Partial Commutation and Traces

Volker Diekert
Institut für Informatik, Universität Stuttgart
Breitwiesenstr. 20-22, D-70565 Stuttgart, Germany

Yves Métivier
LaBRI, Université Bordeaux I, ENSERB,
351 cours de la Libération, F-33405 Talence, France

Table of Contents

DRAFT January 9, 1996

1 Introduction

Parallelism and concurrency are a fundamental concepts in computer science. Specification and verification of concurrent programs are of first importance. It concerns our daily life whether software written for distributed systems behaves correctly.

It is clear that a satisfactory notion of correctness has to be based on a rigorous mathematical model. Several formalisms have been proposed. Among others there are Petri nets, Hoare's and Milner's CSP and CCS, event structures, and branching temporal logics. The mathematical analysis of these models may become however sometimes very complicated. Based on the behavior of elementary net systems Mazurkiewicz introduced the concept of partial commutation to the computer science community. The abstract description of a concurrent process is called a *trace*, being defined as a congruence class of a word (sequence) modulo identities of the form $ab = ba$ for some pairs of letters.

The success of Mazurkiewicz' approach is that on one hand partial commutation copes with some important phenomena in concurrency and on the other hand it is close to the classical theory of free monoids describing sequential programs. In particular it is possible to transfer the notion of finite sequential state control to the notion of finite asynchronous state control. There is a satisfactory theory of recognizable languages relating finite semigroups, c-rational operations, asynchronous automata, and logic. This leads to decidability results and effective operations.

The theory of partial commutation and of trace monoids has been developed both by its interpretation as a model for parallel computation and by its mathematical interest in algebra, formal languages, and combinatorics. Since the beginning in combinatorics by Cartier and Foata (1969) and the formulation of trace theory by Mazurkiewicz (1977) the theory has grown in breadth and depth. It led to significant results with interesting applications. The present contribution reflects some important topics including basic properties and infinite traces. Most of the results are from the monograph [34], but we covered also some new material. Each section gives a short bibliographical remark and leads to further references.

Acknowledgments

This work has been partially supported by the ESPRIT Basic Research Action No. 6317 ASMICS II and the French-German research program PROCOPE. We indebted to Michael Bertol, Manfred Droste, Paul Gastin, Anca Muscholl and Holger Petersen. Without the help of Michael Bertol the manuscript would not have been written in time. Last but not least, we thank Professors Grzegorz Rozenberg and Arto Salomaa for including our contribution in the present volume.

2 Free Partially Commutative Monoids

2.1 First Definitions and Basic Properties

Let Σ be a finite alphabet, its elements are called letters. We denote by Σ^* the set of all words over Σ. Formally, Σ^* with the concatenation operation forms the free monoid with the set of generators Σ, the empty word, denoted by 1, plays the role of the unit element. For any word x of Σ^*, $|x|$ denotes the length of x, $|x|_a$ denotes the number of occurrences of a letter a in x. The notation $\mathrm{alph}(x) = \{a \in \Sigma \mid |x|_a \neq 0\}$ is used for the set of letters of Σ actually appearing in the word x.

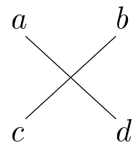
Throughout we mean by $I \subseteq \Sigma \times \Sigma$ a symmetric and irreflexive relation over the alphabet Σ, called the *independence* (or *commutation)* relation. Intuitively, $(a, b) \in I$ means that a and b act on disjoint sets of resources. As a consequence, the order in which they are performed does not matter, $ab = ba$. They can also be performed in parallel or simultaneously.

For any letter a of Σ, we denote by $I(a)$ the set of letters which commute with a :

$$I(a) = \{b \in \Sigma \mid (a, b) \in I\}.$$

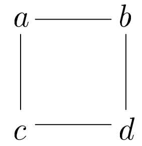
The pair (Σ, I) is called the *independence alphabet* and can be conveniently represented by an undirected graph (also called the *commutation graph*). The vertex set is Σ, edges are between independent letters.

Example 1. Let $\Sigma = \{a, b, c, d\}$ and $I = \{(a, d), (d, a), (b, c), (c, b)\}$, then the commutation graph is given in the picture below.

$(\Sigma, I) \quad =$ 

The complement $D = \Sigma \times \Sigma \setminus I$ of I is called the *dependence* relation. Intuitively, two letters a and b such that $(a, b) \notin I$ are dependent and cannot be executed simultaneously.

The pair (Σ, D) is called the *dependence alphabet*. Again, we identify (Σ, D) with an undirected graph (the *non-commutation graph*). In the pictures we omit however the self-loops. If we take the same example as above, the dependence alphabet is given by the following non-commutation graph.

$(\Sigma, D) \quad =$ 

For any letter a of Σ, $D(a)$ denotes the set of letters which depend on a:

$$D(a) = \{b \in \Sigma \mid (a, b) \notin I\}.$$

Since I is irreflexive, we have $a \in D(a)$. We extend this notation for all $A \subseteq \Sigma$ by setting:

$$D(A) = \{b \in \Sigma \mid \exists a \in A : (a, b) \in D\}.$$

The relation I induces an equivalence relation $\sim_I$ over Σ^*. Two words x and y are equivalent under $\sim_I$, denoted by $x \sim_I y$, if there exists a sequence $z_1, z_2, \ldots, z_k$ of words such that $x = z_1$, $y = z_k$, and for all i, $1 \leq i < k$, there exist words z_i', z_i'', and letters a_i, b_i satisfying:

$$z_i = z_i' a_i b_i z_i'', \quad z_{i+1} = z_i' b_i a_i z_i'', \text{ and } (a_i, b_i) \in I.$$

Thus, two words are equivalent by $\sim_I$ if one can be obtained from the other by successive transpositions of neighboring independent letters. It is easy to verify that $\sim_I$ is the least congruence over Σ^* such that $ab \sim_I ba$ for all pairs $(a, b) \in I$. The quotient of Σ^* by the congruence $\sim_I$ is the *free partially commutative monoid* induced by the relation I, it is denoted by $\mathbb{M}(\Sigma, I)$. The elements of $\mathbb{M}(\Sigma, I)$, which are equivalence classes of words of Σ^* under the relation $\sim_I$, are called *traces*. Consequently, $\mathbb{M}(\Sigma, I)$ is called *trace monoid*, too.

If I is empty, then $\mathbb{M}(\Sigma, I)$ is the free monoid Σ^*; if any two different letters of Σ commute then $\mathbb{M}(\Sigma, I)$ is the free commutative monoid denoted $\mathbb{N}^{\Sigma}$. Other families of free partially commutative monoids are the direct products of free monoids:

$$\Sigma_1^* \times \Sigma_2^* \times \cdots \times \Sigma_n^*$$

and free products of free commutative monoids:

$$\mathbb{N}^{k_1} * \mathbb{N}^{k_2} * \cdots * \mathbb{N}^{k_n}$$

For a word x of Σ^* the equivalence class of x under $\sim_I$ is denoted by $[x]_I$. Thus, $[x]_I$ is the set of words which are equivalent to a given word x, hence

$$[x]_I = \{y \in \Sigma^* \mid y \sim_I x\}.$$

For instance, if we consider I defined above, we have:

$$[baadcb]_I = \{baadcb, baadbc, badacb, badabc, bdaabc, bdaacb\}.$$

Since the relation $\sim_I$ is a congruence, the concatenation in $\mathbb{M}(\Sigma, I)$ is given by $[x]_I[y]_I = [xy]_I$ for all $x, y \in \Sigma^*$. The class of the empty word is called the empty trace, also denoted by 1. The definition of length, of a-length, and of the alphabet is invariant under the commutation of letters, hence it can be transferred to a trace from any representing word. We can write $|t|$, $|t|_a$, and $\mathrm{alph}(t)$ for a trace t and a letter a. Following the same example above with $t = [baadcb]_I$ we have $|t| = 6$, $|t|_a = |t|_b = 2$, $|t|_c = |t|_d = 1$, and $\mathrm{alph}(t) = \{a, b, c, d\}$. A trace t (word x, resp.) is called *connected,* if $\mathrm{alph}(t)$ ($\mathrm{alph}(x)$, resp.) induces a connected subgraph of (Σ, D). The trace $t = [baadcb]_I$ above is connected, but $[ad]_I$ is not.

Two traces u and v of $\mathbb{M}(\Sigma, I)$ are said to be *independent* and this fact is denoted by uIv, if

$$\mathrm{alph}(u) \times \mathrm{alph}(v) \subseteq I.$$

Independence of traces is equivalent to the following conditions:

$$uv = vu \quad \text{and} \quad \mathrm{alph}(u) \cap \mathrm{alph}(v) = \emptyset.$$

Trace monoids find their place between free and free commutative monoids. There are two canonical homomorphisms:

$$\begin{aligned} \varphi : \Sigma^* &\longrightarrow \mathbb{M}(\Sigma, I) \\ x &\longmapsto [x]_I \end{aligned}$$

and

$$\begin{aligned} \pi : \mathbb{M}(\Sigma, I) &\longrightarrow \mathbb{N}^{\Sigma} \\ t &\longmapsto (|t|_a)_{a \in \Sigma}. \end{aligned}$$

The image $\pi(t) \in \mathbb{N}^{\Sigma}$ is called the *Parikh-image* of $t \in \mathbb{M}(\Sigma, I)$. The composition

$$\Sigma^* \quad \xrightarrow{\varphi} \quad \mathbb{M}(\Sigma, I) \quad \xrightarrow{\pi} \quad \mathbb{N}^{\Sigma}$$

is the usual Parikh-mapping from words to vectors.

A *trace language* is any subset of $\mathbb{M}(\Sigma, I)$. If $X \subseteq \Sigma^*$, then the trace language $\varphi(X) \subseteq \mathbb{M}(\Sigma, I)$ is also denoted by $[X]$. This allows to write

$$[X] = \{[x]_I \mid x \in X\}.$$

For any trace language $T \subseteq \mathbb{M}(\Sigma, I)$, the language $\varphi^{-1}(T)$ consists of all representing words. The operation $X \mapsto \varphi^{-1}([X])$ can be viewed as a closure operation on languages of Σ^* in a topological sense and in the sense of closure modulo the partial commutation I. Therefore we prefer to write $\overline{X}$ instead of $\varphi^{-1}([X])$ or $\varphi^{-1}(\varphi(X))$. A subset X of Σ^* is said to be *I-closed* (or simply *closed*, if there is no ambiguity), if $\overline{X} = X$.

For example, let I be defined as in Ex. 1, if $X = (ad)^*$ then

$$\overline{X} = \{x \in \{a, d\}^* \mid |x|_a = |x|_d\}.$$

For sake of simplicity we sometimes use words to denote the corresponding trace. Thus, if $x \in \Sigma^*$ and the context $\mathbb{M}(\Sigma, I)$ is clear then we may write $x \in \mathbb{M}(\Sigma, I)$ to denote in fact the class $[x]_I$. Following the same convention, we view Σ as a subset of $\mathbb{M}(\Sigma, I)$, too.

2.2 Projection Techniques and Levi's Lemma

Let $A \subseteq \Sigma$ be a subset and $I_A = (A \times A) \cap I$ be the induced independence relation. We may define a canonical homomorphism $\pi_A : \mathbb{M}(\Sigma, I) \to \mathbb{M}(A, I_A)$ by erasing all letters from a trace which do not belong to A, hence for $a \in \Sigma$ we have:

$$\pi_A(a) = a, \text{ if } a \in A \text{ and } \pi_A(a) = 1 \text{ otherwise.}$$

Of particular interest is $I_A = \emptyset$. Then A is a clique of the dependence alphabet and π_A a projection onto the free monoid A^*. This is in particular the case when $A = \{a, b\}$ and $(a, b) \in D$. (We shall write $\pi_{a,b}$ instead of $\pi_{\{a,b\}}$.) The following proposition is called *Projection Lemma*. It states that every trace has a unique representation as a tuple of words. For (Σ, I) fixed, this canonical representation is computable in linear time from any word defining the congruence class of a trace.

Proposition 1. *Let u and v be two traces of $\mathbb{M}(\Sigma, I)$; then $u = v$ if and only if we have*

$$\pi_{a,b}(u) = \pi_{a,b}(v) \quad \textit{for all } (a, b) \in D.$$

Proof. Clearly, the condition is necessary. Conversely, we prove by induction on the length of u (and of v). If u or v is the empty trace we have nothing to do. Let us suppose $u = au'$ and $v = bv'$, where a and b are letters of Σ and where u' and v' are traces of $\mathbb{M}(\Sigma, I)$.

We assume first $a \neq b$. Since $|u|_a = |v|_a$, we have $|v|_a \geq 1$ and $v' = t'at''$ with $|t'|_a = 0$. If $(a, c) \in D$, then $\pi_{a,c}(u) = \pi_{a,c}(v)$ and hence $(bt')Ia$. Finally $v = bt'at'' = abt't''$. Thus, we have reduced to the case $a = b$, or $u = au'$ and $v = av''$ for some traces u', v''. The inductive hypothesis yields $u' = v''$, hence $u = v$.

Since direct products of free monoids are cancellative, we deduce the same property for trace monoids.

Corollary 2. *The monoid $\mathbb{M}(\Sigma, I)$ is cancellative, i.e., the equation $uxv = uyv$ implies $x = y$ for all $u, v, x, y \in \mathbb{M}(\Sigma, I)$.*

Another corollary shows the following version of an embedding theorem.

Corollary 3. *Let $(\Sigma, D) = \bigcup_{i=1}^{k} (A_i, D_i)$ be a union of subalphabets with $I_i = (A_i \times A_i) \setminus D_i$, $\mathbb{M}_i = \mathbb{M}(A_i, I_i)$, and $\pi_i : \mathbb{M}(\Sigma, I) \to \mathbb{M}_i$ the canonical homomorphism for $1 \leq i \leq k$. Then we obtain a canonical injective homomorphism*

$$\begin{array}{ccc} \mathbb{M}(\Sigma, I) & \longrightarrow & \mathbb{M}_1 \times \cdots \times \mathbb{M}_k \\ t & \longmapsto & (\pi_1(t), \cdots, \pi_k(t)). \end{array}$$

The following proposition is called *Levi's Lemma.* It is one of the most important tools in trace theory.

Proposition 4. *Let t, u, v, and w be traces of $\mathbb{M}(\Sigma, I)$. Then the following assertions are equivalent.*

i) $tu = vw$
ii) *There exist $p, q, r, s \in \mathbb{M}(\Sigma, I)$ such that*
$t = pr,\ u = sq,\ v = ps,\ w = rq$ *with* $alph(r) \times alph(s) \subseteq I$.

Proof. Clearly, $ii)$ implies $i)$. Conversely, we use an induction on the length of t. It t is the empty trace then the property is true with $p = r = 1$, $q = w$ and $s = v$. Therefore we may assume that $t = at'$ for some $a \in \Sigma$.

– If $|v|_a = 0$ then necessarily aIv and $w = aw'$. As $at'u = vaw'$ and aIv we have $t'u = vw'$. Now applying the inductive hypothesis we find $p, q, r', s \in \mathbb{M}(\Sigma, I)$ such that

$$t' = pr', \quad u = sq, \quad v = ps, \quad w' = r'q \quad \text{and} \quad r'Is.$$

Let $r = ar'$, as aIv, we also have aIp and aIs. Hence

$$t = at' = apr' = par' = pr$$

with rIs, and the result follows.
– If $|v|_a > 0$, then $v = av'$. In this case we apply the inductive hypothesis with $t'u = v'w$ we find p', q, r, s such that $t' = p'r, u = sq, v = p's, w = rq$ and rIs. The desired formula is obtained considering $t = at'$ and $p = ap'$.

From this proposition, using an induction, we obtain the following more general form of Levi's Lemma:

Corollary 5. *Let $u, v, t_1, \ldots, t_n \in \mathbb{M}(\Sigma, I)$ be traces. Then the following assertions are equivalent.*

i) $uv = t_1 \cdots t_n$,

ii) There exist $p_1, \ldots, p_n, q_1, \ldots, q_n \in \mathbb{M}(\Sigma, I)$ *such that* $u = p_1 p_2 \cdots p_n$, $v = q_1 \cdots q_n$ *with* $p_i q_i = t_i$ *and* $q_i I (p_{i+1} \ldots p_n)$ *for all* $1 \leq i < n$.

Together with the existence of a positive weight, Levi's Lemma characterizes free partially commutative monoids. By a *positive weight* we mean any homomorphism $\gamma : \mathbb{M} \to \mathbb{N}$ such that $\gamma^{-1}(0) = \{1\}$. For trace monoids the length-function is a positive weight.

Proposition 6. *A monoid* M *with a positive weight is isomorphic to a free partially commutative monoid if and only if for all* $x, y, z, t \in M$ *the equation* $xy = zt$ *implies the existence of* $r, u, v, s \in M$ *such that* $x = ru$, $y = vs$, $z = rv$, $t = us$, *and* $uv = vu$.

Proof. We need a proof only for the if-part. Let $\Sigma = (M \setminus \{1\}) \setminus (M \setminus \{1\})^2$. A straightforward verification shows that Σ generates M and that it is contained in any set generating M: it is the minimal generating set of M. We define the independence relation I_M on Σ by $(x, y) \in I_M$, if $yx = xy$ in M.

Let φ be the canonical surjective homomorphism from Σ^* onto M; it induces a surjective homomorphism $\overline{\varphi} : \mathbb{M}(\Sigma, I_M) \to M$. We prove by induction on the weight that $\overline{\varphi}$ is injective. Let $t', t'' \in \mathbb{M}(\Sigma, I_M)$ such that $\overline{\varphi}(t') = \overline{\varphi}(t'')$. Using Noetherian induction, we assume that every proper factor of t' has exactly one inverse image. If $t' \in \Sigma \cup \{1\}$ then $t' = t''$ since $\overline{\varphi}^{-1}(x) = \{x\}$ for all $x \in \Sigma \cup \{1\}$. If $t' \notin \Sigma \cup \{1\}$ then $t' = xy, t'' = zt$ with $x, z \in \Sigma$ and $y \neq 1, t \neq 1$. Since $\overline{\varphi}(x)\overline{\varphi}(y) = \overline{\varphi}(z)\overline{\varphi}(t)$ and $\overline{\varphi}$ is surjective, the hypothesis about M implies that

$$\overline{\varphi}(x) = \overline{\varphi}(r)\overline{\varphi}(u), \quad \overline{\varphi}(y) = \overline{\varphi}(v)\overline{\varphi}(s), \quad \overline{\varphi}(z) = \overline{\varphi}(r)\overline{\varphi}(v), \quad \overline{\varphi}(t) = \overline{\varphi}(u)\overline{\varphi}(s)$$

$$\text{and} \quad \overline{\varphi}(uv) = \overline{\varphi}(vu),$$

where r, u, v, s are traces of $\mathbb{M}(\Sigma, I_M)$. Since $\overline{\varphi}(x), \quad \overline{\varphi}(y), \overline{\varphi}(z), \quad \overline{\varphi}(t)$ are proper factors of $\overline{\varphi}(t')$, we have

$$x = ru, \quad y = vs, \quad z = rv, \quad \textit{and} \quad t = us.$$

- If $r \neq 1$ or $s \neq 1$, then $\overline{\varphi}(uv)$ is a proper factor of $\overline{\varphi}(t')$, it follows $uv = vu$ in $\mathbb{M}(\Sigma, I_M)$. Hence $t' = ruvs = rvus = t''$.
- If $r = s = 1$, then $u = x \in \Sigma$ and $v = z \in \Sigma$; thus, $u, v \in \Sigma$ are independent elements and $t' = uv = vu = t''$ are the same trace by the very definition of I_M.

2.3 Normal Forms

Two main normal forms are defined in a free partially commutative monoid, the *lexicographic* normal form (studied first by Anisimov and Knuth [4]) and the *Foata* normal form. We assume that the alphabet Σ is totally ordered, and we consider the corresponding lexicographic ordering $<$ on Σ^*. Let X be a set of words, the unique minimal element of X with respect to the lexicographic ordering (if it exists) is denoted by $\mathrm{Min}(X)$. A word x is said to be in the lexicographic normal form if it is minimal among the set of words which are equivalent to x, i.e.,

$$x = \mathrm{Min}([x]).$$

As the lexicographic ordering is a total order and $[x]$ is finite, each trace has a unique minimal representative.

Let I be the independence relation of Ex. 1 above, assuming that $a < b < c < d$, then the word *baadbc* is in lexicographic normal form.

Theorem 7. *A word x is the lexicographic normal form of a trace if and only if for all factorizations $x = ybuaz$, where $y, u, z \in \Sigma^*$, $(a, b) \in I$, and $a < b$, there exists a letter of u which does not commute with a.*

Proof. If x is minimal then, clearly, it satisfies the condition of the statement.

Conversely, suppose there is a word w equivalent to x such that $w < x$; then $x = x'bx''$ and $w = x'az'$ with $a < b$. The words x and w are equivalent, thus aIb. Since $|x|_a = |w|_a$ we have $x'' = x_1ax_2$ with $|x_1|_a = 0$. Now from the equivalence between x and w we deduce that aIx_1. Therefore x does not satisfy the condition of the proposition.

From this we deduce immediately that the set *LexNF* of words in lexicographic normal form is a regular language; indeed it is equal to the following (star-free) set

$$\mathit{LexNF} = \Sigma^* \setminus \bigcup_{(a,b)\in I, a<b} \Sigma^* bI(a)^* a\Sigma^*.$$

A word x of Σ^* is in the Foata normal form, if it is the empty word or if there exist an integer $n > 0$ and non-empty words x_i $(1 \leq i \leq n)$ such that

i) $x = x_1 \cdots x_n$,
ii) for each i, the word x_i is a product of pairwise independent letters and x_i is minimal with respect to the lexicographic ordering,
iii) For each $1 \leq i < n$ and for each letter a of x_{i+1} there exists a letter b of x_i such that $(a, b) \in D$.

If we consider the factorization of Foata given above, each x_i is called a step. Let I be defined by the first picture, the Foata normal form of *baadcb* is $(b)(ad)(a)(bc)$.

Theorem 8. *Every trace has a unique Foata normal form.*

Proof. Let x be a word of Σ^*. We prove the existence of a normal form by induction on the length of x. The result is trivial if $x = 1$. Let $x = x'a$ with $a \in \Sigma$; let $x'_1x'_2 \cdots x'_n$ be a decomposition in steps for x'. If aIx' then let $i = 0$, else let i be the integer such that x'_i is the right-most step containing a letter b such that $(a, b) \in D$.

Let x'' be the word obtained inserting a in x'_{i+1} with respect to the lexicographic ordering. The decomposition $x'_1x'_2 \cdots x'_ix''x'_{i+2} \cdots x'_n$ is a normal form for x.

We prove the uniqueness by contradiction. Assume that there exists a word x with two different normal forms. We choose x with minimal length, let $x'_1 x'_2 \cdots x'_n$ and $x''_1 x''_2 \cdots x''_p$ be two step decompositions of x. As x has minimal length and trace monoids are cancellative, Cor. 2, necessarily x'_1 and x''_1 are different: $x'_1 = yay'$ and $x''_1 = yby''$ with $a \neq b$. Thus, $x'_1 x'_2 \cdots x'_n = yay'x'_2 \cdots x'_n$ and $x''_1 x''_2 \cdots x''_p = yby''x''_2 \cdots x''_p$. If $b \notin \mathrm{alph}(y')$ we have $x'_2 \cdots x'_n = z'bz''$ with $|z'|_b = 0$. As $yay'z'bz'' \sim yby''x''_2 \cdots x''_p$ necessarily $bI(ay'z')$ by Prop. 1. This yields a contradiction to the third condition of the definition of a Foata normal form; thus $b \in \mathrm{alph}(y')$ and by symmetry $a \in \mathrm{alph}(y'')$. Using once more the definition of the normal form, we have $a < b$ and $b < a$. This yields the final contradiction and the result.

2.4 A Simple Algorithm to Compute Normal Forms

Let us describe a simple method which enables to compute normal forms. Let $\mathbb{M}(\Sigma, I)$ be a free partially commutative monoid, we use a stack for each letter of the alphabet Σ. Let x be a word of Σ^*, we scan x from right to left; when processing a letter a it is pushed on its stack and a marker is pushed on the stack of all the letters b ($b \neq a$) which do not commute with a.

When all of the word has been processed we can compute either the lexicographic normal form or the Foata normal form.

- To get the lexicographic normal form: it suffices to take among the letters being on the top of some stack that letter a being minimal with respect to the given lexicographic ordering. We pop a marker on each stack corresponding to a letter b ($b \neq a$) which does not commute with a. We repeat this loop until all stacks are empty.
- To get the Foata normal form we take within a loop the set formed by letters being on the top of stacks; arranging the letters in the lexicographic order yields a step. As previously this loop is repeated until all stacks are empty.

For example, with (Σ, I) as in Ex. 1 and the word *badacb* we get the stacks given below. The lexicographic normal form is *baadbc*, and the Foata normal form is $(b)(ad)(a)(bc)$.

$*$	b		
a	$*$	$*$	$*$
a	$*$	$*$	d
$*$	$*$	$*$	$*$
$*$	b	c	$*$
a	b	c	d

The advantage of the algorithm above is the easy implementation. (Of course, the use of so many markers is redundant, the algorithm does not claim to be the most efficient one.) The algorithm (given in Sect. 6) computing the Hasse diagram is a slight variant of the method above.

2.5 Möbius Functions and Normal Forms

This section presents the relation between liftings of Möbius functions and normal forms for trace monoids. It will not be used in the sequel. A reader not being familiar with the basic notions might skip this section. It is worth mentioning however that the study of the Möbius function was really at the beginning of trace theory in combinatorics. In particular, Thm. 10 below is from the original Lecture Notes [13] by Cartier and Foata.

Let $\mathbb{M} = \mathbb{M}(\Sigma, I)$ and $\varphi : \Sigma^* \longrightarrow \mathbb{M}$ the canonical homomorphism. A set of *normal forms* (or a *cross-section*) is a subset $S \subseteq \Sigma^*$ such that φ induces a bijection between S and $\mathbb{M}$. For example, *LexNF* is a set of normal forms. By χ_S we denote the characteristic function of S:

$$\chi_s(w) = 1, \text{ if } w \in S \text{ and } \chi_S(w) = 0 \text{ otherwise.}$$

We view χ_S as a function from Σ^* to $\mathbb{Z}$. Functions from Σ^* (or more generally from $\mathbb{M}$) to $\mathbb{Z}$ are called *formal power series*. The set of all power series is denoted by $\mathbb{Z}\langle\langle\Sigma^*\rangle\rangle$ (by $\mathbb{Z}\langle\langle\mathbb{M}\rangle\rangle$ resp.). A convenient notation of power series is $f = \sum_w f(w)w$. The set of power series is a ring by the usual addition and the non-commutative Cauchy-product.

$$(f+g)(w) = f(w) + g(w)$$

$$(f \cdot g)(w) = \sum_{uv=w} f(u)g(v)$$

The unit of this ring is $1 = \chi_{\{1\}}$, which fits perfectly to the notation of power series above. If $f(1) \in \{-1, +1\}$, then there exists a unique power series f^{-1} such that $f \cdot f^{-1} = f^{-1} \cdot f = 1$, it is called the *formal inverse* of f. Clearly, $(f^{-1})^{-1} = f$.

There is a canonical ring homomorphism from $\mathbb{Z}\langle\langle\Sigma^*\rangle\rangle$ onto $\mathbb{Z}\langle\langle\mathbb{M}\rangle\rangle$, denoted (by abuse of language) again by φ. We have $(\varphi(f))(t) = \sum_{\varphi(w)=t} f(w)$ for all $f \in \mathbb{Z}\langle\langle\Sigma^*\rangle\rangle$ and $t \in \mathbb{M}$. Observe that $S \subseteq \Sigma^*$ is a set of normal forms if and only if $\varphi(\chi_S) = \chi_{\mathbb{M}}$. The function $\chi_{\mathbb{M}} : \mathbb{M} \longrightarrow \mathbb{Z}$ is the constant function with value 1. A fundamental combinatorial property of finitely generated free partially commutative monoids is that the formal inverse $(\chi_{\mathbb{M}})^{-1} \in \mathbb{Z}\langle\langle\mathbb{M}\rangle\rangle$ is a polynomial, i.e., only finitely many values are non-zero.

Definition 9. Denote by $\mathcal{F} = \{F \subseteq \Sigma \mid (a,b) \in I \text{ for all } a,b \in F, a \neq b\}$ the set of independence cliques. For each $F \in \mathcal{F}$ let $[F] \in \mathbb{M}$ be the trace $[F] = \prod_{a \in F} a$. (The product is well-defined, since the elements of F commute.) The *clique polynomial* of $\mathbb{M}$ is defined as the formal power series

$$\mu_{\mathbb{M}} = \sum_{F \in \mathcal{F}} (-1)^{|F|}[F].$$

Theorem 10. *The clique polynomial $\mu_{\mathbb{M}}$ is the Möbius function of $\mathbb{M}(\Sigma, I)$, i.e., $\mu_{\mathbb{M}}$ is the formal inverse of the constant function with value 1:*

$$\mu_{\mathbb{M}} = (\chi_{\mathbb{M}})^{-1}.$$

Proof. We have to show $(\mu_{\mathbb{M}} \cdot \chi_{\mathbb{M}})(t) = 1$, if $t = 1$ and $(\mu_{\mathbb{M}} \cdot \chi_{\mathbb{M}})(t) = 0$ otherwise. For $t \in \mathbb{M}$ let $\min(t)$ be the set of minimal elements, i.e., $\min(t) = \{a \in \Sigma \mid t = as$ for some $s \in \mathbb{M}\}$.

Clearly, $\min(t) \in \mathcal{F}$ and we have

$$(\mu_{\mathbb{M}} \cdot \chi_{\mathbb{M}})(t) = \sum_{t=[F]\cdot s} (-1)^{|F|}.$$

Since $\mathbb{M}$ is left cancellative we can write

$$(\mu_{\mathbb{M}} \cdot \chi_{\mathbb{M}})(t) = \sum_{F \subseteq \min(t)} (-1)^{|F|}.$$

Indeed, if $t = [F] \cdot s$, then $F \subseteq \min(t)$ and s is uniquely defined by F. Conversely, if $F \subseteq \min(t)$, then $F \in \mathcal{F}$ and there exists some unique $s \in \mathbb{M}$ such that $t = [F] \cdot s$. Therefore the claim follows (with $n = |\min(t)|$) from the well-known identity

$$\sum_{F \subseteq \min(t)} (-1)^{|F|} = \sum_k \binom{n}{k} (-1)^k = \begin{cases} 1 & \text{if } n = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Remark. For infinitely generated free partially commutative monoids Thm. 10 holds as well. The only difference is that $\mu_{\mathbb{M}}$ may be a formal power series with infinitely many non-zero coefficients. The classical Möbius inversion formula becomes a corollary. Let $\mathbb{N}^{\otimes} = \{n \in \mathbb{N} \mid n > 0\}$ the multiplicative monoid of positive integers. It is a commutative monoid, freely generated by the primes. The Möbius function from elementary number theory is

$$\mu : \mathbb{N}^{\otimes} \longrightarrow \mathbb{Z}, \quad \mu(n) = \begin{cases} (-1)^k & \text{if } n \text{ is a product of } k \text{ distinct primes, } k \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Thus, μ is the Möbius function of the monoid $\mathbb{N}^{\otimes}$. Let $f, g : \mathbb{N}^{\otimes} \to \mathbb{Z}$ be two functions. Möbius inversion states the equivalence:

$$\forall n : g(n) = \sum_{d|n} f(d) \iff \forall n : f(n) = \sum_{d|n} \mu(d) g(\frac{n}{d}).$$

This is precisely the assertion of Thm. 10 in the special case of the commutative monoid $\mathbb{N}^{\otimes}$.

Definition 11. An *unambiguous lifting* of the Möbius function $\mu_{\mathbb{M}}$ is a polynomial $\mu \in \mathbb{Z}\langle\langle \Sigma^* \rangle\rangle$ such that first $\varphi(\mu) = \mu_{\mathbb{M}}$ and second for each $F \in \mathcal{F}$ there exists exactly one word $w_F \in \Sigma^*$ satisfying both $\varphi(w_F) = [F]$ and $\mu(w_F) \neq 0$.

Remark. If $\mu \in \mathbb{Z}\langle\langle \Sigma^* \rangle\rangle$ is an unambiguous lifting of $\mu_{\mathbb{M}}$ then we can write $\mu = \sum (-1)^{|w_F|} w_F$.

The following corollary is a direct consequence of Thm. 10 and of the definition of an unambiguous lifting.

Corollary 12. *If $\mu \in \mathbb{Z}\langle\langle \Sigma^* \rangle\rangle$ is an unambiguous lifting of $\mu_{\mathbb{M}}$, then the formal inverse μ^{-1} is the characteristic function over a set of normal forms.*

Another way to obtain a set of normal forms is by complete rewriting systems. We shall establish a close relation between unambiguous liftings, complete rewriting systems, and transitive orientations of the independence alphabet. Before we can do so, we have however to recall some basic notions and facts.

A *semi-Thue system* is a set of rules $R \subseteq \Sigma^* \times \Sigma^*$. It defines a one-step reduction $\underset{R}{\Longrightarrow}$ by $ulv \underset{R}{\Longrightarrow} urv$ for $u, v \in \Sigma^*, (l, r) \in R$. By $\overset{*}{\underset{R}{\Longrightarrow}}$ (and $\overset{*}{\underset{R}{\Longleftrightarrow}}$, resp.) we denote the reflexive and transitive closure (reflexive, symmetric and transitive closure resp.) of $\underset{R}{\Longrightarrow}$. By Σ^*/R we mean the quotient monoid $\Sigma^*/\overset{*}{\underset{R}{\Longleftrightarrow}}$.

Example 2. For an independence relation $I \subseteq \Sigma \times \Sigma$ and a partial order $<$ of Σ let

$$\begin{aligned} S(I) &= \{(ab, ba) \mid (a, b) \in I\} \\ S(I, <) &= \{(ba, ab) \mid (a, b) \in I \text{ and } a < b\} \end{aligned}$$

Then we have $\Sigma^*/S(I) = \mathbb{M}(\Sigma, I)$ and if for all $(a, b) \in I$ either $a < b$ or $b < a$, then $\Sigma^*/S(I, <) = \mathbb{M}(\Sigma, I)$.

- A semi-Thue system R is called *Noetherian*, if there is no infinite chain $w_1 \underset{R}{\Longrightarrow} w_2 \underset{R}{\Longrightarrow} \cdots$,
- it is called *confluent*, if for all $u \overset{*}{\underset{R}{\Longleftarrow}} w \overset{*}{\underset{R}{\Longrightarrow}} v$, there exists some $z \in \Sigma^*$ such that $u \overset{*}{\underset{R}{\Longrightarrow}} z \overset{*}{\underset{R}{\Longleftarrow}} v$.
- Confluence of Noetherian systems is equivalent with *local confluence*, i.e., for all $u \underset{R}{\Longleftarrow} w \underset{R}{\Longrightarrow} v$ there exists some $z \in \Sigma^*$ such that $u \overset{*}{\underset{R}{\Longrightarrow}} z \overset{*}{\underset{R}{\Longleftarrow}} v$.
- A system being both Noetherian and (locally) confluent is called *complete.*
- For a complete system R, the set of irreducible words

  $$\mathrm{Irr}(R) = \{w \in \Sigma^* \mid \nexists v : (w \underset{R}{\Longrightarrow} v)\}$$

  is in canonical bijection with the quotient monoid Σ^*/R. Hence, $\mathrm{Irr}(R)$ is a set of normal forms for the monoid Σ^*/R.

Definition 13. Let (Σ, I) be an independence alphabet. A *transitive orientation* of I is a partial order $<$ of Σ such that first $I^+ = \{(a, b) \in I \mid a < b\}$ is a transitive relation and second, it holds either $a < b$ or $b < a$ for all $(a, b) \in I$.

Lemma 14. *i) Let $<$ be a transitive orientation of I and the lexicographical order of Σ^* be defined by some linear extension of $<$. Then $S(I, <)$ is complete and the set of irreducible words is the set of lexicographic normal forms, $Irr(S(I, <)) = LexNF$.*

ii) *Let $R \subseteq \Sigma^* \times \Sigma^*$ be any finite complete semi-Thue system such that $\Sigma^*/R = \mathbb{M}(\Sigma, I)$. Then there exists a transitive orientation $<$ such that $S(I,<) \subseteq R$. In particular, by i) we have $Irr(R) = LexNF$.*

Proof. i): By the characterization of lexicographic normal forms, c.f. Thm. 7, the following (infinite (!)) system is easily seen to be complete:

$$R = \{(bua, abu) \mid a < b \text{ and } aI(bu)\}.$$

Of course $S(I,<) \subseteq R$ and $\mathrm{Irr}(R) = LexNF$. Since $<$ is a transitive orientation, one can show that $(bua, abu) \in R$ implies $bua \underset{S(I,<)}{\overset{*}{\Longrightarrow}} z$ for some z with $abu \underset{R}{\overset{*}{\Longrightarrow}} z$. Since R is Noetherian, we have $\underset{R}{\overset{*}{\Longrightarrow}} = \underset{S(I,<)}{\overset{*}{\Longrightarrow}}$ and *i)* follows.

ii): Consider the relation $<$ satisfying $a < b \iff (ba, ab) \in R$. The relation $<$ is asymmetric and irreflexive and it holds $a < b$ or $b < a$ for all $(a,b) \in I$. Moreover, an easy reflection shows $a^* b^* \subseteq \mathrm{Irr}(R)$ for all $a < b$. Consider $a < b < c$ and $c^n b^n a^n$ for $n \geq 0$. Then

$$b^n c^n a^n \underset{R}{\overset{*}{\Longleftarrow}} c^n b^n a^n \underset{R}{\overset{*}{\Longrightarrow}} c^n a^n b^n$$

Since R is finite and complete, we have $c^n a^n \notin \mathrm{Irr}(R)$ for n large enough. Hence, $(a,c) \in \Sigma$ but we do not have $c < a$. Therefore $a < c$, and $<$ is transitive. Since $<$ is asymmetric, the relation is a partial order, and it is a transitive orientation of I. The assertion follows.

There is a surprising relation between unambiguous liftings of Möbius functions, finite complete semi-Thue systems, and transitive orientations of I. We can state the following theorem.

Theorem 15. *Let (Σ, I) be an independence alphabet $\mathbb{M}(\Sigma, I)$. Then the following assertion are equivalent.*

i) *There exists an unambiguous lifting of the Möbius function $\mu_{\mathbb{M}}$.*
ii) *There exists a finite complete semi-Thue system R such that $\Sigma^*/R = \mathbb{M}$.*
iii) *There exists a transitive orientation of I.*

In fact, Thm. 15 is a corollary of Prop. 14 and the following more precise theorem.

Theorem 16. *i)* *Let $\mu \in \mathbb{Z}\langle\langle \Sigma^* \rangle\rangle$ be an unambiguous lifting. For $(a,b) \in I$ define $a < b$ by $\mu(ba) = 1$. Then $<$ is a transitive orientation, $S(I,<)$ is complete, and we have $\mu^{-1} = \chi_{\mathrm{Irr}(S(I,<))}$.*
ii) *Let $<$ be a transitive orientation of I. Then there exists a unique unambiguous lifting μ such that $\mu(ba) = 1$ for all $a < b, (a,b) \in I$. This function μ is characterized by $\mu = (\chi_{\mathrm{Irr}(S(I,<))})^{-1}$.*

2.6 Bibliographical Remarks

The theory of free partially commutative monoids has its origins in combinatorics, [13]. The existence of the Foata normal form and the characterization of the Möbius function as the polynomial over the independence cliques is shown there. In computer science they became popular mainly by the work of Mazurkiewicz, see [68, 69]. Early ideas can also be found in the paper by Keller (1973), where the Projection Lemma is (implicitly) stated in [64, Lem. 2.5]. The Projection Lemma in the more general context of semi-commutations is due to Clerbout [15]. Levi's Lemma and the Projection Lemma, as stated here, can be found in a paper of Cori and Perrin, [21]. Prop. 6 has been shown by Duboc [40]. The characterization of lexicographic normal forms, Thm. 7, is due to Anisimov and Knuth [4]. The simple algorithm to compute normal forms has been proposed by Perrin, see [81].

The characterization, when lexicographic normal forms are the irreducible words of some finite complete semi-Thue system, is from Métivier and Ochmański [74] and Otto [79]. The bridge to unambiguous lifting of the Möbius functions is from Diekert [23]. A generalization of Thms. 15 and 16 has been conjectured in [24] and then been proven in [26].

3 Combinatorial Properties

3.1 Equations

Let M be any monoid. Two elements $x, y \in M$ are called *conjugated*, if $xz = zy$ for some $z \in M$. They are called *transposed*, if $x = uv$ and $y = vu$ for some $u, v \in M$. Clearly, transposed elements are conjugated, and conjugacy is transitive. In free commutative monoids conjugacy is the identity relation. In free monoids conjugated elements are transposed. Moreover, if $x = uv$ and $y = vu$, then $u, v \in r^*$ for some word r. These results generalize immediately to direct products of free monoids. (The word r has however to be replaced by a tuple of pairwise independent words $(r_1, \ldots, r_k)$ such that $u, v \in r_1^* \cdots r_k^*$.)

For trace monoids where the dependence relation D is not transitive, the situation is different. Let $(\Sigma, D) = a - b - c$. Then the traces abc and cba are not transposed, but they are conjugated: $(abc)(aba) = (aba)(cba)$, since $ac = ca$. Note that abc and $bca = bac$ are transposed, as well as bac and cba. Thus, the conjugation of abc and cba has been realized by two transpositions. This is a general fact: In trace monoids conjugacy is always equal to the equivalence relation generated by transposition.

Proposition 17. *Let $x, y, z \in \mathbb{M}(\Sigma, I)$ be traces. Then we have $xz = zy$ if and only if there are traces $z_1, z_2, u_1, \ldots, u_k$, $k \geq 0$, satisfying the following conditions*

i) $x = u_1 \cdots u_k$, $y = u_k \cdots u_1$,
ii) $u_i I u_j$ *for* $1 \leq i < j - 1 \leq k - 1$,
iii) $z_1 = (u_1 \cdots u_{k-1}) \cdots (u_1 u_2) u_1$,

iv) $z = z_1 z_2$ *with* xIz_2.

Proof. Using the Projection Lemma, Prop. 1, an easy reflection shows that the conditions imply $xz = zy$. For the converse let $xz = zy$. By Levi's Lemma, Prop. 4, we can write $x = x'u$, $z = z'y' = x'z'$, and $y = uy'$ with uIz'. If $x' = 1$ then we are done with $x = y = u$ and $z = z'$. If $x' \neq 1$, then $|z'| < |z|$ and we can use induction. Hence we may assume $x' = u_1 \cdots u_k$, $y' = u_k \cdots u_1$, $u_i I u_j$ for $1 \leq i < j-1 \leq k-1$, $z_1' = (u_1 \cdots u_{k-1}) \cdots (u_1 u_2)(u_1)$ and $z' = z_1' z_2$ with $x' I z_2$. Putting $u_{k+1} = u$ and $z_1 = x' z_1'$ we obtain the result, since uIz_2.

The diameter of a dependence alphabet(Σ, D) is the maximum over the lengths of the shortest paths connecting letters from Σ; e.g., if $\mathbb{M}(\Sigma, I)$ is commutative, then the diameter is zero, if it is free and $|\Sigma| \geq 2$, then it is one. For $(\Sigma, D) = a - b - c$ the diameter is two. Observe that some of the u_i in Prop. 17 may be empty, this allows to regroup them. One can derive the following statement.

Corollary 18. *Let* (Σ, D) *be a dependence alphabet of diameter* d*. Then traces* $x, y \in \mathbb{M}(\Sigma, I)$ *are conjugated if and only if they can transformed into each other by at most* d *transpositions.*

A basic problem is to determine the solutions of the equation $xy = yx$ in trace monoids. As one might hope, the equation $xy = yx$ holds if and only if there are pairwise independent and connected traces $t_1, \ldots, t_k$ such that $x, y \in t_1^* \cdots t_k^*$. This will be seen from a more general approach (making proofs thereby simpler).

Consider $X = \{x, y\}$ with $x \neq y$ as an alphabet of two *unknowns*. A (non-trivial) equation in two unknowns $e = f$ is a pair of (distinct) words $e, f \in X^*$. A *solution* of $e = f$ is a homomorphism $\Theta : X^* \to \mathbb{M}(\Sigma, I)$ such that $\Theta(e) = \Theta(f)$. If a solution Θ is defined by $\Theta(x) = u$ and $\Theta(y) = v$, we also say that (u, v) is a solution for $e = f$, or more conveniently we write that $x = u$ and $y = v$. A solution (u, v) is called *cyclic*, if $u, v \in t^*$ for some trace t.

Lemma 19. *Let* $x = u$, $y = v$ *with* $u, v \in \mathbb{M}(\Sigma, I)$ *a solution of a non-trivial equation* $e = f$ *in unknowns* $\{x, y\}$*. The solution is cyclic, if one of the following conditions is satisfied:*

i) $\mathbb{M}(\Sigma, I)$ *is free, i.e.,* $I = \emptyset$.
ii) $\alpha\pi(u) = \beta\pi(v)$ *for some rational numbers* $\alpha, \beta > 0$*, where* $\pi(u)$*,* $\pi(v)$ *denote the Parikh-images of* u *and* v *in* $\mathbb{N}^{\Sigma}$.

Proof. Assume by contradiction that the assertion would be wrong. Let (u, v) be a pair where $|u| + |v|$ is minimal under the condition that (u, v) is a non-cyclic solution for some non-trivial equation $e = f$ satisfying *i)* or *ii)*. Clearly, by cancellation and symmetry we may assume that e begins with x, f begins with y, and $|u| \leq |v|$. By Levi's Lemma we can write $u = pw'$ and $v = pw$ with wIw'. If $I = \emptyset$, then $w' = 1$. However, the same is true for *ii)*, because $\alpha\pi(u) = \beta\pi(v)$ with $\alpha, \beta > 0$ and $|u| \leq |v|$ imply $|u|_a \leq |v|_a$ for all $a \in \Sigma$.

Hence we have $v = uw$ in both cases. Moreover, we may assume $u \neq 1$ and $w \neq 1$. Now replace the unknown y by xz. This yields a new non-trivial equation $e' = f'$ in the set of unknowns $\{x, z\}$. Clearly (u, w) is a solution of $e' = f'$ and we have $|u| + |w| < |u| + |v|$. If $I = \emptyset$ then (u, w) is a cyclic solution by minimality of $|u| + |v|$. If $\alpha\pi(u) = \beta\pi(v)$, then $w \neq 1$ implies $\alpha > \beta$. Hence $\alpha\pi(u) = \beta(\pi(u) + \pi(w))$ implies $(\alpha - \beta)\pi(u) = \beta\pi(w)$ with $(\alpha - \beta), \beta > 0$ Again, by minimality of $|u| + |v|$, we find that (u, w) is a cyclic solution. Thus, in both cases $u, w \in t^*$ for some t, hence $u, v \in t^*$ being a contradiction.

Remark. An equation $e = f$ is called *unbalanced,* if the number of occurrences of x (or of y) in the left-hand side of the equation differs from that of the right-hand side. Of course, if (u, v) is a solution of an unbalanced equation $e = f$, then $\alpha\pi(u) = \beta\pi(v)$ with $\alpha, \beta > 0$. In fact we may take $\alpha = |(|e|_x - |f|_x)|$ and $\beta = |(|f|_y - |e|_y)|$.

The lemma above yields therefore a special case:

Proposition 20. *Every solution in trace monoids of an unbalanced equation in two unknowns is cyclic.*

Lemma 21. *Let $x = u$, $y = v$ be a solution of a non-trivial equation in two unknowns in a trace monoid. Write $u = u_1 \cdots u_m$, $v = v_1 \cdots v_n$ such that u_i, v_j are connected and $u_i I u_j, v_i I v_j$ for all $i \neq j$. Then for all (i, j) with $1 \leq i \leq m$ and $1 \leq j \leq n$ we have either alph(u_i) = alph(v_j) or $u_i I v_j$.*

Proof. Let $a \in \mathrm{alph}(u)$, $b \in \mathrm{alph}(v)$ with $(a, b) \in D$. Projection to $\{a, b\}^*$ and applying $i)$ of Lem. 19 yields $\{a, b\} \subseteq \mathrm{alph}(u) \cap \mathrm{alph}(v)$. The lemma follows.

Proposition 22. *Let $x = u$, $y = v$ be a solution of a non-trivial equation in two unknowns in a trace monoid. Then there are pairwise independent traces $t_1, \ldots, t_k$ such that $u, v \in t_1^* \cdots t_k^*$.*

Proof. By Lem. 21 we are reduced to the case where $\mathrm{alph}(u) = \mathrm{alph}(v)$ and u is connected. Let $a, b \in \mathrm{alph}(u)$, $(a, b) \in D$. Choose $\alpha, \beta > 0$ such that $\alpha|u|_a = \beta|v|_a$. By Lem. 19 it is enough to show that $\alpha|u|_b = \beta|v|_b$. By projection to $\{a, b\}^*$ and $i)$ of the same lemma we find integers $t_a, t_b > 0$ such that $|u|_a = mt_a$, $|v|_a = nt_a$, $|u|_b = mt_b$, and $|v|_b = nt_b$. The above implies $\alpha m = \beta n$ and this in turn $\alpha|u|_b = \beta|v|_b$.

A connected trace $r \in \mathbb{M}(\Sigma, I)$ is called a *primitive root*, if $r = s^n$ implies $n = 1$. From Prop. 22 and Lem. 21 we may derive the following corollaries:

Corollary 23. *Every non-empty connected trace is the power of a unique primitive root. Moreover, if $uv = vu \in \mathbb{M}(\Sigma, I)$ and uv is connected, then $u, v \in r^*$ for the primitive root r of u.*

Corollary 24. *Let $u_1, \ldots, u_k \in \mathbb{M}(\Sigma, I)$ be pairwise commuting traces, i.e., $u_i u_j = u_j u_i$ for all i, j. Then there are pairwise independent and connected traces $t_1, \ldots, t_m$ such that $u_i \in t_1^* \cdots t_m^*$ for all $1 \leq i \leq k$.*

The following consequence will be used in the next section.

Corollary 25. *Let $h : \mathbb{N}^k \to \mathbb{M}(\Sigma, I)$ be an injective homomorphism. Then Σ contains at least k pairwise independent letters.*

Proof. Let u_i be the image under h of the i-th unit-vector, $1 \leq i \leq k$. According to Cor. 24 write $u_i = t_1^{n_{i,1}} \cdots t_m^{n_{i,m}}$. The matrix $(n_{i,j})_{1 \leq i \leq k, 1 \leq j \leq m}$ has rank at least k. Hence $m \geq k$. It is enough to pick one letter from each $\mathrm{alph}(t_i)$, $1 \leq i \leq m$.

3.2 Strong Homomorphisms and Codings

A homomorphism between trace monoids $h : \mathbb{M}(\Sigma, I) \to \mathbb{M}(\Sigma', I')$ is given by a mapping $h : \Sigma \to \mathbb{M}(\Sigma', I')$ such that $h(a)h(b) = h(b)h(a)$ for all $(a, b) \in I$. We say that h is a *strong homomorphism,* if moreover we have $h(a)I'h(b)$ for all $(a, b) \in I$. This means that independent letters are mapped to independent traces. In this section we show that the existence of injective strong homomorphisms is directly related to morphisms of graphs. By minor modifications the following correspondences could be made functorial, but we do not intend to introduce categories here.

Definition 26. A *morphism* H of dependence alphabets from (Σ', D') to (Σ, D) is a morphism of the underlying undirected graphs (with self-loops), i.e., H is a mapping $H : \Sigma' \to \Sigma$ on letters such that $(H(a'), H(b')) \in D$ for all $(a', b') \in D'$.

Example 3. Write $(\Sigma, D) = (\bigcup_{i=1}^k A_i, \bigcup_{i=1}^k A_i \times A_i)$ and let (Σ', D') be the disjoint union of the complete graphs $(A_i, A_i \times A_i)$. We obtain a morphism $H : (\Sigma', D') \longrightarrow (\Sigma, D)$ which is induced by set inclusions $A_i \subseteq \Sigma, 1 \leq i \leq k$.

The following proposition is another variant of the Projection Lemma Prop. 1 and of Cor. 3. The proof is an easy exercise.

Proposition 27. *Let $H : (\Sigma', D') \longrightarrow (\Sigma, D)$ be a morphism of dependence graphs and for each $a \in \Sigma$ let $h(a) \in \mathbb{M}(\Sigma', I')$ be a trace such that $alph(h(a)) = H^{-1}(a)$. Then h induces a strong homomorphism $h : \mathbb{M}(\Sigma, I) \longrightarrow \mathbb{M}(\Sigma', I')$. This homomorphism is injective if and only if H is surjective on vertices and edges.*

An injective homomorphism between trace monoids is denoted henceforth as a *coding.* According to [12] we call an injective strong homomorphism a *strong coding.* A strong coding allows to encode a trace in such a way that independency is preserved. The question arises whether there exists a strong coding for given dependence alphabets $(\Sigma, D), (\Sigma', D')$. In general, the answer is at least *NP*-hard, due to the following fact rephrasing Cor. 25.

Proposition 28. *Let (Σ, D) be a dependence alphabet and $k \geq 0$. Then the following assertions are equivalent:*

i) There exists a coding from $\mathbb{N}^k$ into $\mathbb{M}(\Sigma, I)$.
ii) There exists a strong coding from $\mathbb{N}^k$ into $\mathbb{M}(\Sigma, I)$.
iii) The dependence alphabet contains an independent set of size k.

Definition 29. A *relational morphism* $H : (\Sigma', D') \longrightarrow (\Sigma, D)$ of dependence alphabets is a relation $H \subseteq \Sigma' \times \Sigma$ such that $(a', b') \in D'$ implies $H(a') \times H(b') \subseteq D$. It is called surjective on vertices and edges, if both for all $a \in \Sigma$ there exists $a' \in \Sigma'$ with $(a', a) \in H$ and for all $(a, b) \in D, a \neq b$ there exists $(a', b') \in D'$, $a' \neq b'$ such that $(a, b) \in H(a') \times H(b')$.

The following result shows that there exists a strong coding

$$h : \mathbb{M}(\Sigma, I) \longrightarrow \mathbb{M}(\Sigma', I')$$

if and only if there exists some relational morphism

$$H : (\Sigma', D') \longrightarrow (\Sigma, D)$$

being surjective on vertices and edges.

Theorem 30. *i) Let $h : \mathbb{M}(\Sigma, I) \longrightarrow \mathbb{M}(\Sigma', I')$ be a strong coding and $H = \{(a', a) \in \Sigma' \times \Sigma \mid a' \in alph(h(a))\}$. Then H is a relational morphism being surjective on vertices and edges.*
ii) Let $H : (\Sigma', D') \longrightarrow (\Sigma, D)$ be a relational morphism being surjective on vertices and edges. Then we can construct a strong coding $h : \mathbb{M}(\Sigma, I) \longrightarrow \mathbb{M}(\Sigma', I')$ such that $H = \{(a', a) \in \Sigma' \times \Sigma \mid a' \in alph(h(a))\}$.

Proof. Part *i)* is easy and omitted. We sketch *ii)*: First we order the alphabets, i.e., we assume $\Sigma = \{a_1, a_2 \ldots\}$ with $a_1 < a_2 < \cdots$ and $\Sigma' = \{a'_1, a'_2, \ldots\}$ with $a'_1 < a'_2 < \cdots$. For each $i = 1, 2, \ldots$, we define a set H_i by $H_i = \{a'_j \in \Sigma' \mid (a'_j, a_i) \in H\}$. If $(a_i, a_j) \in I$, then $H_i \times H_j \subseteq I'$, since H is a relational morphism. We have $H_i \neq \emptyset$ for all i, since H is surjective on vertices; and if $(a_i, a_j) \in D, i \neq j$, then there are $b' \in H_i, c' \in H_j$ with $(b', c') \in D', b' \neq c'$, since H is surjective on edges.

Assume that $H_i = \{a'_{i_1}, \ldots, a'_{i_k}\}$ with $a'_{i_1} < \cdots < a'_{i_k}$. Define the traces $\overrightarrow{H}_i = a'_{i_1} \cdots a'_{i_k}$ and $\overleftarrow{H_i} = a'_{i_k} \cdots a'_{i_1}$. Thus, $\overrightarrow{H}_i$ is the product of $a'_{i_l} \in H_i$ in increasing order and $\overleftarrow{H_i}$ is the product in decreasing order. Now define

$$h : \mathbb{M}(\Sigma, I) \longrightarrow \mathbb{M}(\Sigma', I'), \quad h(a_i) = (\overrightarrow{H}_i)^i \overleftarrow{H}_i.$$

The equality $H = \{(a', a) \in \Sigma' \times \Sigma \mid a' \in \mathrm{alph}(h(a))\}$ is obvious. Let us show by contradiction that h is injective. Assume $h(a_i x) = h(y)$, with $a_i \in \Sigma$ and $x, y \in \Sigma^*$ such that $a_i x \neq y$. Then y must contain a letter depending on a_i, hence we can write $y = u a_j z$ with $a_i I u$ and $(a_i, a_j) \in D$. By cancellation we may assume $i \neq j$. Therefore we find $b' \in H_i, c' \in H_j$ with $(b', c') \in D', b' \neq c'$. The result now follows by projection onto $\{b', c'\}^*$ and some few calculations left to the reader.

Remark. The proof of Thm. 30 is valid for countable alphabets as well, as long as we demand that $\{a' \in \Sigma' \mid (a', a) \in H\}$ is finite for all $a \in \Sigma$. (This is why we wrote $\Sigma = \{a_1, a_2, \ldots\}$ instead of $\Sigma = \{a_1, \ldots, a_n\}$.)

Corollary 31. *It is NP-complete to decide whether there exists a strong coding between trace monoids.*

Proof. It is clearly in *NP* to decide whether there exists a relational morphism being surjective on vertices and edges. The hardness follows by Prop. 28, *iii)*.

Corollary 32. *There is a strong coding of* $\mathbb{M}(\Sigma, I)$ *into a* k*-fold direct product of free monoids, if and only if* (Σ, D) *has a covering by* k *cliques, i.e.,*

$$(\Sigma, D) = \left(\bigcup_{i=1}^{k} A_i, \bigcup_{i=1}^{k} (A_i \times A_i) \right)$$

Little is known about the existence of codings. According to the following example we can construct codings in some cases, where strong codings do not exist.

Example 4. Let $(\Sigma, D) = \begin{array}{ccc} a & - & b \\ | & & | \\ d & - & c \end{array}$ and $(\Sigma', D') = p - q \cup r - s$. Algebraically $\mathbb{M}_1 = \mathbb{M}(\Sigma, D)$ is a free product of commutative monoids, $\mathbb{M}_1 = \mathbb{N}^2 * \mathbb{N}^2$, and $\mathbb{M}_2 = \mathbb{M}(\Sigma', D')$ is a direct product of free monoids, $\mathbb{M}_2 = \{p, q\}^* \times \{r, s\}^*$. By Cor. 32 there is no strong coding of $\mathbb{M}_1$ into $\mathbb{M}_2$. But there is a coding. Take any non-singular 2×2-matrix with non-zero entries, say the matrix with columns $\binom{1}{1}$ and $\binom{2}{1}$. Then define accordingly by using the columns of the matrix as exponents for the letters:

$$h(a) = \begin{pmatrix} p \\ r \end{pmatrix}, \quad h(c) = \begin{pmatrix} p^2 \\ r \end{pmatrix}, \quad h(b) = \begin{pmatrix} q \\ s \end{pmatrix}, \quad \text{and } h(d) = \begin{pmatrix} q^2 \\ s \end{pmatrix}.$$

An easy exercise shows that the homomorphism $h : \mathbb{M}_1 \longrightarrow M_2$ is a coding.

3.3 Trace Codes

Codes over words are widely studied, [5] for a comprehensive treatise. They play a fundamental role in computer science. There are well-known algorithms to test whether a finite (or regular) set $X \subseteq \Sigma^*$ is a code, i.e., whether it generates a free submonoid with basis X. In trace monoids the same question turns out to be undecidable.

Proposition 33. *It is undecidable whether a finite subset* X *of the direct product* $\{a, b\}^* \times \{c, d\}^*$ *generates a free submonoid (with basis* X*).*

Proof. An instance of the Post correspondence problem (PCP) consists of two lists $(u_1, \ldots, u_n), (v_1 \ldots, v_n)$ with $n \geq 2, u_i, v_i \in \Sigma^+$ for $1 \leq i \leq n$. A *special solution* to this instance of the PCP is a finite sequence $(1, i_1, \ldots, i_k, n)$ such that $u_1 u_{i_1} \cdots u_{i_k} u_n = v_1 v_{i_1} \cdots v_{i_k} v_n$. It is well-known that the existence of a special solution is undecidable. In order to reduce PCP to the problem above one employs markers. Let $\#$ be a new symbol; replace each letter $a \in \Sigma$ in the words $u_i, 1 \leq i \leq n$, by $a\#$, and in the words $v_i, 1 \leq i \leq n$, replace each letter $a \in \Sigma$ by $\#a$. Then, add the symbol $\#$ in front of the new word u_1, and add $\#$ at the end of the new v_n. We have now two lists of words $(\tilde{u_1}, \ldots, \tilde{u_n}), (\tilde{v_1}, \ldots, \tilde{v_n})$ over the new alphabet$\Sigma \cup \{\#\}$. Using any coding of $(\Sigma \cup \{\#\})^*$ into $\{c, d\}^*$, we may view $\tilde{u_i}, \tilde{v_i} \in \{c, d\}^+$ for $1 \leq i \leq n$. Finally define $X = \{(ab^i, \tilde{u_i}) \mid 1 \leq i \leq n\} \cup \{(ab^i, \tilde{v_i}) \mid 1 \leq i \leq n\}$. It is not difficult to see that X generates a free submonoid (with basis X) of $\{a, b\}^* \times \{c, d\}^*$ if and only if the given instance of PCP has no special solution.

Let $\mathbb{M} = \mathbb{M}(\Sigma, I)$ be a trace monoid and $X \subseteq \mathbb{M}$ be any subset. Define an independence relation $I_X \subseteq X \times X$ by $x I_X y$ if and only if $xy = yx$ in $\mathbb{M}$. This yields a free partially commutative monoid $\mathbb{M}_X = \mathbb{M}(X, I_X)$ and the inclusion $X \subseteq \mathbb{M}$ induces a homomorphism $h_X : \mathbb{M}_X \longrightarrow \mathbb{M}$. The set X is called a *trace code,* if h_X is a coding.

Remark. i) If we can decide whether $X \subseteq \mathbb{M}(\Sigma, I)$ is a trace code (X being finite), then we can decide whether X generates a free monoid, i.e., whether X is a code. Indeed, X is a code if and only if X is a trace code and I_X is empty. Whether or not the converse is true remains unclear.

ii) The question whether a finite subset X of $\mathbb{M}$ is a trace code can easily be reduced to the problem whether the intersection of two rational sets is empty. Define any total order $<$ on X. For $x \in X$ let $\mathit{LexNF}(x, X) \subseteq X^*$ be the rational set of lexicographic normal forms from $\mathbb{M}(X, I_X)$ having as first letter x. Let $\varphi_X : X^* \longrightarrow \mathbb{M}(X, I_X)$ the canonical homomorphism. Clearly, φ_X restricted to some $\mathit{LexNF}(x, X)$ is injective and $h_X(\varphi_X(\mathit{LexNF}(x, X)))$ is a rational set of $\mathbb{M}(\Sigma, I)$. Now observe that X is a trace code if and only if $h_X(\varphi_X(\mathit{LexNF}(x, X))) \cap h_X(\varphi_X(\mathit{LexNF}(y, X))) = \emptyset$ for all $x, y \in X$, $x \neq y$. This finishes the reduction to the intersection problem.

This remark and the results of the next section imply the following fact.

Proposition 34. *Let $X \subseteq \mathbb{M}(\Sigma, I)$ be a finite set of connected traces. Then it is decidable whether X is a trace code.*

Proof. In the terminology of the next section, $\mathit{LexNF}(x, X)$ is star-connected by Cor. 46 below. Hence $\varphi_X(\mathit{LexNF}(x, X)) \subseteq \mathbb{M}(X, I_X)$ is star-connected, and since h_X maps connected traces of $\mathbb{M}(X, I_X)$ to connected traces of $\mathbb{M}(\Sigma, I)$ (by definition of I_X), we see that $h_X(\varphi_X(\mathit{LexNF}(x, X)))$ is recognizable, c.f. Thm. 38. There is an effective procedure testing the emptiness of the intersection of two recognizable sets.

Emptiness of the intersection of two rational sets is known to be decidable, if the independence alphabet is a transitive forest, [1]. Therefore we can state by the remark above:

Proposition 35. *Let the independence alphabet be a transitive forest. Then it is decidable whether a finite set $X \subseteq \mathbb{M}(\Sigma, I)$ is a trace code.*

A transitive forest is the transitive closure of the descendant relation in a union of trees. It can be defined by forbidden structures. An independence alphabet is a transitive forest if and only if it does not contain neither C_4 nor P_4 as an induced subgraph. By C_4 we mean the chordless cycle and by P_4 the chordless path with 4 letters.

The proof of Prop. 33 shows that the trace code property is undecidable in the monoid $\{a, b\}^* \times \{c, d\}^*$, which means that the independence alphabet is the graph C_4. An undecidability result for P_4 would have given a characterization theorem, but the decidability, even decidability in polynomial time, is known in this particular case.

Proposition 36. *If (Σ, I) or (Σ, D) is a P_4, then the trace code property can be decided in polynomial time.*

If the trace code property can be decided for (Σ, I) or for (Σ, D), then it can be decided for all induced subgraphs and for all dependence alphabets, obtained by edge-contraction. Unfortunately, this is not enough to conclude that the undecidability of the trace code property can be characterized by some finite list of forbidden induced subgraphs. The classification of all independence alphabets where the (trace) code property is decidable is a challenge.

3.4 Bibliographical Remarks

Equations over traces have been studied first by Cori and Métivier [19] and Duboc. The results of Sect. 3.1 are due to Duboc, see [40]. The results presented here are also exposed by Choffrut and by Duchamp and Krob in [34, Chapt. 3,4]. A celebrated result of Makanin states that it is decidable whether a system of equations with constants has a solution over words, [67]. The extension of the result to traces would be of great interest, but it has not been attacked yet.

The undecidability of the trace code property is due to Hotz and Claus [62] and Chrobak and Rytter [14]. This result and the proof techniques are close to a classical result of Greibach [57] which shows the ambiguity of a context-free grammar is undecidable. The decidability for the intersection problem in case of transitive forests is from Aalbersberg and Hoogeboom [1]. It led directly to Prop. 35. After that result the characterization of dependence graphs where the trace code property is decidable concentrated on the graph P_4. Recently Hoogeboom and Muscholl [61] proved the positive result Prop. 36 for the code property (being therefore negative for completing the picture). It has been independently observed by Matiyasevich, who also proposed the notion of trace code. Matiyasevich announced another decidability result for the code-property provided the independence relation is acyclic (personal communication). The topic of free trace submonoids is also mentioned in the chapter by Choffrut of [34].

The question whether the existence of codings (injective homomorphisms) between trace monoids can be decided has been raised by Ochmański in [78], where among others Prop. 28 is stated. The notion of a strong coding has been defined by Bruyère et al. in [12]. Thm. 30 is the main result of [33]. It answers a question of the former paper and solves the problem above for strong codings. However, the decidability for codings is still open, some partial results and conjectures are in [11, 33].

4 Recognizable Trace Languages

4.1 Basic Facts about Recognizable and Rational Sets

Let M be a monoid with the unit element 1, a subset T of M is said to be *recognizable* if there exists a homomorphism η from M to a finite monoid S such that $T = \eta^{-1}(F)$ for some subset $F \subseteq S$. We also say that the homomorphism η recognizes T.

For a monoid M, an $M-$automaton $\mathcal{A} = (M, Q, \delta, q_0, F)$ consists of a finite set Q of states, an initial state $q_0 \in Q$, a subset $F \subseteq Q$ of final states, and a transition function δ from $Q \times M$ to Q satisfying the following conditions:

$$\forall q \in Q \qquad \delta(q, 1) = q,$$

$$\forall q \in Q, \quad \forall m_1,\ m_2 \in M \qquad \delta(q, m_1 m_2) = \delta(\delta(q, m_1), m_2).$$

The subset T of M recognized by the automaton $\mathcal{A}$ is defined by

$$T = \{m \in M \mid \delta(q_0, m) \in F\}.$$

Let $T \subseteq M$ be any subset. The *syntactic congruence* $\equiv_T \subseteq M \times M$ is defined by setting $x \equiv_T y$, if $uxv \in T$ is equivalent to $uyv \in T$ for all $u, v \in M$. The quotient monoid $M/\equiv_T$ is called the *syntactic monoid* of T. The canonical homomorphism $M \to M/\equiv_T$ recognizes T. If $\varphi : \Sigma^* \to M$ is a surjective homomorphism, then the syntactic monoids of $T \subseteq M$ and of $\varphi^{-1}(T) \subseteq \Sigma^*$ are isomorphic (via φ). We shall use the following classical equivalences.

Proposition 37. *Let $\varphi : \Sigma^* \to M$ be a surjective homomorphism onto a monoid M and $T \subseteq M$. Then the following assertions are equivalent.*

i) The set T is recognizable.
ii) The syntactic congruence $\equiv_T$ is of finite index.
iii) There exists an M-automaton recognizing T.
iv) The set $\varphi^{-1}(T)$ is a recognizable (or regular) subset of Σ^*.

The family of recognizable sets is denoted by $\mathrm{Rec}(M)$. It is a Boolean algebra. The family of *rational sets,* denoted by $\mathrm{Rat}(M)$, is inductively defined by starting from the finite subsets of M and the closure under the operations of union $X \cup Y$, concatenation $X \cdot Y = \{xy \in M \mid x \in X,\, Y \in Y\}$, and iteration (or Kleene-star) $X^* = \bigcup_{i \geq 0} X^i$, where $X^0 = \{1\}$ and $X^i = X^{(i-1)} \cdot X$ for $i \geq 1$.

If we replace the iteration-operation by complementation, we obtain the family of *star-free* sets, denoted by $\mathrm{SF}(M)$.

For finitely generated free monoids Kleene's Theorem asserts $\mathrm{Rec}(\Sigma^*) = \mathrm{Rat}(\Sigma^*)$; and we can speak of regular languages without ambiguity. Due to Schützenberger, the star-free languages $\mathrm{SF}(\Sigma^*)$ have been characterized by the fact that they are recognized by some finite aperiodic monoid. (A finite monoid S is aperiodic if and only if there exists some $p \geq 0$ such that $x^{p+1} = x^p$ for all $x \in S$.) For example, the language $(ab)^* \subseteq \{a,b\}^*$ is star-free (!), the language $(aa)^* \subseteq \{a\}^*$ is not star-free. Star-free trace languages are recognizable by Cor. 41 below.

The rational subsets of the free commutative monoid $\mathbb{N}^\Sigma$ are exactly the semi-linear sets. There are semi-linear sets, which are not recognizable: Consider $(ab)^* \subseteq \mathbb{N}^{\{a,b\}}$. The commutative closure $\overline{(ab)^*}$ of the star-free language $(ab)^*$ is the non-regular, context-free language $L = \{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$.

For a trace monoid $\mathbb{M} = \mathbb{M}(\Sigma, I)$ we therefore have the following situation with strict inclusions, in general:

$$\mathrm{SF}(\mathbb{M}) \quad \subseteq \quad \mathrm{Rec}(\mathbb{M}) \quad \subseteq \quad \mathrm{Rat}(\mathbb{M}).$$

4.2 Recognizability and Rational Operations

A rational expression over words or traces is called *star-connected,* if the Kleene-star is used over connected languages, only. Let $X \subseteq \Sigma^*$ ($X \subseteq \mathbb{M}(\Sigma, I)$, resp.) be a languages. A word t (trace t, resp.) is called an *iterative factor* of X, if $ut^*v \subseteq X$ for some u, v.

Remark. If all iterative factors of X are connected, then every rational expression for X is star-connected. Indeed, if the star is used over a non-connected word or trace t inside the expression for some rational language L, then t becomes an iterative factor of L.

Another operation for trace languages is called *concurrent-star* (or *c-star* for short). It is in fact the composition of two operations. For a language X we define X^{c-*} to be the Kleene-star $(c(X))^*$, where $c(X)$ is the set of connected components:

$$c(X) = \{u \in \mathbb{M}(\Sigma, I) \mid u \text{ is connected, } u \neq 1 \text{ and } \exists v : uv \in X,\, uIv\}.$$

Thus, X^{c-*} is the Kleene-star over the connected language of all connected components.

The family of c-rational trace languages is defined as the rational sets, but instead of the Kleene-star we allow the c-star operation, only. The following assertion is called Ochmański's Theorem.

Theorem 38. *Let $\varphi : \Sigma^* \to \mathbb{M}(\Sigma, I)$ be the canonical homomorphism and $T \subseteq \mathbb{M}(\Sigma, I)$ be a trace language. Then the following assertions are equivalent.*

i) T is recognizable.
ii) There exist a rational language $X \subseteq \Sigma^*$ such that every iterative factor of X is connected and $\varphi(X) = T$.
iii) T is star-connected.
iv) T is a c-rational language.

Proof. The implication $i) \Rightarrow ii)$ follows from Lem. 45 below.
$ii) \Rightarrow iii)$: As we have remarked above, if every iterative factor is connected, then X is star-connected.
$iii) \Rightarrow iv)$: Every star-connected expression is also a c-rational expression.
$iv) \Rightarrow i)$: It remains to show that recognizable trace languages are closed under taking connected components, concatenation, and the Kleene-star over connected languages. The closure under taking connected components is trivial. The closure under concatenation is stated in Cor. 41, and the closure under Kleene-star over connected languages is stated in Cor. 44. In order to establish these results we shall use below the notion of rank as a tool.

Remark. In Thm. 71 we will see that, in addition to the equivalences above and just as for words, recognizable trace languages can also be characterized by monadic second-order logic.

4.3 The Rank

The *rank* is defined with respect to a fixed independence relation $I \subseteq \Sigma \times \Sigma$. Let $X \subseteq \Sigma^*$ be a subset of words and $x, y \in \Sigma^*$. By Cor. 5 we have $xy \in \overline{X}$ if and only if there are $x_0, y_0, \ldots, x_n, y_n \in \Sigma^*$ such that $x_0 y_0 \cdots x_n y_n \in X$, $[x] = [x_0 \cdots x_n]$, $[y] = [y_0 \cdots y_n]$, and $x_i I (y_{i+1} \cdots y_n)$ for $0 \leq i < n$. For $xy \in \overline{X}$ we define $\mathrm{Rank}(x, y, X)$ to be the least number n such that a factorization as above is possible. The rank of the language X is

$$\mathrm{Rank}(X) = \max\{\mathrm{Rank}(x, y, X) \mid \overline{xy} \in X\}$$

(with the convention $\mathrm{Rank}(\emptyset) = 0$).

We say that X has a finite rank, if there exists an integer k such that $\mathrm{Rank}(X) = k$.

For example, if we assume that aIb then, considering the set of couples $\{(a^n, b^n) \mid n \geq 0\}$, we see that $\mathrm{Rank}((ab)^*)$ is infinite. The rank of a^*b^* is one.

Remark. A subset X of Σ^* has a rank equal to 0, if and only if for all couples (x, y) of words such that $xy \in \overline{X}$, there exist two words x', y' such that $x'y' \in X$, $x \sim x'$, and $y \sim y'$, where $\sim$ is a shorthand of $\sim_I$.

Theorem 39. *If a regular subset X of Σ^* has a finite rank, then $[X]$ is recognizable.*

Proof. Let $T = [X]$, we prove that the family $\{u^{-1}T \mid u \in \mathbb{M}(\Sigma, I)\}$ is finite.

Let k be the rank of X. Let η be a morphism from Σ^* to a finite monoid S recognizing X (i.e., $X = \eta^{-1}(\eta(X))$) such that in addition

$$\eta(u) = \eta(v) \text{ implies } \mathrm{alph}(u) = \mathrm{alph}(v).$$

Clearly such an η exists. Define a finite set $R(u)$ for $u \in \mathbb{M}(\Sigma, I)$ as follows

$$R(u) = \{(s_0, \ldots, s_k) \mid \exists x_0, \ldots, x_k \in \Sigma^*,\, u = [x_0 \cdots x_k],\, s_i = \eta(x_i),\, 0 \le i \le k\}.$$

Note that the number of different $R(u)$ is bounded by $2^{|S|^{(k+1)}}$. We prove that if $R(u) = R(u')$, then $u^{-1}T = u'^{-1}T$. For this assume that $R(u) = R(u')$ and $uv \in T$. We prove that $u'v \in T$. Let $x, y \in \Sigma^*$ be words such that $[x] = u$, $[y] = v$; the set X has a finite rank equal to k and $xy \in \overline{X}$. Thus,

- $x \sim x_0 x_1 \cdots x_k$,
- $y \sim y_0 y_1 \cdots y_k$,
- $x_0 y_0 x_1 y_1 \cdots x_k y_k \in X$,
- $x_i I y_i$ for all $i < j$.

As the morphism η recognizes X,

$$\eta^{-1}(\eta(x_0 y_0 x_1 y_1 \cdots x_k y_k)) \subseteq X.$$

Let $\alpha = \eta^{-1}(\eta(x_0 y_0 x_1 y_1 \cdots x_k y_k))$. Since $x_j I y_i$ for all $i < j$ we can deduce

$$\begin{aligned}\alpha &= \eta^{-1}(\eta(x_0 x_1 \cdots x_k y_0 y_1 \cdots y_k)) \\ &= \eta^{-1}(\eta(x_0)\eta(x_1) \cdots \eta(x_k)\eta(y_0 y_1 \cdots y_k)).\end{aligned}$$

Let $x' \in \Sigma^*$ such that $[x'] = u'$ and x' admits $x'_1, \ldots, x'_k$ as a decomposition verifying $\eta(x'_0) = \eta(x_0), \ldots, \eta(x'_k) = \eta(x_k)$. This is possible due to $R(u) = R(u')$. Now

$$\eta^{-1}(\eta(x'_0)\eta(x'_1) \cdots \eta(x'_k)\eta(y_0 y_1 \cdots y_k)) = \eta^{-1}(\eta(x_0 x_1 \cdots x_k y_0 y_1 \cdots y_k)).$$

Using the fact that η is compatible with alphabets we obtain

$$\alpha \quad = \quad \eta^{-1}(\eta(x'_0 y_0 x'_1 y_1 \cdots x'_k y_k)).$$

Finally, since $\alpha \subseteq X$ we have $x'_0 y_0 x'_1 y_1 \cdots x'_k y_k \in X$ and thus $u'v \in T$.

Remark. Let us note that the converse of this theorem is false. For $\Sigma = \{a, b\}$ with aIb, the closure of the set of words $X = (ab)^* \cdot (a^* \cup b^*)$ is recognizable, since $\overline{X} = \{a, b\}^*$, but the rank of X is not finite.

Proposition 40. *Given two sets of words X_1 and X_2, if they are closed, then the rank of the concatenation $X_1 X_2$ is finite.*

Proof. Let $xy \in \overline{X_1X_2}$, by definition we have $xy \sim x_1x_2$ with $x_1 \in X_1$ and $x_2 \in X_2$. Applying Levi's Lemma there exist $x_1', x_2', x_1'', x_2'' \in \Sigma^*$ such that

$$x \sim x_1'x_1'',\ y \sim x_2'x_2'',\ x_1 \sim x_1'x_2',\ x_2 \sim x_1''x_2'' \quad \textit{and} \quad x_1''Ix_2'.$$

As X_1 and X_2 are closed we have $x_1'x_2' \in X_1$ and $x_1''x_2'' \in X_2$. Thus $x_1'x_2'x_1''x_2'' \in X_1X_2$ and the rank of X_1X_2 equals 1.

Corollary 41. *The family of recognizable trace languages is closed under concatenation.*

Proposition 42. *Let X be a set of connected words. If it is closed, then the rank of X^* is finite.*

This proposition is a direct consequence of the following technical lemma:

Lemma 43. *Let X be a closed subset of Σ^* such that each word is connected. The product xy of two words of Σ^* belongs to $\overline{X^*}$ if and only if there exist $n \geq 0$, $z_i, z_i' \in X^*$, and $x_i, y_i \in \Sigma^*$, $1 \leq i \leq n$, with*

i) $z_0x_1z_1x_2 \cdots x_nz_n \sim x$,
ii) $z_0'y_1z_1'y_2 \cdots y_nz_n' \sim y$,
iii) $x_iy_i \in X$ *for* $1 \leq i \leq n$,
iv) $z_i'I(x_{i+1}z_{i+1} \cdots x_nz_n)$ *for* $0 \leq i \leq n$,
v) $y_iI(z_ix_{i+1} \cdots x_nz_n)$ *for* $1 \leq i \leq n$,
vi) *alph*$(x_i) \neq$ *alph*(x_j) *for* $1 \leq i < j \leq n$.

Proof. This condition is obviously sufficient.

Conversely, we use an induction on the minimal integer k such that $xy \in \overline{X^k}$. If $k = 0$, there is nothing to do. Otherwise, there exist $x' \in X$, and $x'' \in X^{k-1}$ such that $xy \sim x'x''$. By Levi's Lemma, there exist $p, p', q, q' \in \Sigma^*$ such that

$$x \sim pp', y \sim qq', x' \sim pq, x'' \sim p'q' \quad \text{and} \quad p'Iq.$$

As $p'q' \in \overline{X^{k-1}}$, by the induction hypothesis, there are $n \geq 1$, $z_i, z_i' \in X$, $1 \leq i \leq n$, and $x_i, y_i \in \Sigma^*$, $2 \leq i \leq n$, with

i) $z_1x_2z_2x_3 \cdots x_nz_n \sim p'$,
ii) $z_1'y_2z_2'y_3 \cdots y_nz_n' \sim q'$,
iii) $x_iy_i \in X$ for $2 \leq i \leq n$,
iv) $z_i'I(x_{i+1}z_{i+1} \cdots x_nz_n)$ for $1 \leq i \leq n)$,
v) $y_iI(z_ix_{i+1} \cdots x_nz_n)$ for $2 \leq i \leq n)$,
vi) $\mathrm{alph}(x_i) \neq \mathrm{alph}(x_j)$ $2 \leq i < j \leq n$.

Furthermore $\overline{X} = X$ thus $pq \in X$. We distinguish three cases:

1. $p = 1$: Then $q \in X$. We have $p' \sim x$ and $qq' \sim y$. Let $z = qz_1'$. It holds $qz_1' \in X^*$ and qIp', thus $zI(x_2z_2 \cdots x_nz_n)$ and we obtain the lemma with z instead of z_1'.
2. $q = 1$: Let $z = pz_1$ then it suffices to take z instead of z_1.

3. $p \neq 1, q \neq 1$: The word pq is connected and qIx_i for all $i > 1$. Thus, $\text{alph}(p) \neq \text{alph}(x_i)$ for all i. Let $x_1 = p$, $y_1 = q, z_0 = z'_0 = 1$. This yields the desired decomposition.

As $\text{alph}(x_i) \neq \text{alph}(x_j)$ for all $i \neq j$, we have $n \leq |\Sigma|$. Hence we deduce that the rank of X^* is bounded by $|\Sigma|$.

By Thm. 39 and Prop. 42 we obtain the following missing piece of in the proof of the implication Thm. 38, $iv) \Rightarrow i)$.

Corollary 44. *Let T be a recognizable trace language which is connected, then T^* is recognizable.*

4.4 Recognizability and the Lexicographic Normal Form

Consider a total order on the alphabet and the set of lexicographic normal forms of traces *LexNF*. Recall that if $T \subseteq \mathbb{M}(\Sigma, I)$ is recognizable and if $\varphi^{-1}(T)$ denotes the set of representing words, then $X = \varphi^{-1}(T) \cap \mathit{LexNF}$ is a recognizable word language with $T = [X]$.

Lemma 45. *Let $X \subseteq \mathit{LexNF}$ be a regular language. Then every iterative factor of X is connected. In particular, $[X] \subseteq \mathbb{M}(\Sigma, I)$ is a recognizable trace language.*

Proof. We show a stronger assertion: Let $w \in \mathit{LexNF}$ such that $w^2 \in \mathit{LexNF}$ (this condition holds for every iterative factor of X), then w is a connected word. Let $t = [w] \in \mathbb{M}(\Sigma, I)$ and assume by contradiction that t is not connected, i.e., $t = uv$ with $u \neq 1$, $v \neq 1$, and uIv.

Then, assuming the first letter from w being from u, we can factorize w such that

$$\begin{aligned} w &= a_1x_1a_2x_2a_3x_3a_4x_4 \cdots a_{n-1}x_{n-1}a_nx_n, \\ u &= a_1x_1a_3x_3 \cdots a_kx_k, \\ v &= a_2x_2a_4x_4 \cdots a_mx_m \end{aligned}$$

with $a_i \in \Sigma$, $x_i \in \Sigma^*$, $\{k, m\} = \{n-1, n\}$, and $m = n$, if n is even and $k = n$ otherwise. Moreover, due to uIv we have $a_1 < \cdots < a_n$.

Since $w^2 \in \mathit{LexNF}$, all factors of w^2 belong to *LexNF*, too. Hence, if n is even, then $a_mx_ma_1 \in \mathit{LexNF}$. If n is odd, then $n \geq 3$ and $a_kx_ka_1x_1a_2 \in \mathit{LexNF}$. In the first case, we have $a_1 < a_m$ and $(a_mx_m)Ia_1$; in the second case, we have $a_2 < a_k$ and $(a_kx_ka_1x_1)Ia_2$. Hence, a contradiction follows in both cases.

Note that Lem. 45 implies $i) \Rightarrow ii)$ of Thm. 38. The following statement is a corollary. It is also a consequence of Thm. 71 below. This theorem will be shown without using any result from this section.

Corollary 46. *For a trace language $T \subseteq \mathbb{M}(\Sigma, I)$ let $\mathit{Min}(T) = \{x \in \mathit{LexNF} \mid [x] \in T\}$. Then T is a recognizable trace language if and only if $\mathit{Min}(T) \subseteq \Sigma^*$ is regular.*

4.5 The Star Problem and the Finite Power Property

The star problem is to decide for a given recognizable trace language $T \subseteq \mathbb{M}(\Sigma, I)$ whether T^* is recognizable. It is not known whether the star problem is decidable. There is a close connection to the finite power property (FPP).

A language $T \subseteq \mathbb{M}(\Sigma, I)$ is said to satisfy FPP, if $T^* = (T \cup \{1\})^k$ for some $k \geq 0$. We say that FPP is decidable on $\mathbb{M}(\Sigma, I)$, if FPP for recognizable languages of $\mathbb{M}(\Sigma, I)$ is decidable.

If a recognizable language T satisfies FPP, then T^* is recognizable, too. The converse need not be true, e.g., let $\emptyset \neq \Gamma \subseteq \Sigma$. Then $\Gamma^* \subseteq \mathbb{M}(\Sigma, I)$ is recognizable, it is even star-free, but Γ does not satisfy FPP.

Theorem 47 [84]. *If the independence alphabet (Σ, I) does not contain a C_4 (cycle of four letters) as an induced subgraph, then both, the star problem and the finite power problem are decidable.*

Remark. Thm. 47 applies to free commutative monoids. Note however that for commutative monoids the star problem and FPP are questions about semi-linear sets (or equivalently questions which can be formulated in Presburger arithmetic), hence the decidability is also clear from classical results, [43, 55].

Both, the star problem and FPP are semi-decidable. For FPP this is clear, simply compute T^k until $T^k \subseteq T^{k-1}$, $k \geq 1$. For the star problem, the existence of a semi-algorithm to this problem is a consequence of the following fact:

Proposition 48. *Let $T \subseteq \mathbb{M}(\Sigma, I)$ be a recognizable trace language. Then T^* is recognizable if and only if there exists a recognizable language $K \subseteq \mathbb{M}(\Sigma, I) \setminus \{1\}$ such that both $K^2 \subseteq K$ and $(K \setminus K^2) \subseteq (T \setminus \{1\}) \subseteq K$. Moreover, if such a K exists, then $K = T^* \setminus \{1\}$, and $K \setminus K^2$ is the minimal set of generators for T^*.*

A semi-algorithm to decide the star problem computes all K, patiently, one after the other, testing each time the three inclusions $K^2 \subseteq K$ and $(K \setminus K^2) \subseteq (T \setminus \{1\}) \subseteq K$.

Let $X \subseteq \Sigma^*$. If X possesses the finite power property in Σ^*, then $[X]$ possesses the finite power property in $\mathbb{M}(\Sigma, I)$. This implication may not be reversed, in general:

Example 5. Let $\Sigma = \{a, b\}$, aIb, and $X = (a^2)^* + (b^2)^* + ab + ba$. This set does not possess the finite power property, since for all integer n, the word $(ab)^n$ belongs to X^n but not to X^i for all $i < n$. Nevertheless, the subset $[X]$ of $\mathbb{M}(\Sigma, I)$ verifies $[X]^* = [X]^3$.

The basic relation between the star problem and FPP is due to the following proposition:

Proposition 49. *Let $T \subseteq \mathbb{M}(\Sigma, I)$ be recognizable and let b be a new letter. Define $K = T \cdot b^+ \subseteq \mathbb{M}(\Sigma, I) \times b^*$. Then the following assertions are equivalent*

i) T satisfies the finite power property.
ii) K satisfies the finite power property.
iii) K^* is recognizable.

Proof. Without restriction we have $1 \in T$. Clearly, if $T^* = T^k$ for some $k \geq 0$, then we have $K^{k+1} = K^k$. This proves $i) \Rightarrow ii) \Rightarrow iii)$.
For the converse observe that if $xb^k \in K^*$ with $x \in \mathbb{M}(\Sigma, I)$, $k \geq 1$, then we must have $x \in T^{k'}$ for some $k' \leq k$. (In fact $x \in T^k$, since $1 \in T$.) Now assume that K^* is recognizable. By the pumping lemma for regular languages (uvw-Theorem) there exists some $n \geq 0$ such that whenever $xb^m \in K^*$ we find some $k \leq n$ with $xb^k \in K^*$. As we have just seen this implies $x \in T^k \subseteq T^n$, and hence $T^* \subseteq T^n$.

As a final result of this section, let us show that FPP is decidable for connected languages. For this we recall the notion of *distance function* on automata. A path in the automaton $\mathcal{A}$ is a sequence $\mathcal{C} = (f_1, \ldots, f_n)$ of consecutive transitions $f_i = (q_i, a_i, q_{i+1})$, $1 \leq i \leq n$. The integer n is called the length of the path $\mathcal{C}$. The word $w = a_1 \cdots a_n$ is the label of $\mathcal{C}$. The state q_1 is the origin of the path and the state q_{n+1} its end. By convention, there is for each state $q \in Q$ a path of length 0 from q to q. Its label is the empty word 1. A path $\mathcal{C}$ from q to q' is successful if q is an initial state, q' is final, i.e., $q \in Q_0$ and $q' \in F$. The set recognized by $\mathcal{A}$, denoted by $L(\mathcal{A})$ is defined as the set of labels of successful paths.

A *distance automaton* $(\mathcal{A}, d)$ is a pair where $\mathcal{A} = (\Sigma, Q, \Delta, Q_0, F)$ is a nondeterministic automaton and d is a distance function

$$d : Q \times \Sigma \times Q \longrightarrow \{0, 1, \infty\}$$

satisfying for all $(q, a, q') \in Q \times \Sigma \times Q$ the property $d(q, a, q') = \infty$ if and only if $(q, a, q') \notin \Delta$.

For a path $\mathcal{C}$, we note $d(\mathcal{C}) = \sum_{i=1}^{n} d(q_i, a_i, q_{i+1})$. The function d is extended on $Q \times \Sigma^* \times Q$ by : $d(q, w, q')$ being the minimum of the $d(\mathcal{C})$ over all paths $\mathcal{C}$ from q to q' labeled by w. If there does not exist such a path, we set $d(q, w, q') = \infty$.

A distance automaton $(\mathcal{A}, d))$ is called *limited in distance,* if there exists an integer k such that for all $w \in L(\mathcal{A})$ we have $d(q, w, q') \leq k$ for some $q \in Q_0$ and $q' \in F$.

The following result of Hashiguchi [59] will be used.

Theorem 50. *It is decidable whether a finite distance automaton $(\mathcal{A}, d)$ is limited in distance.*

As in the word case this theorem can be used for the following assertion.

Theorem 51. *Let T be a recognizable subset of $\mathbb{M}(\Sigma, I)$ such that each trace of T is connected. Then it is decidable whether T possesses the finite power property.*

Proof. From Corollary 44 we know that T^* is recognizable. We describe a finite distance automaton recognizing T^* and having limited distance if and only if T^* has the finite power property.

To simplify notations we use a finite monoid S recognizing at the same time T and T^*, and where moreover the elements of S code the alphabet. Thus, we let η be a homomorphism from $\mathbb{M}(\Sigma, I)$ onto S such that

- $T = \eta^{-1}(\eta(T))$,
- $T^* = \eta^{-1}(\eta(T^*))$,
- and $\eta(u) = \eta(v)$ implies $\mathrm{alph}(u) = \mathrm{alph}(v)$.

Note that $\eta(T^* \setminus \{1\}) = \eta(T^*) \setminus \{1\}$ due to the assumptions above. We now give the description of the automaton. The states are tuples of various length $(s_0, s_1, \ldots, s_{2m}) \in S^{2m+1}$ with $m \leq n$, where $n = |\Sigma|$. Furthermore we have $s_i \in \eta(T^*)$, if i is even. The s_i, where i is odd, are called active elements. There is a single initial state (1) where $1 \in S$ denotes the unit element.

There are two sorts of $\epsilon-$transitions.

- The first one has distance zero and is always allowed if $m < n$. For any $0 \leq i \leq 2m$ we may perform the following operation

  $$\text{change } (s_0, \ldots, s_i, s_{i+1}, \ldots, s_{2m}) \text{ to } (s_0, \ldots, s_i, 1, 1, s_{i+1}, \ldots, s_{2m}).$$

  This transition creates a new active component.
- The other $\epsilon-$transition has distance one; it decides that an active component is not used anymore, since a factor is completed. For $s_i \in \eta(T)$, i odd, we may perform with distance one:

  $$\text{change } (s_0, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_{2m}) \text{ to } (s_0, \ldots, s_{i-1} \cdot s_i \cdot s_{i+1}, \ldots, s_{2m})$$

Thus, with distance one a $(2m+1)$-tuple is transformed into a $(2m-1)$-tuple. Now we describe the action of a letter. Let $a \in \Sigma$; there are two types of transitions. If i is odd and a is independent of the alphabets corresponding to s_j where $i < j$, then we may perform:

$$\text{reading } a \text{ change } (s_0, \ldots, s_i, \ldots, s_{2m}) \text{ to } (s_0, \ldots, s_i \cdot \eta(a), \ldots, s_{2m}),$$

This transition has distance zero.

For i even, we allow the same transformation, if a is independent of the alphabet corresponding to s_j where $i < j$ and if, in addition, $\eta(a) \in \eta(T)$. In this case the transition has distance one.

The final states are the 1-tuples (s_0) with $s_0 \in \eta(T^*)$.

To see the correctness of this construction, observe first that when there is a path with label t from the initial to a final state with distance k then $t \in T^k$. Conversely, we have to prove that if $t \in T^k$ then there is a path with distance k labeled with t starting at (1) and ending in some (s_0) with $s_0 \in \eta(T^*)$.

Let $t = uv \in T^k$, we can write $u = u_1 \cdots u_k$, $v = v_1 \cdots v_k$ such that $u_i v_i \in T$, $v_i I u_j$ for all $i < j$. We call the index i active when $u_i \neq 1$, and $v_i \neq 1$. At most n indices can be active, since T is a connected language. Hence, if m

is the number of active indices, we have $m \leq n$. Let $d = |\{v_i \mid v_i = 1\}|$. The set $\{v_i \mid v_i = 1\}$ contains the factors which are completed. We group the corresponding u_i together into one factor, including those u_i where $u_i = 1$, and we rename them as u_i with i even. We rewrite $u = u_0 u_1 \cdots u_{2m}$ such that $u_1, u_3, u_5, \ldots$ agree with the active u_i from the old factorization. It follows that the u_i, i even, are products such that $\eta(u_i) \in \eta(T^*)$. Now, using an induction on $|u|$ one proves that reading u we may reach the state $(\eta(u_0), \eta(u_1), \ldots, \eta(u_{2m}))$ with distance d. At the end of this procedure there are no active components anymore, hence $m = 0$, $d = k$. The result follows.

4.6 An Algorithm to Compute Closures

Let (Σ, I) be an independence alphabet. Given two words x and y of Σ^*, the *I-shuffle* of x and y, denoted by $x \amalg_I y$, is the set of words of the form $x_1 y_1 \cdots x_n y_n$ with $x = x_1 \cdots x_n$, $y = y_1 \cdots y_n$, $x_i, y_i \in \Sigma^*$ for all $1 \leq i \leq n$, and $x_j I y_i$ for all $1 \leq i < j \leq n$. The inclusion $x \amalg_I y \subseteq \overline{x \cdot y}$ is straightforward. The I-shuffle is extended to sets by

$$X \amalg_I Y = \bigcup \{x \amalg_I y \mid x \in X, \, y \in Y\}.$$

Remark. The standard shuffle operation $\amalg$ on words can not be expressed with the I-shuffle. We have $bab \in ab \amalg b$, but never $bab \in ab \amalg_I b$, since $(b, b) \notin I$. On the other hand:

$$x \amalg y = x \amalg_I y \text{ for all } xIy.$$

We will apply the I-shuffle mainly to closed languages. Then the I-shuffle becomes more powerful.

Lemma 52. *For all $x, y \in \Sigma^*$ we have*

$$\overline{x \cdot y} = \overline{x} \amalg_I \overline{y}$$

We are now ready to define the procedure P. This procedure adds to X the words belonging to the sets:

$$z_0 (x_1 \amalg_I y_1) z_1 \cdots z_{n-1} (x_n \amalg_I y_n) z_n$$

where $z_0 x_1 y_1 z_1 \cdots z_{n-1} x_n y_n z_n \in X, x_i, y_i \in \Sigma^+, z_0, z_i \in \Sigma^*$ for all $1 \leq i \leq n, n \geq 0$.

There is no need to emphasize the factors z_i explicitly. Observing that $\{z_i\} = z_i \amalg_I 1$ and allowing the factors x_i (and/or y_i) to be the empty word we obtain a more compact notation. Formally, let x be an element of Σ^*, then we define:

$$P(x) = \{y \in \Sigma^* \mid y \in (x_1 \amalg_I y_1) \cdots (x_n \amalg_I y_n), \; x = x_1 y_1 \cdots x_n y_n, \; x_i, y_i \in \Sigma^*\}.$$

The definition of P is extended in a natural way to a set X of words by

$$P(X) = \bigcup_{x \in X} P(x).$$

For example, let a and b be two letters such that aIb. We have

- $P(a^*b) = a^*ba^*$,
- $P(\{abab\}) = \{abab, baab, abba, aabb, baba\}$,
- $P^2(\{abab\}) = P(\{abab\}) \cup \{bbaa\} = \overline{\{abab\}}$.

Due to the definition of P, for every set of words X we have $P(X) \cup P(Y) = P(X \cup Y)$, $P(X) \cdot P(Y) \subseteq P(X \cdot Y)$, and $(P(X))^* \subseteq P(X^*)$. If $X = X^*$, then we have $P(X) = (P(X))^*$.

Moreover, we have $X \subseteq P(X) \subseteq \overline{X}$, and $X = P(X)$ if and only if $X = \overline{X}$. Note also that for any word $x = a_1 \cdots a_n a_{n+1}$, the n-fold application P^n computes the closure of x:

$$[x] = P(a_1 \cdot P(\cdots P(a_n a_{n+1}) \cdots)) \subseteq P^n(x).$$

(The exponent n is an overestimation, Cor. 55 below yields that an $\lceil \log n \rceil$-fold application is enough in order to compute the closure of a word of length n.) We have

$$\overline{X} = \bigcup_{i \geq 0} P^i(X) = P^*(X).$$

Remark. The procedure P is a modification of the procedure S introduced by Métivier [73] The procedure S adds to X the words belonging to the sets:

$$(x_1 \text{Ш} y_1) \cdots (x_n \text{Ш} y_n),$$

where $x_1 y_1 \cdots x_n y_n \in X$ and for all i we have $\mathrm{alph}(x_i) \times \mathrm{alph}(y_i) \subseteq I$. The relation between both procedures is as follows:

$$X \subseteq S(X) \subseteq P(X) \subseteq S^{|\Sigma|}(X) \subseteq P^{|\Sigma|}(X) \subseteq \overline{X}.$$

Theorem 53. *If X is recognizable, then $P(X)$ is recognizable.*

Proof. Let $\eta : \Sigma^* \longrightarrow S$ be a homomorphism onto a finite monoid S recognizing X, i.e., $X = \eta^{-1}(\eta(X))$. As usual, we may assume that $\eta(x) = \eta(y)$ implies $\mathrm{alph}(x) = \mathrm{alph}(y)$ for all $x, y \in \Sigma^*$. Thus, for $p \in S$ we define by $\mathrm{alph}(p) = \mathrm{alph}(x)$ where $p = \eta(x)$. Define the state set $Q = S \times S$. We allow ϵ-transitions and two types of a-transitions, $a \in \Sigma$.

$$\begin{array}{ll} \delta((p,q),1) = (pq,1) & \text{for all } (p,q) \in Q, \\ \delta((p,q),a) = (p,q\eta(a)) & \text{for all } (p,q) \in Q, a \in \Sigma, \\ \delta((p,q),a) = (p\eta(a),q) & \text{for all } (p,q) \in Q, a \in \Sigma, \text{ where } \mathrm{alph}(q) \times \{a\} \subseteq I. \end{array}$$

The initial state is $(1,1)$ and the final states are the pairs $(p,1)$ with $p \in \eta(X)$.

It is not difficult to see that the automaton just defined recognizes $P(X)$.

The following example is due to Arnold. It shows that every finite iteration of P may fail to compute the closure of a recognizable set even when its closure is recognizable:

Example 6. Let $\Sigma = \{a, b\}$ with aIb, and $X = (ab)^*(\{1\} \cup a^+ \cup b^+)$. We have $\overline{X} = \Sigma^*$, and by induction, we can prove that $a^{3^{n+1}} b^{3^{n+1}} \notin P^n(X)$ for all $n \geq 0$. Thus, for each integer n we have $P^n(X) \neq \overline{X}$.

Remark. This behavior above is not a particular failure of the procedure P. In fact, let for a moment $\tilde{P} : \mathcal{P}(\Sigma^*) \longrightarrow \mathcal{P}(\Sigma^*)$ be any procedure satisfying the following properties:

I $\tilde{P}(X \cup Y) = \tilde{P}(X) \cup \tilde{P}(Y)$,
II $X \subseteq \tilde{P}(X) \subseteq \overline{X}$,
III If X is recognizable, then $\tilde{P}(X)$ is recognizable.

Then $\tilde{P}^n(X) \neq \overline{X}$ for all n, in general, even when X and $\overline{X}$ are recognizable. Indeed, assume that for $\Sigma = \{a, b\}$ with aIb we would have

$$\tilde{P}^n((ab)^*(\{1\} \cup a^+ \cup b^+)) = \Sigma^* \text{ for some } n \in \mathbb{N}.$$

Then $\tilde{P}^n((ab)^*) \cup \tilde{P}^n((ab)^*(a^+ \cup b^+)) = \Sigma^*$ due to I. By II we obtain $\tilde{P}^n((ab)^*) = \{w \in \Sigma \mid |w|_a = |w|_b\}$. Finally the latter set would be recognizable by III, hence a contradiction.

We deduce from the previous remark that neither P nor any other procedure satisfying I, II, and III provides us with a semi-algorithm to compute the I-closure on the class of recognizable languages. However, the following results show that we can use the procedure P to compute closures in the cases described in previous sections.

Proposition 54. *Let (Σ, I) be an independence alphabet. For all I-closed subsets X_1, X_2 of Σ^*, we have:*

$$\overline{X_1 \cdot X_2} = P(X_1 \cdot X_2).$$

Proof. This follows from Lem. 52.

Corollary 55. *Let (Σ, I) be an independence alphabet and $X_1, \ldots, X_n$ be I-closed subsets of Σ^*. Then we have:*

$$\overline{X_1 \cdots X_n} = P^{\lceil \log n \rceil}(X_1 \cdots X_n).$$

Theorem 56. *Let (Σ, I) be an independence alphabet and $\omega(\Sigma, I)$ be the maximal number of pairwise independent letters. Let X be any I-closed subset of Σ^* containing connected words, only. Then we have:*

$$\overline{X^*} = P^{2|\Sigma|\omega(\Sigma,I)}(X^*).$$

Proof. Let $w_1, \ldots, w_m \in X \setminus \{1\}$ be a list of m non-empty words, and $w \in \Sigma^*$ such that $w \in [w_1 \cdots w_m]$. Each w_i can be identified with a (scattered) subword $\hat{w}_i$ of w. More precisely, let $n = |w|$ and $w = a(1) \cdots a(n)$ with $a(j) \in \Sigma$. We say that a position j with $a = a(j)$, $1 \leq j \leq n$ belongs to $\hat{w}_i$, if

$$|w_1 \cdots w_{i-1}|_a < |a(1) \cdots a(j)|_a \leq |w_1 \cdots w_{i-1} w_i|_a.$$

For each w_i let first(i) (last(i) resp.) be the first (last resp.) position belonging to $\hat{w}_i$. This yields an interval

$$\mathrm{int}(i) = [\mathrm{first}(i), \mathrm{last}(i)] \subseteq \{1, \ldots, n\}.$$

The crucial observation is due to the fact that all words of X are connected. As it can be seen from the proof of Prop. 42 (or be shown directly) there is no position j where more than $|\Sigma|$ intervals intersect:

$$|\{i \mid j \in \mathrm{int}(i)\}| \leq |\Sigma| \quad \text{for all } 1 \leq j \leq n.$$

We now use at most $|\Sigma|$ values *blue, green, red,* etc. in order to give a first coloring of w. We demand that positions belonging to the same $\hat{w}_i$ have the same color and in addition that the colors of $\hat{w}_i$ and $\hat{w}_j$ are different, if $\mathrm{int}(i) \cap \mathrm{int}(j) \neq \emptyset$ for all $i \neq j$. We need in fact a finer coloring (or a second coloring). To define this consider a certain color, say *red*. Let $\hat{w}_{i_1}, \ldots, \hat{w}_{i_r}$, $1 \leq i_1 \leq \cdots \leq i_r \leq n$ be the list of red subwords. We define a graph $G(\text{red}) = (V(\text{red}), E(\text{red}))$ with vertex set $V(\text{red}) = \{1, \ldots, r\}$ and $pq \in E(\text{red})$ if and only if $p < q$, but $\mathrm{first}(i_p) > \mathrm{last}(i_q)$. Since the intervals of the red subwords do not intersect, $G(\text{red})$ is a permutation graph. Hence it is a perfect graph and its chromatic number is equal to the cardinality of its largest clique. A clique in $G(\text{red})$ however corresponds to pairwise independent subwords. This is clear, since $pq \in E(\text{red})$ implies $\mathrm{alph}(\hat{w}_{i_p}) \times \mathrm{alph}(\hat{w}_{i_q}) \subseteq I$. Therefore at most $\omega(\Sigma, I)$ different shades of red suffice such that p and q have different colors for all $pq \in E(\text{red})$. In total we need at most $|\Sigma|\omega(\Sigma, I)$ colors such that the following invariant is satisfied:

> Whenever $i \neq j$ and $\mathrm{int}(i) \cap \mathrm{int}(j) \neq \emptyset$ or $i < j$ and $\mathrm{first}(i) > \mathrm{last}(j)$, then i and j have different colors.

Finally, we *mark* a color, if all subwords having this color are factors of w, i.e., for all subwords $\hat{w}_i$ of a marked color it holds $|\mathrm{int}(i)| = |w_i|$. Now let k be the number of unmarked colors, $k \leq |\Sigma|\omega(\Sigma, I)$.

Since X is I-closed and we are interested in $\overline{X^*}$, it is enough to prove by induction on k that for some permutation σ of $\{1, \ldots, m\}$ we have

$$w \in P^{2k}(\hat{w}_{\sigma(1)} \cdots \hat{w}_{\sigma(m)}).$$

If all colors are marked, i.e., $k = 0$, then we have $w = \hat{w}_{\sigma(1)} \cdots \hat{w}_{\sigma(m)}$ and the claim is true. Let $k > 0$. Since X is I-closed we may henceforth assume that $w_i = \hat{w}_i$ for all i. We define a new word w' by grouping the subwords with color k into factors. Assume that color k is *purple* (viewed as a shade of red). Then formally, let $w_{i_1}, \ldots, w_{i_p}$ be the list of all purple subwords in w, listed from left to right as they appear in w. Write $w = u_1 \cdots u_p$ such that each u_q contains exactly one purple subword $w_{i_q}, 1 \leq p$ and such that the first letter of each $u_2, \ldots u_p$ is purple too. Let u'_q (u''_q, resp.) denote the scattered subword of u_q containing the positions belonging to $w_1, \ldots, w_{i_q - 1}$ ($w_{i_q + 1}, \ldots, w_m$, resp.). Then $u_q \in [u'_q w_{i_q} u''_q]$ and more precisely

$$u_q \in (u'_q \text{Ш}_I w_{i_q}) \text{Ш}_I u''_q.$$

Next, we define

$$w' = (u'_1 w_{i_1} u''_1) \cdots (u'_p w_{i_p} u''_p).$$

Clearly, $w \in P^2(w')$. Moreover, $w' \in [w_1 \cdots w_m]$ and the same coloring being inherited from w satisfies the invariant above. (This is the main point of the proof. It is not totally obvious since in w' there may be new intersections of intervals.) Now the color k (= purple) can be marked. Hence, by induction $w' \in P^{2(k-1)}(w_{\sigma(1)} \cdots w_{\sigma(m)})$ for some permutation σ. The result follows from

$$w \in P^2(P^{2(k-1)}(w_{\sigma(1)} \cdots w_{\sigma(m)})) \subseteq P^{2|\Sigma|\omega(\Sigma,I)}(w_{\sigma(1)} \cdots w_{\sigma(m)})).$$

Corollary 57. *Let X be a recognizable set of words such that all its iterative factors are connected or, more general, let X be star-connected. Then there exists an integer n such that $P^n(X) = \overline{X}$.*

Proof. This can be seen by structural induction over the star-connected expression for X using Cor. 55 and Thm. 56.

Corollary 58. *Let $\mathbb{M}(\Sigma, I)$ be a trace monoid. There exists an integer n such that for each subset X of Σ^* containing $Min([X])$ we have $P^n(X) = \overline{X}$.*

Proof. By Lem. 45 all iterative factors of the set of lexicographic normal forms *LexNF* are connected. By Cor. 57 there is an integer n such that $P^n(\mathrm{Min}([x])) = \overline{x}$ for all $x \in \Sigma^*$. Hence $P^n(X) = \overline{X}$, if $\mathrm{Min}([X]) \subseteq X$.

The hypothesis $\mathrm{Min}([X]) \subseteq X$ is not always necessary.

Example 7. Let $\Sigma = \{a, b\}$ with aIb and $a < b$. Let $X = b^+a^+$. Then

$$\overline{X} = P(X) = b^+ Ш_I a^+ = \Sigma^* \setminus (a^* \cup b^*)$$

is recognizable. But $\mathrm{Min}(X) = a^+b^+$ (and hence $\mathrm{Min}(X) \cap X = \emptyset$).

4.7 Bibliographical Remarks

The investigation of recognizable subsets is central in the theory of traces. The closure under concatenation has been shown by Fliess [46], a much simpler proof for this result has been given by Cori and Perrin [21]. The recognizability of star-connected languages, Cor. 42, has been independently proved by Clerbout (for semi-commutations) [15], Métivier [71], and by Ochmański [77]. The notion of iterative factor is from [70]. The notion of rank has been introduced by Hashiguchi, [60] allowing to shorten some proofs. The closure properties can also be deduced from logic, see [34, Chapt. 10] for details. Lem. 45 is from [77]. For a generalization to concurrency monoids see [36]. A straightforward proof of Cor. 46 by monadic second order logic (without involving c-rational operations) has been independently given by Courcelle [22].

The relation between the star problem and FPP in Sect. 4.5 is due to Richomme [84]. Preliminary results are shown in [49].

The procedure P has been defined first in a technical report being the basis of [75], where the results are shown with respect to the procedure S. This procedure is originally from the thesis [73], see also [25, Problem 246]. Thm. 56 is a positive answer to this Problem 246 and Cor. 58 leads to another proof for showing that star-connected languages are recognizable.

5 Rational Trace Languages

This section contains some characterization- and decidability results about rational trace languages known in the literature. Due to the lack of space we deal with some few topics, only. In particular we do not speak about any counting techniques.

5.1 Unambiguous Languages

Definition 59. A trace language $T \subseteq \mathbb{M}(\Sigma, I)$ is called *k-sequential*, $k \geq 1$, if there is a regular word language $L \subseteq \Sigma^*$ such that $\varphi(L) = T$ and $|\varphi^{-1}(t) \cap L| \leq k$ for all $t \in T$. The family of k-sequential languages is denoted by $\mathrm{Rat}_k(\mathbb{M})$.

It is clear from Cor. 46 that every recognizable trace language is 1-sequential. The inclusion $\mathrm{Rat}_k(\mathbb{M}) \subseteq \mathrm{Rat}(\mathbb{M})$ is trivial for all $k \geq 1$. It has been shown by Bertoni, Mauri, and Sabadini [6] that the following inclusions are proper for some independence alphabets:

$$\mathrm{Rec}(\mathbb{M}) \subseteq \mathrm{Rat}_1(\mathbb{M}) \subseteq \mathrm{Rat}_2(\mathbb{M}) \subseteq \cdots \subseteq \bigcup_{k \geq 1} \mathrm{Rat}_k(\mathbb{M}) \subseteq \mathrm{Rat}(\mathbb{M}).$$

Example 8. Let $(a, b) \in I$, then $(ab)^* \subseteq \mathbb{M}(\Sigma, I)$ is 1-sequential but not recognizable.

Example 9. Let $(\Sigma, I) = a - b - c$. Then the language $T = (ab)^* c^* \cup a^* (bc)^*$ is 2-sequential but not 1-sequential. Clearly, T is 2-sequential. We sketch the proof for the fact that T is not 1-sequential. Assume by contradiction that there exists a regular language $L \subseteq \{a, b, c\}^*$ such that $\varphi(L) = T$ and $|\varphi^{-1}(t) \cap L| = 1$ for all $t \in T$. Consider a deterministic finite automaton accepting L. Replace all b-transitions by ϵ-transitions. In this way we obtain an automaton $\mathcal{A}$ accepting $a^* c^*$. (Note that this automaton has no ϵ-loops.) By the property of L there exists for $a^m c^m$, $m \geq 0$ exactly one accepting path in $\mathcal{A}$, for $a^m c^n$, $m \neq n$ there are exactly two. Based upon a product automaton construction for $\mathcal{A}$ one can show that the following language is regular:

$$\{w \in a^* c^* \mid \text{ there exists more than one path of } \mathcal{A} \text{ accepting } w \}$$

We obtain a contradiction since $\{a^m c^n \mid n \neq m\}$ is not regular.

Definition 60. The *unambiguous rational operations* are defined by the following restrictions of the rational operations.

- The union $X \cup Y$ is allowed only if $X \cap Y = \emptyset$.
- The concatenation $X \cdot Y$ is allowed only if for all $x, x' \in X$, $y, y' \in Y$ the equality $xy = x'y'$ implies $x = x'$ and $y = y'$.
- The Kleene-star X^* is allowed only if X is the basis of a free monoid, i.e., only if $x_1 \cdots x_m = y_1 \cdots y_n$ with $x_i, y_j \in X$, $1 \leq i \leq m$, $1 \leq j \leq n$ implies $m = n$ and $x_i = y_i$ for all $1 \leq i \leq m$.

The family of *unambiguous* rational languages, $\mathrm{UR}(\mathbb{M})$, is the closure of the set of finite languages under unambiguous rational operations.

It is a classical result that regular languages are unambiguous rational. From this statement we may derive:

Proposition 61. *The family of unambiguous rational trace languages coincides with the family of* 1*-sequential rational languages, i.e.,* $UR(\mathbb{M}) = Rat_1(\mathbb{M})$.

Proof. The inclusion $\mathrm{UR}(\mathbb{M}) \subseteq \mathrm{Rat}_1(\mathbb{M})$ is shown by structural induction. For the converse, let $T \in \mathrm{Rat}_1(\mathbb{M})$ and $L \subseteq \Sigma^*$ be a regular language such that $\varphi(L) = T$ and $|\varphi^{-1}(t) \cap L| = 1$ for all $t \in T$. Then any unambiguous rational expression for L yields an unambiguous expression for T.

A well-known result of Eilenberg and Schützenberger [43] says that semi-linear sets are unambiguous rational. This statement has a generalization:

Theorem 62. *We have* $UR(\mathbb{M}(\Sigma, I)) = Rat(\mathbb{M}(\Sigma, I))$ *if and only if* I *is transitive, i.e.,* $\mathbb{M}(\Sigma, I)$ *is a free product of commutative monoids.*

Proof. A proof of the if-part by generating functions is given in [34, Chapt. 5]. For the converse consider three distinct letters a, b, and c such that $(a,b),(b,c) \in I$, but $(a,c) \notin I$. Let $T = (ab)^*c^* \cup a^*(bc)^*$. It is shown in Ex. 9 that T is not 1-sequential, hence $T \in \mathrm{Rat}(\mathbb{M}(\Sigma, I)) \setminus \mathrm{UR}(\mathbb{M}(\Sigma, I))$.

Theorem 63. *The family of rational trace languages is an effective Boolean algebra if and only if* (Σ, I) *is transitive.*

Proof. For a proof of the if-part see [86]. The other direction is easy and shown in the next lemma.

Lemma 64. *If* (Σ, I) *is not transitive, then* $Rat(\mathbb{M}(\Sigma, I))$ *is not closed under intersection.*

Proof. Take three distinct letters a, b, and c such that $(a,b),(b,c) \in I$, but $(a,c) \notin I$. Consider $T = (ab)^*c^* \cap a^*(bc)^*$. We have $T = \{a^n b^n c^n \in \mathbb{M}(\Sigma, I) \mid n \geq 0\}$. If T would be rational, then $\pi_{a,c}(T) \subseteq \{a,c\}^*$ would be rational, too. However $\pi_{a,c}(T) = \{a^n c^n \in \{a,c\}^* \mid n \geq 0\}$. Hence, a contradiction.

5.2 Decidability Results

Consider the following decision problems, where each instance consists of rational trace languages $R, T \subseteq \mathbb{M}(\Sigma, I)$.

- **Inclusion** $\mathrm{INC}(\Sigma, I)$:
  Question: Does $R \subseteq T$ hold?

- **Equality** EQU(Σ, I):
  Question: Does $R = T$ hold?

- **Universality** UNI(Σ, I):
  Question: Does $R = \mathbb{M}(\Sigma, I)$ hold?

- **Complementation** COM(Σ, I):
  Question: Is $\mathbb{M}(\Sigma, I) \setminus R$ a rational (finite resp.) trace language?

- **Recognizability** REC(Σ, I):
  Question: Is R recognizable?

- **Intersection** INT(Σ, I):
  Question: Does $R \cap T = \emptyset$ hold?

The intersection problem plays a special role. It turns out that it is the easiest among the problems above.

Theorem 65. *INT(Σ, D) is decidable if and only if (Σ, I) is a transitive forest.*

The proof of this result is given in [1]. Recall that transitive forests are characterized by forbidden induced subgraphs C_4 and P_4. In particular, every transitive independence alphabet (Σ, I) is a transitive forest.

Theorem 66. *If the independence relation I is transitive, then all decision problems above are decidable.*

Proof. The decidability of INC(Σ, I), EQU(Σ, I), UNI(Σ, I), and INT(Σ, I) follows from Thm. 63. The answer to COM(Σ, I) is always "yes". REC(Σ, D) is for commutative and free monoids a question about semi-linear sets being decidable by Presburger arithmetic [55]. In [86] it is shown that this property is preserved under taking free products of them yielding the result.

Theorem 67. *If any of the problems*

$$INC(\Sigma, I),\ EQU(\Sigma, I),\ UNI(\Sigma, I),\ COM(\Sigma, I),\ or\ REC(\Sigma, I)$$

is decidable, then the independence relation I is transitive.

Proof. All undecidability proofs for the problems above follow the same scheme. This scheme is due to Ibarra [63], who proved that universality of rational transductions from $\{a, c\}^*$ to b^* is undecidable. Our notation is borrowed from [86].

Let us start with the following formulation of the PCP.

Instance: Two homomorphisms $f : A^+ \longrightarrow B^+$ and $g : A^+. \longrightarrow B^+$
Question: Does exist some word $w \in A^+$ such that $f(w) = g(w)$.

Let $C = A\dot{\cup}B$ be the disjoint union of the two alphabets and $\mathbb{M} = C^* \times \mathbb{N}$. Consider the following two languages

$$W(f) = \{(uf(u), n) \in \mathbb{M} \mid u \in A^+, n = |f(u)|\}$$
$$W(g) = \{(ug(u), n) \in \mathbb{M} \mid u \in A^+, n = |g(u)|\}$$

The complements of $W(f)$ and $W(g)$ are rational languages. This can easily be seen by using a non-deterministic two-tape automaton: Assume, we want to accept all $(x, n) \in \mathbb{M}$ not belonging to $W(f)$. (We may assume that the number $n \in \mathbb{N}$ is coded in unary on the second tape.) Of course, the automaton is able to guess and to check whether $x \notin A^+B^+$ or, if $x = uv$, $u \in A^+$, $v \in B^+$, but $n \neq |f(u)|$. Thus, we may assume that the input is of the form (x, n) where $x = u_1au_2f(u_1)v_av$ with $a \in A, u_1, u_2 \in A^*$, $f(u_1)v_av \in B^+$, $f(a) \neq v_a$, but $n = |f(u_1au_2)|$, $|v_a| = |f(a)|$, and $|v| = |f(u_2)|$. (We put $f(u_i) = 1$, if $u_i = 1, i = 1, 2$.) The automaton reads non-deterministically u_1 on the first tape and in parallel it scans $|f(u_1)|$ fields on the second tape. The automaton remembers the letter a and proceeds non-deterministically, on the first tape only, to the position of the word v_a. It checks that, indeed, $v_a \neq f(a)$. It now accepts, if the rest of the input satisfies $|v| = n - |v_a| - |f(u_1)|$. Hence, $\overline{W(f)}$ and $\overline{W(g)}$ are rational languages.

The following statements are equivalent.

i) The instance of the PCP has no solution, i.e., $f(w) \neq g(w)$ for all $w \in A^+$.
ii) $W(f) \cap W(g) = \emptyset$
iii) $\overline{W(f)} \cup \overline{W(g)} = \mathbb{M}(\Sigma, I)$
iv) $W(f) \cap W(g)$ is finite.
v) $\overline{W(f)} \cup \overline{W(g)}$ is recognizable.
vi) $W(f) \cap W(g)$ is rational.

The implications $i) \Rightarrow ii) \Rightarrow iii) \Rightarrow iv) \Rightarrow v) \Rightarrow vi)$ are trivial. We show $vi) \Rightarrow i)$. Assume by contradiction that $f(w) = g(w)$ for some $w \in A^+$. The trace language $w^+(f(w))^+ \times \mathbb{N}$ is recognizable. Now, if $W(f) \cap W(g)$ is rational, then

$$W(f) \cap W(g) \cap (w^+(f(w))^+ \times \mathbb{N}) = \{(w^n(f(w))^n, n \cdot |f(w)|) \mid n \geq 1\}$$

is again rational, since it is the intersection of a rational with a recognizable language. However, the projection to the first component yields the non-rational language

$$\{w^n(f(w))^n \mid n \geq 1\} \subseteq A^+B^+ \subseteq C^*.$$

Contradiction! So far, we have dealt with the monoid $\mathbb{M}(\Sigma, I) = C^* \times \mathbb{N}$. Encoding C^* into $\{a, c\}^*$, we see that the results holds for $\mathbb{M} = \{a, c\}^* \times b^*$ as well.

This shows that the problems UNI(Σ, I) by $iii)$, REC(Σ, I) by $v)$, and COM(Σ, I) by $vi)$ (by $iv)$ resp.) are undecidable, as soon as I is not transitive. A fortiori, (by the undecidability of UNI(Σ, I)) the problems EQU(Σ, I) and INC(Σ, I) are undecidable in this case.

5.3 Bibliographical Remarks

The presentation of Sect. 5 has been inspired by the work of Bertoni, Goldwurm, Mauri, and Sabadini [34, Chapt. 5] and by Sakarovitch [86]. The results of the present section can be found there. Prop. 61 is originally stated in [8]. Thms. 62 and 63 are from [2, 8, 85]. The problems of Sect. 5.2 have been considered by Gibbons and Rytter [53] thereby showing that, up to $\mathrm{INT}(\Sigma, I)$, the other problems become undecidable for the dependence alphabet $(\Sigma, D) = a - b - c$. The decidability of $\mathrm{REC}(\Sigma, I)$ for a transitive independence relation is due to Sakarovitch [86]. The characterization stated in Thm. 65 has been obtained by Aalbersberg and Hoogeboom [1].

6 Dependence Graphs and Logic

6.1 Dependence Graphs

So far, a trace has been defined as a congruence class of a word modulo a partial commutation. A trace has also a unique representation as a labeled, directed, and acyclic graph, defining therefore a labeled partial order or a *pomset,* (*p*artially *o*rdered *m*ulti*set*). We start with an abstract definition. Let (Σ, D) be a dependence alphabet. A *dependence graph* is (an isomorphism class of) a node-labeled acyclic graph $[V, E, \lambda]$, where

- V is an at most countable set of vertices,
- $E \subseteq V \times V$ is the edge (or arc) relation such that the directed graph (V, E) is acyclic and the induced partial order is well-founded.
- $\lambda : V \to \Sigma$ is the node-labeling such that $(\lambda(x), \lambda(y)) \in D$ if and only if $(x, y) \in E \cup E^{-1} \cup \mathrm{id}_V$.

Remark. A partial order is called well-founded, if every non-empty set has minimal elements. The assumption of well-foundedness will become crucial in Prop. 68 below. However, as long as we deal with finite traces only, well-foundedness has no significance. (Hence we can forget about it.) We are more general here, since we want a basis which allows to include a theory of infinite traces, c.f. Sect. 8.

The set of dependence graphs is denoted by $\mathbb{G}(\Sigma, D)$. It is a monoid, the empty graph $1 = [\emptyset, \emptyset.\emptyset]$ is the neutral element and the concatenation of the dependence graphs $[V_1, E_1, \lambda_1]$ and $[V_2, E_2, \lambda_2]$ is defined as follows. First, we take the disjoint union of labeled acyclic graphs, and then we introduce additionally arcs from V_1 to V_2 between all nodes with dependent labels. Formally

$$\begin{aligned}
&[V_1, E_1, \lambda_1] \cdot [V_2, E_2, \lambda_2] \quad = \\
&[V_1 \dot\cup V_2, E_1 \dot\cup E_2 \dot\cup \{(x, y) \in V_1 \times V_2 \mid (\lambda_1(x), \lambda_2(y)) \in D\}, \lambda_1 \dot\cup \lambda_2].
\end{aligned}$$

Remark. Let α be any countable ordinal, e.g. $\alpha = \omega, \alpha = \omega + 1$ or $\alpha = \omega^\omega$. Consider any subset $L \subseteq \mathbb{G}(\Sigma, D)$ and let $g : \alpha \longrightarrow L$ be an arbitrary mapping.

Then we can easily define the ordered product

$$mathop\overrightarrow{\prod}_{i\in\alpha} g(i) \in \mathbb{G}(\Sigma, D)$$

as follows. First, we take the disjoint union of the labeled graphs $g(i)$ over all $i \in \alpha$. Then we introduce additional arcs from a vertex x of the graph $g(i)$ to a vertex y of $g(j)$ for all $i < j$ and $(\lambda_i(x), \lambda_j(y)) \in D$.

We define

$$L^\alpha = \{mathop\overrightarrow{\prod}_{i\in\alpha} g(i) \mid g : \alpha \longrightarrow L \text{ is a mapping } \}.$$

In particular, for $L \subseteq \mathbb{G}(\Sigma, D)$ the ω-product $L^\omega \subseteq \mathbb{G}(\Sigma, D)$ is defined. According to this definition we have the following situation for subsets containing the empty graph:

$$L^\alpha = \bigcup_{\beta\le\alpha}(L \setminus \{1\})^\beta \text{ , if } 1 \in L.$$

In particular:

$$L^\omega = L^* \cup (L \setminus \{1\})^\omega \text{ , if } 1 \in L.$$

Finite dependence graphs form a submonoid of $\mathbb{G}(\Sigma, D)$. For a dependence graph $t = [V, E, \lambda]$ and a letter $a \in \Sigma$ let $V_a = \{x \in V \mid \lambda(x) = a\}$. This is a linearly ordered set, hence a well-order and therefore a countable ordinal. This ordinal is denoted by $|t|_a$ and gives the a-length of t. It leads to a standard representation where $V = \{(a, i+1) \mid a \in \Sigma, 0 \le i < |t|_a\}$. The intended semantics is that $(a, i+1)$ denotes the $(i+1)$-st node of V having label a. (The notation $i + 1$ is used to exclude limit ordinals, it is of no importance for the finite case.) We have $\lambda(a, i) = a$, and $((a, i), (b, j)) \in E$ if and only if $(a, b) \in D$ and $i < j$.

Proposition 68. *The monoid $\mathbb{G}(\Sigma, D)$ of dependence graphs is left-cancellative. Its submonoid of finite graphs is cancellative.*

Proof. The result for finite dependence graphs is easily seen from the standard representation. For the general case one may use ordinal arithmetic.

A letter $a \in \Sigma$ is given as a one-point graph, labeled with a. Let us denote this graph by $\varphi_{\mathbb{G}}(a)$. Then $\varphi_{\mathbb{G}}$ can be extended to a homomorphism

$$\varphi_{\mathbb{G}} : \Sigma^* \to \mathbb{G}(\Sigma, D).$$

Moreover: $\varphi_{\mathbb{G}}(ab) = \varphi_{\mathbb{G}}(ba)$ for $(a, b) \in I$, since $\varphi_{\mathbb{G}}(ab)$ consists of two nodes, one labeled with a the other one labeled with b, and the graph has no edges. Thus, $\varphi_{\mathbb{G}}$ factorizes and we obtain a homomorphism

$$\overline{\varphi_{\mathbb{G}}} : \mathbb{M}(\Sigma, I) \to \mathbb{G}(\Sigma, D).$$

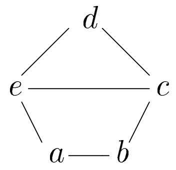
For $t = a_1 \cdots a_n \in \mathbb{M}(\Sigma, I)$, $a_i \in \Sigma$ for all $1 \le i \le n$ we have the following explicit description of $\overline{\varphi_{\mathbb{G}}}(t)$. We take any set of n nodes, say $V = \{1, \ldots, n\}$, then we label node i with a_i and we put $(i, j) \in E$ if and only if $(a_i, a_j) \in D$ and $i < j$.

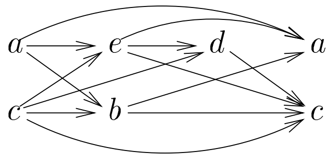
Proposition 69. *The monoid of traces $\mathbb{M}(\Sigma, I)$ can be identified with the submonoid of finite graphs of $\mathbb{G}(\Sigma, D)$. The homomorphism $\varphi_{\mathbb{G}}$ above then becomes the canonical homomorphism $\varphi : \Sigma^* \to \mathbb{M}(\Sigma, I)$.*

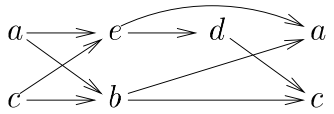
Proof. Let $t = [V, E, \lambda]$ be a finite dependence graph. Choosing a linear order of V which refines the partial order (V, E^*), we may assume that $V = \{1, \ldots, n\}$ with $(i, j) \in E$ only if $i < j$. An easy reflection yields $\varphi_{\mathbb{G}}(\lambda(1) \cdots \lambda(n)) = t$, hence $\overline{\varphi_{\mathbb{G}}}$ is onto. To see that $\overline{\varphi_{\mathbb{G}}}$ is injective, let $\overline{\varphi_{\mathbb{G}}}(u) = \overline{\varphi_{\mathbb{G}}}(v) = [V, E, \lambda]$ and $(a, b) \in D$. Consider the set $V_{a,b} = \{x \in V \mid \lambda(x) \in \{a, b\}\}$. Since $(a, b) \in D$, this is a labeled linear order, which can be identified with a word of $\{a, b\}^*$. From the very definition of $\varphi_{\mathbb{G}}$ this word is equal to $\pi_{a,b}(u)$ (where $\pi_{a,b}(u)$ denotes the projection of u to $\{a, b\}^*$). Hence $\pi_{a,b}(u) = \pi_{a,b}(v)$ for all $(a, b) \in D$. Therefore $u = v$ by Prop. 1.

For the rest of this section we deal with finite traces only and we think of (Σ, D) as being fixed. In particular the size of (Σ, D) is viewed as a constant. As we have mentioned above, it is sometimes convenient to identify a trace t (or its dependence graph $[V, E, \lambda]$) with its induced labeled partial order $[V, E^*, \lambda]$. (Following standard notations E^* means the reflexive, transitive closure of E.) However, if t is of length n then both $[V, E, \lambda]$ and $[V, E^*, \lambda]$ are representations of t of size $\Theta(n^2)$, since the dependence graph of a^n has $\frac{n(n-1)}{2}$ edges. Hence, often we content to represent t by its Hasse diagram $[V, H, \lambda]$. In the Hasse diagram all redundant edges have been removed. Thus, $H \subseteq E$ is the smallest subset such that $H^* = E^*$. In the Hasse diagram the in- and out-degree of every node is bounded by $|\Sigma|$. Therefore it is a representation which is linear in its length. For $|t| = n$ the Hasse diagram has size $\Theta(n)$.

Example 10. Let

$(\Sigma, D) =$  and $t = [acebdac]$

dependence graph 

Hasse diagram 

The following algorithm computes the Hasse diagram in time $\mathcal{O}(n)$. The input is a trace t given as a string $a_1 \cdots a_n$, $a_i \in \Sigma$ for $1 \leq i \leq n$. It uses a set *MIN* which represents the actual minimal elements of the induced labeled partial order. Since we focus on *MIN*, the computation is done from right to left.

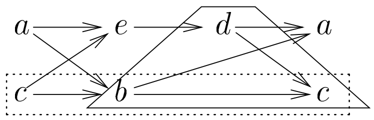
```
function HASSE(a_1 ··· a_n)
    var H : labeled acyclic digraph := [∅, ∅, ∅]
        MIN : subset of labeled vertices := ∅

    for i = n downto 1 do
        create a new node v(i) with label a_i
        add node v(i) to the node list of t and to MIN
        for all x ∈ MIN, x ≠ v(i) do
            if (a_i, label(x)) ∈ D then connect v(i) with x
                                        add the new edge to H
                                        remove x from MIN
            end_if
        end_forall
    end_for
    return H
end_function
```

At the end of the computation, *MIN* is the first step of the Foata normal form of the trace. In fact, the task of computing the Foata- or the lexicographic normal form can easily be performed in linear time by HASSE.

The Hasse diagram allows a nice visualization of factors. We say that u is a *factor* of t, if we can write $t = puq$ for some $p, q \in \mathbb{M}(\Sigma, I)$. Given such a factorization $t = puq$ and a representation of t by its Hasse diagram $[V, H, \lambda]$ we can identify p, u and q as a partition of $V = P \dot\cup U \dot\cup Q$. We say that $U \subseteq V$ corresponds to some factor in this situation.

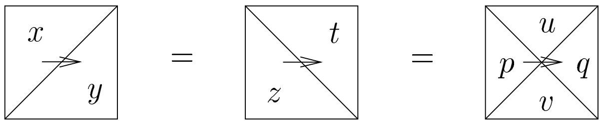
Proposition 70. *Let $[V, H, \lambda]$ ($[V, E, \lambda]$ resp.) the Hasse diagram (the dependence graph, resp.) of a trace and $U \subseteq V$. Then U corresponds to some factor if and only if for all $x, y \in U$ every directed path from x to y is entirely contained in $[U, H \cap (U \times U)]$ ($[U, E \cap (U \times U)]$ resp.).*

Example 11. Let (Σ, D) as in the previous example.

$t \quad =$ 

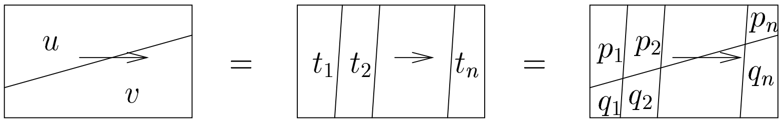
The polygon *bdc* is a factor of t, the dotted rectangle *cbc* is not.

Another advantage of the graph interpretation of traces is the existence of "visual proofs". Here is such a proof for Levi's Lemma, Prop. 4.



The independence of u and v is clear since there is neither an arc from y to x nor from t to z.

The proof of Cor. 5 is no more difficult:



There are no arcs between p_j and q_i for $i < j$.

6.2 Traces and Logic

Some classical results about first- and monadic second-order logic are extendible from words to traces. Monadic second-order formulae are built up upon first-order variables $x, y, \ldots$ ranging over elements of V and second-order variables $X, Y, \ldots$ ranging over subsets of V. There are four types of atomic formulae:

$$x \in X,\ x = y,\ (x, y) \in E, \text{ and } \lambda(x) = a \text{ for } a \in \Sigma.$$

We also allow Boolean constants *true* and *false*, the logical connectives $\vee$, $\wedge$, $\neg$, and quantification $\exists$, $\forall$ for the first- and second-order variables. A first-order formula is a formula without any second-order variable. A *sentence* is a closed formula, i.e., a formula without free variables. Identifying a trace $t \in \mathbb{M}(\Sigma, I)$ with its dependence graph $t = [V, E, \lambda]$, every sentence Ψ has an obvious interpretation over t. Thus, the truth value of $t \models \Psi$ is well-defined. The trace language defined by a sentence Ψ is $L(\Psi) = \{t \in \mathbb{M}(\Sigma, I) \mid t \models \Psi\}$. We can speak therefore of first-order and second-order definability of trace languages.

Example 12. Let $b \in \Sigma$ be a letter. The trace (or word) language defined by an even number of b is definable in monadic second-order logic. We divide the set of positions where a letter b occurs in two disjoint sets X and Y. Then we say that between any two different positions of X there is at least one position of Y, and vice versa.

Example 13. Let Σ have a linear ordering and let $LexNF \subseteq \Sigma^*$ be the set of words which are lexicographic normal forms of traces from $\mathbb{M}(\Sigma, I)$. Then $LexNF$ is definable in first-order. In fact, by Thm. 7 a word is *not* in $LexNF$ if and only if there are positions i, k such that $i \leq k$, $\lambda(i) > \lambda(k)$, and $(\lambda(j), \lambda(k)) \in I$ for all $i \leq j < k$.

The predicate $x = y$, belonging to our syntax, is redundant as long as we work with dependence graphs. It could be viewed as an abbreviation of $\lambda(x) = \lambda(y) \wedge \neg(x, y) \in E \wedge \neg(y, x) \in E$, and $\lambda(x) = \lambda(y)$ is the formula $\bigvee_{a \in \Sigma}(\lambda(x) = a \wedge \lambda(y) = a)$. In general, transitive closure cannot be expressed in first-order. Due to the structure of dependence graphs we can do it here. The formula $(x, y) \in E^+$

(meaning that there is a non-empty path from x to y) can be written in first-order with the help of at most $|\Sigma| - 2$ additional variables. In fact, $(x, y) \in E^+$ is equivalent with

$$(x,y) \in E \vee \bigvee_{k \leq |\Sigma|-2} (\exists z_1 \cdots \exists z_k : (x, z_1) \in E \wedge \bigwedge_{1<i\leq k} (z_{i-1}, z_i) \in E \wedge (z_k, y) \in E)$$

There is also a first-order expression for the edge relation in the Hasse diagram: $(x, y) \in H$ is equivalent with

$$(x,y) \in E \wedge \neg(\exists z : (x,z) \in E^+ \wedge (z,y) \in E^+)$$

In the word case we have $E = E^+$ and we prefer to write $x < y$ instead of $(x, y) \in E$, where $<$ refers to the total order on positions in words.

Theorem 71. *Let $\varphi : \Sigma^* \to \mathbb{M}(\Sigma, I)$ be the canonical homomorphism and $T \subseteq \mathbb{M}(\Sigma, I)$ be a trace language.*

i) Then T is definable in monadic second-order logic (first-order resp.) if and only if $\varphi^{-1}(T)$ has this property.
ii) Let LexNF $\subseteq \Sigma^*$ denote the set of lexicographic normal forms and $K \subseteq$ LexNF. Then K is definable in monadic second-order logic (first-order resp.) if and only if $\varphi(K)$ has this property.

Proof. i): First, let $T = L(\Psi) \subseteq \mathbb{M}(\Sigma, I)$ for some sentence Ψ. Let $t = [V, E, \lambda]$ be any trace and $w \in \Sigma^*$ a representing word, i.e., $\varphi(w) = t$. There is a bijective one-to-one correspondence between positions of w and nodes of V, and we have $(x, y) \in E$ if and only if $x < y$ and $(\lambda(x), \lambda(y)) \in D$ in the word w. Thus, replacing every atomic formula $(x, y) \in E$ by the first-order conjunction

$$x < y \wedge (\lambda(x), \lambda(y)) \in D,$$

we obtain a sentence $\tilde{\Psi}$ such that $t \models \tilde{\Psi}$ is equivalent with $w \models \Psi$. Hence $\varphi^{-1}(T) = L(\tilde{\Psi})$ For the converse let $K = \varphi^{-1}(T) \cap$ *LexNF*. Since $T = \varphi(K)$ and *LexNF* is a first-order language, it is enough to prove the second part of the theorem.

ii): Let $t = [V, E, \lambda]$ be a trace and let $\hat{t} = x_1 \cdots x_n \in$ *LexNF* be its representation in lexicographic normal form. Every node $x \in V$ corresponds to some x_i in $\hat{t}$ and vice versa. We write $\mathrm{lex}(x, y)$, if x corresponds to x_i, y corresponds to x_j, and if we have $i < j$. Thus, $\mathrm{lex}(x, y)$ means that the node x is before y in its lexicographic normal form. If this happens, then either $(x, y) \in E^+$ or there is some minimal index k such that $i < k \leq j$, where x_k corresponds to some node $z \in V$ such that $(z, y) \in E^*$. Moreover, since it is the lexicographic normal form we have $\lambda(x) < \lambda(z)$ with respect to the ordering of the alphabet.

Based on this observation we obtain the following relation:

$$\begin{aligned}\mathrm{lex}(x,y) = {} & (x,y) \in E^+ \text{ or} \\ & \lambda(x) < \lambda(y) \text{ and } \neg\mathrm{lex}(y,x) \text{ or} \\ & \lambda(y) < \lambda(x) \text{ and } \exists z : \lambda(x) < \lambda(z) \wedge \mathrm{lex}(x,z) \wedge (z,y) \in E^*.\end{aligned}$$

Since the alphabet is finite, it is enough to unfold the recursion $(2 \cdot |\Sigma| - 1)$ times and the unfolded formula is a first-order formula (of exponential size in $|\Sigma|$) over traces. Now, let $K \subseteq \mathit{LexNF}$ be defined by a sentence Ψ. In Ψ we will use the symbol $\mathrm{lex}(x, y)$ as the atomic formula to denote the ordering between positions. We then replace $\mathrm{lex}(x, y)$ by the first-order formula above. We obtain a formula $\tilde{\Psi}$ over traces. By construction, we have for all $w \in \mathit{LexNF}$ that $w \models \Psi$ holds if and only if $\varphi(w) \models \tilde{\Psi}$. The result follows.

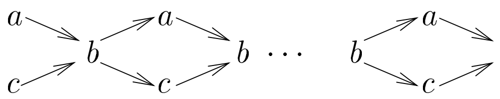
By classical results on words, definability in monadic second-order logic is equivalent with recognizability. The first-order counterpart of words generalizes directly to traces, too.

Corollary 72. *Let $T \subseteq \mathbb{M}(\Sigma, I)$ be a trace language. Then the following assertions are equivalent.*

i) T is recognizable by some finite (finite and aperiodic, resp.) monoid.
ii) T is recognizable (star-free, resp.).
iii) T is definable in monadic second-order (first-order, resp.) logic.

Proof. Due to Thm. 71 and well-known results about words, it is enough to show $ii) \Leftrightarrow iii)$ for the first-order assertion, only. Both directions are obtained analogously to the word case: The direction (first-order) $ii) \Rightarrow iii)$ is more simple and we sketch the main idea for the concatenation. Let $t = [V, E, \lambda]$ be a trace and $t = uw$ a factorization. Then u, w correspond to subsets $U, W \subseteq V$. In fact, we find a finite set $X = \{x_1, \ldots, x_k\}$, $0 \le k \le |\Sigma|$, such that $U = \{y \in V \mid \exists x_i \in X : (y, x_i) \in E^*\}$ and $W = \{z \in V \mid \forall x_i \in X : (x_i, z) \notin E^*\}$. By standard techniques, see [88], we can construct for given two first-order sentences Ψ_1, Ψ_2 a new first-order sentence specifying those traces such that there exists a finite set X as above and the dependence graph restricted to U satisfies Ψ_1, whereas the dependence graph restricted to W satisfies Ψ_2. The implication $iii) \Rightarrow ii)$ follows from Thm. 78 below giving a more precise statement.

Example 14. Let $(\Sigma, D) = a - b - c$ and $T \subseteq \mathbb{M}(\Sigma, I)$ the trace language $T = (acbcab)^+$. Since $ac = ca$, this is the set of traces



with an even number of b. By the method of the first example of this section, this trace language is definable in monadic second-order, but it is not first-order-definable. On the other hand, the word language $T = (acbcab)^+ \subseteq \{a, b, c\}^*$ has a first-order description. It is the set of words starting with $acbcab$ and every occurrence of the factor ab is either the end or the word followed by another factor $acbcab$.

Remark. It is known by [76] that the commutative closure of a star-free word language is either still star-free or not recognizable anymore. The example above shows that for $(\Sigma, D) = a - b - c$ the image of the word language $(acbcab)^+$ in $\mathbb{M}(\Sigma, I)$ is recognizable, but not star-free. This means in particular that the result about the commutative closure of star-free languages can not be generalized to arbitrary trace monoids $\mathbb{M}(\Sigma, I)$. There is a precise characterization [76]: For all star-free word languages it holds that the I-closure is either star-free or not recognizable if and only if the dependence relation D is transitive.

6.3 Ehrenfeucht-Fraïssé-Games

Let $t = [V(t), E(t), \lambda(t)]$ be any node-labeled graph. Then $V(t)$ denotes the set of vertices, $E(t) \subseteq V(t) \times V(t)$ is the edge relation, and $\lambda_t : V(t) \to \Sigma$ is the node-labeling. For the moment it is not necessary to put any further restrictions on $E(t)$. Thus, the graphs may be finite or infinite, directed or undirected, cyclic or acyclic, etc. By $\overline{a} = (a_1, \ldots, a_k)$ we denote a k-tuple of nodes, $\overline{a} \in V^k$, $k \geq 0$. If $\overline{x} = (x_1, \ldots, x_k)$, then $\Psi(\overline{x})$ denotes a first-order formula with free variables $x_1, \ldots, x_k$. Any pair $(t, \overline{a})$ yields a natural interpretation for $\Psi(\overline{x})$. The interpretation of x_i is a_i for $1 \leq i \leq k$. Thus, the expression $(t, \overline{a}) \models \Psi(\overline{x})$ has a well-defined truth value.

An Ehrenfeucht-Fraïssé-game has two players, Player I and Player II. The players take two structures $(s, \overline{a})$ and $(t, \overline{b})$ as above: $s = [V(s), E(s), \lambda_s]$, $\overline{a} \in V(s)^k$, $t = [V(t), E(t), \lambda_t]$, and $\overline{b} \in V(t)^k$. They first decide on the number of rounds m with $m \geq 0$.

If $m = 0$, we say that Player II wins the game, if $\lambda_s(a_i) = \lambda_t(b_i)$ and $(a_i, a_j) \in E(s) \Leftrightarrow (b_i, b_j) \in E(t)$ for all $1 \leq i, j \leq k$. If $m \geq 1$, then Player I begins and takes either a vertex $a_{k+1} \in V(s)$ or a vertex $b_{k+1} \in V(t)$. Player II answers by taking either some $b_{k+1} \in V(t)$ (if Player I has chosen in the graph s), or by taking some $a_{k+1} \in V(s)$ (otherwise). This finishes the first round. The game is continued with $(m-1)$ rounds over the new structures $(s, (a_1, \ldots, a_k, a_{k+1}))$ and $(t, (b_1, \ldots, b_k, b_{k+1}))$.

The relation $\simeq_{(m,k)}$ is defined by setting $(s, \overline{a}) \simeq_{(m,k)} (t, \overline{b})$, if Player II has a winning strategy for a game on m rounds. The following lemmata are well-known facts in logic.

Lemma 73. *The relation $\simeq_{(m,k)}$ is an equivalence relation of finite index.*

The *quantifier depth* of a formula is defined inductively. For atomic formulae it is zero, the use of the logical connectives does not change it, and adding a quantifier in front increase the quantifier depth by one. For example, the following formula has quantifier depth two:

$$\forall x((\exists y(\lambda(x) \neq \lambda(y))) \vee (\exists z(\lambda(z) = b \wedge ((x, z) \in E \vee \lambda(x) = a))))$$

Lemma 74. *Let $\Psi(\overline{x})$ be a first-order formula with free variables $x_1, \ldots, x_k$ and $(s, \overline{a}) \models \Psi(\overline{x})$. If $\Psi(\overline{x})$ has quantifier depth m and $(s, \overline{a}) \simeq_{(m,k)} (t, \overline{b})$, then we have $(t, \overline{b}) \models \Psi(\overline{x})$, too.*

Let us denote by $[s, \overline{a}; m, k]$ the equivalence class $\{(t, \overline{b}) \mid (s, \overline{a}) \simeq_{(m,k)} (t, \overline{b})\}$.

Lemma 75. *Each equivalence class $[s, \overline{a}; m, k]$ can be specified by a first-order formula $\Psi[s, \overline{a}; m, k](\overline{x})$ of quantifier depth m and with free variables $x_1, \ldots, x_k$.*

Proof. For $m = 0$ this is clear. Consider $m + 1$ and assume inductively that formulae for all classes $[s, (\overline{a}, a_{k+1}); m, k + 1]$ are already specified. The next formula satisfies the requirement, it defines $\Psi[s, \overline{a}; m + 1, k](\overline{x})$:

$$\bigwedge_{a_{k+1}} (\exists x_{k+1} \Psi[s, (\overline{a}, a_{k+1}); m, k](\overline{x}, x_{k+1})) \wedge$$
$$\forall x_{k+1} (\bigvee_{a_{k+1}} \Psi[s, (\overline{a}, a_{k+1}); m, k](\overline{x}, x_{k+1})).$$

Observe that the conjunction $\bigwedge_{a_{k+1}}$ and the disjunction $\bigvee_{a_{k+1}}$ can be made finite by Lem. 73. Details are left to the reader.

Next, we need a concatenation on node-labeled graphs $s = [V(s), E(s), \lambda_s]$. We adopt the definition for dependence graphs. Thus, we define $s \cdot t$ by

$$[V(s) \,\dot\cup\, V(t), E(s) \,\dot\cup\, E(t) \,\dot\cup \{(x, y) \in V(s) \times V(t) \mid (\lambda_s(x), \lambda_t(y)) \in D\}, \lambda_s \,\dot\cup\, \lambda_t].$$

Having this, we can state the following *Congruence Lemma*. Its proof is clear, since due to the definition of concatenation, whether or not $(x, y) \in E(s \cdot t)$ for $x \in V(s), y \in V(t)$ depends on the labels of x and y, only.

Lemma 76. *Let u be any fixed node-labeled graph and $\{u_1, \ldots, u_n\}$ its vertex set. Let $(s, \overline{a}) \simeq_{(m,k_1)} (s', \overline{a}')$ and $(t, \overline{b}) \simeq_{(m,k_2)} (t', \overline{b}')$.*
Then it holds:

$$(s \cdot u \cdot t, (\overline{a}, u_1, \ldots, u_n, \overline{b})) \simeq_{(m,k_1+n+k_2)} (s' \cdot u \cdot t', (\overline{a}', u_1, \ldots, u_n, \overline{b}')).$$

In order to close the bridge to the star-free sets, we need a finitely generated monoid. For convenience we restrict our attention therefore again to finite dependence graphs, only. It is a nice coincidence that we have the following fact.

Proposition 77. *The set of traces (i.e., the set of finite dependence graphs) can be specified by a first-order sentence.*

Proof. The first requirement for dependence graphs is a first-order statement

$$\forall x \forall y : (\lambda(x), \lambda(y)) \in D \Leftrightarrow (x, y) \in E \vee (y, x) \in E \vee x = y.$$

The second requirement asks for acyclic graphs. Because of the first formula, it is enough to exclude cycles of short length; more precisely, we do not allow cycles of length less or equal $|\Sigma| + 1$. Short cycles however can be specified in first-order.

A *dot-depth hierarchy* for words can also be defined for traces. The empty set $\emptyset$ and the set of all traces $\mathbb{M}(\Sigma, I)$ is of dot-depth zero. To obtain level k of the dot-depth hierarchy, $k \geq 1$, one forms the Boolean closure of the languages $L \cdot a \cdot K$, where $a \in \Sigma$ and L, K are of level $k - 1$.

Theorem 78. *Let Ψ be a first-order sentence of quantifier depth m and $L(\Psi) = \{t \in \mathbb{M}(\Sigma, I) \mid t \models \Psi\}$. Then $L(\Psi)$ is in the m-th level of the dot-depth hierarchy, and the corresponding star-free expression for $L(\Psi)$ can effectively been constructed.*

Proof. We sketch the proof, only. For $m = 0$ the result is true. By induction, we assume the result to be correct for m and that the sentence is in fact of the form $\exists x \Psi'(x)$, where the formula $\Psi'(x)$ has quantifier m. Consider the following union

$$T = \bigcup \{U \cdot a \cdot V \mid a \in \Sigma,\ U = [u, (); m, 0],\ V = [v, (); m, 0], \text{ and } uav \models \Psi\}.$$

We now make several observations. The union is finite by Lem. 73. By induction, it is a star-free expression of dot-depth $m + 1$. Since $u \in U$, $v \in V$, we have $uav \in T$, hence $L(\Psi) \subseteq T$. The converse follows by known techniques by the Congruence Lemma (Lem. 76), Lem. 75, and Lem. 74.

6.4 Bibliographical Remarks

Dependence graphs for traces were already used by Mazurkiewicz in [68]. In Anisimov and Knuth we find the description as labeled partial orders [4]. The characterization of recognizable trace languages by monadic second-order logic over traces is due to Thomas [89]. Our proof follows Ebinger [41], see also [34, Chapt.10]. This allows to include the first-order statements in Thm. 71. As noted earlier, Cor. 46 is a simple direct consequence. Independently Courcelle [22] gave another proof for Cor. 46 using monadic second-order logic over traces. The equivalence of *i)* and *ii)* of Cor. 72 has been shown by Guaiana, Restivo, and Salemi [58]. The equivalence to *iii)* is from [42]. Thm. 71 and Cor. 72 have a generalization to concurrency monoids by Droste and Kuske [37, 39]. Our proof technique using Ehrenfeucht-Fraïssé-games follows [88]. It led to Thm. 78 being a partial analogue of a result of Thomas [87] on words.

7 Asynchronous Automata

7.1 Zielonka's Theorem

An *asynchronous automaton* $\mathcal{A}$ has a distributed finite state control such that independent actions may be performed in parallel. The set of global states is modeled as a Cartesian product $Q = \prod_{i \in J} Q_i$, where the Q_i are states of the local component $i \in J$ and J is some index set. With each letter $a \in \Sigma$ we associate a *read domain* $R(a) \subseteq J$ and a *write domain* $W(a) \subseteq J$. We henceforth assume that $W(a) \subseteq R(a)$ being a rather natural (but technically important) restriction. The transitions are given by a family of partially defined functions (throughout we deal for simplicity with deterministic automata, only):

$$\left(\delta_a : \prod_{i \in R(a)} Q_i \longrightarrow \prod_{i \in W(a)} Q_i\right)_{a \in \Sigma}$$

Thus, each a reads the status in the local components of its read domain and changes states in local components of its write domain. Accordingly to the read-and-write-conflicts being allowed, we distinguish four basic types:

- Concurrent-Read-Exclusive-Write (*CREW*), if $R(a) \cap W(b) = \emptyset$ for all $(a, b) \in I$.
- Concurrent-Read-Owner-Write (*CROW*), if $R(a) \cap W(b) = \emptyset$ for all $(a, b) \in I$ and $W(a) \cap W(b) = \emptyset$ for all $a \neq b$.
- Exclusive-Read-Exclusive-Write (*EREW*), if $R(a) \cap R(b) = \emptyset$ for all $(a, b) \in I$.
- Exclusive-Read-Owner-Write (*EROW*), if $R(a) \cap R(b) = \emptyset$ for all $(a, b) \in I$ and $W(a) \cap W(b) = \emptyset$ for all $a \neq b$.

The local transition functions $(\delta_a)_{a \in \Sigma}$ give rise to a partially defined transition function on global states

$$\delta : (\prod_{i \in J} Q_i) \times \Sigma \longrightarrow \prod_{i \in J} Q_i$$

where $\delta\left((q_i)_{i \in J}, a\right) = (q_i')_{i \in J}$ is defined if and only if $\delta_a\left((q_i)_{i \in R(a)}\right)$ is defined. In this case we have $q_j' = \left(\delta_a\left((q_i)_{i \in R(a)}\right)\right)_j$ for $j \in W(a)$ and $q_j = q_j'$ otherwise. If $\mathcal{A}$ is of any of the four types above, then it is clear that we can define the action of a trace $t \in \mathbb{M}(\Sigma, I)$ on global states. It is convenient to denote this function again by δ. Therefore, we may view it as a partially defined function $\delta : Q \times \mathbb{M}(\Sigma, I) \to Q$. Given an initial state $q_0 \in Q$ and a set $F \subseteq Q$, we obtain an $\mathbb{M}(\Sigma, I)$-automaton. The automaton $\mathcal{A} = \left(\prod_{i \in J} Q_i, (\delta_a)_{a \in \Sigma}, q_0, F\right)$ accepts the language $L(\mathcal{A}) = \{t \in \mathbb{M}(\Sigma, I) \mid \delta(q_0, t) \in F\}$. If Q is finite, then $L(\mathcal{A})$ is a recognizable trace language. As well-known in automata theory, we may add local dead states in order to have a totally defined transition function.

Before we continue we need some more notations. Let $t \in \mathbb{M}(\Sigma, I)$ be a trace and $a \in \Sigma$ be a letter. By $\partial_a(t)$ we denote the smallest prefix of t which contains all occurrences a of t. Thus, $\partial_a(t)$ is the trace of minimal length satisfying $|\partial_a(t)|_a = |t|_a$ such that $t = \partial_a(t)q$ for some $q \in \mathbb{M}(\Sigma, I)$. Viewing t as a labeled partial order, $\partial_a(t)$ contains all vertices with label a and the past of them. This notion is extended to subsets $A \subseteq \Sigma$. The trace $\partial_A(t)$ is the minimal prefix of t satisfying $|\partial_A(t)|_a = |t|_a$ for all $a \in A$. Note that $\partial_\emptyset(t) = 1$, $\partial_\Sigma(t) = t$, and $\partial_a(t)$ is a prefix of $\partial_A(t)$ for all $a \in A$. Moreover, viewing t as labeled partial order, we find that $\partial_A(t)$ is the union of the $\partial_a(t)$, $a \in A$.

Definition 79. A mapping $\alpha : \mathbb{M}(\Sigma, I) \to Q$ to some finite set Q is called *asynchronous,* if the following two conditions are satisfied.

i) The value of $\alpha(\partial_{A \cup B}(t))$ is computable from $A \subseteq \Sigma$, $B \subseteq \Sigma$, and the values $\alpha(\partial_A(t))$ and $\alpha(\partial_B(t))$.
ii) If $t = \partial_a(t)$, then the value of $\alpha(ta)$ is computable from $a \in \Sigma$ and the value $\alpha(t)$.

A trace language $T \subseteq \mathbb{M}(\Sigma, I)$ is *recognized by an asynchronous mapping* α, if $T = \alpha^{-1}\alpha(T)$.

The following theorem is far from being trivial. It is crucial for the general construction of asynchronous automata.

Theorem 80. *A trace language is recognizable if and only if it is recognized by some asynchronous mapping.*

We cannot go into details of the difficult only-if-part of the theorem. Once this has been shown, then the proof of the if-part (c.f. Prop. 83) yields *Zielonka's Theorem* as a corollary.

Corollary 81. *A trace language is recognizable if and only if it is recognized by some asynchronous automaton.*

The next sections give explicit transformations between the different types of asynchronous automata. Therefore we have Zielonka's Theorem for all types considered here.

7.2 Asynchronous Cellular Automata

Definition 82. An asynchronous automaton $\mathcal{A}$ is called *asynchronous cellular,* if the state space Q can be decomposed as $Q = \prod_{a \in \Sigma} Q_a$ such that $W(a) = \{a\}$ and $R(a) = D(a) = \{b \in \Sigma \mid (a, b) \in D\}$ for all $a \in \Sigma$.

Remark. Every *CROW*-type asynchronous automata can be viewed as asynchronous cellular by a trivial transformation (regrouping components) which does not change the number of reachable global states.

The following proposition yields the if-part of Thm. 80.

Proposition 83. *Let* $\alpha : \mathbb{M}(\Sigma, I) \to Q$ *be an asynchronous mapping recognizing a trace language* T*. Then* Q^{Σ} *is the state space of an asynchronous cellular automaton* $\mathcal{A}$ *with* $L(\mathcal{A}) = T$*.*

Proof. The tuple $q_0 = (\alpha(1))_{a \in \Sigma}$ is used as initial state. The global transition function is defined such that

$$\delta(q_0, t) \quad = \quad (\alpha(\partial_b(t)))_{b \in \Sigma}\,.$$

The reason why this works is based on the equation $\partial_a(ta) = \partial_{D(a)}(t)a$, which holds for all $t \in \mathbb{M}(\Sigma, I)$ and $a \in \Sigma$. In fact, consider a tuple $(q_b)_{b \in D(a)}$ with $q_b = \alpha(\partial_b(t))$. Due to the first condition of an asynchronous mapping, we can compute the value of $\alpha(\partial_{D(a)}(t))$. Clearly, $\partial_a(\partial_{D(a)}(t)) = \partial_{D(a)}(t)$, hence by the second condition, we may compute $\alpha(\partial_{D(a)}(t)a)$, which is $\alpha(\partial_a(ta))$. Therefore we can change the local state q_a to the new local state $q'_a = \alpha(\partial_a(ta))$. Finally, observe that for all $b \in \Sigma$, $b \neq a$ we have $\partial_b(ta) = \partial_b(t)$. Hence, the new global state satisfies indeed $\delta(q_0, ta) = (\alpha(\partial_b(ta))_{b \in \Sigma}$. It is enough to define the final states by $F = \{(\alpha(\partial_b(t)))_{b \in \Sigma} \mid t \in T\}$. The result follows.

7.3 Changing Concurrent-Read to Exclusive-Read

The original definition an asynchronous automaton demands an *EREW*-type with $R(a) = W(a)$ for all $a \in \Sigma$. Obviously, *EROW* is even a stronger condition. (Recall our general assumption $W(a) \subseteq R(a)$ for $a \in \Sigma$). Therefore we content with the following proposition.

Proposition 84. *For every asynchronous automaton $\mathcal{A}$ of CROW-type there exists an asynchronous automaton $\mathcal{A}'$ of EROW-type with the same number of reachable global states and recognizing the same language, $L(\mathcal{A}) = L(\mathcal{A}')$.*

Proof. By a remark above we may assume that the state space of $\mathcal{A}$ has the form $Q = \prod_{a\in\Sigma} Q_a$ and $R(a) = D(a) = \{b \in \Sigma \mid (a,b) \in D\}$ for all $a \in \Sigma$. Define $P_{(a,b)} = Q_a$ for all $b \in D(a)$, $a \in \Sigma$ by using different copies. The new set of global states is defined by $P = \prod\limits_{a\in\Sigma} \prod\limits_{b\in D(a)} P_{(a,b)}$. For $a \in \Sigma$ we define the new partially defined transition function.

$$\begin{aligned}\delta'_a : \prod_{b\in D(a)} P_{(b,a)} &\longrightarrow \prod_{c\in D(a)} P_{(a,c)}\\ (q_b)_{b\in D(a)} &\longmapsto \left(\delta_a\left((q_b)_{b\in D(a)}\right)\right)_{c\in D(a)}\end{aligned}$$

In $\mathcal{A}'$ the write domain of a letter a is $W'(a) = \{((a,b)) \mid b \in D(a)\}$. The read domain above has reduced to $R'(a) = \{(b,a) \mid b \in D(a)\}$ rather than putting artificially $R'(a) \cup W'(a)$ (only in order to satisfy our restriction $W'(a) \subseteq R'(a)$). Thus our construction realizes even an owner-read-owner-write (*OROW*) concept. In any case, it is clear how to define the initial state and the global final states in order to archive $L(\mathcal{A}) = L(\mathcal{A}')$. Note that, although the number of global states became larger, the number of reachable states did not change. In some sense the automaton did not change at all.

7.4 Changing Exclusive-Write to Owner-Write

Let $\mathcal{A} = \left(\prod\limits_{i\in J} Q_i, (\delta_a)_{a\in\Sigma}, q_0, F\right)$ be a finite asynchronous automaton of *EREW*-type. The transformation into an asynchronous cellular automaton is more involved. We are looking for a *CROW*-type. However, as a matter of fact, we will construct an *EROW*-type. Thus, the possibility of concurrent-read does not help us for the requirement owner-write.

We are going to explain the transformation in detail. For each $i \in J$ define $\Sigma_i = \{a \in \Sigma \mid i \in R(a)\}$. Recall $W(a) \subseteq R(a)$ for all $a \in \Sigma$. Therefore, we may assume $\Sigma = \bigcup_{i\in J} \Sigma_i$. For $i \in J$ let $n_i = |\Sigma_i|$ and let $f_i : \Sigma_i \to \mathbb{Z}/n_i\mathbb{Z}$ be a fixed bijection. For all (a,i) with $a \in \Sigma_i$ we define a local component $P_{(a,i)} = \mathbb{Z}/n_i\mathbb{Z} \times Q_i$. The new initial local state is $(f_i(a), (q_0)_i)$. (In fact the right-hand side is always $(q_0)_i$, so the value on the left-hand side is of no importance for the initial state. Another choice would be $(42, (q_0)_i)$, if you believe that $42 \in \mathbb{Z}/n_i\mathbb{Z}$

is a nice value [3], but our choice involves more mathematical philosophy.) The notion for a local state of $P_{(b,i)}$ is in the following $(n(b,i), q(b,i))$ and the new set of global states is $P = \prod_{i \in J} \prod_{b \in \Sigma_i} Q_i$. The description of an a-transition, $a \in \Sigma$, is as follows.

Read all local states $(n(b,i), q(b,i))$ where $i \in R(a)$ and $b \in \Sigma_i$. Note that this is exclusive read, since $(a,c) \in I$ implies $R(a) \cap R(c) = \emptyset$ for the automaton $\mathcal{A}$. The problem is to know which value $q(b,i)$ for $b \in \Sigma_i$ is actual valid. To decide this we compute for each $i \in R(a)$ the sum

$$\sum_{b \in \Sigma_i} n(b,i) \quad \in \quad \mathbb{Z}/n_i\mathbb{Z}.$$

This defines a unique letter $b_i \in \Sigma_i$ such that $f_i(b_i) \equiv \sum_{b \in \Sigma_i} n(b,i) (\mathrm{mod} n_i)$.

Using this letter b_i we decide $q(b_i, i)$ to be valid and set $q_i = q(b_i, i) \in Q_i$. Now we can use table-look-up from $\mathcal{A}$ and define

$$q_i' \quad = \quad \left(\delta_a\left((q_i)_{i \in R(a)}\right)\right)_i \qquad \text{for } i \in W(a) = R(a).$$

Since we are interested in owner-write, we change the components $P_{(a,i)}$ for $i \in R(a)$, only.

The new state $(n'(a,i), q'(a,i))$ is given by $q'(a,i) = q_i'$ and $n'(a,i)$ such that

$$n'(a,i) + \sum_{b \neq a} n(b,i) \quad \equiv \quad f_i(a) (\mathrm{mod} n_i).$$

By this trick, whenever one is allowed to read the components $\prod_{b \in \Sigma_i} Q_i$, one will find by summing up the modular values that actually $P_{(a,i)}$ knows the correct value of q_i.

It is therefore clear that we can simulate $\mathcal{A}$ step by step and how to define final states in order to complete the transformation.

We should add a remark on the size of the new automaton. A reasonable definition of the size of $\mathcal{A}$ is the sum $|\Sigma| + \sum_{i \in J} |Q_i| +$ length of the program which implements the family $(\delta_a)_{a \in \Sigma}$. (At first sight this definition might look strange, but it is a quite realistic measure.) With this notion of size, we can state the following proposition.

Proposition 85. *For every asynchronous automaton of EREW-type we can construct in polynomial time an asynchronous cellular automaton recognizing the same language.*

7.5 The Construction for Triangulated Dependence Alphabets

A surprisingly simple construction of asynchronous automata is known for triangulated dependence alphabets. An undirected graph is called *triangulated,* if all its chordless cycles are of length three. Particular cases are therefore complete graphs and acyclic graphs (acyclic means a disjoint union of trees). A dependence alphabet (Σ, D) is called triangulated, if the underlying undirected graph has this property. A *perfect vertex elimination scheme* of (Σ, D) is a linear ordering $\geq$ of Σ such that for all $a \in \Sigma$ the set $E(a) = \{b \in D(a) \mid a \geq b\}$ forms a clique (i.e., a complete subgraph; $(E(a))^*$ is therefore a free monoid). We may represent a perfect vertex elimination scheme by a list $[a_1, \ldots, a_n]$ such that $a_i \geq a_j$ if and only if $i \leq j$. It is well-known, see e.g. the textbook of Golumbic [56, Thm. 4.1] that (Σ, D) has a perfect vertex elimination scheme if and only if it is triangulated. If e,g, (Σ, D) is acyclic, then any ordering which represents a topological sorting yields a perfect vertex elimination scheme. For a complete graph, every total order is a perfect vertex elimination scheme.

Let (Σ, D) be any dependence alphabet and let $\leq$ be a linear ordering of Σ such that for all $a, b, c \in \Sigma$, $a \leq b \leq c$, $(a, c) \in D$ and $(b, c) \in D$, we have $(a, b) \in D$, too. Thus, if $\Sigma = \{a_1, \ldots, a_n\}$, $n = |\Sigma|$ is written in increasing order $a_1 < \cdots < a_n$, then $[a_n, \ldots, a_1]$ is a perfect vertex elimination scheme of (Σ, D). Hence (Σ, D) is triangulated. Consider now a homomorphism to a finite monoid $\eta : \mathbb{M}(\Sigma, I) \to S$.

Define for $\Gamma \subseteq \Sigma$ and $s_c \in S$ $(c \in \Gamma)$ the ordered product $\overrightarrow{\prod}_{c \in \Gamma} s_c$ by multiplying the elements according to the ordering in Σ. Thus, $\overrightarrow{\prod}_{c \in \Gamma} s_c = s_{c_1} \cdots s_{c_m}$, if $\Gamma = \{c_1, \ldots, c_m\}$, $m \geq 0$, with $c_1 < \cdots < c_m$. All products of elements of S used here will be ordered ones. We are going to construct an asynchronous automaton $\mathcal{A}$ recognizing a trace language $T = \eta^{-1}(\eta(T))$.

The state set of $\mathcal{A}$ will be an n-fold direct product of S, $n = |\Sigma|$. Let $\mathcal{A} = (\prod_{a \in \Sigma} Q_a, \delta, q_0, F)$ be the automaton with $Q_a = S$ for all $a \in \Sigma$ such that the global transition function

$$\delta : (\prod_{a \in \Sigma} Q_a) \times \Sigma \to \prod_{a \in \Sigma} Q_a,$$

is defined for $q = (q_b)_{b \in \Sigma}$ and $a \in \Sigma$ as follows:

$$\delta(q, a) = q' \text{ with } \; q_b{}' = \begin{cases} \Big(\overrightarrow{\prod}\limits_{a \leq c,\ (a,c) \in D} q_c \Big) \cdot \eta(a) & \text{if } b = a \\ 1 & \text{if } a < b,\ (a, b) \in D \\ q_b & \text{otherwise} \end{cases}$$

Furthermore, let $q_0 = (1)_{a \in \Sigma}$ and $F = \{(q_a)_{a \in \Sigma} \in \prod_{a \in \Sigma} Q_a \mid \overrightarrow{\prod}\limits_{a \in \Sigma} q_a \in \eta(T)\}$.

Proposition 86. *Let $T \subseteq \mathbb{M}(\Sigma, I)$ be recognized by the homomorphism $\eta : \mathbb{M}(\Sigma, I) \to S$ to the finite monoid S. Let $\mathcal{A}$ be the automaton defined above. Then $\mathcal{A}$ is asynchronous of EREW-type and we have $L(\mathcal{A}) = T$.*

Proof. The write-and-read domain of a letter $a \in \Sigma$ is given by the index set $R(a) = W(a) = \{b \in D(a) \mid a \leq b\}$. Assume that for some $a, b, c \in \Sigma$, $a \leq b \leq c$ we have $(a, c) \in D$ and $(b, c) \in D$. Then $(a, b) \in D$ follows, due to the ordering. Hence, if $(a, b) \in I$, then a and b have disjoint write-and-read domains. Thus,

$\mathcal{A}$ is asynchronous of *EREW*-type. In particular, for any trace $w \in \mathbb{M}(\Sigma, I)$, the global state $\delta((1)_{a\in\Sigma}, w)$ is well-defined. We denote this global state also by $q_0 \cdot w$.

For $(q_a)_{a\in\Sigma} = q_0 \cdot w$ the following two invariants can be shown:

i) $a \leq b$ and $(a,b) \in I$ imply $q_b\, \eta(a) = \eta(a)\, q_b$.
ii) $a \leq b \leq c$, $(a,b) \in I$, and $(a,c) \in D$ imply $q_b\, q_c = q_c\, q_b$.

The proposition follows from the following claim:

$$\text{For } (q_a)_{a\in\Sigma} = q_0 \cdot w \quad \text{we have} \quad \overrightarrow{\prod_{a\in\Sigma}} q_a = \eta(w).$$

The claim is satisfied for $|w| = 0$ since $q_0 = (1)_{a\in\Sigma}$. By induction assume that the claim holds for $q = q_0 \cdot w$ and let $q' = q \cdot a$ for some letter $a \in \Sigma$. Since $\overrightarrow{\prod\limits_{c\in\Sigma}} q_c = (\overrightarrow{\prod\limits_{c<a}} q_c) \cdot (\overrightarrow{\prod\limits_{a\leq c}} q_c)$, it is enough to show:

$$(\overrightarrow{\prod_{a\leq c}} q_c)\ \eta(a) = (\overrightarrow{\prod_{a\leq c,\ (a,c)\in D}} q_c)\ \eta(a)\ (\overrightarrow{\prod_{a<b,\ (a,b)\in I}}) q_b).$$

However, this last formula is immediate from the two invariants above.

Remark. Note that as soon as the dependence graph contains a chordless cycle of length greater than three, the automaton constructed above is not asynchronous anymore. More precisely, for any ordering $\leq$ of the alphabetΣ there exist letters $a < b < c$ (on the cycle) satisfying $(a,c) \in D$, $(b,c) \in D$, but $(a,b) \in I$. In this case however, c belongs to both write-and-read domain of a and b.

7.6 Bounded Time-Stamps in a Distributed System

Suppose that in a distributed system some *agents* communicate by means of messages. Usually, to execute correctly the prescribed protocol the agents should have some knowledge about the relative order of messages. To this end, they add to every message a *tag*, called a *time-stamp*, enabling them to find out the necessary information about the ordering of messages. The importance of an appropriate stamping algorithm was for the first time emphasized by Lamport (1978) [66], to which we refer the reader for further discussion. In most cases, it is a relatively easy task to construct an appropriate stamping system if no bounds on the size of stamps are imposed. But if we allow only a finite set of time-stamps then the construction of an appropriate stamping system becomes difficult or sometimes even impossible. In this section we show how to use the result about asynchronous automata to construct a special finite time-stamp system.

The distributed system considered here consists of a finite set Σ of agents and a finite set B of boxes. The agents communicate by messages that they leave in some boxes. Every agent $a \in \Sigma$ has access only to a subset $Dom(a) \subseteq B$ of boxes (following previous notations, we consider here that $Dom(a) = R(a) = W(a)$.) Conversely, for every box $i \in B$, $\Sigma_i = \{a \in \Sigma \mid i \in Dom(a)\}$ is the set of agents which have access to i. If $i \in Dom(a)$ then we say that the box i and the agent a are adjacent.

By B_i, for $i \in B$, we shall denote the contents of the box i, i.e. the set of messages containing i. We assume that at the beginning all boxes are empty, i.e., $B_i = \emptyset$ for all $i \in B$.

Every message is a triple (m, a, d), where m is the contents of the message taken from some set M of possible contents; $a \in \Sigma$ identifies the sender of the message, and finally d is a time-stamp from some set *Stamps* of time-stamps. Thus formally the Cartesian product $U = M \times A \times Stamps$ is the set of all messages. In the following, by *contents*, *sender*, and *stamp* we shall denote the projection of U onto M, Σ, and *Stamps* respectively. Furthermore, we assume that any box contains for any $a \in \Sigma$ at most one message sent by a.

During their moves the agents not only will send new messages but also will retransmit messages sent by other agents. For this reason, besides messages left by agents adjacent to i, every box $i \in B$ can contain messages sent by other agents and retransmitted by agents from Σ_i.

A single move of each agent $a \in \Sigma$ consists of four phases. During the first phase a reads the contents of all adjacent boxes, emptying them in this way. Let R be the set of messages that were read in this phase.

In the second phase, for every $b \in \Sigma \setminus \{a\}$, if R contains messages sent by b then a selects the last of them (the most recent), denoted it u_b.

In the third phase, a chooses $m \in M$ that it wishes to send and computes a time-stamp $d \in Stamps$. Let $u_a = (m, a, d)$.

Finally, in the last phase a transmits to all adjacent boxes all messages from the set $\{u_c \mid c \in \Sigma\}$.

The entire move (consisting of reading, selecting, constructing a new message, and sending) is considered to be atomic. This implies that the access to every box is sequential, and moreover, at a given moment, an agent a has access either to all its adjacent boxes or to none of them. Note that immediately after the move all boxes adjacent to a have the same contents: for every agent $b \in \Sigma$, they contain at most one message issued by b, namely the last message sent by b and known to a. We assume that for every message $u \in U$ the field $contents(u)$ does not provide any information concerning the relative order of messages. Thus during the second phase of every move, agent a can use only the fields $sender(u)$ and $stamp(u)$ of $u \in R$ to find out for every $b \in \Sigma \setminus \{a\}$ the last message in R sent by b.

To implement this system we should specify: the set *Stamps*, the algorithm selecting messages in the second phase of each move, and the algorithm assigning a stamp to the new message created in the third phase.

A simple implementation exists if we allow the set *Stamps* to be infinite. Let $Stamps = \mathbb{N}_+$, and assume that every agent is equipped with a counter initially set to 0. Then during its move, agent a increases its counter by 1 and takes the obtained value as the time-stamp d for its new message in the third phase,

$u_a = (m, a, d)$. The selection procedure in the second phase of the move is trivial in this implementation. For every $b \in \Sigma \setminus \{a\}$, a takes all messages in R with the sender field equal to b and selects among them the one with the greatest stamp field.

The aim of this section is to present a different implementation with a bounded number of time-stamps. First we define some auxiliary notions. By a serial event we shall mean any finite sequence of elements of the set $M \times \Sigma$, $\mathrm{SE} = (M \times \Sigma)^*$. Any occurrence of $(m, a) \in M \times \Sigma$ in a serial event $x \in \mathrm{SE}$ represents a move performed by the agent a such that m is the contents of the new message sent by a during this move. Let us suppose that α is an algorithm implementing the system. For every $x \in \mathrm{SE}$ and $i \in \mathit{Box}$ by $B_i^\alpha(x)$ we denote the contents of the box i after the execution of the serial event x in the implementation α; $B_i^\alpha(x)$ can be defined in the following inductive manner:

(i) for all $i \in \mathit{Box}$, $B_i^\alpha(\epsilon) = \emptyset$,

(ii) if $y = x(m, a) \in \mathrm{SE}$ then

(1) for all $i \in B \setminus \mathit{Dom}(a)$, $B_i^\alpha(y) = B_i^\alpha(x)$,

(2) to obtain the new contents of all boxes adjacent to a apply the algorithm α to $B_i = B_i^\alpha(x)$ with $i \in \mathit{Dom}(a)$.

Example 15. Let $\Sigma = \{a, b, c, d\}$ and $B = \{B_{ab}, B_{bc}, B_{cd}, B_{da}\}$, where B_{xy} with $x, y \in \Sigma$, denotes the box adjacent to x and y.

Let $S = (m_1, a)(m_2, b)(m_3, a)(m_4, c)(m_5, d)$ be a serial event and let s_i denote the prefix of s of length $i = 0, \ldots, 5$.

The contents of the boxes after the execution of S_i in the counter implementation that was considered previously are as follows.

After S_0: $B_{ab} = \emptyset$, $B_{bc} = \emptyset$, $B_{cd} = \emptyset$, $B_{da} = \emptyset$;
After S_1: $B_{ab} = \{(m_1, a, 1)\}$, $B_{bc} = \emptyset$, $B_{cd} = \emptyset$, $B_{da} = \{(m_1, a, 1)\}$;
After S_2: $B_{ab} = \{(m_1, a, 1), (m_2, b, 1)\}$, $B_{bc} = \{(m_1, a, 1), (m_2, b, 1)\}$, $B_{cd} = \emptyset$, $B_{da} = \{(m_1, a, 1)\}$;
After S_3: $B_{ab} = \{(m_3, a, 2), (m_2, b, 1)\}$, $B_{bc} = \{(m_1, a, 1), (m_2, b, 1)\}$, $B_{cd} = \emptyset$, $B_{da} = \{(m_3, a, 2), (m_2, b, 1)\}$;
After S_4: $B_{ab} = \{(m_3, a, 2), (m_2, b, 1)\}$, $B_{bc} = \{(m_1, a, 1), (m_2, b, 1), (m_4, c, 1)\}$, $B_{cd} = \{(m_1, a, 1), (m_2, b, 1), (m_4, c, 1)\}$, $B_{da} = \{(m_3, a, 2), (m_2, b, 1)\}$;
After S_5: $B_{ab} = \{(m_3, a, 2), (m_2, b, 1)\}$, $B_{bc} = \{(m_1, a, 1), (m_2, b, 1), (m_4, c, 1)\}$, $B_{cd} = \{(m_3, a, 2), (m_2, b, 1), (m_4, c, 1), (m_5, d, 1)\}$, $B_{da} = \{(m_3, a, 2), (m_2, b, 1), (m_4, c, 1), (m_5, d, 1)\}$.

Let I be the independence relation over Σ defined in the following way:

$$I = \{(a, b) \in \Sigma^2 \mid \mathit{Dom}(a) \cap \mathit{Dom}(b) = \emptyset\}.$$

To define time-stamps we need some definitions. First we define the set of *prime elements,* denoted $Pr(\Sigma, I)$, by

$$Pr(\Sigma, I) = \{\partial_a(t) \mid a \in \Sigma \quad \mathit{and} \quad t \in \mathbb{M}(\Sigma, I)\}.$$

We can associate to a message a trace which encodes the partial order corresponding to the communication between agents. As we want bounded time-stamps, we define a map having a finite image and carrying enough information on a trace t to know the prefix order on the traces $\partial_a(t)$ for $a \in \Sigma$. We thus define a labeling λ from $Pr(\Sigma, I)$ into the set of positive integers by

$$\lambda(1) = |\Sigma|,$$

and for $a \in \Sigma$, $t \in \mathbb{M}(\Sigma, I)$ such that $ta \in Pr(\Sigma, I)$,

$$\lambda(ta) = Min\{i \in \mathbb{N} \setminus \{0\} \mid \forall b \in \Sigma \setminus \{a\} \quad i \neq \lambda(\partial_a(\partial_b(t)))\}.$$

From this mapping λ we construct for every trace t a mapping ν_t from $\Sigma \times \Sigma$ into $\mathbb{N} \setminus \{0\}$ setting for any pair (a, b) of letters

$$\nu_t(a, b) = \lambda(\partial_a(\partial_b(t))).$$

We denote by $F(X; Y)$ the family of all partial mappings from X to Y. It turns out that the mapping ν associating with a trace t of $\mathbb{M}(\Sigma, I)$ the element ν_t of $F(\Sigma \times \Sigma; \{1, \dots, |\Sigma|\})$ is asynchronous. Its importance is emphasized by the fact that all asynchronous mappings we need in the proof of Zielonka's Theorem are refinements of this basic mapping ν.

We set $Stamps = F(\Sigma; \{1, \dots, n\})$, where $n = |\Sigma|$. Let tag be the mapping from $Pr(\Sigma, I)$ into $Stamps$ defined in the following way:

$$\forall t \in Pr(\Sigma, I), \quad \forall a \in \Sigma : \quad tag(t)(a) = \lambda(\partial_a(t)).$$

Now the idea of the implementation γ is to assign to messages elements of *Stamps* in such a way that the following condition is satisfied (we recall that $\Sigma_i = \{a \in \Sigma \mid i \in Dom(a)\}$).
Condition (Γ) **:** $\forall h \in H$, $\forall b \in \Sigma$, $\forall i \in B$, $\forall t \in \mathbb{M}(\Sigma, I)$, $\forall k > 0$:

- $t = sender(h) \in \mathbb{M}(\Sigma, I)$,
- $|\partial_b(\partial_{\Sigma_i}(t))|_b = |\partial_{\Sigma_i}(t)|_b = k > 0$,
- (m, b) is the k-th message of b in h,

then:

$$(m, b, tag(\partial_b(\partial_{\Sigma_i}(t)))) \in B_i^\gamma(h).$$

If Condition (Γ) holds, $h \in H$, and $t = sender(h)$, then for each $i \in B$ the time-stamps of the messages in the box i determine the mapping $\nu_{\partial_{\Sigma_i}(t)}$.

Lemma 87. *Suppose that an implementation γ satisfies (Γ). Let $h \in H$, $t = sender(h) \in \mathbb{M}(\Sigma, I)$. Then:*

$$\nu_{\partial_{\Sigma_i}(t)}(c, b) = \begin{cases} n & \textit{if } B_i^\gamma(h) \textit{ does not contain messages sent by } b \\ f(c) & \textit{if } (m, b, f) \in B_i^\gamma(h). \end{cases}$$

Now we can present the details of the algorithm γ. Let $a \in \Sigma$ and let $h \in H$ be the history executed up to now. Let $t = \mathit{sender}(h) \in \mathbb{M}(\Sigma, I)$. Inspecting the contents of box i, a can calculate $\nu_{\partial_{\Sigma_i}(t)}$ for all $i \in \mathit{Dom}(a)$. Now by the uniformity of ν, a obtains $\nu_{\partial_{D(a)}(t)}$. During the selection phase, a chooses for every $b \in \Sigma \setminus \{a\}$ from the reading set R the message $u_b = (m, b, f)$ such that:

$$\forall c \in \Sigma \quad f(c) = \nu_{\partial_{D(a)}(t)}(c, b).$$

As ν is asynchronous, a can now get $\nu_{\partial_a(ta)}$. Let g be the mapping from Σ into $\{1, \ldots, n\}$ such that:

$$\forall c \in \Sigma, \quad g(c) = \nu_{\partial_a(ta)}(c, a).$$

Then g is the time-stamp for the new message that a creates during the third phase, $u_a = (m, a, g)$. To show the correctness of γ it suffices to observe that if Condition (Γ) holds for a history h then it holds for the history $h(m, a)$ for every $(m, a) \in M \times \Sigma$.

7.7 Bibliographical Remarks

Zielonka's Theorem is from [90]. Proofs based on the notion of asynchronous mappings can be found in [20, 24], and [34, Chapt.8]. Asynchronous cellular automata have been introduced in [91]. The transformations between different types of asynchronous automata have been studied in detail by Pighizzini [82]. The construction for triangulated dependence graphs is from Diekert and Muscholl [32]. It heavily uses the main idea from Métivier [72], who developed this method for acyclic dependence graphs. Bertoni, Mauri, and Sabadini used in [7] a similar method for acyclic Petri nets, but the latter paper was not widely distributed. Bounded time-stamps are from [20].

8 Infinite Traces

8.1 Real Traces

An infinite word is a mapping $u : \mathbb{N} \longrightarrow \Sigma$. We can write $u = u(0)u(1)\cdots$ with $u(i) \in \Sigma$. The set of infinite words is denoted by Σ^ω; hence $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$ is the set of finite or infinite words. The mapping $\varphi_\mathbb{G} : \Sigma^* \longrightarrow \mathbb{G}(\Sigma, D)$ from words to dependence graphs has a natural extension to Σ^∞ For $u = u(0)u(1)\cdots \in \Sigma^\omega$ the dependence graph $\varphi_\mathbb{G}(u) = [V, E, \lambda]$ is defined as follows: The vertex set is $V = \mathbb{N}$ with $\lambda(i) = u(i)$. There is an arc $(i, j) \in E$ if and only if $i < j$ and $(u(i), u(j)) \in D$. The image $\varphi_\mathbb{G}(u) \in \mathbb{G}$, $u \in \Sigma^\infty$ is called *real trace*. The set of real traces is denoted by $\mathbb{R}(\Sigma, D)$. This set contains finite and infinite dependence graphs. For simplicity we write φ instead of $\varphi_\mathbb{G}$. Hence $\varphi : \Sigma^\infty \longrightarrow \mathbb{R}(\Sigma, D)$ is a surjective mapping. A *finitary* language is a subset of $\mathbb{M}(\Sigma, I)$, i.e., a language over finite traces.

For a dependence graph $g = [V, E, \lambda]$ we infer standard notations from the finite case. In particular, we can speak of $\mathrm{alph}(g)$, of g being connected, and of a decomposition into connected components. As usual, we write gIh, if $\mathrm{alph}(g) \times \mathrm{alph}(h) \subseteq I$.

Let $g = [V, E, \lambda] \in \mathbb{G}(\Sigma, D)$ and $x \in V$ be a vertex. By $\downarrow x$ we mean the dependence graph being induced by the set $\{y \in V \mid (y, x) \in E^*\}$. Thus, it is the dependence graph (having a unique maximal element), being induced by all nodes below or equal to x. By Prop. 68 we can write $g = (\downarrow x)h$ for some uniquely determined $h \in \mathbb{G}(\Sigma, D)$.

Remark. A dependence graph $g = [V, E, \lambda]$ is a real trace, i.e., $g = \varphi_{\mathbb{G}}(u)$ for some word $u \in \Sigma^\infty$, if and only if $\downarrow x$ is finite for all $x \in V$. This property characterizes therefore $\mathbb{R}(\Sigma, D) \subseteq \mathbb{G}(\Sigma, D)$.

Having this notation, we may define the decomposition of a dependence graph into its real (standard) and transfinite part:

Definition 88. For a dependence graph $g = [V, E, \lambda] \in \mathbb{G}(\Sigma, D)$ we define its *real part* $Re(g) \in \mathbb{R}(\Sigma, D)$ to be the dependence graph being induced by the set $\{x \in V \mid \downarrow x \text{ is finite}\}$. The *transfinite part* $Tr(g) \in \mathbb{G}(\Sigma, D)$ is the dependence graph which is induced by the set $\{x \in V \mid \downarrow x \text{ is infinite}\}$. The *alphabet at infinity*, $\mathrm{alphinf}(g) \subseteq \Sigma$, is the set of labels

$$\mathrm{alphinf}(g) = \{\lambda(x) \in \Sigma \mid \lambda(x) \text{ appears infinitely often or } \downarrow x \text{ is infinite}\}.$$

The following proposition is obvious.

Proposition 89. *Let $g, h \in \mathbb{R}(\Sigma, D)$ be real traces. Then*

$$\mathit{alphinf}(g) = \{\lambda(x) \in \Sigma \mid \lambda(x) \textit{ appears infinitely often }\}.$$

The product $g \cdot h \in \mathbb{G}(\Sigma, D)$ is a real trace if and only if $\mathit{alphinf}(g) \times \mathit{alph}(h) \subseteq I$.

It is straight forward to define iterations and ω-products for dependence graphs. Closing the family of rational languages over finite traces under the operation union, concatenation, Kleene-star, and ω-product, we obtain the family of *ω-rational* (or *rational* for short) dependence graphs $\mathrm{Rat}(\mathbb{G}(\Sigma, D))$. The family of rational real trace languages is defined by

$$\mathrm{Rat}(\mathbb{R}(\Sigma, D)) = \{L \subseteq \mathbb{R}(\Sigma, D) \mid L \in \mathrm{Rat}(\mathbb{G}(\Sigma, D))\}.$$

Using Prop. 89 and well-known closure properties for the family of ω regular word languages $\mathrm{Rat}(\Sigma^\infty)$ the following characterization can be shown.

Proposition 90. *The family of rational real trace languages is the smallest family containing $Rat(\mathbb{M}(\Sigma, I))$ and being closed under union, concatenation with finitary languages on the left, Kleene-star and ω-products over finitary languages.*

Example 16. Let $a, b, c \in \Sigma$, $(a, b) \in I$, $(a, c) \in D$, $t \in \mathbb{M}(\Sigma, I)$, and $g \in \mathbb{G}(\Sigma, D)$. Then we have:

– $a^\omega b^\omega = (ab)^\omega \in \mathrm{Rat}(\mathbb{R}(\Sigma, D))$.

– $a^\omega c$ is not a real trace, $Re(a^\omega c) = a^\omega$, $Tr(a^\omega c) = c$, and $\text{alphinf}(a^\omega c) = \{a, c\}$. Hence, $\text{alphinf}(a^\omega c) = \text{alphinf}(a^\omega)$ if and only if $a = c$.
– $tg^2 \in \mathbb{R}(\Sigma, D)$ if and only if g is finite.

Proposition 91. *A real trace language $L \subseteq \mathbb{R}(\Sigma, D)$ is rational if and only if it can be written as a finite union*

$$L = \bigcup_{\text{finite}} RT^\omega$$

over finitary rational trace languages R and T.

As in the case of finitary languages we can define the families of star-connected, c-rational, and star-free real trace languages.

For star-connectedness: the Kleene-star and the ω-product are allowed over finitary and connected languages, only.

For c-star: the Kleene-star and the ω-product are replaced by operations, where first we take the language of connected components and then apply the Kleene-star (ω-product resp.) over this connected language.

In Sect. 6.2 we have define first-order and monadic second-order sentences. Of course, being a dependence graph every real trace yields an interpretation for such a sentence.

Finally, we extend the notion of recognizability from infinite words to real traces.

We say that a homomorphism $\eta : \mathbb{M}(\Sigma, I) \longrightarrow S$ to a finite semigroup *recognizes* a real trace language $L \subseteq \mathbb{R}(\Sigma, D)$, if for all infinite sequences $s_0 s_1 s_2 \cdots$ and $t_0 t_1 t_2 \cdots$ with $\eta(s_i) = \eta(t_i)$ for all $i \geq 0$ we have the equivalence

$$s_0 s_1 s_2 \cdots \in L \iff t_0 t_1 t_2 \cdots \in L.$$

Definition 92. A real trace language $L \subseteq \mathbb{R}(\Sigma, D)$ is called *recognizable*, if there exists some recognizing homomorphism $\eta : \mathbb{M}(\Sigma, I) \longrightarrow S$ to a finite semigroup S.

For star-freeness we start with the star-free finitary trace language. Then we take the closure under Boolean operations (with respect to $\mathbb{R}(\Sigma, D)$) and under concatenation with finitary languages on the left. Neither Kleene-star nor ω-product are allowed. It is easy to verify that for all $A, B \subseteq \Sigma$ the set $\{g \in \mathbb{R}(\Sigma, D) \mid \text{alph}(g) \subseteq A, \text{alphinf}(g) \subseteq B\}$ is star-free. Note also that the language $(ab)^\omega$ is star-free, whether or not $(a, b) \in D$.

Let $L \subseteq \mathbb{R}(\Sigma, D)$ be a real trace language. We define the syntactic congruence $\equiv_L \subseteq \mathbb{M}(\Sigma, I) \times \mathbb{M}(\Sigma, I)$ by setting $u \equiv_L v$ for $u, v \in \mathbb{M}(\Sigma, I)$ if and only if for all $x, y, z \in \mathbb{M}(\Sigma, I)$ we have:

$$xuyz^\omega \in L \iff xvyz^\omega \in L$$
$$x(uy)^\omega \in L \iff x(vy)^\omega \in L$$

As for infinite words we have the following proposition:

Proposition 93. *A real trace language $L \subseteq \mathbb{R}(\Sigma, D)$ is recognizable if and only if we have both, the syntactic congruence $\equiv_L$ is of finite index and the syntactic homomorphism $\eta_L : \mathbb{M}(\Sigma, I) \longrightarrow \mathbb{M}(\Sigma, I)/\equiv_L$ recognizes L.*

Proposition 94. *Let $\varphi : \Sigma^\infty \longrightarrow \mathbb{R}(\Sigma, D)$ be the canonical mapping, $L \subseteq \mathbb{R}(\Sigma, D)$ be a real trace language, and $K = \varphi^{-1}(L)$. Then the following assertions hold:*

i) The syntactic monoids $\Sigma^*/\equiv_K$ and $\mathbb{M}(\Sigma, I)/\equiv_K$ are canonically isomorphic.
ii) The syntactic homomorphism $\eta_K : \Sigma^* \longrightarrow \Sigma^*/\equiv_K$ recognizes K if and only if $\eta_L : \mathbb{M}(\Sigma, I) \longrightarrow \mathbb{M}(\Sigma, I)/\equiv_L$ recognizes L.

Corollary 95. *A real trace language $L \subseteq \mathbb{R}(\Sigma, D)$ is recognizable if and only if $\varphi^{-1}(L) \subseteq \Sigma^\infty$ is recognizable.*

A word language $K \subseteq \Sigma^\infty$ is called *closed*, if $K = \varphi^{-1}\varphi(K)$.

Proposition 96. *A recognizable word language $K \subseteq \Sigma^\infty$ is closed if and only if $ab \equiv_K ba$ for all $(a, b) \in I$.*

The implication "$\Longrightarrow$" in the proposition above is trivial and holds for every closed language. Unlike the finitary case the converse is less obvious. The proof uses Ramsey-factorization and the fact that the language K is recognized by the syntactic homomorphism $\eta_K : \Sigma^* \longrightarrow \Sigma^*/\equiv_K$. The formal proof is left to the reader. The proposition above yields a decidability result immediately.

Corollary 97. *It is decidable whether a recognizable word language is closed.*

Bringing together the different notions introduced so far, we can state the following theorem.

Theorem 98. *Let $\varphi : \Sigma^\infty \longrightarrow \mathbb{R}(\Sigma, D)$ be the canonical mapping, $L \subseteq \mathbb{R}(\Sigma, D)$ be real trace language, and $\eta_L : \mathbb{M}(\Sigma, I) \longrightarrow \mathbb{M}(\Sigma, I)/\equiv_L$ be the syntactic homomorphism. Then the following assertions are equivalent.*

i) L is a recognizable of $\mathbb{R}(\Sigma, D)$.
ii) $\varphi^{-1}(L)$ is a recognizable of Σ^∞.
iii) The syntactic congruence $\equiv_L$ is of finite index and $\eta_L$recognizes L.
iv) L is star-connected.
v) L is c-rational.
vi) L is definable in monadic second-order logic.
vii) The language L can be written as a finite union

$$L = \bigcup_{\text{finite}} RT^\omega,$$

where R and T are finitary recognizable trace languages such that $T = T^*$.

The first-order counterpart can be stated as follows:

Theorem 99. *Under the same assumptions as above, the following assumptions are equivalent:*

i) L *is star-free.*
ii) $\varphi^{-1}(L)$ *is star-free.*
iii) *The syntactic monoid* $\mathbb{M}(\Sigma, I)/\equiv_L$ *is finite, aperiodic, and* η_L *recognizes* L.
iv) L *is definable in first-order logic.*
v) *The language* L *can be written as a finite union*

$$L = \bigcup_{\text{finite}} RT^{\omega},$$

where R *and* T *are finitary star-free languages such that* $T = T^*$.

8.2 Asynchronous Büchi- and Muller Automata

A *non-deterministic finite asynchronous cellular automaton* $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, s, (\delta_a)_{a\in\Sigma}, \mathcal{T})$ where $Q = \prod_{a\in\Sigma} Q_a$ is the set of global states, $s = (s_a)_{a\in\Sigma}$ is the initial state, $\delta_a \subseteq \left(\prod_{b\in D(a)} Q_b\right) \times Q_a$ is the local transition relation, and $\mathcal{T} = \{T_1, \ldots, T_n\}$ is the acceptance table for runs, and each T_i satisfies $T_i = \prod_{a\in\Sigma} T_{i,a}$ for some $T_{i,a} \subseteq Q_a$.

The automaton is called *deterministic*, if the local transition relations δ_a are partially defined functions.

A *run* r of $\mathcal{A}$ on a real trace $g = [V, E, \lambda]$ is a labeling function $r : V \longrightarrow \dot{\bigcup}_{a\in\Sigma} Q_a$ satisfying the following conditions:

i) If $\lambda(x) = a$ for $x \in V$, then we have $r(x) \in Q_a$.
ii) Let $x \in V$, $\lambda(x) = a$, and $r(x) = q_a$. For each $b \in D(a)$ let q_b either be $r(x_b)$, if there exists the maximal vertex x_b with $(x_b, x) \in E^*$ and $\lambda(x_b) = b$, or let $q_b = s_b$ (the b-component of the initial state) otherwise. Then we have

$$\left((q_b)_{b\in D(a)}, q_a\right) \in \delta_a.$$

We define the infinite behavior of a run r on g as follows:

$$\inf(r) = \Big\{ q \in Q \mid q = (q_a)_{a\in\Sigma},\ \text{either } r^{-1}(q_a) \text{ contains infinitely many vertices or it contains the maximal vertex with label } a, \text{ or if } a \notin \mathrm{alph}(g), \text{ then } q_a = s_a \Big\}.$$

For $\mathcal{A}$ a *Büchi*- and a *Muller* acceptance condition are defined. Accordingly $\mathcal{A}$ is called a *Büchi*- or *Muller automaton.* A real trace $g \in \mathbb{R}(\Sigma, D)$ is Büchi-accepted if there exist a run r on g and some $T_i \in \mathcal{T}$ such that $T_i \subseteq \inf(r)$. It is Muller-accepted if we demand equality $T_i = \inf(r)$ for some $T_i \in \mathcal{T}$.

Although the Muller acceptance is more powerful, it is easy to see that the non-deterministic models are equivalent in the sense that they lead to the same class of accepted languages.

Theorem 100. *Let $L \subseteq \mathbb{R}(\Sigma, D)$ be a real trace language. Then the following assertions are equivalent:*

i) *L is recognizable.*
ii) *L is accepted by some non-deterministic finite asynchronous cellular Büchi-automaton.*
iii) *L is accepted by some non-deterministic finite asynchronous cellular Muller-automaton.*
iv) *L is accepted by some deterministic finite asynchronous cellular Muller-automaton.*

Remark. The equivalence $i) \Leftrightarrow iv)$ above generalizes McNaughton's Theorem to real traces. The original proof in [31] transforms via an implicit double exponential algorithm a given non-deterministic Büchi-automaton for a closed word language into a deterministic finite asynchronous cellular Muller-automaton. This proof does not use the equivalence $i) \Leftrightarrow ii)$. Therefore the acceptance of a recognizable language by some non-deterministic finite asynchronous cellular Büchi-automaton can be viewed as a corollary. But for this fact a much simpler and direct construction exists by [50].

8.3 Complex Traces

There is no convenient way to define a concatenation on real traces, in general. To see this consider the following two axioms:

I $a^\omega b^\omega = a^\omega$ for $(a, b) \in D$
II $a^\omega b^\omega \neq a^\omega$ for $(a, b) \in I$

There is an associative operation satisfying I and II if and only if D is transitive. Indeed: If D is transitive, then $\mathbb{M}(\Sigma, I) = \Sigma_1^* \times \cdots \times \Sigma_k^*$ is a direct product of free monoids. The sets Σ_i^∞ are monoids with a right-absorbent multiplication if the left-hand side is an infinite word. This definition yields a concatenation on $\mathbb{R}(\Sigma, D) = \Sigma_1^\infty \times \cdots \times \Sigma_k^\infty$ satisfying I and II. If D is not transitive then consider $a^\omega b^\omega c^\omega$ with $(a, b), (b, c) \in D$, a, b, c pairwise distinct, and $(a, c) \in I$. An operation satisfying I and II is never associative:

$$(a^\omega b^\omega)c^\omega = a^\omega c^\omega \neq a^\omega = a^\omega b^\omega = a^\omega(b^\omega c^\omega)$$

On the other hand, $\mathbb{G}(\Sigma, D)$ is a monoid and we may consider the greatest congruence respecting real parts. The quotient monoid by this congruence is called the monoid $\mathbb{C}(\Sigma, D)$ of *complex traces*. This monoid has a very convenient and concrete characterization.

Theorem 101. *Let $\equiv_{\mathbb{C}}$ be the greatest congruence on $\mathbb{G}(\Sigma, D)$ satisfying $Re(g) = Re(h)$ for all $g \equiv_{\mathbb{C}} h$. Then for all $g, h \in \mathbb{G}(\Sigma, D)$ it holds:*

$$g \equiv_{\mathbb{C}} h \iff (Re(g) = Re(h) \text{ and } D(alphinf(g)) = D(alphinf(h)))\,.$$

Hence, the quotient monoid $\mathbb{G}(\Sigma, D)/\equiv_{\mathbb{C}}$ can be identified with the following set:

$$\mathbb{C}(\Sigma, D) = \{(Re(g), D(\text{alphinf}(g))) \in \mathbb{R}(\Sigma, D) \times \mathcal{P}(\Sigma) \mid g \in \mathbb{G}(\Sigma, D)\}.$$

By definition $\mathbb{C}(\Sigma, D)$ is a monoid and the concatenation is inherited from the concatenation of dependence graphs. It is necessary (otherwise the whole formalism would be rather useless) to have an explicit formula. First, we need a few more notations. Let $g \in \mathbb{G}(\Sigma, D)$ be a dependence graph and $A \subseteq \Sigma$ be a subset. We define $\mu_A(g)$ as the maximal real prefix g being independent of A. If $\leq$ denotes the prefix ordering on $\mathbb{G}(\Sigma, D)$, then it is easy to see that every directed set of real traces has a real least upper bound. Therefore we have

$$\mu_A(g) = \bigsqcup\{p \in \mathbb{M}(\Sigma, I) \mid p \leq g \text{ and } \text{alph}(p) \times A \subseteq I\} \in \mathbb{R}(\Sigma, D).$$

If p is a prefix of g, we denote by $p^{-1}g$ the unique dependence graph such that $p(p^{-1}g) = g$, and by $D(p^{-1}g)$ we denote the set $D(\text{alph}(p^{-1}g))$. We define

$$\begin{aligned}\sigma_A(g) &= D(\text{alphinf}(g)) \cup D(\mu_A(g)^{-1}g)\\ &= \bigcap\{D(p^{-1}g) \mid p \in \mathbb{M}(\Sigma, I), p \leq g, \text{ and } \text{alph}(p) \times A \subseteq I\}.\end{aligned}$$

Note that we always have $D(\text{alphinf}(g)) \subseteq \sigma_A(g)$. Obviously, $D(A) = D(A')$ implies $\mu_A(g) = \mu_{A'}(g)$ and $\sigma_A(g) = \sigma_{A'}(g)$ for all $g \in \mathbb{G}(\Sigma, D)$. The following formula yields the concatenation of complex traces:

$$(r, D(A)) \cdot (s, D(B)) = (r\mu_A(s), \sigma_A(s) \cup D(A \cup B)).$$

The formula above is fundamental for the calculus on complex traces. Once one has defined when a complex trace is connected (there are several suitable choices, take any of them), it is straightforward to define the notion of rational, star-connected, and c-rational complex trace languages. We can go even further and define an α-product of a trace language for every ordinal α. Details are left to the reader.

There are also many equivalent definitions of recognizable complex trace languages. We content with the following

Definition 102. A complex trace language $L \subseteq \mathbb{C}(\Sigma, D)$ is called *recognizable*, if the language of real traces

$$L_A = \{r \in \mathbb{R}(\Sigma, D) \mid (r, D(A)) \in L\}$$

is recognizable for all $A \subseteq \Sigma$.

Example 17. Let $D = \Sigma \times \Sigma$ be full. Then $\mathbb{C}(\Sigma, D) = \Sigma^\infty$. For $L \subseteq \Sigma^\infty$ we have $L_\emptyset = L \cap \Sigma^*$ and $L_\Sigma = L \cap \Sigma^\omega$. The language L is recognizable if both $L_\emptyset \subseteq \Sigma^*$ and $L_\Sigma \subseteq \Sigma^\omega$ are recognizable in the classical sense.

Theorem 103. *Let $L \subseteq \mathbb{C}(\Sigma, D)$ be a complex trace language. Then the following assertions are equivalent:*

i) L is recognizable.
ii) L is star-connected.
iii) L is c-rational.

8.4 Topological Properties and the Domain of Approximating Traces

Consider the following two functions from $\mathbb{M}(\Sigma, I) \times \mathbb{M}(\Sigma, I)$ to $\mathbb{N} \cup \{\infty\}$:

$$\ell_{\mathbb{R}}(u,v) = \sup\{n \in \mathbb{N} \mid \forall p \in \mathbb{M}(\Sigma, I), |p| \leq n : p \leq u \Leftrightarrow p \leq v\}$$
$$\ell_{\mathbb{C}}(u,v) = \sup\{n \in \mathbb{N} \mid \forall p \in \mathbb{M}(\Sigma, I), |p| < n : D(p^{-1}u) = D(p^{-1}v)\}$$

Note that $D(p^{-1}u) \subseteq \Sigma$ is defined only if $p \leq u$. Thus, the equation $D(p^{-1}u) = D(p^{-1}v)$ means that either p is a prefix of both traces u and v or of none. The functions $\ell_{\mathbb{R}}$ and $\ell_{\mathbb{C}}$ yield two ultra-metrics on $\mathbb{M}(\Sigma, I)$:

$$d_{\mathbb{R}}(u,v) = 2^{-\ell_R(u,v)}$$
$$d_{\mathbb{C}}(u,v) = 2^{-\ell_C(u,v)}.$$

Theorem 104. *The completion of the metric space* $(\mathbb{M}(\Sigma, I), d_{\mathbb{R}})$ *is* $\mathbb{R}(\Sigma, D)$*; the completion of* $(\mathbb{M}(\Sigma, I), d_{\mathbb{C}})$ *is* $\mathbb{C}(\Sigma, D)$*. Both spaces are compact and totally disconnected, containing the set of finite traces* $\mathbb{M}(\Sigma, I)$ *as an open and discrete subspace.*

The concatenation of $(\mathbb{C}(\Sigma, D), d_{\mathbb{C}})$ *is uniformly continuous; its continuous extension to* $\mathbb{C}(\Sigma, D)$ *coincides with the concatenation defined explicitly above. The concatenation of* $(\mathbb{M}(\Sigma, I), d_{\mathbb{R}})$ *is uniformly continuous if and only if* D *is transitive (if and only if* $d_{\mathbb{R}}$ *and* $d_{\mathbb{C}}$ *are equivalent metrics).*

As it is well-known from the sequential calculus, the concatenation is not monotone with respect to the prefix ordering, i.e., $a \leq ab$, but we do not have $ac \leq abc$, in general. In the sequential calculus the solution is to introduce a special symbol for termination. Here, we provide some alphabetic information about the set of actions which can be executed in parallel before waiting of termination. Formally, we add to a (finite) trace a second component of the form $D(A)$ with $A \subseteq \Sigma$. The semantics is that before termination only actions $a \in \Sigma$ may occur such that $D(a) \subseteq D(A)$. Therefore the semantics of $D(A) = \Sigma$ is that everything is possible; the semantics of $D(A) = \emptyset$ is explicit termination. Note that for the full dependency $D = \Sigma \times \Sigma$ the whole information reduces to explicit termination, since there are no other sets $D(A)$ than Σ or $\emptyset$.

We define a partially ordered monoid $\mathbb{M}_{\delta}(\Sigma, I)$ of finite *approximating δ-traces* (the sign δ refers to *dependence information*) as follows.

$$\mathbb{M}_{\delta}(\Sigma, D) = \{(u, D(A)) \mid u \in \mathbb{M}(\Sigma, I), A \subseteq \Sigma\}.$$

It is the monoid with the concatenation

$$(u, D(A)) \cdot (v, D(B)) = (u\mu_A(v), \sigma_A(v) \cup D(A \cup B)).$$

The approximation ordering $\sqsubseteq$ is defined by

$$(u, D(A)) \sqsubseteq (v, D(B)) \iff u \leq v \text{ and } D(u^{-1}v) \cup D(B) \subseteq D(A).$$

Example 18.

$$(u, \emptyset)(v, D(B)) = (uv, D(B)),$$
$$(u, \Sigma)(v, D(B)) = (u, \Sigma),$$
$$(1, \Sigma) \sqsubseteq (u, D(A \cup \mathrm{alph}(v))) \sqsubseteq (uv, D(A \setminus \mathrm{alph}(v))) \sqsubseteq (uv, \emptyset),$$

$(1, \emptyset)$ is the neutral element, and $(1, \Sigma)$ is the bottom element being right-absorbent.

It is clear that $\mathbb{M}(\Sigma, I)$ is a submonoid of $\mathbb{M}_\delta(\Sigma, D)$. The canonical embedding is $u \longmapsto (u, \emptyset)$. If $D = \Sigma \times \Sigma$ is full, then using a new symbol $\surd$ for explicit termination we have a canonical identification $\mathbb{M}_\delta(\Sigma, D) = \Sigma^* \cup \Sigma^* \surd$. We identify $u \in \Sigma^*$ with $(u, \emptyset)$ and $u\surd$ with (u, Σ).

The set of *δ-traces* $\mathbb{F}(\Sigma, D)$ is defined by

$$\mathbb{F}_\delta(\Sigma, D) = \{(u, D(A)) \mid u \in \mathbb{R}(\Sigma, D), \mathrm{alphinf}(u) \subseteq A \subseteq \Sigma\}.$$

The same formulae for the concatenation and the approximation ordering apply. The concatenation is well-defined since we demand $\mathrm{alphinf}(u) \subseteq A$ for $(u, D(A)) \in \mathbb{F}_\delta(\Sigma, D)$.

The various spaces we have considered so far can be put into a commuting diagram of inclusions.

$$\begin{tikzcd}
\mathbb{M}(\Sigma, I) \arrow[rr, hook] \arrow[d, hook] & & \mathbb{M}_\delta(\Sigma, D) \arrow[d, hook] \\
\mathbb{R}(\Sigma, D) \arrow[r, hook] & \mathbb{C}(\Sigma, D) \arrow[r, hook] & \mathbb{F}_\delta(\Sigma, D)
\end{tikzcd}$$

The following theorem states that $\mathbb{F}_\delta(\Sigma, D)$ with the approximation ordering is in fact a good semantic domain.

Theorem 105. *The set of δ-traces $\mathbb{F}_\delta(\Sigma, D)$ with the ordering $\sqsubseteq$ is a complete partial order (CPO). The concatenation is continuous in both arguments, i.e.,*

$$(\bigsqcup X) \cdot (\bigsqcup Y) = \bigsqcup (X \cdot Y)$$

for all directed subsets $X, Y \subseteq \mathbb{F}_\delta(\Sigma, D)$. The CPO $\mathbb{F}_\delta(\Sigma, D)$ is coherently complete and algebraic. In particular, it is a Scott domain. The subset of compact (or finite) elements is $\mathbb{M}_\delta(\Sigma, D)$.

For $\mathbb{F}_\delta(\Sigma, D)$ the *Scott-topology* and its refinement the *Lawson-topology* are defined. The Lawson-topology is induced by the extension of the metric $d_\mathbb{C}$ from complex traces to δ-traces. Formally $d_\delta(x, y) = 2^{-\ell_\delta(x,y)}$ and

$$\ell_\delta((u, D(A)), (v, D(B))) = \sup\{n \in \mathbb{N} \mid \forall p \in \mathbb{M}(\Sigma, I), |p| < n : \\ D(p^{-1}u) \cup D(A) = D(p^{-1}v) \cup D(B)\}.$$

By general facts about Lawson-topology [54] it yields in the present case a compact and totally disconnected space. (In particular $\mathbb{F}_\delta(\Sigma, D)$ is a complete ultra-metric space.) Every compact and totally disconnected space is a projective limit. We make this explicit. For each $n \geq 0$ and a trace $u \in \mathbb{M}(\Sigma, I)$ let $u[n] = \bigsqcup\{p \leq u \mid |p| \leq n\}$, i.e., $u[n]$ is the least upper bound of the prefixes of length n in u. Using this notations we define finite and aperiodic monoids M_n and F_n as follows:

$$\begin{aligned} M_n &= \{(u[n], D((u[n])^{-1}u)) \mid u \in \mathbb{M}(\Sigma, I)\} \\ F_n &= \{(u[n], D(A)) \mid u \in \mathbb{M}(\Sigma, I), A \subset \Sigma\} \end{aligned}$$

The multiplication is given in both monoids by the same formula:

$$(u, D(A)) \cdot (v, D(B)) = ((uv)[n], D(((uv)[n])^{-1}uv) \cup D(A \cup B)).$$

It is an exercise to verify that he formula is well-defined. It also shows that M_n is a submonoid of F_n and that for all $n \geq 0$ we have a canonical homomorphism

$$\psi_n : \mathbb{M}(\Sigma, I) \longrightarrow F_n.$$

The image of ψ_n is exactly M_n. For each $m \geq n$ there are surjective homomorphisms $\psi'_{m,n}$ and $\psi_{m,n}$ such that the following diagram commutes:

$$\begin{tikzcd} \mathbb{M}(\Sigma, I) \arrow[r, "\psi_m"] \arrow[dr, "\psi_n"] & M_m \arrow[r, hook] \arrow[d, "\psi'_{m,n}"] & F_m \arrow[d, "\psi_{m,n}"] \\ & M_n \arrow[r, hook] & F_n \end{tikzcd}$$

It is clear that this leads to projective systems. Recall the definition

$$\begin{aligned} \varprojlim M_n &= \{(x_m) \in \prod_{m \geq 0} M_n \mid \psi'_{m,n}(x_m) = x_n \text{ for all } m \geq n\} \\ \varprojlim F_n &= \{(x_m) \in \prod_{m \geq 0} F_n \mid \psi_{m,n}(x_m) = x_n \text{ for all } m \geq n\} \end{aligned}$$

Being finite we endow the sets M_n, F_n with the discrete topology. The projective limits are then compact (by Tychonov's Theorem) and totally disconnected. The are topological monoids and the multiplication is uniformly continuous. We have just constructed the corresponding monoids of complex and δ-traces with the Lawson-topology. The final theorem reflects most of what we have done so far for complex and for δ-traces in two lines.

Theorem 106. *We have the following equalities of topological monoids:*

$$\begin{aligned} \mathbb{C}(\Sigma, D) &= \varprojlim M_n \\ \mathbb{F}_\delta(\Sigma, D) &= \varprojlim F_n. \end{aligned}$$

8.5 Bibliographical Remarks and Further Reading

The theory of infinite traces has its origins in the mid eighties. Flé and Roucairol [44, 45] considered the problem of serializability of iterated transactions in data bases. Their definition of an infinite trace is equivalent to our definition of an infinite real trace. Best and Devillers [9] defined the behavior of Petri nets by an equivalence relation on Σ^∞. This equivalence relation yields real traces. A definition of a real trace as a prefix-closed and directed subset of real traces and its characterization by dependence graphs is given in a survey by Mazurkiewicz [69]. This characterization is the basis for an investigation of the poset properties, as e.g. studied by Gastin and Rozoy in [52]. Kwiatkowska [65] introduced real traces as a suitable model for distributed systems to reason about fairness, liveness, and safety properties. She also showed that the set $\mathbb{R}(\Sigma, D)$ with the prefix-metric $d_{\mathbb{R}}$ yields a compact ultra-metric space. Bonizzoni, Mauri, and Pighizzini gave a Foata normal form for infinite traces [10].

The theory of recognizable real trace languages has been initiated by Gastin [47, 48]. The generalization of the Kleene-Büchi-Ochmański-Theorem, Thm. 98, to real traces is due to Gastin, Petit, and Zielonka [51]. (It has been generalized to complex trace languages in [30].)

Diekert and Muscholl gave a construction for a deterministic asynchronous Muller automaton accepting a given recognizable real trace language, see [31]. Together with [42] the results mentioned above provide a satisfactory and rather complete generalization of regular infinitary word languages to real traces.

A basic difference between Σ^∞ and $\mathbb{R}(\Sigma, D)$ is however that a natural associative operation of concatenation cannot be defined for real traces, in general. This led Diekert to the notion of complex trace [27]. The results about rational and recognizable complex trace languages have been established in [30]. A theory of δ-traces as a natural generalization of this concept has been proposed in [29].

There are other generalizations of the theory of finite traces we have not dealt with in this chapter. For example, consider asymmetric (in-)dependencies leading to the notion of semi-trace. This is the theory of *semi-commutation* initiated by Clerbout and Latteux [15, 16, 17, 18]. An overview on semi-commutations has been given by Clerbout, Latteux, and Roos in [34, Chap. 12]. A generalization of semi-traces based on the pomset model of Pratt [83] has been proposed in [28]. It combines main features from trace theory (e.g., an embedding theorem yielding a canonical representation as a tuple of words and a synchronization operator) with certain aspects of dynamic concurrency. It leads to a semantics of Petri nets (place-transition nets) reflecting a modular construction.

Another approach to express dynamic behavior is due to investigations of Panangaden and Stark [80] who introduced the notion of trace automata. Close to this is the notion of *concurrent automata* being transition systems where the dependence relation may vary with the state and their monoids of computations due to Droste [35]. Many important theorems of trace theory have been generalized to these monoids, see [38].

References

1. IJsbrand Jan Aalbersberg and Hendrik Jan Hoogeboom. Characterizations of the decidability of some problems for regular trace languages. *Mathematical Systems Theory*, 22:1–19, 1989.
2. IJsbrand Jan Aalbersberg and Emo Welzl. Trace languages defined by regular string languages. *R.A.I.R.O. — Informatique Théorique et Applications*, 20:103–119, 1986.
3. Douglas Adams. *The Hitch Hiker's Guide to the Galaxy.* Pan Books, London, 1979.
4. Anatolij V. Anisimov and Donald E. Knuth. Inhomogeneous sorting. *International Journal of Computer and Information Sciences*, 8:255–260, 1979.
5. Jean Berstel and Dominique Perrin. *Theory of Codes.* Pure and Applied Mathematics; 117. Academic Press, Orlando, Florida, 1985.
6. Alberto Bertoni, Giancarlo Mauri, and Nicoletta Sabadini. A hierarchy of regular trace languages and some combinatorial applications. In A. Ballester, D. Cardus, and E. Trillas, editors, *Proceedings of the 2nd World Conf. on Mathematics at the Service of Man, Las Palmas (Canary Island) Spain*, pages 146–153. Universidad Politecnica de Las Palmas, 1982.
7. Alberto Bertoni, Giancarlo Mauri, and Nicoletta Sabadini. Concurrency and commutativity. Technical report, Instituto di Cibernetica, Università di Milano, 1983. Presented at the Workshop on Petri nets, Varenna (Italy), 1982.
8. Alberto Bertoni, Giancarlo Mauri, and Nicoletta Sabadini. Unambiguous regular trace languages. In G. Demetrovics, J. Katona and Arto Salomaa, editors, *Proceedings of the Coll. on Algebra, Combinatorics and Logic in Computer Science*, volume 42 of *Colloquia Mathematica Soc. J. Bolyai*, pages 113–123. North Holland, Amsterdam, 1985.
9. Eike Best and Raymond Devillers. Sequential and concurrent behaviour in Petri net theory. *Theoretical Computer Science*, 55:87–136, 1987.
10. Paola Bonizzoni, Giancarlo Mauri, and Giovanni Pighizzini. About infinite traces. In Volker Diekert, editor, *Proceedings of the ASMICS workshop Free Partially Commutative Monoids, Kochel am See, Oktober 1989*, Report TUM-I9002, Technical University of Munich, pages 1–10, 1990.
11. Véronique Bruyère and Clelia De Felice. Trace codings. In E.W. Mayr and C. Puech, editors, *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95), 1995*, number 900 in Lecture Notes in Computer Science, pages 373–384, Berlin-Heidelberg-New York, 1995. Springer.
12. Véronique Bruyère, Clelia De Felice, and Giovanna Guaiana. Coding with traces. In P. Enjalbert, E.W. Mayr, and K. W. Wagner, editors, *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science (STACS'94), 1994*, number 775 in Lecture Notes in Computer Science, pages 353–364, Berlin-Heidelberg-New York, 1994. Springer.
13. Pierre Cartier and Dominique Foata. *Problèmes combinatoires de commutation et réarrangements.* Number 85 in Lecture Notes in Mathematics. Springer, Berlin-Heidelberg-New York, 1969.
14. Marek Chrobak and Wojciech Rytter. Unique decipherability for partially commutative alphabets. *Fundamenta Informaticae*, X:323–336, 1987.
15. Mireille Clerbout. *Commutations Partielles et Familles de Langages.* Thèse, Université des Sciences et Technologies de Lille (France), 1984.

16. Mireille Clerbout and Michel Latteux. Partial commutations and faithful rational transductions. *Theoretical Computer Science*, 34:241–254, 1984.
17. Mireille Clerbout and Michel Latteux. Semi-Commutations. *Information and Computation*, 73:59–74, 1987.
18. Mireille Clerbout, Michel Latteux, and Yves Roos. Decomposition of partial commutations. In M. S. Paterson, editor, *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP'90), Warwick (England) 1990*, number 443 in Lecture Notes in Computer Science, pages 501–511, Berlin-Heidelberg-New York, 1990. Springer.
19. Robert Cori and Yves Métivier. Recognizable subsets of some partially abelian monoids. *Theoretical Computer Science*, 35:179–189, 1985.
20. Robert Cori, Yves Métivier, and Wiesław Zielonka. Asynchronous mappings and asynchronous cellular automata. *Information and Computation*, 106:159–202, 1993.
21. Robert Cori and Dominique Perrin. Automates et commutations partielles. *R.A.I.R.O. — Informatique Théorique et Applications*, 19:21–32, 1985.
22. Bruno Courcelle. The monadic second-order logic of graphs X: linear orderings. *Theoretical Computer Science*, 1996. to appear.
23. Volker Diekert. Transitive orientations, Möbius functions and complete semi-Thue systems for free partially commutative monoids. In T. Lepistö et al., editors, *Proceedings of the 15th International Colloquium on Automata, Languages and Programming (ICALP'88), Tampere (Finland) 1988*, number 317 in Lecture Notes in Computer Science, pages 176–187, Berlin-Heidelberg-New York, 1988. Springer.
24. Volker Diekert. *Combinatorics on Traces*. Number 454 in Lecture Notes in Computer Science. Springer, Berlin-Heidelberg-New York, 1990.
25. Volker Diekert. Research topics in the theory of free partially commutative monoids. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 40:479–491, Feb 1990.
26. Volker Diekert. Möbius functions and confluent semi-commutations. *Theoretical Computer Science*, 108:25–43, 1993.
27. Volker Diekert. On the concatenation of infinite traces. *Theoretical Computer Science*, 113:35–54, 1993. Special issue STACS'91.
28. Volker Diekert. A partial trace semantics for Petri nets. *Theoretical Computer Science*, 134:87–105, 1994. Special issue of ICWLC 92, Kyoto (Japan).
29. Volker Diekert and Paul Gastin. A domain for concurrent termination: A generalization of Mazurkiewicz traces. In Zoltán Fülöp and Ferenc Gécseg, editors, *Proceedings of the 22nd International Colloquium on Automata, Languages and Programming (ICALP'95), Szeged (Hungary) 1995*, number 944 in Lecture Notes in Computer Science, pages 15–26. Springer, 1995.
30. Volker Diekert, Paul Gastin, and Antoine Petit. Rational and recognizable complex trace languages. *Information and Computation*, 116:134–153, 1995.
31. Volker Diekert and Anca Muscholl. Deterministic asynchronous automata for infinite traces. *Acta Informatica*, 31:379–397, 1994. A preliminary version was presented at STACS'93, Lecture Notes in Computer Science 665 (1993).
32. Volker Diekert and Anca Muscholl. A note on Métivier's construction of asynchronous automata for triangulated graphs. *Fundamenta Informaticae*, to appear 1996. Special issue on Formal Language Theory.
33. Volker Diekert, Anca Muscholl, and Klaus Reinhardt. On codings of traces. In E.W. Mayr and C. Puech, editors, *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95), 1995*, number 900 in

Lecture Notes in Computer Science, pages 385–396, Berlin-Heidelberg-New York, 1995. Springer.

34. Volker Diekert and Grzegorz Rozenberg, editors. *The Book of Traces.* World Scientific, Singapore, 1995.
35. Manfred Droste. Concurrent automata and domains. *International Journal of Foundations of Computer Science*, 3:389–418, 1992.
36. Manfred Droste. A Kleene Theorem for recognizable languages over concurrency monoids. In Serge Abiteboul and Eli Shamir, editors, *Proceedings of the 21st International Colloquium on Automata, Languages and Programming (ICALP'94), Jerusalem (Israel) 1994*, number 820 in Lecture Notes in Computer Science, pages 388–398, 1994.
37. Manfred Droste. Aperiodic languages over concurrency monoids. 1995. submitted.
38. Manfred Droste. Recognizable languages in concurrency monoids. *To appear in Theoretical Computer Science*, 1995.
39. Manfred Droste and Dietrich Kuske. Languages and logical definability in concurrency monoids. Presented at Computer Science Logic, Paderborn, 1995.
40. Christine Duboc. Commutations dans les monoïdes libres: un cadre théorique pour l'étude du parallelisme. Thèse, Faculté des Sciences de l'Université de Rouen, 1986.
41. Werner Ebinger. *Charakterisierung von Sprachklassen unendlicher Spuren durch Logiken.* Dissertation, Institut für Informatik, Universität Stuttgart, 1994.
42. Werner Ebinger and Anca Muscholl. Logical definability on infinite traces. In Andrzej Lingas, Rolf Karlsson, and Svante Carlsson, editors, *Proceedings of the 20th International Colloquium on Automata, Languages and Programming (ICALP'93), Lund (Sweden) 1993*, number 700 in Lecture Notes in Computer Science, pages 335–346, Berlin-Heidelberg-New York, 1993. Springer.
43. Samuel Eilenberg and Marcel Paul Schützenberger. Rational sets in commutative monoids. *Journal of Algebra*, 13:173–191, 1969.
44. Marie-Paule Flé and Gérard Roucairol. On serializability of iterated transactions. In *Proceedings of the 15th ACM SIGACT-SIGOPS Symp. on Princ. of Distrib. Comp., Ottawa (1982)*, pages 194 – 200, 1982.
45. Marie-Paule Flé and Gérard Roucairol. Fair serializability of iterated transactions using fifo-nets. In Grzegorz Rozenberg, editor, *Advances in Petri Nets*, number 188 in Lecture Notes in Computer Science, pages 154–168. Springer, Berlin-Heidelberg-New York, 1985.
46. Michel Fliess. Matrices de Hankel. *J. Math. Pures et Appl.*, 53:197–224, 1974.
47. Paul Gastin. Infinite traces. In I. Guessarian, editor, *Proceedings of the Spring School of Theoretical Computer Science on Semantics of Systems of Concurrent Processes*, number 469 in Lecture Notes in Computer Science, pages 277–308, Berlin-Heidelberg-New York, 1990. Springer.
48. Paul Gastin. Recognizable and rational trace languages of finite and infinite traces. In Christian Choffrut et al., editors, *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science (STACS'91), Hamburg 1991*, number 480 in Lecture Notes in Computer Science, pages 89–104, Berlin-Heidelberg-New York, 1991. Springer.
49. Paul Gastin, Edward Ochmański, Antoine Petit, and Brigitte Rozoy. Decidability of the Star problem in $A^* \times \{b\}^*$. *Information Processing Letters*, 44:65–71, 1992.
50. Paul Gastin and Antoine Petit. Asynchronous automata for infinite traces. In W. Kuich, editor, *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP'92), Vienna (Austria) 1992*, number 623 in

Lecture Notes in Computer Science, pages 583–594, Berlin-Heidelberg-New York, 1992. Springer.

51. Paul Gastin, Antoine Petit, and Wiesław Zielonka. An extension of Kleene's and Ochmański's theorems to infinite traces. *Theoretical Computer Science*, 125:167–204, 1994. A preliminary version was presented at ICALP'91, Lecture Notes in Computer Science 510 (1991).
52. Paul Gastin and Brigitte Rozoy. The poset of infinitary traces. *Theoretical Computer Science*, 120:101–121, 1993.
53. Alan Gibbons and Wojciech Rytter. On the decidability of some problems about rational subsets of free partially commutative monoids. *Theoretical Computer Science*, 48:329–337, 1986.
54. Gerhard Gierz, Karl Heinrich Hofmann, Klaus Keimel, Jimmie D. Lawson, Michael W. Mislove, and Dana S. Scott. *A Compendium of Continuous Lattices.* Springer, Berlin-Heidelberg-New York, 1980.
55. Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.
56. Martin C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.
57. Sheila A. Greibach. The undecidability of the ambiguity problem for minimal linear grammars. *Information and Control*, 6:119–125, 1963.
58. Giovanna Guaiana, Antonio Restivo, and Sergio Salemi. Star-free trace languages. *Theoretical Computer Science*, 97:301–311, 1992. A preliminary version was presented at STACS'91, Lecture Notes in Computer Science 480 (1991).
59. Kosaburo Hashiguchi. Limitedness theorem on finite automata with distance functions. *Journal of Computer and System Sciences*, 24:233–244, 1982.
60. Kosaburo Hashiguchi. Recognizable closures and submonoids of free partially commutative monoids. *Theoretical Computer Science*, 86:233–241, 1991.
61. Hendrik Jan Hoogeboom and Anca Muscholl. The code problem for traces – improving the boundaries. Submitted for publication., 1995.
62. Günter Hotz and Volker Claus. *Automatentheorie und Formale Sprachen, Band III.* Bibliographisches Institut, Mannheim, 1972.
63. Oscar H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the Association of Computing Machinery*, 25(1):116–133, 1978.
64. Robert M. Keller. Parallel program schemata and maximal parallelism I. Fundamental results. *Journal of the Association of Computing Machinery*, 20(3):514–537, 1973.
65. Marta Z. Kwiatkowska. A metric for traces. *Information Processing Letters*, 35:129–135, 1990.
66. Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the Association of Computing Machinery*, 21:558–564, 1978.
67. Gennadií Semjonovich Makanin. The problem of solvability of equations in free semigroups. *Math. USSR Izvestiya*, 21:483–546, 1983.
68. Antoni Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.
69. Antoni Mazurkiewicz. Trace theory. In W. Brauer et al., editors, *Petri Nets, Applications and Relationship to other Models of Concurrency*, number 255 in Lecture Notes in Computer Science, pages 279–324, Berlin-Heidelberg-New York, 1987. Springer.
70. Yves Métivier. On recognizable subsets of free partially commutative monoids. In L. Kott, editor, *Proceedings of the 13th International Colloquium on Automata,*

Languages and Programming (ICALP'86), Rennes (France) 1986, number 226 in Lecture Notes in Computer Science, pages 254–264, Berlin-Heidelberg-New York, 1986. Springer.
71. Yves Métivier. Une condition suffisante de reconnaissabilité dans un monoïde partiellement commutatif. *R.A.I.R.O. — Informatique Théorique et Applications*, 20:121–127, 1986.
72. Yves Métivier. An algorithm for computing asynchronous automata in the case of acyclic non-commutation graph. In Th. Ottmann, editor, *Proceedings of the 14th International Colloquium on Automata, Languages and Programming (ICALP'87), Karlsruhe (FRG) 1987*, number 267 in Lecture Notes in Computer Science, pages 226–236, Berlin-Heidelberg-New York, 1987. Springer.
73. Yves Métivier. Contribution à l'étude des monoïdes de commutations. Thèse d'état, 1987. Université Bordeaux I.
74. Yves Métivier and Edward Ochmański. On lexicographic semi-commutations. *Information Processing Letters*, 26:55–59, 1987/88.
75. Yves Métivier, Gwénaël Richomme, and Pierre-André Wacrenier. Computing the closure of sets of words under partial commutations. In Zoltán Fülöp and Ferenc Gécseg, editors, *Proceedings of the 22nd International Colloquium on Automata, Languages and Programming (ICALP'95), Szeged (Hungary) 1995*, number 944 in Lecture Notes in Computer Science, pages 75–86. Springer, 1995.
76. Anca Muscholl and Holger Petersen. A note on the commutative closure of star-free languages, 1996. To appear in Information Processing Letters.
77. Edward Ochmański. Regular behaviour of concurrent systems. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 27:56–67, Oct 1985.
78. Edward Ochmański. On morphisms of trace monoids. In Robert Cori and M. Wirsing, editors, *Proceedings of the 5th Annual Symposium on Theoretical Aspects of Computer Science (STACS'88)*, number 294 in Lecture Notes in Computer Science, pages 346–355, Berlin-Heidelberg-New York, 1988. Springer.
79. Friedrich Otto. Finite canonical rewriting systems for congruences generated by concurrency relations. *Mathematical Systems Theory*, 20:253–260, 1987.
80. Prakas Panangaden and Eugene W. Stark. Computations, residuals and the power of indeterminacy. In Timo Lepistö et al., editors, *Proceedings of the 15th International Colloquium on Automata, Languages and Programming (ICALP'88), Tampere (Finland) 1988*, number 317 in Lecture Notes in Computer Science, pages 439–454, Berlin-Heidelberg-New York, 1988. Springer.
81. Dominique Perrin. Words over a partially commutative alphabet. In Alberto Apostolico, editor, *Combinatorial Algorithms on Words*, volume F12 of *NATO-ASI Series*, pages 329–340. Springer, Berlin-Heidelberg-New York, 1986.
82. Giovanni Pighizzini. Asynchronous automata versus asynchronous cellular automata. *Theoretical Computer Science*, 132:179–207, 1994.
83. Vaughan R. Pratt. Modelling concurrency with partial orders. *International Journal of Parallel Programming*, 15(1):33–71, 1986.
84. Gwénaël Richomme. Some trace monoids where both the Star Problem and the Finite Power Property Problem are decidable. In I. Privara et al., editors, *Proceedings of the 19th Symposium on Mathematical Foundations of Computer Science (MFCS'94), Košice (Slovakia) 1994*, number 841 in Lecture Notes in Computer Science, pages 577–586, Berlin-Heidelberg-New York, 1994. Springer.
85. Jaques Sakarovitch. On regular trace languages. *Theoretical Computer Science*, 52:59–75, 1987.

86. Jaques Sakarovitch. The “last” decision problem for rational trace languages. In Imre Simon, editor, *Proceedings of the 1st Latin American Symposium on Theoretical Informatics (LATIN'92)*, number 583 in Lecture Notes in Computer Science, pages 460–473, Berlin-Heidelberg-New York, 1992. Springer.
87. Wolfgang Thomas. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences*, 25:360–376, 1982.
88. Wolfgang Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 4, pages 133–191. Elsevier Science Publishers B. V., 1990.
89. Wolfgang Thomas. On logical definability of trace languages. In Volker Diekert, editor, *Proceedings of a workshop of the ESPRIT Basic Research Action No 3166: Algebraic and Syntactic Methods in Computer Science (ASMICS), Kochel am See, Bavaria, FRG (1989)*, Report TUM-I9002, Technical University of Munich, pages 172–182, 1990.
90. Wiesław Zielonka. Notes on finite asynchronous automata. *R.A.I.R.O. — Informatique Théorique et Applications*, 21:99–135, 1987.
91. Wiesław Zielonka. Safe executions of recognizable trace languages by asynchronous automata. In A. R. Mayer et al., editors, *Proceedings of the Symposium on Logical Foundations of Computer Science, Logic at Botik '89, Pereslavl-Zalessky (USSR) 1989*, number 363 in Lecture Notes in Computer Science, pages 278–289, Berlin-Heidelberg-New York, 1989. Springer.

Index