Theory of Computing Systems manuscript No.
(will be inserted by the editor)

On Boolean closed full trios and rational Kripke frames

Georg Zetzsche · Dietrich Kuske · Markus Lohrey

the date of receipt and acceptance should be inserted later

Abstract We study what languages can be constructed from a non-regular language L using Boolean operations and synchronous or non-synchronous rational transductions. If all rational transductions are allowed, one can construct the whole arithmetical hierarchy relative to L. In the case of synchronous rational transductions, we present non-regular languages that allow constructing languages arbitrarily high in the arithmetical hierarchy and we present non-regular languages that allow constructing only recursive languages.

A consequence of the results is that aside from the regular languages, no full trio generated by a single language is closed under complementation. Another consequence is that there is a fixed rational Kripke frame such that assigning an arbitrary non-regular language to some variable allows the definition of any language from the arithmetical hierarchy in the corresponding Kripke structure using multimodal logic.

1 Introduction

The study of closure properties of language classes has a long tradition in automata and language theory; it can be traced back to the introduction of regular languages [25]. One reason for this interest is that they have numerous applications. This holds in particular if one considers classes of languages that have finite representations (e.g., regular languages can be represented by nondeterministic finite automata). In this case, the closure properties can even be effective. For instance, in the case of regular languages, automata constructions for various

Georg Zetzsche
Fachbereich Informatik, Technische Universität Kaiserslautern
E-mail: zetzsche@cs.uni-kl.de

Dietrich Kuske
Institut für Theoretische Informatik, Technische Universität Ilmenau
E-mail: dietrich.kuske@tu-ilmenau.de

Markus Lohrey
Department für Elektrotechnik und Informatik, Universität Siegen
E-mail: lohrey@eti.uni-siegen.de

operations (Boolean operations, concatenation, Kleene star, homomorphic images and preimages) are available. Closure properties can be also used to show that a language L does not belong to a language class by constructing from L a language that is known to be outside of the class. Moreover, closure properties also often serve as a way to describe language classes without reference to concrete generating or accepting devices: In many cases, a language class can be described as the smallest class of languages that possesses a given collection of closure properties and contains certain languages. For instance, by the theorem of Chomsky and Schützenberger [8], the class of context free languages is the smallest class of languages that contains the Dyck languages and that is closed under intersection with regular languages and homomorphisms. Similar descriptions are available for various types of counter languages [18,23], the arithmetical hierarchy [5], the recursively enumerable languages [21,13] and many others [34].

In this paper, we are concerned with language classes that are closed under Boolean operations and under rational transductions, i.e., we consider Boolean closed full trios [3]. This particular combination of closure properties is interesting for several reasons:

1. Automatic structures are an important class of infinite structures in algorithmic model theory. A relational structure is automatic if its universe is a regular language and every relation is synchronous rational (i.e., accepted by a two-head automaton whose heads move synchronously). The first-order theory of every automatic structure is decidable [24]. To prove this fundamental result, one uses that
   - regular languages can be represented by finite automata ("finite representation"),
   - using this representation, the class of regular languages is effectively closed under Boolean operations and images and preimages of length-preserving morphisms[1] ("effective closure"), and
   - finite automata have a decidable emptiness problem ("decidable emptiness").

   The closure properties follow from the fact that the class of regular languages is a Boolean closed full trio. Thus, identifying a Boolean closed full trio $\mathcal{C}$ beyond the regular languages that enjoys finite representations, effective closure, and decidable emptiness would mean that *$\mathcal{C}$-automatic structures* (i.e., relational structures with universe in $\mathcal{C}$ and synchronous rational relations) guarantee decidability of the first-order theory. Formal language theory has yielded a wealth of union-closed full trios (i.e., language classes that are closed under union and rational transductions) with finite representation, effective closure and decidable emptiness (see, for example, [3,11,23]). It therefore seems prudent to seek Boolean closed full trios among them.

   Suppose the language class $\mathcal{C}$ has finite representation, effective closure, and decidable emptiness. Then also the universality problem (given a language $L \in \mathcal{C}$, does L equal X^*?) and, more generally, the regular inclusion problem (given regular R and $L \in \mathcal{C}$, does L include R?) are decidable.
2. Bekker and Goranko [2] investigated rational Kripke frames and the model-checking problem for multimodal logic on rational Kripke frames (see [4] for

[1] These latter closure properties are needed in order to realize projection and cylindrification of relations.

more details on modal logic). A Kripke frame (which is basically an edge labelled graph) is rational if the set of worlds forms a regular language and the visibility relations are given by rational transductions. Then the languages definable by multimodal logic are always confined to the Boolean closed full trio generated by the values (that is, languages) assigned to the variables. This was observed by Bekker and Goranko [2] and then used to show that the model checking problem for multimodal logic and rational Kripke frames is decidable if all variables are assigned regular languages. As in the case of automatic structures, larger language classes with finite representation, effective closure, and decidable emptiness would allow extending this result.

3. The principal full trio generated by the language L is the class of images of L under arbitrary rational transductions. Examples of principal full trios are the context-free languages, languages accepted by multicounter automata (for a bounded number of counters and blind, partially blind, or with zero test [18]), and the languages accepted by valence automata over a finitely generated monoid [14]. See [3] for more examples.
   Since such principal full trios are always closed under union, the closure of a principal full trio under complementation is equivalent to the class being a Boolean closed full trio.

Hence, the question arises whether there are language classes beyond the regular languages that enjoy these closure properties and still admit decision procedures for simple properties such as emptiness. This work answers this question in an extremely negative way. Our first main result (Theorem 3.1) states that every Boolean closed full trio that contains a *non-regular language* already includes the whole arithmetical hierarchy relative to this language (and therefore in particular all recursively enumerable languages) and thus loses virtually all decidability properties. Our result means that in a full trio beyond the regular languages, *virtually no decidability property can coexist with Boolean closure.*

A large number of grammar and automata models are easily seen to exceed the regular languages but stay within the recursively enumerable languages. Hence, Theorem 3.1 also implies that the corresponding language classes are never Boolean closed full trios. We can also conclude that *other than the regular languages, no principal full trio* is closed under complementation.

It might seem to be a consequence of our first main result that there is no class of "$\mathcal{C}$-automatic structures" with decidable first-order theory. However, such a class $\mathcal{C}$ has to be closed under Boolean operations and *synchronous* rational transductions [15], only. In other words, $\mathcal{C}$ need not really be a Boolean closed full trio. Since our proofs make heavy use of asynchronous rational transductions, the question arises whether the situation changes if we use only synchronous rational transductions. In this context, we present as our second main result non-regular languages L (e.g., the language $\{w \in \{a,b\}^* \mid |w|_a = |w|_b\}$) that allow constructing, for every $n \in \mathbb{N}$, a language that is hard for the n-th level of the arithmetical hierarchy (Theorem 4.5). Consequently, there are "simple" non-regular languages L and $\{L\}$-automatic structures with first-order theory arbitrarily high in the arithmetical hierarchy. On the other hand, we also provide examples of non-regular languages that only produce recursive languages.

Coming back to arbitrary rational transductions, it turns out that three fixed rational transductions, together with the Boolean operations, suffice to construct all arithmetical languages from any non-regular language. Therefore, our third main result (Theorem 5.1) states that there is a fixed rational Kripke frame with three modalities such that assigning any non-regular language to a variable allows the definition of every arithmetical language using multimodal logic.

Related work. Other results of a similar spirit on closure properties of language classes have been known for a long time. For example, Hartmanis and Hopcroft [20] proved that every intersection closed full AFL containing $E = \{a^n b^n \mid n \in \mathbb{N}\}$ already includes the recursively enumerable languages (see [17] for similar results). Since every Boolean closed full trio is an intersection closed full AFL, compared to the result of Hartmanis and Hopcroft, we require more closure properties, but in return, we may replace E by an arbitrary non-regular language. See page 8 for more information.

Furthermore, Book [5] has shown that the arithmetical languages constitute the smallest Boolean closed full trio that is closed under homomorphic replication, a generalization of homomorphisms. Hence, our result means that in Book's result one can replace homomorphic replication by containment of a non-regular language.

Seibert [37] has shown that applying projections and Boolean operations to asynchronously recognizable relations (a generalization of rational transductions to arities beyond 2) allows constructing all arithmetical languages and relations over any fixed alphabet. Differently from Seibert, we do not apply Boolean operations to relations, but only to languages. Furthermore, we present a single finite set of rational transductions that allows constructing all arithmetical languages over $\{0,1\}$ from any non-regular language over $\{0,1\}$.

This is an extended version of the conference contribution [26] that did not contain the proofs nor the results on synchronous rational transductions.

2 Preliminaries

For more details on automata and formal languages, the reader can consult [3, 22]. Let Σ be a fixed countable set of abstract symbols, the finite subsets of which are called *alphabets*. Given an alphabet X, the set of words over X is denoted by X^* and the empty word by λ. A *homomorphism* is a mapping $h\colon X^* \to Y^*$ (for alphabets X, Y) such that $h(\lambda) = \lambda$ and $h(uv) = h(u)h(v)$ for all $u, v \in X^*$. Subsets of X^* for alphabets X are called *languages*. For a language L, the smallest alphabet X with $L \subseteq X^*$ is denoted by $\alpha(L)$. The *complement* of L is defined as $\overline{L} = \alpha(L)^* \setminus L$.

For two languages $L, K \subseteq X^*$, we define their *shuffle* as

$$\begin{aligned} L \sqcup\!\sqcup K = \{u_1 v_1 \cdots u_n v_n \mid \quad & u_1, \ldots, u_n, v_1, \ldots, v_n \in X^*, \\ & u_1 \cdots u_n \in L, \quad v_1 \cdots v_n \in K\}. \end{aligned}$$

Let M be a monoid with neutral element 1. An *automaton over M* is a tuple $\mathcal{A} = (Q, M, E, q_0, Q_f)$, in which Q is a finite set of *states*, E is a finite subset of $Q \times M \times Q$ called the set of *edges*, $q_0 \in Q$ is the *initial state*, and $Q_f \subseteq Q$ is the set of *final states*. A *path (from p_0 to p_m)* is a sequence $p_0\, a_1\, p_1\, a_2\, p_2\, \cdots\, a_m\, p_m$ with $(p_i, a_{i+1}, p_{i+1}) \in E$ for all $0 \leq i < m$. Its *label* is $a_1 a_2 \cdots a_m \in M$. In case $m = 0$

we have the empty path from p_0 to p_0 and its label is the neutral element $1 \in M$. If $p_0 = q_0$ is the initial state and $p_m \in Q_f$ is some final state, then this path is *accepting*. The set generated by $\mathcal{A}$ is then

$$[\![\mathcal{A}]\!] = \{a \in M \mid a \text{ is the label of some accepting path in } \mathcal{A}\}.$$

A set $R \subseteq M$ is called *rational* if there is some automaton $\mathcal{A}$ over M with $R = [\![\mathcal{A}]\!]$. A rational language is also called *regular*.

Given alphabets X and Y, a *rational transduction* is a rational subset of the monoid $X^* \times Y^*$. We allow ourselves to abbreviate "rational transduction" to "transduction". Automata over the monoid $X^* \times Y^*$ are called *transducer over* (X, Y), if $X = Y$, we simply speak of transducer over the alphabet X. Homomorphisms $h\colon X^* \to Y^*$ are the simplest examples of rational transductions (when viewed as relations $\{(h(x), x) \mid x \in X^*\}$).

Let $T \subseteq Y^* \times Z^*$ be a rational transduction and let $L \subseteq Z^*$ be a language. Then we write TL for $\{y \in Y^* \mid \exists z \in L\colon (y, z) \in T\}$ for the image of L under T.

Remark 2.1. Consequently, we view the rational transduction $T \subseteq Y^* \times Z^*$ as a mapping from 2^{Z^*} to 2^{Y^*}. Now let $\mathcal{A}$ be a transducer generating T. Then edges in $\mathcal{A}$ are of the form $(p, (y, z), q)$ where p and q are states, $y \in Y^*$ and $z \in Z^*$. In line with our understanding of T as a mapping from 2^{Z^*} to 2^{Y^*}, the word z is the input and the word y is the output in such an edge.

This also explains why we consider the homomorphism $h\colon X^* \to Y^*$ as the set of pairs $(h(x), x) \in Y^* \times X^*$ for $x \in X^*$ since only then $h(L) = hL$.

Furthermore the *composition* of the rational transductions $S \subseteq X^* \times Y^*$ and $T \subseteq Y^* \times Z^*$ is defined as

$$ST = \{(x, z) \in X^* \times Z^* \mid \exists y \in Y^*\colon (x, y) \in S,\ (y, z) \in T\}.$$

Note that $(ST)L = S(TL)$ for any language $L \subseteq Z^*$. Moreover, the *product* of $R, S \subseteq X^* \times Y^*$ is given by

$$R \cdot S = \{(u_0v_0, u_1v_1) \in X^* \times Y^* \mid (u_0, u_1) \in R,\ (v_0, v_1) \in S\}\,.$$

It is well known that the relations $R \cup S, R^{-1}, R \cdot S \subseteq X^* \times Y^*$ and $ST \subseteq X^* \times Z^*$ are rational transductions. If the language L is regular, then also TL is regular. Even more, from transducers for R, S, and T and an automaton for L, one can compute transducers for these relations and an automaton for TL. These properties and many more results about rational transductions can be found in [3].

Remark 2.2. In the literature, one often writes $S \circ T$ for the composition and RS for the product. We deviate from this convention since we will use the composition of rational transductions far more often than the product.

A *language class* is a class of languages that contains at least one non-empty language. We call a language class *Boolean closed* if it is closed under all Boolean operations (union, intersection, and complementation).

A language class $\mathcal{C}$ is called a *full trio* (or *cone*) if it is closed under (arbitrary) homomorphisms, inverse homomorphisms, and intersection with regular languages. According to Nivat's normal form theorem [30], for every rational transduction $R \subseteq X^* \times Y^*$ there exist a regular language L and two homomorphisms g and h such that $R = \{(g(u), h(u)) \mid u \in L\}$. Consequently, for every language $K \subseteq X^*$ we have $RK = g(h^{-1}(K) \cap L)$. It follows that a class $\mathcal{C}$ is a full trio if and only if it is closed under rational transductions, i.e., for every $L \in \mathcal{C}$ and every rational transduction R, we have $RL \in \mathcal{C}$. By the *full trio generated by* the language L we mean the smallest full trio that contains L. Since the intersection of all full trios containing L is a full trio, this class exists for every L. It is not difficult to describe it explicitly: If L is empty, it is the class of regular languages (since $\{\emptyset\}$ is no language class, it is no full trio either); if L is non-empty, it is the class of all languages RL where R is a rational transduction. A full trio is called a *principal full trio* if it is generated by some language. Full trios that are closed under finite unions are called *full semi-AFLs* ("Abstract Family of Languages"). Note that a full semi-AFL is closed under complementation if and only if it is a full trio that is closed under all Boolean operations. A *full AFL* is a full trio that is closed under union and Kleene star, i.e., a full semi-AFL that is closed under Kleene star.

For any class of languages $\mathcal{C}$, we write $\mathsf{RE}(\mathcal{C})$ for the class of languages accepted by some Turing machine with an oracle $L \in \mathcal{C}$. We also write RE for $\mathsf{RE}(\{\emptyset\})$ and $\mathsf{RE}(L)$ for $\mathsf{RE}(\{L\})$. Then the *arithmetical hierarchy* (see [36] for more details) is defined as

$$\Sigma_1 = \mathsf{RE}, \qquad \Sigma_{n+1} = \mathsf{RE}(\Sigma_n) \text{ for } n \geq 1, \qquad \mathsf{AH} = \bigcup_{n\geq 1} \Sigma_n.$$

Languages in AH are called *arithmetical.* For every $n \geq 1$, there exist sets that are Σ_n-complete with respect to many-one reductions. An example of such a set is the set of all (suitable encodings of) first-order sentences of the form $\theta = \exists \bar{x}_1 \forall \bar{x}_2 \cdots \exists/\forall \bar{x}_n \colon \varphi(\bar{x}_1, \ldots, \bar{x}_n)$, where $\bar{x}_1, \ldots, \bar{x}_n$ are tuples of variables and φ is a quantifier-free formula such that θ is true in the arithmetical model $(\mathbb{N}, +, \times)$.

The *arithmetical hierarchy relative to the language* L is defined as

$$\Sigma_1(L) = \mathsf{RE}(L), \quad \Sigma_{n+1}(L) = \mathsf{RE}(\Sigma_n(L)) \text{ for } n \geq 1, \quad \mathsf{AH}(L) = \bigcup_{n\geq 1} \Sigma_n(L).$$

Note that every class $\mathsf{AH}(L)$ is a Boolean closed full trio. Indeed, each $\Sigma_n(L)$ is a full trio: Given $K \in \Sigma_n(L)$ and a rational transduction R, then the language $RK = \{x \mid \exists y \in K \colon (x, y) \in R\}$ belongs to $\Sigma_n(L)$ as well. Moreover, $\Sigma_n(L)$ is closed under union and intersection. Finally, the complement of each member of $\Sigma_n(L)$ is contained in $\Sigma_{n+1}(L)$.

We will often encode words over an alphabet X by words over the Boolean alphabet $\mathbb{B} = \{0, 1\}$. If X is an alphabet, then any homomorphism $g \colon X^* \to \mathbb{B}^*$ with $\{g(a) \mid a \in X\} = \{10^i \mid 1 \leq i \leq |X|\}$ will be called a *standard encoding.* It is an injective homomorphism. Hence, a language $L \subseteq X^*$ is regular if and only if $g(L)$ is regular: Clearly, if L is regular, then $g(L)$ is regular, since regular languages are closed under homomorphic images. On the other hand, if $g(L)$ is regular, then also $g^{-1}(g(L))$ is regular, since regular languages are closed under homomorphic preimages. But this set equals L since g is injective.

For two alphabets X and Y with $Y \subseteq X$, the homomorphism $\pi_Y \colon X^* \to Y^*$ is defined by $\pi_Y(x) = x$ for $x \in Y$ and $\pi_Y(x) = \lambda$ for $x \in X \setminus Y$, i.e., $\pi_Y(u)$ is obtained from u by deleting all letters not belonging to Y.

3 Boolean closed full trios

The first main result of this work is the following.

Theorem 3.1. *Let $\mathbb{B} = \{0, 1\}$. There are rational transductions R, S, T over $\mathbb{B}$ such that for any non-regular language $L \subseteq \mathbb{B}^*$, each $K \in \mathsf{AH}(L)$, $K \subseteq \mathbb{B}^*$, can be obtained from L using R, S, T and Boolean operations.*

Before proving this theorem, we first record a few of its consequences.

Corollary 3.2. *Let X be an alphabet and let $L \subseteq X^*$ be a non-regular language. Then $\mathsf{AH}(L)$ is the smallest Boolean closed full trio containing L.*

Proof. Let $\mathcal{T}$ be the smallest Boolean closed full trio containing L. Since, as remarked earlier, $\mathsf{AH}(L)$ is a Boolean closed full trio, we immediately obtain $\mathcal{T} \subseteq \mathsf{AH}(L)$.

Let $g\colon X^* \to \mathbb{B}$ be a standard encoding. Note that $g(L)$ and L are images of each other under the rational transductions g and g^{-1}, respectively. Hence the smallest Boolean closed full trio $\mathcal{T}_{g(L)}$ containing $g(L)$ equals $\mathcal{T}$. Since L is non-regular, also $g(L)$ is non-regular and we have $\mathsf{AH}(L) = \mathsf{AH}(g(L))$. Hence, also in this case, Theorem 3.1 implies that $\mathsf{AH}(L) = \mathsf{AH}(g(L)) \subseteq \mathcal{T}_{g(L)} = \mathcal{T}$ which concludes the proof. □

A large number of language classes studied in formal language theory are full semi-AFLs, i.e., union closed full trios [11,18,3,14]. Although the authors are not aware of any particular full semi-AFL for which it is not known whether complementation closure is available, the following fact is interesting because of its generality.

Corollary 3.3. *Other than the class of regular languages, no full semi-AFL $\mathcal{C} \subsetneq \mathsf{RE}$ is closed under complementation.*

Proof. Suppose $\mathcal{C}$ were a complementation closed full semi-AFL (i.e., a Boolean closed full trio) that contains a non-regular language L. According to Corollary 3.2, it would already include $\mathsf{AH}(L) \supseteq \mathsf{AH}$ and thus not be included in RE. □

Note that the following corollary is not a special case of Corollary 3.3 as it is not restricted to language classes below RE.

Corollary 3.4. *A principal full trio is closed under complementation if and only if it is the class of regular languages.*

Proof. Let $\mathcal{T}$ be a principal full trio and let L be a language generating $\mathcal{T}$. If L is regular, then $\mathcal{T}$ is the class of regular languages and is therefore closed under complementation.

Suppose L is not regular. Since then L is non-empty, $\mathcal{T}$ consists of all languages of the form RL, where R is a rational transduction. Hence, $\mathcal{T}$ is contained in $\mathsf{RE}(L)$ and is closed under union. The latter follows from the fact that the class of rational transductions is closed under union. If $\mathcal{T}$ were closed under complementation, it would be closed under all Boolean operations and thus, by Corollary 3.2, include $\mathsf{AH}(L)$. Since $\mathsf{RE}(L) \subsetneq \mathsf{AH}(L)$, this is a contradiction. □

Let us also mention a corollary concerning valence automata. A *valence automaton over the monoid M* is an automaton $\mathcal{A}$ over the monoid $X^* \times M$, where X is an alphabet. The *language accepted by* $\mathcal{A}$ is defined as $L(\mathcal{A}) = \{w \in X^* \mid (w, 1) \in [\![\mathcal{A}]\!]\}$. The class of languages accepted by valence automata over M is denoted by $\mathsf{VA}(M)$. We call a monoid M *finitely generated* if there is an alphabet X and a surjective monoid morphism $\varphi\colon X^* \to M$.

Corollary 3.5. *For finitely generated monoids M, the following are equivalent:*

(i) $\mathsf{VA}(M)$ *is closed under complementation.*
(ii) $\mathsf{VA}(M)$ *is the class of all regular languages.*
(iii) M *has only finitely many right-invertible elements.*

Proof. Let X be some alphabet and $\varphi\colon X^* \to M$ a surjective monoid morphism. We consider the language $L = \varphi^{-1}(1)$. It is easy to verify that $\mathsf{VA}(M)$ is the principal full trio generated by L. Hence Corollary 3.4 yields the equivalence between (i) and (ii). The equivalence between (ii) and (iii) has been shown in [35] (and independently in [40]). □

A classic result in the same spirit as Theorem 3.1 has been obtained by Hartmanis and Hopcroft [20], and it concerns intersection closed full AFLs. A *full AFL* is a full trio that is closed under union and Kleene star. Hartmanis and Hopcroft proved that every intersection closed full AFL that contains $E = \{a^n b^n \mid n \in \mathbb{N}\}$ already includes the recursively enumerable languages (see [17] for similar results). Many language classes studied in formal language theory are full AFLs, such as the context-free languages, the one-counter languages, the languages of higher-order pushdown automata (see [3] for the first two classes and [9,10] for the last).

A Boolean closed full trio is also closed under the Kleene star (equation (3), page 17, demonstrates how to express the Kleene star using complementation and transductions) and thus constitutes an intersection closed full AFL. Therefore, Theorem 3.1 tells us that if we expand the set of closure properties in the result by Hartmanis and Hopcroft to Boolean closed full trios, we may replace E with an arbitrary non-regular language. Note that we cannot replace E with an arbitrary non-regular language in the case of intersection closed full AFLs. Ginsburg and Goldstine [17] have constructed non-regular languages L such that the smallest intersection closed full AFL containing L does not contain all of RE. Another counterexample is the class of languages defined by labeled transfer Petri nets with upward-closed target sets: They constitute an intersection closed full AFL [16, Theorem 7] and contain the non-regular language $\{a^n b^m \mid n \geq m\}$ [16]. Moreover, the emptiness problem is decidable for this class.

The rest of this section is devoted to the proof of Theorem 3.1.

3.1 Simplifying the task

Theorem 3.1 claims the existence of only three rational transductions over $\{0,1\}$ that allow, for all non-regular languages $L \subseteq \{0,1\}^*$ and all $n \in \mathbb{N}$, constructing all languages $K \subseteq \{0,1\}^*$ from $\Sigma_n(L)$. Note that $K \in \Sigma_{n+1}(L)$ if there is $L' \in \Sigma_n(L)$ with $K \in \mathsf{RE}(L')$ (and we can even assume $L' \subseteq \{0,1\}^*$). Hence the central problem is the construction of languages from $\mathsf{RE}(L)$. This construction is much easier to understand with larger alphabets and more than three rational transductions. For these reasons, our proof involves the following weaker version of Theorem 3.1:

Theorem 3.6. *Let $\mathbb{B} = \{0,1\}$. There is a finite set F of rational transductions over some alphabet Z such that for any languages $K, L \subseteq \mathbb{B}^*$ with $K \in \mathsf{RE}(L)$ and L non-regular, the language K can be constructed from the language L using the rational transductions from F and Boolean operations.*

We first show how to derive the main result, Theorem 3.1, from this weaker version. This proof splits into two lemmas, one dealing with the number of rational transductions and the other with the alphabets.

Lemma 3.7. *Let $\mathbb{B} = \{0,1\}$. For each finite set F of rational transductions over $\mathbb{B}$, there are rational transductions R, S, T over $\mathbb{B}$ such that every composition of transductions from F can be written in the form $T^n S^m R$ with $m, n \in \mathbb{N}$.*

Proof. For $x \in \mathbb{B}$, let A_x be the transduction that appends x to each input word, hence $A_x = \{(wx, w) \mid w \in \mathbb{B}^*\}$. Furthermore, let $F = \{U_0, \dots, U_{k-1}\}$, $b = k+1$, and let U'_i be the rational transduction

$$U'_i = \{(u10^m, v10^{bm+i}) \mid (u,v) \in U_i, m \in \mathbb{N}\}, \quad U'_k = \{(w, w10^k) \mid w \in \mathbb{B}^*\}$$

for each $0 \le i < k$. We shall prove that $R = A_1$, $S = A_0$, and $T = \bigcup_{0 \le i \le k} U'_i$ have the desired property. Let $U_{i_n} \cdots U_{i_0}$ be a composition of elements of F and let $i_{n+1} = k$. We claim that

$$U_{i_n} \cdots U_{i_0} = T^{n+2} S^m R \quad \text{for } m = \sum_{j=0}^{n+1} i_j b^j.$$

The idea is that the exponent m encodes the sequence of indices $i_{n+1}, i_n, \dots, i_0$ in base b. Applying $S^m R$ appends 10^m to each input word. Then, each application of T to a word $w10^\ell$ chooses some U'_j, but this choice will only lead to a valid computation of the transducer if ℓ is congruent to j modulo b. Hence, applying T^{n+1} to $w10^m$ has the same effect as applying $U'_{i_n} \cdots U'_{i_0}$. Since the most significant digit in the b-ary representation of m is $i_{n+1} = k$, applying T once more means applying U'_k and hence removing the 10^k suffix of the input word. In the end, we applied $U_{i_n} \cdots U_{i_0}$. □

Lemma 3.8. *Let $\mathbb{B} = \{0,1\}$, let X be an arbitrary alphabet and let F be a finite set of rational transductions over X. There exists a finite set F' of rational transductions over $\mathbb{B}$ such that the following holds for all languages $K, L \subseteq \mathbb{B}^*$: If K can be constructed from L using transductions from F and Boolean operations, then it can be constructed from L using transductions from F' and Boolean operations.*

Proof. Let $g\colon X^* \to \mathbb{B}^*$ be some standard encoding. In F', we collect the following rational transductions over $\mathbb{B}$:

- gRg^{-1} for $R \in F$,
- $R_Y = \{(g(w), g(w)) \mid w \in Y^*\}$ for $Y \subseteq X$,
- $R_g = \{(g(0), 0), (g(1), 1)\}^*$ and R_g^{-1}.

We clearly have $g(L \cap K) = g(L) \cap g(K)$ and $g(L \cup K) = g(L) \cup g(K)$ for $K, L \subseteq X^*$. Since g is injective, the mapping $g^{-1}g$ is the identity on X^*. Hence also $g(RL) = gRg^{-1}(g(L))$ for any transduction R over X and any language $L \subseteq X^*$. Since g is injective, we also get

$$g(\overline{L}) = g(\alpha(L)^* \setminus L) = g(\alpha(L))^* \setminus g(L) = g(\alpha(L))^* \cap \overline{g(L)} = R_{\alpha(L)}(\overline{g(L)})$$

for $L \subseteq X^*$.

Let $K, L \subseteq \mathbb{B}^*$ and suppose that K can be constructed from L using transductions from F and Boolean operations. Then, by induction on the construction of

K from L, the language $g(K)$ can be constructed from $g(L)$ using transductions from F' and Boolean operations. Since $K = R_g^{-1}(g(K))$ and $g(L) = R_g(L)$, we can therefore construct K from L using transductions from F' and Boolean operations. □

Proof of Theorem 3.1 assuming Theorem 3.6. Let $\mathbb{B}$, F and Z be as in Theorem 3.6. By Lemma 3.8, we find a finite set F' of rational transductions over $\mathbb{B}$ such that any language $K \subseteq \mathbb{B}^*$ with $K \in \mathsf{RE}(L)$ can be constructed from any non-regular language $L \subseteq \mathbb{B}^*$ using transductions from F' and Boolean operations. Applying Lemma 3.7 to this set gives three rational transductions R, S, and T over $\mathbb{B}$ that, for any non-regular language $L \subseteq \mathbb{B}^*$ allow generating (from L) all languages $K \subseteq \mathbb{B}^*$ from $\mathsf{RE}(L) = \Sigma_1(L)$ using also Boolean operations. By induction, this allows constructing all languages $K \subseteq \mathbb{B}^*$ in $\mathsf{AH}(L)$ from L. Hence, Theorem 3.1 follows indeed from Theorem 3.6. □

Given this proof, it remains to demonstrate the correctness of Theorem 3.6. To this aim, we will proceed in three steps:

1. As a kind of warm-up (and for later use), we show how to generate all regular languages from X^* with a fixed finite set of rational transductions (without using any Boolean operations, Lemma 3.9).
2. The central part of the proof is the second step: Let $H \subseteq \{0,1\}^*$ be recursively enumerable and $L \subseteq \{0,1\}^*$ be non-regular. Then H can be accepted by some 2-counter automaton. We encode the computations of this 2-counter automaton by certain words. The main idea is that counter values are not encoded in any particular number system, but by the infinitely many equivalence classes of the Myhill-Nerode-equivalence of the non-regular language L. As a result, any recursively enumerable language H can be constructed from any non-regular language L (Lemma 3.16).
3. In the third and final step, let $K \subseteq \{0,1\}^*$ be recursively enumerable in $L \subseteq \{0,1\}^*$ (with L non-regular), i.e., $K \in \mathsf{RE}(L)$. Then there is an oracle Turing machine accepting K with oracle L. The set of accepting computations of this oracle Turing machine (when ignoring the correctness of the oracle answers) is computable. Consequently, the sequence of oracle answers in accepting computations is recursively enumerable and therefore in RE. Hence it can be constructed from L (by the second step of our proof) and, again using L, we can also verify that all the oracle answers are correct (this is the sketch of the final proof of Theorem 3.6 from page 17).

3.2 Constructing regular languages

In this subsection, we show that using a fixed set of rational transductions, one can obtain every regular language from X^*.

Lemma 3.9. *Let X be an alphabet and $Y = X \uplus \{\#, \$\}$. There is a finite set F of rational transductions over Y such that for any languages $K, L \subseteq X^*$ with K regular and $L \neq \emptyset$, the language K can be constructed from the language L using the rational transductions from F.*

Proof. We may assume that K is accepted by an automaton

$$\mathcal{A} = (Q, X^*, E, 1, Q_f),$$

where $Q = \{1, \ldots, k\}$, $Q_f \subseteq Q$, and $E \subseteq Q \times X \times Q$.

Our goal is to produce the language $T_{\mathcal{A}}$ of all encodings of accepting runs of $\mathcal{A}$, i.e., of all words $\$^{i_0} x_1 \$^{i_1} \cdots x_n \$^{i_n}$, such that $i_0 = 1$, $i_n \in Q_f$, $x_j \in X$ and $(i_j, x_{j+1}, i_{j+1}) \in E$ for $0 \leq j < n$. Then, clearly, the rational transduction P that outputs only the x_j (i.e., deletes all occurrences of \$) will satisfy $PT_{\mathcal{A}} = K$. In the following, the additional symbol # is called a *marker*.

First we use the initial transduction

$$I = \$(X\$^+)^* \times X^*$$

to produce the set $\$(X\$^+)^* = IL$ from $L \neq \emptyset$. In the following, a word

$$\$^{i_0} x_1 \$^{i_1} \cdots x_n \$^{i_n}$$

is called an *encoding*. Its factors $\$^{i_j}$ are called *state blocks* and its factors $\$^{i_j} x \$^{i_{j+1}}$ are called *transition blocks*.

The transduction I already guarantees that the leftmost state block corresponds to the initial state. We now wish to remove all words that contain a state block of length greater than k. In order to do this, we use the transduction S, which inserts the marker # in the beginning of every state block. Furthermore, we have the transduction M, which moves each occurrence of the marker one position to the right (i.e. outputs \$# on input #\$) if its right neighbor is a \$, and drops the occurrence otherwise. We also have the transduction R, which rejects all inputs that contain the marker #. All other words are unchanged by R. Then applying $RM^{k+1}S$ yields the set of encodings with state blocks of length at most k.

In the next step, we wish to remove from the language all encodings whose rightmost state block does not correspond to any accepting state from Q_f. To this end, we use the transduction S' that inserts a marker in the beginning of the last state block. The transduction R' rejects all words that end with # and removes all occurrences of # from any other word. Then $R'M^iS'$ yields the set of encodings with rightmost state block not of length i. Hence, applying $R'M^iS$ for each $i \in Q \setminus Q_f$ in succession yields the set of encodings whose rightmost state block represents an accepting state.

In the final step, we wish to remove from the language all encodings that contain a transition block $\$^\ell x \$^m$ with $x \in X$, $0 \leq \ell, m \leq k$, and $(\ell, x, m) \notin E$. Again, we use S to introduce # at the beginning of each state block. Then, we use M^ℓ to move every # by ℓ positions to the right. Next, we apply M_x, which replaces $\#x$ by $x\#$. Occurrences of # that are not followed by x are removed by M_x. Then we apply M^m. At the end, we use R'', which rejects all words that have a factor from $\#X$ or that end with #, and removes all # that are followed by \$. In total, applying $R''M^mM_xM^\ell S$ clearly yields the set of encodings that do not contain the transition block $\$^\ell x \$^m$. Therefore, we apply this sequence of transductions for each triple (ℓ, x, m) with $0 \leq \ell, m \leq k$, $x \in X$, and $(\ell, x, m) \notin E$. This clearly produces the language $T_{\mathcal{A}}$ and hence $K = PT_{\mathcal{A}}$ is obtained. Since we only used transductions in $F = \{I, S, S', M, R, R', R''\} \cup \{M_x \mid x \in X\}$, the lemma is proved. $\square$

Lemmas 3.7, 3.8, and 3.9 together immediately imply the following byproduct, which might be of independent interest.

Corollary 3.10. *Let $\mathbb{B} = \{0, 1\}$. There are rational transductions S, T in $\mathbb{B}^* \times \mathbb{B}^*$ and a regular language $L \subseteq \mathbb{B}^*$ such that every regular language over $\mathbb{B}$ can be written as $T^n S^m L$ for some $m, n \in \mathbb{N}$.*

Proof. By Lemma 3.9, there is a finite set of rational transductions over $\mathbb{B} \cup \{\$, \#\}$ that allows constructing any regular language over $\mathbb{B}$ from $\mathbb{B}^*$. By Lemma 3.8, there is a finite set of rational transductions over $\mathbb{B}$ that allows constructing any regular language over $\mathbb{B}$ from $\mathbb{B}^*$. Hence, by Lemma 3.7, there are three rational transductions R, S, T such that any regular language $K \subseteq \mathbb{B}^*$ equals $T^n S^m R\mathbb{B}^*$ for some $m, n \in \mathbb{N}$. Hence, the corollary holds with $L = R\mathbb{B}^*$. □

3.3 Constructing the counter language C

This subsection is the technical heart of the proof of Theorem 3.6. Here, we show that from an arbitrary non-regular language and using a fixed finite set of rational transductions and Boolean operations, one can construct the language C of valid sequences of counter operations.

We define the alphabet $\Delta = \{+, -, z\}$, whose elements will represent the operations *increment*, *decrement*, and *zero test*, respectively.

Definition 3.11. Let $C \subseteq \Delta^*$ be the set of words $\delta_1 \cdots \delta_m$, $\delta_1, \ldots, \delta_m \in \Delta$ for which there are numbers $x_0, \ldots, x_m \in \mathbb{N}$ such that $x_0 = 0$ and for $1 \leq i \leq m$:

1. if $\delta_i = +$, then $x_i = x_{i-1} + 1$,
2. if $\delta_i = -$, then $x_i = x_{i-1} - 1$, and
3. if $\delta_i = z$, then $x_i = x_{i-1} = 0$.

The main difficulty in proving Theorem 3.6 is to construct C from a language L, where the only information we have about L is that it is not regular. A central role in this construction is played by the Myhill-Nerode equivalence (and the non-regularity of L is used since only then, we have infinitely many equivalence classes): Let X be an alphabet and $L \subseteq X^*$. For words $u, v \in X^*$, we write $u \equiv_L v$ if for each $w \in X^*$, we have

$$uw \in L \quad \text{if and only if} \quad vw \in L.$$

The equivalence relation $\equiv_L$ is called the *Myhill-Nerode equivalence.* The well-known Myhill-Nerode Theorem (see e.g. [22]) states that L is regular if and only if $\equiv_L$ has a finite index. Using the Myhill-Nerode equivalence, we define another language from C, which can be thought of as making the counter values x_i explicit and encoding them as Myhill-Nerode classes.

Definition 3.12. Consider the alphabets $\mathbb{B} = \{0, 1\}$ and $\Delta = \{+, -, z\}$, some symbol $\# \notin \mathbb{B} \cup \Delta$ and a language $L \subseteq \mathbb{B}^*$. Let $\hat{C}_L \subseteq (\Delta \cup \mathbb{B} \cup \{\#\})^*$ be the set of all words

$$v_0 \delta_1 v_1 \cdots \delta_m v_m \# u_0 \cdots \# u_n \tag{1}$$

with $m, n \geq 0$, $\delta_i \in \Delta$, $v_i, u_j \in \mathbb{B}^*$, such that $u_k \not\equiv_L u_\ell$ for $k \neq \ell$, $v_0 \equiv_L u_0$, and for each $1 \leq i \leq m$ and $0 \leq j \leq n$, we have

1. if $\delta_i = +$ and $v_{i-1} \equiv_L u_j$, then $v_i \equiv_L u_{j+1}$,
2. if $\delta_i = -$ and $v_{i-1} \equiv_L u_j$, then $v_i \equiv_L u_{j-1}$, and
3. if $\delta_i = z$, then $v_{i-1} \equiv_L v_i \equiv_L u_0$.

The idea behind this definition is that counter values are represented by Myhill-Nerode classes (with respect to L), which are denoted by the words u_i and v_j in (1). This means, words that are Myhill-Nerode equivalent represent the same counter value. The words $v_0, \ldots, v_m$ describe the counter values as they are attained over time (the class of v_i represents the value at time $i \in \{0, \ldots, m\}$), and the words $u_0, \ldots, u_n$ describe the counter values sorted by their magnitude (the class u_j represents the value $j \in \{0, \ldots, n\}$). Therefore, to distinguish different counter values, we require that $u_k \not\equiv_L u_\ell$ for $k \neq \ell$. The word v_{i-1} (resp., v_i) in (1) represents the counter value before (resp., after) the counter operation δ_i. For instance, if $\delta_i = +$, then the counter value represented by v_i should be one more than the counter value represented by v_{i-1}. This is expressed by requiring that for some j, $v_{i-1} \equiv_L u_j$ and $v_i \equiv_L u_{j+1}$.

Note that since in $\hat{C}_L$, the counter values are represented by Myhill-Nerode classes, $\hat{C}_L$ can mimic counter operations up to the number of these classes: Encoding k distinct counter values requires k Myhill-Nerode classes. This means, if L is non-regular, projecting to the counter instruction symbols yields precisely C. If L is regular, this is not true: In that case, the projection only contains instruction sequences where the counter value stays below the index of $\equiv_L$.

Lemma 3.13. *Let $\mathbb{B} = \{0, 1\}$. If the language $L \subseteq \mathbb{B}^*$ is not regular, then we have $\pi_\Delta(\hat{C}_L) = C$.*

Proof. In order to prove the inclusion "$\supseteq$", let $\delta_1 \cdots \delta_m \in C$. Then there are numbers $x_0, \ldots, x_m \in \mathbb{N}$ as in Definition 3.11. There is $n \in \mathbb{N}$ such that $\{x_0, \ldots, x_m\}$ is included in $\{0, \ldots, n\}$. Since L is not regular, we can find words $u_0, \ldots, u_n \in \mathbb{B}^*$ such that $u_k \not\equiv_L u_\ell$ for $k \neq \ell$. Now for each $0 \leq i \leq m$, let $v_i = u_{x_i}$. Then it can be checked straightforwardly that $v_0\delta_1 v_1 \cdots \delta_m v_m \# u_0 \cdots \# u_n \in \hat{C}_L$ and hence $\delta_1 \cdots \delta_m \in \pi_\Delta(\hat{C}_L)$.

For the inclusion "$\subseteq$", suppose $\delta_1 \cdots \delta_m \in \pi_\Delta(\hat{C}_L)$. Then there are words $v_0, \ldots, v_m, u_0, \ldots, u_n \in \mathbb{B}^*$ with

$$v_0\delta_1 v_1 \cdots \delta_m v_m \# u_0 \cdots \# u_n \in \hat{C}_L.$$

Using the fact that the u_k are pairwise incongruent wrt. $\equiv_L$ and by induction on i, one can easily verify that for each $0 \leq i \leq m$, there is a unique $x_i \in \{0, \ldots, n\}$ such that $v_i \equiv_L u_{x_i}$. By the definition of $\hat{C}_L$, this choice of $x_0, \ldots, x_n$ satisfies the conditions 1–3 of Definition 3.11. ◻

The following lemma is the central ingredient in our proof. We show that from each language L, one can construct $\hat{C}_L$ using Boolean operations and a fixed finite set of rational transductions. Then, when L is non-regular, Lemma 3.13 allows us to obtain C.

Lemma 3.14. *Let $\mathbb{B} = \{0, 1\}$ and $Y = \mathbb{B} \cup \Delta \cup \{\#\}$. There exists a finite set F of rational transductions such that for any (possibly regular) language $L \subseteq \mathbb{B}^*$, the language $\hat{C}_L \subseteq Y^*$ can be obtained from L using transductions in F and Boolean operations.*

Proof. We construct $\hat{C}_L$ from L using a sequence of Boolean operations and transductions $T_1, \dots, T_{19}$ over Y for which it will be clear that they do not depend on L.

There are clearly rational transductions T_1 and T_2 with

$$\begin{aligned} W_1 &= \{u\#v\#w \mid u, v, w \in \mathbb{B}^*,\ uw \in L\} = T_1 L, \\ W_2 &= \{u\#v\#w \mid u, v, w \in \mathbb{B}^*,\ vw \in L\} = T_2 L, \end{aligned}$$

which means we can construct W_1 and W_2. Hence,

$$\begin{aligned} W' &= \{u\#v\#w \mid u, v, w \in \mathbb{B}^*, (uw \in L, vw \notin L) \text{ or } (uw \notin L, vw \in L)\} \\ &= (W_1 \cap \overline{W_2}) \cup (\overline{W_1} \cap W_2) \end{aligned}$$

can also be constructed. We can clearly find a rational transduction T_3 with

$$\begin{aligned} W &= \{u\#v \mid u, v \in \mathbb{B}^*, u \not\equiv_L v\} \\ &= \{u\#v \mid u\#v\#w \in W' \text{ for some } w \in \mathbb{B}^*\} \\ &= T_3 W'. \end{aligned}$$

This means $P = \{u\#v \mid u, v \in \mathbb{B}^*, u \equiv_L v\} = \mathbb{B}^*\#\mathbb{B}^* \setminus W = \mathbb{B}^*\#\mathbb{B}^* \cap \overline{W} = T_4\overline{W}$, for some transduction T_4, can be constructed. With suitable rational transductions T_5, T_6, we have

$$\begin{aligned} S &= \{u_0\#u_1 \cdots \#u_n \mid u_i \not\equiv_L u_j \text{ for all } i \neq j\} \\ &= (\mathbb{B}^*\#)^*\mathbb{B}^* \setminus \{ru\#svt \mid r, s \in (\mathbb{B}^*\#)^*, t \in (\#\mathbb{B}^*)^*, u\#v \in P\} \\ &= T_6\overline{T_5 P}, \end{aligned}$$

meaning that S can be constructed as well. Let M (*matching*) be the set of all words $v_1\delta v_2\#u_1\#u_2$ where $v_1, v_2, u_1, u_2 \in \mathbb{B}^*$ and $\delta \in \Delta$ with

- if $\delta = +$, then $v_1 \equiv_L u_1$ and $v_2 \equiv_L u_2$,
- if $\delta = -$, then $v_1 \equiv_L u_2$ and $v_2 \equiv_L u_1$, and
- if $\delta = z$, then $v_1 \equiv_L v_2 \equiv_L u_1$.

Since

$$\begin{aligned} M &= \{v_1{+}v_2\#u_1\#u_2 \mid v_1\#u_1 \in P,\ v_2\#u_2 \in P\} \\ &\quad \cup \{v_1{-}v_2\#u_1\#u_2 \mid v_1\#u_2 \in P,\ v_2\#u_1 \in P\} \\ &\quad \cup \{v_1 z v_2\#u_1\#u_2 \mid v_1\#v_2 \in P,\ v_1\#u_1 \in P,\ u_2 \in \mathbb{B}^*\} \\ &= (T_7 P \cap T_8 P) \cup (T_9 P \cap T_{10} P) \cup (T_{11} P \cap T_{12} P) \end{aligned}$$

for suitable rational transductions $T_7, \dots, T_{12}$, we can also construct M.

Let E (which stands for *error*) be the set of words $v_1\delta v_2\#u_0 \cdots \#u_n$ such that for every $1 \leq j \leq n$, we have $v_1\delta v_2\#u_{j-1}\#u_j \notin M$ or we have $\delta = z$ and $v_1 \not\equiv_L u_0$. Since

$$\begin{aligned} E' &= \{v_1\delta v_2 r\#u_1\#u_2 s \mid v_1\delta v_2\#u_1\#u_2 \in M, r, s \in (\#\mathbb{B}^*)^*\} \\ &= T_{13} M \end{aligned}$$

for some rational transduction T_{13}, we can construct E'. Furthermore, since

$$\begin{aligned} E &= \left[(\mathbb{B}^*\varDelta\mathbb{B}^*\#\mathbb{B}^*(\#\mathbb{B}^*)^* \setminus E'\right] \\ &\quad \cup \{v_1 z v_2 \# u_0 r \mid v_1 \not\equiv_L u_0,\ r \in (\#\mathbb{B}^*)^*, v_2 \in \mathbb{B}^*\} \\ &= T_{14}\overline{E'} \cup T_{15}\overline{P}, \end{aligned}$$

for some rational transductions T_{14}, T_{15}, we can construct E.

Let N (*no error*) be the set of words $v_0\delta_1 v_1 \cdots \delta_m v_m \# u_0 \cdots \# u_n$ such that for every $1 \le i \le m$, there is a $1 \le j \le n$ with $v_{i-1}\delta_i v_i \# u_{j-1} \# u_j \in M$ and if $\delta_i = z$, then $v_{i-1} \equiv_L u_0$. Since

$$N' = \{w \in (\mathbb{B}^*\varDelta)^* v_1 \delta v_2 (\varDelta\mathbb{B}^*)^* \# u_0 \cdots \# u_n \mid v_1 \delta v_2 \# u_0 \cdots \# u_n \in E\} = T_{16}E$$

and

$$N = (\mathbb{B}^*\varDelta)^+\mathbb{B}^*\#\mathbb{B}^*(\#\mathbb{B}^*)^* \setminus N' = T_{17}\overline{N'}$$

for some rational transductions T_{16}, T_{17}, we can construct N.

Finally, we define I (*initial condition*) to be the language of those

$$v_0\delta_1 v_1 \cdots \delta_m v_m \# u_0 \cdots \# u_n \in N$$

with $v_0 \equiv_L u_0$. Since

$$I = N \cap \{v_0(\varDelta\mathbb{B}^*)^*\#u_0(\#\mathbb{B}^*)^* \mid v_0\#u_0 \in P\} = N \cap T_{18}P,$$

for some rational transduction T_{18}, we can construct I.

Now we have $\hat{C}_L = I \cap (\mathbb{B}^*\varDelta)^*\mathbb{B}^*\#S = I \cap T_{19}S$ for some rational transduction T_{19}, meaning we can construct $\hat{C}_L$. This proves our claim and hence the lemma. □

Lemma 3.15. *Let $\mathbb{B} = \{0,1\}$. There is a finite set F of rational transductions such that for any non-regular language $L \subseteq \mathbb{B}^*$, the language C can be obtained from L using transductions in F and Boolean operations.*

Proof. Let $L \subseteq \mathbb{B}^*$ be non-regular. By Lemma 3.13, we have $C = \pi_\varDelta(\hat{C}_L)$. Thus, if we add $\pi_\varDelta$ to the set of transductions from Lemma 3.14, we obtain a finite set that allows constructing C from L. □

3.4 Constructing all recursively enumerable languages

This subsection is the last technical step in proving Theorem 3.6. Using the counter language constructed in Section 3.3, we construct all recursively enumerable languages. To this aim, we use two-counter automata. Recall that $\varDelta$ is the alphabet $\{+,-,z\}$. A *two-counter automaton* is a tuple

$$\mathcal{A} = (Q, X, E, q_0, Q_f),$$

where (i) Q is a finite set of *states*, (ii) X is its *input alphabet*, (iii) E is a finite subset of $Q \times X^* \times \varDelta \times \varDelta \times Q$ whose elements are called *edges*, (iv) $q_0 \in Q$ is its *initial state*, and (v) $Q_f \subseteq Q$ is its set of *final states*. A *configuration* is an element of $Q \times X^* \times \mathbb{N} \times \mathbb{N}$. For configurations (q, u, n_0, n_1) and (q', u', n_0', n_1'), we write $(q, u, n_0, n_1) \vdash_\mathcal{A} (q', u', n_0', n_1')$ if there is an edge $(q, v, \delta_0, \delta_1, q') \in E$ such that $u' = uv$ and for each $i \in \{0,1\}$, we have

1. $\delta_i = +$ and $n'_i = n_i + 1$,
2. $\delta_i = -$ and $n'_i = n_i - 1$, or
3. $\delta_i = z$ and $n'_i = n_i = 0$.

The language *accepted* by $\mathcal{A}$ is then

$$L(\mathcal{A}) = \{w \in X^* \mid \exists f \in Q_f, n_0, n_1 \in \mathbb{N}\colon (q_0, \lambda, 0, 0) \vdash^*_{\mathcal{A}} (f, w, n_0, n_1)\}.$$

The definition here forces the automaton to operate on both counters in each step, whereas in the usual definition, these automata can also use only one counter at a time. This is not a serious restriction: A two-counter automaton that accesses only one counter at a time can be simulated as follows. Instead of incrementing counter i, we first increment both counters and then decrement counter $1-i$ and increment counter i again. If we proceed analogously for decrement (decrement i and increment $1-i$, then decrement i and decrement $1-i$) and zero test (zero test on i and increment on $1-i$, then zero test on i and decrement on $1-i$), we represent the counter values (n_0, n_1) of the old automaton by the values $(2n_0, 2n_1)$ and thus accept the same language.

Lemma 3.16. *Let $\mathbb{B} = \{0,1\}$ and $X = \mathbb{B} \cup \Delta \cup \{\#, \$\}$. There is a finite set F of rational transductions over X such that for any non-regular language $L \subseteq \mathbb{B}^*$, every language $K \subseteq \mathbb{B}^*$ with $K \in \mathsf{RE}$ can be obtained from L using transductions in F and Boolean operations.*

Proof. Let F_1 be the set of rational transductions provided by Lemma 3.9 when the alphabet X is used. Furthermore, let F_2 be the set of rational transductions provided by Lemma 3.15.

Suppose $K \subseteq \mathbb{B}^*$ is recursively enumerable. There is a two-counter automaton $\mathcal{A} = (Q, \mathbb{B}, E, 1, \{2\})$ that accepts K and satisfies $Q = \{1, \ldots, k\}$. Let R be the regular language of all words

$$\$^{m_0} \prod_{i=1}^{n} \#w_i\#\delta_i^{(0)}\delta_i^{(1)}\$^{m_i}$$

with $(m_{i-1}, w_i, \delta_i^{(0)}, \delta_i^{(1)}, m_i) \in E$ for every $1 \le i \le n$, $m_0 = 1$, and $m_n = 2$. By the choice of F_1, we can obtain R from L using only transductions in F_1.

Recall the definition of the language C from Definition 3.11. Clearly, there are rational transductions T_1 and T_2 such that

$$\begin{aligned} U &= \left\{ \$^{m_0} \prod_{i=1}^{n} \#w_i\#\delta_i^{(0)}\delta_i^{(1)}\$^{m_i} \in R \;\middle|\; \delta_1^{(j)} \cdots \delta_n^{(j)} \in C \text{ for } j \in \{0,1\} \right\} \\ &= R \cap T_1 C \cap T_2 C. \end{aligned}$$

By the choice of F_2, we can obtain U from L using only Boolean operations and transductions in $F_1 \cup F_2 \cup \{T_1, T_2\}$. Finally, applying to U the transduction $\pi_{\mathbb{B}}$ that outputs all occurrences of letters from $\mathbb{B}$ clearly yields K. Therefore, setting $F = F_1 \cup F_2 \cup \{T_1, T_2, \pi_{\mathbb{B}}\}$ proves the lemma. $\square$

3.5 Proof of Theorem 3.6

Let F' be the set of transductions provided by Lemma 3.16 and let $K, L \subseteq \mathbb{B}^*$ with $K \in \mathsf{RE}(L)$ and L non-regular. This means that there is an oracle Turing machine $\mathcal{A}$ such that K is accepted by $\mathcal{A}^L$. We turn the oracle Turing machine $\mathcal{A}^L$ into an ordinary Turing machine $\mathcal{A}'$ as follows. The Turing machine $\mathcal{A}'$ simulates $\mathcal{A}^L$, except for oracle queries: Whenever $\mathcal{A}^L$ uses the oracle for a word, $\mathcal{A}'$ just guesses an answer nondeterministically. Moreover, $\mathcal{A}'$ has two additional tapes where it records all oracle queries for which it guessed "yes" and "no", respectively. After simulating $\mathcal{A}^L$, the Turing machine $\mathcal{A}'$ outputs

$$u_1\#_1 \cdots u_n\#_1 v_1\#_2 \cdots v_m\#_2 w, \tag{2}$$

where $w \in \mathbb{B}^*$ is the input read by the computation, $u_1, \dots, u_n \in \mathbb{B}^*$ are the queries where $\mathcal{A}'$ guessed "yes", and $v_1, \dots, v_m \in \mathbb{B}^*$ are the queries where $\mathcal{A}'$ guessed "no". Let $K' \subseteq Y^*$ with $Y = \mathbb{B} \cup \{\#_1, \#_2\}$ be the set of words (2) output by $\mathcal{A}'$. Then, by construction, K' is a recursively enumerable language. We have

$$\begin{aligned} K = \{w \in \mathbb{B}^* \mid \exists u_1, \dots, u_n \in L,\ v_1, \dots, v_m \in \mathbb{B}^* \setminus L\colon \\ u_1\#_1 \cdots u_n\#_1 v_1\#_2 v_1\#_2 \cdots v_m\#_2 w \in K'\}. \end{aligned}$$

Let $g\colon Y^* \to \mathbb{B}^*$ be a standard encoding. Then also $g(K')$ is recursively enumerable. By Lemma 3.16, $g(K')$ can be obtained from L by transductions in F' and Boolean operations. Hence, we can obtain $K' = g^{-1}(g(K'))$ from L.

Furthermore, since

$$\begin{aligned} (L\#_1)^* &= \overline{(\mathbb{B}^*\#_1)^*\overline{L}\#_1(\mathbb{B}^*\#_1)} \cap (\mathbb{B}^*\#_1)^* = T_2\left(\overline{T_1\overline{L}}\right), \\ (\overline{L}\#_2)^* &= \overline{(\mathbb{B}^*\#_2)^*L\#_2(\mathbb{B}^*\#_2)} \cap (\mathbb{B}^*\#_2)^* = T_4\left(\overline{T_3L}\right) \end{aligned} \tag{3}$$

for some rational transductions T_1, T_2, T_3, T_4, we can construct $(L\#_1)^*$ and $(\overline{L}\#_2)^*$ from K. Moreover, since

$$\begin{aligned} K'' &:= \{u_1\#_1 \cdots u_n\#_1 v_1\#_2 \cdots v_m\#_2 w \in K' \mid u_1, \dots, u_n \in L, \quad v_1, \dots, v_m \in \overline{L}\} \\ &= K' \ \cap\ (L\#_1)^*(\mathbb{B}^*\#_2)^*\mathbb{B}^* \ \cap\ (\mathbb{B}^*\#_1)^*(\overline{L}\#_2)^*\mathbb{B}^* \\ &= K' \ \cap\ T_5(L\#_1)^* \ \cap\ T_6(\overline{L}\#_2)^* \end{aligned}$$

for suitable rational transductions T_5, T_6, we can construct K'' from L. Finally, we apply a transduction T_7 that, for an input from Y^*, outputs the longest suffix in $\mathbb{B}^*$. This yields K from L. Since, apart from the transductions in F', we only used g^{-1} and $T_1, \dots, T_7$, Theorem 3.6 follows. $\square$

4 Synchronous rational transductions

Let $L \subseteq \{0,1\}^*$ be any non-regular language. Then, by Theorem 3.1, we can construct from L any arithmetical language $K \subseteq \{0,1\}^*$ using rational transductions and Boolean operations. The proof makes crucial use of *asynchronous* rational transductions, i.e., transductions accepted by automata whose edges are labeled by pairs of words *of possibly different length*. In this section, we study the question of whether this is avoidable. We give two answers to this question:

1. If L is non-regular and has a neutral word (see Definition 4.4 below), then we can construct a non-recursively-enumerable language from L using synchronous rational transductions and Boolean operations. Moreover, if L is also recursive, then we can construct for each $n \in \mathbb{N}$ a language that is hard for Σ_n.
2. There is a non-regular language L such that only recursive languages can be constructed from L using synchronous rational transductions and Boolean operations.

We start with the definition of synchronous rational transductions. We follow [15, Def. 4.1].

Definition 4.1. Let X be an alphabet. A transducer $\mathcal{A} = (Q, X^* \times X^*, E, q_0, Q_f)$ over the alphabet X is *synchronous* if $(p,(u,v),q) \in E$ implies $|u| = |v| = 1$. A relation $R \subseteq X^* \times X^*$ is a *synchronous rational transduction* if it is a finite union of relations of the form

$$[\![\mathcal{A}]\!] \cdot (L \times \{\lambda\}) \text{ or } [\![\mathcal{A}]\!] \cdot (\{\lambda\} \times L)$$

where $\mathcal{A}$ is a synchronous transducer over X and $L \subseteq X^*$ is a regular language.

It is easily verified that the relations $R \cup S$, R^{-1} and $ST \subseteq X^* \times Z^*$ are effectively synchronous rational transductions whenever $R, S \subseteq X^* \times Y^*$ and $T \subseteq Y^* \times Z^*$ are synchronous rational transductions. As in the case of arbitrary rational transductions, the language TL is effectively regular if $L \subseteq Z^*$ is regular. Differently from the general case, the product $R \cdot S$ of two synchronous rational transductions R and S need not be synchronous rational (e.g., consider $R = \{a\}^* \times \{\lambda\}$ and $S = \{(b^n, b^n) \mid n \in \mathbb{N}\}$). The following shows that $R \cdot S$ is synchronous rational whenever S is the direct product of two regular languages.

Example 4.2. *If R is a synchronous rational transduction and K and L are regular languages (all over the alphabet X), then also $R \cdot (K \times L)$ is a synchronous rational transduction:*

Since for languages K_1, K_2, L_1, L_2, the relation $(K_1 \times L_1) \cdot (K_2 \times L_2)$ equals $(K_1K_2) \times (L_1L_2)$, it suffices to consider the case $R = [\![\mathcal{A}]\!]$ for some synchronous transducer $\mathcal{A}$. For $x \in X^*$, let $x^{-1}K = \{y \mid xy \in K\}$. Then $[\![\mathcal{A}]\!] \cdot (K \times L)$ is the union of all relations of the form

– $\underbrace{[\![\mathcal{A}]\!] \cdot \{(x', y) \mid |x'| = |y|, x'^{-1}K = x^{-1}K, y \in L\}}_{=K_x} \cdot (x^{-1}K \times \{\lambda\})$ *for* $x \in X^*$ *and*

– $\underbrace{[\![\mathcal{A}]\!] \cdot \{(x, y') \mid |x| = |y'|, y'^{-1}L = y^{-1}L, x \in K\}}_{=L_y} \cdot (\{\lambda\} \times y^{-1}L)$ *for* $y \in X^*$.

Since K and L are regular, we have the following:

1. *There are only finitely many sets $x^{-1}K$ and $y^{-1}L$, i.e., the union above is finite.*
2. *For any $x \in X^*$, the set of words x' with $x'^{-1}K = x^{-1}K$ is regular (and similarly for L). Hence the relations K_x and L_y can be accepted by synchronous transducers.*

It will be convenient to have a shorthand for "can be constructed from L using synchronous rational transductions and Boolean operations":

Definition 4.3. For a language L, let $\mathcal{STB}_L$ denote the class of all languages (over arbitrary alphabets) that can be constructed from L by synchronous rational transductions and Boolean operations.

4.1 Languages that generate undecidable languages

In this section, we identify languages L such that $\mathcal{STB}_L$ contains complicated languages.

Definition 4.4. Let $L \subseteq X^*$. A *neutral word for L* is a word $v \in X^+$ such that $uvw \in L$ if and only if $uw \in L$ for any $u, w \in X^*$.

An example of non-regular languages with a neutral word is the language $\{w \in \{0,1\}^* \mid |w|_0 = |w|_1\}$ with neutral word 01 or, more generally, every identity language of a finitely generated group. Another example is the one-sided Dyck language, which consists of all $w \in \{a,b\}^*$ such that $|w|_a = |w|_b$ and for every prefix p of w, we have $|p|_a \geq |p|_b$. Here, ab is a neutral word. Note that the alphabet of every non-regular language with a neutral word contains at least two letters.

The following main result of this section is similar to Theorem 3.1.

Theorem 4.5. *Let $L \subseteq \{0,1\}^*$ be a non-regular language with a neutral word. Using synchronous rational transductions and Boolean operations, one can construct a non-recursively-enumerable language from L. If, in addition, L is recursive, one can construct for each $n \in \mathbb{N}$, a Σ_n-hard language from L.*

Remark 4.6. There are a few differences between this theorem and Theorem 3.1:

- We allow only synchronous rational transductions.
- We construct only some arithmetical languages, but these languages are arbitrarily high in the arithmetical hierarchy.
- We do not show that a fixed finite set of synchronous rational transductions suffices. In our proof, the used synchronous rational transductions will depend on the language L and the level n we want to reach.

The rest of Section 4.1 prepares the proof of Theorem 4.5. This proof can be found on page 27.

4.1.1 Languages with neutral words and synchronous rational transductions

The aim of this section is to rescue as much as possible from the proof of Theorem 3.1, namely Lemma 3.14. To this end, we observe that its proof only uses a certain type of transducers, which we call end-erasing. Furthermore we show that, in the presence of neutral words, transducers of this type can be replaced by synchronous rational transductions.

Suppose $\mathcal{A} = (Q, X^* \times X^*, E, q_0, Q_f)$ is a transducer over an arbitrary alphabet X. We call $\mathcal{A}$ *end-erasing* if $E \subseteq Q \times (X \cup \{\lambda\}) \times (X \cup \{\lambda\}) \times Q$ and if $(p, (\lambda, x), q), (q, (y, z), r) \in E$ implies $y = \lambda$. In other words, if the transducer outputs nothing on the first tape in one step, it will never output anything again on the first tape. In particular, every computation consists of two parts: The first part uses only edges with labels $X \times (X \cup \{\lambda\})$ and the second part uses only those with labels in $\{\lambda\} \times (X \cup \{\lambda\})$. Transductions generated by an end-erasing transducer are also called *end-erasing*. The following lemma will allow us to replace the transductions used in the proof of Lemma 3.14 by synchronous rational transductions (provided L has a neutral word).

Lemma 4.7. *Let X be any alphabet and $L \subseteq X^*$ be a language with a neutral word and T be an end-erasing rational transduction. Then there is a synchronous rational transduction S such that $SL = TL$.*

Proof. We call a transducer *non-erasing* if the label of each of its edges belongs to $X \times (X \cup \{\lambda\})$. A transduction is *non-erasing* if it is generated by a non-erasing transducer. Then, a transduction is end-erasing if and only if it can be written as a finite union of transductions $R \cdot (\{\lambda\} \times L)$, where R is a non-erasing transduction and L is a regular language. Using Example 4.2, it suffices to prove the lemma in the case that T is non-erasing.

Let $\mathcal{A}$ be a non-erasing transducer for T and let $w \in X^+$ be a neutral word for L. We transform $\mathcal{A}$ into a transducer $\mathcal{A}'$ as follows. At each state of $\mathcal{A}$, we attach a cycle that reads the pair (λ, w). Then $[\![\mathcal{A}]\!] \subseteq [\![\mathcal{A}']\!]$ implies $[\![\mathcal{A}]\!]L \subseteq [\![\mathcal{A}']\!]L$. For the other inclusion, let $u \in [\![\mathcal{A}']\!]L$, i.e., there is $v \in L$ with $(u,v) \in [\![\mathcal{A}']\!]$. Consider an accepting path in $\mathcal{A}'$ that is labeled by (u,v). By the construction of $\mathcal{A}'$, we can write v as $v = v_0wv_1wv_2\dots wv_n$ for some words $v_1, v_2, \dots, v_n$ such that $(u, v_0v_1\dots v_n) \in [\![\mathcal{A}]\!]$. Since w is a neutral word for L, we get $v_0v_1\dots v_n \in L$ and therefore $u \in [\![\mathcal{A}]\!]L$. Consequently, $[\![\mathcal{A}]\!]L = [\![\mathcal{A}']\!]L$.

Suppose $q_0(x_0,y_0)q_1 \cdots (x_n,y_n)q_n$ is a path in a transducer. The *delay* of this path is

$$\max_{0 \le i \le n} \{||x_0 \cdots x_i| - |y_0 \cdots y_i||\}.$$

If $\mathcal{B}$ is a transducer, then the *delay of*
autB is the maximal delay in an accepting path of $\mathcal{B}$. Moreover, for $k \in \mathbb{N}$, we write $[\![\mathcal{B}]\!]_k$ for the set of labels of accepting paths of $\mathcal{B}$ with delay at most k.

We let $k = |w|$ and claim that $[\![\mathcal{A}']\!]_kL = [\![\mathcal{A}']\!]L$. Since $[\![\mathcal{A}']\!]_k \subseteq [\![\mathcal{A}']\!]$ by definition, we have to show $[\![\mathcal{A}']\!]L \subseteq [\![\mathcal{A}']\!]_kL$. Given a word $u \in [\![\mathcal{A}']\!]L = [\![\mathcal{A}]\!]L$, we consider a path labeled (u,v) in $\mathcal{A}$ with $v \in L$. Since $\mathcal{A}$ is non-erasing, in each state, the output word produced so far is at least as long as the read input word. We turn this path into a path of $\mathcal{A}'$ as follows. Whenever the delay grows to k, we execute a cycle (λ, w), which reduces the length difference to 0. This new path exists in $\mathcal{A}'$ and has delay at most k since, in every step in $\mathcal{A}$, the delay can grow by at most one. Since w is a neutral word, the path also reads a word from L, so that we have $u \in [\![\mathcal{A}']\!]_kL$.

Clearly, one can turn $\mathcal{A}'$ into a transducer $\mathcal{A}''$ that has delay at most k and satisfies $[\![\mathcal{A}'']\!] = [\![\mathcal{A}']\!]_k$. In particular, we have $[\![\mathcal{A}'']\!]L = [\![\mathcal{A}']\!]_kL = TL$. Since every transducer with finite delay generates a synchronous rational transduction [15], this completes our proof. □

Remark 4.8. One can show that Lemma 4.7 fails in general for languages without a neutral word. Take for instance the language $L = \{a^nb^n \mid n \ge 0\}$ (which does not have a neutral word). There is an end-erasing rational transduction T such that $TL = \{a^nc^mb^n \mid n,m \ge 0\}$. On the other hand, using pumping arguments one can show that there is no synchronous rational transduction S such that $SL = \{a^nc^mb^n \mid n,m \ge 0\}$.

Lemma 4.9. *Let $L \subseteq \{0,1\}^*$ be a (possibly regular) language with a neutral word. Then $\hat{C}_L$ belongs to $\mathcal{STB}_L$.*

Proof. An inspection of the proof of Lemma 3.14 shows that the rational transductions $T_1, \dots, T_{19}$, which, together with Boolean operations, are used to construct $\hat{C}_L$ are all end-erasing. Furthermore, every language obtained in the process has a neutral word, namely the neutral word of L. Hence, by Lemma 4.7, $\hat{C}_L \in \mathcal{STB}_L$.
□

4.1.2 Construction of encodings of recursively enumerable relations

Notation We consider the alphabets $\mathbb{B} = \{0, 1\}$, $\varDelta = \{+, -, z\}$ and $Y = \mathbb{B} \cup \varDelta \cup \{\#\}$. For $i \in \mathbb{N}$, define $\mathbb{B}_i = \{0_i, 1_i\}$, $\varDelta_i = \{+_i, -_i, z_i\}$, and $Y_i = \mathbb{B}_i \cup \varDelta_i \cup \{\#_i\}$. Then we have a homomorphism $h_i : Y^* \to Y_i^*$ given by $h_i(y) = y_i$ for all $y \in Y$.

Analogous to the proof of Lemma 3.14, we assign to a language $L \subseteq \mathbb{B}^*$ the language

$$\begin{aligned} S_L &= \{v_0{+}v_1{+}v_2 \cdots {+}v_m \mid v_i \in \mathbb{B}^*, v_i \not\equiv_L v_j \text{ for all } 0 \le i < j \le m\} \\ &= \{w \mid w \in (\{+\} \cup \mathbb{B})^*, \exists y \in Y^* : w\#y \in \hat{C}_L\} . \end{aligned}$$

Note that the language S_L here differs from the language S of Lemma 3.14 only by replacing # with +. A word $w = v_0{+}v_1{+}v_2 \cdots {+}v_m$ (with $v_0, v_1, \ldots, v_m \in \mathbb{B}^*$) from this language will serve as an encoding of the natural number $m = |w|_+$. In the same spirit, a word $h_1(w_1)h_2(w_2) \cdots h_n(w_n)$ (with $w_1, w_2, \ldots, w_n \in S_L$) will serve as an encoding of the tuple $(|w_1|_+, |w_2|_+, \ldots, |w_n|_+) \in \mathbb{N}^n$.

Definition 4.10. Let $A \subseteq \mathbb{N}^n$ be some numerical relation and let $L \subseteq \mathbb{B}^*$ be some language. The *encoding of A wrt. L* is the language

$$\begin{aligned} \mathrm{Enc}_{A,L} = \{h_1(w_1)h_2(w_2) \cdots h_n(w_n) \mid\ & w_1, w_2, \ldots, w_n \in S_L, \\ & (|w_1|_+, |w_2|_+, \ldots, |w_n|_+) \in A\} . \end{aligned}$$

In this section, we show that $\mathrm{Enc}_{A,L} \in \mathcal{STB}_L$ whenever $L \subseteq \mathbb{B}^*$ is non-regular with a neutral word and $A \subseteq \mathbb{N}^n$ is recursively enumerable. Since $\mathrm{Enc}_{A,L}$ for $A \subseteq \mathbb{N}^n$ is a subset of $h_1(S_L) \cdots h_n(S_L)$, we start by showing that this language belongs to $\mathcal{STB}_L$.

Lemma 4.11. *Let $L \subseteq \mathbb{B}^*$ be a language with a neutral word. Then the languages S_L and $h_1(S_L) \cdots h_n(S_L)$ belong to $\mathcal{STB}_L$ for each $n \in \mathbb{N}$.*

Proof. An inspection of the proof of Lemma 3.14 shows that S_L can be constructed from L by end-erasing transductions and Boolean operations. In addition, any of the intermediate languages has a neutral word, namely the neutral word of L. Hence, by Lemma 4.7, the language S_L belongs to $\mathcal{STB}_L$.

Since the length-preserving homomorphisms $h_1, \ldots, h_n$ can be realized by synchronous rational transductions, the languages $h_1(S_L), \ldots, h_n(S_L)$ all belong to $\mathcal{STB}_L$. The end-erasing rational transduction

$$(Y_1^* \cdots Y_{i-1}^* \times \{\lambda\}) \cdot \{(w, w) \mid w \in Y_i^*\} \cdot (Y_{i+1}^* \cdots Y_n^* \times \{\lambda\})$$

maps $h_i(S_L)$ to $H_i = Y_1^* \cdots Y_{i-1}^* h_i(S_L) Y_{i+1}^* \cdots Y_n^*$. Since any neutral word w for L is also neutral for S_L, each language $h_i(S_L)$ also has a neutral word, namely $h_i(w)$. Consequently, Lemma 4.7 implies that also each H_i belongs to $\mathcal{STB}_L$.

Hence, the language $h_1(S_L) \cdots h_n(S_L) = \bigcap_{i=1}^n H_i$ is the intersection of n languages from $\mathcal{STB}_L$ and therefore belongs to $\mathcal{STB}_L$ as well. □

Machines with n counters are the main tool in our proof that $\mathrm{Enc}_{A,L}$ belongs to $\mathcal{STB}_L$ for recursively enumerable sets $A \subseteq \mathbb{N}^{n-2}$. An *$n$-counter machine* is a tuple $\mathcal{M} = (Q, E, q_0, Q_f)$, where Q is the finite set of *states*, $E \subseteq Q \times \bigcup_{1 \le i \le n} \varDelta_i \times Q$ is the set of *edges*, $q_0 \in Q$ is the *initial state*, and $Q_f \subseteq Q$ is the set of *accepting states*.

Tuples from $Q \times \mathbb{N}^n$ are called *configurations* of $\mathcal{M}$. The one-step relation $\vdash_{\mathcal{M}}$ is defined by

$$(p, m_1, \ldots, m_n) \vdash_{\mathcal{M}} (q, m'_1, \ldots, m'_n)$$

if there exists an edge $(p, \delta_i, q) \in E$ with $\delta_i \in \varDelta_i$ such that $m'_j = m_j$ for all $j \neq i$ and

- $\delta_i = +_i$ and $m'_i = m_i + 1$ or
- $\delta_i = -_i$ and $m'_i = m_i - 1$ or
- $\delta_i = z_i$ and $m'_i = m_i = 0$.

The machine $\mathcal{M}$ accepts the input $(m_1, \ldots, m_n) \in \mathbb{N}^n$ if there exist $q \in Q_f$ and a tuple $(n_1, \ldots, n_n) \in \mathbb{N}^n$ such that $(q_0, m_1, \ldots, m_n) \vdash^*_{\mathcal{M}} (q, n_1, \ldots, n_n)$. The crucial property of these counter machines is that a set $A \subseteq \mathbb{N}^n$ is recursively enumerable if and only if the set $A \times \{(0,0)\}$ is accepted by some $(n+2)$-counter machine [27].

Lemma 4.12. *Let $L \subseteq \mathbb{B}^*$ be some non-regular language with neutral word and let $A \subseteq \mathbb{N}^n$ be accepted by an n-counter machine. Then the encoding* $\mathrm{Enc}_{A,L}$ *of A wrt. L belongs to $\mathcal{STB}_L$.*

Proof. Let A be accepted by the n-counter machine $\mathcal{M} = (Q, E, q_0, Q_f)$. Note that $\mathcal{M}' = (Q, \left(\bigcup_{1 \le i \le n} \varDelta_i\right)^*, E, q_0, Q_f)$ is an automaton, which we call the *underlying automaton.* Then $R = [\![\mathcal{M}']\!]$ is a regular language, namely the set of all sequences of counter operations permitted by $\mathcal{M}$ irrespective of whether

- a counter has a positive value when it should be decremented and
- a counter has value 0 when it is tested for emptiness.

Intermezzo: proof idea. We explain the general idea of the proof ignoring the factors from $\mathbb{B}_i^*$: Then, the aim is to generate the language

$$\{+_1^{m_1} \cdots +_n^{m_n} \mid (m_1, \ldots, m_n) \in A\} \tag{4}$$

(which is the projection of $\mathrm{Enc}_{A,L}$ to $\{+_i \mid 1 \le i \le n\}^*$). Recall that the language C is the set of valid sequences of counter operations, i.e., this language ensures that the two conditions ignored by R are satisfied (for a single counter). Consequently, $\sqcup\!\sqcup_{1 \le i \le n} h_i(C)$ is the set of valid sequences of operations on n different counters. It follows that the language

$$+_1^{m_1} \cdots +_n^{m_n} R \cap \sqcup\!\sqcup_{1 \le i \le n} h_i(C)$$

is the set of valid counter operations where first, the n counters are initialized to m_1, m_2, etc, and then the n-counter machine $\mathcal{M}$ is started. Therefore, for any $m_1, \ldots, m_n \in \mathbb{N}$, we have the following:

$$\begin{gathered}\mathcal{M} \text{ accepts } (m_1, \ldots, m_n) \\ \iff \\ +_1^{m_1} \cdots +_n^{m_n} R \cap \sqcup\!\sqcup_{1 \le i \le n} h_i(C) \neq \emptyset\end{gathered}$$

Now suppose the language

$$+_1^* \cdots +_n^* R \cap \sqcup\!\sqcup_{1 \le i \le n} h_i(C) \tag{5}$$

can be generated from L. Then also the language from (4) can be generated since it is the image of this latter language under the synchronous rational transduction

$$\{(w,w) \mid w \in +_1^* +_2^* \cdots +_n^*\} \cdot (\{\lambda\} \times R).$$

The main tasks of the actual proof are therefore:

1. To generate the language from (5) from L.
2. To not ignore the factors from $\mathbb{B}_i^*$.

Back to the actual proof. Recall the definition of the language $\hat{C}_L$ from (1): it is the set of words $w\#y$ where $w \in (\mathbb{B} \cup \Delta)^*$ is a valid sequence of counter operations from C interspersed with words over $\mathbb{B}^*$ that encode counter values, the actual meaning of such an encoding is defined by the word $y \in (\mathbb{B} \cup \{\#\})^*$. We consider the language

$$\begin{aligned} \hat{D}_L &= \{w \in (\Delta \cup \mathbb{B})^* \mid w\#Y^* \cap \hat{C}_L \neq \emptyset\} \\ &= \{w \in (\Delta \cup \mathbb{B})^* \mid \exists y \in Y^* \colon w\#y \in \hat{C}_L\} \end{aligned}$$

of all words w of the above form (i.e., w gives a valid sequence of counter operations and the "proof of validity" is given by the maximal factors from $\mathbb{B}^*$). Then $\hat{D}_L$ is the image of $\hat{C}_L$ under the synchronous rational transduction

$$\{(w,w) \mid w \in (\mathbb{B} \cup \Delta)^*\} \cdot (\{\lambda\} \times \#Y^*)\,.$$

Since $\hat{C}_L \in \mathcal{STB}_L$ by Lemma 4.9, we obtain $\hat{D}_L \in \mathcal{STB}_L$. It follows that also $h_i(\hat{D}_L) \in \mathcal{STB}_L$ for all $1 \leq i \leq n$.

Consider the projection $\pi_i \colon \left(\bigcup_{1\leq i\leq n} Y_i\right)^* \to Y_i^*$ defined by

$$\pi_i(y) = \begin{cases} y & \text{if } y \in Y_i \\ \lambda & \text{if } y \in \displaystyle\bigcup_{1\leq j\leq n, j\neq i} Y_j \end{cases}$$

Then π_i^{-1} can be realized by an end-erasing transducer since it nondeterministically inserts factors over $\bigcup_{1\leq j\leq n, j\neq i} Y_j$ into words over Y (whose letters are first indexed by i). Since $\hat{D}_L$ has a neutral word (e.g. the neutral word of L), so does $h_i(\hat{D}_L)$. Hence, by Lemma 4.7, there is a synchronous rational transduction mapping $h_i(\hat{D}_L)$ to $\pi_i^{-1}(h_i(\hat{D}_L))$, i.e., this language belongs to $\mathcal{STB}_L$. It follows that also the language

$$H = \bigcap_{1\leq i\leq n} \pi_i^{-1}(h_i(\hat{D}_L))$$

belongs to $\mathcal{STB}_L$ since it is the intersection of n languages from $\mathcal{STB}_L$.

We want to select words from H that correspond to computations of $\mathcal{M}$. To this end, recall that the language R of the underlying automaton $\mathcal{M}'$ is regular. Intersecting R and H directly makes little sense: R encodes computations using only operations in Δ, whereas H also contains representatives for $\equiv_L$-classes. Therefore, we pad R with words from $\mathbb{B}_i^*$. Let R' be the set of all words

$$\delta_1 w_1 \cdots \delta_k w_k\,,$$

where $\delta_1 \cdots \delta_k \in R$ and for each $j \in \{1, \ldots, k\}$, there is an $i \in \{1, \ldots, n\}$ with $\delta_j \in \Delta_i$ and $w_j \in \mathbb{B}_i^*$. In other words, after each symbol $\delta \in \Delta_i$ in a word from R, we add a word from $\mathbb{B}_i^*$. Clearly, R' is regular. Therefore, the language

$$h_1(S_L) \cdots h_n(S_L)\, R'$$

is the image of $h_1(S_L) \cdots h_n(S_L)$ under a synchronous rational transduction. Since $h_1(S_L) \cdots h_n(S_L) \in \mathcal{STB}_L$ by Lemma 4.11, also the language

$$h_1(S_L) \cdots h_n(S_L)\, R' \cap H \tag{6}$$

belongs to $\mathcal{STB}_L$ since it is the intersection of two languages from $\mathcal{STB}_L$ (note that the projection of this language to the alphabet $\bigcup_{1 \le i \le n} \Delta_i$ is the language from (4)). A word belongs to this language if

- its projection to the counter operations $\bigcup_{1 \le i \le n} \Delta_i$ first increments the counters $1, \ldots, n$ to some values and then follows some accepting path in the underlying automaton $\mathcal{M}'$,
- its projection to the counter operations is valid,
- the "proof of validity" is provided by the maximal factors from $\bigcup_{1 \le i \le n} \mathbb{B}_i^*$, and
- these maximal factors from $\mathbb{B}_i^*$ (apart from the first one) immediately follow some counter operation of counter i.

Consequently, also the language

$$K = \{w \in h_1(S_L) \cdots h_n(S_L) \mid \exists y \in R' \colon wy \in H\}$$

belongs to $\mathcal{STB}_L$ since it can be obtained from the language from (6) by the synchronous rational transduction

$$\{(w, w) \mid w \in Z_1^* \cdots Z_n^*\} \cdot (\{\lambda\} \times R'),$$

where $Z_i = \{+_i, 0_i, 1_i\}$ for $1 \le i \le n$. We claim that $K = \mathrm{Enc}_{A,L}$:

- Suppose $w \in K$. There are $w_1, \ldots, w_n \in S_L$ with $w = h_1(w_1) \cdots h_n(w_n)$. For $1 \le i \le n$, let $m_i = |w_i|_+$. Since $w \in K$, there is a $y \in R'$ such that $wy \in H$. Let z be the sequence of counter operations in y, i.e., $z = \pi_{\{+_i, -_i, z_i \mid 1 \le i \le n\}}(y)$. Then $z \in R$ and

$$\pi_{\{+_i, -_i, z_i \mid 1 \le i \le n\}}(wy) = +_1^{m_1} \cdots +_n^{m_n} z$$

  belongs to $\bigsqcup_{1 \le i \le n} h_i(C)$ (since wy belongs to H). Since $z \in R$, it follows that z is the sequence of counter operations of an accepting computation of $\mathcal{M}$ with input $(m_1, \ldots, m_n)$. Thus, the tuple $(m_1, \ldots, m_n)$ belongs to A. This ensures $w \in \mathrm{Enc}_{A,L}$ by the very definition.
- Conversely suppose $w \in \mathrm{Enc}_{A,L}$. There are $w_1, \ldots, w_n \in S_L$ such that we have $w = h_1(w_1) \cdots h_n(w_n)$. For $1 \le i \le n$, let $m_i = |w_i|_+$. Since $w \in \mathrm{Enc}_{A,L}$, there is an accepting computation of $\mathcal{M}$ with input $(m_1, \ldots, m_n)$. Let z be the sequence of counter operations of this computation. Then the word

$$+_1^{m_1} \cdots +_n^{m_n} z$$

  belongs to $\bigsqcup_{1 \le i \le n} h_i(C)$. Since this sequence of counter operations is valid, we find $y = \delta_1 w_1 \cdots \delta_k w_k \in R'$ with $z = \delta_1 \cdots \delta_k$ such that $wy \in H$. Hence, we have $w \in K$.

□

Lemma 4.13. *Let $L \subseteq \mathbb{B}^*$ be nonempty and suppose that $H \subseteq Y_1^* Y_2^* \cdots Y_n^*$ belongs to $\mathcal{STB}_L$. Then also*

$$H_\exists = \{w \in Y_1^* Y_2^* \cdots Y_{n-1}^* \mid \exists y \in Y_n^* \colon wy \in H\}$$

and

$$H_\forall = \{w \in Y_1^* Y_2^* \cdots Y_{n-1}^* \mid \forall y \in Y_n^* \colon wy \in H\}$$

belong to $\mathcal{STB}_L$.

Proof. Note that $H_\exists$ is the image of H under the synchronous rational relation

$$\left\{(w,w) \mid w \in \left(\bigcup_{1 \le i < n} Y_i\right)^*\right\} \cdot (\{\lambda\} \times Y_n^*)$$

and does therefore belong to $\mathcal{STB}_L$.

Since L is nonempty, any regular language K is the image of L under the synchronous rational relation $K \times \left(\bigcup_{1 \le i \le n} Y_i\right)^*$, i.e., any regular language belongs to $\mathcal{STB}_L$. This applies in particular to the regular language $Y_1^* Y_2^* \cdots Y_n^*$, hence

$$Y_1^* Y_2^* \cdots Y_n^* \setminus H$$

belongs to $\mathcal{STB}_L$. It follows as above that

$$\{w \in Y_1^* Y_2^* \cdots Y_{n-1}^* \mid \exists y \in Y_n^* \colon wy \notin H\}$$

belongs to $\mathcal{STB}_L$. From

$$H_\forall = Y_1^* Y_2^* \cdots Y_{n-1}^* \setminus \{w \in Y_1^* Y_2^* \cdots Y_{n-1}^* \mid \exists y \in Y_n^* \colon wy \notin H\},$$

we get $H_\forall \in \mathcal{STB}_L$ as a Boolean combination of languages from $\mathcal{STB}_L$. □

Lemma 4.14. *Let $L \subseteq \mathbb{B}^*$ be some non-regular language with a neutral word and let $A \subseteq \mathbb{N}^n$ be recursively enumerable. Then the encoding $\mathrm{Enc}_{A,L}$ of A wrt. L belongs to $\mathcal{STB}_L$.*

Proof. Since $A \subseteq \mathbb{N}^n$ is recursively enumerable, there is an $(n+2)$-counter machine $\mathcal{M}$ accepting $A' = A \times \{(0,0)\}$ [27]. Then, we have

$$\mathrm{Enc}_{A,L} \subseteq Y_1^* \cdots Y_n^*, \quad \mathrm{Enc}_{A',L} \subseteq Y_1^* \cdots Y_{n+2}^*,$$

and, in the notation of Lemma 4.13, we have $\mathrm{Enc}_{A,L} = ((\mathrm{Enc}_{A',L})_\exists)_\exists$. Hence, according to Lemma 4.12, the language $\mathrm{Enc}_{A',L}$ belongs to $\mathcal{STB}_L$. By Lemma 4.13, the same is true of $\mathrm{Enc}_{A,L}$. □

4.1.3 Construction of Σ_n-hard languages

Theorem 4.15. *Let $L \subseteq \mathbb{B}^*$ be some non-regular and recursive language with a neutral word and let $n \in \mathbb{N}$. There exists a Σ_n-hard language in $\mathcal{STB}_L$. If, in addition, the Myhill-Nerode equivalence $\equiv_L$ is recursive, then $\mathcal{STB}_L$ contains even some Σ_n-complete language.*

Proof. We only spell out the proof for $n = 3$, the general case can be easily deduced from our exposition (but is notationally cumbersome and therefore omitted here). Let $K \subseteq \mathbb{N}$ be some Σ_3-complete set. We prove that the language $\mathrm{Enc}_{K,L}$ belongs to $\mathcal{STB}_L$ and is Σ_3-hard (Σ_3-complete if $\equiv_L$ is decidable).

Since K is in Σ_3, there exists a recursive relation $P \subseteq \mathbb{N}^4$ such that

$$K = \{m_1 \mid \exists m_2 \forall m_3 \exists m_4 \colon (m_1, m_2, m_3, m_4) \in P\}\,.$$

We consider the following relations:

$$\begin{aligned}
K_0 &= \{(m_1, m_2, m_3) \in \mathbb{N}^3 \mid \exists m_4 \colon (m_1, m_2, m_3, m_4) \in P\} \\
K_1 &= \{(m_1, m_2) \in \mathbb{N}^2 \mid \forall m_3 \exists m_4 \colon (m_1, m_2, m_3, m_4) \in P\} \\
&= \{(m_1, m_2) \in \mathbb{N}^2 \mid \forall m_3 \colon (m_1, m_2, m_3) \in K_0\} \\
K_2 &= \{m_1 \in \mathbb{N} \mid \exists m_2 \forall m_3 \exists m_4 \colon (m_1, m_2, m_3, m_4) \in P\} \\
&= \{m_1 \in \mathbb{N} \mid \exists m_2 \colon (m_1, m_2) \in K_1\}
\end{aligned}$$

We will show that $\mathrm{Enc}_{K_i,L} \in \mathcal{STB}_L$ for all $0 \leq i \leq 2$. Since $K = K_2$, this will in particular imply $\mathrm{Enc}_{K,L} \in \mathcal{STB}_L$.

Since P is recursive, the relation K_0 is recursively enumerable. Hence, by Lemma 4.14, the language $\mathrm{Enc}_{K_0,L}$ belongs to $\mathcal{STB}_L$. Note that

$$\begin{aligned}
\mathrm{Enc}_{K_1,L} = \{w \in h_1(w_1)h_2(w_2) \mid\ & w_1, w_2 \in S_L, \\
& \forall w_3 \in S_L \colon h_1(w_1)h_2(w_2)h_3(w_3) \in \mathrm{Enc}_{K_0,L}\} \\
= (\mathrm{Enc}_{K_0,L})_\forall\,.&
\end{aligned}$$

Hence, by Lemma 4.13, $\mathrm{Enc}_{K_1,L}$ belongs to $\mathcal{STB}_L$.

Similarly note that

$$\begin{aligned}
\mathrm{Enc}_{K_2,L} = \{w \in h_1(w_1) \mid\ & w_1 \in S_L, \\
& \exists w_2 \in S_L \colon h_1(w_1)h_2(w_2) \in \mathrm{Enc}_{K_1,L}\} \\
= (\mathrm{Enc}_{K_1,L})_\exists\,.&
\end{aligned}$$

Again by Lemma 4.13, $\mathrm{Enc}_{K_2,L} = \mathrm{Enc}_{K,L}$ belongs to $\mathcal{STB}_L$.

Next, we show that $\mathrm{Enc}_{K,L}$ is Σ_3-hard, i.e., that we can reduce K to $\mathrm{Enc}_{K,L}$. So let $m \in \mathbb{N}$. Since L is recursive[2] and non-regular, we can compute words $x_0, x_1, \ldots, x_m$ that are mutually non-equivalent with respect to the Myhill-Nerode-equivalence $\equiv_L$. Let

$$w = h_1(x_0 + x_1 + \cdots + x_m)\,.$$

Then $m \in K$ if and only if $w \in \mathrm{Enc}_{K,L}$.

Finally, suppose that $\equiv_L$ is decidable. We demonstrate that then, $\mathrm{Enc}_{K,L}$ can be reduced to K: First, fix some number $n \notin K$. Furthermore, let $w \in Y_1^*$ be

[2] This is the only point were we need the recursiveness of L.

an arbitrary word. Then we have $w = h_1(v)$ for some (unique) $v \in Y^*$. The recursiveness of $\equiv_L$ allows us to decide whether $v \in S_L$. If this is not the case, we map the word w to n. Otherwise, we map w to $m = |v|_+$. Then, by the very definition, $w \in \mathrm{Enc}_{K,L}$ iff m belongs to K. $\square$

Remark 4.16. Note that when L is recursive, $\equiv_L$ is not necessarily recursive. In fact, there are even context-free languages with an undecidable Myhill-Nerode equivalence. For instance, there exist fixed alphabets X, Y and homomorphisms $\alpha, \beta \colon X^* \to Y^*$ such that given $u \in Y^*$, it is undecidable whether there is a $v \in X^*$ with $u\alpha(v) = \beta(v)$ [19]. Consider the language

$$\begin{aligned} L =& \{u\$\alpha(v)\#\beta(v)^{\mathrm{rev}} \mid u \in Y^*, v \in X^*\} \\ & \cup \{usv\#w^{\mathrm{rev}} \mid s \in \{\$, \#\}, u, v, w \in Y^*, uv \neq w\}, \end{aligned}$$

where w^{rev} denotes w in reverse. The language L is (linear) context-free and we have $u\$ \not\equiv_L u\#$ if and only if there is a $v \in X^*$ with $u\alpha(v) = \beta(v)$. Hence, $\equiv_L$ is undecidable. On the other hand, the identity language of any infinite group with a decidable word problem satisfies all conditions in Theorem 4.15.

4.1.4 Proof of Theorem 4.5

Let $L \subseteq \{0,1\}^*$ be a non-regular language with a neutral word. If L is not recursively enumerable, then $\mathcal{STB}_L$ contains (trivially) some language that is not recursively enumerable (namely, L). If L is recursively enumerable, but not recursive, then the complement of L is not recursively enumerable and an element of $\mathcal{STB}_L$. It remains to consider the case when L is recursive. But then, by Theorem 4.15, $\mathcal{STB}_L$ contains some Σ_n-hard language that cannot be recursively enumerable. This finishes the proof of the first claim. The second is Theorem 4.15. $\square$

4.2 Languages that generate only recursive languages

The main tool in this section will be Büchi-automata or, more precisely, regular ω-languages. For an alphabet X, let X^ω denote the set of *ω-words*, i.e., of sequences $a_0a_1a_2\cdots$ of elements of X. Subsets of X^ω are called *ω-languages.*

A *Büchi-automaton over* X is an automaton $\mathcal{A} = (Q, X^*, E, q_0, Q_f)$ over the monoid X^* with $E \subseteq Q \times X \times Q$, i.e., edges are labeled by single letters. An *infinite path from* p_0 in $\mathcal{A}$ is a sequence $p_0\, a_1\, p_1\, a_2\, p_2 \cdots$ with $(p_i, a_{i+1}, p_{i+1}) \in E$ for all $i \geq 0$, its *label* is the ω-word $a_1a_2\cdots$. The path is *accepting* if $p_0 = q_0$ and there are infinitely many $i \geq 0$ with $p_i \in Q_f$. The ω-language accepted by $\mathcal{A}$ is the set

$$[\![\mathcal{A}]\!]_\omega = \{u \in X^\omega \mid \text{ there is some accepting path in } \mathcal{A} \text{ labeled } u\}.$$

An ω-language L is *regular* if there is some Büchi-automaton $\mathcal{A}$ over X such that $L = [\![\mathcal{A}]\!]_\omega$.

In order to state the main technical property, we need a final definition: Let $u_i = a_{0,i}a_{1,i}\cdots$ be ω-words over the alphabet X for $1 \leq i \leq k$. Then we define the ω-word

$$\otimes(u_1, u_2, \ldots, u_k) = (a_{0,1}, a_{0,2}, \ldots, a_{0,k})(a_{1,1}, a_{1,2}, \ldots, a_{1,k})(a_{2,1}, a_{2,2}, \ldots, a_{2,k})\cdots$$

over the alphabet X^k. For $\otimes(u_1, u_2)$, we often write $u_1 \otimes u_2$. For an alphabet X with $\Box \notin X$, we set $X^{(k)} = (X \cup \{\Box\})^k$.

Lemma 4.17. *Let X be an alphabet with $\Box \notin X$, let $u \in X^\omega$ and let $P_u \subseteq X^*$ be its set of prefixes. Let the language K be constructed from P_u using synchronous rational transductions and Boolean operations. From a description of this construction one can effectively construct a Büchi-automaton $\mathcal{A}_K$ over $X^{(2)}$ such that, for all words $w \in X^*$, we have*

$$u \otimes (w\Box^\omega) \in [\![\mathcal{A}_K]\!]_\omega \iff w \in K\,. \tag{7}$$

Proof. We prove the lemma by induction on the construction of K from P_u. If $K = P_u$, then $w \in K$ if and only if

$$u \otimes (w\Box^\omega) \in \{(a,a),(a,\Box) \mid a \in X\}^\omega\,.$$

The ω-language $\{(a,a),(a,\Box) \mid a \in X\}^\omega$ is effectively regular. Hence, one can construct a Büchi-automaton $\mathcal{A}_{P_u}$ that describes the language P_u in the sense of (7).

For the induction step, suppose we have Büchi-automata $\mathcal{A}_{K_1}$ and $\mathcal{A}_{K_2}$ that describe the languages K_1 and K_2, resp., in the sense of (7).

The class of regular ω-languages is effectively closed under Boolean operations. Hence we can construct a Büchi-automaton $\mathcal{A}_K$ describing K from $\mathcal{A}_{K_1}$ and $\mathcal{A}_{K_2}$ if $K = K_1 \cup K_2$, $K = K_1 \cap K_2$ or $K = \alpha(K_1)^* \setminus K_1$.

It remains to consider the case $K = RK_1$ where R is a synchronous rational transduction. Then R is the union of finitely many synchronous rational transductions R_i of the form

$$[\![\mathcal{A}]\!] \cdot (L \times \{\lambda\}) \text{ or } [\![\mathcal{A}]\!] \cdot (\{\lambda\} \times L)\,, \tag{8}$$

where $\mathcal{A}$ is a synchronous transducer over X and $L \subseteq X^*$ is a regular language (see Definition 4.1). Since $K = RK_1 = \bigcup_{1 \le i \le n} R_i K_1$, it suffices to consider R to be of the form (8). We will handle the case

$$R = [\![\mathcal{A}]\!] \cdot (L \times \{\lambda\});$$

the other one can be dealt with analogously. From the synchronous transducer $\mathcal{A}$, an automaton accepting the language L, and the Büchi-automaton $\mathcal{A}_{K_1}$, we can easily build a Büchi-automaton $\mathcal{A}'$ over $X^{(3)}$ accepting the set of ω-words of the form

$$\otimes(x, y\Box^\omega, z\Box^\omega)$$

with $x \in X^\omega$ and $y, z \in X^*$ such that

– $x \otimes (z\Box^\omega)$ is accepted by the Büchi-automaton $\mathcal{A}_{K_1}$ and
– $(y, z) \in R$.

Let $\mathcal{A}_K$ be obtained from $\mathcal{A}'$ by projecting away the third component of every letter.

Now let $w \in X^*$. Then we have $u \otimes (w\Box^\omega) \in [\![\mathcal{A}_K]\!]_\omega$ if and only if there is some finite word $z \in X^*$ with $\otimes(u, w\Box^\omega, z\Box^\omega) \in [\![\mathcal{A}']\!]_\omega$. But this is equivalent to saying "there is some word $z \in K_1$ with $(w, z) \in R$". Hence, indeed, the Büchi-automaton $\mathcal{A}_K$ describes the language K in the sense of (7). $\Box$

Note that the Büchi-automaton $\mathcal{A}_K$ does not depend on the ω-word u, but only on the expression that constructs K from P_u. In other words, the same Büchi-automaton $\mathcal{A}_K$ works for all ω-words u.

We now come to the main result of this section.

Theorem 4.18. *There is a non-regular language $L \subseteq X^*$ such that for languages constructed from L using synchronous rational transductions and Boolean operations, emptiness is decidable. Furthermore, all these languages are (effectively) recursive.*

A remark regarding the decidability of emptiness is in order: In order for this statement to make sense, we have to agree on a finite description of languages from $\mathcal{STB}_L$. Here, we chose an expression using Boolean operations, synchronous rational transductions (which are represented by synchronous transducers) and the language L as a constant. Given such an expression, emptiness of the resulting language K is decidable and a Turing machine deciding K can be constructed.

Proof of Theorem 4.18. There exists an ω-word u such that the following hold:

– The ω-word u is not ultimately periodic, i.e., the language P_u of prefixes of u is not regular.
– The set of Büchi-automata $\mathcal{A}$ with $u \in [\![\mathcal{A}]\!]_\omega$ is decidable.

(Examples of such ω-words are the characteristic sequences of the set of factorial numbers $n!$, of the set of k-th powers n^k for any fixed k, of the set of powers of k [12], and of the set of Fibonacci numbers and, more generally, any not ultimately periodic morphic ω-word [7]. See [38,33,32] for complete characterizations of these ω-words.) Now let $K \subseteq X^*$ be constructed from $L = P_u$ using synchronous rational transductions and Boolean operations. Then, by the above lemma, we can construct a Büchi-automaton $\mathcal{A}_K$ such that, for any word $w \in X^*$, we have

$$u \otimes (w\square^\omega) \in [\![\mathcal{A}_K]\!]_\omega \iff w \in K\,.$$

From $\mathcal{A}_K$, we can build a Büchi-automaton $\mathcal{A}$ with

$$u \in [\![\mathcal{A}]\!]_\omega \iff \exists w \in X^* \colon u \otimes (w\square^\omega) \in [\![\mathcal{A}_K]\!]_\omega.$$

Since it is decidable whether u is accepted by a given Büchi-automaton, we can decide whether $u \in [\![\mathcal{A}]\!]_\omega$, i.e. whether $K \neq \emptyset$. Since intersecting with the regular language $\{w\}$ can be realized by a synchronous rational transduction, this means, in particular, that all languages obtained from L by synchronous rational transductions and Boolean operations are recursive. □

5 Rational Kripke frames

Theorem 3.1 can be also restated in terms of multimodal logic. See [4] for more details on modal logic. A *Kripke structure* (or edge- and node-labeled graph) is a tuple

$$\mathcal{K} = (V, (E_a)_{a\in A}, (U_p)_{p\in P}),$$

where V is a set of nodes (also called worlds), A and P are finite sets of actions and propositions, respectively, for every $a \in A$, $E_a \subseteq V \times V$, and for every $p \in P$, $U_p \subseteq V$. The tuple $\mathcal{F} = (V, (E_a)_{a\in A})$ is then also called a *Kripke frame*. We say that $\mathcal{K}$ (and $\mathcal{F}$) is *word-based* if $V = X^*$ for some finite alphabet X. Formulas of *multimodal logic* are defined by the following grammar, where $p \in P$ and $a \in A$:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box_a\varphi \mid \Diamond_a\varphi.$$

The semantics $[\![\varphi]\!]_{\mathcal{K}} \subseteq V$ of formulas φ in $\mathcal{K}$ is defined inductively as follows:

$$\begin{aligned}
[\![p]\!]_{\mathcal{K}} &= U_p,\\
[\![\neg\varphi]\!]_{\mathcal{K}} &= V \setminus [\![\varphi]\!]_{\mathcal{K}},\\
[\![\varphi \wedge \psi]\!]_{\mathcal{K}} &= [\![\varphi]\!]_{\mathcal{K}} \cap [\![\psi]\!]_{\mathcal{K}},\\
[\![\varphi \vee \psi]\!]_{\mathcal{K}} &= [\![\varphi]\!]_{\mathcal{K}} \cup [\![\psi]\!]_{\mathcal{K}},\\
[\![\Box_a\varphi]\!]_{\mathcal{K}} &= \{v \in V \mid \forall u \in V\colon (v,u) \in E_a \to u \in [\![\varphi]\!]_{\mathcal{K}}\},\\
[\![\Diamond_a\varphi]\!]_{\mathcal{K}} &= \{v \in V \mid \exists u \in V\colon (v,u) \in E_a \wedge u \in [\![\varphi]\!]_{\mathcal{K}}\}.
\end{aligned}$$

A word-based Kripke frame $\mathcal{F} = (X^*, (E_a)_{a\in A})$ is called *rational* if every E_a is a rational transduction. Rational Kripke frames with a single relation are also known as *rational graphs* and have been studied intensively [6,28,29]. A word-based Kripke structure $\mathcal{K} = (X^*, (E_a)_{a\in A}, (U_p)_{p\in P})$ is called *rational* if every relation E_a is a rational transduction and every U_p is a regular language. The closure properties of regular languages imply that for every rational Kripke structure $\mathcal{K}$ and every multimodal formula φ, the set $[\![\varphi]\!]_{\mathcal{K}}$ is a regular language that can be effectively constructed from φ and (automata describing the structure) $\mathcal{K}$. Using this fact, Bekker and Goranko [2] proved that the *model-checking problem* for rational Kripke structures and multimodal logic is decidable. This problem has as input a rational Kripke structure $\mathcal{K}$ (given by a tuple of automata and transducers), a word $w \in X^*$ (where X^* is the node set of $\mathcal{K}$), and a multimodal formula φ, and it is asked whether $w \in [\![\varphi]\!]_{\mathcal{K}}$ holds. In contrast, there exist rational graphs (even acyclic ones) with an undecidable first-order theory [6,39], but every rational tree has a decidable first-order theory [6]. Rational Kripke structures and frames were also considered in the context of querying graph databases [1].

Our reformulation of Theorem 3.1 in terms of multimodal logic is:

Theorem 5.1. *Let $X = \{0,1\}$. There are rational transductions E_r, E_s, E_t in X^* such that the rational Kripke frame $\mathcal{F} = (X^*, E_r, E_s, E_t)$ has the following property: For every non-regular language $U_p \subseteq X^*$ and every language $K \in \mathsf{AH}(U_p)$, $K \subseteq X^*$, there exists a multimodal formula φ such that $K = [\![\varphi]\!]_{\mathcal{K}}$, where $\mathcal{K} = (X^*, E_r, E_s, E_t, U_p)$.*

Proof. Take the rational transductions R, S, T provided by Theorem 3.1. Suppose $U_p \subseteq X^*$ is a non-regular language and let $\mathcal{K} = (X^*, E_r, E_s, E_t, U_p)$ be the Kripke structure with $E_r = R$, $E_s = S$, and $E_t = T$. By induction, we can construct for every language K obtainable from U_p by the transductions R, S, T and Boolean operations a multimodal formula φ with $K = [\![\varphi]\!]_{\mathcal{K}}$. For instance, if $K = [\![\psi]\!]_{\mathcal{K}}$, then $RK = [\![\Diamond_r\psi]\!]_{\mathcal{K}}$. The theorem follows immediately. ◻

The question arises whether an analogous statement holds when we allow choosing an arbitrary non-rational transduction instead of an arbitrary non-regular language. In other words: Are there rational transductions $R_1, \ldots, R_n$ and regular languages $L_1, \ldots, L_m$ over an alphabet X such that for any non-rational transduction T, the Kripke structure $(X^*, R_1, \ldots, R_n, T, L_1, \ldots, L_m)$ allows the definition of every arithmetical language in multimodal logic? The answer is no, since there are non-rational transductions T that preserve regularity, i.e., for which TL is regular whenever L is regular. Take, for example, the transduction $T = \{(w, ww) \mid w \in X^*\}$. It is clearly not rational, since $T^{-1}X^* = \{ww \mid w \in X^*\}$ is not regular. However, it is not hard to see that TL is effectively regular for regular languages L [31]. In particular, for every choice of $R_1, \ldots, R_n$ and $L_1, \ldots, L_m$ as above, every language definable in $(X^*, R_1, \ldots, R_n, T, L_1, \ldots, L_m)$ is regular and effectively constructible, implying that the model-checking problem is decidable.

6 Open problems

An interesting open question is whether the number of rational transductions in Theorem 3.1 can be reduced to 1 or 2.

Finally, Corollary 3.10 raises the question whether a similar statement holds with just one rational transduction: Is there a rational transduction S and a regular language L such that any non-empty regular language can be written as $S^m L$ for some $m \in \mathbb{N}$? (Note that with one transduction, we cannot include the empty set.)

References

1. P. Barceló, D. Figueira, and L. Libkin. Graph logics with rational relations and the generalized intersection problem. In *Proc. of the 27th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2012)*, pages 115–124. IEEE Computer Society, 2012.
2. W. Bekker and V. Goranko. Symbolic model checking of tense logics on rational Kripke models. In *Selected papers of the International Conference on Infinity and Logic in Computation (ILC 2007)*, Lecture Notes in Computer Science, pages 2–20. Springer-Verlag, 2009.
3. J. Berstel. *Transductions and context-free languages*. Teubner, Stuttgart, 1979.
4. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.
5. R. V. Book. Simple representations of certain classes of languages. *Journal of the ACM*, 25(1):23–31, 1978.
6. A. Carayol and C. Morvan. On rational trees. In *Proc. of the 15th Annual EACSL Conference on Computer Science Logic (CSL 2006)*, volume 4207 of *Lecture Notes in Computer Science*, pages 225–239. Springer-Verlag, 2006.
7. O. Carton and W. Thomas. The monadic theory of morphic infinite words and generalizations. *Information and Computation*, 176(1):51–65, 2002.
8. N. Chomsky and M.-P. Schützenberger. The algebraic theory of context-free languages. In *Computer Programming and Formal Systems*, pages 118–161. North-Holland, Amsterdam, 1963.
9. W. Damm. The IO- and OI-hierarchies. *Theoretical Computer Science*, 20(2):95–207, 1982.
10. W. Damm and A. Goerdt. An automata-theoretical characterization of the OI-hierarchy. *Information and Control*, 71(1–2):1–32, 1986.
11. J. Dassow and G. Păun. *Regulated rewriting in formal language theory*. Springer-Verlag, Berlin, Heidelberg, 1989.
12. C. C. Elgot and M. O. Rabin. Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. *Journal of Symbolic Logic*, 31(2):169–181, 1966.
13. J. Engelfriet and G. Rozenberg. Fixed point languages, equality languages, and representation of recursively enumerable languages. *Journal of the ACM*, 27(3):499–518, 1980.
14. H. Fernau and R. Stiebe. Sequential grammars and automata with valences. *Theoretical Computer Science*, 276:377–405, 2002.
15. C. Frougny and J. Sakarovitch. Synchronized rational relations of finite and infinite words. *Theoretical Computer Science*, 108(1):45–82, 1993.

16. G. Geeraerts, J.-F. Raskin, and L. Van Begin. Well-structured languages. *Acta Informatica*, 44(3–4):249–288, 2007.
17. S. Ginsburg and J. Goldstine. Intersection-closed full AFL and the recursively enumerable languages. *Information and Control*, 22(3):201–231, 1973.
18. S. A. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7(3):311–324, 1978.
19. T. Harju, J. Karhumäki, and D. Krob. Remarks on generalized post correspondence problem. In *Proc. of the 13th International Symposium on Theoretical Aspects of Computer Science (STACS 1996)*, volume 1046 of *Lecture Notes in Computer Science*, pages 39–48. Springer-Verlag, 1996.
20. J. Hartmanis and J. Hopcroft. What makes some language theory problems undecidable. *Journal of Computer and System Sciences*, 4(4):368–376, 1970.
21. D. Haussler and H. P. Zeiger. Very special languages and representations of recursively enumerable languages via computation histories. *Information and Control*, 47(3):201–212, 1980.
22. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison–Wesley, Reading, MA, 1979.
23. M. Jantzen and A. Kurganskyy. Refining the hierarchy of blind multicounter languages and twist-closed trios. *Information and Computation*, 185(2):159–181, 2003.
24. B. Khoussainov and A. Nerode. Automatic presentations of structures. In *LCC: International Workshop on Logic and Computational Complexity*, volume 960 of *Lecture Notes in Computer Science*, pages 367–392. Springer-Verlag, 1995.
25. S. C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, Princeton, NJ, 1956.
26. M. Lohrey and G. Zetzsche. On Boolean closed full trios and rational Kripke frames. In *Proc. of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, volume 25 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 530–541, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
27. M. Minsky. Recursive unsolvability of Post's problem of 'tag' and other topics in theory of Turing machines. *Annals of Mathematics*, 74(3):437–455, 1961.
28. C. Morvan. On rational graphs. In *Proc. of the 3rd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2000)*, volume 2303 of *Lecture Notes in Computer Science*, pages 252–266. Springer-Verlag, 2000.
29. C. Morvan and C. Stirling. Rational graphs trace context-sensitive languages. In *Proc. of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS 2001)*, volume 2136 of *Lecture Notes in Computer Science*, pages 548–559. Springer-Verlag, 2001.
30. M. Nivat. Transductions des langages de Chomsky. *Annales de l'Institut Fourier*, 18(1):339–455, 1968.
31. J.-É. Pin and J. Sakarovitch. Some operations and transductions that preserve rationality. In *Proc. of the 6th GI Conference*, volume 145 of *Lecture Notes in Computer Science*, pages 277–288. Springer-Verlag, 1983.
32. A. Rabinovich. On decidability of monadic logic of order over the naturals extended by monadic predicates. *Information and Computation*, 205:870–889, 2007.
33. A. Rabinovich and W. Thomas. Decidable theories of the ordering of natural numbers with unary predicates. In *Proc. of the 15th Annual EACSL Conference on Computer Science Logic (CSL 2006)*, volume 4207 of *Lecture Notes in Computer Science*, pages 562–574, Berlin Heidelberg, 2006. Springer-Verlag.
34. K. Reinhardt. The "trio-zoo"–classes of formal languages generated from one language by rational transduction. Unpublished manuscript.
35. E. Render. *Rational Monoid and Semigroup Automata.* PhD thesis, University of Manchester, 2010.
36. H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1968.
37. S. Seibert. Quantifier hierarchies over word relations. In *Proc. of the 5th Annual EACSL Conference on Computer Science Logic (CSL 1991)*, volume 626 of *Lecture Notes in Computer Science*, pages 329–352. Springer-Verlag, 1992.
38. A. Semenov. Decidability of monadic theories. In *Proc. of the 11th International Symposium on Mathematical Foundations of Computer Science (MFCS 1984)*, volume 176 of *Lecture Notes in Computer Science*, pages 162–175. Springer-Verlag, 1984.

39. W. Thomas. A short introduction to infinite automata. In *Proc. of the 5th International Conference on Developments in Language Theory (DLT 2001)*, volume 2295 of *Lecture Notes in Computer Science*, pages 130–144. Springer-Verlag, 2001.
40. G. Zetzsche. On the capabilities of grammars, automata, and transducers controlled by monoids. In *Proc. of the 38th International Colloquium on Automata, Languages and Programming (ICALP 2011)*, volume 6756 of *Lecture Notes in Computer Science*, pages 222–233. Springer-Verlag, 2011.