DECISION PROBLEMS FOR NON-REGULAR LANGUAGES

GEORG ZETZSCHE

This part of the lecture is an introduction to techniques both for devising decision procedures and for proving undecidability of problems concerning non-regular languages. Here, we will frequently make use of connections between decidability and expressiveness.

1. Rational transductions

We begin with studying closure properties of language classes. Informally, a *closure property* of a languages class $\mathcal{C}$ states that a certain type of transformation applied to languages in $\mathcal{C}$ always yields languages that belong to $\mathcal{C}$ as well.

Examples for such transformations are:

(i) Homomorphisms $\alpha\colon X^* \to Y^*$. They transform a language $L \subseteq X^*$ into the language $\alpha(L) = \{\alpha(w) \mid w \in L\}$.
(ii) Inverse homomorphisms. If $\alpha\colon X^* \to Y^*$ is a homomorphism, then, inversely applied to $L \subseteq Y^*$, it yields $\alpha^{-1}(L) = \{w \in X^* \mid \alpha(w) \in L\}$.
(iii) Intersection with regular sets. For a regular language $R \subseteq X^*$, this transformation turns a language $L \subseteq X^*$ into $L \cap R$.

A *language* is a subset of X^* for some finite alphabet X. A *language class* is a collection of languages that contains at least one non-empty language. A language class $\mathcal{C}$ is said to be a *full trio* if it is closed homomorphisms, inverse homomorphisms, and intersection with regular languages. If instead of arbitrary homomorphisms, we only require closure under *non-erasing* homomorphisms (i.e. $\alpha(x) \neq \varepsilon$ for all $x \in X$), then we have a *trio*.

Examples of full trios are:

- the regular languages,
- the context-free languages, and
- the recursively enumerable languages

(as we will see later). The context-sensitive languages constitute a trio, but not a full trio.

Proposition 1.1. *Every full trio includes the regular languages.*

Proof. Suppose $\mathcal{C}$ is a full trio and let $R \subseteq X^*$ be regular. Since $\mathcal{C}$ is a language class, there is a non-empty $L \subseteq Y^*$ in $\mathcal{C}$. We define $\alpha\colon Y^* \to Y^*$ and $\beta\colon X^* \to Y^*$ to be the homomorphisms with $\alpha(y) = \varepsilon$ for all $y \in Y$ and $\beta(x) = \varepsilon$ for all $x \in X$. Then we have $\alpha(L) = \{\varepsilon\}$ and thus $\beta^{-1}(\alpha(L)) = X^*$. Hence $\beta^{-1}(\alpha(L)) \cap R = R$. Since $\mathcal{C}$ is a full trio, R belongs to $\mathcal{C}$. □

We will now see a characterization of full trios that

- sometimes simplifies proofs that a class is a full trio and
- provides an intuition for the power of the trio operations.

A *finite-state transducer* is a tuple $A = (Q, X, Y, E, q_0, F)$, where

- Q is a finite set of *states*,
- X and Y are the *input alphabet* and the *output alphabet*, respectively,
- E is a finite subset of $Q \times X^* \times Y^* \times Q$, whose elements are called *edges*,
- $q_0 \in Q$ is the *initial state*,
- $F \subseteq Q$ is the set of *final states*.

A *configuration of* A is a triple $(q, u, v) \in Q \times X^* \times Y^*$. We write $(q, u, v) \to_A (q', u', v')$ is there is an edge (q, x, y, q') with $u' = ux$ and $v' = vy$. If there is an edge (q, x, y, q'), we sometimes denote this fact by $q \xrightarrow{(x,y)}_A q'$.

A *transduction* is a subset of $X^* \times Y^*$ for some finite alphabets X, Y. The *transduction defined by* A is

$$T(A) = \{(u, v) \in X^* \times Y^* \mid (q_0, \varepsilon, \varepsilon) \to_A^* (f, u, v) \text{ for some } f \in F\}.$$

A transduction is called *rational* if it is defined by some finite-state transducer.

Note that if we regard (nondeterministic) finite automata as automata with labels in the monoid X^*, then finite-state transducers can be seen as automata with labels in the monoid $X^* \times Y^*$. Formulating this for arbitrary monoids leads to the concept of *rational subsets* of monoids.

We say that a language class $\mathcal{C}$ is *closed under rational transductions* if for each language L in $\mathcal{C}$, say $L \subseteq X^*$, and each rational transduction $R \subseteq X^* \times Y^*$, the language

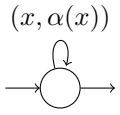
$$LR := \{v \in Y^* \mid (u, v) \in R \text{ for some } u \in L\}$$

also belongs to $\mathcal{C}$.

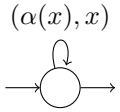
Proposition 1.2. *A language class is a full trio if and only if it is closed under rational transductions.*

In order to prove Proposition 1.2, we first need to show that the (full) trio operations can all be realized by rational transductions.

- Homomorphisms. If $\alpha\colon X^* \to Y^*$ is a homomorphism, then the corresponding transduction $T_\alpha = \{(w, \alpha(w)) \mid w \in X^*\}$ is defined by the transducer

  where the loop $(x, \alpha(x))$ exists for each $x \in X$.
- Inverse homomorphisms. If $\alpha\colon X^* \to Y^*$ is a homomorphism, then the transduction $T_{\alpha^{-1}} = (\alpha(w), w) \mid w \in X^*\}$ applies α in reverse and is realized by

  where, again, the loop $(\alpha(x), x)$ exists for each $x \in X$.
- If $R \subseteq X^*$ is a regular language, then we take a finite automaton accepting R and replace every edge (q, w, q') by an edge (q, w, w, q'). This yields a finite-state transducer A with defining $T_R = \{(w, w) \in X^* \times X^* \mid w \in R\}$, meaning $LR = L \cap R$ for every language $L \subseteq X^*$.

In slight abuse of terminology, we sometimes regard homomorphisms, inverse homomorphisms, and regular intersections as rational transductions. Then, we actually mean the transductions T_α, $T_{\alpha^{-1}}$, and T_R as constructed here.

Hence, every language class that is closed under rational transductions is a full trio. The converse direction is a consequence of the next lemma.

Lemma 1.3 (Nivat)**.** *For each rational transduction $R \subseteq X^* \times Y^*$, there is an alphabet Z, homomorphisms $\alpha\colon Z^* \to X^*$, $\beta\colon Z^* \to Y^*$, and a regular language $K \subseteq Z^*$ such that $R = \{(\alpha(w), \beta(w)) \mid w \in K\}$.*

Proof. Let $A = (Q, X, Y, E, q_0, F)$ be a finite-state transducer defining R. We choose as Z the set of all pairs $(x, y) \in X^* \times Y^*$ that appear as an edge label in A. Then, wen can interpret A as a finite automaton A' over Z and define K as the language accepted by A'. Let $\alpha\colon Z^* \to X^*$ and $\beta\colon Z^* \to Y^*$ be defined by

$$\alpha((x, y)) = x, \qquad \beta((x, y)) = y$$

for every $(x, y) \in Z$. Then $(u, v) \in R$ if and only if there is a run

$$q_0 \xrightarrow{(x_1,y_1)}_A q_1 \xrightarrow{(x_2,y_2)}_A \cdots \xrightarrow{(x_n,y_n)}_A q_n$$

in A with $u = x_1 \cdots x_n$ and $v = y_1 \cdots y_n$. This equivalent to the existence of a run

$$q_0 \xrightarrow{(x_1,y_1)}_{A'} q_1 \xrightarrow{(x_2,y_2)}_{A'} \cdots \xrightarrow{(x_n,y_n)}_{A'} q_n$$

with $u = \alpha((x_1, y_1) \cdots (x_n, y_n))$ and $v = \beta((x_1, y_1) \cdots (x_n, y_n))$. This, in turn, is equivalent to there being a $w \in K$ with $\alpha(w) = u$ and $\beta(w) = v$.

On the other hand, if $R = \{(\alpha(w), \beta(w)) \mid w \in K\}$, then we can turn a finite-state automaton A accepting K into a finite-state transducer A' by replacing every edge (q, z, q') with an edge $(q, \alpha(z), \beta(z), q')$. Then, clearly, $T(A') = R$. □

Corollary 1.4. *For each rational transduction $R \subseteq X^* \times Y^*$, there is an alphabet Z, homomorphisms $\alpha\colon Z^* \to X^*$, $\beta\colon Z^* \to Y^*$, and a regular language $K \subseteq Z^*$ such that for every language $L \subseteq X^*$, we have $LR = \beta(\alpha^{-1}(L) \cap K)$.*

Proof. Take Z, α, β, and K as provided by Lemma 1.3. Then $v \in LR$ if and only if there is a $u \in L$ with $(u, v) \in R$. By the choice of Z, α, β, and K, this is equivalent to there being a $w \in K$ with $\alpha(w) \in L$ and $\beta(w) = v$. This, in turn, is equivalent to $v \in \beta(\alpha^{-1}(L) \cap K)$. □

Observe that Corollary 1.4 completes the proof of Proposition 1.2: It tells us that every rational transductions can be expressed by homomorphisms, inverse homomorphisms, and intersections with regular languages.

The next theorem allows us to compose rational transductions.

Theorem 1.5 (Elgot & Mezei)**.** *If $R \subseteq X^* \times Y^*$ and $S \subseteq Y^* \times Z^*$ are rational transductions, then their* composition

$$R \circ S = \{(u, w) \in X^* \times Z^* \mid \exists v \in Y^*\colon (u, v) \in R,\ (v, w) \in S\}$$

is rational as well.

Proof. We apply a product construction. Let R and S be given by the finite-state transducers $A = (Q_A, X, Y, E_A, q_{0,A}, F_A)$ and $B = (Q_B, Y, Z, E_B, q_{0,B}, F_B)$, respectively. By introducing intermediate states, we can ensure that every edge either reads at most one symbol (and outputs nothing) or it outputs at most one

symbol (and reads nothing). Hence, we may assume that for $(q, x, y, q') \in E_A$, we have $|x| + |y| \leq 1$ and for $(q, y, z, q') \in E_B$, we have $|y| + |z| \leq 1$. We construct the new finite-state transducer $C = (Q_A \times Q_B, X, Y, E_C, (q_{0,A}, q_{0,B}), F_A \times F_B)$ with the following edges.

- For each $(q_A, x, \varepsilon, q'_A) \in E_A$, we add an edge
$$((q_A, q_B), x, \varepsilon, (q'_A, q_B))$$
for each $q_B \in Q_B$.
- For each $y \in Y$ and edges $(q_A, \varepsilon, y, q'_A) \in E_A$ and $(q_B, y, \varepsilon, q'_B) \in E_B$, we add an edge
$$((q_A, q_B), \varepsilon, \varepsilon, (q'_A, q'_B)).$$
- For each $(q_B, \varepsilon, z, q'_B) \in E_B$, we add an edge
$$((q_A, q_B), \varepsilon, z, (q_A, q'_B))$$
for each $q_A \in Q_A$.

Then we clearly have $T(C) = R \circ S$. □

Observe that Corollary 1.4 and Theorem 1.5 immediately imply the following.

Corollary 1.6. *The rational transductions are the smallest class of transductions that is closed under composition and contains homomorphisms, inverse homomorphisms, and regular intersections.*

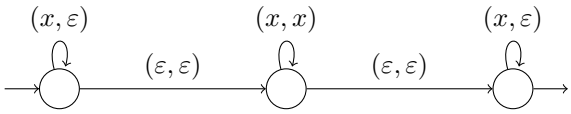
Furthermore, Theorem 1.5 permits a simple proof that the regular languages are closed under rational transductions.

Proposition 1.7. *The regular languages constitute a full trio.*

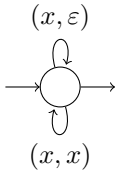
Proof. A language is regular if and only if it is accepted by some finite automaton. This means, a language is regular if and only if it can be written as $\{\varepsilon\}R$ for some rational transduction R. Suppose $L \subseteq X^*$ is regular and $S \subseteq X^* \times Y^*$ is rational. Let $L = \{\varepsilon\}R$ for some rational transduction $R \subseteq X^* \times X^*$. Then we have $LS = (\{\varepsilon\}R)S = \{\varepsilon\}(R \circ S)$ and since $R \circ S \subseteq X^* \times Y^*$ is rational by Theorem 1.5, LS is regular. □

Examples. Let us now see a few examples of relations that are rational.

- Factors. The relation $\{(uvw, v) \mid u, v, w \in X^*\}$ is rational, since it is defined by the transducer



where every edge involving x exists for each $x \in X$.
- Scattered subwords. The relation $\{(u, v) \in X^* \times X^* \mid v \preceq u\}$ is defined by the transducer



where, again, every edge involving x exists for each $x \in X$.

- Concatenation with regular languages. We want to show that for each regular language $K \subseteq X^*$, the relation $R = \{(u, wu) \mid u \in X^*, w \in K\}$ is rational. Note that $LR = KL$.

  In this example, it is more convenient to use trio operations. We define the alphabets $X' = \{x' \mid x \in X\}$ and $Y = X \cup X'$. Furthermore, let $\alpha\colon Y^* \to X^*$ and $\beta\colon Y^* \to X^*$ be the homomorphisms with $\alpha(x') = x$ and $\alpha(x) = \varepsilon$ and $\beta(x) = \beta(x') = x$ for $x \in X$. Then, clearly,

  $$KL = \beta(\alpha^{-1}(L) \cap KX'^*).$$

  Note that if for transductions S, T, we have $LS = LT$ for every language L, then $S = T$. Hence, this shows that R is rational.

As an illustration of how to work with full trios, we prove the following.

Proposition 1.8. *Every full trio that is closed under union and Kleene iteration (which transforms L into L^*) is also closed under concatenation.*

Proof. Suppose $\mathcal{C}$ is a full trio that is closed under union and under Kleene iteration. Let $K \subseteq X^*$ and $L \subseteq Y^*$ be members of $\mathcal{C}$. Choose $c, d \notin X \cup Y$. Since $\mathcal{C}$ is a full trio, we have cK and dL in $\mathcal{C}$. Since $\mathcal{C}$ is union closed, we also find $cK \cup dL$ in $\mathcal{C}$. By closure under Kleene iteration, we also have $M = (cK \cup dL)^*$ in $\mathcal{C}$. Let $\beta\colon (X \cup Y \cup \{c, d\})^* \to (X \cup Y)^*$ be the homomorphism with $\beta(z) = z$ for $z \in X \cup Y$ and $\beta(c) = \beta(d) = \varepsilon$. Then clearly

$$KL = \beta(M \cap cX^*dX^*),$$

which is in $\mathcal{C}$ because cX^*dX^* is regular. □

An inspection of the proof of Proposition 1.8 shows that the Kleene iteration is much stronger than what we require here: For example, a squaring operation would have sufficed. In fact, we will see later that the converse of Proposition 1.8 is not true: There are language classes that are closed under concatenation (and hence under finite powers $L \mapsto L^k$), but not under Kleene iteration.

2. Context-free languages

Let us now apply our knowledge of closure properties to the context-free languages. Of course, these are a very well-known language class and were originally meant to describe grammatical sentences of natural languages. In the context of verification, they are used to describe the behavior of recursive programs, since they are characterized by pushdown automata.

Here, we define context-free languages using pushdown automata. Their definition, in turn, involves the following description of pushdown operations. Let X be an alphabet. We define the new alphabets

$$\bar{X} = \{\bar{x} \mid x \in X\}, \qquad \tilde{X} = X \cup \bar{X}.$$

We will regard symbols $x \in X$ as push operations and symbols $\bar{x} \in \bar{X}$ as pop operations, and then for $w \in X^*$ and $v \in \tilde{X}^*$, define $w \cdot v$ as the result of applying the operations in v to the word w.

The words over $\tilde{X}$ (and, in fact, the monoid $\tilde{X}^*$) act on the set $X^* \cup \{\bot\}$ as follows. For $w \in X^*$ and $x \in X$, we have

$$w \cdot x := wx, \qquad w \cdot \bar{x} := \begin{cases} u & \text{if } w = ux \text{ with } u \in X^*, \\ \bot & \text{otherwise,} \end{cases}$$
$$\bot \cdot x := \bot, \qquad \bot \cdot \bar{x} = \bot.$$

Moreover, we define $s \cdot v$ for $s \in X^* \cup \{\bot\}$ and $v \in \tilde{X}^*$ recursively by setting

$$s \cdot \varepsilon = s, \qquad s \cdot (\tilde{x}v) := (s \cdot \tilde{x}) \cdot v$$

for $s \in X^* \cup \{\bot\}$, $v \in \tilde{X}^*$, and $\tilde{x} \in \tilde{X}$.

A *pushdown automaton* is a tuple $A = (Q, X, Y, E, q_0, F)$, where

- Q is a finite set of *states*,
- X and Y are its *input alphabet* and *stack alphabet*, respectively,
- E is a finite subset of $Q \times X^* \times \tilde{Y}^* \times Q$, the members of which are its *edges*,
- $q_0 \in Q$ is the *initial state*,
- $F \subseteq Q$ is the set of *final states*.

A *configuration of* A is a tuple $(q, u, v) \in Q \times X^* \times Y^*$. We write $(q, u, v) \to_A (q', u', v')$ if there is an edge $(q, x, y, q') \in E$ with $u' = ux$ and $v' = v \cdot y$. Note that in particular, we can only enter a new configuration if $v \cdot y$ is in Y^* (and hence differs from $\bot$). The *language accepted by* A is

$$L(A) = \{w \in X^* \mid (q_0, \varepsilon, \varepsilon) \to_A^* (f, w, \varepsilon) \text{ for some } f \in F\}.$$

We call a language *context-free* if it is accepted by some pushdown automaton. The class of context-free languages is denoted by CF.

In the theory of context-free languages, the semi-Dyck languages play a prominent role. For each $n \in \mathbb{N}$, let $X_n = \{a_1, \ldots, a_n\}$. Then, the language

$$D'_n = \{w \in \tilde{X}_n^* \mid \varepsilon \cdot w = \varepsilon\}$$

is called *semi-Dyck language (over n pairs of parentheses)*. The special role of the semi-Dyck languages is expressed in the Chomsky-Schützenberger Theorem and in order to formulate it, we need some notions.

Let $\mathcal{L}$ be a class of languages. The *full trio generated by* $\mathcal{L}$ is defined as the smallest full trio containing $\mathcal{L}$. If this raises no ambiguities, we sometimes just say that $\mathcal{C}$ is *generated* by $\mathcal{L}$.

Lemma 2.1. *The full trio generated by $\mathcal{L}$ consists of precisely those languages of the form LR, where L is in $\mathcal{L}$ and R is a rational transduction.*

Proof. Let $\mathcal{C}$ be the full trio generated by $\mathcal{L}$ and let $\mathcal{D}$ consist of the languages of the form LR. Since $\mathcal{C}$ includes $\mathcal{L}$, it has to include $\mathcal{D}$. On the other hand, $\mathcal{D}$ is a full trio: If LR is in $\mathcal{D}$ and S is a rational transduction, then $(LR)S = L(R \circ S)$ is again in $\mathcal{D}$ since $R \circ S$ is rational. Therefore, $\mathcal{C} = \mathcal{D}$. □

Proposition 2.2. *The context-free languages constitute a full trio that is generated by the semi-Dyck languages.*

Proof. Let $\mathcal{D}$ be the full trio generated by the semi-Dyck languages. We show that CF coincides with $\mathcal{D}$.

Suppose $L \subseteq X^*$ is context-free and accepted by the pushdown automaton A. Without loss of generality, we may assume that the stack alphabet in A is X_n

for some $n \in \mathbb{N}$. Switching the components in the labels of A results in a finite-state transducer A' with $R = T(A') \subseteq \tilde{X}_n^* \times X^*$. By definition of $L(A)$, we have $L = D'_n R$. Hence, L belongs to $\mathcal{D}$.

For the other direction, let $L \subseteq X^*$ be in $\mathcal{D}$. Then, according to Lemma 2.1, we have $L = D'_n R$ for some rational transduction $R \subseteq \tilde{X}_n^* \times X^*$. Again, by switching the components in the labels of a finite-state transducer for R, we obtain a pushdown automaton with stack alphabet X_n that accepts L. □

From Proposition 2.2, we can now deduce the first version of the Chomsky-Schützenberger Theorem.

Theorem 2.3 (Chomsky-Schützenberger I)**.** *A language is context-free if and only if it can be written as $\beta(\alpha^{-1}(D'_n) \cap K)$ with $n \in \mathbb{N}$, homomorphisms α, β, and a regular language K.*

Proof. The "if" direction follows directly from Proposition 2.2 and the "only if" direction is provided by Proposition 2.2, Lemma 2.1, and Corollary 1.4. □

In order to get some intuition for the languages D'_n, let us characterize them in terms of insertions of factors.

Proposition 2.4. *D'_n is the smallest subset of $\tilde{X}_n^*$ that contains ε and such that whenever $uv \in D'_n$ with $u, v \in \tilde{X}_n^*$, we also have $ua_i\bar{a}_i v \in D'_n$ for $1 \leq i \leq n$.*

Proof. Let L_n be the smallest subset of $\tilde{X}_n^*$ such that $\varepsilon \in L_n$ and such that $uv \in L_n$ implies $ua_i\bar{a}_i v \in L_n$.

Let $w \in D'_n$. We show by induction on $|w|$ that $w \in L_n$. If $|w| = 0$, then $w = \varepsilon \in L_n$, so suppose $|w| > 0$. We claim that w starts in a symbol from X_n and ends in a symbol from $\bar{X}_n$.

- Suppose w started in a symbol from $\bar{X}_n$, say $w = \bar{x}v$ for some $v \in \tilde{X}_n^*$. Then $\varepsilon \cdot w = (\varepsilon \cdot \bar{x}) \cdot v = \bot \cdot v = \bot \neq \varepsilon$, contradicting $w \in D'_n$.
- Suppose w ended in a symbol from X_n, say $w = vx$ for $v \in \tilde{X}_n^*$, $x \in X_n$. Then, $\varepsilon \cdot w = (\varepsilon \cdot v) \cdot x$ is either $\bot$ or a word ending in x, again a contradiction to $w \in D'_n$.

This means w must have a factor in $X_n\bar{X}_n$, say $w = ua_i\bar{a}_j v$ for $u, v \in \tilde{X}_n^*$ and $a_i, a_j \in X_n$. Now suppose $i \neq j$. First, observe that $p := \varepsilon \cdot u$ is in X_n^*, because $\varepsilon \cdot u = \bot$ would imply $\varepsilon \cdot w = \bot$, in contradiction to $w \in D'_n$. Then, however,

$$\varepsilon \cdot w = (\varepsilon \cdot u) \cdot a_i\bar{a}_j v = ((pa_i) \cdot \bar{a}_j) \cdot v = \bot \cdot v = \bot.$$

Therefore, we have $i = j$ and hence $w = ua_i\bar{a}_i v$. Now we claim that $uv \in D'_n$. Since $s \cdot a_i\bar{a}_i = s$ for any $s \in X_n^* \cup \{\bot\}$, We have

$$\begin{aligned} \varepsilon \cdot uv &= (\varepsilon \cdot u) \cdot v = ((\varepsilon \cdot u) \cdot a_i\bar{a}_i) \cdot v \\ &= \varepsilon \cdot (ua_i\bar{a}_i v) = \varepsilon \cdot w = \varepsilon, \end{aligned}$$

and thus $uv \in D'_n$. By induction, this means $uv \in L_n$ and hence $w = ua_i\bar{a}_i v \in L_n$.

Now let $w \in L_n$. We prove $w \in D'_n$ by induction on $|w|$. This is clear if $|w| = 0$, so assume that $|w| > 0$. Then $w = ua_i\bar{a}_i v$ with $uv \in L_n$, $u, v \in \tilde{X}_n^*$, and $a_i \in X_n$. Since $|uv| < |w|$, the induction hypothesis yields $uv \in D'_n$, meaning $\varepsilon \cdot uv = \varepsilon$. Since, again, $s \cdot a_i\bar{a}_i = s$ for every $s \in X_n^* \cup \{\bot\}$, we have

$$\varepsilon \cdot w = \varepsilon \cdot ua_i\bar{a}_i v = ((\varepsilon \cdot u) \cdot a_i\bar{a}_i) \cdot v = (\varepsilon \cdot u) \cdot v = \varepsilon \cdot uv = \varepsilon.$$

and hence $w \in D'_n$. □

If there is a language L in $\mathcal{C}$ such that $\mathcal{C}$ is the full trio generated by the single language L, then $\mathcal{C}$ is said to be a *principal full trio*. Our next goal is to show that the context-free languages are in fact a principal full trio, by proving that in Theorem 2.3, we can always use D_2' instead of D_n'.

The homomorphism $\iota_n \colon \tilde{X}_n^* \to \tilde{X}_2^*$ is defined by

$$\iota_n(a_i) = a_1 a_2^i, \qquad \iota(\bar{a}_i) = \bar{a}_2^i \bar{a}_1.$$

Moreover, the map $\hat{\iota}_n \colon X_n^* \cup \{\bot\} \to X_2^* \cup \{\bot\}$ is given by

$$\hat{\iota}_n(s) = \begin{cases} \iota_n(s) & \text{if } s \in X_n^*, \\ \bot & \text{otherwise.} \end{cases}$$

These two maps allow us to translate between stack operations on X_n^* and stack operations on X_2^*, which is expressed in the following lemma.

Lemma 2.5. *For every $u \in X_n^*$ and $v \in \tilde{X}_n^*$, we have $\iota_n(u) \cdot \iota_n(v) = \hat{\iota}_n(u \cdot v)$.*

Proof. We proceed by induction on $|v|$. If $|v| = 0$, then both sides equal $\iota_n(u)$, so let $|v| > 0$ and write $v = xv'$.

- Let $x \in X_n$, say $x = a_i$. Then by the induction hypothesis, we have $\hat{\iota}_n(ux \cdot v') = \iota_n(ux) \cdot \iota_n(v')$. Moreover, we have $\iota_n(x) \in X_n^*$, so that $ux \cdot v' = u \cdot xv'$ and $\iota_n(u) \cdot \iota_n(x)\iota_n(v') = \iota_n(u)\iota_n(x) \cdot \iota_n(v')$. Therefore,

$$\begin{aligned} \hat{\iota}_n(u \cdot v) = \hat{\iota}_n(u \cdot xv') = \hat{\iota}_n(ux \cdot v') = \iota_n(ux) \cdot \iota_n(v') = \iota_n(u)\iota_n(x) \cdot \iota_n(v') \\ = \iota_n(u) \cdot \iota_n(x)\iota_n(v') = \iota_n(u) \cdot \iota_n(xv') = \iota_n(u) \cdot \iota_n(v). \end{aligned}$$

- Let $x \in \bar{X}_n$, say $x = \bar{a}_i$. Now we have to distinguish three cases.
  - Suppose $u = \varepsilon$. Then $u \cdot v = \bot$ and hence $\hat{\iota}_n(u \cdot v) = \bot$. On the other hand, we have

    $$\iota_n(u) \cdot \iota_n(v) = \varepsilon \cdot \iota_n(v) = \varepsilon \cdot \iota_n(xv') = \varepsilon \cdot \bar{a}_2^i \bar{a}_1 \iota_n(v') = \bot$$

    and hence $\iota_n(u) \cdot \iota_n(v) = \hat{\iota}_n(u \cdot v)$.
  - Suppose $u = u'a_j$ with $j \neq i$. Then, $\hat{\iota}_n(u \cdot v) = \hat{\iota}_n(u'a_j \cdot \bar{a}_i v') = \bot$. Furthermore, we have $\iota_n(u')a_1 a_2^j \cdot \bar{a}_2^i \bar{a}_1 \iota_n(v') = \bot$. Therefore,

    $$\iota_n(u) \cdot \iota_n(v) = \iota_n(u'a_j) \cdot \iota_n(\bar{a}_i v') = \iota_n(u')a_1 a_2^j \cdot \bar{a}_2^i \bar{a}_1 \iota_n(v') = \bot.$$

□

Proposition 2.6. *For every $n \in \mathbb{N}$, we have $D_n' = \iota_n^{-1}(D_2')$.*

Proof. "$\subseteq$": Let $w \in D_n'$. Then $\varepsilon \cdot w = \varepsilon$ and hence

$$\varepsilon \cdot \iota_n(w) = \iota_n(\varepsilon) \cdot \iota_n(w) = \hat{\iota}_n(\varepsilon \cdot w) = \hat{\iota}_n(\varepsilon) = \varepsilon$$

and thus $\iota_n(w) \in D_2'$, meaning $w \in \iota_n^{-1}(D_2')$. "$\supseteq$": Let $w \in \iota_n^{-1}(D_2')$. Then $\varepsilon \cdot \iota_n(w) = \varepsilon$ and therefore

$$\hat{\iota}_n(\varepsilon \cdot w) = \iota_n(\varepsilon) \cdot \iota_n(w) = \varepsilon \cdot \iota_n(w) = \varepsilon,$$

which implies $\varepsilon \cdot w = \varepsilon$. This means $w \in D_n'$. □

This allows us to prove that the context-free languages are a principal full trio:

Theorem 2.7 (Chomsky-Schützenberger II)**.** *A language is context-free if and only if it can be written as $\beta(\alpha^{-1}(D_2') \cap K)$ for homomorphisms α, β and a regular language K.*

Proof. In light of Theorem 2.3, it suffices to show the "only if" direction. Let L be context-free. According to Theorem 2.3, there are is an $n \in \mathbb{N}$, homomorphisms α, β, and a regular language K such that $L = \beta(\alpha^{-1}(D_n') \cap K)$. By Proposition 2.6, this means

$$L = \beta(\alpha^{-1}(\iota_n^{-1}(D_n)) \cap K) = \beta((\alpha \circ \iota_n)^{-1}(D_2') \cap K).$$

□

3. The hardest context-free language

Theorem 2.7 tells us that the single language D_2' suffices to obtain every context-free language by way of rational transductions. The context-free languages share this property with many other language classes–essentially all those that can be characterized by an automata model with a finite-state control and a finite set of storage instructions (whose application needs to be valid in some way).

However, in the case of the context-free languages, one can prove something much stronger: There is even a single language L_0, called the "hardest context-free language", from which every context-free language (that does not contain ε) can be obtained via some inverse homomorphisms. To prove this, we first need to show that a slight change in the operation of a pushdown automaton allows us to eliminate ε-transitions. This, in turn, makes use of the Greibach normal form.

A *context-free grammar* is a tuple $G = (N, T, P, S)$, where

- N, T are disjoint alphabets, called *nonterminals* and *terminals*, respectively,
- P is a finite subset of $N \times (N \cup T)^*$, its elements are called *productions*, and
- $S \in N$ is its *start symbol*.

A production (A, w) is usually denoted by $A \to w$. Let us quickly recall how such a grammar operates. For words $u, v \in (N \cup T)^*$, we have $u \Rightarrow_G v$ if $u = xAy$ and $v = xwy$ for some production $A \to w$ and $x, y \in (N \cup T)^*$. The *language generated by* G is then

$$L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\},$$

where, as usual, $\Rightarrow_G^*$ denotes the reflexive transitive closure of $\Rightarrow_G$. We assume that the reader knows that a language is generated by a context-free grammar if and only if it is accepted by a pushdown automaton and we assume basic familiarity with derivation trees.

A context-free grammar is in *Greibach normal form* (GNF) if every production is in $N \times TN^*$. In other words, every right-hand side begins with a terminal symbol. We will show that every context-free grammar (that does not generate ε) has an equivalent in GNF. This requires the elimination of ε-productions. A production $A \to w$ is called *ε-production* if $w = \varepsilon$.

Proposition 3.1. *For each context-free grammar G with $\varepsilon \notin L(G)$, we can compute an equivalent context-free grammar G' without ε-productions.*

Proof. As a first step, we compute the set $E = \{A \in N \mid A \Rightarrow_G^* \varepsilon\}$. To this end, let

$$E_i = \{A \in N \mid A \text{ can derive } \varepsilon \text{ with a derivation tree of height} \leq i\}.$$

Here, we assume that a single leaf has height 0. Observe that $E = \bigcup_{i \geq 0} E_i$ and $\emptyset = E_0 \subseteq E_1 \subseteq \cdots$.

$$E_{i+1} = \{A \in N \mid \exists A \to w \in P \colon w \in E_i^*\}. \tag{1}$$

This means if $E_i = E_{i+1}$, then $E_j = E_i$ for all $j \geq i$. This implies $E = E_n$, where $n = |N|$: Indeed, if $E_n \subsetneq E$, then $E_n \subsetneq E_{n+1}$ and thus $E_i \subsetneq E_{i+1}$ for all $1 \leq i \leq n$ and therefore $|E| \geq n + 1$, which is impossible.

Hence, we can compute $E = E_n$ by successively computing E_{i+1} according to Eq. (1).

We now transform G into the new grammar G' as follows. For every production $A \to x_1 \cdots x_m$ in P with $x_1, \ldots, x_m \in N \cup T$ and every subset $\Delta \subseteq \{1, \ldots, m\}$ with $x_i \in E$ for $i \in \Delta$, we add a new production $A \to x_{i_1} \cdots x_{i_k}$, where $i_1 < \cdots < i_k$ and $\{i_1, \ldots, i_k\} = \{1, \ldots, m\} \setminus \Delta$. Moreover, we remove all ε-productions. The new grammar G' clearly satisfies $L(G') \subseteq L(G)$.

For the converse direction, let $w \in L(G)$ and let t be a derivation tree for w. In t, we delete every subtree with yield ε. Since $w \neq \varepsilon$, this affects only proper subtrees and by construction of G', the new tree t' is a derivation tree of G'. It derives the word w, meaning $w \in L(G')$. □

Let us now describe how to achieve the Greibach normal form.

Proposition 3.2. *For each context-free grammar G with $\varepsilon \notin L(G)$, one can construct an equivalent in Greibach normal form.*

Proof. If clearly suffices to construct a grammar with $P \subseteq N \times T(N \cup T)^*$. Let $G = (N, T, P, S)$ be a context-free grammar without ε-productions. We call a G *ordered* if there is a linear order on $N \cup T$ such that $a < A$ for each $a \in T$ and $A \in N$. Let $A \to xw$ be a production, where $x \in N \cup T$ and $w \in (N \cup T)^*$. Then we call this production

- *increasing* if $A < x$,
- *decreasing* if $A > x$, and
- *looping* if $A = x$.

Moreover, we call a production A-production if A is its left-hand side.

Suppose, for a moment, that in G, every production is decreasing. Then we can *unfold* the first symbol of each right-hand side, i.e. replace the production $A \to Bw$ with $A \to w_1 w, \ldots, A \to w_n w$, where $B \to w_1, \ldots, B \to w_n$ are all the B-productions. Since N is finite, if we unfold again and again, we end up with a grammar in Greibach normal form. This means, it suffices to construct a grammar in which every production is decreasing. We do this as follows.

Suppose $N = \{A_1, \ldots, A_n\}$. We add new nonterminals $N' = \{B_1, \ldots, B_n\}$ and order the symbols so that $A_1 < \cdots < A_n < B_1 < \cdots < B_n$. Now, step by step, we make all productions in our grammar decreasing. For each $i = n, \ldots, 1$, we change the productions so that every A_j-production with $j \geq i$ is decreasing.

We start with $i = n$ and repeat the following for each $n \geq i \geq 1$. We have the following invariant in every step:

- for every $C > A_i$, every C-production is decreasing, and
- for every $C \leq A_i$, the right-hand side of every C-production is in $(N \cup T)^*$.

We call this invariant $(*)$. We proceed in two steps.

(i) If there are increasing A_i-productions, we can unfold them until all A_i-productions are looping or decreasing (this is possible because of $(*)$).

(ii) After the first step, we may assume that all A_i-productions are looping or decreasing. Let $A_i \to A_i w_1, \ldots, A_i \to A_i w_m$ be all looping A_i-productions and let $A_i \to z_1, \ldots, A_i \to z_k$ be all decreasing A_i-productions. Of course, we may assume that $w_j \neq \varepsilon$ for all j, since otherwise, the production $A_i \to A_i w_j$ could be removed. Note that using left derivations, these permit precisely the derivation of sentential forms in

$$(z_1 + \cdots + z_k)(w_1 + \cdots + w_m)^*.$$

Therefore, we replace the looping productions with the new productions

$$\begin{aligned} &A_i \to z_1 B_i, \ldots, A_i \to z_k B_i, \\ &B_i \to w_1 B_i, \ldots, B_i \to w_m B_i, \\ &B_i \to w_1, \ldots, B_i \to w_m. \end{aligned}$$

Observe that these new productions are all decreasing: Because of $(*)$, the words $w_1, \ldots, w_m$ are all in $(N \cup T)^*$, meaning their first symbol is smaller than B_i. Moreover, our invariant $(*)$ now holds for $i - 1$.

In the end, our invariant implies that all productions are decreasing, which we wanted to achieve. □

We now use the Greibach normal form to get one step closer to the hardest context-free language: We show that every context-free language can be obtained from a particular language using finite-state transducers with only one state. A finite-state transducer $A = (Q, X, Y, E, q_0, F)$ is called *ε-free* if for every edge $(q, x, y, q') \in E$, we have $y \neq \varepsilon$. It is said to be *stateless* if $Q = \{q_0\}$. For each $n \in \mathbb{N}$, we define the language $I_n \subseteq \tilde{X}_n^*$ as

$$I_n = \{w \in \tilde{X}_n^* \mid a_1 \cdot w = \varepsilon\}.$$

Note that I_n is very similar to the semi-Dyck language, the only difference being that the action sequences in I_n are applied to a_1 instead of ε.

Proposition 3.3. *For each context-free language L with $\varepsilon \notin L$, there is an ε-free stateless finite-state transducer A such that $L = I_n(T(A))$ for some $n \in \mathbb{N}$.*

Proof. Let $G = (N, T, P, S)$ be a grammar in Greibach normal form for L. Without loss of generality, let $N = X_n$, so $N = \{a_1, \ldots, a_n\}$, and $S = a_1$. For each production $a_i \to xw$ with $x \in T$ and $w \in X_n^*$, we include an edge

$$(q_0, \bar{a}_i w^R, x, q_0),$$

where w^R denotes the word w in reverse. The reader will easily verify by induction on $m \in \mathbb{N}$: A word vw, $v \in T^*$, $w \in N^*$, is derivable by a left derivation in m steps in G if and only if there is a word $u \in \tilde{X}_n^*$ such that $w = (a_1 \cdot u)^R$ and $(q_0, \varepsilon, \varepsilon) \to_A^m (q_0, u, v)$.

Since L consists of those words that are derivable in G using left derivations, this implies $L = I_n(T(A))$. □

As we will see, Proposition 3.3 means that I_n (or, via a variant of ι_n, I_2) is fit to be used as a building block for the hardest language. However, we will use $\#D'_n$ (more precisely: $\#D'_2$) instead, because (i) this highlights the similarity to

the Chomsky-Schützenberger theorem and (ii) Greibach herself used this language. In the following, # is a fresh symbol, i.e. $\# \notin \tilde{X}_n$. This means, $\#D'_n$ is the set of all words $\#w$ with $w \in D'_n$.

Proposition 3.4. *For each context-free language L with $\varepsilon \notin L$, there is an ε-free stateless finite-state transducer A such that $L = \#D'_2(T(A))$.*

Proof. We proceed in two steps. First, we show that there is always an $n \in \mathbb{N}$ such that the proposition holds for $\#D'_n$ instead of $\#D'_2$.

Suppose $L = I_n(T(A))$ for a stateless ε-free finite-state transducer A and $n \in \mathbb{N}$. Then, for each edge (q_0, w, x, q_0) in A with $w \in \tilde{X}_n^*$ and $x \in X$, we add the edge

$$(q_0, \#a_1 w, x, q_0)$$

and thereby obtain the transducer A'. Since for $w \in \tilde{X}_n^*$, we have $w \in I_n$ if and only if $\#a_1 w \in \#D'_n$, this means $L = I_n(T(A)) = \#D'_n(T(A'))$.

Now going from $\#D'_n$ to $\#D'_2$ is again an application of Proposition 2.6: If $\alpha\colon (\tilde{X}_n \cup \{\#\})^* \to (\tilde{X}_2 \cup \{\#\})^*$ is the homomorphism that agrees with ι_n on $\tilde{X}_n$ and fixes #, then clearly $\#D'_n = \alpha^{-1}(\#D'_2)$. This means, we can take A' and replace every edge (q_0, y, x, q_0) with $(q_0, \alpha(y), x, q_0)$ and obtain an ε-free stateless finite-state transducer A'' with $L = \#D'_n(T(A')) = \#' D_2(T(A''))$. □

Note that in Proposition 3.4, one cannot replace $\#D'_2$ with D'_2: Just like in the exercises, one can show that this would imply that every context-free language L satisfies $LL \subseteq L$, which fails already for regular languages. Here, the # breaks the symmetry of D'_2 and thereby enables us to get rid of states.

Observe that a stateless ε-free finite-state transducer is almost a homomorphism: Assume that it outputs at most one letter per edge. Then, for each letter x, there are finitely many edges $(q_0, w_1, x, q_0), \ldots, (q_0, w_n, x, q_0)$. Hence, when producing x, the transducer appends one of the words $w_1, \ldots, w_n$ to the word that, in the end, has to be a word from $\#D_2$. In contrast, an inverse homomorphism assigns to each input letter precisely one word. Therefore, the last step is to expand the control language so that it can receive finitely many potential factors of $\#D'_2$ at once.

Let $Y_0 = \tilde{X}_2 \cup \{\#, +, [,]\}$. The *Greibach language* $L_0 \subseteq Y_0^*$ is the set of all words

$$[w_{1,1} + \cdots + w_{1,n_1}][w_{2,1} + \cdots + w_{2,n_2}] \cdots [w_{m,1} + \cdots + w_{m,n_m}] \tag{2}$$

for which $w_{i,j} \in (\tilde{X}_2 \cup \{\#\})^*$ and there is a choice function $f\colon \{1, \ldots, m\} \to \mathbb{N}$ such that $1 \leq f(i) \leq n_i$ for $1 \leq i \leq m$ and the word

$$w_{1,f(1)} w_{2,f(2)} \cdots w_{m,f(m)}$$

belongs to $\#D'_2$. Observe that in particular, this means $m \geq 1$, since $\#D'_2$ does not contain ε. As the choice of symbols suggests, the word (2) corresponds to a product of polynomials in non-commuting variables. Then, the condition of belonging to L_0 states: If we were to compute this product as a polynomial, then one of its monomials would be a word in $\#D'_2$.

Proposition 3.5. *L_0 is context-free.*

Proof. Exercise. □

We are finally ready to prove the well-known result of Greibach that L_0 is a hardest context-free language.

Theorem 3.6 (Greibach)**.** *For every context-free language $L \subseteq X^*$ with $\varepsilon \notin L$, there is a homomorphism $\alpha\colon X^* \to Y_0^*$ such that $L = \alpha^{-1}(L_0)$.*

Proof. Take the ε-free stateless finite-state transducer A with $L = \#D_2'(T(A))$ provided by Proposition 3.4. By introducing intermediate states, we may assume that every edge in A is of the form (q_0, w, x, q_0) with $x \in X$. Fix $x \in X$ and let

$$(q_0, w_1, x, q_0), \dots, (q_0, w_k, x, q_0)$$

be all edges of A that read x. Then, we put

$$\alpha(x) := [w_1 + \cdots + w_k].$$

Now the definition of L_0 immediately yields that for $w \in X^*$, we have $\alpha(w) \in L_0$ holds if and only if $w \in \#D_2'(T(A))$. This means $L = \alpha^{-1}(L_0)$. □

4. The triple construction

Here, we shall become acquainted with a versatile construction for grammars. Specifically, we see a proof that the context-free grammars define a full trio. Of course, this also follows from the equivalence of these grammars with pushdown automata. However, instead of providing a direct translation, we present a proof employing the triple construction. The advantage is that variants of this construction can be applied to almost every type of grammar.

First, we have another collection of closure properties that characterize full trios. It involves the *shuffle product*, which is defined as follows. For languages $K, L \subseteq X^*$, we put

$$K \sqcup\!\sqcup L = \{v_0 u_1 v_1 \cdots u_n v_n \mid u_1 \cdots u_n \in K,\ v_0 \cdots v_n \in L,\\ u_1, \dots, u_n, v_0, \dots, v_n \in X^*\}.$$

Proposition 4.1. *A language class is a full trio if and only if it is closed under*

- *homomorphisms,*
- *intersection with regular sets, and*
- flooding*, which, for some $a \in X$, maps $L \subseteq X^*$ to $L \sqcup\!\sqcup \{a\}^*$.*

Proof. Exercise. □

The reason we use Proposition 4.1 is that inverse homomorphisms are usually awkward to realize in grammar constructions. Flooding, on the other hand, is very natural:

Proposition 4.2. *For each context-free grammar $G = (N, T, P, S)$, homomorphism $\alpha\colon T^* \to U^*$, and $a \in T$, one can construct context-free grammars G' and G'' with*

$$L(G') = \alpha(L(G)), \qquad L(G'') = L(G) \sqcup\!\sqcup \{a\}^*.$$

Proof. Exercise. □

Since we want to show that the context-free grammars generate a full trio, Proposition 4.2 leaves us with the task of proving closure under regular intersecion. This is where the triple construction comes into play. Our proof will employ the following notation. For a context-free grammar $G = (N, T, P, S)$ and $A \in N$, we write

$$L_G(A) = \{w \in T^* \mid A \Rightarrow_G^* w\}.$$

Similarly, if $A = (Q, X, E, q_0, F)$ is a finite automaton and $p, q \in Q$, we write

$$L_A(p, q) = \{w \in X^* \mid p \xrightarrow{w} q\}.$$

Proposition 4.3. *Given a context-free grammar $G = (N, T, P, S)$ and a regular language $R \subseteq T^*$, one can construct a context-free grammar G' such that we have $L(G') = L(G) \cap R$.*

Proof. Take a finite automaton $A = (Q, T, E, q_0, F)$ that accepts R. We may assume that the edges of A are all in $Q \times T \times Q$ and $F = \{q_f\}$ (why?). Our new grammar has triples as nonterminals

$$N' = Q \times (N \cup T) \times Q,$$

hence the name "triple construction". As the new start symbol S', we choose (q_0, S, q_f). The productions of G' are set up as follows. For every production $B \to w$ in P with $w = x_1 \cdots x_n$, $x_1, \ldots, x_n \in N \cup T$, we include all productions

$$(p, B, q) \to (p_1, x_1, q_1) \cdots (p_n, x_n, q_n)$$

where $q_i = p_{i+1}$ for $1 \leq i < n$, $p_1 = p$, and $q_n = q$. Moreover, for every edge (p, a, q) in A, we add a production

$$(p, a, q) \to a.$$

We claim that for each $(p, B, q) \in N'$, we have

$$L_{G'}((p, B, q)) = L_G(B) \cap L_A(p, q). \tag{3}$$

"$\subseteq$": Suppose $w \in L_{G'}((p, B, q))$. Then $(p, B, q) \Rightarrow_{G'}^n w$ for some $n \geq 1$. By induction on n, one can easily show that then $w \in L_G(B) \cap L_A(p, q)$.

"$\supseteq$": Suppose $w \in L_G(B) \cap L_A(p, q)$. Then, for some $n \geq 1$, we have $B \Rightarrow_G^n w$ and $p \xrightarrow{w} q$. Again, by induction on n, we can convince ourselves that then w belongs to $L_{G'}((p, B, q))$.

We have thus shown Eq. (3). Since $R = L_A(q_0, q_f)$, this implies

$$L(G') = L_{G'}((q_0, S, q_f)) = L_G(S) \cap L_A(q_0, q_f) = L(G) \cap R.$$

□

5. General undecidability results

In this section, we will see some general techniques to prove undecidability.

We consider decision problems whose input is a language from a full trio. Since formally, decision problems always have strings as input, we need to describe our languages finitely. Therefore, we assume that for the full trios we consider, there is some form of representation for the languages. This is then used as the actual input. For example, regular languages can be represented by finite automata and context-free languages are represented by pushdown automata. We also assume that the application of all mentioned closure properties can be carried out effectively. For example, given a representations of a language L and of a finite-state transducer A, one can compute a representation of $L(T(A))$.

When we prove undecidability of a problem A, we usually take a problem B that is known to be undecidable and reduce it to A. Here, the role of B will be played by the *Post Correspondence Problem (PCP)*, which is defined as follows.

Input: Homomorphisms $\alpha, \beta \colon X^* \to Y^*$ for some alphabets X, Y
Question: Is there a word $w \in X^+$ such that $\alpha(w) = \beta(w)$?

Observe that it is important to look for a word in X^+ (as opposed to X^*), because otherwise, the answer would always be yes, rendering the problem trivially decidable.

Theorem 5.1 (Post)**.** *The PCP is undecidable.*

The first decision problem we consider is the universality problem. Formally, the if $\mathcal{C}$ is a language class, then the *universality problem for* $\mathcal{C}$ is defined as follows.

Input: A language $L \subseteq X^*$ in $\mathcal{C}$ for some alphabet X
Question: Does L equal X^*?

We will see that the universality problem is undecidable for a large variety of full trios, namely all those that contain the language

$$S_= = \{a^n b^n \mid n \geq 0\}.$$

In fact, the most difficult part has already been shown in the exercises. For the sake of completeness, we record the statement of the exercise. For each homomorphism $\alpha\colon X^* \to Y^*$ with $X \cap Y = \emptyset$, we have the language

$$P_\alpha = \{w\alpha(w) \mid w \in X^*\}.$$

Proposition 5.2. *Let X and Y be disjoint alphabets and let $\alpha\colon X^* \to Y^*$ be a homomorphism. Then, the language $(X \cup Y)^* \setminus P_\alpha$ is contained in the full trio generated by $S_=$. Moreover, given α, the representation of $(X \cup Y)^* \setminus P_\alpha$ can be computed.*

(The part after "Moreover" was not asked in the exercise, but is obvious from the solution.)

We also need the fact that principal full trios are closed under union. (In fact, among the finitely generated full trios, this characterizes principal full trios, whichis why we extend the exercise a bit.)

Proposition 5.3. *Let $\mathcal{C}$ be a finitely generated full trio, i.e. generated by a finite set of languages. Then, $\mathcal{C}$ is a principal full trio if and only if $\mathcal{C}$ is closed under union.*

Proof. Exercise. □

We can now use Proposition 5.2 to prove the undecidability of the universality problem. The following applies to a fairly large class of full trios, given that the language $S_=$ is very simple.

Theorem 5.4. *Let $\mathcal{C}$ be a full trio that contains $S_=$. Then, universality is undecidable for $\mathcal{C}$.*

Proof. We reduce the PCP to the universality problem for $\mathcal{C}$. Hence, suppose we are given two homomorphisms $\alpha, \beta\colon X^* \to Y^*$. We may clearly assume that $X \cap Y = \emptyset$. Let $Z = X \cup Y$. Then, the PCP asks whether $P_\alpha \cap P_\beta \cap X^+Y^*$ is non-empty. Equivalently, it asks whether the language

$$K = (Z^* \setminus P_\alpha) \cup (Z^* \setminus P_\beta) \cup (Z^* \setminus X^+Y^*) \tag{4}$$

includes all of Z^*. Since $Z^* \setminus X^+Y^*$ is regular, applying Proposition 5.2 to the principal full trio generated by $S_=$, we see that K belongs to $\mathcal{C}$ (and we can compute a representation for it). Hence, we have $Z^* \subseteq K$ if and only if the answer to the PCP is "no". □

The input to the universality problem is a language over some alphabet X. Of course, this raises the question whether the problem becomes decidable for small, fixed alphabets. This motivates the *binary universality problem for* $\mathcal{C}$:

Input: A language $L \subseteq \{a,b\}^*$ in $\mathcal{C}$
Question: Does L equal $\{a,b\}^*$?

Theorem 5.5. *For each full trio $\mathcal{C}$, the universality problem for $\mathcal{C}$ can be reduced to the binary universality problem for $\mathcal{C}$.*

Proof. Let $L \subseteq X^*$ be an instance of the universality problem for $\mathcal{C}$ and suppose $X = \{x_1, \dots, x_n\}$. Consider the homomorphism $\alpha\colon X^* \to \{a,b\}^*$ with $\alpha(a_i) = ba^i$ for $1 \leq i \leq n$. We claim that

$$X^* \subseteq L \qquad \text{if and only if} \qquad \{a,b\}^* \subseteq \alpha(L) \ \cup \ \{a,b\}^* \setminus \alpha(X^*).$$

"$\Longrightarrow$": Suppose $X^* \subseteq L$ and $w \in \{a,b\}^*$. We distinguish two cases. If $w \in \alpha(X^*)$, say $w = \alpha(u)$ with $u \in X^*$, then $u \in L$ and hence $w = \alpha(u) \in \alpha(L)$. If $w \notin \alpha(X^*)$, then of course, $w \in \{a,b\}^* \setminus \alpha(X^*)$.

"$\Longleftarrow$": Suppose $\{a,b\}^* \subseteq \alpha(L) \ \cup \ \{a,b\}^* \setminus \alpha(X^*)$ and $w \in X^*$. Then, by assumption, $\alpha(w) \in \alpha(L) \ \cup \ \{a,b\}^* \setminus \alpha(X^*)$, but we also have $\alpha(w) \in \alpha(X^*)$. This only leaves $\alpha(w) \in \alpha(L)$. Since α is injective, this implies $w \in L$. This proves our claim.

Since $\alpha(L) \ \cup \ \{a,b\}^* \setminus \alpha(X^*)$ belongs to $\mathcal{C}$ (why?) and we can compute a representation of it, this reduces the universality problem to the binary universality problem. □

Next, we want to apply the undecidability of the universality problem: We consider the problem of deciding whether a given language belongs to a particular class. Unfortunately, this will again be undecidable in most cases.

The proof requires a simple lemma. For languages $K, L \subseteq X^*$, we define

$$K^{-1}L = \{w \in X^* \mid \exists v \in K\colon vw \in L\}.$$

We call $K^{-1}L$ the *quotient of L with respect to K*.

Lemma 5.6. *Full trios are closed under quotients with respect to regular languages. In other words, if L belongs to the full trio $\mathcal{C}$ and K is regular, then $K^{-1}L$ belongs to $\mathcal{C}$ as well.*

Proof. Let $X' = \{x' \mid x \in X\}$ consist of fresh symbols. We define the homomorphisms $\alpha, \beta\colon (X \cup X')^* \to X^*$ by $\alpha(x) = \alpha(x') = x$, $\beta(x') = x$, and $\beta(x) = \varepsilon$ for all $x \in X$. Then, we clearly have $K^{-1}L = \beta(\alpha^{-1}(L) \cap KX'^*)$. □

Theorem 5.7 (Greibach)**.** *Let $\mathcal{C}$ and $\mathcal{D}$ be full trios such that*

(i) $\mathcal{D}$ is not included in $\mathcal{C}$
(ii) $\mathcal{D}$ contains $S_= = \{a^n b^n \mid n \geq 0\}$, and
(iii) $\mathcal{D}$ is closed under union.

Then, the $\mathcal{C}$-membership problem for $\mathcal{D}$ *is undecidable:*

Input: *A language L in $\mathcal{D}$.*
Question: *Does L belong to $\mathcal{C}$?*

Proof. Since $\mathcal{D}$ is not included in $\mathcal{C}$, there is a language $D \subseteq X^*$ in $\mathcal{D}$ that does not belong to $\mathcal{C}$.

According to Theorem 5.4, the universality problem is undecidable for $\mathcal{D}$. Therefore, we reduce the undecidability problem for $\mathcal{D}$ to the $\mathcal{C}$-membership problem.

Hence, let $L \subseteq Y^*$ be a given member of $\mathcal{D}$. Moreover, let $\# \notin X \cup Y$ be a fresh symbol. By the assumed closure properties, the language

$$K = L\#X^* \ \cup \ Y^*\#D$$

is contained in $\mathcal{D}$. We now show that K is in $\mathcal{C}$ if and only if $L = Y^*$, clearly completes our reduction. We begin with the "if" statement. Suppose $L = Y^*$. Then

$$K = L\#X^* \cup Y^*\#D = Y^*\#X^*,$$

which is regular and hence contained in $\mathcal{C}$. Now suppose $L \neq Y^*$ and $K \in \mathcal{C}$. Then there is a word $w \in Y^* \setminus L$. By Lemma 5.6, we have $\{w\#\}^{-1}K \in \mathcal{C}$. However, we will show that $\{w\#\}^{-1}K = D$.

- "$\subseteq$": Let $v \in \{w\#\}^{-1}K$. Then $w\#v \in K = L\#X^* \cup Y^*\#D$. Since $w \notin L$, this means $w\#v \in Y^*\#D$ and hence $v \in D$.
- "$\supseteq$": Let $v \in D$. Then $w\#v \in Y^*\#D \subseteq K$ and thus $v \in \{w\#\}^{-1}K$.

This means, however, that D belongs to $\mathcal{C}$, in contradiction to the choice of D. □

Next, we mention an undecidability results that will permit us to prove that the set of palindromes (i.e. the set of words w with $w^R = w$) is not a Petri net language. Recall that $X_n = \{a_1, \ldots, a_n\}$, $\bar{X}_n = \{\bar{a}_1, \ldots, \bar{a}_n\}$, and $\tilde{X}_n = X_n \cup \bar{X}_n$. For each homomorphism $\alpha\colon X^* \to Y^*$, let

$$Q_\alpha = \{w\alpha(w^R) \mid w \in X^*\},$$

where, again, w^R is the word w in reverse. Moreover, let $\gamma_n\colon X_n^* \to \bar{X}_n^*$ be the homomorphism that adds bars: $\gamma_n(a_i) = \bar{a}_i$ for $1 \leq i \leq n$. With this, we define

$$Q_n = Q_{\gamma_n} = \{w\gamma_n(w^R) \mid w \in X_n^*\}.$$

The set of *palindromes over* X is the set of all $w \in X^*$ such that $w^R = w$. First, we want to understand which full trios contain the languages Q_α and the set of palindromes.

Proposition 5.8. *Let $\mathcal{C}$ be a full trio. Then, the following are equivalent:*

- *$\mathcal{C}$ contains Q_2*
- *$\mathcal{C}$ contains the set of palindromes over $\{a, b\}$.*
- *$\mathcal{C}$ contains Q_α for every homomorphism $\alpha\colon X^* \to Y^*$.*

Proof. Exercise. □

Theorem 5.9. *Let $\mathcal{C}$ be a full trio that contains Q_2 (or the set of palindromes over $\{a, b\}$). Then, the* intersection problem for $\mathcal{C}$ *is undecidable:*

Input: *Languages K, L in $\mathcal{C}$.*
Question: *Is the intersection $K \cap L$ empty?*

Proof. We reduce the PCP to the intersection problem, so suppose $\alpha, \beta\colon X^* \to Y^*$ are homomorphisms and define the homomorphisms $\alpha^R, \beta^R\colon X^* \to Y^*$ so that $\alpha^R(x) = \alpha(x)^R$ and $\beta^R(x) = \beta(x)^R$ for $x \in X$. According to Proposition 5.8, $\mathcal{C}$ effectively contains Q_{α^R} and Q_{β^R}. Observe that our PCP instance has a solution if and only if $Q_{\alpha^R} \cap Q_{\beta^R} \cap X^+Y^* \neq \emptyset$ and since $Q_{\alpha^R} \in \mathcal{C}$ and $Q_{\beta^R} \cap X^+Y^* \in \mathcal{C}$, this is an instance of the intersection problem for $\mathcal{C}$. □

6. The recursively enumerable languages

In this section, we present a language that generates the recursively enumerable languages as a full trio. This will allow us to show that the Petri net languages are not closed under Kleene iteration.

Recall that $D'_n = \{w \in \tilde{X}_n^* \mid \varepsilon \cdot w = \varepsilon\}$. Here, we also need

$$\hat{D}'_1 = \{w \in \{a_2, \bar{a}_2\}^* \mid \varepsilon \cdot w = \varepsilon\}.$$

Clearly, $\hat{D}'_1$ is a homomorphic image of D'_1. Let $\#_1, \#_2 \notin \tilde{X}_n$ be fresh symbols and define

$$C = (D'_1 \#_1)^* \sqcup\!\sqcup (\hat{D}'_1 \#_2)^*.$$

We want to show that C generates the recursively enumerable languages as a full trio. Observe that if we interpret

- a_i as 'increment counter i',
- $\bar{a}_i$ as 'decrement counter i', and
- $\#_i$ as 'test counter i for zero' (i.e. proceed only if counter i is zero),

then C is the set of instruction sequences that operate faithfully on two counters. Therefore, we can use C and rational transductions to simulate two-counter machines.

A *two-counter machine* is a tuple $A = (Q, X, E, q_0, F)$, where

- Q is a finite set of *states*,
- X is its *input alphabet*,
- E is a finite subset of $Q \times X^* \times \{0, 1\} \times \{-1, 0, 1\} \times Q$, called the set of *edges*,
- $q_0 \in Q$ is the *initial state*,
- $F \subseteq Q$ is the set of *final states*.

The counters in a two-counter machine can only assume non-negative values. Therefore, a *configuration* is a tuple $(q, w, c_0, c_1) \in Q \times X^* \times \mathbb{N} \times \mathbb{N}$. Intuitively, an edge (q, u, i, d) tells us to operate on counter i and apply the operation indicated by $d \in \{-1, 0, 1\}$, where -1, 0, 1 means decrement, zero test, and increment, respectively. Let us define this formally. For configurations $(q, w, c_0, c_1), (q', w', c'_0, c'_1)$, we write $(q, w, c_0, c_1) \to_A (q', w', c'_0, c'_1)$ if there is an edge (q, u, i, d, q') with $w' = wu$, $c'_{1-i} = c_{1-i}$, and

- If $d \in \{-1, 1\}$, then $c'_i = c_i + d$ and
- if $d = 0$, then $c'_i = c_i = 0$.

The *language accepted by* A is defined as

$$L(A) = \{w \in X^* \mid (q_0, \varepsilon, 0, 0) \to_A^* (f, w, 0, 0) \text{ for some } f \in F\}.$$

Theorem 6.1 (Minsky)**.** *The languages accepted by two-counter machines are precisely the recursively enumerable languages.*

Corollary 6.2. *The recursively enumerable languages are the smallest full trio containing C.*

Proof. Since C is recursively enumerable, every language CR with a rational transduction R is recursively enumerable as well. Hence, the smallest full trio containing C is included in the recursively enumerable languages.

Let L be recursively enumerable and let A be a two-counter machine accepting L. From A, we construct the finite-state transducer A' as follows. A' has the same

states, initial state, and final states as A. Moreover, it has the following edges: For each edge (q, u, d_1, d_2, q') in A, we add an edge (q, u, v, q'), where

$$v = \begin{cases} a_1 & \text{if } i = 0,\ d = 1, \\ \bar{a}_1 & \text{if } i = 0,\ d = -1, \\ \#_1 & \text{if } i = 0,\ d = 0, \\ a_2 & \text{if } i = 1,\ d = 1, \\ \bar{a}_2 & \text{if } i = 1,\ d = -1, \\ \#_2 & \text{if } i = 1,\ d = 0. \end{cases}$$

Then, since C is the set of legal instruction sequences for two counters, we have $CT(A') = L(A) = L$. □

Remark. In this section, we have seen a number of undecidability results for full trios. Since the problems we proved undecidable are often useful in the context of verification, an important line of research seeks restricted models that still permit decision procedures. Notable examples are *deterministic pushdown automata* [Koz97] and *visibly pushdown automata* [AM04]. Compared to general pushdown automata, these models have less freedom to operate for a given input word. This means, it is easier to reason about how an automaton will behave for certain inputs.

For example, for deterministic or visibly pushdown automata, it is decidable whether they accept a regular language [Ste67; BLS06]. Moreover, a famous result of Sénizergues [Sen97] states that the *equivalence problem* (given two languages K, L, does $K = L$?) is decidable for *deterministic* pushdown automata. (Note that Theorem 5.4 implies that this is undecidable for general pushdown automata.) For visibly pushdown automata, an analogous problem is also decidable [AM04].

7. Petri net languages

In this section, we apply the general results on full trios to the class of Petri net languages. Specifically, we show how decidability and undecidability results can be used to make conclusions about expressiveness. The definition of Petri net languages already appeared in the exercises, but we include it here as a reminder. A *labeled Petri net* is a tuple $N = (S, T, W, M_0, X, \lambda, F)$, where

- (S, T, W, M_0) is a marked Petri net,
- X is its *label alphabet*,
- $\lambda\colon T \to X \cup \{\varepsilon\}$ is its *labeling function*,
- $F \subseteq \mathbb{N}^S$ is a finite set of *final markings*.

The language *accepted by* N is defined as

$$L(N) = \{w \in X^* \mid \exists \sigma \in T^*, M \in F\colon M_0[\sigma\rangle M,\ w = \lambda(\sigma)\}.$$

Note that here, we identified λ with the unique extension of λ to a homomorphism $T^* \to X^*$. It was an exercise to show the following.

Proposition 7.1. *The Petri net languages form a full trio.*

We have already mentioned the following result.

Theorem 7.2 (Mayr)**.** *The reachability problem for Petri nets is decidable.*

As a simple consequence of Theorem 7.2, we have:

Proposition 7.3. *The* emptiness problem *is decidable for Petri net languages:*
Input: *A Petri net language L.*
Question: *Is L empty?*
In particular, each Petri net language itself is decidable.

The following is almost a characterizing property of the Petri net languages: In an exercise, it will be shown that the Petri net languages are the smallest full trio that contains D_1' and is closed under intersection.

Proposition 7.4. *The Petri net languages are closed under shuffle. In particular, they are closed under intersection.*

Proof. In order to construct a Petri net for the shuffle language, one can just take the disjoint union of two Petri nets and combine the initial and the final markings. It was an exercise to show that a full trio is closed under shuffle if and only if it is closed under intersection. This proves the second statement. □

We are ready to deduce inexpressibility results from decidability and undecidability. Together, Corollary 6.2 and Proposition 7.3 almost immediately imply the following.

Proposition 7.5. *The following languages are not Petri net languages:*

- C,
- *the palindromes over* $\{a, b\}$,
- $(D_1'\#_1)^*$.

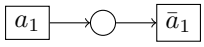
In particular, the Petri net languages are not closed under Kleene iteration.

Proof. If C were a Petri net language, then every recursively enumerable language would be a Petri net language because of Corollary 6.2. However, Proposition 7.3 tells us that every Petri net language is decidable.

Since C can be obtained from $(D_1'\#_1)^*$ by applications of a homomorphism and shuffle, this means $(D_1'\#_1)^*$ is not a Petri net language either.

Furthermore, if the palindromes over $\{a, b\}$ were a Petri net language, this would imply undecidability of the intersection problem for the Petri net languages (Theorem 5.9). However, Propositions 7.3 and 7.4 imply that this problem is decidable.

Finally, the language D_1' is easily seen to be accepted by the labeled Petri net



with empty initial and final marking. This means, $D_1'\#_1$ is also a Petri net language. □

Corollary 7.6. *The Petri net languages and the context-free languages are incomparable.*

Proof. The language $(D_1'\#_1)^*$ and the palindromes over $\{a, b\}$ are two examples of languages that are context-free but do not belong to the Petri net languages. Moreover, the language $\{a^n b^n c^n \mid n \geq 0\}$ is a Petri net language (exercise!), but not context-free. □

Corollary 7.7. *The following two problems are undecidable:*

- *Given a Petri net language L, is L context-free?*
- *Given a context-free language L, is L a Petri net language?*

Proof. Since $S_=$ is both context-free and a Petri net language, and both classes are closed under union, this follows immediately from Corollary 7.6 and Theorem 5.7. □

8. Downward closures

In this section, we learn about the computation of downward closure. In other words, we study the question for which language classes $\mathcal{C}$ there is an algorithm that, given a language L in $\mathcal{C}$, computes a finite automaton for $L{\downarrow}$. We are interested in this because downward closures are abstractions of languages that reflect important properties:

- L is infinite if and only if $L{\downarrow}$ is infinite.
- a can occur somewhere after b in a word in L if and only if $ba \in L{\downarrow}$.

Moreover, maybe more importantly, if we can compute finite automata for downward closures, we can decide whether $K{\downarrow} \subseteq L{\downarrow}$ for given languages K, L in $\mathcal{C}$. Note that straight inclusion ("Does $K \subseteq L$?") is almost always undecidable!

This means, downward closure are an alternative when we want to abstract languages from a class that does not guarantee semilinearity of Parikh images: We know that the downward closure is *always* regular, but Parikh images are not always semilinear (for example, in the case of Petri net languages). However, while downward closures are always regular, it is not clear when we can effectively construct finite automata for them. Let us formalize this problem.

If $\mathcal{C}$ is a language class, we say that *downward closures are computable for* $\mathcal{C}$ if there is an algorithm that, given a language L in $\mathcal{C}$, computes a finite automaton for the language $L{\downarrow}$.

A useful tool for the analysis of downward closed languages are the so-called ideals. An *ideal* is a language of the form

$$X_0^*\{x_1, \varepsilon\}X_1^* \cdots \{x_n, \varepsilon\}X_n^*,$$

where $X_0, \ldots, X_n$ are alphabets and $x_1, \ldots, x_n$ are letters. Clearly, every ideal is downward closed. The converse is not true. For example, the language $\{a, b, \varepsilon\}$ is downward closed, but not an ideal. However, the following theorem tells us that the converse is almost true.

Theorem 8.1 (Jullien '69, Abdulla et. al. '04)**.** *A language L is downward closed if and only if it a finite union of ideals.*

In order to prove Theorem 8.1, we need a lemma.

Lemma 8.2. *If I and J are ideals, then IJ is a finite union of ideals.*

Proof. Supppose $I = X_0^*\{x_1, \varepsilon\}X_1^* \cdots \{x_n, \varepsilon\}X_n^*$ and $J = Y_0\{y_1, \varepsilon\}Y_1^* \cdots \{y_m, \varepsilon\}Y_m^*$. Observe that $Y_0^* = \{\varepsilon\} \cup \bigcup_{y_0 \in Y_0} \{y_0, \varepsilon\}Y_0^*$. Therefore, we have

$$\begin{aligned} IJ =& X_0^*\{x_1, \varepsilon\}X_1^* \cdots \{x_n, \varepsilon\}X_n^*\{y_1, \varepsilon\}Y_1^* \cdots \{y_m, \varepsilon\}Y_m^* \\ &\cup \bigcup_{y_0 \in Y_0} X_0^*\{x_1, \varepsilon\}X_1^* \cdots \{x_n, \varepsilon\}X_n^*\{y_0, \varepsilon\}Y_0^*\{y_1, \varepsilon\}Y_1^* \cdots \{y_m, \varepsilon\}Y_m^*. \end{aligned}$$

□

This allows us to prove Theorem 8.1.

Proof of Theorem 8.1. In the lecture, we have already seen that for every language L, the language $L{\downarrow}$ is regular. This means in particular that every downward closed language L can be written in the form $R{\downarrow}$ with a regular language R. Therefore, it suffices to prove: for every regular language $R \subseteq X^*$, its downward closure $R{\downarrow}$ is a finite union of ideals.

We proceed by induction with respect to a rational (also known as: regular) expression for R.

- If $R = \emptyset$, then R is trivially a finite union of ideals.
- If $R = \{x\}$, then $R{\downarrow} = \{x, \varepsilon\}$ is an ideal.
- If $R = ST$, then by induction we have $S = \bigcup_{1 \leq i \leq n} I_n$ and $T = \bigcup_{1 \leq i \leq m} J_i$ with ideals I_i and J_i. Observe that
$$R = ST = \bigcup_{1 \leq i \leq n, 1 \leq j \leq m} I_i J_j$$
and since each $I_i J_j$ is a finite union of ideals (Lemma 8.2), this proves that R is such a finite union.
- If $R = S \cup T$, then this follows directly from the fact that S and T are each a finite union of ideals.
- If $R = S^*$, then we define $Y = \{x \in X \mid \exists w \in S \colon |w|_x \geq 1\}$ the set of all letters that occur in some member of S. We claim that $R{\downarrow} = Y^*$. Of course, we have $R \subseteq Y^*$ and hence $R{\downarrow} \subseteq Y^*$. On the other hand, if $w \in Y^*$, then by definition of Y, each letter in w occurs in a word from S: We can write $w = x_1 \cdots x_n$, $x_1, \ldots, x_n \in Y$ and find $w_1, \ldots, w_n \in S$ such that $|w_i|_{x_i} \geq 1$ for $1 \leq i \leq n$. Then the word $w_1 \cdots w_n$ belongs to S^*, and since w is a subword of $w_1 \cdots w_n$, this implies $w \in S^*{\downarrow} = R{\downarrow}$. This proves $R{\downarrow} = Y^*$. In particular, $R{\downarrow}$ is an ideal.

□

Our characterization of when downward closures are computable involves a decision problem. The *simultaneous unboundedness problem (SUP) for $\mathcal{C}$* is the following decision problem:

Input: A language $L \subseteq a_1^* \cdots a_n^*$ in $\mathcal{C}$.
Question: Does $L{\downarrow}$ equal $a_1^* \cdots a_n^*$?

Theorem 8.3 (Z. '15)**.** *Let $\mathcal{C}$ be a full trio. Then, the following are equivalent:*

(i) Downward closures are computable for $\mathcal{C}$
(ii) The SUP for $\mathcal{C}$ is decidable.

Proof. The implication "(i)⇒(ii)" is easy: If we can compute downward closures for $\mathcal{C}$, then we can compute a finite automaton for $L{\downarrow}$ and compare its language with $a_1^* \cdots a_n^*$. Hence, let us prove "(ii)⇒(i)".

First, we observe that decidability of the SUP implies decidability of the emptiness problem: Given $L \subseteq X^*$, we can use the rational transduction
$$R = \{(w, a^n) \mid w \in X^*, n \geq 0\}$$
and apply the SUP to the language $LR \subseteq a^*$. Clearly, $L \neq \emptyset$ if and only if $(LR){\downarrow} = a^*$. Hence, we can decide the emptiness problem for $\mathcal{C}$.

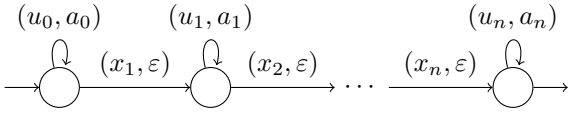
Let us now compute the downward closure for $L \subseteq X^*$. In order to find a finite automaton for $L{\downarrow}$, we use the fact that $L{\downarrow}$ has a decomposition into ideals: We enumerate all finite unions R of ideals. This means, at some point we will arrive

at a representation for $L{\downarrow}$, so that we merely have to check whether $R = L{\downarrow}$. We have to verify two inclusions.

- In order to check whether $L{\downarrow} \subseteq R$, we notice that $L{\downarrow}$ is effectively contained in $\mathcal{C}$ ($L{\downarrow}$ can be obtained from L using a rational transduction). Furthermore, $\mathcal{C}$ is effectively closed under intersection with regular languages, so that $L{\downarrow} \cap (X^* \setminus R)$ belongs to $\mathcal{C}$. Since $L{\downarrow} \subseteq R$ if and only if $L{\downarrow} \cap (X^* \setminus R) = \emptyset$ and since emptiness is decidable for $\mathcal{C}$, we can decide whether $L{\downarrow} \subseteq R$.
- Now, we want to check whether $R \subseteq L{\downarrow}$. Since R is a finite union of ideals, we can check whether each of these ideals is contained in $L{\downarrow}$. Hence, we assume that we have an ideal $I = X_0^*\{x_1, \varepsilon\}X_1^* \cdots \{x_n, \varepsilon\}X_n^*$ and want to decide whether $I \subseteq L{\downarrow}$.

  For each $i \in \{0, \ldots, n\}$, we choose a word u_i such that each letter from X_i occurs precisely one in u_i. Observe that then, we have $I \subseteq L{\downarrow}$ if and only if for every $k \geq 0$, there are numbers $m_0, \ldots, m_n \geq k$ with $u_0^{m_0} x_1 u_1^{m_1} \cdots x_n u_n^{m_n} \in L{\downarrow}$. Hence, it remains to decide whether such numbers $m_0, \ldots, m_n$ exists for every $k \geq 0$.

  Let A be the following finite-state transducer:



  and let T be the transduction generated by A. Then, we have

$$(L{\downarrow})T = \{a_0^{m_0} \cdots a_n^{m_n} \mid u_0^{m_0} x_1 u_1^{m_1} \cdots x_n u_n^{m_n} \in L{\downarrow}\}.$$

  In particular, $(L{\downarrow})T \subseteq a_0^* \cdots a_n^*$ and

$$I \subseteq L{\downarrow} \text{ if and only if } ((L{\downarrow})T){\downarrow} = a_0^* \cdots a_n^*$$

  and the latter is an instance of the SUP. Therefore, we can decide whether $I \subseteq L{\downarrow}$.

□

Coming back to our motivation, we can now use Theorem 8.3 to show that downward closures can be used at least as often as Parikh images. We say that a language class $\mathcal{C}$ *exhibits effectively semilinear Parikh images* if given a language L from $\mathcal{C}$, we can effectively compute a semilinear representation (equivalently, a Presburger formula) for its Parikh image $\Psi(L)$.

Corollary 8.4. *If $\mathcal{C}$ exhibits effectively semilinear Parikh images, then downward closures are computable for $\mathcal{C}$.*

Proof. Exercise. □

This already allows us to compute downward closures for a number of language classes. In the next section, we will see that Parikh images of context-free languages are effectively semilinear.

9. Algebraic extensions

In this section, we learn about a generalization of context-free grammars. What makes this generalization interesting is that its additional capabilities permit very

elegant inductive proofs: When we prove a statement about the generalization instead of context-free grammars, the statement, and hence the induction hypothesis, becomes stronger.

Let $\mathcal{C}$ be a language class. A *$\mathcal{C}$-grammar* is a tuple $G = (N, T, P, S)$, where

- N and T are disjoint alphabets, its elements are called *nonterminals* and *terminals*, respectively,
- P is a finite set of productions $A \to M$, where $A \in N$ and $M \subseteq (N \cup T)^*$ is a language in $\mathcal{C}$,
- $S \in N$ is called the *start symbol*.

Of course, the difference from context-free grammars is that the right-hand sides of productions are *languages* instead of words. Here, the idea is that a production $A \to M$ can be thought of as the (potentially infinite) set of productions $A \to w$ for $w \in M$. Therefore, we write $u \Rightarrow_G v$ if there are words $x, y \in (N \cup T)^*$ with $u = xAy$ and $v = xwy$ for some production $A \to M$ with $w \in M$. Then, again, $\Rightarrow_G^*$ denotes the reflexive, transitive closure of $\Rightarrow_G$ and the *language generated by* G is defined as

$$L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}.$$

We call a language L *algebraic over* $\mathcal{C}$ if $L = L(G)$ for some $\mathcal{C}$-grammar G. The class of all languages that are algebraic over $\mathcal{C}$ is denoted by $\mathsf{Alg}(\mathcal{C})$. The class $\mathsf{Alg}(\mathcal{C})$ is also called the *algebraic extension* of $\mathcal{C}$.

Examples. In order to get an intuition about algebraic extension, we mention two examples.

- Observe that $\mathsf{Alg}(\mathsf{CF}) = \mathsf{CF}$, since a CF-grammar can always be turned into an equivalent context-free grammar.
- We also have $\mathsf{Alg}(\mathsf{Reg}) = \mathsf{CF}$, where Reg denotes the regular languages. The inclusion $\mathsf{Alg}(\mathsf{Reg}) \subseteq \mathsf{CF}$ follows from the last example and $\mathsf{CF} \subseteq \mathsf{Alg}(\mathsf{Reg})$ follows because already singleton right-hand sides would suffice to obtain all context-free languages.

The reason we mention algebraic extension here is that they permit elegant inductive proofs. As a representative of such a proof, we present van Leeuwen's generalization of Parikh's theorem. In fact, this result was the reason van Leeuwen introduced algebraic extensions. The statement of this result requires another concept.

Let $\mathcal{C}$ be a language class. A *substitution* is a map $\sigma \colon X \to \mathbb{P}(Y^*)$. If it satisfies $\sigma(x) \in \mathcal{C}$ for each $x \in X$, we call σ a *$\mathcal{C}$-substitution*. For $w \in X^*$, $w = x_1 \cdots x_n$, $x_1, \ldots, x_n \in X$, we write $\sigma(w)$ for the set of all words $w_1 \cdots w_n$ where $w_i \in \sigma(x_i)$ for $1 \leq i \leq n$. In particular, we have $\sigma(\varepsilon) = \{\varepsilon\}$. Moreover, for $L \subseteq X^*$, we define

$$\sigma(L) = \bigcup_{w \in L} \sigma(w).$$

We say that $\mathcal{C}$ is *substitution closed* if $\sigma(L)$ belongs to $\mathcal{C}$ whenever L belongs to $\mathcal{C}$ and σ is a $\mathcal{C}$-substitution.

Examples. Language classes that are subsitution closed are:

- the context-free languages,
- more generally, $\mathsf{Alg}(\mathcal{C})$ for each language class $\mathcal{C}$,
- the regular languages.

However, as the following shows, not every full trio is closed under substitution:

Proposition 9.1. *Substitution closed full trios are also closed under Kleene iteration and union. In particular, the Petri net languages are not closed under substitution.*

Proof. Let $\mathcal{C}$ be a substitution closed full trio and let $K, L \subseteq X^*$ belong to $\mathcal{C}$. Consider the $\mathcal{C}$-substitution $\sigma\colon \{a,b\} \to \mathbb{P}(X^*)$ with $\sigma(a) = L$ and $\sigma(b) = K$. Since $\mathcal{C}$ is a full trio, it contains the regular languages $\{a\}^*$ and $\{a,b\}$. Therefore, the substitution closure of $\mathcal{C}$ yields that $L^* = \sigma(\{a\}^*)$ and $L \cup K = \sigma(\{a,b\})$ belong to $\mathcal{C}$. This proves that $\mathcal{C}$ is closed under Kleene iteration and union.

Recall that the Petri net languages are not closed under Kleene iteration. □

The announced generalization of Parikh's theorem is the following.

Theorem 9.2 (van Leeuwen)**.** *For every substitution closed full trio $\mathcal{C}$, we have $\Psi(\mathsf{Alg}(\mathcal{C})) = \Psi(\mathcal{C})$.*

The main tool in the proof is what we call the van Leeuwen decomposition.

Let $G = (N, T, P, S)$ be a $\mathcal{C}$-grammar with $|N| \geq 2$ and choose $A \in N$. The *van Leeuwen decomposition (with respect to A)* consists of two grammars:

- The first is the $\mathcal{C}$-grammar $G_A = (\{A\}, T \cup N \setminus \{A\}, P_A, A)$, where P_A is the set of productions $P_A = \{B \to M \in P \mid B = A\}$. In other words, P_A contains all productions from G whose left-hand side is A.
- The second grammar is defined using a substitution in the following way. Let $\sigma\colon N \cup T \to \mathbb{P}((T \cup N \setminus \{A\})^*)$ be the substitution with $\sigma(x) = \{x\}$ for $x \in T \cup N \setminus \{A\}$ and $\sigma(A) = L(G_A)$. Then, we define the $\mathsf{Alg}(\mathcal{C})$-grammar $G' = (N \setminus \{A\}, T, P', S)$, where P' is the set
$$P' = \{B \to \sigma(M) \mid B \to M \in P,\ B \neq A\}.$$
In other words, P' is obtained from P by taking all productions with a left-hand side $\neq A$ and replacing in the right-hand side every occurrence of A with $L(G_A)$.

The name 'decomposition' is justified by the fact that we have $L(G') = L(G)$. Indeed, G' simulates the applications of productions of G merely in a different order: Whenever in G an occurrence of A is produced, G' instead immediately produces all possible sentential forms that can result from A by applying only A-productions until there is no A left.

Furthermore, observe that G_A comprises just one nonterminal and G' has one nonterminal less than G. The right-hand sides of G' are substitutions of $L(G_A)$ and languages form $\mathcal{C}$. Therefore, in order to prove something about all languages in $\mathsf{Alg}(\mathcal{C})$, it often suffices to show it for the one-nonterminal case and for substitutions. This is the case in the proof of Theorem 9.2.

Lemma 9.3. *Let $\mathcal{C}$ be a substitution closed full trio. If G is a $\mathcal{C}$-grammar with one nonterminal, then $\Psi(L(G)) = \Psi(L)$ for some effectively constructible L in $\mathcal{C}$.*

Proof. Let $G = (N, T, P, S)$ with $N = \{S\}$. Since $\mathcal{C}$ is union closed (Proposition 9.1), we may assume that there is just one production $S \to K$ in G. We define
$$K_0 = K \cap T^*,$$
$$K_1 = \{uv \mid uSv \in K,\ u, v \in (N \cup T)^*\}.$$

Note that since $\mathcal{C}$ is a full trio, the languages K_0 and K_1 belong to $\mathcal{C}$. Moreover, we use the substitution $\sigma\colon T \cup N \to \mathbb{P}(T^*)$ by $\sigma(x) = \{x\}$ for $x \in T$ and $\sigma(S) = K_0$. Observe that then, we have $\Psi(L(G)) = \Psi(\sigma(SK_1^*))$. Since $\mathcal{C}$ is a substitution closed full trio and thus Kleene closed (Proposition 9.1), the language $\sigma(SK_1^*)$ belongs to $\mathcal{C}$ and can therefore serve as the desired L. □

We are now ready to prove Theorem 9.2.

Proof of Theorem 9.2. Let $G = (N, T, P, S)$ be a $\mathcal{C}$-grammar. We proceed by induction on the number of nonterminals. The case $|N| = 1$ has been covered in Lemma 9.3. Hence, suppose $|N| \geq 2$. Let $A \in N \setminus \{S\}$ and consider the van Leeuwen decomposition into G' and G_A. According to Lemma 9.3, there is a language L_A in $\mathcal{C}$ with $\Psi(L_A) = \Psi(L(G_A))$. Since the right-hand sides of G' result from substituting $L(G_A)$ for A, we construct another grammar G'' analogous to G', but instead of substituting $L(G_A)$, we substitute L_A for A. Since $\Psi(L_A) = \Psi(L(G_A))$, this guarantees $\Psi(L(G'')) = \Psi(L(G')) = \Psi(L(G))$. Moreover, G'' is a $\mathcal{C}$-grammar because L_A belongs to $\mathcal{C}$.

Hence, G'' is a $\mathcal{C}$-grammar with $\Psi(L(G'')) = \Psi(L(G))$ and that has one nonterminal less than G. Therefore, the induction hypothesis yields a K in $\mathcal{C}$ with $\Psi(K) = \Psi(L(G'')) = \Psi(L(G))$. □

Note that Theorem 9.2 generalizes Parikh's theorem because it tells us, in particular, that for every context-free language L, we can find a regular language R with $\Psi(R) = \Psi(L)$. To obtain Parikh's theorem, one just has to show that every regular language has a semilinear Parikh image. This, however, is not hard to do using another induction on the structure of regular expressions.

References

[AM04] Rajeev Alur and P. Madhusudan. "Visibly Pushdown Languages". In: *Proceedings of STOC 2004*. New York, NY, USA: ACM, 2004, pp. 202–211.

[BLS06] Vince Bárány, Christof Löding, and Olivier Serre. "Regularity Problems for Visibly Pushdown Languages". In: *Proceedings of STACS 2006*. Ed. by Bruno Durand and Wolfgang Thomas. Vol. 3884. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 420–431.

[Koz97] Dexter C. Kozen. *Automata and computability*. New York: Springer-Verlag, 1997.

[Sen97] Géraud Sénizergues. "The equivalence problem for deterministic pushdown automata is decidable". In: *Proceedings of ICALP 1997*. Vol. 1256. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1997, pp. 671–681.

[Ste67] R. E. Stearns. "A regularity test for pushdown machines". In: *Information and Control* 11.3 (1967), pp. 323–340.