and similarly $D_{j+0,n+j_0} \subseteq [0, \infty[$, since otherwise we may construct, by means of Proposition 8.1.5, a $2n$-signal $\sigma \in \|D\|$ with $\ell(\sigma_{i_0,n+i_0}) < 0$, hence contradicting the assumption that $\|D\|$ is non-elastic.

We will replace first D by the "sub-DBM" D' which is obtained from D by transforming closed parentheses into open parentheses on all *nonpoint* components of D. That is,

$$D'_{ij} = \begin{cases}]\alpha, \beta[& \text{iff } \inf D_{ij} = \alpha, \sup D_{ij} = \beta \text{ and } \alpha \neq \beta \\ D_{ij} & \text{otherwise, that is, iff } D_{ij} = \{\alpha\} \text{ for some } \alpha \in \mathbb{R} \end{cases}$$

Let us first observe that D' is also a $2n$-DBM in normal form: for each $i, j, k \in [1 \ldots 2n]$, since $D_{ik} \subseteq D_{ij} + D_{jk}$ it follows also that $intD_{ik} \subseteq intD_{ij} + intD_{jk}$, (where we have denoted by $intA$ the interior of a set of reals $A \subseteq \mathbb{R}$). Hence the triangle inclusion is valid if all three components are nonpoint sets. The triangle inclusion also holds for all triplet of point components of D' since such components are copied from D. It remains to check the triangle inclusion for the case when one or two components are point intervals and the other (or the others) is (are) nonpoint interval(s).

Observe first that the situation with D_{ij} and D_{jk} being point sets and D_{ik} nonpoint set is impossible, since the sum of two point sets is also a point set.

Suppose D_{ij} is a point set and D_{jk}, D_{ik} are nonpoint sets, say

$$D_{ij} = \{\alpha\},\ D_{jk} = [\beta, \beta'],\ D_{ik} = [\gamma, \gamma']$$

the cases with other parentheses for D_{jk} and D_{ik} being treated similarly. Since D is a DBM we have $D_{ik} \subseteq D_{ij} + D_{jk}$, that is, $[\beta, \beta'] \subseteq [\alpha + \gamma, \alpha + \gamma']$. Therefore

$$D'_{ik} =]\beta, \beta'[\subseteq]\alpha+\gamma, \alpha+\gamma'[= D'_{ij} + D'_{jk}$$

A similar proof can be done when D_{ij} and D_{ik} are both point intervals and D_{jk} is nonpoint.

The last distinct case is when D_{ik} is a point interval and one of D_{ij} or D_{jk} is a nonpoint interval: suppose, w.l.o.g., that

$$D_{ik} = \{\alpha\},\ D_{ij} = [\beta, \beta'],\ D_{jk} = [\gamma, \gamma'] \text{ with } \beta < \beta'$$

Since by hypothesis $D_{ij} \subseteq D_{ij} + D_{jk}$, we must then have

$$[\beta, \beta'] \subseteq \{\alpha\} + [-\gamma', -\gamma] = [\alpha - \gamma', \alpha - \gamma]$$

which means that $\beta' \leq \alpha - \gamma$, hence

$$\alpha \geq \beta' + \gamma > \beta + \gamma$$

Similarly we may prove that $\alpha < \beta' + \gamma'$, hence in fact we must have $\alpha \in]\beta+\gamma, \beta'+\gamma'[$. But this is equivalent to the triangle inclusion $D'_{ik} \subseteq D'_{ij} + D'_{jk}$. Hence D' is a DBM.

Observe now that, following our observation that $D_{i_0,n+i_0}, D_{j_0,n+j_0} \subseteq [0, \infty[$, we must have $D_{i_0,n+i_0}, D_{j_0,n+j_0} \subseteq]0, \infty[$.

We utilize then D' as follows: take some negative number $\alpha \in D'_{i_0,n+j_0}$. Such a number must exist, because we have assumed that $D_{i_0,n+j_0} \not\subseteq [0, \infty[$ and $D'_{i_0,n+j_0} = intD_{i_0,n+j_0}$. The number α can also be regarded as a 2-signal in the semantics of the 2-DBM $D'\big|_{\{i_0,n+j_0\}}$. Then, by recursively applying the construction from Proposition 8.1.5 we extend this to a $2n$-signal $\overline{\alpha} \in \|D'\|$.

But obviously $\overline{\alpha}$ is elastic since

- $\overline{\alpha}_{i_0,n+i_0} \in D'_{i,n+i}$ hence $\ell(\overline{\alpha}_{i_0,n+i_0}) > 0$,
- similarly $\ell(\overline{\alpha}_{j_0,n+j_0}) > 0$,
- and $\ell(\overline{\alpha}_{i_0,n+j_0}) = \alpha < 0$.

which gives a contradiction with the non-elasticity of $\|D\|$, since $\|D'\| \subseteq \|D\|$. □

Non-elasticity extends to $2n$-word representations in the following way: a $2n$-word representation (W, M) is called **non-elastic** if the $2n$-DBM represented by it is non-elastic. Consequently, a non-elastic $2n$-word representation must satisfy the following property: for each $i, j \in [1 \dots n]$,

if
- $W_{i,n+i} > 0$ or ($W_{i,n+i} = 0$ and $M_{i,n+i} = \text{`} > \text{`}$)

and
- $W_{j,n+j} > 0$ or ($W_{j,n+j} = 0$ and $M_{j,n+j} = \text{`} > \text{`}$)

then $W_{i,n+j} > 0$ or ($W_{i,n+j} = 0$ and $M_{i,n+j} \in \{\text{`} = \text{`}, \text{`} > \text{`}\}$).

Observe that if a $2n$-word representation (W, M) is non-elastic then the $2n$-word W is non-elastic.

8.4.3 Closure under concatenation and star

Proposition 8.4.11. *Given two convex $2n$-region automata $\mathcal{A}$ and $\mathcal{B}$, there exists a $2n$-region automaton $\mathcal{D}$ with the property that $L_{sig}(\mathcal{D}) = L_{sig}(\mathcal{A}) \odot L_{sig}(\mathcal{B})$. Moreover, if both $\mathcal{A}$ and $\mathcal{B}$ are non-elastic then $\mathcal{D}$ is non-elastic too.*

Proof. We adapt the concatenation construction in Section 7.4 for $2n$-region automaton: denote the given automata as $\mathcal{A} = (Q, \delta, Q_1, \dots, Q_{2n}, \nu)$ and $\mathcal{B} = (Q', \delta', Q'_1, \dots, Q'_{2n}, \nu')$. There are two ideas that guide this adaptation (we refer the reader to the construction on page 127):

- First, we require that, in each tuple (q, q', X), the labels of q and q' are "consistent", that is, the projection of $\nu(q)$ onto the last n components equals the projection of $\nu'(q')$ onto the first components.
- We then attach to each tuple (q, q', X) a $2n$-relation label M which is in the concatenation of the $2n$-relation labels $\nu(q)$ and $\nu(q')$.

The formalization is the following: we construct first $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$, the completions of $\mathcal{A}$ and $\mathcal{B}$, as in the proof of Proposition 8.4.7. Hence $\hat{\mathcal{A}} = (\hat{Q}, \hat{\delta}, Q_1, \dots, Q_{2n}, \hat{\nu})$ with $\hat{Q} = Q \cup Q_* \cup Q_{**}$ where

$$Q_* = \{(q_*, M) \mid M \in \Gamma_{2n}\} \qquad \hat{\nu}(q_*, M) = M$$
$$Q_{**} = \{(q_{**}, M) \mid M \in \Gamma_{2n}\} \qquad \hat{\nu}(q_*, M) = M$$

and $(q_*, M) \xrightarrow{1} (q_*, M)$, $(q_*, M) \xrightarrow{1} q$, respectively $(q_{**}, M) \xrightarrow{1} (q_{**}, M)$, $q \xrightarrow{1} (q_{**}, M)$ for all $q \in \bigcup_{i \in 1 \dots n} Q_i$.

Similarly $\hat{\mathcal{B}} = (\hat{Q}', \hat{\delta}', Q'_1, \dots, Q'_{2n}, \hat{\nu}')$ with $\hat{Q}' = Q' \cup Q'_* \cup Q'_{**}$ where

$$Q'_* = \{(q'_*, M) \mid M \in \Gamma_{2n}\} \qquad \hat{\nu}'(q'_*, M) = M$$
$$Q'_{**} = \{(q'_{**}, M) \mid M \in \Gamma_{2n}\} \qquad \hat{\nu}'(q'_{**}, M) = M$$

and $(q'_*, M) \xrightarrow{1} (q'_*, M)$, $(q'_*, M) \xrightarrow{1} q'$, respectively $(q'_{**}, M) \xrightarrow{1} (q'_{**}, M)$, $q' \xrightarrow{1} (q'_{**}, M)$ for all $q' \in \bigcup_{i\in 1\ldots n} Q'_i$. We will also assume, as stated by Remark 8.4.2, that in both $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$ the transition relation is consistent w.r.t. the $2n$-relation labeling, that is, two states are connected by a transition iff they are both labeled with the same $2n$-relation.

Then build the following $2n$-region automaton:

$$\begin{aligned}
\mathcal{C} &= \big(Q_\odot, \theta, S_1, \ldots, S_{2n}, \nu_\odot\big) \text{ with}\\
Q_\odot &= \big\{(q, q', X, M) \mid X \subseteq [1\ldots n], q \in \hat{Q}, q' \in \hat{Q}' \text{ with } \hat{\nu}(q)\big|_{[n+1\ldots 2n]} = \hat{\nu}'(q')\big|_{[1\ldots n]}\\
&\qquad \text{and } M = \hat{\nu}(q) \odot \hat{\nu}'(q')\big\}\\
\theta &= \big\{(q, q', X, M) \xrightarrow{a} (r, r', Y, M) \mid (q, q', X, M), (r, r', Y, M) \in Q_\odot, q \xrightarrow{a} r \in \tilde{\delta}, q' \xrightarrow{a} r' \in \tilde{\delta}',\\
&\qquad X \subseteq Y \subseteq [1\ldots n] \text{ and for all } i \in Y \setminus X, r \in Q_{n+i} \text{ and } r' \in Q'_i\big\}\\
S_i &= \big\{(q, q', X, M) \in Q_\odot \mid q \in Q_i\big\}\\
S_{n+i} &= \big\{(q, q', X, M) \in Q_\odot \mid q' \in Q'_{n+i}\big\}\\
\nu_\odot(q, q', X, M) &= M \text{ for all } (q, q', X, M) \in Q_\odot
\end{aligned}$$

Finally drop all the states of $\mathcal{C}$ that are not reachable from $Q_* \times Q'_* \times \{\emptyset\} \times \Gamma_{2n}$ or not coreachable from $Q_{**} \times Q'_{**} \times [1\ldots n] \times \Gamma_{2n}$ and denote $\mathcal{D}$ the resulting automaton.

To prove that $L_{rep}(\mathcal{A}) \odot L_{rep}(\mathcal{B}) \subseteq L_{rep}(\mathcal{D})$, take some $2n$-word representation $(W, M) \in L_{rep}(\mathcal{A}) \odot L_{rep}(\mathcal{B})$, hence there must exist $(W_1, M_1) \in L_{rep}(\mathcal{A})$ and $(W_2, M_2) \in L_{rep}(\mathcal{B})$ such that $W = W_1 \odot W_2$ and $M \in M_1 \odot M_2$. It follows that there exists an accepting run $\rho_1 = (r_i)_{i\in[1\ldots k]}$ in $\mathcal{A}$ (actually we will consider it in $\hat{\mathcal{A}}$)) and a sequence of indices $(i_l)_{l\in[1\ldots 2n]}$ which witness the acceptance of (W_1, M_1) by $\mathcal{A}$, hence all its states are labeled with the $2n$-relation M_1; also we may assume that ρ_1 starts in Q_* and ends in Q_{**}. Similarly, there exists a run $\rho_2 = (r'_i)_{i\in[1\ldots k']}$ in $\hat{\mathcal{B}}$ and a sequence of indices $(j_l)_{l\in[1\ldots 2n]}$ which witness the acceptance of (W_2, M_2) by $\mathcal{B}$ and whose states are all labeled with M_2. We may also assume that ρ_2 starts in Q'_* and ends in Q'_{**}.

As in the proof of Proposition 7.4.1, we may transform the two runs by addition of loops in Q_* and Q_{**}, respectively Q'_* and Q'_{**}, and "translate" the two witnessing sequences $(i_l)_{l\in[1\ldots n]}$ and $(j_l)_{l\in[1\ldots 2n]}$ such that the runs have equal length (we assume thence $k = k'$) and the following property holds:

$$\text{For all } l \in [1\ldots n], i_{n+l} = j_l.$$

Then we construct the run $\rho = (r_i, r'_i, I_i, M)_{i\in[1\ldots k]}$ in which

$$\begin{aligned}
I_1 &= \emptyset\\
I_{i+1} &= I_i \cup \big\{l \in [1\ldots n] \mid r_i = q_{i_{l+n}}\big\}
\end{aligned}$$

and the sequence of witnessing points $(p_l)_{l\in[1\dots 2n]}$ with

$$p_l = \begin{cases} i_l & \text{for } l \in [1\dots n] \\ j_l & \text{for } l \in [n+1\dots 2n] \end{cases}$$

Observe first that each tuple in the run ρ is in $Q_\odot$, since

$$M = M_1 \odot M_2 = \hat{\nu}(r_i) \odot \hat{\nu}'(r'_i) \text{ for all } i \in [1\dots k]$$

Moreover, the first state being in $Q_* \times Q'_* \times \Gamma_{2n}$ and the second state in $Q_{**} \times Q'_{**} \times \Gamma_{2n}$, it follows that all the states are also states of $\mathcal{D}$.

Then observe that $(r_i, r'_i, I_i)_{i\in[1\dots k]}$ and the sequence $(p_l)_{l\in[1\dots 2n]}$ are exactly the run and the witnessing sequence associated with $W_1 \odot W_2$ in the underlying automaton $\mathcal{D}^\flat$. Hence the run and the accepting sequence are associated with the $2n$-word representation (W, M), which shows that $(W, M) \in L_{rep}(\mathcal{D})$.

To prove that $L_{rep}(\mathcal{D}) \subseteq L_{rep}(\mathcal{A}) \odot L_{rep}(\mathcal{B})$, take some $2n$-word representation $(W, M) \in L_{rep}(\mathcal{D})$, which is thence associated with a run $\rho = (r_i, r'_i, X_i, M)_{i\in[1\dots k]}$ in $\mathcal{D}$ and a sequence $(l_i)_{i\in[1\dots 2n]}$.

Observe then that $\rho_1 = (r_i)_{i\in[1\dots 2n]}$ is an accepting run in $\hat{\mathcal{A}}$ because we have assumed that the transition relation is consistent with the $2n$-relation labeling. Similarly, $\rho_2 = (r'_i)_{i\in[1\dots 2n]}$ is an accepting run in $\hat{\mathcal{B}}$. Let's denote M_1 the common $2n$-relation label of all states in ρ_1, that is $M_1 = \hat{\nu}(\rho_1)$ and $M_2 = \hat{\nu}'(\rho_2)$.

Then we construct the following sequence of indices: $(p_j)_{j\in[1\dots n]}$ with $p_j \in [1\dots k]$ and

$$p_j = i \text{ iff } j \in X_{i+1} \setminus X_i$$

It follows that the sequence $\big((l_i)_{i\in[1\dots n]}, (p_i)_{i\in[1\dots n]}\big)$ witnesses the acceptance of some $2n$-word representation (W_1, M_1) by the run ρ_1 in $\hat{\mathcal{A}}$ and the sequence $\big((p_i)_{i\in[1\dots n]}, (l_i)_{i\in[n+1\dots 2n]}\big)$ witnesses the acceptance of some $2n$-word representation (W_2, M_2) by the run ρ_2 in $\hat{\mathcal{B}}$. Moreover $W_1\big|_{[n+1\dots 2n]} = W_2\big|_{[1\dots n]}$ because both are associated with the run ρ and the sequence of indices $(p_i)_{i\in[1\dots n]}$. Hence $(W, M) \in (W_1, M_1) \odot (W_2, M_2)$, which shows that $(W, M) \in L_{rep}(\mathcal{A}) \odot L_{rep}(\mathcal{B})$. □

In the previous chapter we have also proved the closure under indexed juxtaposition for n-automata. The respective construction can be easily adapted to n-region automata, along the same lines of the above proof. We have preferred to present here only the proof for concatenation since it offers insights for the proof for star closure.

Theorem 8.4.12. *Given a convex $2n$-region automaton $\mathcal{A}$, suppose that, for any $k \in \mathbb{N}$, $L_{rep}(\mathcal{A})^{k\odot}$ is a non-elastic $2n$-signal language. Then $L_{rep}(\mathcal{A})^{\circledast}$ is accepted by a $2n$-region automaton.*

Proof. We adapt the construction from Theorem 7.4.3 as follows: first, we replace $\mathcal{A}$ by its completion $\hat{\mathcal{A}}$ in which we assume, following Remark 8.4.2, that the transition function connects only states labeled with the same $2n$-relation.

The automaton that accepts $L_{sig}(\mathcal{A})^{\circledast \geq 2}$ is denoted as

$$\mathcal{C} = (Q_{\circledast}, \delta_{\circledast}, U_1, \ldots, U_{2n}, \nu_{\circledast})$$

$Q_{\circledast}$ consists of tuples $(S, \alpha, \beta, T, M, M')$ in which the first 4 components have the same meaning and utility as in Theorem 7.4.3, while the last two components give the information concerning the $2n$-relation which is to be accepted. Namely, the fifth component is exactly the $2n$-relation-label of the macrostate, while the sixth serves for succesively concatenating all the $2n$-relation labels of the states which have passed through the right active component. The idea is that, at the end of the parse we want to have $M = M'$. In some sense, in M we make a guess at the beginning for the M' that we will get at the end of the parse.

Formally, $Q_{\circledast}$ consists of the following types of states and $\nu_{\circledast}$ gives the following $2n$-relation labeling (we utilize here the notations $\mathcal{Q}$ and $\mathcal{X}$ from the proof of Theorem 7.4.3):

1. $(\emptyset, \emptyset, (X, q, X'), T, M, M')$ where $(X, q, X') \in \mathcal{Q}$, $T \subseteq \mathcal{Q}$, and $M, M' \in \Gamma_{2n}$, with the property that, for all $(Y, r, Y') \in T$,
   a) $\nu(q) = M'$.
   b) $\nu_{\circledast}(\emptyset, \emptyset, (X, q, X'), T, M, M') = M$.
   c) $X \cap [n+1 \ldots 2n] \supseteq (Y \cap [1 \ldots n]) + n$.
   d) $X' \cap [n+1 \ldots 2n] \supseteq (Y' \cap [1 \ldots n]) + n$.
   e) $(Y' \setminus Y) \cap [n+1 \ldots 2n] \subseteq ((Y' \setminus Y) \cap [1 \ldots n]) + n \subseteq (X' \setminus X) \cap [n+1 \ldots 2n]$.
2. $(S, (X, q, X'), (Y, r, Y'), T, M, M')$ with $(X, q, X'), (Y, r, Y') \in \mathcal{Q}$, $S, T \subseteq \mathcal{Q}$, $M, M' \in \Gamma_{2n}$, with the following properties:
   a) $\nu(q)\big|_{[n+1 \ldots 2n]} = \nu(r)\big|_{[1 \ldots n]}$.
   b) $\nu_{\circledast}(\emptyset, \emptyset, (X, q, X'), T, M, M') = M$.
   c) $X \cap [n+1 \ldots 2n] = (Y \cap [1 \ldots n]) + n$.
   d) $X' \cap [n+1 \ldots 2n] = (Y' \cap [1 \ldots n]) + n$.
   e) For each $(U, s, U') \in S$,
      i. $U \cap [n+1 \ldots 2n] \supseteq (X \cap [1 \ldots n]) + n$.
      ii. $U' \cap [n+1 \ldots 2n] \supseteq (X' \cap [1 \ldots n]) + n$.
      iii. $(U' \setminus U) \cap [1 \ldots n] \subseteq ((U' \setminus U) \cap [n+1 \ldots 2n]) - n \subseteq (X' \setminus X) \cap [1 \ldots n]$.
   f) For each $(V, t, V') \in T$,
      i. $Y \cap [n+1 \ldots 2n] \supseteq (V \cap [1 \ldots n]) + n$.
      ii. $Y' \cap [n+1 \ldots 2n] \supseteq (V' \cap [1 \ldots n]) + n$.
      iii. $(V' \setminus V) \cap [n+1 \ldots 2n] \subseteq ((V' \setminus V) \cap [1 \ldots n]) + n \subseteq (Y' \setminus Y) \cap [n+1 \ldots 2n]$.
3. $(S, (X, q, X'), \emptyset, \emptyset, M, M')$ with $(X, q, X') \in \mathcal{Q}$, $S \subseteq \mathcal{Q}$, and $M, M' \in \Gamma_{2n}$, with the property that for all $(Y, r, Y') \in S$,
   a) $\nu_{\circledast}(\emptyset, \emptyset, (X, q, X'), T, M, M') = M$.
   b) $U \cap [n+1 \ldots 2n] \supseteq (X \cap [1 \ldots n]) + n$.
   c) $U' \cap [n+1 \ldots 2n] \supseteq (X' \cap [1 \ldots n]) + n$.
   d) $(U' \setminus U) \cap [1 \ldots n] \subseteq ((U' \setminus U) \cap [n+1 \ldots 2n]) - n \subseteq (X' \setminus X) \cap [1 \ldots n]$.

The transitions are the following:

1. $(\emptyset,\emptyset,(X,q,X'),T,M,M') \xrightarrow{a} (\emptyset,\emptyset,(X',q',X''),T',M,M')$ iff
   a) $q \xrightarrow{a} q'$;
   b) For all $(V',t',V'') \in T'$ there exists $(V,t,V') \in T$ such that $t \xrightarrow{a} t'$.
2. $(S,(X,q,X'),(Y,r,Y'),T,M,M') \xrightarrow{a} (S',(X',q',X''),(Y',r',Y''),T',M,M')$ iff
   a) $q \xrightarrow{a} q'$, $r \xrightarrow{a} r'$;
   b) For all $(U,s,U') \in S$ there exists $(U,' s',U'') \in S'$ such that $s \xrightarrow{a} s'$;
   c) For all $(V',t',V'') \in T'$ there exists $(V,t,V') \in T$ such that $t \xrightarrow{a} t'$.
3. $(S,(X,q,X'),\emptyset,\emptyset,M,M') \xrightarrow{a} (S',(X',q',X''),\emptyset,\emptyset,M,M')$ iff
   a) $q \xrightarrow{a} q'$;
   b) For all $(U,s,U') \in S$ there exists $(U',s',U'') \in S'$ such that $s \xrightarrow{a} s'$.
4. $(S,(X,q,X'),(Y,r,Y'),T,M,M') \xrightarrow{\varepsilon} (S',(Y,r,Y'),(Z,s,Z'),T',M,M'')$ iff
   - $M'' \in M' \odot \nu(s)$;
   - There exists $X'' \in \mathcal{X}$ such that $(X',q,X'') \in S$;
   - There exists $Y \in \mathcal{X}$ such that $(Y,r,Y') \in T$;
   - For each $(Z,s,Z') \in S$ there exists $Z'' \in \mathcal{X}$ such that $(Z',s,Z'') \in S$.
   - For each $(Z',s,Z'') \in T'$ there exists $Z \in \mathcal{X}$ such that $(Z,s,Z') \in T$.
5. $(\emptyset,\emptyset,(X,q,X'),T,M,M') \xrightarrow{\varepsilon} (\emptyset,(X',q,X''),(Y',r,Y''),T',M,M'')$ iff
   - $M'' = M' \odot \nu(r)$;
   - There exists $Y \in \mathcal{X}$ such that $(Y,r,Y') \in T$;
   - For each $(Z',s,Z'') \in T'$ there exists $Z \in \mathcal{X}$ such that $(Z,s,Z') \in T$.
6. $(S,(X,q,X'),(Y,r,Y'),\emptyset,M,M) \xrightarrow{\varepsilon} (S',(Y,r,Y'),\emptyset,\emptyset,M,M)$ iff
   - There exists $X'' \in \mathcal{X}$ such that $(X',q,X'') \in S$;
   - For each $(Z,s,Z') \in S$ there exists $Z'' \in \mathcal{X}$ such that $(Z',s,Z'') \in S$.

The accepting sets are, for all $i \in [1 \ldots n]$,

$$\begin{aligned} U_i = {} & \big\{(\emptyset,\emptyset,(X,q,X'),T,M,M') \mid i \in X' \setminus X, M, M' \in \Gamma_{2n}, M' = \nu(q)\big\} \cup \\ & \big\{(S,(X,q,X'),(Y,r,Y'),T,M,M') \mid i \in X \setminus X', M, M' \in \Gamma_{2n}, \text{ and} \\ & \qquad \text{for all } (Z,s,Z') \in S, i \in Z' \setminus Z\big\} \end{aligned} \tag{8.58}$$

$$\begin{aligned} U_{n+i} = {} & \big\{(S,(X,q,X'),\emptyset,\emptyset,M,M) \mid n+i \in X' \setminus X, M \in \Gamma_{2n}\big\} \cup \\ & \big\{(S,(X,q,X'),(Y,r,Y'),T,M,M') \mid i+n \in Y' \setminus Y, M, M' \in \Gamma_{2n} \text{ and} \\ & \qquad \text{for all } (Z,s,Z') \in T, i+n \in Z' \setminus Z\big\} \end{aligned} \tag{8.59}$$

Finally, the state space is reduced to the states reachable from the following set

$$Q_0 = \Big\{\big(\emptyset,\emptyset,(\emptyset,q_*,M_0,\emptyset),T,M,M'\big) \mid T \subseteq Q_*, M_0, M, M' \in \Gamma_{2n}\Big\} \tag{8.60}$$

and coreachable from

$$Q_f = \Big\{\big(S,([1 \ldots 2n],q_{**},M_f,[1 \ldots 2n]),\emptyset,\emptyset,M,M \mid S \subseteq Q_{**}, M_f, M \in \Gamma_{2n}\Big\} \tag{8.61}$$

We will denote this reduced state space as $\overline{Q}_{\circledast}$ and the resulting automaton as $\mathcal{D}$. The correctness of this construction is almost the same as the proof of Theorem 7.4.3, with some extra considerations on the relation labels of the states. Though it might look redundant, we have to retrace the constructions in Theorem 7.4.3, for showing why they work for $2n$-region automata too.

For proving the inclusion $L_{rep}(\mathcal{D}) \subseteq L_{rep}(\mathcal{A})^{\circledast \geq 2}$, take some $2n$-word representation (W, M) accepted by $\mathcal{D}$, that is, associated with some accepting run of $\mathcal{D}$ $\rho = (S_i, \alpha_i, \beta_i, T_i, M, M_i)_{i\in[1\ldots m]}$, with

$$(S_1, \alpha_1, \beta_1, T_1, M, M_1) = \Big(\emptyset, \emptyset, (q_*, M'_1, \emptyset), T, M_1\Big)$$
$$(S_m, \alpha_m, \beta_m, T_m, M, M_m) = \Big(\big\{(S, (q_{**}, M', [1\ldots 2n]), \emptyset, \emptyset, M, M\Big)$$

and with a sequence of indices $\boldsymbol{h} = (h_i)_{i\in[1\ldots 2n]}$ with $h_i \in [1\ldots m]$ and $(S_{h_i}, \alpha_{h_i}, \beta_{h_i}, T_{h_i}, M, M_{h_i}) \in U_i$ for all $i \in [1\ldots 2n]$. Hence, for all $i, j \in [1\ldots 2n]$,

$$(S_{h_i}, \alpha_{h_i}, \beta_{h_i}, T_{h_i}, M, M_{h_i}) \xrightarrow{W_{ij}} (S_{h_j}, \alpha_{h_j}, \beta_{h_j}, T_{h_j}, M, M_{h_j})$$

Similarly to the proof of Theorem 7.4.3 we identify a number p of times the run ρ passes through transitions that "move around" states from the right active to the left active component. Denote $(k_1, \ldots, k_p)$ the indices at which transitions of the type 4,5 or 6 occur in ρ. We then use ρ and build p runs in "history" presentation, $\rho_j = (Z_i^j, s_i^j, \overline{Z}_i^j)_{i\in[1\ldots k]}$ in $\hat{\mathcal{A}}$ ($i \in [1\ldots m]$) such that the following properties are satisfied:

1. For each $j \in [1\ldots p]$ and $i \in [k_{j-1}+1\ldots k_j]$, $(Z_i^j, s_i^j, \overline{Z}_i^j)$ is the right active component, $(Z_i^j, s_i^j, \overline{Z}_i^j) = \beta_i$.
2. For each $j \in [2\ldots p+1]$ and $i \in [k_{j-1}+1\ldots k_j]$, $(Z_i^{j-1}, s_i^{j-1}, \overline{Z}_i^{j-1})$ is the right active component, $(Z_i^{j-1}, s_i^{j-1}, \overline{Z}_i^{j-1}) = \alpha_i$.
3. For each $j \in [1\ldots p]$, and $i \in [1\ldots k_{j-1}]$, $(Z_i^j, s_i^j, \overline{Z}_i^j)$ is part of the prophecy component, $(Z_i^j, s_i^j, \overline{Z}_i^j) \in T_i$.
4. For each $j \in [1\ldots p-1]$, and $i \in [k_{j+1}+1\ldots m]$, $(Z_i^j, s_i^j, \overline{Z}_i^j)$ is part of the history component, $(Z_i^j, s_i^j, \overline{Z}_i^j) \in S_i$.
5. $\overline{\rho}_j$ passes through some accepting set Q_{n+i} at the same moment when $\overline{\rho}_{j+1}$ passes through the accepting set Q_i for the same $i \in [1\ldots n]$, that is, the essential property (*) utilized in the proof of Theorem 7.4.3:
   For all $j \in [1\ldots p-1]$ and for all $i \in [1\ldots m]$,
   $Z_i^j \cap [n+1\ldots 2n] = (Z_i^{j+1} \cap [1\ldots n]) + n$ and $\overline{Z}_i^j \cap [n+1\ldots 2n] = (\overline{Z}_i^{j+1} \cap [1\ldots n]) + n$.

Here we have denoted $k_0 = 0$ and $k_{p+1} = m$.

Once having these runs, we translate them to the witnessing presentation by building p sequences of indices $\boldsymbol{l}^i = (l_u^i)_{u\in[1\ldots 2n]}$ ($i \in [1\ldots p]$), with $l_i^j = u$ iff $u \in \overline{Z}_i^j \setminus Z_i^j$, and observe that these sequences witness the acceptance of p $2n$-words $(w_j)_{j\in[1\ldots p]}$ for which we may prove, similarly to the proof of Theorem 7.4.3, that for each $j \in [1\ldots p-1]$,

$$W = w_1 \odot \ldots \odot w_p \qquad w_j\big|_{[n+1\ldots 2n]} = w_{j+1}\big|_{[1\ldots n]}$$
$$w_1\big|_{[1\ldots n]} = w\big|_{[1\ldots n]} w_p\big|_{[n+1\ldots 2n]} \qquad = w\big|_{[n+1\ldots 2n]}$$

As $\overline{\rho}_j = (Z_i^j, s_i^j, \overline{Z}_i^j)_{i\in[1\ldots m]}$ is a run in $\hat{\mathcal{A}}$, it follows that $\nu(s_i^j) = \nu(s_{i+1}^j)$ for all $i, i' \in [1\ldots m]$. Let us denote then $M'_j = \nu(s_i^j)$ for some $i \in [1\ldots n]$. We would like now to relate these $2n$-relation labels with the components M_i in the run ρ, in order to prove that these labels correctly concatenate and that M is in their concatenation.

Let us note first that for each $j \in [1\ldots p-1]$ the tuple $(Z_i^j, s_i^j, \overline{Z}_i^j)$ is the right active component while the tuple $(Z_i^{j+1}, s_i^{j+1}, \overline{Z}_i^{j+1})$ is the left active component. Hence we must have, by requirement 2.b from the construction of $Q_{\circledast}$,

$$M'_j\big|_{[n+1\ldots 2n]} = \nu(s_i^j)\big|_{[n+1\ldots 2n]} = \nu(s_i^{j+1})\big|_{[1\ldots n]} = M'_{j+1}\big|_{[1\ldots n]}$$

Therefore the M'_is correctly concatenate, it only remains to prove that their concatenation is M.

To this end, observe that, for each $j \in [1\ldots p+1]$ and each $i \in [k_{j-1}+1, k_j]$ (we consider $k_0 = 0$ and $k_{p+1} = m$), the i-th transition is of type 1,2 or 3 and therefore $M_i = M_{i+1}$. By requirement 1.a from the construction of $Q_{\circledast}$, M_1 must be the label of the right component of the first tuple, hence $M_1 = M'_1$. It follows that $M_i = M'_1$ for all $i \in [1\ldots k_1]$.

Consider now the k_1-th transition. It is an ε-transition of type 4 and it pulls the state s_i^2 out of the prophecy component into the right active component. Therefore, by construction, $M_{k_1+1} \in M_{k_1} \odot \nu(s_i^2)$, that is $M_{k_1+1} \in M'_1 \odot M'_2$.

By induction we may then prove that, if the i-th right active component is $(Z_i^j, s_i^j, \overline{Z}_i^j)$ then $M_i \in M'_1 \odot \ldots \odot M'_j$. Hence, for $i = m$ we have that $M_m \in M'_1 \odot \ldots \odot M'_p$. But $M_m = M$, and therefore $(W, M) \in (w_1, M'_1) \odot \ldots \odot (w_p, M'_p)$, fact which shows that $(W, M) \in L_{rep}(\mathcal{A})^{\circledast \geq 2}$.

For the reverse proof, take p $2n$-word representations $(w_i, M_i) \in L_{rep}(\mathcal{A})$ which correctly concatenate, that is, $(w_i, M_i)\big|_{[n+1\ldots 2n]} = (w_{i+1}, M_{i+1})\big|_{[1\ldots n]}$ for all $i \in [1\ldots p-1]$, and consider p accepting runs in the completed $2n$-automaton $\tilde{\mathcal{A}}$, one for each (w_i, M_i), together with their witnessing sequences of indices:

$$\rho_i = (q_j^i)_{j\in m_i} \text{ with witnessing index sequence } (l_k^i)_{k\in[1\ldots 2n]}$$

We assume that each run starts in Q_* and ends in Q_{**}.

Then consider some $2n$-word representation $(w, M) \in (w_1, M_1) \odot \ldots \odot (w_p, M_p)$, that is, take $w = w_1 \odot \ldots \odot w_p$ and $M \in M_1 \odot \ldots M_p$. We would like to show that $(w, M) \in L(\mathcal{D})$.

The first step is to transform each run ρ_i into a run in the "history" presentation, that is, denote X_j^i the set of indices of the accepting states which were visited by each run ρ_i *just before* the j-th step and by $\overline{X}_j^i$ the set of indices of accepting states visited by ρ_i *up to* the j-th step, and also denote Δ_j^i their difference:

$$X_j^i = \{u \in [1\ldots 2n] \mid \exists v \in [1\ldots j-1] \text{ such that } l_u^i = v\}$$
$$\overline{X}_j^i = \{u \in [1\ldots 2n] \mid \exists v \in [1\ldots j] \text{ such that } l_u^i = v\}$$
$$\Delta_j^i = \overline{X}_j^i \setminus X_j^i.$$

Then, similarly to the proof of Theorem 7.4.3, we bring all the runs ρ_i to equal length, say m, and build up a sequence of tuples $\theta = (S_i, \alpha_i, \beta_i, T_i, M, M'_i)_{i\in[1\ldots m+p]}$, as follows:

1. The first tuple in θ is

$$\big(\emptyset, \emptyset, (X_1^1, q_1^1, \overline{X}_1^1), \{(X_1^i, q_1^i, \overline{X}_1^i) \mid i \in [2 \ldots p]\}, M, \nu(q_1^1)\big)$$

and the last tuple is

$$\big(\{(X_m^i, q_m^i, \overline{X}_m^i) \mid i \in [1 \ldots p-1]\}, (X_m^p, q_m^p, \overline{X}_m^p), \emptyset, \emptyset, M, M\big)$$

2. If $\beta_{k-1} = (X_{j-1}^i, q_{j-1}^i, \overline{X}_{j-1}^i)$ and for all $i' > i$, $\Delta_j^{i'} \cap [n+1 \ldots 2n] \subseteq (\Delta_j^{i'} \cap [1 \ldots n]) + n$, then we append to the run the tuple $(S_k, \alpha_k, \beta_k, T_k, M, M'_k)$ with:

$$\begin{aligned}
S_k &= \big\{(X_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' < i-1\big\}\\
\alpha_k &= \begin{cases} \emptyset & \text{iff } i = 1\\ (X_j^{i-1}, q_j^{i-1}, \overline{X}_j^{i-1}) & \text{iff } i \geq 2 \end{cases}\\
\beta_k &= (X_j^i, q_j^i, \overline{X}_j^i)\\
T_k &= \big\{(X_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' > i\big\}\\
M'_k &= M'_{k-1}
\end{aligned}$$

3. If $\beta_{k-1} = (X_{j-1}^i, q_{j-1}^i, \overline{X}_{j-1}^i)$ and there exists some $i' > i$ for which

$$\big(\Delta_j^{i'} \cap [n+1 \ldots 2n]\big) \setminus \big((\Delta_j^{i'} \cap [1 \ldots n]) + n\big) \neq \emptyset$$

then let

$$\overline{\imath} = \max\big\{i' \geq i \mid \big(\Delta_j^{i'} \cap [n+1 \ldots 2n]\big) \setminus \big((\Delta_j^{i'} \cap [1 \ldots n]) + n\big) \neq \emptyset\big\}$$

We then append $\overline{\imath} - i + 2$ tuples as follows:

a) The first tuple to be appended is the tuple $(S_k, \alpha_k, \beta_k, T_k, M, M'_k)$ in which:

$$\begin{aligned}
S_k &= \big\{(X_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' \leq i-1\big\}\\
\alpha_k &= (X_j^i, q_j^i, \overline{X}_j^i)\\
\beta_k &= \big(X_j^{i+1}, q_j^{i+1}, X_j^{i+1} \cup (\Delta_j^{i+1} \setminus [n+1 \ldots 2n])\big)\\
T_k &= \big\{(X_j^{i'}, q_j^{i'}, X_j^{i'}) \mid i' > i+1\big\}\\
M'_k &\in M'_{k-1} \odot \nu(q_j^{i+1})
\end{aligned}$$

b) For each $l \in [1 \ldots \overline{\imath} - i - 1]$ we append the tuple $(S_{k+l}, \alpha_{k+l}, \beta_{k+l}, T_{k+l}, M, M'_{k+l})$ with:

$$\begin{aligned}
S_{k+l} &= \big\{(\overline{X}_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' < i+l-1\big\}\\
\alpha_{k+l} &= \big(X_j^{i+l-1} \cup (\Delta_j^{i+l-1} \setminus [n+1 \ldots 2n]), q_j^{i+l-1}, \overline{X}_j^{i+l-1}\big)\\
\beta_{k+l} &= \big(X_j^{i+l}, q_j^{i+l}, X_j^{i+l} \cup (\Delta_j^{i+l} \setminus [n+1 \ldots 2n])\big)\\
T_{k+l} &= \big\{(X_j^{i'}, q_j^{i'}, X_j^{i'}) \mid i' > i+l\big\}\\
M'_{k+l} &= M'_{k+l-1} \odot \nu(q_j^{i+l})
\end{aligned}$$

c) For $l = \overline{\iota} - i$ we append the tuple $(S_{k+\overline{\iota}-i}, \alpha_{k+\overline{\iota}-i}, \beta_{k+\overline{\iota}-i}, T_{k+\overline{\iota}-i}, M, M'_{k+\overline{\iota}-i})$ where:

$$
\begin{aligned}
S_{k+\overline{\iota}-i} &= \left\{(\overline{X}_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' < i-1\right\} \\
\alpha_{k+\overline{\iota}-i} &= \left(X_j^{\overline{\iota}} \cup \Delta_j^{\overline{\iota}} \setminus [n+1 \ldots 2n]), q_j^{\overline{\iota}}, \overline{X}_j^{\overline{\iota}}\right) \\
\beta_{k+\overline{\iota}-i} &= \begin{cases} \emptyset & \text{iff } \overline{\iota} = p \\ (X_j^{\overline{\iota}+1}, q_j^{\overline{\iota}+1}, \overline{X}_j^{\overline{\iota}+1}) & \text{iff } \overline{\iota} \leq p-1 \end{cases} \\
T_{k+\overline{\iota}-i} &= \left\{(X_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' > \overline{\iota}+1\right\} \\
M'_{k+\overline{\iota}-i} &= M'_{k+\overline{\iota}-i-1} \odot \nu(q_j^{k+\overline{\iota}-i})
\end{aligned}
$$

4. If $\beta_{k-1} = \emptyset$, which can only happen when $i = p$, we append the tuple $(S_k, \alpha_k, \beta_k, T_k, M, M'_k)$ with:

$$
\begin{aligned}
S_k &= \left\{(X_j^{i'}, q_j^{i'}, \overline{X}_j^{i'}) \mid i' < p\right\} \\
\alpha_k &= (X_j^p, q_j^p, \overline{X}_j^p) \\
\beta_k &= \emptyset \\
T_k &= \emptyset \\
M'_k &= M'_{k-1}
\end{aligned}
$$

Consider the sequence of indices $\boldsymbol{t} = (t_u)_{u \in [1 \ldots 2n]}$ such that $t_u = l_u^1$ and $t_{n+u} = l_{n+u}^p$ for all $u \in [1 \ldots n]$. Also consider the run in $\mathcal{A}^\flat$ $\theta^\flat = (S_i, \alpha_i, \beta_i, T_i)_{i \in [1 \ldots m+p]}$ that is, we purge the $2n$-relations from the components of θ. Then the pair $(\theta^\flat, \boldsymbol{t})$ witnesses the acceptance of w by the underlying automaton $\mathcal{D}^\flat$.

To end the proof, we only need to show that θ is a run in $\mathcal{D}$. The specific requirements (for $2n$-region automata) that need to be checked are 1.a and 2.a, since all the other requirements are either trivially true (the case of 1.b, 2.b and 3.a) or implied by the fact that $\theta^\flat$ is an accepting run in $\mathcal{D}^\flat$ (the case of requirements 1.c, 1.d, 1.e, 2.c, 2.d, 2.e.i, 2.e.ii, 2.e.iii, 2.f.i, 2.f.ii, 2.f.iii, 3.b, 3.c, 3.d).

The validity of requirement 1.a is straightforward, since $M'_1 = \nu(q_1^1)$ and each step before the first ε-transition preserves M'_1.

For proving the validity of requirement 2.a, we have to observe that, according to the definition of θ, at each moment $k = i+j$ at which $\alpha_k \neq \emptyset$ and $\beta_k \neq \emptyset$, the left and the right active components are the j-th tuple of the run ρ_i, respectively the run ρ_{i+1}, that is,

$$
\alpha_k = (X_j^{i-1}, q_j^{i-1}, \overline{X}_j^{i-1}), \qquad \beta_k = (X_j^i, q_j^i, \overline{X}_j^i)
$$

This implies that $\nu(q_j^{i-1}) = M_{i-1}$ and $\nu(q_j^i) = M_i$, and then, due to the hypothesis that M_i correctly concatenates to M_{i+1}, we will get that $\nu(q_j^{i-1}) = \nu(q_j^i)$. Hence requirement 2.a also holds. □

9. Applications

In this chapter we gather together all the result and techniques developed so far, and provide a method for checking whether the semantics of a $2n$-signal regular expression is empty. Consequently we get a method for checking whether the language of a timed automaton is empty.

We have seen how to study untimed behavior and timing behavior of systems by using $2n$-automata. We might then think to decompose each $2n$-regsignal into the "untimed" part, which represents the qualitative behavior of the modeled system, and the "timing" part, which give the temporal constraints on the behavior of the system.

More formally, given $R \in \mathcal{RS}ig_{2n}(\Sigma)$ where $R_{ij} = \langle E_{ij} \rangle_{I_{ij}} + \langle E'_{ij} \rangle_{I'_{ij}}$ with E_{ij} regular expression over Σ, E'_{ij} regular expression over Σ^{-1}, $I_{ij} \subseteq \mathbb{R}_{\geq 0}$ and $I'_{ij} \subseteq \mathbb{R}_{\leq 0}$, we define the following two n-regsignals:

1. R^u, called the *untiming* of R, $R^u_{ij} = E_{ij} + E'_{ij}$ for all $i, j \in [1 \dots 2n]$.
2. R^t, called the *timing* of R, $R^t_{ij} = \langle \Sigma^* \rangle_{I_{ij}} + \langle (\Sigma^{-1})^* \rangle_{I'_{ij}}$ for all $i, j \in [1 \dots 2n]$.

Then $\|R\| = \|R^u \cap R^t\|$.

R^u can be considered as a $2n$-regword and R^t as a $2n$-DBM. This implies that from each $2n$-signal $\sigma \in \|R\|$ we keep only the untiming information and the duration of each component σ_{ij}, for $i, j \in [1 \dots 2n]$. Hence the two aspects, untimed behavior and timing behavior, can be studied separately.

However, for systems in which both the untimed behavior and the timing constraints are important, studying each one separately might prove an incomplete method, since we might miss the interconnections which limit the behaviors. In our setting, this amounts to some expressions which, when decomposed into timing and untimed and studied separately, give nonempty semantics, while it is clear that their semantics is empty. For example, the following 4-signal regular expressions clearly has an empty semantics:

$$\begin{pmatrix} \varepsilon & \langle a \rangle_1 & \varepsilon & \langle ab \rangle_3 \\ \langle a^{-1} \rangle_{-1} & \varepsilon & \langle a^{-1} \rangle_{-1} & \langle b \rangle_2 \\ \varepsilon & \langle a \rangle_1 & \varepsilon & \langle ab \rangle_3 \\ \langle b^{-1}a^{-1} \rangle_{-3} & \langle b^{-1} \rangle_{-2} & \langle b^{-1}a^{-1} \rangle_{-3} & \varepsilon \end{pmatrix} \odot \begin{pmatrix} \varepsilon & \langle ab \rangle_3 & \langle a \rangle_2 & \langle ab \rangle_3 \\ \langle b^{-1}a^{-1} \rangle_{-3} & \varepsilon & \langle b^{-1} \rangle_{-1} & \varepsilon \\ \langle a^{-1} \rangle_{-2} & \langle b \rangle_1 & \varepsilon & \langle b \rangle_1 \\ \langle b^{-1}a^{-1} \rangle_{-3} & \varepsilon & \langle b^{-1} \rangle_{-1} & \varepsilon \end{pmatrix} \tag{9.1}$$

The reason why the concatenation of the two 4-regsignals given in 9.1 has an empty semantics lies in the fact that the first 4-regsignal requires an a state of length 1 while the second imposes an a state of length 2. The Figure 9.1 below gives a graphical interpretation of this empty concatenation.

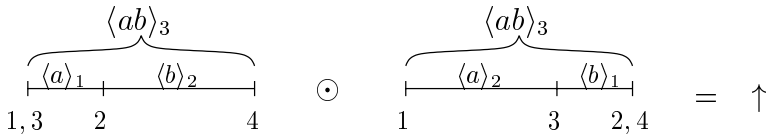


Fig. 9.1. A concatenation of two 4-regsignals that has an empty semantics.

On the contrary both the untimed and the timing of the 4-signal regular expression in 9.1 have nonempty semantics:

$$\begin{pmatrix} 0 & 1 & 0 & 3 \\ -1 & 0 & -1 & 2 \\ 0 & 1 & 0 & 3 \\ -3 & -2 & -3 & 0 \end{pmatrix} \odot \begin{pmatrix} 0 & 3 & 2 & 3 \\ -3 & 0 & -1 & 0 \\ -2 & 1 & 0 & 1 \\ -3 & 0 & -1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 \\ -3 & -2 & -1 & 0 \end{pmatrix}$$

respectively

$$\begin{pmatrix} \varepsilon & a & \varepsilon & ab \\ a^{-1} & \varepsilon & a^{-1} & b \\ \varepsilon & a & \varepsilon & ab \\ b^{-1}a^{-1} & b^{-1} & b^{-1}a^{-1} & \varepsilon \end{pmatrix} \odot \begin{pmatrix} \varepsilon & ab & a & ab \\ b^{-1}a^{-1} & \varepsilon & b^{-1} & \varepsilon \\ a^{-1} & b & \varepsilon & b \\ b^{-1}a^{-1} & \varepsilon & b^{-1} & \varepsilon \end{pmatrix} = \begin{pmatrix} \varepsilon & a & a & ab \\ a^{-1} & \varepsilon & \varepsilon & b \\ a^{-1} & \varepsilon & \varepsilon & b \\ b^{-1}a^{-1} & b^{-1} & b^{-1} & \varepsilon \end{pmatrix}$$

The correct handling of such expressions requires working with both the untimed structure and the timing structure together. But we only know to handle each one on its own.

The solution is the following: to decompose first each $2n$-regsignal into the untimed and the timing part, then to build the $2n$-word representation of the timed part, and finally to recombine the $2n$-regword in the untimed part with the $2n$-regword over a one-letter alphabet from the $2n$-word representation of the timing part.

This recombination is simply the *shuffle* of the two $2n$-regwords. The simple but essential properties of shuffle that we will take advantage of is the fact that, for any two sets L, L', $L \sqcup\!\sqcup L'$ is empty iff both L and L' are empty. Then what remains to be shown is that the union/concatenation/star constructions correctly "combine" with this shuffle operation.

We show here that this idea works fine, in spite of the noncompositionality of projection on our shuffled items. The reason this time noncompositionality is no longer harmful is that we are able to provide a "weak compositionality" result, saying that the shuffle representation has a nonempty semantics iff the semantics of the initial $2n$-signal regular expression is nonempty.

We end this section with an expected result, namely that the $2n$-regsignals that we have produced for timed automata satisfy the non-elasticity assumption, hence we can use the technique developed here for checking timed automata for emptiness.

9.1 Decomposition and recomposition of $2n$-signal regular expressions

Throughout this section we will extend several operations from (2-dimensional) words/signals to n-words/n-signals:

1. $\overbrace{\cdot}$ is the extension of the canonical projection $\overbrace{\cdot} : \Sigma^* \to \Sigma^*/\tilde{\rho}$ (defined in Chapter 2, page 27) to n-words: for each n-word $w \in \mathcal{WD}_{2n}(\Sigma)$ and $i, j \in [1 \dots n]$,
$$\big(\overbrace{w}\big)_{ij} = \overbrace{w_{ij}}$$
Observe that we first need to extend $\overbrace{\cdot}$ to *antiwords* and only after that to n-words.
2. $\mathcal{U}$ is the extension of the untiming morphism $\mathcal{U} : \mathsf{Sig}(\Sigma) \to SF(\Sigma)$ to arbitrary n-signals: for each n-signal $\sigma \in \mathsf{Sig}_n(\Sigma)$ and $i, j \in [1 \dots n]$,
$$\big(\mathcal{U}(\sigma)\big)_{ij} = \mathcal{U}(\sigma_{ij})$$
Observe again that we first extend $\mathcal{U}$ to *antisignals* and only after that to n-signals.
3. ℓ is the extension of the length morphism $\ell : \mathsf{Sig}(\Sigma) \to \mathbb{R}_{\geq 0}$ to arbitrary n-signals: for each n-signal $\sigma \in \mathsf{Sig}_n(\Sigma)$ and $i, j \in [1 \dots n]$,
$$\big(\ell(\sigma)\big)_{ij} = \ell(\sigma_{ij})$$
Similarly to above we first need to extend ℓ to *antisignals*.

The definition of the shuffle operation on words is the following: given two words $w, w' \in \Sigma^*$, the **shuffle** of w and w', denoted $w \sqcup\!\sqcup w'$, is the language obtained as follows: for each $k \in \mathbb{N}$, we decompose w in k words, $w = w_1 \dots w_k$ and w' in k words too, $w' = w'_1 \dots w'_k$, and then recombine these pieces into a single word by interleaving subwords of w with subwords of w'. More formally,

$$\begin{aligned} w \sqcup\!\sqcup w' = \big\{ w'' \mid \exists w_1, \dots w_k, w'_1, \dots, w'_k \in \Sigma^* \text{ such that} \\ w = w_1 \dots w_k, w' = w'_1 \dots w'_k \text{ and } w'' = w_1 w'_1 \dots w_k w'_k \big\} \end{aligned}$$

We will take advantage of the fact that we utilize disjoint sets of symbols (the set of symbols which represent states within the signals, and the singleton set which is used in $2n$-word representations) and redefine shuffle with the aid of monoid morphisms as follows:

Let us consider two disjoint sets of symbols $\Sigma \cap \Omega = \emptyset$. We will define the shuffle of $w \in \Sigma^*$ and $w' \in \Omega^*$ as the set of words w'' with the property that, if we delete from w'' the symbols from Σ, the result is w, and if we delete the symbols from Ω we get w'.

The formal definition of "deletion of symbols" is the following: denote first κ_Σ and κ_Ω the applications

$$\kappa_\Sigma : (\Sigma \cup \Omega) \to \Sigma^*, \kappa_\Sigma(a) = \begin{cases} a & \text{for all } a \in \Sigma \\ \varepsilon & \text{for all } a \in \Omega \end{cases}$$

$$\kappa_\Omega : (\Sigma \cup \Omega) \to \Omega^*, \kappa_\Omega(a) = \begin{cases} a & \text{for all } a \in \Omega \\ \varepsilon & \text{for all } a \in \Sigma \end{cases}$$

Then the "deletion of symbols" are the induced morphisms $\kappa_\Sigma^\flat : (\Sigma \cup \Omega)^* \to \Sigma^*$ and resp. $\kappa_\Omega^\flat : (\Sigma \cup \Omega)^* \to \Omega^*$. In the sequel we will utilize the notations κ_Σ and κ_Ω for the induced morphisms too.

Definition 9.1.1. *Given two words $w \in \Sigma^*$, $w' \in \Omega^*$, the **shuffle** of w and w', denoted $w \sqcup\!\sqcup w'$, is the following set:*

$$w \sqcup\!\sqcup w' = \left\{ w'' \in (\Sigma \cup \Omega)^* \mid \kappa_\Sigma(w'') = w \text{ and } \kappa_\Omega(w'') = w' \right\} \tag{9.2}$$

The generalization to n-words gives the following:

Definition 9.1.2. *Suppose we are given two $2n$-words $w, w' \in \mathcal{WD}_n(\Sigma)$. The **shuffle** of w with w', denoted $w \sqcup\!\sqcup w'$, is the set of $2n$-words for which, for each $i, j \in [1 \dots n]$, the (i,j)-component belongs to the shuffle of w_{ij} with w'_{ij}:*

$$w \sqcup\!\sqcup w' = \left\{ w'' \in \mathcal{WD}_n(\Sigma \cup \Omega) \mid \text{ for all } i, j \in [1 \dots n], w''_{ij} \in w_{ij} \sqcup\!\sqcup w'_{ij} \right\} \tag{9.3}$$

Note however that "random shuffling" of components does not give in general $w \sqcup\!\sqcup w'$ because some results might not satisfy the triangle identity 6.1.

Even more, we may define a class of $2n$-word regular expressions with shuffle, generated by the following grammar:

$$E ::= R \mid E + E \mid E \sqcup\!\sqcup E \mid E \odot E \mid E^{\circledast}$$

The following proposition shows, in essence, that shuffle is expressible by the other operations, that is, its use does not increase the expressive power of $2n$-word regular expressions:

Proposition 9.1.3. *The class of n-word languages accepted by n-automata is closed under shuffle.*

Proof. The construction is a generalization of a well-known construction for the shuffle of two regular languages. It can be described as an *asynchronous* composition of two automata: at each moment, the automaton for the shuffled language has the possibility to choose between a transition in the first automaton and a transition in the second automaton.

Formally, for any two n-automata $\mathcal{A} = (Q, \delta, Q_1, \dots, Q_n)$ and $\mathcal{B} = (Q', \delta', Q'_1, \dots, Q'_n)$, the automaton accepting $L(\mathcal{A}) \sqcup\!\sqcup L(\mathcal{B})$ is the following:

$$\begin{aligned} &\mathcal{C} = \left(Q \times Q', theta, Q_1 \times Q'_1, \dots, Q_n \times Q'_n\right) \text{ where} \\ &\theta = \left\{ (q,q') \xrightarrow{a} (r,q') \mid q \xrightarrow{a} r \in \delta \right\} \cup \left\{ (q,q') \xrightarrow{a} (q,r') \mid q' \xrightarrow{a} r' \in \delta' \right\} \end{aligned}$$

The proof that $L(\mathcal{C}) = L(\mathcal{A}) \sqcup\!\sqcup L(\mathcal{B})$ is based on the argument that all accepting runs of $\mathcal{C}$ can be obtained by shuffling accepting runs of $\mathcal{A}$ with accepting runs of $\mathcal{B}$. □

9.2 Shuffled n-words

Definition 9.2.1. *An n-**dimensional shuffled word**, or **shuffled** n-**word**, is a tuple (γ, M) consisting of an n-word $\gamma \in \mathcal{WD}_n(\Sigma \cup \{1\})$ and an n-relation $M \in \Gamma_n$.*

The set of shuffled n-words with symbols in Σ is denoted $\mathsf{SW}_n(\Sigma)$.

The semantics of shuffled n-words is based upon the following observation: if we have a n-word w and a n-signal over a one-letter alphabet $\sigma \in \mathsf{Sig}_n(\{a\})$, then we may combine them and build (uncountably many) n-signals whose untiming is stuttering equivalent to w and whose timing is exactly σ.

The **semantics** of a shuffled n-word is the following set:

$$[\![\gamma, M]\!] = \left\{\sigma \in \mathsf{Sig}_n(\Sigma) \mid \mathcal{U}(\sigma) = \overbrace{\kappa_\Sigma(\gamma)} \text{ and } \ell(\sigma) \in \left[\kappa_{\{1\}}(\gamma), M\right]\right\} \tag{9.4}$$

Proposition 9.2.2. *Each shuffled n-word (γ, M) has a nonempty semantics.*

Similarly to n-word representations, we may define an equivalence relation on $\mathsf{SW}_n(\Sigma)$ as follows:

$$(\gamma_1, M_1) \simeq (\gamma_2, M_2) \text{ if and only if } [\![\gamma_1, M_1]\!] = [\![\gamma_2, M_2]\!].$$

Then call a set $S \subseteq \mathsf{SW}_n(\Sigma)$ as **convex** if it is saturated by this equivalence relation, that is, whenever $(\gamma_1, M_1) \in S$ and $[\![\gamma_1, M_1]\!] = [\![\gamma_2, M_2]\!]$ then $(\gamma_2, M_2) \in S$.

9.2.1 Projection on shuffled words

Definition 9.2.3. *Given a shuffled n-word (γ, M) and a set $Y \subseteq [1 \dots n]$, the projection of (γ, M) onto Y is the shuffled $card(Y)$-word*

$$\left(\gamma\big|_Y, M\big|_Y\right)$$

As for n-regsignals, projection poses problems: it is not compositional w.r.t. the semantics. As an example, consider the following shuffled 3-word:

$$\left(\begin{pmatrix} \varepsilon & a1a & a1ab1b \\ a^{-1}1^{-1}a^{-1} & \varepsilon & b1b \\ b^{-1}1^{-1}b^{-1}a^{-1}1^{-1}a^{-1} & b^{-1}1^{-1}b^{-1} & \varepsilon \end{pmatrix}, \begin{pmatrix} \text{`}=\text{`} & \text{`}=\text{`} & \text{`}=\text{`} \\ \text{`}=\text{`} & \text{`}=\text{`} & \text{`}=\text{`} \\ \text{`}=\text{`} & \text{`}=\text{`} & \text{`}=\text{`} \end{pmatrix}\right) \tag{9.5}$$

whose semantics consists exactly of the singleton 3-signal language

$$\left\{\begin{pmatrix} \varepsilon & a^1 & a^1b^1 \\ a^{-1} & \varepsilon & b^1 \\ b^{-1}a^{-1} & b^{-1} & \varepsilon \end{pmatrix}\right\}$$

On the contrary, the projection of the shuffled 3-word defined in 9.5 onto the set $\{1, 3\}$, gives the following shuffled 2-word with its nonsingleton semantics

$$[\![a1ab1b, \text{`}=\text{`}]\!] = \left\{a^\alpha b^\beta \mid \alpha + \beta = 2\right\}$$

We have used here the convention that any word $w \in \Sigma^*$ can be regarded as the 2-word $\begin{pmatrix} \varepsilon & w \\ w^{-1} & \varepsilon \end{pmatrix}$.

However we have the following "weakly compositional" characterization of the emptiness problem for the semantics of shuffled n-words:

Proposition 9.2.4. *1. For each shuffled n-word (γ, M) and $Y \subseteq [1 \dots n]$,*

$$[\![\gamma|_Y, M|_Y]\!] \supseteq [\![\gamma, M]\!]|_Y \tag{9.6}$$

2. For each two equivalent shuffled n-words $(\gamma_1, M_1), (\gamma_2, M_2) \in \mathsf{SW}_n(\Sigma)(\Sigma)$, and subset of indices $Y \subseteq [1 \dots n]$, if $[\![\gamma_1, M_1]\!] = [\![\gamma_2, M_2]\!]$ then the projections of both shuffled n-words on Y are equivalent.

Proof. 1) Straightforward, since for each $\sigma \in [\![\gamma, M]\!]$, $\sigma|_Y \in [\![\gamma|_Y, M|_Y]\!]$ by easy verification.

2) Let us observe that, if $[\![\gamma_1, M_1]\!] = [\![\gamma_2, M_2]\!]$ then $\overbrace{\kappa_\Sigma(\gamma_1)} = \overbrace{\kappa_\Sigma(\gamma_2)}$ and $\left[\kappa_{\{1\}}(\gamma_1), M_1\right] = \left[\kappa_{\{1\}}(\gamma_2), M_2\right]$. This implies that

$$\overbrace{\kappa_\Sigma(\gamma_1)}|_Y = \overbrace{\kappa_\Sigma(\gamma_2)}|_Y \text{ and}$$
$$\left[\kappa_{\{1\}}(\gamma_1), M_1\right]|_Y = \left[\kappa_{\{1\}}(\gamma_2), M_2\right]|_Y$$

By the compositionality of projection on word representation, the last line is equivalent to

$$\left[\kappa_{\{1\}}(\gamma_1)|_Y, M_1|_Y\right] = \left[\kappa_{\{1\}}(\gamma_2)|_Y, M_2|_Y\right]$$

But this means that $[\![\gamma_1|_Y, M_1|_Y]\!] = [\![\gamma_2|_Y, M_2|_Y]\!]$. □

Remark 9.2.5. The inclusion 9.6 is the same as the property 6.26 on page 100, for n-regsignals. Hence we may never get "false negative" answers to the emptiness problem by working with projection at the syntactic level.

The question is then whether we may get "false positive" answers. The answer to this question is negative, due to the fact that we always work with shuffled words, that is, items satisfying the triangle identity and whose timing denote some nonempty region. It will be the task of the normal form algorithm for DBMs, respectively the emptiness algorithm for n-automata, to "purge" the possible items that have empty semantics.

The difference with n-regsignals is that, with these ones, we had no algorithm for checking whether a n-regsignal has an empty semantics or not. For shuffled n-words we have n-automata.

9.2.2 Juxtaposition on shuffled words

The juxtaposition operation on shuffled words can be defined similarly to word representations: one juxtaposes both the word parts and the relational parts in each operand, and the result must be a set, due to juxtaposition on relation matrices:

Definition 9.2.6. *Given a shuffled m-word (γ_1, M_1), a shuffled n-word (γ_2, M_2) and an integer $p \leq \min(m, n)$, the p-**juxtaposition** of (γ_1, M_1) with (γ_2, M_2) is the following* set *of shuffled $(m + n - p)$-words*

$$(\gamma_1, M_1)\square_p(\gamma_2, M_2) = \left\{(\gamma_1 \square_p \gamma_2, M) \mid M \in M_1 \square_p M_2\right\}$$

Remind that, even for word representations of regions, juxtaposition is compositional only when defined on *convex* sets, that is, on sets which were closed under region equivalence.

Proposition 9.2.7. *For each m-word representations $(\gamma_1, M_1) \in \mathsf{SW}_m(\Sigma)$, $(\gamma_2, M_2) \in \mathsf{SW}_n(\Sigma)$ and integer $p \leq \min(m, n)$,*

$$[\![\gamma_1, M_1]\!] \Box_p [\![\gamma_2, M_2]\!] = \bigcup \big\{ [\![\gamma_1' \Box_p \gamma_2', M]\!] \mid (\gamma_1', M_1') \simeq (\gamma_1, M_1), \\ (\gamma_2', M_2') \simeq (\gamma_2, M_2) \text{ and } M \in M_1' \Box_p M_2' \big\} \quad (9.7)$$

Proof. The proof of the inverse inclusion is straightforward: given any $(m + n - p)$-signal

$$\sigma \in \big\{ [\![\gamma_1' \Box_p \gamma_2', M]\!] \mid (\gamma_1', M_1') \equiv (\gamma_1, M_1), (\gamma_2', M_2') \equiv \gamma_2, M_2) \text{ and } M \in M_1' \Box_p M_2' \big\},$$

we have, by the inclusion 9.6 from Proposition 9.2.4, that $\sigma\big|_{[1\ldots m]} \in [\![\gamma_1', M_1']\!] = [\![\gamma_1, M_1]\!]$ and $\sigma\big|_{[m-p+1\ldots m+n-p]} = [\![\gamma_2', M_2']\!] = [\![\gamma_2, M_2]\!]$. But this implies that $\sigma \in [\![\gamma_1, M_1]\!] \Box_p [\![\gamma_2, M_2]\!]$ by definition of $\Box_p$ on signals.

The left-to-right inclusion follows this way: given $\sigma \in [\![\gamma_1, M_1]\!] \Box_p [\![\gamma_2, M_2]\!]$, we will try construct a shuffled $(m + n - p)$-word, denote it (γ, M), whose semantics contains σ. (γ, M) is a juxtaposition of a shuffled m-word equivalent to (γ_1, M_1) with a shuffled n-word equivalent to (γ_2, M_2). This shuffled $(m + n - p)$-word arises as a shuffle of the untiming of σ with any $(m + n - p)$-word representation of the $(m + n - p)$-region which contains $\ell(\sigma)$.

Formally, we pick a $(m+n-p)$-word $w \in \mathcal{U}(\sigma)$, which is possible since $\mathcal{U}(\sigma)$ is nonempty for any n-signal. Then, for the *unique* region $R \in \mathcal{R}\mathsf{egn}_n$ which contains $\ell(\sigma)$ we pick an $(m+n-p)$-word representation (ω, M), hence $\sigma \in [\omega, M]$. We then shuffle w and ω and pick some $(m+n-p)$-word γ in this shuffle. Consequently,

$$\begin{aligned} \mathcal{U}(\sigma\big|_{[1\ldots m]}) &= \overbrace{\kappa_\Sigma(\gamma\big|_{[1\ldots m]})} = \overbrace{\kappa_\Sigma(\gamma_1)}\, \big[\kappa_{\{1\}}(\gamma\big|_{[1\ldots m]}), M\big|_{[1\ldots m]}\big] \\ &= \big[\kappa_{\{1\}}(\gamma_1), M_1\big] \end{aligned}$$

since $\big[\kappa_{\{1\}}(\gamma\big|_{[1\ldots m]}), M\big|_{[1\ldots m]}\big]$ is the *unique* region which contains $\sigma\big|_{[1\ldots m]}$. It then follows that

$$[\![\gamma\big|_{[1\ldots m]}, M\big|_{[1\ldots m]}]\!] = [\![\gamma_1, M_1]\!] \text{ and } [\![\gamma\big|_{[m-p+1\ldots m+n-p]}, M\big|_{[m-p+1\ldots m+n-p]}]\!] = [\![\gamma_2, M_2]\!].$$

The equality $[\![\gamma\big|_{[m-p+1\ldots m+n-p]}, M\big|_{[m-p+1\ldots m+n-p]}]\!] = [\![\gamma_2, M_2]\!]$ follows similarly. □

Remark 9.2.8. Observe that in the above proof we have utilized the fact that the mappings $\mathcal{U}$, κ_Σ, $\kappa_{\{1\}}$ and ℓ commute with projection, fact which can be easily established.

The rest of the good properties of juxtaposition hold as expected:

Proposition 9.2.9. *1. For each convex set of shuffled n-words $S \subseteq \mathsf{SW}_n(\Sigma)$ and each $X \subseteq [1\ldots n]$, $S\big|_X$ is also a convex set of shuffled $card(X)$-words.*

2. *For each two convex sets of shuffled words* $S_1 \subseteq \mathsf{SW}_m(\Sigma)$, $S_2 \subseteq \mathsf{SW}_n(\Sigma)$, *and integer* $p \leq \min(m,n)$, $S_1 \square_p S_2$ *is a convex set of* $(m+n-p)$*-word representations and*

$$[\![S_1 \square_p S_2]\!] = [\![S_1]\!] \square_p [\![S_2]\!] \tag{9.8}$$

3. *For each three convex sets of shuffled words* $S_1 \subseteq \mathsf{SW}_m(\Sigma)$, $S_2 \subseteq \mathsf{SW}_n(\Sigma)$, $S_3 \subseteq \mathsf{SW}_p(\Sigma)$, *and integers* $q \leq \min(m,n)$, $r \leq \min(n,p)$,

$$(S_1 \square_q S_2) \square_r S_3 = S_1 \square_q (S_2 \square_r S_3) \tag{9.9}$$

Proof. The second property is a corollary of Proposition 9.2.7.

The first property can be proved as follows: given a shuffled $card(X)$-word $(\gamma, M) \in S$ and another shuffled n-word (γ', M') with $[\![\gamma\big|_X, M\big|_X]\!] = [\![\gamma', M']\!]$, we get that $\overbrace{\kappa_\Sigma(\gamma\big|_X)} = \overbrace{\kappa_\Sigma(\gamma')}$ and $[\kappa_{\{1\}}(\gamma\big|_X), M\big|_X] = [\kappa_{\{1\}}(\gamma'), M']$.

Let us denote $w_1 = \kappa_\Sigma(\gamma)$ and $w_2 = \kappa_\Sigma(\gamma')$. We may then build a n-word w' such that $\mathcal{U}(w'\big|_X) = \overbrace{w_2}$ and $\mathcal{U}(w') = \overbrace{w_1}$, as follows:

- For each $i, j \in X$, if $(w_1)_{ij} = a_1^{l_1} \ldots a_k^{l_k}$ and $w_{l_X(i) l_X(j)} = a_1^{m_1} \ldots a_k^{m_k}$ then put

$$w'_{ij} = a_1^{\max(l_1, m_1)} \ldots a_k^{\max(l_k, m_k)}$$

- Let i_0 denote one of the *maximal* indices[1] in any ordering compatible w_2, that is, $(w_2)_{ji_0} \in \Sigma^*$ for any $j \in [1 \ldots n]$. Let also j_0 denote one of the *minimal* indices in any ordering compatible w_2, that is, $(w_2)_{j_0 i} \in \Sigma^*$ for any $i \in [1 \ldots n]$.
  Then for all $i \notin X$ for which $(w_1)_{j_0 i} \in \Sigma^*$ put $w'_{i_0 i} = (w_1)_{j_0 i}$, while for all $i \notin X$ for which $(w_1)_{i i_0} \in \Sigma^*$ put $w'_{i_0 i} = (w_1)_{i i_0}$. Finally complete w' by the triangle identity.

  It is easy to observe then that $\overbrace{w'} = \overbrace{w_1}$ while $\overbrace{w'\big|_X} = \overbrace{w_2}$.

Further, let us build an n-word representation (ω_1, M_1) which is equivalent with $(\kappa_{\{1\}}(\gamma), M)$ and extends $(\kappa_{\{1\}}(\gamma'), M')$. This part of the proof has already been done in Proposition 8.3.15.

We may then conclude that any shuffled n-word (γ'', M_2) in which $\gamma'' \in w' \sqcup\!\sqcup \omega_1$ has the property that $[\![\gamma'', M_2]\!] = [\![\gamma, M]\!]$. By convexity of S we get that $(\gamma'', M_2) \in S$, which assures, on its turn, that $(\gamma', M') \in S\big|_X$. □

Proposition 9.2.10. *1. For each five integers* $m, n, p, q, r \in \mathbb{N}$ *with* $r \leq \min(p, q)$, *shuffled words* $(\gamma_1, M_1) \in \mathsf{SW}_m(\Sigma)$, $(\gamma_2, M_2) \in \mathsf{SW}_n(\Sigma)$, *given* $X \subseteq [1 \ldots m]$ *with* $[m - r + 1 \ldots m] \subseteq X$ *and* $card(X) = p$, *and given also* $Y \subseteq [1 \ldots n]$ *with* $[1 \ldots r] \subseteq Y$ *and* $card(Y) = q$, *we have that*

$$(\gamma_1, M_1)\big|_X \square_r (\gamma_2, M_2)\big|_Y = \big((\gamma_1, M_1) \square (\gamma_2, M_2)\big)\big|_{X \cup (Y + m - r)}. \tag{9.10}$$

[1] Such maximal indices are not unique.

2. *Juxtaposition is associative on shuffled words: for each* $(\gamma_1, M_1) \in \mathsf{SW}_m(\Sigma)$, $(\gamma_2, M_2) \in \mathsf{SW}_n(\Sigma)$ *and* $(\gamma_3, M_3) \in \mathsf{SW}_p(\Sigma)$, *and for each* $k \leq \min(m, n)$ *and* $l \leq \min(n, p)$,

$$\big((\gamma_1, M_1)\Box_k(\gamma_2, M_2)\big)\Box_l(\gamma_3, M_3) = (\gamma_1, M_1)\Box_k\big((\gamma_2, M_2)\Box_l(\gamma_3, M_3)\big) \tag{9.11}$$

Proof. Both properties are straightforward corollaries of the respective properties concerning juxtaposition and projection on n-words and n-word representations. □

Note that this proposition refers only to the syntactic properties relating juxtaposition and projection, not to their semantic properties. Therefore it hides no contradiction with the noncompositionality of projection.

9.2.3 Concatenation and star on shuffled words

This is defined as usual, by means of projection an juxtaposition: for each two shuffled $2n$-words $(\gamma_1, M_1), (\gamma_2, M_2) \in \mathsf{SW}_{2n}(\Sigma)$,

$$(\gamma_1, M_1) \odot (\gamma_2, M_2) = \big\{(\gamma_1 \odot \gamma_2, M) \mid M \in M_1 \odot M_2\big\}$$

Composition enjoys the well-known properties at the syntactic level, namely associativity, existence of two units per each shuffled $2n$-word and existence of pseudoinverses, but, due to the use of projection, it is not compositional. It can also be extended to sets of shuffled $2n$-words in the usual way, and gives rise to the star operation

$$\text{for each } S \subseteq \mathsf{SW}_{2n}(\Sigma), S^{\circledast} = \bigcup S^{k\odot} \text{ where } S^0 = \mathbf{1}_{2n} \text{ and } S^{(k+1)\odot} = S^{k\odot} \odot S$$

Here $\mathbf{1}_{2n}$ is the set of all units:

$$\mathbf{1}_{2n} = \{(\gamma, M) \mid \forall i \in X, \gamma_{i,n+i} = \varepsilon, M_{i,n+i} = \text{`} = \text{`}\}$$

The following properties are essential in our "shuffle" approach to checking $2n$-signal regular expressions for emptiness:

Proposition 9.2.11. *1. For each pair of sets of convex shuffled* $2n$*-words* $S_1, S_2 \subseteq \mathsf{SW}_{2n}(\Sigma)$,

$$[\![S_1 \odot S_2]\!] \neq \emptyset \text{ if and only if } [\![S_1]\!] \odot [\![S_2]\!] \neq \emptyset$$

2. *For each set of convex shuffled* $2n$*-words* $S \subseteq \mathsf{SW}_{2n}(\Sigma)$,

$$[\![S^{\circledast}]\!] \neq \emptyset \text{ if and only if } [\![S]\!]^{\circledast} \neq \emptyset$$

Proof. For both properties, the right-to-left implication follows by means of Proposition 9.2.4 and of compositionality of juxtaposition. Note that convexity is essential in this implication. Moreover, the second property follows by induction from the first.

For the proof of the left-to-right implication for the first property, suppose $[\![S_1 \odot S_2]\!] \neq \emptyset$. This implies in fact that there exist $(\gamma_1, M_1) \in S_1$ and $(\gamma_2, M_2) \in M_2)$ such that $\gamma_1 \odot \gamma_2$ is defined and $M_1 \odot M_2$ is nonempty. But this implies also that $\gamma_1 \Box_n \gamma_2$ is defined and $M_1 \Box_n M_2$ is nonempty, hence for any $M \in M_1 \Box_n M_2$ we have that $[\![\gamma_1 \Box_n \gamma_2, M]\!]$ is a nonempty set. The proof ends if we pick up any $\sigma \in [\![\gamma_1 \Box_n \gamma_2, M]\!]$ and observe that $\sigma|_{[1\ldots 2n]} \in [\![\gamma_1, M_1]\!]$ and $\sigma|_{[n+1\ldots 3n]} \in [\![\gamma_2, M_2]\!]$, which means that

$$\sigma|_{[1\ldots 2n]} \odot \sigma|_{[n+1\ldots 3n]} \in [\![\gamma_1, M_1]\!] \odot [\![\gamma_2, M_2]\!]$$

or, in other words, that $[\![\gamma_1, M_1]\!] \odot [\![\gamma_2, M_2]\!]$ is nonempty. □

Shuffle regwords are naturally associated with n-automata that generalize n-region automata: these are tuples $\mathcal{A} = (Q, \delta, Q_1, \ldots, Q_n, \lambda)$, with λ labeling states with n-relations. Proposition 9.1.3 implies that we may construct n-automata for shuffled n-regwords by shuffling n-automata for the untiming with n-region automata.

Proposition 9.2.12. *For each n-regword $R \in \mathcal{RW}_n(\Sigma)$ and convex set of n-word representations $\mathcal{W} \subseteq WdRep_n$, the shuffled n-word semantics of the shuffle regword $R \sqcup\!\sqcup \mathcal{W}$ is a convex set of shuffled n-words.*

Proof. Easy corollary of the convexity of $\mathcal{W}$. □

9.2.4 A method for checking whether the semantics of a $2n$-signal regular expression is empty

1. Given a $2n$-signal regular expression E, we decompose each $2n$-regsignal occurring in E into the untimed part and the timing part.
2. We then interpret the untimed $2n$-regsignal as a $2n$-regword (this implies that stuttering is added) and associate a $2n$-automaton to it.
3. On the other hand, we interpret the timing $2n$-regsignal as a $2n$-EDBM and hence associate a $2n$-region automaton to it.
4. Subsequently, we produce the shuffle of the two automata.
5. Then we apply the union/concatenation/star constructions for the resulting automata (provided the non-elasticity assumption holds) until we associate a $2n$-automaton to the whole expression.
6. Finally, we check whether the semantics of this final $2n$-automaton is void.

In order for this algorithm to be correct, we need to redefine the non-elasticity property for $2n$-signals, and to prove that this definition is consistent with the representations of sets of $2n$-signals.

Definition 9.2.13. *A $2n$-signal σ is called **non-elastic** if the following property holds:*

For each $i, j \in [1 \ldots n]$, if $\sigma_{i,n+i} \neq \varepsilon$ and $\sigma_{j,n+j} \neq \varepsilon$ then both $\sigma_{i,n+j}$ and $\sigma_{j,n+i}$ are not antisignals, that is$\sigma_{i,n+j} \in \mathsf{Sig}(\Sigma)$ and $\sigma_{j,n+i} \in \mathsf{Sig}(\Sigma)$.

Proposition 9.2.14. *For each $2n$-signal $\sigma \in \mathsf{Sig}_{2n}(\Sigma)$, σ is non-elastic if and only if $\mathcal{U}(\sigma)$ is non-elastic, if and only if $\ell(\sigma)$ is non-elastic.*

Proof. Straightforward, due to the fact that $\sigma_{ij} = \varepsilon$ if and only if $\mathcal{U}(\sigma_{ij}) = \varepsilon$ if and only if $\ell(\sigma_{ij}) = 0$. □

Proposition 9.2.15. *Given a $2n$-regword $R \in \mathcal{RW}_{2n}(\Sigma \cup \{1\})$ and a $2n$-relation M, $[\![R, M]\!]$ contains only non-elastic $2n$-signals if and only if $\kappa_\Sigma(\|R\|)$ contains only non-elastic $2n$-words and $[\kappa_{\{1\}}, M]$ is a non-elastic region.*

Proof. Corollary of the above Proposition and of the Proposition 8.4.10. □

Theorem 9.2.16. *The above procedure terminates with some $2n$-automaton with a nonempty language if and only if the semantics of the initial expression is nonempty, provided the $2n$-signal regular expression this initial expression E satisfies the following non-elasticity assumption:*

> *Denote $\mathcal{L}_E$ the union of the semantics of all $2n$-regsignals from which E is built of. Then $\mathcal{L}_E^{k\odot}$ consists of non-elastic $2n$-signals for any $k \in \mathbb{N}$.*

Proof. Corollary of Proposition 9.2.11, which is applied by structural induction on the $2n$-signal regular expression involved. The essential property in the proof is that for each $2n$-regsignal, the shuffle $2n$-regword associated as above is convex. □

9.3 Checking emptiness of timed automata with $2n$-signal regular expressions

Remind that in Chapter 6 we have proved that (languages of) timed automata are embeddable in $2n$-signal regular expressions. Then, our search for a property that assures decidability was guided in part by some observations on the $2n$-signals that occur during this embedding. We will show here that, indeed, timed automata can be simulated by $2n$-signal regular expressions which have the non-elasticity property.

Remind that, in Theorem 6.5.2, for each timed automaton with n clocks $\mathcal{A} = (Q, \Sigma, \delta, \lambda, Q_0, Q_f)$ in which each transition resets at least one clock, and for each transition $q \xrightarrow{(C,X)} r$ in this automaton, in which $X \neq \emptyset$ and $\lambda(q) = a$ we have associated a $2n$-regsignal $R(C, X)$. More specifically, for $C = \big(\bigwedge_{i \in [1\ldots n]} x_i \in I_i \big) \wedge \big(\bigwedge_{i,j \in [1\ldots n], i \neq j} x_i - x_j \in J_{ij} \big)$, the $2n$-regsignal $R(C, X)$ is:

$$R(C,X)_{ij} = \begin{cases} \langle \Sigma^* \cup (\Sigma^{-1})^* \rangle_{J_{ij}} & \text{for } i,j \in [1 \dots n] \\ \langle \Sigma^* \cup (\Sigma^{-1})^* \rangle_{J_{kl}} & \text{for } i = n+k, j = n+l, k, l \in \overline{X} \\ \varepsilon & \text{for } i = n+k, j = n+l, k, l \in X \\ \langle \Sigma^* \cup (\Sigma^{-1})^* \rangle_{J_{ik}} & \text{for } i \in [1 \dots n], j = n+k, k \in \overline{X}, i \neq k \\ \langle \Sigma^* \cup (\Sigma^{-1})^* \rangle_{J_{kj}} & \text{for } j \in [1 \dots n], i = n+k, k \in \overline{X}, j \neq k \\ \varepsilon & \text{for } j = n+i, i \in \overline{X} \text{ or } i = n+j, j \in \overline{X} \\ \langle \Sigma^* \cdot a \rangle_{I_i} & \text{for } i \in [1 \dots n], j = n+k, k \in X \\ & \quad \text{or } i = n+l, j = n+k, k \in X, l \in \overline{X} \\ \langle a^{-1} \cdot (\Sigma^{-1})^* \rangle_{(-I_i)} & \text{for } j \in [1 \dots n], i = n+k, k \in X \\ & \quad \text{or } i = n+k, j = n+l, k \in X, l \in \overline{X} \end{cases}$$

Denote $\mathcal{L}$ the union of the semantics of all $2n$-regsignals $R(C,X)$ for all n-constraints C and subsets $\emptyset \neq X \subseteq [1 \dots n]$, $\mathcal{L} = \bigcup \{ \|R(C,X)\| \mid C \in \mathcal{C}(\Xi), \emptyset \neq X \subseteq [1 \dots n] \}$.

Proposition 9.3.1. *$\mathcal{L}$ satisfies the hypothesis in Theorem 7.4.3, that is, $\mathcal{L}^{k\odot}$ consists of non-elastic $2n$-signals for all $k \in \mathbb{N}$.*

Proof. Observe that sets $R(C,X)$ bear the following important property:

(*) For each $C \in \mathcal{C}(X)$, each $X \subseteq [1 \dots n]$ and $\sigma \in \|R(C,X)\|$, if $\sigma_{i,n+i} \neq \varepsilon$ then for all $j \in [1 \dots n]$, $\sigma_{j,n+i}$ is not an antisignal, that is, $\sigma_{j,n+i} \in \mathsf{Sig}(\Sigma)$.

We will then actually show that the set of all $2n$-signals with property (*) satisfies the hypothesis in Theorem 7.4.3:

Take k $2n$-signals $\sigma_1, \dots, \sigma_k$, all satisfying property (*). Suppose that $\sigma_1 \odot \dots \odot \sigma_k$ is an *elastic* $2n$-signal. That is, there exist $i, j \in [1 \dots n]$ such that

1. $(\sigma_1 \odot \dots \odot \sigma_k)_{i_0,n+i_0} \neq \varepsilon$, $(\sigma_1 \odot \dots \odot \sigma_k)_{j_0,n+j_0} \neq \varepsilon$ and
2. $(\sigma_1 \odot \dots \odot \sigma_k)_{i_0,n+j_0}$ is an antisignal, i.e. $(\sigma_1 \odot \dots \odot \sigma_k)_{i_0,n+j_0} \in \mathsf{Sig}(\Sigma^{-1}) \setminus \{\varepsilon\}$.

Let us show that $(\sigma_l)_{j_0,n+i_0}$ is an antisignal. We prove this by contradiction: since for all $l \in [1 \dots k]$, σ_h is a non-elastic $2n$-signal, we must then have $(\sigma_h)_{i_0,n+i_0} \in \mathsf{Sig}(\Sigma)$ and, similarly, $(\sigma_h)_{j_0,n+j_0} \in \mathsf{Sig}(\Sigma)$. But then

$$\begin{aligned} &(\sigma_1 \odot \dots \odot \sigma_k)_{j_0,n+i_0} = \\ &\qquad = (\sigma_1)_{j_0,n+j_0} \dots (\sigma_{l-1})_{j_0,n+j_0} (\sigma_l)_{j_0,n+i_0} (\sigma_{l+1})_{i_0,n+i_0} \dots (\sigma_k)_{i_0,n+i_0} \end{aligned}$$

hence $(\sigma_1 \odot \dots \odot \sigma_k)_{j_0,n+i_0}$ is a signal, fact which contradicts condition 2 above.

On the other hand, condition 1 above implies that there exists some $l_0 \in [1 \dots k]$ such that $(\sigma_{l_0})_{i_0,n+i_0} \neq \varepsilon$. But by property (*), we must have then $(\sigma_{l_0})_{j_0,n+i_0} \in \mathsf{Sig}(\Sigma)$, which we have already seen to be in contradiction with condition 2. □

Remark 9.3.2. Observe that using non-elasticity instead of property (*) does not suffice to prove the above result.

Observe also that Proposition 9.3.1 assures that each concatenation produces non-elastic languages.

10. Conclusions

We have presented an approach on checking timed automata for language emptiness, approach based on regular expressions. The regular expressions we use arise as a generalization of timed regular expressions of [ACM97], with the use of colored parentheses. Our method associates to each atomic regular expression a class of automata with $2n$ accepting sets, and then applies union/concatenation/star constructions to build an automaton representing the whole regular expression.

The essential steps that give this method are the following:

- The possibility to represent timing constraints over the continuous time domain with the aid of n-automata. This possibility is based upon region decomposition of each timing constraint, and on the representation of each region as a pair consisting of an n-word over a one-letter alphabet and a matrix of relational symbols.
- The star-closure theorem for $2n$-automata with the property that all the powers of its accepted language are composed of only non-elastic $2n$-words.
- The decomposition of each regular expression into the untimed and the timing part, decomposition which allows representing the timing part with $2n$-word representations. And then, the recomposition of the untiming part with the $2n$-word representation of the timing part, by means of the shuffle operation. This recomposition replaces a (perhaps uneasy) synchronous application of the union/concatenation/star constructions for both the untimed and the timing part, which is needed when the interactions between untiming and timing are more involved and may lead to emptiness – that is, to unfeasible specifications.

We hope that our study gives new insights in better understanding the theory of timed systems. The difficulty of the emptiness checking for timed automata keeps the performances modeling systems with timed automata and model-checking their properties far from the performances reached for untimed modeling. Therefore, any alternative insight might help in identifying subclasses with nice properties. Our theory of $2n$-signal regular expressions and $2n$-automata is such an alternative insight.

For the comparison of our approach to the emptiness problem for timed automata with the classical approaches [Yov98, LPWY95] based on the region construction of [AD94], we may observe the following points:

- The essential property that gives a terminating algorithm in the classical approach is the possibility to collapse the infinite region space into a finite one, which is of the cardinality of the

set of regions included into some hypercupe. The length of the edge of this hypercube is $k+1$, with k being computable from the largest constants used in the clock constraints of the given timed automaton. In the reference paper on timed automata [AD94] this constant is exactly the largest constant in any clock constraint, but this is due to the fact that clock constraints do not use diagonal atomic constraints $x_i - x_j \in I$. In general, this constant needs to be computed by "propagating" also the diagonal constraints [Tri98].

On the other hand, in our approach, termination is assured by the non-elasticity property of the $2n$-signals which give the $2n$-signal-semantics of a timed automaton, property which allows the possibility to iteratively build a finite representation for the reachability relation defined by the timed automaton. This still means that we get a finite decomposition of the set of clock regions, but this decomposition is "finer" (that is, gives in general more equivalence classes) than Alur's decomposition.

As a consequence, our algorithm might sometimes require more memory than the classical approach. Of contrary, sometimes our approach might give faster results due to the possibility to get the behavior of a loop in the timed automaton in a single application of the star closure algorithm, fact which is not available in the classical approach since there one needs to iteratively pass through the loop until the fixpoint is reached.

- The complexity of our algorithm is nonelementary, in contrast with the PSPACE complexity of the classical approach. This follows due to the fact that each star produces an exponential explosion of the state space. However this result concerns only the worst-case complexity, and is highly dependent on the number of nested stars in the regular expression that one may associate to a timed automaton.

Our contributions are partly theoretical, but with a certain interest for the domain of verification of timed systems. In this sense, our main contribution is the verification method for timed automata by translation to $2n$-signal regular expressions. It can be argued that building a regular expression from a timed automaton is a difficult task, a legitimate observation. But we expect to further study the possibility of directly translating the timed regular expressions of [ACM97] into our regular expressions and hence making available our technique without passing through timed automata.

This gives one of the directions for future research: the introduction of parallel composition in regular expressions with colored parentheses. Specifically, we would like to have some "distributivity" laws of parallel composition over atomic blocks. Such laws would allow transformation of parallel compositions of timed regular expressions of [ACM97] into regular expressions with colored parentheses. This would allow doing emptiness checking for timed regular expressions of [ACM97] without passing through timed automata.

Another promising direction is given by the applicability of n-automata in representing timing constraints. We have found that an argument in favor of n-automata is that they sometimes give a more compact representation of timing constraints. Unfortunately n-automata are nondeterministic, hence the problem of finding small representation of timing constraints is somewhat related to the problem of finding small nondeterministic finite automata. In particular, the union construc-

tion of several n-automata, which in theory is done by simply joining the state spaces, has to be accompanied by a technique which identifies some states, in order to reduce the state space.

We would also like to transform our closure algorithms for n-automata (that is, the union, concatenation and star closure algorithms) into symbolic algorithms. One might be tempted to say that this would bring us back to the existing symbolic algorithms for reachability of timed automata. But this is not true, because we would not code disjunctions of constraints. Symbolic algorithms would allow one to symbolically represent states in n-automata, and we have already pointed out that there is no connection between states in an n-automaton and clock regions.

Another direction of further study is the possibility to combine our emptiness checking technique with partial order reduction methods. For timed automata, partial order reductions involve the possibilty to split, at certain moments, the clock set into subsets of dependent clocks, such that two clocks belonging to different subsets are not related by any constraint during the respective moments. For our setting, this would mean to split a $2n$-regsignal into smaller regsignals and to do concatenation and star on these regsignals. Some care needs to be put in order to define a "parallel" composition of these smaller rsgsignals such that one does not obtain elastic regsignals.

And a final mention for the idea of finding more general classes of timed systems that may be modeled by $2n$-automata. We mainly think of hybrid automata with stopwatches [Hen96]. It has been already observed that preemptive scheduling can be modeled by such hybrid automata [AM]. The essence is that preemptive scheduling is intimately related to the *shuffle* operation, hence it remains to be studied how to model this into n-automata and what would be the resulting class of timed languages.

References

[ABK+97] E. Asarin, M. Bozga, A. Kerbrat, O. Maler, A. Pnueli, and A. Rasse. Data-structures for the verification of timed automata. In *Proceedings of HART'97*, volume 1201 of *LNCS*, pages 346–360, 1997.

[ACM97] E. Asarin, P. Caspi, and O. Maler. A Kleene theorem for timed automata. In *Proceedings of LICS'97*, pages 160–171, 1997.

[ACM01] E. Asarin, P. Caspi, and O. Maler. Timed regular expressions. submitted, 2001.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AFH94] R. Alur, L. Fix, and T. A. Henzinger. A determinizable class of timed automata. In *Proceedings of CAV'94*, volume 818 of *LNCS*, pages 1–13, 1994.

[AFH96] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43:116–146, 1996.

[AFH99] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: a determinizable class of timed automata. *Theoretical Computer Science*, 211:253–273, 1999.

[AH92] R. Alur and T. A. Henzinger. Back to the future: Towards a theory of timed regular languages. In *Proceedings of FOCS'92*, pages 177–186, 1992.

[AM] Y. Abdeddaïm and O. Maler. Preemptive job-shop scheduling using stopwatch automata. Submitted to TACAS 2002.

[AMP98] E. Asarin, O. Maler, and A. Pnueli. On discretization of delays in timed automata and digital circuits. In *Proceedings of Concur'98*, volume 1466 of *LNCS*, pages 470–484, 1998.

[Asa98] E. Asarin. Equations on timed languages. In *Proceedings of HSCC'98*, volume 1386 of *LNCS*, pages 1–12, 1998.

[BB91] J. C. M. Baeten and J. A. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3:142–188, 1991.

[BC96] A. Boudet and H. Comon. Diophantine equations, Presburger arithmetic and finite automata. In *Proceedings of CAAP'96*, volume 1059 of *LNCS*, pages 30–43. Springer Verlag, 1996.

[BDFP00a] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Are timed automata updatable? In *Proceedings of CAV'2000*, volume 1855 of *LNCS*, pages 464–479, 2000.

[BDFP00b] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Expressiveness of updatable timed automata. In *Proceedings of MFCS'2000*, volume 1893 of *LNCS*, pages 232–242, 2000.

[BDGP98] B. Bérard, V. Diekert, P. Gastin, and A. Petit. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36:145–182, 1998.

[BDM+98] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, , and S. Yovine. Kronos: a model-checking tool for real-time systems. In *Proceedings of CAV'98*, LNCS, pages 546–550, 1998.

[BE97] S. L. Bloom and Z. Ésik. Axiomatizing shuffle and concatenation in languages. *Information and Computation*, 139:62–91, 1997.

[Bel57] R. Bellmann. *Dynamic Programming*. Princeton University Press, 1957.

[Bir79] G. Birkhoff. *Lattice theory*. American Mathematical Society, Providence, R.I, 1979.

[BJLWY98] J. Bengtsson, B. Jonsson, J. Lilius, and Wang Yi. Partial order reductions for timed systems. In *Proceedings of CONCUR'98*, volume 1466 of *LNCS*, pages 485–500, 1998.

[BMT99] M. Bozga, O. Maler, and S. Tripakis. Efficient verification of timed automata using dense and discrete time semantics. In *Proceedings of CHARME'99*, volume 1703 of *LNCS*, pages 125–141, 1999.

[Boi99] B. Boigelot. *Symbolic Methods for Exploring Infinite State Spaces*. PhD thesis, Faculté des Sciences Appliquées de l'Université de Li'ege, 1999. volume 189, Collection des Publications de la Faculté des Sciences Appliquées de l'Université de Li'ege.

[BP99] P. Bouyer and A. Petit. Decomposition and composition of timed automata. In *Proceedings of ICALP'99*, volume 1644 of *LNCS*, pages 210–219, 1999.

[BP01] P. Bouyer and A. Petit. A Kleene/Büchi-like theorem for clock languages. *Journal of Automata, Languages and Combinatorics*, 2001. To appear.

[BPT01] P. Bouyer, A. Petit, and D. Thérien. An algebraic characterization of data and timed languages. In *Proceedings of CONCUR'2001*, volume 2154 of *LNCS*, pages 248–261, 2001.

[Bry86] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986.

[Büc60] J. Büchi. On a decision method in restricted second-order arithmetic. In *Proceedings of the Int. Congress on Logic, Methodology and Philosophy of Science*, 1960.

[CG00] Ch. Choffrut and M. Goldwurm. Timed automata with periodic clock constraints. *Journal of Automata, Languages and Combinatorics*, 5:371–404, 2000.

[CJ99] H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In *Proceedings of CONCUR'99*, volume 1664 of *LNCS*, pages 242–257, 1999.

[Con71] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.

[Dim99a] C. Dima. Automata and regular expressions for real-time languages. In *Proceedings of the 9th International Conference on Automata and Formal Languages (AFL'99)*, Vasszeczeny, Hungary, 1999.

[Dim99b] C. Dima. Kleene theorems for event-clock automata. In *Proceedings of FCT'99*, volume 1684 of *LNCS*, pages 215–225, 1999.

[Dim99c] C. Dima. Relating signals and timed traces. *Annals of the University of Bucharest*, XLVIII:79–89, 1999.

[Dim00a] C. Dima. Real-time automata and the Kleene algebra of sets of real numbers. In *Proceedings of STACS'2000*, volume 1770 of *LNCS*, pages 279–289, 2000.

[Dim00b] C. Dima. Removing epsilon transitions from event-clock automata. In *Proceedings of the National Conference on Theoretical Computer Science and Information Technology (CITTI'2000)*, pages 75–81, Constanţa, Romania, May 25-27, 2000.

[Dim01a] C. Dima. Distributed real-time automata. In C. Martin-Vide and V. Mitrana, editors, *Grammars and Automata for String Processing: from Mathematics and Computer Science to Biology, and Back*. Gordon and Breach, London, 2001. to appear.

[Dim01b] C. Dima. Real-time automata. *Journal of Automata, Languages and Combinatorics*, 6:3–23, 2001.

[DMP91] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[DP89] J. Dassow and G. Paun. *Regulated rewriting in formal language theory*, volume 18 of *EATCS Monographs on theoretical computer science*. Springer Verlag, 1989.

[Eil74] S. Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, 1974.

[EKR82] A. Ehrenfreucht, J. Karhumäki, and G. Rozenberg. The (generalized) Post Correspondence Problem with lists consisting of two words is decidable. *Theoretical Computer Science*, 21:119–144, 1982.

[Gau92] S. Gaubert. *Théorie des systèmes linéaires dans les dioïdes*. PhD thesis, École des Mines de Paris, 1992.

[Gau99] S. Gaubert. *Introduction aux systèmes dynamiques à événements discrets*. Polycopié de cours ENSTA–ENSMP–Orsay (DEA ATS), 1992 (revised 1999).

[GJ79] M. R. Garey and D. S. Johnson. *Computers and intractability*. W.H. Freeman & Co., 1979.

[GP97] S. Gaubert and Max Plus. Methods and applications of $(\max, +)$ linear algebra. In *Porceedings of STACS'97*, volume 1200 of *LNCS*, pages 261–282, 1997.

[Hen96] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of LICS'96*, pages 278–292, 1996.

[Her99] P. Herrmann. Renaming is necessary in timed regular expressions. In *Proceedings of FST&TCS'99*, volume 1738 of *LNCS*, pages 47–59, 1999.

[HHH99] V. Halava, T. Harju, and M. Hirvensalo. Generalized PCP is decidable for marked morphisms. In *Proceedings of FCT'99*, volume 1684 of *LNCS*, pages 304–315, 1999.

[HJ96] Dang Van Hung and Wang Ji. On the design of hybrid control systems using automata models. In *Proceedings of FST&TCS'96*, volume 1180 of *LNCS*, pages 156–167, 1996.

[HNSY94] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.

[HRS98] T. Henzinger, J.-F. Raskin, and P.-Y. Schobbens. The regular real-time languages. In *Proceedings of ICALP'98*, volume 1443 of *LNCS*, pages 580–591, 1998.

[HU92] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley/Narosa Publishing House, 1992.

[HZC97] M. R. Hansen and Zhou Chaochen. Duration calculus: Logical foundations. *Formal Aspects of Computing*, 9:283–330, 1997.

[Jur99] Y. Jurski. *Expression de la relation binaire d'accessibilité pour les automates à compteurs plats et les automates temporisés*. PhD thesis, École Normale Supérieure de Cachan, France, 1999.

[Kle56] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. Shannon and M. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.

[KMM97] M. Kudlek, S. Marcus, and A. Mateescu. Contextual grammars with distributed catenation and shuffle. In *Proceedings of FCT'97*, volume 1279 of *LNCS*, pages 269–280, 1997.

[Kop97] H. Kopetz. *Real-Time Systems. Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.

[Koz94] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110:366–390, 1994.

[LPWY95] K. G. Larsen, P. Pettersson, and Wang Yi. Model-checking for real-time systems. In *Proceedings of FCT'95*, volume 965 of *LNCS*, pages 62–88, 1995. Invited paper.

[LPY97] K. G. Larsen, Paul Petterson, and Wang Yi. Uppaal: Status & developments. In *Proceedings of CAV'97*, LNCS, pages 456–459, 1997.

[LWYP99] K. G. Larsen, C. Weise, Wang Yi, and J. Pearson. Clock difference diagrams. *Nordic Journal of Computing*, 6:271–298, 1999.

[Mac71] S. MacLane. *Categories for the Working Mathematician*. Springer Verlag, 1971.

[Mil80] R. Milner. *Communication and Concurrency*, volume 92 of *LNCS*. Springer Verlag, 1980.

[MLAH99] J. Møller, J. Lichtenberg, H. R. Andersen, and H. Hulgaard. Difference decision diagrams. Technical Report IT-TR-1999-023, Department of Information Technology, Technical University of Denmark, 1999.

[MP90] Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *Proceedings of PODC'90*, pages 377–410, 1990.

[MP92] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems: Specification*. Springer Verlag, 1992.

[MP95] Z. Manna and A. Pnueli. *Temporal verification of reactive systems: Safety*. Springer Verlag, 1995.

[Nic92] X. Nicollin. *ATP: une algèbre pour la spécification et l'analyse des systèmes temps réel*. PhD thesis, Institut National Polytechinque de Grenoble, 1992.

[NSY93] X. Nicollin, Joseph Sifakis, and Sergio Yovine. From ATP to timed graphs and hybrid systems. *Acta Informatica*, 30:181–202, 1993.

[Pos46] E. Post. A variant of a recursively unsolvable problem. *Bull. AMS*, 52:264–268, 1946.

[Pra90] V. R. Pratt. Dynamic algebras as a well-behaved fragment of relation algebras. In *Algebraic Logic and Universal Algebra in Computer Science*, volume 425 of *LNCS*, pages 77–110, 1990.

[Rab] A. Rabinovich. Automata over continuous time. Available at `http://www.math.tau.ac.il/~rabinoa/aut-cont.ps.gz`.

[Ras99] J.-F. Raskin. *Logics, Automata and Classical Theories for Deciding Real-Time*. PhD thesis, Facultés Universitaires Notre Dame de la Paix, Namur, Belgique, 1999.

[RS97] J.-F. Raskin and P.-Y. Schobbens. State-clock logic: a decidable real-time logic. In *Proceedings of HART'97*, volume 1201 of *LNCS*, pages 31–47, 1997.

[Saf88] S. Safra. On the complexity of omega-automata. In *Proceedings of FOCS'88*, pages 319–327, 1988.

[Sal66] A. Salomaa. Two complete axiom systems for the algebra of regular events. *Journal of ACM*, 13:158–169, 1966.

[Sor01] M. Sorea. Tempo: A model checker for event-recording automata. In *Proceedings of RT-Tools'01*, 2001.

[Sta97] L. Staiger. ω-languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, Beyond words, pages 339–387. Springer Verlag, 1997.

[Tho90] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–191. Elsevier, 1990.

[Tho97] W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, Beyond Words, pages 389–455. Springer Verlag, 1997.

[Tri98] S. Tripakis. *L'analyse formelle des systèmes temporisés en pratique*. PhD thesis, Université Joseph Fourier Grenoble, 1998.

[vH89] P. van Hentenryck. *Constraint Satisfaction in Logic Programming*. MIT Press, 1989.

[VW94] M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1–37, 1994.

[WB95] P. Wolper and B. Boigelot. An automata-theoretic approach to Presburger arithmetic constraints. In *Proceedings of SAS'95*, volume 983 of *LNCS*, pages 21–32, 1995.

[WB00] P. Wolper and B. Boigelot. On the construction of automata from linear arithmetic constraints. In *Proceedings of TACAS'99*, volume 1785, pages 1–19, 2000.

[Wil94] T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *Proceedings of FTRTFT'94*, volume 863 of *LNCS*, pages 694–715, 1994.

[WY91] Wang Yi. CCS + Time = An interleaving model for real time systems. In *Proceedings of ICALP'91*, volume 510 of *LNCS*, pages 217–228, 1991.

[Yov98] S. Yovine. Model-checking timed automata. In *Lectures on Embedded Systems*, volume 1494 of *LNCS*, pages 114–152, 1998.

[ZCHR91] Zhou Chaochen, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Information Processing Letters*, 40:269–276, 1991.

[ZCHS93] Zhou Chaochen, M. R. Hansen, and P. Sestoft. Decidability and undecidability results for duration calculus. In *Proceedings of STACS'93*, volume 665 of *LNCS*, pages 58–68, 1993.

[ZDPK01] Zhe Dang, P. San Pietro, and R.A. Kemmerer. Presburger liveness verification of discrete timed automata. In *Proceedings of STACS'01*, volume 2102 of *LNCS*, pages 132–143, 2001.

Index

Glossary

$\mathsf{BiSig}(\Sigma)$, 84

$D \sqsubseteq R$, for $D \in \mathcal{D}\mathsf{bm}_n$ and $R \in \mathcal{R}\mathsf{egn}_n$, 165
$D_n(\Sigma)$, the set of n-dominoes, 84
$\mathcal{D}\mathsf{bm}_n$, 158
$\mathcal{DL}_n(\Sigma)$, 96
$\mathcal{D}\mathsf{nf}_n$, 159
$\|D\|$, for $D \in \mathcal{E}\mathsf{dbm}_n$, 158
dom, 22

$\mathcal{EC}(\Xi_n)$, 56
$\mathcal{E}\mathsf{dbm}_n$, 158
$|E|$, for E a classical regular expression, 111
$\equiv_n$, 176
$\|E\|$, for E a $2n$-word regular expression, 111
$\|E\|$, for a timed regular expression E, 70
$\|E\|$, for $E \in \mathsf{RegD}_{2n}(\Sigma)$, 101
$\|E\|$, for a real-time regular expression E, 36
$\|E\|$, for an n-clocked regular expression E, 60
$\|E\|_s$, for $E \in \mathsf{RegSig}_{2n}(\Sigma)$, 102

$\mathbb{Q}Int$, 22
$\mathcal{K}(\mathbb{Q}Int)$, 41
$\mathbb{Z}Int$, 22

$L(E)$, for $E \in \mathsf{RegSig}_{2n}(\Sigma)$, 103
$L(\rho)$, 58
$L^{\circledast}$, for $L \subseteq \mathcal{DL}_{2n}(\Sigma)$, 96
L^{-1}, for $L \subseteq \mathsf{Sig}(\Sigma)$, 84
$L_{rep}(\mathcal{A})$, 181
$L_{rgn}(\mathcal{A})$, 181
$L_{sig}(\mathcal{A})$, 182
ℓ, 22, 199
$\ell(\sigma)$, for $\sigma \in \mathsf{Sig}_n(\Sigma)$, 151
l_X, 86

$M\big|_X$, for $M \in \Gamma_n$, 171
$M_1 \square_p M_2$, for $M_1 \in \Gamma_m$ and $M_2 \in \Gamma_n$, 171
$M_1 \odot M_2$, for $M_1, M_2 \in \Gamma_{2n}$, 174

$\mathbb{N}$, 22
$\mathbb{Z}$, 22
$\mathbb{Q}_{\geq 0}$, 22

$X + Y$, for $X, Y \subseteq \mathbb{R}$, 40
X^*, for $X \subseteq \mathbb{R}$, 40
$\mathcal{P}(X)$, 24

$R_1 \odot R_2$, for $R_1, R_2 \in \mathcal{R}\mathsf{egn}_{2n}$, 167
$\mathsf{RegD}_{2n}(\Sigma)$, 101
$\mathcal{RD}_n(\Sigma)$, 98
$\mathcal{RS}ig_n(\Sigma)$, 98
$\mathcal{RW}_n(\Sigma)$, 111
$\mathsf{Rat}(\Sigma_{\mathbb{Q}Int})$, 35
$\mathcal{R}\mathsf{egn}_n$, 164
$\mathbb{R}_{\leq 0}$, 22
$\mathbb{R}_{\geq 0}$, 22
$\mathbb{R}_{>0}$, 22
$\mathsf{RegSig}_{2n}(\Sigma)$, 101
$|R|$, for $R \in \mathcal{RW}_n(\Sigma)$, 111
$\|R\|$, for $R \in \mathcal{RD}_n(\Sigma)$, 98
$\tilde{\rho}$, 27

$TRec(\Sigma)$, 34
$\mathsf{SW}_n(\Sigma)$, 201
$\mathsf{Sig}(\Sigma^{-1})$, 84
$\mathsf{Sig}_n(\Sigma)$, the set of n-signals, 84
$\mathsf{Sigclk}(\Sigma)$, 59
$\mathsf{Sigreset}(\Sigma)$, 66
$\mathcal{U}$, 28
σ^{-1}, for $\sigma \in \mathsf{Sig}(\Sigma)$, 84
$SF(\Sigma)$, 27
σ_ϵ, 23
$\mathsf{Sig}(\Sigma)$, 22
$(\gamma, M)\big|_X$, 201
$[\![\gamma, M]\!]$, for $(\gamma, M) \in \mathsf{SW}_n(\Sigma)$, 201

$\mathcal{U}$, 199

$[W, M]$, for $(W, M) \in \mathcal{W}\mathbb{R}_n \times \Gamma_n$, 176
$\mathcal{UD}_n(\Sigma)$, 111
$\mathcal{WD}_n(\Sigma)$, 111
$\mathcal{WL}_n(\Sigma)$, 96
$\overset{\frown}{\cdot}$, 27, 199
$w \sqcup\!\sqcup w'$, for $w, w' \in \mathcal{WD}_n(\Sigma)$, 200
$w \square_p w'$, for $w \in D_m(\Sigma)$, $w' \in D_n(\Sigma)$, 88
$w \odot w'$, for $w, w' \in D_{2n}(\Sigma)$, 93
$w\big|_X$, for $w \in D_n(\Sigma)$, 87
$w \sqcup\!\sqcup w'$, for $w, w' \in \Sigma^*$, 199
$\mathsf{word}(\rho, i_1, i_2)$, 112

Résumé. Un automate temporisé est un automate augmenté avec plusieurs horloges qui mesurent le passage de temps et peuvent conditionner la modification de l'état du système. Les automates temporisés ont été introduits en tant que modèle formel pour les systèmes temps-réel, en espérant que leur rôle dans la vérification de tels systèmes sera similaire au rôle des automates finis dans la recherche systématique des erreurs de conception de systèmes non-temporisés. Dans notre thèse nous étudions plusieurs questions théoriques liés aux automates temporisés et aux langages temporisés.

Dans une première partie nous étudions une sous-classe simple d'automates temporisés à une seule horloge qui est remise à zéro pendant chaque transition. Nous montrons que cette sous-classe supporte des résultats similaires à la théorie classique des automates finis: des théorèmes de Kleene, de Myhill-Nerode et de fermeture par complémentation.

La deuxième et principale partie de la thèse est motivée par les expressions régulières temporisés de Asarin, Caspi et Maler. Depuis leur introduction, on sait qu'il faut employer l'intersection dans les expressions régulières pour que leur expressivité soit égale aux automates temporisés. Nous poursuivons alors une approche alternative en utilisant des parenthèses colorées pour définir les contraintes temporelles sur une séquence d'événements. Cette idée aboutit à une représentation alternative des langage des automates temporisés, basée sur une nouvelle classe de langages formels que nous appelons *langages des regminos*. Nous développons alors la théorie des expressions régulières sur les regminos et nous montrons que le problème de sémantique vide est indécidable en cas général, et décidable pour une sous-classe large de langages. L'application de ces résultats nous amène à des nouvelles structures de données et à des algorithmes pour le problème du langage vide dans les automates temporisés et les expressions régulières.

Mots clés. automate temporisé, expressions régulières, théorème de Kleene, contraintes temporelles, decidabilité, langages formels.

Title. An algebraic theory of real-time formal languages.

Abstract. A timed automaton is an automaton augmented with several clocks that measure the time passage and may influence state changes in the system. Timed automata were introduced as a formal model for real-time systems hoping that their role in the verification of such systems will be similar to the role of finite automata in the systematic search of errors in the design of untimed systems. In our thesis we are concerned with several theoretical questions related to timed automata and timed languages.

In the first part of the thesis we investigate a simple sub-class of timed automata with one clock which is reset at each transition. We show that for this sub-class we can obtain simple analogs of the classical results of automata theory, namely Kleene and Myhill-Nerode theorems and closure under complementation.

The second and main part of the thesis is motivated by the timed regular expressions of Asarin, Caspi and Maler where it was shown that, in order to match the expressive power of timed automata, one needs to introduce intersection into the expressions. We investigate an alternative to intersection by using colored parentheses for defining timing restrictions on overlapping parts of a sequence. This idea leads to an alternative representation of languages of timed automata, which is based on a new class of languages called here *regmino languages*. We develop the theory of regular expressions over regminoes and prove that their emptiness problem is undecidable in general, and decidable for a large subclass of languages. From these results we develop new data-structures and algorithms for solving emptiness and reachability problems for timed automata and regular expressions.

Keywords. timed automata, regular expressions, Kleene theorem, timing constraints, decidability, formal languages.

Adresse du laboratoire d'accueil. Vérimag, 2 av. de Vignate, 38610 Gières.