Computing Downward Closures for Stacked Counter Automata

Georg Zetzsche

AG Concurrency Theory
Fachbereich Informatik
TU Kaiserslautern
zetzsche@cs.uni-kl.de

Abstract

The downward closure of a language $L$ of words is the set of all (not necessarily contiguous) subwords of members of $L$. It is well known that the downward closure of any language is regular. Although the downward closure seems to be a promising abstraction, there are only few language classes for which an automaton for the downward closure is known to be computable.

It is shown here that for stacked counter automata, the downward closure is computable. Stacked counter automata are finite automata with a storage mechanism obtained by adding blind counters and building stacks. Hence, they generalize pushdown and blind counter automata.

The class of languages accepted by these automata are precisely those in the hierarchy obtained from the context-free languages by alternating two closure operators: imposing semilinear constraints and taking the algebraic extension. The main tool for computing downward closures is the new concept of Parikh annotations. As a second application of Parikh annotations, it is shown that the hierarchy above is strict at every level.

1998 ACM Subject Classification F.4.3 Formal languages

Keywords and phrases abstraction, downward closure, obstruction set, computability

1 Introduction

In the analysis of systems whose behavior is given by formal languages, it is a fruitful idea to consider abstractions: simpler objects that preserve relevant properties of the language and are amenable to algorithmic examination. A well-known such type of abstraction is the Parikh image, which counts the number of occurrences of each letter. For a variety of language classes, the Parikh image of every language is known to be effectively semilinear, which facilitates a range of analysis techniques for formal languages (see [12] for applications).

A promising alternative to Parikh images is the downward closure $L_↓$, which consists of all (not necessarily contiguous) subwords of members of $L$. Whereas for many interesting classes of languages the Parikh image is not semilinear in general, the downward closure is regular for any language [10], suggesting wide applicability. Moreover, the downward closure encodes properties not visible in the Parikh image: Suppose $L$ describes the behavior of a system that is observed through a lossy channel, meaning that on the way to the observer, arbitrary actions can get lost. Then, $L_↓$ is the set of words received by the observer [9]. Hence, given the downward closure as a finite automaton, we can decide whether two systems are equivalent under such observations, and even whether the behavior of one system includes the other. Hence, even if Parikh images are effectively semilinear for a class of languages, computing the downward closure is still an important task. See [2, 3, 15] for further applications.

However, while there always exists a finite automaton for the downward closure, it seems difficult to compute them and there are few language classes for which computability has been...
Computing Downward Closures for Stacked Counter Automata

established. The downward closure is known to be computable for context-free languages and algebraic extensions [5, 13], 0L-systems and context-free FIFO rewriting systems [1], and Petri net languages [9]. It is not computable for reachability sets of lossy channel systems [16] and for Church-Rosser languages [8]. For considerations of complexity, both descriptional and computational, see [3, 7, 11, 17] and the references therein.

It is shown here that downward closures are computable for stacked counter automata. These are automata with a finite state control and a storage mechanism obtained by two constructions (of storage mechanisms): One can build stacks and add blind counters. The former is to construct a new mechanism that stores a stack whose entries are configurations of an old mechanism. One can then manipulate the topmost entry, pop it if empty, or start a new one on top. Adding a blind counter to an old mechanism yields a new mechanism in which the old one and a blind counter (i.e., a counter that can attain negative values and has to be zero in the end of a run) can be used simultaneously.

Stacked counter automata are interesting because among a large class of automata with storage, they are expressively complete for those storage mechanisms that guarantee semilinear Parikh images. This is due to the fact that they accept precisely those languages in the hierarchy obtained from the context-free languages by alternating two closure operators: imposing semilinear constraints (with respect to the Parikh image) and taking the algebraic extension. These two closure operators correspond to the constructions of storage mechanisms in stacked counter automata (see Section 3).

The main tool to show the computability of downward closures is the concept of Parikh annotations. As another application of this concept, it is shown that the aforementioned hierarchy is strict at every level.

The paper is structured as follows. After Section 2 defines basic concepts and notation, Section 3 introduces the hierarchy of language classes. Section 4 presents Parikh annotations, the main ingredient for the computation of downward closures. The main result is then presented in Section 5, where it is shown that downward closures are computable for stacked counter automata. As a second application of Parikh annotations, it is then shown in Section 6 that the hierarchy defined in Section 3 is strict at every level. Because of space restrictions, most proofs can only be found in the long version of this work [18].

2 Preliminaries

A monoid is a set $M$ together with a binary associative operation such that $M$ contains a neutral element. Unless the monoid at hand warrants a different notation, we will denote the neutral element by $1$ and the product of $x, y \in M$ by $xy$. The trivial monoid that contains only the neutral element is denoted by $1$.

If $X$ is an alphabet, $X^*$ denotes the set of words over $X$. The empty word is denoted by $\varepsilon \in X^*$. For a symbol $x \in X$ and a word $w \in X^*$, let $|w|_x$ be the number of occurrences of $x$ in $w$ and $|w| = \sum_{x \in X} |w|_x$. For an alphabet $X$ and languages $L, K \subseteq X^*$, the shuffle product $L \shuffle K$ is the set of all words $uw_1u_1 \cdots v_nu_n$ where $u_0, \ldots, u_n, v_1, \ldots, v_n \in X^*$, $u_0 \cdots u_n \in L$, and $v_1 \cdots v_n \in K$. For a subset $Y \subseteq X$, we define the projection morphism $\pi_Y : X^* \to Y^*$ by $\pi_Y(y) = y$ for $y \in Y$ and $\pi_Y(x) = \varepsilon$ for $x \in X \setminus Y$. By $\mathcal{P}(S)$, we denote the power set of the set $S$. A substitution is a map $\sigma : X \to \mathcal{P}(X^*)$ and given $L \subseteq X^*$, we write $\sigma(L)$ for the set of all words $v_1 \cdots v_n$, where $v_i \in \sigma(x_i)$, $1 \leq i \leq n$, for $x_1 \cdots x_n \in L$ and $x_1, \ldots, x_n \in X$. If $\sigma(x) \subseteq Y$ for each $x \in X$, we call $\sigma$ a letter substitution.

For words $u, v \in X^*$, we write $u \preceq v$ if $u = u_1 \cdots u_n$ and $v = v_0u_1v_1 \cdots u_nv_n$ for some $u_1, \ldots, u_n, v_0, \ldots, v_n \in X^*$. It is well-known that $\preceq$ is a well-quasi-order on $X^*$ and
that therefore the downward closure \( L_\downarrow = \{ u \in X^* \mid \exists v \in L: u \preceq v \} \) is regular for any \( L \subseteq X^* \) [10].

If \( X \) is an alphabet, \( X^\oplus \) denotes the set of maps \( \alpha: X \to \mathbb{N} \). The elements of \( X^\oplus \) are called multiset maps. Let \( \alpha, \beta \in X^\oplus \) be defined by \( (\alpha + \beta)(x) = \alpha(x) + \beta(x) \). With this operation, \( X^\oplus \) is a monoid. We consider each \( x \in X \) to be an element of \( X^\oplus \). For a subset \( S \subseteq X^\oplus \), we write \( S^\oplus \) for the smallest submonoid of \( X^\oplus \) containing \( S \). For \( \alpha \in X^\oplus \) and \( k \in \mathbb{N} \), we define \( (k \cdot \alpha)(x) = k \cdot \alpha(x) \), meaning \( k \cdot \alpha \in X^\oplus \). A subset of the form \( \mu + F^\oplus \) for \( \mu \in X^\oplus \) and a finite \( F \subseteq X^\oplus \) is called linear. A finite union of linear sets is called semilinear. The Parikh map is the map \( \Psi: X^* \to X^\oplus \) defined by \( \Psi(w)(x) = |w|_x \) for all \( w \in X^* \) and \( x \in X \). Given a morphism \( \varphi: X^\oplus \to Y^\oplus \) and a word \( w \in X^* \), we use \( \varphi(w) \) as a shorthand for \( \varphi(\Psi(w)) \). We lift \( \Psi \) to sets in the usual way: \( \Psi(L) = \{ \Psi(w) \mid w \in L \} \). If \( \Psi(L) \) is semilinear, we will also call \( L \) itself semilinear.

Let \( M \) be a monoid. An automaton over \( M \) is a tuple \( A = (Q, M, E, q_0, F) \), in which (i) \( Q \) is a finite set of states, (ii) \( E \) is a finite subset of \( Q \times M \times Q \) called the set of edges, (iii) \( q_0 \in Q \) is the initial state, and (iv) \( F \subseteq Q \) is the set of final states. We write \( (q, m) \to_A (q', m') \) if there is an edge \( (q, r, q') \in E \) such that \( m' = mr \). The set generated by \( A \) is then \( S(A) = \{ m \in M \mid \{ (q_0, 1) \to_A^* (f, m) \text{ for some } f \in F \} \} \).

A class of languages is a collection of languages that contains at least one non-empty language. The class of regular languages is denoted by \( \text{REG} \). A finite state transducer is an automaton over \( Y^* \times X^* \) for alphabets \( X, Y \). Relations of the form \( S(A) \) for finite state transducers \( A \) are called rational transductions. For \( L \subseteq X^* \) and \( T \subseteq Y^* \times X^* \), we write \( TL = \{ u \in Y^* \mid \exists v \in L: (u, v) \in T \} \). If \( TF \) is finite for every finite language \( F \), \( T \) is said to be locally finite. A class \( C \) of languages is called a full trio (full semi-trio) if it is closed under (locally finite) rational transductions, i.e. if \( TL \in C \) for every \( L \in C \) and every (locally finite) rational transduction \( T \). A full semi-AFL \( T \) is a union closed full trio.

**Stacked counter automata**

In order to define stacked counter automata, we use the concept of valence automata, which combine a finite state control with a storage mechanism defined by a monoid \( M \). A valence automaton over \( M \) is an automaton \( A \) over \( X^* \times M \) for an alphabet \( X \). The language accepted by \( A \) is then \( L(A) = \{ w \in X^* \mid (w, 1) \in S(A) \} \).

The class of languages accepted by valence automata over \( M \) is denoted \( \text{VA}(M) \). By choosing suitable monoids \( M \), one can obtain various kinds of automata with storage as valence automata. For example, blind counters, partially blind counters, pushdown storages, and combinations thereof can all be realized by appropriate monoids [19].

If one storage mechanism is realized by a monoid \( M \), then the mechanism that builds stacks is realized by the monoid \( \mathbb{B} \ast M \). Here, \( \mathbb{B} \) denotes the bicyclic monoid, presented by \( \langle a, \tilde{a} \mid a \tilde{a} = 1 \rangle \), and \( * \) denotes the free product of monoids. For readers not familiar with these concepts, it will suffice to know that a configuration of the storage mechanism described by \( \mathbb{B} \ast M \) consists of a sequence \( c_0 a c_1 \cdots a c_n \), where \( c_0, \ldots, c_n \) are configurations of the mechanism realized by \( M \). We interpret this as a stack with the entries \( c_0, \ldots, c_n \).

One can open a new stack entry on top (by multiplying \( a \in \mathbb{B} \)), remove the topmost entry if empty (by multiplying \( \tilde{a} \in \mathbb{B} \)) and operate on the topmost entry using the old mechanism (by multiplying elements from \( M \)). For example, the monoid \( \mathbb{B} \) describes a partially blind counter (i.e. a counter that cannot go below zero and is only tested for zero in the end) and \( \mathbb{B} \ast \mathbb{B} \) describes a pushdown with two stack symbols. Given a storage mechanism realized by a monoid \( M \), we can add a blind counter by using the monoid \( M \times \mathbb{Z} \), where \( \mathbb{Z} \) denotes the group of integers. We define \( \text{SC} \) to be the smallest class of monoids with \( 1 \in \text{SC} \) such that whenever \( M \in \text{SC} \), we also have \( M \times \mathbb{Z} \in \text{SC} \) and \( \mathbb{B} \ast M \in \text{SC} \). A stacked counter automaton
is a valence automaton over $M$ for some $M \in \mathbb{SC}$. For more details, see [19]. In Section 3, we will turn to a different description of the languages accepted by stacked counter automata.

## 3 A hierarchy of language classes

This section introduces a hierarchy of language classes that divides the class of languages accepted by stacked counter automata into levels. This will allow us to apply recursion with respect to these levels. The hierarchy is defined by alternating two operators on language classes, algebraic extensions and semilinear intersections.

### Algebraic extensions

Let $C$ be a class of languages. A $C$-grammar is a quadruple $G = (N, T, P, S)$ where $N$ and $T$ are disjoint alphabets and $S \in N$. The symbols in $N$ and $T$ are called the nonterminals and the terminals, respectively. $P$ is a finite set of pairs $(A, M)$ with $A \in N$ and $M \subseteq (N \cup T)^*$, $M \in C$. A pair $(A, M) \in P$ is called a production of $G$ and also denoted by $A \rightarrow M$. The set $M$ is the right-hand side of the production $A \rightarrow M$.

We write $x \Rightarrow_G y$ if $x = uvw$ and $y = uwv$ for some $u, v, w \in (N \cup T)^*$ and $(A, M) \in P$ with $w \in M$. A word $w$ with $S \Rightarrow_G^n w$ is called a sentential form of $G$ and we write $SF(G)$ for the set of sentential forms of $G$. The language generated by $G$ is $L(G) = SF(G) \cap T^*$. Languages generated by $C$-grammars are called algebraic over $C$. The class of all languages that are algebraic over $C$ is called the algebraic extension of $C$ and denoted $Alg(C)$. We say a language class $C$ is algebraically closed if $Alg(C) = C$. If $C$ is the class of finite languages, $C$-grammars are also called context-free grammars.

We will use the operator $Alg(\cdot)$ to describe the effect of building stacks on the accepted languages of valence automata. In [19], it was shown that $VA(M_0 \ast M_1) \subseteq Alg(VA(M_0) \cup VA(M_1))$. Here, we complement this by showing that if one of the factors is $\mathbb{B} \ast \mathbb{B}$, the inclusion becomes an equality. Observe that since $VA(\mathbb{B} \ast \mathbb{B})$ is the class of languages accepted by pushdown automata and $Alg(\text{REG}) = Alg(VA(\{1\}))$ is clearly the class of languages generated by context-free grammars, the first statement of the following theorem generalizes the equivalence between pushdown automata and context-free grammars.

**Theorem 1.** For every monoid $M$, $Alg(VA(M)) = VA(\mathbb{B} \ast \mathbb{B} \ast M)$.

### Semilinear intersections

The second operator on language classes lets us describe the languages in $VA(M \times \mathbb{Z}^n)$ in terms of those in $VA(M)$. Consider a language class $C$. By $SLI(C)$, we denote the class of languages of the form $h(L \cap \Psi^{-1}(S))$, where $L \subseteq X^*$ is in $C$, the set $S \subseteq X^0$ is semilinear, and $h : X^* \rightarrow Y^*$ is a morphism. We call a language class $C$ Presburger closed if $SLI(C) = C$. Proving the following requires only standard techniques.

**Proposition 2.** Let $M$ be a monoid. Then $SLI(VA(M)) = \bigcup_{n \geq 0} VA(M \times \mathbb{Z}^n)$.

The hierarchy is now obtained by alternating the operators $Alg(\cdot)$ and $SLI(\cdot)$. Let $F_0$ be the class of finite languages and let

$$G_i = Alg(F_i), \quad F_{i+1} = SLI(G_i) \quad \text{for each } i \geq 0, \quad F = \bigcup_{i \geq 0} F_i.$$

Then we clearly have the inclusions $F_0 \subseteq G_0 \subseteq F_1 \subseteq G_1 \subseteq \cdots$. Furthermore, $G_0$ is the class of context-free languages, $F_1$ is the smallest Presburger closed class containing $\text{CF}$, $G_1$ the algebraic extension of $F_1$, etc. In particular, $F$ is the smallest Presburger closed and algebraically closed language class containing the context-free languages.

The following proposition is due to the fact that both $Alg(\cdot)$ and $SLI(\cdot)$ preserve (effective) semilinearity. The former has been shown by van Leeuwen [13].
Proposition 3. The class $F$ is effectively semilinear.

The work [4] characterized all those storage mechanisms among a large class (namely among those defined by graph products of the bicyclic monoid and the integers) that guarantee semilinar Parikh images. Each of the corresponding language classes was obtained by alternating the operators $\text{Alg}()$ and $\text{SLI}()$, meaning that all these classes are contained in $F$. Hence, the following means that stacked counter automata are expressively complete for these storage mechanisms. It follows directly from Theorem 1 and Proposition 2.

Theorem 4. Stacked counter automata accept precisely the languages in $F$.

One might wonder why $F_0$ is not chosen to be the regular languages. While this would be a natural choice, our recursive algorithm for computing downward closures relies on the following fact. Note that the regular languages are not Presburger closed.

Proposition 5. For each $i \geq 0$, the class $F_i$ is an effective Presburger closed full semi- trio. Moreover, for each $i \geq 0$, $G_i$ is an effective full semi-AFL.

4 Parikh annotations

This section introduces Parikh annotations, the key tool in our procedure for computing downward closures. Suppose $L$ is a semilinear language. Then for each $w \in L$, $\Psi(w)$ can be decomposed into a constant vector and a linear combination of period vectors from the semilinear representation of $\Psi(L)$. We call such a decomposition a Parikh decomposition.

The main purpose of Parikh annotations is to provide transformations of languages that make reference to Parikh decompositions without leaving the respective language class. For example, suppose we want to transform a context-free language without leaving the respective language class. For instance, suppose we want to transform a context-free language $L$ into the language $L'$ of all those words $w \in L$ whose Parikh decomposition does not contain a specified period vector.

This may not be possible with rational transductions: If $L = \{a^m b^n \mid m = n \text{ or } m = 2n\}$, then the Parikh image is $(a + b)^{(\infty)} \cup (a + 2b)^{(\infty)}$, but a finite state transducer cannot determine whether the input word has a Parikh image in $(a + b)^{(\infty)}$ or in $(a + 2b)^{(\infty)}$. Therefore, a Parikh annotation for $L$ is a language $K$ in the same class with additional symbols that allow a finite state transducer (that is applied to $K$) to access the Parikh decomposition.

Definition 6. Let $L \subseteq X^*$ be a language and $C$ be a language class. A Parikh annotation (PA) for $L$ in $C$ is a tuple $(K, C, P, (P_c)_{c \in C}, \varphi)$, where

(i) $C, P$ are alphabets such that $X, C, P$ are pairwise disjoint,
(ii) $K \subseteq C(X \cup P)^*$ is in $C$,
(iii) $\varphi$ is a morphism $\varphi : (C \cup P)^{(\infty)} \rightarrow X^{(\infty)}$, and
(iv) $P_c$ is a subset $P_c \subseteq P$ for each $c \in C$, such that

(i) $\pi_X(K) = L$ (the projection property),
(ii) $\varphi(\pi_C(w)) = \Psi(\pi_X(w))$ for each $w \in K$ (the counting property), and
(iii) $\Psi(\pi_C(w)) = \bigcup_{c \in C} c + P_c^{(\infty)}$ (the commutative projection property).

A Parikh annotation describes for each $w$ in $L$ one or more Parikh decompositions of $\Psi(w)$. The symbols in $C$ represent constant vectors and symbols in $P$ represent period vectors. The symbols in $P_c \subseteq P$ correspond to those that can be added to the constant vector corresponding to $c \in C$. Furthermore, for each $x \in C \cup P$, $\varphi(x)$ is the vector represented by $x$. The projection property states that removing the symbols in $C \cup P$ from words in $K$ yields $L$. The commutative projection property requires that after $c \in C$ only symbols representing periods in $P_c$ are allowed and that all their combinations occur. Finally, the counting property says that the additional symbols in $C \cup P$ indeed describe a Parikh decomposition of $\Psi(\pi_X(w))$. Of course, only semilinear languages can have a Parikh annotations.
Example 7. Let \( X = \{a, b, c, d\} \) and consider the regular set \( L = (ab)^*(ca^* \cup db^*) \). For \( K = e(ab)c(qa)^* \cup f(rab)^*d(sb)^* \), \( P = \{p, q, r, s\} \), \( C = \{e, f\} \), \( P_e = \{p, q\} \), \( P_f = \{r, s\} \), and \( \varphi: (C \cup P)^\oplus \rightarrow X^\oplus \) with \( e \rightarrow c \), \( f \rightarrow d \), \( p \rightarrow a + b \), \( q \rightarrow a \), \( r \rightarrow a + b \), and \( s \rightarrow b \), the tuple \((K, C, P, (P_e, P_f))_{g \in C, \varphi}\) is a Parikh annotation for \( L \) in REG.

In a Parikh annotation, for each \( cw \in K \) and \( \mu \in P_e^\oplus \), we can find a word \( cw' \) in \( K \) such that \( \Psi(\pi_{C \cup P}(cw')) = \Psi(\pi_C(cw)) + \mu \). In particular, this implies the equality \( \Psi(\pi_X(cw')) = \Psi(\pi_X(cw)) + \varphi(\mu) \). In our applications, we will need a further guarantee that provides such words, but with additional information on their structure. Such a guarantee is granted by Parikh annotations with insertion marker. Suppose \( \odot \notin X \) and \( u \in (X \cup \odot)^* \) with \( u = u_0 \odot u_1 \cdots \odot u_n \) for \( u_0, \ldots, u_n \in X^* \). Then we write \( u \preceq_{\odot} v \) if \( v = u_0 v_1 u_1 \cdots v_n u_n \) for some \( v_1, \ldots, v_n \in X^* \).

Definition 8. Let \( L \subseteq X^* \) be a language and \( C \) be a language class. A Parikh annotation with insertion marker (PAIM) for \( L \) in \( C \) is a tuple \((K, C, P, (P_e, P_f))_{g \in C, \varphi, \odot}\) such that:

(i) \( \odot \notin X \) and \( K \subseteq C(X \cup P \cup \{\odot\})^* \) is in \( C \),
(ii) \( \pi_{C \cup P \cup \{\odot\}}(K, C, P, (P_e, P_f))_{\varphi, \odot} \) is a Parikh annotation for \( L \) in \( C \),
(iii) there is a \( k \in \mathbb{N} \) such that every \( w \in K \) satisfies \( |w|_\odot \leq k \) (boundedness), and
(iv) for each \( cw \in K \) and \( \mu \in P_e^\oplus \), there is a \( w' \in L \) with \( \pi_{X \cup \odot}(cw) \preceq_{\odot} w' \) and with
\[
\Psi(w') = \Psi(\pi_X(cw)) + \varphi(\mu).
\]
This property is called the insertion property.

If \(|C| = 1\), then the PAIM is called linear and we also write \((K, c, P_e, \varphi, \odot)\) for the PAIM, where \( C = \{c\} \).

In other words, in a PAIM, each \( v \in L \) has an annotation \( cw \in K \) in which a bounded number of positions is marked such that for each \( \mu \in P_e^\oplus \), we can find a \( v' \in L \) with \( \Psi(v') = \Psi(v) + \varphi(\mu) \) such that \( v' \) is obtained from \( v \) by inserting words in corresponding positions in \( v \). In particular, this guarantees \( v \preceq v' \).

Example 9. Let \( L \) and \((K, C, P, (P_e, P_f))_{\varphi, \odot}\) be as in Example 7. Furthermore, let \( K' = e \odot (ab)c \odot (qa)^* \cup f \odot (rab)^*d \odot (sb)^* \). Then \((K', C, P, (P_e, P_f))_{\varphi, \odot}\) is a PAIM for \( L \) in REG. Indeed, every word in \( K' \) has at most two occurrences of \( \odot \). Moreover, if \( cw = e \odot (ab)^m c \odot (qa)^n \in K' \) and \( \mu \in P_e^\oplus \), \( \mu = k \cdot p + \ell \cdot q \), then \( w' = (ab)^k \cdot c \odot a^\ell \odot (ab)^k \odot c \odot a^\ell \odot (ab)^k \odot c \odot a^\ell = w' \) and clearly \( \Psi(\pi_X(w')) = \Psi(\pi_X(cw)) + \varphi(\mu) \) (and similarly for words \( fa \in K' \)).

The main result of this section is that there is an algorithm that, given a language \( L \in F_i \) or \( L \in G_i \), constructs a PAIM for \( L \) in \( F_i \) or \( G_i \), respectively.

Theorem 10. Given \( i \in \mathbb{N} \) and \( L \in F_i \) \( (G_i) \), one can construct a PAIM for \( L \) in \( F_i \) \( (G_i) \).

Outline of the proof The rest of this section is devoted to the proof of Theorem 10. The construction of PAIM proceeds recursively with respect to the level of our hierarchy. This means, we show that if PAIM can be constructed for \( F_i \), then we can compute them for \( G_i \) (Lemma 17) and if they can be constructed for \( G_i \), then they can be computed for \( F_{i+1} \) (Lemma 18). While the latter can be done with a direct construction, the former requires a series of involved steps:

= The general idea is to use recursion with respect to the number of nonterminals: Given a \( F_i \)-grammar for \( L \in G_i \), we present \( L \) in terms of languages whose grammars use fewer nonterminals. This presentation is done via substitutions and by using grammars with one nonterminal. The idea of presenting a language in \( \text{Alg}(C) \) using one-nonterminal grammars and substitutions follows van Leeuwen’s proof of Parikh’s theorem [13].
We construct PAIM for languages generated by one-nonterminal grammars where we are given PAIM for the right-hand-sides (Lemma 16).

We construct PAIM for languages $\sigma(L)$, where $\sigma$ is a substitution, a PAIM is given for $L$ and for each $\sigma(x)$ (Lemma 15). This construction is again divided into the case where $\sigma$ is a letter substitution (i.e., one in which each symbol is mapped to a set of letters) and the general case. Since the case of letter substitutions constitutes the conceptually most involved step, part of its proof is contained in this extended abstract (Proposition 13).

Maybe surprisingly, the most conceptually involved step in the construction of PAIM lies within obtaining a Parikh annotation for $\sigma(L)$ in Alg(\mathcal{C})$, where $\sigma$ is a letter substitution and a PAIM for $L \subseteq X^*$ in Alg(\mathcal{C}) is given. This is due to the fact that one has to substitute the symbols in $X$ consistently with the symbols in $C \cup P$; more precisely, one has to maintain the agreement between $\varphi(\pi_{C \cup P}(\cdot))$ and $\Psi(\pi_X(\cdot))$.

In order to exploit the fact that this agreement exists in the first place, we use the following simple yet very useful lemma. It states that for a morphism $G$ can guarantee $L(G) \subseteq \psi^{-1}(h)$ is by encoding into each nonterminal $A$ the value $\psi(u)$ for the words $u$ that $A$ derives. The $G$-compatible extension of $\psi$ reconstructs this value for each nonterminal. Let $G = (N, T, P, S)$ be a $C$-grammar and $M$ be a monoid. A morphism $\psi: (N \cup T)^* \rightarrow M$ is called $G$-compatible if $u \Rightarrow^*_G v$ implies $\psi(u) = \psi(v)$ for $u, v \in (N \cup T)^*$. Moreover, we call $G$ reduced if for each $A \in N$, we have $A \Rightarrow^*_G w$ for some $w \in T^*$ and $S \Rightarrow^*_G uAv$ for some $u, v \in (N \cup T)^*$.

\textbf{Lemma 11.} Let $H$ be a group, $\psi: T^* \rightarrow H$ be a morphism, and $G = (N, T, P, S)$ be a reduced $C$-grammar with $L(G) \subseteq \psi^{-1}(h)$ for some $h \in H$. Then $\psi$ has a unique $G$-compatible extension $\hat{\psi}: (N \cup T)^* \rightarrow H$. If $H = \mathbb{Z}^n$ and $C = F_i$, $\hat{\psi}$ can be computed.

We will essentially apply Lemma 11 by regarding $X^\oplus$ as a subset of $\mathbb{Z}^n$ and defining $\varphi: (C \cup P \cup X)^* \rightarrow \mathbb{Z}^n$ as the morphism with $\varphi(w) = \Psi(\pi_X(w)) - \varphi(\pi_{C \cup P}(w))$. In the case that $G$ generates the corresponding Parikh annotation, the counting property implies that $L(G) \subseteq \psi^{-1}(0)$. The lemma then states that each nonterminal in $G$ encodes the imbalance between $\Psi(\pi_X(\cdot))$ and $\varphi(\pi_{C \cup P}(\cdot))$ on the words it generates.

We continue with the problem of replacing $C \cup P$ and $X$ consistently. For constructing the PAIM for $\sigma(L)$, it is easy to see that it suffices to consider the case where $\sigma(a) = \{a, b\}$ for some $a \in X$ and $\sigma(x) = \{x\}$ for $x \in X \setminus \{a\}$. In order to simplify the setting and exploit the symmetry of the roles played by $C \cup P$ and $X$, we consider a slightly more general situation. There is an alphabet $X = X_0 \uplus X_1$, morphisms $\gamma_i: X_i^* \rightarrow N$, $i = 0, 1$, and a language $L \subseteq X^*$, $L \in \text{Alg}(F_i)$ with $\gamma_0(\pi_{X_0}(w)) = \gamma_1(\pi_{X_1}(w))$ for every $w \in L$. Roughly speaking, $X_1$ will later play the role of $C \cup P$ and $X_0$ will play the role of $X$. Then, $\gamma_0(w)$ will be the number of $a$'s in $w$ and $\gamma_1(w)$ will be the number of $a$'s represented by symbols from $C \cup P$ in $w$. Therefore, we wish to construct a language $L'$ in Alg($F_i$) such that each word in $L'$ is obtained from a word in $L$ as follows. We substitute each occurrence of $x \in X_i$ by one of $\gamma_i(x) + 1$ many symbols $y$ in an alphabet $Y_i$, each of which will be assigned a value $0 \leq \eta_i(y) \leq \gamma_i(y)$. Here, we want to guarantee that in every resulting word $w \in (X_0 \uplus Y_1)^*$, we have $\eta_0(\pi_{X_0}(w)) = \eta_1(\pi_{Y_1}(w))$, meaning that the symbols in $X_0$ and in $X_1$ are replaced consistently. Formally, we have

$$Y_i = \{(x, j) \mid x \in X_i, 0 \leq j \leq \gamma_i(x)\}, \quad i = 0, 1, \quad Y = Y_0 \cup Y_1,$$

and the morphisms

$$h_i: Y_i^* \rightarrow X_i^*, \quad h: Y^* \rightarrow X^*, \quad \eta_i: Y_i^* \rightarrow N,$$

$$\begin{align*}
(x, j) &\mapsto x, \\
(x, j) &\mapsto x, \\
(x, j) &\mapsto j,
\end{align*}$$

and

$$\begin{align*}
(x, j) &\mapsto x,
\end{align*}$$

where

$$\begin{align*}
(x, j) &\mapsto x,
(x, j) &\mapsto x,
(x, j) &\mapsto j,
\end{align*}$$
and we want to construct a subset of $\hat{L} = \{ w \in h^{-1}(L) \mid \eta_0(\pi_Y(w)) = \eta_1(\pi_Y(w)) \}$ in $\text{Alg}(F_1)$. Observe that we cannot hope to find $\hat{L}$ itself in $\text{Alg}(F_1)$ in general. Take, for example, the context-free language $E = \{ a^n b^n \mid n \geq 0 \}$ and $X_0 = \{ a \}$, $X_1 = \{ b \}$, $\gamma_0(a) = 1$, $\gamma_1(b) = 1$. Then the language $\hat{E}$ would not be context-free. However, the language $E' = \{ wg(w)^R \mid w \in \{(a,0),(a,1)\}^*\}$, where $g$ is the morphism with $(a,j) \mapsto (b,j)$ for $j = 0, 1$, is context-free. Although it is only a proper subset of $\hat{E}$, it is large enough to satisfy $\pi_Y(E') = \pi_Y(\hat{E}) = \pi_Y(h^{-1}(E))$ for $i = 0, 1$. We will see that in order to construct Parikh annotations, it suffices to use such under-approximations of $L$.

**Derivation trees and matchings** In this work, by an $X$-labeled tree, we mean a finite ordered unranked tree in which each node carries a label from $X \cup \{ \varepsilon \}$ for an alphabet $X$. For each node, there is a linear order on the set of its children. For each node $x$, we write $c(x) \in X^*$ for the word obtained by reading the labels of $x$'s children in this order. Furthermore, $\text{yield}(x) \in X^*$ denotes the word obtained by reading leaf labels below the node $x$ according to the linear order induced on the leaves. Moreover, if $r$ is the root of $t$, we also write $\text{yield}(t)$ for $\text{yield}(r)$. The height of a tree is the maximal length of a path from the root to a leaf, i.e., a tree consisting of a single node has height $0$. A subtree of a tree $t$ is the tree consisting of all nodes below some node $x$ of $t$. If $x$ is a child of $t$'s root, the subtree is a direct subtree.

Let $G = (N, T, P, S)$ be a C-grammar. A partial derivation tree (for $G$) is an $(N \cup T)$-labeled tree $t$ in which (i) each inner node $x$ has a label $A \in N$ and there is some $A \to L$ in $P$ with $c(x) \in L$, and (ii) no $\varepsilon$-labeled node has a sibling. If, in addition, the root is labeled $S$ and every leaf is labeled by $T \cup \{ \varepsilon \}$, it is called a derivation tree for $G$.

Let $t$ be a tree whose leaves are $X \cup \{ \varepsilon \}$-labeled. Let $L_i$ denote the set of $X_i$-labeled leaves of $t$. An arrow collection for $t$ is a finite set $A$ together with maps $\nu_i : A \to L_i$ for $i = 0, 1$. Hence, $A$ can be thought of as a set of arrows pointing from $X_0$-labeled leaves to $X_1$-labeled leaves. We say an arrow $a \in A$ is incident to a leaf $\ell$ if $\nu_0(a) = \ell$ or $\nu_1(a) = \ell$. If $\ell$ is a leaf, then $d_A(\ell)$ denotes the number of arrows incident to $\ell$. More generally, for a subtree $s$ of $t$, $d_A(s)$ denotes the number of arrows incident to some leaf in $s$ and some leaf outside of $s$. $A$ is called a $k$-matching if (i) each leaf labeled $x \in X_i$ has precisely $\gamma_i(x)$ incident arrows, and (ii) $d_A(s) \leq k$ for every subtree $s$ of $t$.

The following lemma applies Lemma 11. The latter implies that for nodes $x$ of a derivation tree, the balance $\gamma_0(\pi_{X_0}(\text{yield}(x))) - \gamma_1(\pi_{X_1}(\text{yield}(x)))$ is bounded. This can be used to construct $k$-matchings in a bottom-up manner.

**Lemma 12.** Let $X = X_0 \uplus X_1$ and $\gamma_i : X_\ast \to \mathbb{N}$ for $i = 0, 1$ be a morphism. Let $G$ be a reduced $F_1$-grammar with $L(G) \subseteq X^*$ and $\gamma_0(\pi_{X_0}(w)) = \gamma_1(\pi_{X_1}(w))$ for every $w \in L(G)$. Then one can compute a bound $k$ such that each derivation tree of $G$ admits a $k$-matching.

We are now ready to construct the approximations necessary for obtaining PAIM.

**Proposition 13 (Consistent substitution).** Let $X = X_0 \uplus X_1$ and $\gamma_i : X_\uplus \to \mathbb{N}$ for $i = 0, 1$ be a morphism. Let $L \in \text{Alg}(F_1)$, $L \subseteq X^*$, be a language with $\gamma_0(\pi_{X_0}(w)) = \gamma_1(\pi_{X_1}(w))$ for every $w \in L$. Furthermore, let $Y_i, h_i, \eta_i$ for $i = 0, 1$ and $Y, h$ be defined as in Eq. (1) and Eq. (2). Moreover, let $L$ be given by a reduced grammar. Then one can construct a language $L' \in \text{Alg}(F_1)$, $L' \subseteq Y^*$, with

(i) $L' \subseteq h^{-1}(L)$,
(ii) $\pi_Y(L') = \pi_Y(h^{-1}(L))$ for $i = 0, 1$,
(iii) $\eta_0(\pi_{X_0}(w)) = \eta_1(\pi_{X_1}(w))$ for every $w \in L'$. 


Proof. Let \( G_0 = (N, X, P_0, S) \) be a reduced \( F_t \)-grammar with \( L(G_0) = L \). Let \( G_1 = (N, Y, P_1, S) \) be the grammar with \( P_1 = \{ A \to \vec{h}^{-1}(K) \mid A \to K \in P_0 \} \), where \( \vec{h} : (N \cup Y)^* \to (N \cup X)^* \) is the extension of \( h \) that fixes \( N \). With \( L_1 = L(G_1) \), we have \( L_1 = h^{-1}(L) \).

According to Lemma 12, we can find a \( k \in \mathbb{N} \) such that every derivation tree of \( G_0 \) admits a \( k \)-matching. With this, let \( F = \{ z \in \mathbb{Z} \mid |z| \leq k \} \), \( N_2 = N \times F \), and \( \eta \) be the morphism \( \eta : (N_2 \cup Y)^* \to \mathbb{Z} \) with \( (A, z) \mapsto z \) for \( (A, z) \in N_2 \), and \( y \mapsto \eta_0(\pi_{Y_1}(y)) - \eta_1(\pi_{Y_1}(y)) \) for \( y \in Y \). Moreover, let \( g : (N_2 \cup Y)^* \to (N \cup Y)^* \) be the morphism with \( g((A, z)) = A \) for \( (A, z) \in N_2 \) and \( g(y) = y \) for \( y \in Y \). This allows us to define the set of productions \( P_2 = \{ (A, z) \mapsto g^{-1}(K) \cap \eta^{-1}(z) \mid A \to K \in P_1 \} \). Note that since \( F_t \) is an effective Presburger closed full semi-trio, we have \( g \) is a \( k \)-matching. Furthermore, the construction guarantees that for a production \( (A, z) \to w \in G_2 \), we have \( \eta(w) = z \). In particular, every \( w \in Y^* \) with \( (S, 0) \to w \) exhibits \( \eta_0(\pi_{Y_1}(w)) - \eta_1(\pi_{Y_1}(w)) = \eta(w) = 0 \). Thus, we have shown condition (iii).

Note that the inclusion \( \subseteq \) of condition (ii) follows from condition (i). In order to prove \( \supseteq \), we shall use \( k \)-matchings in \( G_0 \) to construct derivations in \( G_2 \). See Fig. 1 for an example of the following construction of derivation trees. Let \( w \in h^{-1}(L) = L(G_1) \) and consider a derivation tree \( t \) for \( w \) in \( G_1 \). Let \( \bar{t} \) be the \( (N \cup X) \)-tree obtained from \( t \) by replacing each leaf label \( y \) in \( Y \) by \( h(y) \). Then \( \bar{t} \) is a derivation tree of \( G_0 \) and admits a \( k \)-matching \( \bar{A} \). Since \( \bar{t} \) and \( t \) are isomorphic up to labels, we can obtain a corresponding arrow collection \( A \) in \( t \) (see Fig. 1a).

Let \( L_i \) denote the set of \( Y_i \)-labeled leaves of \( t \) for \( i = 0, 1 \). Now fix \( i \in \{0, 1\} \). We choose a subset \( A' \subseteq A \) as follows. Since \( \bar{A} \) is a \( k \)-matching, each leaf \( \ell \) in \( L_i \) of \( \bar{t} \) has precisely \( \gamma_i(h(\lambda(\ell))) \geq \eta_i(\lambda(\ell)) \) incident arrows in \( A \). For each such \( \ell \) in \( L_i \), we include some arbitrary choice of \( \gamma_i(\lambda(\ell)) \) arrows in \( A' \) (see Fig. 1b). The tree \( t' \) is obtained from \( t \) by changing the label of each leaf \( \ell \in L_{1-i} \) from \((x, j)\) to \((x, j')\), where \( j' \) is the number of arrows in \( A' \) incident to \( \ell \) (see Fig. 1c). Note that since we only change labels of leaves in \( L_{1-i} \), we have \( \pi_{Y_i}(\text{yield}(t')) = \pi_{Y_i}(\text{yield}(t)) = \pi_{Y_i}(w) \).

For every subtree \( s \) of \( t' \), we define \( \beta(s) = \eta_0(\pi_{Y_0}(\text{yield}(s))) - \eta_1(\pi_{Y_1}(\text{yield}(s))) \). By construction of \( A' \), each leaf \( \ell \in L_j \) has precisely \( \eta_j(\lambda(\ell)) \) incident arrows in \( A' \) for \( j = 0, 1 \).
Therefore,
\[
\beta(s) = \sum_{\ell \in L_1 \cap s} d_A(\ell) - \sum_{\ell \in L_0 \cap s} d_A(\ell).
\]  
(3)

The absolute value of the right hand side of this equation is at most \(d_A(s)\) and hence
\[
|\eta_0(\pi_Y(yield(s))) - \eta_1(\pi_Y(yield(s)))| = |\beta(s)| \leq d_A(s) \leq d_A(s) \leq k
\]  
(4)
since \(\tilde{A}\) is a \(k\)-matching. In the case \(s = t'\), Eq. (3) also tells us that
\[
\eta_0(\pi_Y_0(yield(t'))) - \eta_1(\pi_Y_1(yield(t'))) = \sum_{\ell \in L_0} d_A'(\ell) - \sum_{\ell \in L_1} d_A'(\ell) = 0.
\]  
(5)

Let \(t''\) be the tree obtained from \(t'\) as follows: For each \(N\)-labeled node \(x\) of \(t'\), we replace the label \(B\) of \(x\) with \((B, \beta(s))\), where \(s\) is the subtree below \(x\) (see Fig. 1d). By Eq. (4), this is a symbol in \(N_2\). The root node of \(t''\) has label \((S, 0)\) by Eq. (5). Furthermore, it follows by an induction on the height of subtrees that if \((B, z)\) is the label of a node \(x\), then \(z = \eta(c(x))\). Hence, the tree \(t''\) is a derivation tree of \(G_2\). This means \(\pi_Y(w) = \pi_Y(yield(t'')) = \pi_Y(yield(t'')) \in L(G_2) = L'\), completing the proof of condition (ii).

Proposition 13 now allows us to construct PAIM for languages \(\sigma(L)\), where \(\sigma\) is a letter substitution. The essential idea is to use a PAIM \((K, C, P, (P_c)_{c \in C}, \varphi, \sigma)\) for \(L\) and then apply Proposition 13 to \(K\) with \(X_0 = Z \cup \{\varnothing\}\) and \(X_1 = C \cup P\). One can clearly assume that a single letter \(a\) from \(Z\) is replaced by \((a, b) \subseteq Z'\). We can therefore choose \(\gamma_0(w)\) to be the number of \(a\)'s in \(w\) and \(\gamma_1(w)\) to be the number of \(a\)'s represented by symbols from \(C \cup P\) in \(w\). Then the counting property of \(K\) entails \(\gamma_0(w) = \gamma_1(w)\) for \(w \in K\) and thus applicability of Proposition 13. Condition (ii) then yields the projection property for \(i = 0\) and the commutative projection property for \(i = 1\) and condition (iii) yields the counting property for the new PAIM.

\[\text{Lemma 14 (Letter substitution). Let } \sigma : Z \rightarrow \mathcal{P}(Z') \text{ be a letter substitution. Given } i \in \mathbb{N} \text{ and a PAIM for } L \in G_i \text{ in } G_i, \text{ one can construct a PAIM for } \sigma(L).\]

The basic idea for the case of general substitutions is to replace each \(x\) by a PAIM for \(\sigma(x)\). Here, Lemma 14 allows us to assume that the PAIM for each \(\sigma(x)\) is linear. However, we have to make sure that the number of occurrences of \(\varnothing\) remains bounded.

\[\text{Lemma 15 (Substitutions). Let } L \subseteq X^* \text{ in } G_i \text{ and } \sigma \text{ be a } G_i\text{-substitution. Given a PAIM for } L \text{ in } G_i \text{ for each } \sigma(x), \text{ one can construct a PAIM for } \sigma(L) \text{ in } G_i.\]

The next step is to construct PAIM for languages \(L(G)\), where \(G\) has just one nonterminal \(S\) and PAIM are given for the right-hand-sides. Here, it suffices to obtain a PAIM for \(\text{SF}(G)\) in the case that \(S\) occurs in every word on the right hand side: Then \(L(G)\) can be obtained from \(\text{SF}(G)\) using a substitution. Applying \(S \rightarrow R\) then means that for some \(w \in R\), \(\Psi(w) - S\) is added to the Parikh image of the sentential form. Therefore, computing a PAIM for \(\text{SF}(G)\) is akin to computing a semilinar representation for \(S^\oplus\), where \(S\) is semilinar.

\[\text{Lemma 16 (One nonterminal). Let } G \text{ be a } G_i\text{-grammar with one nonterminal. Furthermore, suppose } \text{PAIM in } G_i \text{ are given for the right-hand-sides in } G. \text{ Then we can construct a PAIM for } L(G) \text{ in } G_i.\]

Using Lemmas 15 and 16, we can now construct PAIM recursively with respect to the number of nonterminals in \(G\).
Lemma 17 (PAIM for algebraic extensions). Given \( i \in \mathbb{N} \) and an \( F_i \)-grammar \( G \), along with a PAIM in \( F_i \) for each right hand side, one can construct a PAIM for \( L(G) \) in \( G_i \).

The last step is to compute PAIM for languages in \( \text{SLi}(G_i) \). Then, Theorem 10 follows.

Lemma 18 (PAIM for semilinear intersections). Given \( i \in \mathbb{N} \), a language \( L \subseteq X^* \) in \( G_i \), a semilinear set \( S \subseteq X^\oplus \), and a morphism \( h : X^* \rightarrow Y^* \), along with a PAIM in \( G_i \) for \( L \), one can construct a PAIM for \( h(L \cap \Psi^{-1}(S)) \) in \( \text{SLi}(G_i) \).

5 Computing downward closures

The procedure for computing downward closures works recursively with respect to the hierarchy \( F_0 \subseteq G_0 \subseteq \cdots \). For languages in \( G_i = \text{Alg}(F_i) \), we use an idea by van Leeuwen [14], who proved that downward closures are computable for \( \text{Alg}(C) \) if and only if this is the case for \( C \). This means we can compute downward closures for \( G_i \) if we can compute them for \( F_i \). For the latter, we use Lemma 19, which is based on the following idea. Using a PAIM for \( L \) in \( G_i \), one constructs a language \( L' \supseteq L \cap \Psi^{-1}(S) \) in which every word admits insertions that yield a word in \( L \cap \Psi^{-1}(S) \), meaning that \( L' \downarrow = (L \cap \Psi^{-1}(S)) \downarrow \). Here, \( L' \) is obtained from the PAIM using a rational transduction, which implies \( L' \in G_i \).

Lemma 19. Given \( i \in \mathbb{N} \), a language \( L \subseteq X^* \) in \( G_i \), and a semilinear set \( S \subseteq X^\oplus \), one can compute a language \( L' \in G_i \) with \( L' \downarrow = (L \cap \Psi^{-1}(S)) \downarrow \).

Proof. We call \( \alpha \in X^\oplus \) a submultiset of \( \beta \in X^* \) if \( \alpha(x) \leq \beta(x) \) for each \( x \in X \). In analogy with words, we write \( T \downarrow \) for the set of all submultisets of elements of \( T \) for \( T \subseteq X^\oplus \). We use Theorem 10 to construct a PAIM \( (K, C, P_i(P_i)_{i \in C}, \varphi, \circ) \) for \( L \) in \( G_i \). For each \( c \in C \), consider the set \( S_c = \{ \mu \in P_i^\oplus | \varphi(c + \mu) \in S \} \). Since \( \leq \) is a well-quasi-ordering on \( X^\oplus \) [6], membership in \( S_c \downarrow \) can be characterized by a finite set of forbidden submultisets, which is Presburger definable and thus computable. Therefore, the language \( \Psi^{-1}(S_c \downarrow) \) is effectively regular. Hence, the language

\[
L' = \{ \pi_X(cv) | c \in C, cv \in K, \pi_{P_i}(v) \in \Psi^{-1}(S_c \downarrow) \}.
\]

effectively belongs to \( G_i \), since \( G_i \) is an effective full semi-AFL. We claim that \( L \cap \Psi^{-1}(S) \subseteq L' \subseteq (L \cap \Psi^{-1}(S)) \downarrow \). The latter clearly implies \( L' \downarrow = (L \cap \Psi^{-1}(S)) \downarrow \).

The counting property of the PAIM entails the inclusion \( L \cap \Psi^{-1}(S) \subseteq L' \). In order to show \( L' \subseteq (L \cap \Psi^{-1}(S)) \downarrow \), suppose \( w \in L' \). Then there is a \( cv \in K \) with \( w = \pi_X(cv) \) and \( \pi_{P_i}(v) \in \Psi^{-1}(S_c \downarrow) \). This means there is a \( \nu \in P_i^\oplus \) with \( \Psi(\pi_{P_i}(v)) + \nu \in S_c \). The insertion property of \( (K, C, P_i(P_i)_{i \in C}, \varphi, \circ) \) allows us to find a word \( v' \in L \) such that

\[
\Psi(v') = \Psi(\pi_X(cv)) + \varphi(\nu), \quad \pi_{X \cup \{ \circ \}}(cv) \preceq v'.
\]

By definition of \( S_c \), the first part of Eq. (6) implies that \( \Psi(v') \in S \). The second part of Eq. (6) means in particular that \( w = \pi_X(cv) \preceq v' \). Thus, we have \( w \preceq v' \in L \cap \Psi^{-1}(S) \). ◀

Theorem 20. Given a language \( L \) in \( F \), one can compute a finite automaton for \( L \downarrow \).

Proof. We perform the computation recursively with respect to the level of the hierarchy.

- If \( L \in F_0 \), then \( L \) is finite and we can clearly compute \( L \downarrow \).
- If \( L \in F_0 \) with \( i \geq 1 \), then \( L = h(L' \cap \Psi^{-1}(S)) \) for some \( L' \subseteq X^* \) in \( G_{i-1} \), a semilinear set \( S \subseteq X^\oplus \), and a morphism \( h \). Since \( h(M) \downarrow = h(M \downarrow) \) for any \( M \subseteq X^* \), it suffices to describe how to compute \( (L' \cap \Psi^{-1}(S)) \downarrow \). Using Lemma 19, we construct a language \( L'' \in G_{i-1} \) with \( L'' \downarrow = (L' \cap \Psi^{-1}(S)) \downarrow \) and then recursively compute \( L'' \downarrow \).
If \( L \in G_i \), then \( L \) is given by an \( F_i \)-grammar \( G \). Using recursion, we can compute the downward closure of each right-hand-side of \( G \). We obtain a new REG-grammar \( G' \) by replacing each right-hand-side in \( G \) with its downward closure. Then \( L(G') \downarrow = L \downarrow \). Since we can construct a context-free grammar for \( L(G') \), we can compute \( L(G') \downarrow \) using the available algorithms by van Leeuwen [13] or Courcelle [5].

### 6 Strictness of the hierarchy

In this section, we present another application of Parikh annotations. Using PAIM, one can show that the inclusions \( F_0 \subseteq G_0 \subseteq F_1 \subseteq G_1 \subseteq \cdots \) in the hierarchy are, in fact, all strict. It is of course easy to see that \( F_0 \subseteq G_0 \subseteq F_1 \), since \( F_0 \) contains only finite sets and \( F_1 \) contains, for example, \( \{a^n b^n c^n \mid n \geq 0\} \). In order to prove strictness at higher levels, we present two transformations: The first turns a language from \( F_i \setminus G_{i-1} \) into one in \( G_i \setminus F_i \) (Proposition 21) and the second turns one from \( G_i \setminus F_i \) into one in \( F_{i+1} \setminus G_i \) (Proposition 25).

The essential idea of the next proposition is as follows. For the sake of simplicity, assume \( (L\#)^* = L' \cap \Psi^{-1}(S) \) for \( L' \in \mathcal{C}, L' \subseteq (X \cup \{\#\})^* \). Consider a PAIM \( (K', C, P, (P_e)_{e \in E}, \psi, \phi) \) for \( L' \in \mathcal{C} \). Similar to Lemma 19, we obtain from \( K' \) a language \( \hat{L} \subseteq (X \cup \{\#\})^* \) in \( \mathcal{C} \) such that every member of \( \hat{L} \) admits an insertion at \( \circ \) that yields a word from \( (L\#)^* = L' \cap \Psi^{-1}(S) \).

Using rational transduction, we can then pick all words that appear between two \( \# \) in some member of \( \hat{L} \) and contain no \( \circ \). Since there is a bound on the number of \( \circ \) in \( K' \) (and hence in \( \hat{L} \)), every word from \( L \) has to occur in this way. On the other hand, since inserting at \( \circ \) yields a word in \( (L\#)^* \), every such word without \( \circ \) must be in \( L \).

**Proposition 21.** Let \( \mathcal{C} \) be a full trio such that every language in \( \mathcal{C} \) has a PAIM in \( \mathcal{C} \). Moreover, let \( X \) be an alphabet with \( \# \notin X \). If \( (L\#)^* \in \text{SLI}(\mathcal{C}) \) for \( L \subseteq X^* \), then \( L \in \mathcal{C} \).

Using induction on the structure of a rational expression, it is not hard to show that we can construct PAIM for regular languages. This means Propositions 2 and 21 imply the following, which might be of independent interest.

**Corollary 22.** Let \( L \subseteq X^*, \# \notin X \), and \( (L\#)^* \in \text{VA}(\mathbb{Z}^n) \). Then \( L \) is regular.

In order to prove Proposition 25, we need a new concept. A bursting grammar is one in which essentially (meaning: aside from a subsequent replacement by terminal words of bounded length) the whole word is generated in a single application of a production.

**Definition 23.** Let \( \mathcal{C} \) be a language class and \( k \in \mathbb{N} \). A \( \mathcal{C} \)-grammar \( G \) is called \( k \)-bursting if for every derivation tree \( t \) for \( G \) and every node \( x \) of \( t \) we have: \( \text{yield}(x) \geq k \) implies \( \text{yield}(x) = \text{yield}(t) \). A grammar is said to be bursting if it is \( k \)-bursting for some \( k \in \mathbb{N} \).

**Lemma 24.** If \( \mathcal{C} \) is a union closed full semi-trio and \( G \) a bursting \( \mathcal{C} \)-grammar, then \( L(G) \in \mathcal{C} \).

The essential idea for Proposition 25 is the following. We construct a \( \mathcal{C} \)-grammar \( G' \) for \( L \) by removing from a \( \mathcal{C} \)-grammar \( G \) for \( M = (L \cup \{a^n b^n c^n \mid n \geq 0\}) \cap a^*(bX)^*c^* \) all terminals \( a, b, c \). Using Lemma 11, one can then show that \( G' \) is bursting.

**Proposition 25.** Let \( \mathcal{C} \) be a union closed full semi-trio and let \( a, b, c \notin X \) and \( L \subseteq X^* \). If \( L \cup \{a^n b^n c^n \mid n \geq 0\} \in \text{Alg}(\mathcal{C}) \), then \( L \in \mathcal{C} \).

**Theorem 26.** For \( i \in \mathbb{N} \), define the alphabets \( X_0 = \emptyset, Y_i = X_i \cup \{\#\}, X_{i+1} = Y_i \cup \{a_{i+1}, b_{i+1}, c_{i+1}\} \). Moreover, define \( U_i \subseteq X_i^* \) and \( V_i \subseteq Y_i^* \) as \( U_0 = \{\emptyset\}, V_i = (U_i \setminus \#)^* \), and \( U_{i+1} = V_i \cup \{a_{i+1} b_{i+1} c_{i+1} \mid n \geq 0\} \) for \( i \geq 0 \). Then \( V_i \in G_i \setminus F_i \) and \( U_{i+1} \in F_{i+1} \setminus G_i \).
A Proof of Theorem 1

In order to prove Theorem 1, we define the relevant notions in detail.
REFERENCES

Let \( A \) be a (not necessarily finite) set of symbols and \( R \subseteq A^* \times A^* \). The pair \((A, R)\) is called a (monoid) presentation. The smallest congruence of \( A^* \) containing \( R \) is denoted by \( \equiv_R \) and we will write \([w]_R\) for the congruence class of \( w \in A^* \). The monoid presented by \((A, R)\) is defined as \( A^*/\equiv_R \). For the monoid presented by \((A, R)\), we also write \( \langle A \mid R \rangle \), where \( R \) is denoted by equations instead of pairs.

Note that since we did not impose a finiteness restriction on \( A \), every monoid has a presentation. Furthermore, for monoids \( M_1, M_2 \) we can find presentations \((A_1, R_1)\) and \((A_2, R_2)\) such that \( A_1 \cap A_2 = \emptyset \). We define the free product \( M_1 \ast M_2 \) to be presented by \((A_1 \cup A_2, R_1 \cup R_2)\). Note that \( M_1 \ast M_2 \) is well-defined up to isomorphism. By way of the injective morphisms \([w]_{R_i} \mapsto [w]_{R_i \cup R_j}, w \in A_i^* \) for \( i = 1, 2 \), we will regard \( M_1 \) and \( M_2 \) as subsets of \( M_1 \ast M_2 \). It is a well-known property of free products that if \( \varphi_i : M_i \to N \) is a morphism for \( i = 1, 2 \), then there is a unique morphism \( \varphi : M_1 \ast M_2 \to N \) with \( \varphi|_{M_i} = \varphi_i \) for \( i = 1, 2 \). Furthermore, if \( u_0v_1u_2 \cdots v_nu_n = 1 \) for \( u_0, \ldots, u_n \in M_1 \) and \( v_1, \ldots, v_n \in M_2 \) (or vice versa), then \( u_j = 1 \) or \( v_j = 1 \) for some \( 0 \leq j \leq n \). Moreover, we write \( M^{(n)} \) for the \( n \)-fold free product \( M \ast \cdots \ast M \).

One of the directions of the equality \( \varphi \mathcal{A}(B \ast B \ast M) = \mathcal{A}(\varphi \mathcal{A}(M)) \) follows from previous results. In \cite{19} (and, for a more general product construction, in \cite{4}), the following was shown.

\begin{theorem} ([19, 4]) \end{theorem}

Let \( M_0 \) and \( M_1 \) be monoids. Then \( \varphi \mathcal{A}(M_0 \ast M_1) \subseteq \mathcal{A}(\varphi \mathcal{A}(M_0) \cup \varphi \mathcal{A}(M_1)) \).

Let \( M \) and \( N \) be monoids. In the following, we write \( M \twoheadrightarrow N \) if there is a morphism \( \varphi : M \to N \) such that \( \varphi^{-1}(1) = \{1\} \). Clearly, if \( M \twoheadrightarrow N \), then \( \varphi \mathcal{A}(M) \subseteq \varphi \mathcal{A}(N) \): Replacing \( \varphi \mathcal{A}(M) \) with \( \varphi \mathcal{A}(N) \), it suffices to prove \( \varphi \mathcal{A}(B \ast M) = \mathcal{A}(\varphi \mathcal{A}(M)) \).

\begin{lemma} \end{lemma}

If \( M \twoheadrightarrow M' \) and \( N \twoheadrightarrow N' \), then \( M \ast N \twoheadrightarrow M' \ast N' \).

\begin{proof} \end{proof}

Let \( \varphi : M \to M' \) and \( \psi : N \to N' \). Then the morphism \( \kappa : M \ast N \to M' \ast N' \) with \( \kappa|_{M} = \varphi \) and \( \kappa|_{N} = \psi \) clearly satisfies \( \kappa^{-1}(1) = 1 \).

We will use the notation \( R_1(M) = \{a \in M \mid \exists b \in M : ab = 1\} \).

\begin{lemma} \end{lemma}

Let \( M \) be a monoid with \( R_1(M) \neq \{1\} \). Then \( B^{(n)} \ast M \twoheadrightarrow B \ast M \) for every \( n \geq 1 \). In particular, \( \varphi \mathcal{A}(B \ast B \ast M) = \mathcal{A}(\varphi \mathcal{A}(B^{(n)} \ast M)) \) for every \( n \geq 1 \).

\begin{proof} \end{proof}

If \( B^{(n)} \ast M \twoheadrightarrow B \ast B \ast M \) and \( B \ast B \ast M \twoheadrightarrow B \ast M \), then \( B^{(n+1)} \ast M \cong (B^{(n)} \ast M) \twoheadrightarrow B \ast (B \ast M) \twoheadrightarrow B \ast M \).

Therefore, it suffices to prove \( B \ast B \ast M \twoheadrightarrow B \ast M \).

Let \( B_s = \{s, s \bar{s} \mid s \bar{s} = 1\} \) for \( s \in \{p, q, r\} \). We show \( B_p \ast B_q \ast M \twoheadrightarrow B_r \ast M \). Suppose \( M \) is presented by \( (X, R) \). We regard the monoids \( B_p \ast B_q \ast M \) and \( B_r \ast M \) as embedded into \( B_p \ast B_q \ast B_r \ast M \), which by definition of the free product, has a presentation \( (Y, S) \), where \( Y = \{p, \bar{p}, q, \bar{q}, r, \bar{r}\} \cup X \) and \( S \) consists of \( R \) and the equations \( s \bar{s} = 1 \) for \( s \in \{p, q, r\} \). For \( w \in Y^* \), we write \([w]_R\) for the congruence class generated by \( S \). Since \( R_1(M) \neq \{1\} \), we find \( u, v \in X^* \) with \([uv]_R = 1 \) and \([u]_R \neq 1 \). Then let \( \varphi : (\{p, \bar{p}, q, \bar{q}\} \cup X)^* \to (\{r, \bar{r}\} \cup X)^* \) be the morphism with \( \varphi(x) = x \) for \( x \in X \) and

\[
\begin{align*}
p &\mapsto rr, \\
\bar{p} &\mapsto \bar{r}
\end{align*}
\]

\[
\begin{align*}
q &\mapsto rur, \\
\bar{q} &\mapsto \bar{r}vr.
\end{align*}
\]
We show by induction on $|w|$ that $|\varphi(w)| = 1$ implies $|w| = 1$. Since this is trivial for $w = \varepsilon$, we assume $|u| \geq 1$. Now suppose $|\varphi(w)| = |\varepsilon|$ for some $w \in \{\{p, \bar{p}, q, \bar{q}\} \cup X\}^*$. If $w \in X^*$, then $|\varphi(w)| = |w|$ and hence $|w| = 1$. Otherwise, we have $\varphi(w) = xry\bar{r}z$ for some $y \in X^*$ with $|y| = 1$ and $|xz| = 1$. This means $w = fsy\bar{s}g$ for $s, s' \in \{p, q\}$ with $\varphi(fs) = xr$ and $\varphi(\bar{s}g) = \bar{r}z$. If $s \neq s'$, then $s = p$ and $s' = q$; or $s = q$ and $s' = p$. In the former case

$$|\varphi(w)| = |\varphi(f) \cdot r \cdot y \cdot \bar{r} \cdot \varphi(g)| = |\varphi(f) \cdot r \cdot u \cdot \bar{r} \cdot \varphi(g)| \neq 1$$

since $|v| \neq 1$ and in the latter

$$|\varphi(w)| = |\varphi(f) \cdot r \cdot u \cdot \bar{r} \cdot \varphi(g)| = |\varphi(f) \cdot r \cdot u \cdot \bar{r} \cdot \varphi(g)| \neq 1$$

since $|u| \neq 1$. Hence $s = s'$. This means $1 = |w| = |fsy\bar{s}g| = |fg|$ and $1 = |\varphi(w)| = |\varphi(fg)|$ and since $|fg| < |w|$, induction yields $|w| = |fg| = 1$.

Hence, we have shown that $|\varphi(w)| = 1$ implies $|w| = 1$. Since, on the other hand, $|u| = |v|$ implies $|\varphi(u)| = |\varphi(v)|$ for all $u, v \in \{\{p, \bar{p}, q, \bar{q}\} \cup X\}^*$, we can lift $\varphi$ to a morphism witnessing $\mathcal{B}_p \ast \mathcal{B}_q \ast M \rightarrow \mathcal{B}_r \ast M$.

**Proof of Theorem 1.** It suffices to prove the first statement: If $R_1(M) \neq \{1\}$, then by Lemma 29, $VA(\mathcal{B} \ast M) = VA(\mathcal{B} \ast \mathcal{B} \ast M)$. Since $VA(\mathcal{B}) \subseteq CF$, Theorem 27 yields

$$VA(\mathcal{B} \ast N) \subseteq Alg(\mathcal{B} \ast VA(N)) \subseteq Alg(\mathcal{B} \ast VA(N))$$

for every monoid $N$. Therefore,

$$VA(\mathcal{B} \ast \mathcal{B} \ast M) \subseteq Alg(\mathcal{B} \ast VA(M)) \subseteq Alg(\mathcal{B} \ast VA(M)) = Alg(\mathcal{B} \ast VA(M)).$$

It remains to be shown that $Alg(\mathcal{B} \ast VA(M)) \subseteq VA(\mathcal{B} \ast \mathcal{B} \ast M)$.

Let $G = (N, T, P, S)$ be a reduced $VA(M)$-grammar and let $X = N \cup T$. Since $\mathcal{B} \ast VA(M)$ is closed under union, we may assume that for each $B \in N$, there is exactly one production $B \rightarrow L_B$ in $P$. For each $B \in N$, let $A_B = (Q_B, X, M, E_B, q_0^B, F_B)$ by a valence automaton over $M$ with $L(A_B) = L_B$. We may clearly assume that $Q_B \cap Q_C = \emptyset$ for $B \neq C$ and that for each $(p, w, m, q) \in E_B$, we have $|w| \leq 1$.

In order to simplify the correctness proof, we modify $G$. Let $[ \cdot ]$ and $\{ \cdot \}$ be new symbols and let $G'$ be the grammar $G' = (N, T \cup \{[\cdot]\}, P', S)$, where $P'$ consists of the productions $B \rightarrow [L]$ for $B \rightarrow L \in P$. Moreover, let

$$K = \{ v \in (N \cup T \cup \{[\cdot]\})^* \mid u \Rightarrow_{G'}^* v, u \in L_S \}.$$

Then $L(G') = \pi_T(K \cap (T \cup \{[\cdot]\})^*)$ and it suffices to show $K \in VA(\mathcal{B} \ast \mathcal{B} \ast M)$.

Let $Q = \bigcup_{B \in N} Q_B$. For each $q \in Q$, let $\mathcal{B}_q = \langle q, q \mid qq = 1 \rangle$ be an isomorphic copy of $\mathcal{B}$. Let $M' = \mathcal{B}_{q_0} \ast \cdots \ast \mathcal{B}_{q_n} \ast M$, where $Q = \{q_1, \ldots, q_n\}$. We shall prove $K \in VA(M')$, which implies $K \in VA(\mathcal{B} \ast \mathcal{B} \ast M)$ by Lemma 29 since $R_1(\mathcal{B} \ast M) \neq \{1\}$.

Let $E = \bigcup_{B \in N} E_B, F = \bigcup_{B \in N} F_B$. The new set $E'$ consists of the following transitions:

\begin{align*}
(p, x, m, q) & \quad \text{for } (p, x, m, q) \in E, \\
(p, [\cdot, mq, q_0^B]) & \quad \text{for } (p, B, m, q) \in E, \; B \in N, \\
(p, [\cdot, \bar{q}, q]) & \quad \text{for } p \in F, \; q \in Q.
\end{align*}

We claim that with $A' = (Q, N \cup T \cup \{[\cdot]\}, M', E', q_0^B, F)$, we have $L(A') = K$.

Let $v \in K$, where $u \Rightarrow_{G'}^* v$ for some $u \in L_S$. We show $v \in L(A')$ by induction on $n$. For $n = 0$, we have $v \in L_S$ and can use transitions of type (7) inherited from $A_S$ to accept $v$. If

**REFERENCES**

15
REFERENCES

$n \geq 1$, let $u \Rightarrow_G^{n-1} v' \Rightarrow_G v$. Then $v' \in L(A')$ and $v' = xBy$, $v = x|w|y$ for some $B \in N$, $w \in L_B$. The run for $v'$ uses a transition $(p, B, m, q) \in E$. Instead of using this transition, we can use $(p, \lfloor m \rfloor q q, \emptyset)$, then execute the (7)-type transitions for $w \in L_B$, and finally use $(f, \lfloor f \rfloor q, q)$, where $f$ is the final state in the run for $w$. This has the effect of reading $|w|$ from the input and multiplying $mqq = m$ to the storage monoid. Hence, the new run is valid and accepts $v$. Hence, $v \in L(A')$. This proves $K \subseteq L(A')$.

In order to show $L(A') \subseteq K$, consider the morphisms $\varphi : (T \cup \lfloor \rfloor)^* \rightarrow \mathbb{B}$, $\psi : M' \rightarrow \mathbb{B}$ with $\varphi(x) = 1$ for $x \in T$, $\varphi(\lfloor \rfloor) = a$, $\varphi(q) = a$ for $q \in Q$, $\psi(q) = a$, and $\psi(m) = 1$ for $m \in M$. The transitions of $A'$ are constructed such that $(p, \varepsilon, 1) \rightarrow_{A'}^*_q (q, w, m)$ implies $\varphi(w) = \psi(m)$. In particular, if $v \in L(A')$, then $\pi_{\lfloor \rfloor}(v)$ is a semi-Dyck word with respect to $\lfloor \rfloor$ and $\lfloor \rfloor$ can only be produced by transitions of the form (8) and (9), respectively, the run for $v$ has to be of the form

$$(q_0^S, \varepsilon, 1) \rightarrow_{A'}^* (p, x, r)$$

$$
\rightarrow_{A'}^* (q_0^B, x, mqs)$$

$$
\rightarrow_{A'}^* (q', x|w|y, rmqsq't)$$

for some $p, q, q' \in Q$, $B \in N$, $(p, B, m, q') \in E$, $f, f' \in F$, $r, t \in M'$, and $s \in M$ and with $rmqsq't = 1$. This last condition implies $s = 1$ and $q = q'$, which in turn entails $rmq = 1$. This also means $(p, B, m, q') = (p, B, m, q) \in E$ and $(q_0^B, \varepsilon, 1) \rightarrow_{A'}^* (f, w, s) = (f, w, 1)$ and hence $w \in L_B$. Using the transition $(p, B, m, q') \in E$, we have

$$(q_0^S, \varepsilon, 1) \rightarrow_{A'}^* (p, x, r)$$

$$
\rightarrow_{A'}^* (q', xB, rm)$$

$$
\rightarrow_{A'}^* (f', xBy, rmt).$$

Hence $xBy \in L(A')$ and $|xBy| < |v|$. Thus, induction yields $xBy \in K$ and since $xBy \Rightarrow_G x|w|y$, we have $v = x|w|y \in K$. This establishes $L(A') = K$.

**B Proof of Proposition 2**

**Proof.** We start with the inclusion “$\subseteq$”. Since the right-hand side is closed under morphisms and union, it suffices to show that for each $L \in \mathcal{VA}(M)$, $L \subseteq X^*$, and semilinear $S \subseteq X^\oplus$, we have $L \cap \Psi^{-1}(S) \in \mathcal{VA}(M \times \mathbb{Z}^n)$ for some $n \geq 0$. Let $n = |X|$ and pick a linear order on $X$. This induces an embedding $X^\oplus \rightarrow \mathbb{Z}^n$, by way of which we consider $X^\oplus$ as a subset of $\mathbb{Z}^n$.

Suppose $L = L(A)$ for a valence automaton $A$ over $M$. The new valence automaton $A'$ over $M \times \mathbb{Z}^n$ simulates $A$ and, if $w$ is the input read by $A$, adds $\Psi(w)$ to the $\mathbb{Z}^n$ component of the storage monoid. When $A$ reaches a final state, $A'$ nondeterministically changes to a new state $q_1$, in which it nondeterministically subtracts an element of $S$ from the $\mathbb{Z}^n$ component. Afterwards, $A'$ switches to another new state $q_2$, which is the only accepting state in $A'$. Clearly, $A'$ accepts a word $w$ if and only if $w \in L(A)$ and $\Psi(w) \in S$, hence $L(A') = L(A) \cap \Psi^{-1}(S)$. This proves “$\subseteq$”. 

}$
Suppose \( L = L(A) \) for some valence automaton \( A = (Q, X, M \times \mathbb{Z}^n, E, q_0, F) \). We construct a valence automaton \( A' \) over \( M \) as follows. The input alphabet \( X' \) of \( A' \) consists of all those \( (w, \mu) \in X^* \times \mathbb{Z}^n \) for which there is an edge \((p, w, (m, \mu), q) \in E \) for some \( p, q \in Q, m \in M \). \( A' \) has edges

\[
E' = \{ (p, (w, \mu), m, q) \mid (p, w, (m, \mu), q) \in E \}.
\]

In other words, whenever \( A \) reads \( w \) and adds \((m, \mu) \in M \times \mathbb{Z}^n \) to its storage monoid, \( A' \) adds \( m \) and reads \((w, \mu) \) from the input. Let \( \psi: X^\# \to \mathbb{Z}^n \) be the morphism that projects the symbols in \( X' \) to the right component and let \( h: X^\# \to X^* \) be the morphism that projects the symbols in \( X' \) to the left component. Note that the set \( S = \psi^{-1}(0) \subseteq X^\# \) is Presburger definable and hence effectively semilinear. We clearly have \( L(A) = h(L(A') \cap \psi^{-1}(S)) \in \text{SLI}(\text{VA}(M)) \). This proves \( \geq \). Clearly, all constructions in the proof can be carried out effectively.

\section{Proof of Proposition 5}

\begin{itemize}
  \item \textbf{Proposition 30.} Let \( \mathcal{C} \) be an effective full semi-trio. Then \( \text{Alg}(\mathcal{C}) \) is an effective full semi-AFL.
  
  \textbf{Proof.} Since \( \text{Alg}(\mathcal{C}) \) is clearly effectively closed under union, we only prove effective closure under rational transductions.

  Let \( G = (N, T, P, S) \) be a \( \mathcal{C} \)-grammar and let \( U \subseteq X^* \times T^* \) be a rational transduction. Since we can easily construct a \( \mathcal{C} \)-grammar for \( aL(G) \) (just add a production \( S \to \{aS\} \)) and the rational transduction \( (\varepsilon, a)U = \{(v, au) \mid (v, u) \in U\} \), we may assume that \( L(G) \subseteq T^+ \).

  Let \( U \) be given by the automaton \( A = (Q, X^* \times T^*, E, q_0, F) \). We may assume that

\[
E \subseteq Q \times ((X \times \{\varepsilon\}) \cup \{(\varepsilon \times T)\}) \times Q
\]

and \( F = \{f\} \). We regard \( Z = Q \times T \times Q \) and \( N' = Q \times N \times Q \) as alphabets. For each \( p, q \in Q \), let \( U_{p,q} \subseteq N' \times (N \cup T)^* \) be the transduction such that for \( w = w_1 \cdots w_n, w_1, \ldots, w_n \in N \cup T, n \geq 1 \), the set \( U_{p,q}(w) \) consists of all words

\[
(p, w_1, q_1)(q_1, w_2, q_2) \cdots (q_{n-1}, w_n, q)
\]

with \( q_1, \ldots, q_{n-1} \in Q \). Moreover, let \( U_{p,q}(\varepsilon) = \{\varepsilon\} \) if \( p = q \) and \( U_{p,q}(\varepsilon) = \emptyset \) if \( p \neq q \). Observe that \( U_{p,q} \) is locally finite. The new grammar \( G' = (N', Z, P', (q_0, S, f)) \) has productions \((p, B, q) \to U_{p,q}(L)\) for each \( p, q \in Q \) and \( B \to L \in P \). Let \( \sigma: Z^\ast \to \mathcal{P}(X^\ast) \) be the regular substitution defined by

\[
\sigma((p, x, q)) = \{ w \in X^\ast \mid (p, (x, q)) \to_A^* (q, (w, x)) \}.
\]

We claim that \( U(L(G)) = \sigma(L(G')) \). First, it can be shown by induction on the number of derivation steps that \( SF(G') = U_{q_0,f}(SF(G)) \). This implies \( L(G') = U_{q_0,f}(L(G)) \). Since for every language \( K \subseteq T^+ \), we have \( \sigma(U_{q_0,f}(K)) = UK \), we may conclude \( \sigma(L(G')) = U(L(G)) \).

\( \text{Alg}(\mathcal{C}) \) is clearly effectively closed under \( \text{Alg}(\mathcal{C}) \)-substitutions. Since \( \mathcal{C} \) contains the finite languages, this means \( \text{Alg}(\mathcal{C}) \) is closed under \( \text{REG} \)-substitutions. Hence, we can construct a \( \mathcal{C} \)-grammar for \( U(L(G)) = \sigma(L(G')) \).

\item \textbf{Proposition 31.} Let \( \mathcal{C} \) be an effective full semi-AFL. Then \( \text{SLI}(\mathcal{C}) \) is an effective Presburger closed full trio. In particular, \( \text{SLI}(\text{SLI}(\mathcal{C})) = \text{SLI}(\mathcal{C}) \).
\end{itemize}
Proof. Let \( L \in \mathcal{C} \), \( L \subseteq X^* \), \( S \subseteq X^\oplus \) semilinear, and \( h: X^* \to Y^* \) be a morphism. If \( T \subseteq Z^* \times Y^* \) is a rational transduction, then \( Th(L \cap \Psi^{-1}(S)) = U(L \cap \Psi^{-1}(S)) \), where \( U \subseteq Z^* \times X^* \) is the rational transduction \( U = \{(v, u) \in Z^* \times X^* \mid (v, h(u)) \in T \} \). We may assume that \( X \cap Z = \emptyset \). Construct a regular language \( R \subseteq (X \cup Z)^* \) with \( U = \{(\pi_Z(w), \pi_X(w)) \mid w \in R \} \). With this, we have
\[
U(L \cap \Psi^{-1}(S)) = \pi_Z \left((R \cap (L \cdot w Z^*)) \cap \Psi^{-1}(S + Z^\oplus)\right).
\]
Since \( \mathcal{C} \) is an effective full semi-AFL, and thus \( R \cap (L \cdot w Z^*) \) is effectively in \( \mathcal{C} \), the right hand side is effectively contained in \( \text{SLI}(\mathcal{C}) \). This proves that \( \text{SLI}(\mathcal{C}) \) is an effective full trio.

Let us prove effective closure under union. Now suppose \( L_1 \subseteq X_1^* \), \( S_i \subseteq X_i^\oplus \), and \( h_i: X_i^* \to Y_i^* \) for \( i = 1, 2 \). If \( \bar{X}_2 \) is a disjoint copy of \( X_2 \) with bijection \( \varphi: X_2 \to \bar{X}_2 \), then
\[
h_1(L_1 \cap \Psi^{-1}(S_1)) \cup h_2(L_2 \cap \Psi^{-1}(S_2)) = h((L_1 \cup \varphi(L_2)) \cap \Psi^{-1}(S_1 \cup \varphi(S_2))),
\]
where \( h: X_1 \cup \bar{X}_2 \to Y \) is the map with \( h(x) = h_1(x) \) for \( x \in X_1 \) and \( h(x) = h_2(\varphi(x)) \) for \( x \in \bar{X}_2 \). This proves that \( \text{SLI}(\mathcal{C}) \) is effectively closed under union.

It remains to be shown that \( \text{SLI}(\mathcal{C}) \) is Presburger closed. Suppose \( L \in \mathcal{C} \), \( L \subseteq X^* \), \( S \subseteq X^\oplus \) is semilinear, \( h: X^* \to Y^* \) is a morphism, and \( T \subseteq Y^\oplus \) is another semilinear set. Let \( \varphi: X^\oplus \to Y^\oplus \) be the morphism with \( \varphi(\Psi(w)) = \Psi(h(w)) \) for every \( w \in X^* \). Moreover, consider the set
\[
T' = \{ \mu \in X^\oplus \mid \varphi(w) \in T \} = \{ \Psi(w) \mid w \in X^*, \ \Psi(h(w)) \in T \}.
\]
It is clearly Presburger definable in terms of \( T \) and hence effectively semilinear. Furthermore, we have
\[
h(L \cap \Psi^{-1}(S)) \cap \Psi^{-1}(T) = h(L \cap \Psi^{-1}(S \cap T')).
\]
This proves that \( \text{SLI}(\mathcal{C}) \) is effectively Presburger closed.

Proof of Proposition 5. Proposition 5 follows from Propositions 30 and 31. The uniform algorithm recursively applies the transformations described therein.

### D Proof of Proposition 3

Proposition 32. If \( \mathcal{C} \) is semilinear, then so is \( \text{SLI}(\mathcal{C}) \). Moreover, if \( \mathcal{C} \) is effectively semilinear, then so is \( \text{SLI}(\mathcal{C}) \).

Proof. Since morphisms effectively preserve semilinearity, it suffices to show that \( \Psi(L \cap \Psi^{-1}(S)) \) is effectively semilinear for each \( L \in \mathcal{C} \), \( L \subseteq X^* \), and semilinear \( S \subseteq X^\oplus \). This, however, is easy to see since \( \Psi(L \cap \Psi^{-1}(S)) = \Psi(L) \cap S \) and the semilinear subsets of \( X^\oplus \) are closed under intersection (they coincide with the Presburger definable sets). Furthermore, if a semilinear representation of \( \Psi(L) \) can be computed, this is also the case for \( \Psi(L) \cap S \).

Proof of Proposition 3. The semilinearity follows from Proposition 32 and a result by van Leeuwen [13], stating that if \( \mathcal{C} \) is semilinear, then so is \( \text{Alg}(\mathcal{C}) \).

The computation of (semilinear representations of) Parikh images can be done recursively. The procedure in Proposition 32 describes the computation for languages in \( F_i \). In order to compute the Parikh image of a language in \( G_i = \text{Alg}(F_i) \), consider an \( F_i \)-grammar \( G \). Replacing each right-hand side by a Parikh equivalent regular language yields a \( \text{REG} \)-grammar \( G' \) that is Parikh equivalent to \( G \). Since \( G' \) is effectively context-free, one can compute the Parikh image for \( G' \).
**Simple constructions of PAIM**

This section contains simple lemmas for the construction of PAIM.

- **Lemma 33 (Unions).** Given \( i \in \mathbb{N} \) and languages \( L_0, L_1 \in G_i \), along with a PAIM in \( G_i \) for each of them, one can construct a PAIM for \( L_0 \cup L_1 \) in \( G_i \).

  **Proof.** One can find a PAIM \((K^{(i)}, C^{(i)}, P^{(i)}, (P_c^{(i)})_{c \in C^{(i)}}, \varphi^{(i)}, \bigcirc)\) for \( L_i \) in \( C \) for \( i = 0, 1 \) such that \( C^{(i)} \cap C^{(i)} = P^{(i)} \cap P^{(i)} = \emptyset \). Then \( K = K^{(0)} \cup K^{(1)} \) is effectively contained in \( G_i \) and can be turned into a PAIM \((K, C, P, (P_c)_{c \in C}, \varphi, \bigcirc)\) for \( L_0 \cup L_1 \).

- **Lemma 34 (Homomorphic images).** Let \( h : X^* \to Y^* \) be a morphism. Given \( i \in \mathbb{N} \) and a PAIM for \( L \in G_i \) in \( G_i \), one can construct a PAIM for \( h(L) \) in \( G_i \).

  **Proof.** Let \((K, C, P, (P_c)_{c \in C}, \varphi, \bigcirc)\) be a PAIM for \( L \) and let \( h : X^\oplus \to Y^\oplus \) be the morphism with \( h(x) = \Psi(h(x)) \) for \( x \in X \). Define the new morphism \( \varphi' : (C \cup P)^\oplus \to Y^\oplus \) by \( \varphi'(\mu) = h(\varphi(\mu)) \). Moreover, let \( g : (C \cup X \cup P \cup \{\bigcirc\})^\oplus \to (C \cup Y \cup P \cup \{\bigcirc\})^\oplus \) be the extension of \( h \) that fixes \( C \cup P \cup \{\bigcirc\} \). Then \((g(K), C, P, (P_c)_{c \in C}, \varphi', \bigcirc)\) is clearly a PAIM for \( h(L) \) in \( G_i \).

- **Lemma 35 (Linear decomposition).** Given \( i \in \mathbb{N} \) and \( L \in G_i \) along with a PAIM in \( G_i \), one can construct \( L_1, \ldots, L_n \in G_i \), each together with a linear PAIM in \( G_i \), such that \( L = L_1 \cup \cdots \cup L_n \).

  **Proof.** Let \((K, C, P, (P_c)_{c \in C}, \varphi, \bigcirc)\) be a PAIM for \( L \subseteq X^* \). For each \( c \in C \), let \( K_c = K \cap (cX \cup P \cup \{\bigcirc\})^* \). Then \((K_c, \{c\}, P_c, \varphi_c, \bigcirc)\), where \( \varphi_c \) is the restriction of \( \varphi \) to \( \{c\} \cup P_c \), is a PAIM for \( \pi_X(K_c) \) in \( G_i \). Furthermore, \( L = \bigcup_{c \in C} \pi_X(K_c) \).

- **Lemma 36 (Presence check).** Let \( X \) be an alphabet and \( x \in X \). Given \( i \in \mathbb{N} \) and a PAIM for \( L \subseteq X^* \) in \( G_i \), one can construct a PAIM for \( L \cap X^*xX^* \) in \( G_i \).

  **Proof.** Since
  \[(L_1 \cup \cdots \cup L_n) \cap X^*xX^* = (L_1 \cap X^*xX^*) \cup \cdots \cup (L_n \cap X^*xX^*),\]

Lemma 35 and Lemma 33 imply that we may assume that the PAIM \((K, C, P, (P_c)_{c \in C}, \varphi, \bigcirc)\) for \( L \) is linear, say \( C = \{c\} \) and \( P = P_c \). Since in the case \( \varphi(c)(x) \geq 1 \), we have \( L \cap X^*xX^* = L \) and there is nothing to do, we assume \( \varphi(c)(x) = 0 \).

Let \( C' = \{(c, p) \mid p \in P, \varphi(p)(x) \geq 1\} \) be a new alphabet and let \( K' = \{(c, p)uv \mid (c, p) \in C', u, v \in (X \cup P \cup \{\bigcirc\})^*, cupv \in K\} \).

Note that \( K' \) can clearly be obtained from \( K \) by way of a rational transduction and is therefore contained in \( G_i \). Furthermore, we let \( P' = P'_{(c, p)} = P \) and \( \varphi'((c, p)) = \varphi(c) + \varphi(p) \) for \((c, p) \in C'\) and \( \varphi'(p) = \varphi(p) \) for \( p \in P \). Then we have
\[
\pi_X(K') = \{\pi_X(w) \mid w \in K, \exists p \in P : \varphi(\pi_C \cup P(w))(p) \geq 1, \varphi(p)(x) \geq 1\}
= \{\pi_X(w) \mid w \in K, \vert \pi_X(w) \vert_x \geq 1\} = L \cap X^*xX^*.
\]

This proves the projection property. For each \((c, p)uv \in K'\) with \( cupv \in K \), we have
\[
\varphi'(\pi_C \cup P'((c, p)uv)) = \varphi(\pi_C \cup P(cupv)) = \Psi(\pi_X(cupv)) = \Psi(\pi_X((c, p)uv)).
\]
and thus \( \varphi'(\pi_{C \cup P'}(w)) = \Psi(\pi_{X}(w)) \) for every \( w \in K' \). Hence, we have established the counting property. Moreover,

\[
\Psi(\pi_{C \cup P'}(K')) = \bigcup_{p \in P'} (c, p) + P'^{\oplus},
\]

meaning the commutative projection property is satisfied as well. This proves that the tuple \((\pi_{C \cup X \cup P}(K'), C', P', (P'_d)_{d \in C'})\) is a Parikh annotation for \( L \cap X^*xX^* \) in \( G_i \). Since \((K, C, P, (P_c)_{c \in C}, \varphi, \diamond)\) is a PAIM for \( L \), it follows that \((K', C', P', (P'_d)_{d \in C'}, \varphi', \diamond)\) is a PAIM for \( L \cap X^*xX^* \).

**Lemma 37** (Absence check). Let \( X \) be an alphabet and \( x \in X \). Given \( i \in \mathbb{N} \) and a PAIM for \( L \subseteq X^* \) in \( G_i \), one can construct a PAIM for \( L \setminus X^*xX^* \) in \( G_i \).

**Proof.** Since

\[
(L_1 \cup \cdots \cup L_n) \setminus X^*xX^* = (L_1 \setminus X^*xX^*) \cup \cdots \cup (L_n \setminus X^*xX^*),
\]

Lemma 35 and Lemma 33 imply that we may assume that the PAIM \((K, C, P, (P_c)_{c \in C}, \varphi, \diamond)\) for \( L \) is linear, say \( C = \{c\} \) and \( P = P_c \). Since in the case \( \varphi(c)(x) \geq 1 \), we have \( L \setminus X^*xX^* = \emptyset \) and there is nothing to do, we assume \( \varphi(c)(x) = 0 \).

Let \( C' = C, P' = P'_c = \{p \in P \mid \varphi(p)(x) = 0\} \), and let

\[
K' = \{w \in K \mid |w|_p = 0 \text{ for each } p \in P \setminus P'\}.
\]

Furthermore, we let \( \varphi' \) be the restriction of \( \varphi \) to \((C' \cup P')^{\oplus}\). Then clearly \((K', C', (P'_d)_{d \in C'}, \varphi', \diamond)\) is a PAIM for \( L \setminus X^*xX^* \) in \( G_i \).

**F** Proof of Lemma 11

**Proof.** First, observe that there is at most one \( G \)-compatible extension: For each \( A \in N \), there is a \( u \in T^* \) with \( A \Rightarrow_G u \) and hence \( \psi(u) = \psi(u) \).

In order to prove existence, we claim that for each \( A \in N \) and \( A \Rightarrow_G u \) and \( A \Rightarrow_G v \) for \( u, v \in T^* \), we have \( \psi(u) = \psi(v) \). Indeed, since \( G \) is reduced, there are \( x, y \in T^* \) with \( S \Rightarrow_G xAy \). Then \( xuy \) and \( xvy \) are both in \( L(G) \) and hence \( \psi(xuy) = \psi(xvy) = h \). In the group \( H \), this implies

\[
\psi(u) = \psi(x)^{-1}h\psi(y)^{-1} = \psi(v).
\]

This means a \( G \)-compatible extension exists: Setting \( \hat{\psi}(A) = \psi(w) \) for some \( w \in T^* \) with \( A \Rightarrow_G w \) does not depend on the chosen \( w \). This definition implies that whenever \( u \Rightarrow_G v \) for \( u \in (N \cup T)^* \), \( v \in T^* \), we have \( \psi(u) = \psi(v) \). Therefore, if \( u \Rightarrow_G v \) for \( u, v \in (N \cup T)^* \), picking a \( w \in T^* \) with \( v \Rightarrow_G w \) yields \( \psi(u) = \psi(w) = \psi(v) \). Hence, \( \psi \) is \( G \)-compatible.

Now suppose \( H = Z \) and \( C = F_i \). Since \( Z \) is commutative, \( \psi \) is well-defined on \( T^{\oplus} \), meaning there is a morphism \( \hat{\psi}: T^{\oplus} \rightarrow Z \) with \( \hat{\psi}(\Psi(w)) = \psi(w) \) for \( w \in T^* \). We can therefore determine \( \hat{\psi}(A) \) by computing a semilinear representation of the Parikh image of \( K = \{w \in T^* \mid A \Rightarrow_G w\} \in \text{Alg}(F_i) \) (see Proposition 3), picking an element \( \mu \in \Psi(K) \), and compute \( \hat{\psi}(A) = \psi(\mu) \).
\section*{G \ Proof of Lemma 12}

\textbf{Proof.} Let $G = (X, P, S)$ and let $\delta : X^* \to \mathbb{Z}$ be the morphism with $\delta(w) = \gamma_0(\pi_{X_0}(w)) - \gamma_1(\pi_{X_1}(w))$ for $w \in X^*$. Since then $\delta(w) = 0$ for every $w \in L(G)$, by Lemma 11, $\delta$ extends uniquely to a $G$-compatible $\delta : (N \cup X)^* \to \mathbb{Z}$. We claim that with $k = \max\{\delta(A) \mid A \in N\}$, each derivation tree of $G$ admits a $k$-matching.

Consider an $(N \cup X)$-tree $t$ and let $L_t$ be the set of $X_i$-labeled leaves. Let $A$ be an arrow collection for $t$ and let $d_A(\ell)$ be the number of arrows incident to $\ell \in L_0 \cup L_1$. Moreover, let $\lambda(\ell)$ be the label of the leaf $\ell$ and let

$$\beta(t) = \sum_{\ell \in L_0} \gamma_0(\lambda(\ell)) - \sum_{\ell \in L_1} \gamma_1(\lambda(\ell)).$$

$A$ is a partial $k$-matching if the following holds:

1. if $\beta(t) \geq 0$, then $d_A(\ell) \leq \gamma_0(\lambda(\ell))$ for each $\ell \in L_0$ and $d_A(\ell) = \gamma_1(\lambda(\ell))$ for each $\ell \in L_1$.
2. if $\beta(t) \leq 0$, then $d_A(\ell) \leq \gamma_1(\lambda(\ell))$ for each $\ell \in L_1$ and $d_A(\ell) = \gamma_0(\lambda(\ell))$ for each $\ell \in L_0$.
3. $d_A(s) \leq k$ for every subtree $s$ of $t$.

Hence, while in a $k$-matching the number $\gamma_i(\lambda(\ell))$ is the degree of $\ell$ (with respect to the matching), it is merely a capacity in a partial $k$-matching. The first two conditions express that either all leaves in $L_0$ or all in $L_1$ (or both) are filled up to capacity, depending on which of the two sets of leaves has less (total) capacity.

If $t$ is a derivation tree of $G$, then $\beta(t) = 0$ and hence a partial $k$-matching is already a $k$-matching. Therefore, we show by induction on $n$ that every derivation subtree of height $n$ admits a partial $k$-matching. This is trivial for $n = 0$ and for $n > 0$, consider a derivation subtree $t$ with direct subtrees $s_1, \ldots, s_r$. Let $B$ be the label of $t$’s root and $B_j \in N \cup X$ be the label of $s_j$’s root. Then $\delta(B) = \beta(t)$, $\delta(B_j) = \beta(s_j)$ and $\beta(t) = \sum_{j=1}^r \beta(s_j)$. By induction, each $s_j$ admits a partial $k$-matching $A_j$. Let $A$ be the union of the $A_j$. Observe that since $\sum_{\ell \in L_0} d_A(\ell) = \sum_{\ell \in L_1} d_A(\ell)$ in every arrow collection (each side equals the number of arrows), we have

$$\beta(t) = \sum_{\ell \in L_0} (\gamma_0(\lambda(\ell)) - d_A(\ell)) - \sum_{\ell \in L_1} (\gamma_1(\lambda(\ell)) - d_A(\ell)). \tag{10}$$

If $\beta(t) \geq 0$ and hence $p \geq g$, this equation allows us to obtain $A'$ from $A$ by adding $q$ arrows, such that each $\ell \in L_1$ has $\gamma_1(\lambda(\ell)) - d_A(\ell)$ new incident arrows. They are connected to $X_0$-leaves so as to maintain $\gamma_0(\ell) - d_A(\ell) \geq 0$. Symmetrically, if $\beta(t) \leq 0$ and hence $p \leq q$, we add $p$ arrows such that each $\ell \in L_0$ has $\gamma_0(\lambda(\ell)) - d_A(\ell)$ new incident arrows. They also are connected to $X_1$-leaves so as to maintain $\gamma_1(\lambda(\ell)) - d_A(\ell) \geq 0$. Then by construction, $A'$ satisfies the first two conditions of a partial $k$-matching. Hence, it remains to be shown that the third is fulfilled as well.

Since for each $j$, we have either $d_A(\ell) = \gamma_0(\lambda(\ell))$ for all $\ell \in L_0 \cap s_j$ or we have $d_A(\ell) = \gamma_1(\lambda(\ell))$ for all $\ell \in L_1 \cap s_j$, none of the new arrows can connect two leaves inside of $s_j$. This means the $s_j$ are the only subtrees for which we have to verify the third condition, which amounts to checking that $d_{A'}(s_j) \leq k$ for $1 \leq j \leq r$. As in Eq. (10), we have

$$\beta(s_j) = \sum_{\ell \in L_0 \cap s_j} (\gamma_0(\lambda(\ell)) - d_A(\ell)) - \sum_{\ell \in L_1 \cap s_j} (\gamma_1(\lambda(\ell)) - d_A(\ell)).$$
REFERENCES

Since the arrows added in $A'$ have respected the capacity of each leaf, we have $d_{A'}(s_j) \leq u$ and $d_{A'}(s_j) \leq v$. Moreover, since $A_j$ is a partial $k$-matching, we have $u = 0$ or $v = 0$. In any case, we have $d_{A'}(s_j) \leq |u - v| = |\beta(s_j)| = |\delta(B_j)| \leq k$, proving the third condition. ▶

H Proof of Lemma 14

**Lemma 38.** Given an $F_i$-grammar, one can compute an equivalent reduced $F_i$-grammar.

**Proof.** Since $F_i$ is a Presburger closed semi-trio and has a decidable emptiness problem, we can proceed as follows. First, we compute the set of productive nonterminals. We initialize $N_0 = \emptyset$ and then successively compute

$$N_{i+1} = \{A \in N \mid L \cap (N_i \cup T)^* \neq \emptyset \text{ for some } A \to L \text{ in } P\}.$$

Then at some point, $N_{i+1} = N_i$ and $N_i$ contains precisely the productive nonterminals. Using a similar method, one can compute the set of productive nonterminals. Hence, one can compute the set $N' \subseteq N$ of nonterminals that are reachable and productive. The new grammar is then obtained by replacing each production $A \to L$ with $A \to (L \cap (N' \cup T)^*)$ and removing all productions $A \to L$ where $A \notin N'$.

**Proof of Lemma 14.** In light of Lemma 34, it clearly suffices to prove the statement in the case that there are $a \in Z$ and $b \in Z'$ with $Z' = Z \cup \{b\}$, $b \notin Z$ and $\sigma(x) = \{x\}$ for $x \in Z \setminus \{a\}$ and $\sigma(a) = \{a, b\}$. Let $(K, C, P, (P_c)_{c \in C}, \varphi, \sigma)$ be a PAIM for $L$ in $G_i$. According to Lemma 38, we can assume $K$ to be given by a reduced $F_i$-grammar.

We want to use Proposition 13 to construct a PAIM for $\sigma(L)$. Let $X_0 = Z \cup \{\emptyset\}$, $X_1 = C \cup P$, and $\gamma_i: X_i^* \to N$ for $i = 0, 1$ be the morphisms with

$$\gamma_0(w) = |w|_a, \quad \gamma_1(w) = \varphi(w)(a).$$

Then, by the counting property of PAIM, we have $\gamma_0(w) = \gamma_1(w)$ for each $w \in K$. Let $Y, h$ and $Y_1, h_1, \eta_1$ be defined as in Eq. (1) and Eq. (2). Proposition 13 allows us to construct $K' \in G_i, K' \subseteq Y^*$, with $K' \subseteq h^{-1}(K), \pi_X(K) = \pi_X(h^{-1}(K))$ for $i = 0, 1$, and $\eta_0(\pi_X_w(w)) = \eta_1(\pi_X_x(w))$ for each $w \in K$.

For each $f \in C \cup P$, let $D_f = \{\{f', m\} \mid f' = f\}$. With this, let $C' = \bigcup_{c \in C} D_c, P' = \bigcup_{p \in P} D_p$, and $P'_{c, m} = \bigcup_{p \in P_c} D_p$ for $(c, m) \in C'$. The new morphism $\varphi': (C' \cup P')^* \to Z'^*$ is defined by

$$\varphi'((f, m))(z) = \varphi(f)(z) \quad \text{for } z \in Z \setminus \{a\},$$

$$\varphi'((f, m))(b) = m,$$

$$\varphi'((f, m))(a) = \varphi(f)(a) - m.$$
REFERENCES 23

For \( v \in Y^* \), \( \pi_{Y_h}(v) \) implies \( \pi_Z(g(u)) = \pi_Z(g(v)) \) for \( u, v \in Y^* \). Thus, from \( \pi_{Y_h}(K) = \pi_{Y_h}(h^{-1}(K)) \), we deduce

\[
\pi_Z(K') = \pi_Z(g(\hat{K})) = \pi_Z(g(h^{-1}(K))) \\
= \pi_Z(f^{-1}(K)) = f^{-1}(L) = \sigma(L).
\]

- **Projection property.** Note that by the definition of \( \varphi' \) and \( g \), we have

\[
\varphi'(\pi_{C' \cup P'}(x))(b) = \eta_1(x) = \eta_1(g^{-1}(x))
\]

for every \( x \in C' \cup P' \).

For \( w \in K' \), we have \( f(w) \in K \) and hence \( \varphi(\pi_{C' \cup P'}(f(w))) = \Psi(\pi_Z(f(w))) \). Since for \( z \in Z \setminus \{a\} \), we have \( \varphi'(x)(z) = \varphi(f(x))(z) \) for every \( x \in C' \cup P' \), it follows that

\[
\varphi'(\pi_{C' \cup P'}(w))(z) = \varphi(\pi_{C' \cup P'}(f(w)))(z) \\
= \Psi(\pi_Z(f(w)))(z) = \Psi(\pi_Z'(w))(z).
\]

Moreover, by (11) and since \( g^{-1}(w) \in \hat{K} \), we have

\[
\varphi'(\pi_{C' \cup P'}(w))(b) = \eta_1(g^{-1}(w)) = \eta_0(g^{-1}(w)) = |w|_b \\
= \Psi(\pi_Z'(w))(b).
\]

and \( f(w) \in K \) yields

\[
\varphi'(\pi_{C' \cup P'}(w))(a) + \varphi'(\pi_{C' \cup P'}(w))(b) = \varphi(\pi_{C' \cup P'}(f(w)))(a) \\
= \Psi(\pi_Z(f(w)))(a) \\
= \Psi(\pi_Z'(w))(a) + \Psi(\pi_Z'(w))(b).
\]

Together with (13), this implies \( \varphi'(\pi_{C' \cup P'}(w))(a) = \Psi(\pi_Z'(w))(a) \). Combining this with (12) and (13), we obtain \( \varphi'(\pi_{C' \cup P'}(w)) = \Psi(\pi_Z'(w)) \). This proves the countability property.

- **Commutative projection property.** Observe that

\[
\Psi(\pi_{C' \cup P'}(K')) = \Psi(\pi_{Y_h}(\hat{K})) = \Psi(\pi_{Y_h}(h^{-1}(K))) \\
= \Psi(h^{-1}(\pi_{C' \cup P'}(K))) = \bigcup_{c \in C'} c + P_c^\oplus.
\]

- **Boundedness.** Since \( |w|_b = |h(v)|_b \) for each \( w \in K' \) with \( w = g(v) \), there is a constant bounding \( |w|_b \) for \( w \in K' \).

- **Insertion property.** Let \( cw \in K' \) with \( c \in C' \) and \( \mu \in P_c^\oplus \). Then \( f(\mu) \in P_c^\oplus \) and \( f(cw) \in K \). Write

\[
\pi_{Z \cup \{a\}}(cw) = w_0 \circ w_1 \circ \cdots \circ w_n
\]

with \( w_0, \ldots, w_n \in Z^* \). Then

\[
\pi_{Z \cup \{a\}}(f(cw)) = f(\pi_{Z \cup \{a\}}(cw)) = f(w_0) \circ \cdots \circ f(w_n).
\]

By the insertion property of \( K \) and since \( f(cw) \in K \), there is a \( v \in K \) with

\[
\pi_Z(v) = f(w_0) v_1 f(w_1) \cdots v_n f(w_n),
\]

\( v_1, \ldots, v_n \in Z^* \), and \( \Psi(\pi_Z(v)) = \Psi(\pi_Z(f(cw))) + \varphi(f(\mu)) \). In particular, we have \( \Psi(v_1 \cdots v_n) = \varphi(f(\mu)) \). Note that \( \varphi'(\mu) \in Z_c^\oplus \) is obtained from \( \varphi(f(\mu)) \in Z_c^\oplus \) by
We conclude that the tuple \((K', C', P', (P'_c)_{c \in C'}, \varphi', \circ)\) is a PAIM in \(G_i\) for \(\sigma(L)\).

1 Proof of Lemma 15

Proof. Let \(\sigma : X^* \to \mathcal{P}(Y^*)\). Assuming that for some \(a \in X\), we have \(\sigma(x) = \{x\}\) for all \(x \in X \setminus \{a\}\) means no loss of generality. According to Lemma 34, we may also assume that \(\sigma(a) \subseteq Z^*\) for some alphabet \(Z\) with \(Y = X \cup Z\). If \(\sigma(a) = L_1 \cup \cdots \cup L_n\), then first substituting \(a\) by \(\{a_1, \ldots, a_n\}\) and then each \(a_i\) by \(L_i\) has the same effect as applying \(\sigma\). Hence, Lemma 14 allows us to assume further that the PAIM given for \(\sigma(a)\) is linear. Finally, since \(\sigma(L) = (L \setminus X^*aX^*) \cup \sigma(L \cap X^*aX^*)\), Lemmas 33, 36 and 37 imply that we may also assume \(L \subseteq X^*aX^*\).

Let \((K, C, P, (P'_c)_{c \in C}, \varphi, \circ)\) be a PAIM for \(L\) and \((\hat{K}, \hat{c}, \hat{P}, \hat{\varphi}, \hat{\circ})\) be a linear PAIM for \(\sigma(a)\). The idea of the construction is to replace each occurrence of \(a\) in \(K\) by words from \(\hat{K}\) after removing \(\hat{c}\). However, in order to guarantee a finite bound for the number of occurrences of \(\circ\) in the resulting words, we also remove \(\circ\) from all but one inserted words from \(\hat{K}\). The new map \(\varphi'\) is then set up so that if \(f \in C \cup P\) represented \(m\) occurrences of \(a\), then \(\varphi'(f)\) will represent \(m\) times \(\hat{\varphi}(\hat{c})\).

Let \(C' = C\), \(P'_c = P_c \cup \hat{P}\), \(P' = \bigcup_{c \in C'} P'_c\), and \(\varphi' : (C' \cup P')^\circ \to Y^\circ\) be the morphism with

\[
\varphi'(f) = \varphi(f) - \varphi(f)(a) \cdot a + \varphi(f)(a) \cdot \hat{\varphi}(\hat{c}) \quad \text{for } f \in C \cup P,
\]

\[
\varphi'(f) = \hat{\varphi}(\hat{f}) \quad \text{for } f \in \hat{P}.
\]

Let \(a_\circ\) be a new symbol and

\[
\hat{K} = \{ua_\circ v \mid uv \in \hat{K}; |u|_a = 0\}.
\]

In other words, \(\hat{K}\) is obtained by replacing in each word from \(K\) the first occurrence of \(a\) with \(a_\circ\). The occurrence of \(a_\circ\) will be the one that is replaced by all of \(\hat{K}\), the occurrences of \(a\) are replaced by \(\pi_{\{\circ\} \cup Z \cup \hat{P}}(\hat{K})\). Let \(\tau\) be the substitution

\[
\tau: (C \cup X \cup P \cup \{\circ, a_\circ\})^* \to \mathcal{P}((C' \cup Z \cup P' \cup \{\circ\})^*)
\]

\[
x \mapsto \{x\}, \text{ for } x \in C \cup X \cup P \cup \{\circ\}, x \neq a,
\]

\[
a_\circ \mapsto \pi_{\{\circ\} \cup Z \cup \hat{P}}(\hat{K}),
\]

\[
a \mapsto \pi_{Z \cup \hat{P}}(\hat{K}).
\]

We claim that with \(K' = \tau(\hat{K})\), the tuple \((K', C', P', (P'_c)_{c \in C'}, \varphi', \circ)\) is a PAIM in \(G_i\) for \(\sigma(L)\). First, since \(G_i\) is closed under rational transductions and substitutions, \(K'\) is in \(G_i\).
Lemma 39

Projection property. Since \( L = \pi_X(K) \) and \( \sigma(a) = \pi_Z(\hat{K}) \), we have \( \sigma(L) = \pi_Z(K') \).

Counting property. Let \( w \in K' \). Then there is a \( u = cu_0au_1 \cdots au_n \in K \), \( u_i \in (C \cup X \cup P \cup \{\varepsilon\})^* \), \( c \in C \), and \( |u_i|_a = 0 \) for \( i = 0, \ldots, n \) and \( w = cu_0w_1u_1 \cdots w_nu_n \) with \( w_i \in \pi_{Z \cup \rho}(\hat{K}) \), \( w_i \in \pi_{Z \cup \rho}(\hat{K}) \) for \( i = 2, \ldots, n \). This means

\[
\Psi(\pi_Z(w_1)) = \hat{\phi}(\hat{c}) + \hat{\phi}(\pi_\rho(w_1)).
\]

Since \( \varphi(\pi_{C \cup \rho}(u))(a) = \Psi(\pi_X(u)) = n \), we have

\[
\varphi'(\pi_{C \cup \rho}(u)) = \varphi(\pi_{C \cup \rho}(u)) - n \cdot a + n \cdot \hat{\phi}(\hat{c})
\]

\[
= \Psi(\pi_X(u)) - n \cdot a + n \cdot \hat{\phi}(\hat{c})
\]

Equations (14) and (15) together imply

\[
\varphi'(\pi_{C \cup \rho}(u)) = \varphi'(\pi_{C \cup \rho}(u)) + \sum_{i=1}^n \varphi'(\pi_{\rho}(w_i))
\]

\[
= \Psi(\pi_X(u)) - n \cdot a + n \cdot \hat{\phi}(\hat{c}) + \sum_{i=1}^n (\Psi(\pi_Z(w_i))) - \hat{\phi}(\hat{c})
\]

\[
= \Psi(\pi_X(u)) - n \cdot a + \sum_{i=1}^n \Psi(\pi_Z(w_i)) = \Psi(\pi_Z(w)).
\]

Commutative projection property. Let \( c \in C' \) and \( \mu \in P^\rho_\circ \) and write \( \mu = \nu + \hat{\nu} \) with \( \nu \in P^\rho_\circ \) and \( \hat{\nu} \in \hat{P}^\rho_\circ \). Then there is a \( cw \in K \) with \( \Psi(\pi_{C \cup P}(cw)) = c + \nu \). Since \( L \subseteq X*aX^* \), we can write \( w = cu_0au_1 \cdots au_n \) with \( |u_i|_a = 0 \) for \( 0 \leq i \leq n \) and \( n \geq 1 \). Moreover, there are \( \hat{c}\hat{w} \in \hat{K} \) and \( \hat{c}\hat{w}' \in \hat{K} \) with \( \Psi(\pi_{C \cup \rho}(\hat{c}\hat{w})) = \hat{c} + \hat{\nu} \) and \( \Psi(\pi_{C \cup \rho}(\hat{c}\hat{w}')) = \hat{\phi}(\hat{c}) \). By definition of \( K' \), the word

\[
w' = cu_0\hat{w}u_1u_2 \cdots u_n
\]

is in \( K' \) and satisfies \( \Psi(\pi_{C \cup \rho}(w')) = c + \nu + \hat{\nu} = c + \mu \). This proves

\[
\bigcup_{c \in C'} \bigcup_{P^\rho_\circ} \subseteq \Psi(\pi_{C \cup \rho}(K')).
\]

The other inclusion is clear by definition. We have thus established that the tuple \((\pi_{C \cup \rho} \circ \rho)(K'), C', P', (P_{\circ}^\rho)_{c \in C'}, \varphi') \) is a Parikh annotation in \( G_i \) for \( \sigma(L) \).

Boundedness. Note that if \( |w|_\circ \leq k \) for all \( w \in K \) and \( |w|_\circ \leq \ell \) for all \( \hat{w} \in \hat{K} \), then \( |w'|_\circ \leq k + \ell \) for all \( w' \in K' \) by construction of \( K' \), implying boundedness.

Insertion property. The insertion property follows from the insertion property of \( K \) and \( \hat{K} \).

\[\Box\]

Proof of Lemma 16

Lemma 39 (Sentential forms). Let \( G = (N,T,P,S) \) be an \( G_i \)-grammar with \( N = \{S\} \), \( P = \{S \rightarrow L\} \), and \( L \subseteq (N \cup T)^*SN(N \cup T)^* \). Furthermore, suppose a PAIM in \( G_i \) is given for \( L \). Then one can construct a PAIM in \( G_i \) for \( SF(G) \).
Therefore, we have
\[ \text{The symbols representing constant and period vectors for } \mathcal{S}(G) \text{ are therefore set up as follows. Let } (K, C, P, (P_c)_{c \in C}, \varphi, \diamond) \text{ be a PAIM for } K \text{ in } G, \text{ and let } P' \text{ and } S_D \text{ be new symbols for each } D \subseteq C. \text{ Moreover, let } C' = \{d_D \mid D \subseteq C\} \text{ and } P' = C' \cup P \text{ with } P'_D = D \cup \bigcup_{c \in D} P_c \text{ for each } D \subseteq C. \text{ We will use the shorthand } X = N \cup T. \text{ Observe that since } L \subseteq X^*X^*, \text{ we have } \varphi(c)(S) \geq 1 \text{ for each } c \in C. \text{ We can therefore define the morphism } \varphi' : (C' \cup P')^\oplus \to X^\oplus \text{ as}
\]

\[ \varphi'(p) = \varphi(p) \quad \text{for } p \in P, \]
\[ \varphi'(c) = \varphi(c) - S \quad \text{for } c \in C, \quad \text{(16)} \]
\[ \varphi'(d_D) = S + \sum_{c \in D} \varphi'(c). \quad \text{(17)} \]

The essential idea in our construction is to use modified versions of \( K \) as right-hand-sides of a grammar. These modified versions are obtained as follows. For each \( D \subseteq C \), we define the rational transduction \( \delta_D \) which maps each word \( w_0w_1 \cdots w_n \in (C \cup X \cup P \cup \{\diamond\})^* \), \( |w_i|_S = 0 \) for \( 0 \leq i \leq n \), to all words \( w_0S_Dw_1 \cdots S_Dw_n \) for which

\[ D_1 \cup \cdots \cup D_n = D, \quad D_i \cap D_j = \emptyset \text{ for } i \neq j. \]

Thus, \( \delta_D \) can be thought of as distributing the elements of \( D \) among the occurrences of \( S \) in the input word. The modified versions of \( K \) are then given by

\[ K_D = \delta_D(\pi_{C \cup X \cup P}(K)), \quad K_D^c = \delta_{D \cup c}(c^{-1} K). \]

In the new annotation, the symbol \( d_D \) represents \( S + \sum_{c \in D} (\varphi(c) - S) \). Since each symbol \( c \in C \) still represents \( \varphi(c) - S \), we cannot insert a whole word from \( K \) for each inserted word from \( L \). This would insert a whole \( c \in C \) in each step and we would count \( \sum_{c \in D}(\varphi(c) - S) \), twice. Hence, in order to compensate for the new constant symbol \( d_D \), when generating a word starting with \( d_D \), we have to prevent exactly one occurrence of \( c \) for each \( c \in D \) from appearing. To this end, we use the nonterminal \( S_D \), which only allows derivation subtrees in which of each \( c \in D \), precisely one occurrence has been left out, i.e. a production \( S_D \to K_D^c \) (for some \( D \subseteq C \)) has been applied. In the productions \( S_D \to K_D \) the symbol from \( C \) on the right hand side is allowed to appear.

In order to have only a bounded number of occurrences of \( \diamond \), one of our modified versions of \( K \) (namely \( K_D^c \)) introduces \( \diamond \) and the other one (\( K_D \)) does not. Since when generating a word starting with \( d_D \), our grammar makes sure that for each \( c \in D \), a production of the form \( S_E \to K_E^c \) is used precisely once (and otherwise \( S_E \to K_E \)), the set \( K_E^c \) is set up to contain \( \diamond \). This will guarantee that during the insertion process simulating \( S \to L \), we insert at most \( |C| \cdot \ell \) occurrences of \( \diamond \), where \( \ell \) is an upper bound for \( |w|_\diamond \) for \( w \in K \).
Let \( N' = \{ S' \} \cup \{ S_D \mid D \subseteq C \} \) and let \( \hat{P} \) consist of the following productions:

\[
\begin{align*}
S' & \to \{ d_D \circ S_D \circ \mid D \subseteq C \} \tag{18} \\
S_q & \to \{ S \} \tag{19} \\
S_D & \to K_D \quad \text{for each } D \subseteq C \tag{20} \\
S_D & \to K_D \quad \text{for each } D \subseteq C \text{ and } c \in D. \tag{21}
\end{align*}
\]

Finally, let \( M \) be the regular language

\[
M = \bigcup_{D \subseteq C} \{ w \in (C' \cup X \cup P' \cup \{ \phi \})^* \mid \pi_{C' \cup P'}(w) \in d_D P_d^* \}.
\]

By intersecting with \( M \), we make sure that the commutative projection property is satisfied.

We prove that when the grammar \( G'_i = (N', C' \cup X \cup P' \cup \{ \phi \}, \hat{P}, \hat{S}) \) and \( K' = L(G') \cap M \), the tuple \((K', C', P'_i, (P'_i)_{c \in C'}, \varphi', \phi)\) is a PAIM for \( SF(G) \) in \( G_i \). By definition, \( L(G') \) is contained in \( Alg(G_i) = G_i \) and hence \( K' \) since \( G_i \) is a full semi-AFL.

Let \( h: (N' \cup C' \cup X \cup P' \cup \{ \phi \})^* \to (C' \cup X \cup P' \cup \{ \phi \})^* \) be the morphism that fixes \( C' \cup X \cup P' \cup \{ \phi \} \) and satisfies \( h(S') = h(S_D) = S \) for \( D \subseteq C \). Moreover, regard \( P'(C) \) as a monoid with \( \cup \) as its operation. Then \( \rho: (N' \cup X)^* \to P'(C) \) is the morphism with \( \rho(S_D) = D \) and \( \rho(S) = \rho(x) = \emptyset \) for \( x \in X \). Furthermore, let \( |w|_{\phi} \leq l \) for all \( w \in K \). We claim that for each \( n \geq 0, d_D \circ S_D \circ \Rightarrow_{G_i} w \) implies

(i) if \( w = u_0S_D u_1 \cdots S_D u_n \) with \( u_i \in X^* \) for \( 0 \leq i \leq n \), then \( D_i \cap D_j = \emptyset \) for \( i \neq j \).
(ii) \( h(\pi_{N' \cup X}(w)) \in SF(G) \).
(iii) \( \varphi'(\pi_{C' \cup P'}(w)) \protect{=} \Psi(h(\pi_{N' \cup X}(w))) + \sum_{c \in \rho(w)} \varphi'(c) \).
(iv) \( |w|_{\phi} \leq 2 + |D \setminus \rho(w)| \cdot \ell \), and
(v) for each \( \mu \in \bigcap_{c \in D \setminus \rho(w)} P_c \), there is a \( w' \in SF(G) \) such that \( h(\pi_{N' \cup X \cup \{ \phi \}}(w)) \leq w' \) and \( \Psi(w') = \Psi(h(\pi_{N' \cup X}(w))) + \varphi'(\mu) \).

We establish this claim using induction on \( n \). Observe that all these conditions are satisfied in the case \( n = 0 \), i.e. \( w = d_D \circ S_D \circ \), conditions (i) to (iv) follow directly by distinguishing among the productions in \( G' \). Therefore, we only prove condition (v) in the induction step.

Suppose \( n > 0 \) and \( d_D \circ S_D \circ \Rightarrow_{G_i}^* \bar{w} \Rightarrow_{G_i} w \). If the production applied in \( \bar{w} \Rightarrow_{G_i} w \) is \( S_D \to \{ S \} \), then \( \rho(w) = \rho(\bar{w}) \) and \( h(\pi_{N' \cup X \cup \{ \phi \}}(\bar{w})) = h(\pi_{N' \cup X \cup \{ \phi \}}(w)) \), so that condition (v) follows immediately from the same condition for \( \bar{w} \). If the applied production is of the form (20) or (21), then we have \( \rho(w) \subseteq \rho(\bar{w}) \) and hence \( \rho(\bar{w}) = \rho(w) \cup E \) for some \( E \subseteq D, |E| \leq 1 \). Then

\[
\bigcup_{c \in D \setminus \rho(w)} P_c = \bigcup_{c \in D \setminus \rho(\bar{w})} P_c \cup \bigcup_{c \in E} P_c.
\]

We can therefore decompose \( \mu \in \left( D \cup \bigcup_{c \in D \setminus \rho(w)} P_c \right) \) into \( \mu = \tilde{\mu} + \nu \) with \( \tilde{\mu} \in \left( D \cup \bigcup_{c \in D \setminus \rho(\bar{w})} P_c \right) \) and \( \nu \in \left( \bigcup_{c \in E} P_c \right) \). By induction, we find a \( \bar{w}' \in SF(G) \) such that \( h(\pi_{N' \cup X \cup \{ \phi \}}(\bar{w})) \leq \bar{w}' \) and \( \Psi(\bar{w}') = \Psi(h(\pi_{N' \cup X}(\bar{w}))) + \varphi'(\tilde{\mu}) \). Let \( \bar{w} = xSy \) be the decomposition facilitating the step \( \bar{w} \Rightarrow_{G_i} w \) and let \( w = xzy \).

- If the production applied in \( \bar{w} \Rightarrow_{G_i} w \) is of the form (20). Then \( \rho(w) = \rho(\bar{w}) \) and hence \( E = \emptyset \) and \( \nu = 0 \). Furthermore, \( z \in K_F \) for some \( F \subseteq C \). We define \( z' = h(\pi_{N' \cup X}(z)) \). Note that then \( z' \in \pi_X(K) = L \) and \( \Psi(z') = \Psi(h(\pi_{N' \cup X}(z))) + \varphi'(\nu) \).
If the production applied in $\bar{w} \Rightarrow G'$, $w$ is of the form (21). Then $z \in K_F^v$ for some $c \in F \subseteq C$ and thus $h(z) \in c^{-1}K$. This implies $\rho(\bar{w}) = \rho(w) \cup \{c\}$, $E = \{\}$, and hence $\nu \in P_E$. The insertion property of $K$ provides a $z' \in L$ such that $\pi_{X \cup \{\} \cup \{\}}(h(z)) \subseteq z'$ and $\Psi(z') = \Psi(\pi_X(h(z))) + \varphi(\nu) = \Psi(\pi_X(h(z))) + \varphi'(\nu)$.

In any case, we have

$z' \in L, \quad h(\pi_{N \cup X \cup \{\} \cup \{\}}(x)) \subseteq z', \quad \Psi(z') = \Psi(h(\pi_{N \cup X}(x))) + \varphi(\nu)$.

Recall that $\bar{w} = xSy$ and $w = xzy$. Since $\bar{w} \subseteq w'$, we can find $x', y'$ with

$w' = x'Sy', \quad h(\pi_{N \cup X}(x)) \subseteq x', \quad h(\pi_{N \cup X}(y)) \subseteq y'$.

Choose $w' = x'z'y'$. Then $SF(G) \ni \bar{w'} \Rightarrow G w'$ and thus $w' \in SF(G)$. Moreover,

$h(\pi_{N \cup X \cup \{\} \cup \{\}}(w')) = h(\pi_{N \cup X \cup \{\} \cup \{\}}(x))h(\pi_{N \cup X \cup \{\} \cup \{\}}(z))h(\pi_{N \cup X \cup \{\} \cup \{\}}(y)) \subseteq z' x' z'y' = w'$.

Finally, $w'$ has the desired Parikh image:

$\Psi(w') = \Psi(\bar{w'}) - S + \Psi(z')$

$= \Psi(h(\pi_{N \cup X}(\bar{w}))) + \varphi'(\bar{\mu}) - S + \Psi(z')$

$= \Psi(h(\pi_{N \cup X}(\bar{w}))) + \varphi'(\bar{\mu}) - S + \Psi(h(\pi_{N \cup X}(z))) + \varphi'(\nu)$

$= \Psi(h(\pi_{N \cup X}(w))) + \varphi'(\bar{\mu}) + \varphi'(\nu)$

$= \Psi(h(\pi_{N \cup X}(w))) + \varphi'(\mu)$.

This completes the induction step for condition (v).

We now use our claim to prove that we have indeed constructed a PAIM.

Projection property. Our claim already entails $\pi_X(K') \subseteq SF(G)$: For $w \in (C' \cup X \cup P' \cup \{\})^*$ with $d_D \circ S_D \Rightarrow_{G'} w$, we have $\pi_X(w) = h(\pi_{N \cup X}(w)) \in SF(G)$ by condition (ii).

In order to prove $SF(G) \subseteq \pi_X(K')$, suppose $w \in SF(G)$ and let $t$ be a partial derivation tree for $G$ with root label $S$ and $\text{yield}(t) = w$. Since $c(x) \in L$ for each inner node $x$ of $t$, we can find a $c_x w_x \in K'$ with $\pi_X(c_x w_x) = c(x)$. Then in particular $c(x) \preceq c_x w_x$, meaning we can obtain a tree $t'$ from $t$ as follows: For each inner node $x$ of $t$, add new leaves directly below $x$ so as to have $c_x w_x$ as the new sequence of child labels of $x$. Note the set of inner nodes of $t'$ is identical to the one of $t$. Moreover, we have $\pi_X(\text{yield}(t')) = w$.

Let $D = \{c_x \mid x \text{ is an inner node in } t'\}$. We pick for each $c \in D$ exactly one inner node $x$ in $t'$ such that $c_x = c$; we denote the resulting set of nodes by $R$. We now obtain $t''$ from $t'$ as follows: For each $x \in R$, we remove its $c_x$-labeled child; for each $x \notin R$, we remove all $\circ$-labeled children. Note that again, the inner nodes of $t''$ are the same as in $t$ and $t'$. Moreover, we still have $\pi_X(\text{yield}(t'')) = w$.

For each inner node $x$ in $t''$, let $D_x = \{c_y \mid y \in R \text{ is below } x \text{ in } t''\}$. Note that in $t, t', t''$, every inner node has the label $S$. We obtain the tree $t'''$ from $t''$ as follows. For each inner node $x$ in $t''$, we replace its label $S$ by $S_{D_x}$. Then we have $\pi_X(\text{yield}(t''')) = w$. Clearly, the root node of $t'''$ is labeled $S_D$. Furthermore, the definition of $K_E$ and $K'_E$ yields that $t'''$ is a partial derivation tree for $G'$. Hence

$S' \Rightarrow_{G'} d_D \circ S_D \Rightarrow_{\ast_{G'}} d_D \circ \text{yield}(t''') \Rightarrow_{G'}.

Since in $t'''$, every leaf has a label in $T' \cup \{S_E\}$, we have $S' \Rightarrow_{2}^{*_{G'}} d_D \circ h(\text{yield}(t''')) \ast_{G'}$. This means $d_D \circ h(\text{yield}(t''')) \ast \in \text{L}(G')$. Furthermore, we clearly have $d_D \circ h(\text{yield}(t''')) \ast \in M$ and since $\pi_X(d_D \circ h(\text{yield}(t''')) \ast) = w$, this implies $w \in \pi_X(K')$.  

REFERENCES
Counting property. Apply condition (iii) in our claim to a word $w \in (C' \cup X \cup P' \cup \{\emptyset\})^*$ with $d_D \circ S_D \Rightarrow_G \gamma_G w$. Since $\rho(w) = \emptyset$ and $h(\pi_{N \cup X}(w)) = \pi_X(w)$, this yields $\varphi'(\pi_{C' \cup P'}(w)) = \Psi(\pi_X(w))$.

Commutative projection property. Since $K' \subseteq M$, we clearly have $\Psi(\pi_{C' \cup P'}(K')) \subseteq \bigcup_{i \in E_C} c + P^{\oplus}_c$. For the other inclusion, let $D \subseteq C$ with $D = \{c_1, \ldots, c_n\}$. Suppose $\mu \in \bigcup_{i \in E_C} c + P^{\oplus}_c$, $\mu = d_D + \nu + \sum_{i=1}^{n} \xi_i$ with $\nu \in D^{\oplus}$ and $\xi_i \in P^{\oplus}_{c_i}$ for $1 \leq i \leq n$. The commutative projection property of $K$ allows us to choose for $1 \leq i \leq n$ words $u_i, v_i \in K$ such that

$$\Psi(\pi_{C' \cup P'}(u_i)) = c_i, \quad \Psi(\pi_{C' \cup P'}(v_i)) = c_i + \xi_i.$$

The words $v'_0, \ldots, v'_n$ are constructed as follows. Let $v'_0 = d_D \circ S_0$ and let $v'_{i+1}$ be obtained from $v'_i$ by replacing the first occurrence of $S$ by $c_{i+1} v_{i+1}$. Furthermore, let $v''_i$ be obtained from $v'_i$ by replacing the first occurrence of $S$ by $S_{\{c_1, \ldots, c_n\}}$ and all other occurrences by $S_0$. Then clearly $d_D \circ S_D \circ = v''_0 \Rightarrow_G' \cdots \Rightarrow_G' v''_n$ and $v''_n \in (T \cup \{S_0\})^*$.

Moreover, we have $\Psi(\pi_{C' \cup P'}(v''_i)) = d_D + \sum_{i=1}^{n} \xi_i$.

Let $g : X^* \rightarrow (T \cup \{S_0\})^*$ be the morphism with $g(S) = S_0$ and that fixes the elements of $T$. For a word $w \in (N' \cup X)^*$ that contains $S_0$ and $1 \leq i \leq n$, let $U_i(w)$ be the word obtained from $w$ by replacing the first occurrence of $S_0$ by $g(u_i)$. Then $w \Rightarrow_G' U_i(w)$ and $\Psi(\pi_{C' \cup P'}(U_i(w))) = \Psi(\pi_{C' \cup P'}(w)) + c_i$. Thus, with

$$u = U_n(v'_{n-1}) \cdots U_1(v'_{n-1}) v''_n,$$

we have $v''_n \Rightarrow_G' u \Rightarrow_G' h(u)$ and hence $h(u) \in L(G')$. By construction, $h(u)$ is in $M$ and thus $h(u) \in K'$. Moreover, we have

$$\Psi(\pi_{C' \cup P'}(h(u))) = \Psi(\pi_{C' \cup P'}(u)) = \Psi(\pi_{C' \cup P'}(v''_n)) + \nu = d_D + \sum_{i=1}^{n} \xi_i + \nu = \mu.$$

This proves $\bigcup_{i \in E_C} c + P^{\oplus}_c \subseteq \Psi(\pi_{C' \cup P'}(K'))$.

Boundedness. Let $w \in (C' \cup X \cup P' \cup \{\emptyset\})^*$ and $d_D \circ S_D \Rightarrow_G \gamma_G w$. By condition (iv) of our claim, we have $|w|_{\emptyset} \leq 2 + |C| \cdot \ell$.

Insertion property. Let $w \in (C' \cup X \cup P' \cup \{\emptyset\})^*$ and $d_D \circ S_D \Rightarrow_G \gamma_G w$. Then $\rho(w) = \emptyset$ and $h(\pi_{N \cup X \cup \{\emptyset\}}(w)) = \pi_X(w)$. Hence condition (v) states that for each $\mu \in P^{\oplus}_{d_D}$, there is a $w' \in SF(G)$ with $\pi_{X \cup \{\emptyset\}}(w) \preceq w'$ and $\Psi(w') = \Psi(\pi_X(w)) + \varphi'(\mu)$.

Proof of Lemma 16. Let $G = (N, T, P, S)$. By Lemma 33, we may assume that there is only one production $S \rightarrow L$ in $P$. By Lemmas 36 and 37, one can construct PAIM for $L_0 = L \setminus (N \cup T)^* S (N \cup T)$ and for $L_1 = L \cap (N \cup T)^* S (N \cup T)^*$. If $G'$ is the grammar $G' = (N, T, P', S)$, where $P' = \{S \rightarrow L_1\}$ and $\sigma : (N \cup T)^* \rightarrow \mathcal{P}((N \cup T)^*)$ is the substitution with $\sigma(S) = L_0$ and $\sigma(t) = \{t\}$ for $t \in T$, then $L(G) = \sigma(SF(G'))$. Hence, one can construct a PAIM for $L(G)$ using Lemmas 15 and 39.

K Proof of Lemma 17

Proof. Our algorithm works recursively with respect to the number of non-terminals. In order to make the recursion work, we need the algorithm to work with right hand sides in
Thus, by Lemmas 33 and 35, we may assume that the PAIM for $N$ in $G_i$ can easily be turned into a PAIM for $L$ in $G_i$. Therefore, this statement implies the lemma.

Let $G = (N, T, P, S)$ be an $G_i$-grammar and $n = |N|$. For each $A \in N \setminus \{S\}$, let

$$G_A = (N \setminus \{S\}, T \cup \{S\}, P_A, A),$$

where $P_A = \{B \rightarrow L \in P \mid B \neq S\}$. Since $G_A$ has $n - 1$ nonterminals, we can construct a PAIM for $L(G_A)$ in $G_i$ for each $A \in N \setminus \{S\}$.

Consider the substitution $\sigma: (N \cup T)^* \rightarrow P((N \cup T)^*)$ with $\sigma(A) = L(G_A)$ for $A \in N \setminus \{S\}$ and $\sigma(x) = \{x\}$ for $x \in T \cup \{S\}$. Let $G' = (\{S\}, T, P', S)$ be the $G_i$-grammar with $P' = \{S \rightarrow \sigma(L) \mid S \rightarrow L \in P\}$. By Lemma 15, we can construct a PAIM in $G_i$ for each right-hand-side of $G'$. Therefore, Lemma 16 provides a PAIM in $G_i$ for $L(G')$. We claim that $L(G') = L(G)$.

The inclusion $L(G') \subseteq L(G)$ is easy to see: Each $w \in L(G_A)$ satisfies $A \Rightarrow_{G_i}^* w$. Hence, for $S \rightarrow L \in P$ and $w \in \sigma(L)$, we have $S \Rightarrow_{G_i}^* w$. This means $SF(G') \subseteq SF(G)$ and thus $L(G') \subseteq L(G)$.

Consider a derivation tree $t$ for $G$. We show by induction on the height of $t$ that $\text{yield}(t) \in L(G')$. We regard $t$ as a partial order. A cut in $t$ is a maximal antichain. We call a cut $C$ in $t$ special if it does not contain the root, every node in $C$ has a label in $T \cup \{S\}$, and if $x \in C$ and $y \leq x$, then $y$ is the root or has a label in $N \setminus \{S\}$.

There is a special cut in $t$: Start with the cut $C$ of all leaves. If there is a node $x \in C$ and a non-root $y \leq x$ with label $S$, then remove all nodes $\geq y$ in $C$ and add $y$ instead. Repeat this process until it terminates. Then $C$ is a special cut.

Let $u$ be the word spelled by the cut $C$. Since all non-root nodes $y < x$ for some $x \in C$ have a label in $N \setminus \{S\}$, $u$ can be derived using a production $S \rightarrow L$ once and then only productions $A \rightarrow M$ with $A \neq S$. This means, however, that $u \in \sigma(L)$ and hence $S \Rightarrow_{G_i}^* u$. The subtrees below the nodes in $C$ all have height strictly smaller than $t$. Moreover, since all inner nodes in $C$ are labeled $S$, these subtrees are derivation trees for $G$. Therefore, by induction we have $u \Rightarrow_{G_i}^* \text{yield}(t)$ and thus $S \Rightarrow_{G_i}^* \text{yield}(t)$. □

### L Proof of Lemma 18

**Proof.** According to Lemma 34, it suffices to show that we can construct a PAIM for $L \cap \Psi^{-1}(S)$. Moreover, if $L = L_1 \cup \cdots \cup L_m$, then

$$L \cap \Psi^{-1}(S) = (L_1 \cap \Psi^{-1}(S)) \cup \cdots \cup (L_m \cap \Psi^{-1}(S)).$$

Thus, by Lemmas 33 and 35, we may assume that the PAIM for $L$ is linear. Let $(K, e, P, \varphi, \circ)$ be a linear PAIM for $L$ in $G_i$.

The set $T = \{\mu \in P^\oplus \mid \varphi(e + \mu) \in S\}$ is semilinear as well, hence $T = \bigcup_{i=1}^n T_i$ for linear $T_i \subseteq P^\oplus$. Write $T_i = \mu_i + F_i^\oplus$ with $\mu_i \in P^\oplus$, and $F_i \subseteq P^\oplus$ being a finite set. Let $P_i^\oplus$ be an alphabet with new symbols in bijection with the set $F_i$ and let $\psi_i: P_i^\oplus \rightarrow P^\oplus$ be the morphism extending this bijection. Moreover, let $U_i$ be the linear set

$$U_i = \mu_i + \{p + \psi_i(p) \mid p \in P_i^\oplus\} \oplus (X \cup \{\circ\})^\oplus$$

and let $R_i = p_1^i \cdots p_m^i$, where $P_i^\oplus = \{p_1, \ldots, p_m\}$. We claim that with new symbols $e_i'$ for
REFERENCES

1 ≤ i ≤ n, C' = \{ c'_i \mid 1 ≤ i ≤ n \}, P' = \bigcup_{\ell=1}^n P'_\ell and
\[ \varphi' (c'_i) = \varphi (c) + \varphi (\mu_i), \]
\[ \varphi' (p) = \varphi (\psi (p)) \quad \text{for } p \in P'_\ell, \]
\[ K' = \bigcup_{i=1}^n c'_i \pi_{C'\cup X\cup P'\cup \{ \emptyset \}} (c^{-1} K R_i \cap \Psi^{-1} (U_i)) , \]
the tuple \((K', C', P', (P'_\ell))_{\ell \in C'} \) is a PAIM for \( L \cap \Psi^{-1} (S) \).

- **Projection property** For \( w \in L \cap \Psi^{-1} (S) \), we find a \( cv \in K \) with \( \pi_X (cv) = w \). Then

\[ \varphi (\pi_X (cv)) = \Psi(w) \in S \text{ and hence } \Psi (\pi_P (v)) \in T. \]

Let \( \pi_P (v) = \mu_i + \nu \) with \( \nu \in F'^{\emptyset} \), \( P'_\ell = \{ p_1, \ldots, p_m \} \), and \( \psi (\kappa) = \nu \). Then the word

\[ v' = \nu P_1^{\kappa (p_1)} \cdots P_m^{\kappa (p_m)} \]

is in \( c^{-1} K R_i \cap \Psi^{-1} (U_i) \) and satisfies \( \pi_X (v') = \pi_X (v) = w \). Moreover, \( v'' = c'_i \pi_{C'\cup X\cup P'\cup \{ \emptyset \}} (v') \in K' \) and hence \( w = \pi_X (v'') \in \pi_X (K') \). This proves \( L \cap \Psi^{-1} (S) \subseteq \pi_X (K') \).

We clearly have \( \pi_X (K') \subseteq \pi_X (K) = L \). Thus, it suffices to show \( \Psi (\pi_X (K')) \subseteq S \).

- **Counting property** Let \( w = c'_i v \in K' \) with \( v = \pi_{C'\cup X\cup P'\cup \{ \emptyset \}} (v') \) for some \( v' \in c^{-1} K R_i \cap \Psi^{-1} (U_i) \). By definition of \( U_i \), this implies

\[ \pi_P (v') = \mu_i + \psi (\pi_P (v')) \]

and hence

\[ \varphi (\pi_P (v')) = \varphi (\mu_i) + \varphi (\psi (\pi_P (v'))) = \varphi (\mu_i) + \varphi (\pi_P (v')). \]

Moreover, if we write \( v' = v'r \) with \( cr'' \in K \) and \( r \in R_i \), then

\[ \varphi' (\pi_{C'\cup P'} (w)) = \varphi' (c'_i) + \varphi' (\pi_P (v')) \]
\[ = \varphi (c) + \varphi (\mu_i) + \varphi (\pi_P (v')) \]
\[ = \varphi (c) + \varphi (\pi_P (v')) = \varphi (\pi_{C'\cup P'} (cr'')) \]
\[ = \Psi (\pi_X (cr'')) = \Psi (\pi_X (w)). \]

This proves the counting property.

- **Commutative projection property.** Let \( \mu = c'_i + P_i^{\emptyset} \), \( \mu = c'_i + \kappa \) with \( \kappa \in P_i^{\emptyset} \). Let \( P'_i = \{ p_1, \ldots, p_m \} \). Then \( \nu = \psi_i (\kappa) \in P^{\emptyset} \) and the commutative projection property of \( K \) yields a \( cr \in K' \) with \( \Psi (\pi_{C'\cup P'} (cr)) = c + \mu_i + \nu \). This means that the word

\[ v' = \nu P_1^{\kappa (p_1)} \cdots P_m^{\kappa (p_m)} \]

is in \( c^{-1} K R_i \cap \Psi (U_i) \). Furthermore, \( \Psi (\pi_P (v')) = \kappa \) and hence

\[ \Psi (\pi_{C'\cup P'} (c'_i, \pi_{C'\cup X\cup P'\cup \{ \emptyset \}} (v'))) = c'_i + \kappa = \mu. \]

This proves \( \bigcup_{i=1}^n c'_i + P_i^{\emptyset} \subseteq \Psi (\pi_{C'\cup P'} (K')) \). The other inclusion follows directly from the definition of \( K' \).
Lemma 40. ▶

for all \( \mathbf{w} \in K \) and \( \mu \in P_i^\oplus \). Write \( w = \pi_{c' \cup \mathbf{X} \cup P_i \cup \mathbf{O}}(v) \) for some \( v \in e^{-1}KR_i \cap \Psi^{-1}(U_i) \), and \( v = v'r \) for some \( r \in R_i \). Then \( cv' \in K \) and applying the insertion property of \( K \) to \( cv' \) and \( \psi_i(\mu) \in P_i^\oplus \) yields a \( v'' \in L \) with \( \pi_{c' \cup \mathbf{X} \cup \mathbf{O}}(cv') \subseteq v'' \) and \( \Psi(v'') = \Psi(\pi_X(cv')) + \varphi(\psi_i(\mu)) \). This word satisfies

\[
\pi_{c' \cup \mathbf{O}}(c'_i w) = \pi_{c' \cup \mathbf{O}}(v) = \pi_{c' \cup \mathbf{O}}(cv') \subseteq v'',
\]

\[
\Psi(\pi_X(v'')) = \Psi(\pi_X(cv')) + \varphi(\psi_i(\mu)) = \Psi(\pi_X(c'_i w)) + \varphi(\mu).
\]

and it remains to be shown that \( v'' \in L \cap \Psi^{-1}(S) \). Since \( v'' \in L \), this amounts to showing \( \Psi(v'') \in S \).

Since \( \Psi(v') \in U_i \), we have \( \Psi(\pi_P(v')) \in \mu_i + F_i^\oplus \) and \( \psi_i(\mu) \in F_i^\oplus \) and hence also \( \Psi(\pi_P(v')) + \psi_i(\mu) \in \mu_i + F_i^\oplus = T_i \). Therefore,

\[
\Psi(v'') = \Psi(\pi_X(cv')) + \varphi(\psi_i(\mu)) = \varphi(\pi_{c' \cup P}(cv')) + \varphi(\psi_i(\mu)) = \varphi(\pi_{c' \cup P}(cv')) + \psi_i(\mu) = \varphi(c + T_i) \subseteq S.
\]

\[ \blacktriangleright \]

**M** Proof of Lemma 19

First, we need a simple auxiliary lemma. For \( \alpha, \beta \in X^\oplus \), we write \( \alpha \preceq \beta \) if \( \alpha(x) \leq \beta(x) \) for all \( x \in X \). For a set \( S \subseteq X^\oplus \), we write \( S_\downarrow = \{ \mu \in X^\oplus \mid \exists \nu \in S : \mu \leq \nu \} \) and \( S_\uparrow = \{ \mu \in X^\oplus \mid \exists \nu \in S : \nu \leq \mu \} \). The set \( S \) is called upward closed if \( S_\uparrow = S \).

▷ **Lemma 40.** For a given semilinear set \( S \subseteq X^\oplus \), the set \( \Psi^{-1}(S_\downarrow) \) is an effectively computable regular language.

**Proof.** The set \( S' = X^\oplus \setminus (S_\downarrow) \) is Presburger-definable in terms of \( S \) and hence effectively semilinear. Moreover, since \( \preceq \) is a well-quasi-ordering on \( X^\oplus \), \( S' \) has a finite set \( F \) of minimal elements. Again \( F \) is Presburger-definable in terms of \( S' \) and hence computable. Since \( S' \) is upward closed, we have \( S' = F^{\uparrow} \). Clearly, given \( \mu \in X^\oplus \), the language \( R_\mu = \{ w \in X^* \mid \mu \leq \Psi(w) \} \) is an effectively computable regular language. Since \( w \in \Psi^{-1}(S_\downarrow) \) if and only if \( w \notin \Psi^{-1}(F^{\uparrow}) \), we have \( X^* \setminus \Psi^{-1}(S_\downarrow) = \bigcup_{\mu \in F} R_\mu \). Thus, we can compute a finite automaton for the complement, \( \Psi^{-1}(S_\downarrow) \).

\[ \blacktriangleright \]

**N** Proof of Theorem 10

▷ **Lemma 41** (Finite languages). Given \( L \in F_0 \), one can construct a PAIM for \( L \) in \( F_0 \).

**Proof.** Let \( L = \{ w_1, \ldots, w_n \} \subseteq X^* \) and define \( C = \{ c_1, \ldots, c_n \} \) and \( P = P_c = \emptyset \), where the \( c_i \) are new symbols. Let \( \varphi : (C \cup P)^\oplus \rightarrow X^\oplus \) be the morphism with \( \varphi(c_i) = \Psi(w_i) \). It is easily verified that with \( K = \{ c_1 w_1, \ldots, c_n w_n \} \), the tuple \((K,C,P,(P_c)_{c \in C},\varphi,o)\) is a PAIM for \( L \) in \( F_0 \).

**Proof of Theorem 10.** We compute the PAIM for \( L \) recursively:

- If \( L \in F_0 \), we can construct a PAIM for \( L \) in \( F_0 \) using Lemma 41.
Proof of Proposition 21. We write \( Y = X \cup \{\#\} \). Suppose \((L\#)^* \in \text{SLI}(\mathcal{C})\). Then \((L\#)^* = h(L' \cap \Psi^{-1}(S))\) for some \(L' \subseteq X^*\), a semilinear \(S \subseteq Z^\mathbb{B}\), and a morphism \(h: X^* \rightarrow Y^*\). Since \(\mathcal{C}\) has PAIMs, there is a PAIM \((K, C, P, (P_c)_{c \in C}, \varphi, \circ)\) for \(L'\) in \(\mathcal{C}\). Let \(S_c = \{\mu \in P_c^\mathbb{B} \mid \varphi(c + \mu) \in S\}\). Moreover, let \(g\) be the morphism with

\[
\begin{align*}
g: (C \cup Z \cup P \cup \{\circ\})^* & \rightarrow (Y \cup \{\circ\})^* \\
z \mapsto h(z) & \text{ for } z \in Z, \\
x \mapsto \varepsilon & \text{ for } x \in C \cup P, \\
\circ \mapsto \circ.
\end{align*}
\]

Finally, we need the rational transduction \(T \subseteq X^* \times (Y \cup \{\circ\})^*\) with

\[
T(M) = \{ s \in X^* \mid r \# s \# t \in M \text{ for some } r, t \in (Y \cup \{\circ\})^* \}.
\]

We claim that

\[
L = T(\hat{L}), \quad \text{where} \quad \hat{L} = \{ g(cw) \mid c \in C, cw \in K, \pi_P(w) \in \Psi^{-1}(S_c) \}.
\]

According to Lemma 40, the language \(\Psi^{-1}(S_c)\) is regular, meaning \(\hat{L} \in \mathcal{C}\) and hence \(T(\hat{L}) \in \mathcal{C}\). Thus, proving \(L = T(\hat{L})\) establishes the proposition.

We begin with the inclusion \(T(\hat{L}) \subseteq L\). Let \(s \in T(\hat{L})\) and hence \(r \# s \# t = g(cw)\) for \(r, t \in (Y \cup \{\circ\})^*, c \in C, cw \in K\) and \(\pi_P(w) \in \Psi^{-1}(S_c)\). The latter means there is a \(\mu \in P_c^\mathbb{B}\) such that \(\Psi(\pi_P(w)) + \mu \in S_c\) and hence

\[
\Psi(\pi_Z(cw)) + \varphi(\mu) = \varphi(c + \Psi(\pi_P(w)) + \mu) \in S.
\]

By the insertion property of \(K\), there is a \(v \in L'\) with \(\pi_{Z \cup \{\circ\}}(cw) \preceq_v v\) and \(\Psi(v) = \Psi(\pi_Z(cw)) + \varphi(\mu)\). This means \(\Psi(v) \in S\) and thus \(v \in L' \cap \Psi^{-1}(S)\) and hence \(g(v) = h(v) \in (L\#)^*\). Since \(g(\circ) = \circ\), the relation \(\pi_{Z \cup \{\circ\}}(cw) \preceq_v g(v) \in (L\#)^*\).

However, \(\circ\) does not occur in \(s\), meaning \(\# s \# \#\) is a factor of \(g(v) \in (L\#)^*\) and hence \(s \in L\). This proves \(T(\hat{L}) \subseteq L\).

In order to show \(L \subseteq T(\hat{L})\), suppose \(s \in L\). The boundedness property of \(K\) means there is a bound \(k \in \mathbb{N}\) with \(|w|_\circ \leq k\) for every \(w \in K\). Consider the word \(v = (s\#)^{k+2}\). Since \(v \in (L\#)^*\), we find a \(v' \in L' \cap \Psi^{-1}(S)\) with \(v = h(v')\). This, in turn, means there is a \(cw \in K\) with \(c \in C\) and \(\pi_Z(cw) = v'\). Then

\[
\varphi(c + \Psi(\pi_P(w))) = \varphi(c + \Psi(\pi_Z(cw))) = \Psi(\pi_Z(cw)) = \Psi(v') \in S
\]
and hence $\Psi_0(\pi_P(w)) \in S_c \subseteq S_{\hat{c}}$. Therefore, $g(cw) \in \hat{L} \subseteq (Y \cup \{\hat{c}\})^*$. Note that $g$ agrees with $h(\pi_Z(\cdot))$ on all symbols but $\hat{c}$ which is fixed by the former and erased by the latter. Since $h(\pi_Z(cw)) = h(\pi_Z(v)) = v = (s\#)^k + 2$, the word $g(cw)$ is obtained from $(s\#)^k + 1$ by inserting occurrences of $\hat{c}$. In fact, it is obtained by inserting at most $k$ of them since $|g(cw)\hat{c}|_1 = |cw\hat{c}|_1 \leq k$. This means $g(cw)$ has at least one factor $s\# \in \#X^*$ and hence $s \in T(g(cw)) \subseteq T(\hat{L})$. This completes the proof of $L = T(\hat{L})$ and thus of the proposition. ⊢

**Q Proof of Proposition 25.** Let $K = L \cup \{a^n b^n c^n \mid n \geq 0\}$. If $K \in \text{Alg}(\mathcal{C})$, then also $M = K \cap a^*(bX)^*c^* \in \text{Alg}(\mathcal{C})$. Hence, let $M = L(G)$ for a reduced $C$-grammar $G = (N, T, P, S)$. This means $T = X \cup \{a, b, c\}$. Let $\alpha, \beta : T^* \to \mathbb{Z}$ be the morphisms with 

$$
\alpha(w) = |w|_a - |w|_b, \quad \beta(w) = |w|_b - |w|_c.
$$

Then $\alpha(w) = \beta(w) = 0$ for each $w \in M \subseteq K$. Thus, Lemma 11 provides $G$-compatible extensions $\hat{\alpha}, \hat{\beta} : (N \cup T)^* \to \mathbb{Z}$ of $\alpha$ and $\beta$, respectively.

Let $k = \max \{|\hat{\alpha}(A)|, |\hat{\beta}(A)| \mid A \in N\}$ + 1 and consider the $C$-grammar $G' = (N, X, P', S)$, where $P' = \{A \to \pi_{\mathcal{N}\cup\mathcal{X}}(L) \mid A \to L \in P\}$. Then clearly $L(G') = \pi_X(M) = L$. We claim that $G'$ is $k$-bursting. By Lemma 24, this implies $L = L(G') \in \mathcal{C}$ and hence the proposition.

Let $t$ be a derivation tree for $G'$ and $x$ a node in $t$ with $|\text{yield}(x)| > k$. Then by definition of $G'$, then there is a derivation tree $\tilde{t}$ for $G$ such that $t$ is obtained from $\tilde{t}$ by deleting or replacing by an $\varepsilon$-leaf each $\{a, b, c\}$-labeled leaf. Since $x$ has to be an inner node, it has a corresponding node $\bar{x}$ in $\tilde{t}$. Since $G$ generates $M$, we have

$$
\text{yield}(\tilde{t}) = a^n b x_1 b x_2 \cdots b x_n c^n
$$

$\text{prove}$. Suppose $G = (N, T, P, S)$ is $k$-bursting. Let $\sigma : (N \cup T)^* \to \mathcal{P}(T^*)$ be the substitution with $\sigma(x) = \{w \in T^{\leq k} \mid x \Rightarrow^*_G w\}$ for $x \in N \cup T$. Since $\sigma(x)$ is finite for each $x \in N \cup T$, there is clearly a locally finite rational transduction $T$ with $T(M) = \sigma(M)$ for every language $M \subseteq (N \cup T)^*$. In particular, $\sigma(M) \in \mathcal{C}$ whenever $M \in \mathcal{C}$. Let $R \subseteq N$ be the set of reachable nonterminals. We claim that

$$
L(G) \cap T^{>k} = \bigcup_{A \in R, A \to L \in P} \sigma(L) \cap T^{>k}.
$$

(22)

This clearly implies $L(G) \cap T^{>k} \in \mathcal{C}$. Furthermore, since $\mathcal{C}$ is a union closed full semi-trio and thus closed under adding finite sets of words, it even implies $L(G) \in \mathcal{C}$ and hence the lemma.

We start with the inclusion “$\subseteq$”. Suppose $w \in L(G) \cap T^{>k}$ and let $t$ be a derivation tree for $G$ with $\text{yield}(t) = w$. Since $|w| > k$, $t$ clearly has at least one node $x$ with $|\text{yield}(x)| > k$. Let $y$ be maximal among these nodes (i.e. such that no descendent of $y$ has a yield of length $> k$). Since $G$ is $k$-bursting, this means $\text{yield}(y) = w$. Furthermore, each child $c$ of $y$ has $|\text{yield}(c)| \leq k$. Thus, if $A$ is the label of $y$, then $A$ is reachable and there is a production $A \to L$ with $w \in \sigma(L)$. Hence, $w$ is contained in the right-hand side of (22).

In order to show “$\supseteq$” of (22), suppose $w \in \sigma(L) \cap T^{>k}$ for some $A \to L \in P$ and a reachable $A \in N$. By the definition of $\sigma$, we have $A \Rightarrow^*_G w$. Since $A$ is reachable, there is a derivation tree $\tilde{t}$ for $G$ with an $A$-labeled node $x$ such that $\text{yield}(x) = w$. Since $G$ is $k$-bursting and $|w| > k$, this implies $w = \text{yield}(x) = \text{yield}(\tilde{t}) \in L(G)$ and thus $w \in L(G) \cap T^{>k}$. ⊢
for some \( n \geq 0 \) and \( x_1, \ldots, x_n \in X \), \( x_1 \cdots x_n \in L \). Moreover, \( \text{yield}(\bar{x}) \) is a factor of \( \text{yield}(\bar{t}) \) and \( \pi_X(\text{yield}(\bar{x})) = \text{yield}(x) \). This means \(|\pi_X(\text{yield}(\bar{x}))| > k\) and since in \( \text{yield}(\bar{t}) \), between any two consecutive \( X \)-symbols, there is a \( b \), this implies \(|\text{yield}(\bar{x})|_b > k - 1\). Let \( A \) be the label of \( x \) and \( \bar{x} \). By the choice of \( k \), we have \(|\hat{\alpha}(\text{yield}(\bar{x}))| = |\hat{\alpha}(A)| \leq k - 1\) and \(|\hat{\beta}(\text{yield}(\bar{x}))| = |\hat{\beta}(A)| \leq k - 1\). Hence, \(|\text{yield}(\bar{x})|_b > k - 1\) implies \(|\text{yield}(\bar{x})|_a \geq 1\) and \(|\text{yield}(\bar{x})|_c \geq 1\). However, a factor of \( \text{yield}(\bar{t}) \) that contains an \( a \) and a \( c \) has to comprise all of \( bx_1 \cdots bx_n \). Hence

\[
\text{yield}(x) = \pi_X(\text{yield}(\bar{x})) = x_1 \cdots x_n = \pi_X(\text{yield}(\bar{t})) = \text{yield}(t).
\]

This proves that \( G' \) is \( k \)-bursting.

\section{Proof of Theorem 26}

\textbf{Proof of Theorem 26.} First, note that if \( V_i \in G_i \setminus F_i \), then \( U_{i+1} \in F_{i+1} \setminus G_i \): By construction of \( U_{i+1} \), the fact that \( V_i \in G_i \) implies \( U_{i+1} \in \text{SLI}(G_i) = F_{i+1} \). By Proposition 5, \( F_i \) is a union closed full semi-trio. Thus, if we had \( U_{i+1} \in G_i = \text{Alg}(F_i) \), then Proposition 25 would imply \( V_i \in F_i \), which is not the case.

Second, observe that \( U_{i+1} \in F_{i+1} \setminus G_i \) implies \( U_{i+1} \in G_{i+1} \setminus F_{i+1} \): By construction of \( V_{i+1} \), the fact that \( U_{i+1} \in F_{i+1} \) implies \( V_{i+1} \in \text{Alg}(F_{i+1}) = G_{i+1} \). By Proposition 5, \( G_i \) is a full semi-AFL and by Theorem 10, every language in \( G_i \) has a PAIM in \( G_i \). Hence, if we had \( V_{i+1} \in F_{i+1} = \text{SLI}(G_i) \), then Proposition 21 would imply \( U_{i+1} \in G_i \), which is not the case.

Hence, it remains to be shown that \( V_0 \in G_0 \setminus F_0 \). That, however, is clear because \( V_0 = \#_0^* \), which is context-free and infinite.