Adding Monotonic Counters to Automata and Transition Graphs

Wong Karianto

Lehrstuhl für Informatik VII, RWTH Aachen, Germany
karianto@i7.informatik.rwth-aachen.de

Abstract. We analyze models of infinite-state automata extended by monotonic counting mechanisms, starting from the (finite-state) Parikh automata studied by Klaedtke and Rueß. We show that, for linear-bounded automata, this extension does not increase the language recognition power. In the framework of infinite transition systems developed by Caucal and others, we show that adding monotonic counters to synchronized rational graphs still results in synchronized rational graphs, in contrast to the case of pushdown graphs or prefix-recognizable graphs. For prefix-recognizable graphs, however, we show that the extension by monotonic counters retains the decidability of the reachability problem.

1 Introduction

The idea of counting devices is a classical one in automata theory. Counters, which are a special case of pushdown stacks – namely such with only one stack symbol – represent the basic model of such devices. Since 2-counter machines are as powerful as Turing machines [13], regarding algorithmic applications, restrictions on the counters under consideration are necessary in order to keep decidability results. Examples of such restrictions are the so-called blind counters [6], which can be found, for example, in Petri nets. Another kind of restrictions lies in the so-called reversal-bounded counters [7]; for these counters, the number of alternations between increments and decrements in any computation is bounded. A special case of reversal-bounded counters is the class of monotonic counters, which is the subject of the present paper.

Current applications of these models can be found in the field of algorithmic verification and also in database theory; counters allow to capture aspects of infinite-state system modeling. Examples of such infinite-state systems are discrete-timed automata [4], reversal-bounded counter machines [8], and some classes of semi-linear systems [1]. In addition, counters provide the possibility to capture some fragments of arithmetic in a computation. For instance, automata on unranked trees (in particular automata on XML documents) can be equipped with arithmetical conditions referring to the sequences of sibling nodes (of unbounded length). Such tree automata have been suggested by Dal Zilio and Lugiez [3] and Seidl et al. [15, 16].

Another example of the application of arithmetic to automata theory, which also serves as our starting point in this paper, are found in the model of Parikh

automata of Klaedtke and Rueß [10, 11]. In this context, a finite automaton is equipped with some monotonic counters, and at the end of a computation the values of these counters are checked against a constraint of Presburger arithmetic, which is actually given as a semi-linear set.

The purpose of this paper is to consider automata with two infinitary components: an infinite transition graph and a counting mechanism. More precisely, we extend the model of (finite-state) Parikh automata mentioned above using more general automaton models instead of finite automata (for example, pushdown automata and linear-bounded automata[1]). We show results on the expressiveness of the resulting automaton models and on decision problems, in particular with respect to the reachability problem.

Regarding the first aspect, recall that adding counting mechanism to pushdown automata in the form of considering 'Parikh pushdown automata,' as observed in [10], enhances expressive power with respect to language recognition. As first result, we show that for linear-bounded automata this extension does not alter the language recognition power.

Our second result concerning 'expressiveness' is concerned with the classification of infinite transition graphs suggested by Caucal and others (see the survey [18]). In such a graph, the infinite set of vertices is captured by a regular language over an auxiliary alphabet Γ whereas the transition relations are represented by automaton-definable relations over Γ^* (that is, subsets of $\Gamma^* \times \Gamma^*$). Some important classes of such graphs are the class of pushdown graphs, of prefix-recognizable graphs, of synchronized rational graphs, and of rational graphs. We introduce monotonic counters in this context, by attaching vectors of natural numbers to vertices, and observe that, for prefix-recognizable graphs, the resulting graphs, in general, are not prefix-recognizable anymore. As second result, we show that adding monotonic counters to a synchronized rational graph yields again a synchronized rational graph.

Regarding decidability, we observe that there is a prefix-recognizable graph (even a finite graph) such that the extension of this graph by monotonic counters yields a graph with undecidable monadic second-order theory. Nevertheless, we are able to show that the reachability problem for the extension of any prefix-recognizable graph by monotonic counters remains decidable, which might be of interest in the verification of infinite-state systems.

The outline of this paper is as follows. We fix our notations in Sect. 2 and recall the definition of Parikh automata as introduced in [10]. In Sect. 3 we extend the idea of Parikh automata to the automaton classes in the Chomsky hierarchy and show our first result concerning the language recognition power of the resulting automata. In Sect. 4 we introduce the notion of the monotonic-counter extension of transition graphs, starting from the idea of Parikh automata, and show our second result regarding this extension. Section 5 is addressed to the decidability of the reachability problem mentioned above. We conclude with Sect. 6 by giving some final remarks and further perspectives, in particular regarding the more general case of reversal-bounded counters.

[1] linear-bounded Turing machines

Acknowledgements. I would like to thank Wolfgang Thomas, for supervising my work on this paper, and Philipp Rohde, for proofreading the first draft of this paper. I would like to thank the anonymous referees for some useful suggestions.

2 Preliminaries

We denote the set of natural numbers by $\mathbb{N}$ and the set of vectors of natural numbers of dimension $n \geq 1$ by $\mathbb{N}^n$. For a vector $\bar{x} \in \mathbb{N}^n$, we denote its i-th component by $(\bar{x})_i$. For $A \subseteq \mathbb{N}^n$, we define the set $(A)_i := \{(\bar{x})_i \mid \bar{x} \in A\}$. The i-th unit vector in $\mathbb{N}^n$, denoted by $\bar{e}_i$, is the one with the i-th component equal to 1 and all the other components equal to 0. The zero vector of dimension n is denoted by $\bar{0}$.

Let $\bar{x}$ and $\bar{y}$ be vectors of dimension $n \geq 1$, and let k be a natural number. We define the vector addition $\bar{x}+\bar{y}$ and the scalar multiplication $k\bar{x}$ componentwise. For $A, B \subseteq \mathbb{N}^n$, we define $A + B := \{\bar{x} + \bar{y} \mid \bar{x} \in A \text{ and } \bar{y} \in B\}$.

Recall that a set $A \subseteq \mathbb{N}^n$, $n \geq 1$, is said to be *linear* if there are vectors $\bar{x}_0 \in \mathbb{N}^n$ (the *constant vector*) and $\bar{x}_1, \ldots, \bar{x}_m \in \mathbb{N}^n$ (the *periods*), for some $m \geq 0$, such that

$$A = \{\bar{x}_0 + k_1\bar{x}_1 + \ldots + k_m\bar{x}_m \mid k_1, \ldots, k_m \in \mathbb{N}\} \ .$$

The set $A \subseteq \mathbb{N}^n$ is said to be *semi-linear* if it is a finite union of linear sets.

Let $\Sigma = \{a_1, \ldots, a_n\}$, $n \geq 1$, be an alphabet. The *Parikh mapping* $\Phi\colon \Sigma^* \to \mathbb{N}^n$ is defined by $\Phi(w) := (|w|_{a_1}, \ldots, |w|_{a_n})$, for each $w \in \Sigma^*$, where $|w|_a$ denotes the number of occurrences of $a \in \Sigma$ in w. Parikh's theorem [14] asserts that the *Parikh image* (that is, the image under the Parikh mapping) of any context-free language is semi-linear linear and effectively constructible (or *effectively semi-linear*, for short).

The idea underlying a Parikh automaton is the following. We assign a vector of natural numbers, say, of (fixed) dimension $n \geq 1$ to each transition of a finite automaton. Then, at the end of a computation (run) the sum of the vectors associated with the transitions occurring in the computation is checked against a constraint given as a semi-linear set. In this view, the underlying finite automaton is equipped with n-many monotonic counters, which may be incremented each time an input symbol is read, and with the possibility to check the final values of these counters against a semi-linear constraint.

Formally, a *Parikh automaton* of dimension $n \geq 1$ over Σ is a system $(\mathfrak{A}, C)$ where $\mathfrak{A}$ is a finite automaton over an extended alphabet of the form $\Sigma \times D$, for some finite, nonempty set $D \subseteq \mathbb{N}^n$ (the *auxiliary set*), and $C \subseteq \mathbb{N}^n$ (the *constraint set*) is a semi-linear set. We define the *Σ-projection* $\Psi\colon (\Sigma \times D)^* \to \Sigma^*$, mapping $(u_1, \bar{d}_1)\cdots(u_m, \bar{d}_m)$ to $u_1 \cdots u_m \in \Sigma^*$, and the *extended Parikh mapping* $\widetilde{\Phi}\colon (\Sigma \times D)^* \to \mathbb{N}^n$, mapping $(u_1, \bar{d}_1)\cdots(u_m, \bar{d}_m)$ to $\bar{d}_1 + \cdots + \bar{d}_m \in \mathbb{N}^n$. Now, the acceptance of a word $u_1 \cdots u_m \in \Sigma^*$ by the Parikh automaton requires two conditions. First, there must be some vectors $\bar{d}_1, \ldots, \bar{d}_m \in D$ such that the word $(u_1, \bar{d}_1)\cdots(u_m, \bar{d}_m)$ is accepted by $\mathfrak{A}$. Second, the sum of these vectors

must belong to C. To sum up, we define the language recognized by $(\mathfrak{A}, C)$ as

$$L(\mathfrak{A}, C) := \{\Psi(w) \mid w \in L(\mathfrak{A}) \text{ and } \widetilde{\Phi}(w) \in C\} \ .$$

The following proposition is an extension of Parikh's theorem.

Proposition 1 (Klaedtke and Rueß [10]). *Let Σ be an alphabet and let D be a finite, nonempty subset of $\mathbb{N}^n$, $n \geq 1$. If the Parikh image of a language $L \subseteq (\Sigma \times D)^*$ is (effectively) semi-linear, then so is its extended Parikh image.*

3 Parikh Automata and the Chomsky Hierarchy

The definition of a Parikh automaton in the preceding section allows us to use any automaton model for the underlying automaton instead of a finite automaton. Using the well-known automaton models of the Chomsky hierarchy, we obtain *Parikh pushdown automata* (*Parikh-PDA*), *Parikh linear-bounded automata* (*Parikh-LBA*), and *Parikh Turing machines* (*Parikh-TM*). To avoid confusion, we also refer to the Parikh automata of Klaedtke and Rueß (with finite automata as the underlying automata) as *Parikh finite automata* (*Parikh-FA*). Obviously, a finite automaton is equivalent to a Parikh-FA; the corresponding Parikh-FA just does not make use of its counters. Similar fact holds for PDA's, LBA's, and TM's as well.

It is not difficult to construct a Parikh-FA that recognizes the language $\{a^k b^k c^k \mid k \geq 0\}$, which is not context-free. Consequently, Parikh-FA's and Parikh-PDA's are more powerful than finite automata and pushdown automata, respectively. In contrast, we show that any Parikh-LBA can be simulated by an LBA. Furthermore, we show that the class of Parikh-PDA-recognizable languages is properly contained in the class of context-sensitive languages.

Theorem 2. *Each Parikh-LBA is equivalent to an LBA.*

Proof. Let $(\mathfrak{M}, C)$ be a Parikh-LBA of dimension $n \geq 1$ over Σ with the auxiliary set $D \subseteq \mathbb{N}^n$. We construct a TM $\mathfrak{M}'$ over Σ that recognizes $L(\mathfrak{M}, C)$ and show that the amount of space used by $\mathfrak{M}'$ in any computation is linear in the length of its input, which implies that $\mathfrak{M}'$ is indeed an LBA.

The TM $\mathfrak{M}'$ works as follows. Given an input word $u := u_1 \cdots u_m \in \Sigma$, $m \geq 0$, we first guess the vectors $\bar{d}_1, \ldots, \bar{d}_m \in D$ nondeterministically and maintain the sum $\bar{x}$ of these vectors somewhere in the working tape (with a unary representation described below). Then, we simulate the computation of $\mathfrak{M}$ on $(u_1, \bar{d}_1) \cdots (u_m, \bar{d}_m)$. If $\mathfrak{M}$ accepts, then it remains to check whether the vector $\bar{x}$ belongs to C.

Without loss of generality, we assume that C is linear since, otherwise, we can nondeterministically choose one of the finitely many linear sets that constitute C to work with. Let $\bar{x}_0$ be the constant vector and $\bar{x}_1, \ldots, \bar{x}_r$ be the periods of C. First, we subtract $\bar{x}_0$ from $\bar{x}$. Then, one by one, we nondeterministically choose a period $\bar{x}_j$ and subtract it from $\bar{x}$. If $\bar{x}$ belongs to C, then it will eventually become the zero vector, whereupon we go to an accepting state.

We turn to analyzing the amount of space used by $\mathfrak{M}'$ to accept an input word $u \in L(\mathfrak{M}, C)$. Simulating $\mathfrak{M}$ requires only linear space since $\mathfrak{M}$ is an LBA. Thus, the critical point in the construction above lies in how much space is needed to maintain the vector $\bar{x}$ and to check whether $\bar{x}$ belongs to C. For the former task, we will use a unary representation of the vector $\bar{x}$. Then, the latter task does not need extra space since we only subtract some vectors from $\bar{x}$ and do not add some to $\bar{x}$. For the unary representation of $\bar{x}$, we use the word

$$\underbrace{|_1 \cdots |_1}_{(\bar{x})_1\text{-times}} \quad \$ \quad \cdots \quad \$ \quad \underbrace{|_n \cdots |_n}_{(\bar{x})_n\text{-times}} \quad ,$$

where $|_1, \ldots, |_n$, and \$ are new symbols. Let t be the greatest natural number occurring in D, that is, $t := \max\{(\bar{d})_i \mid \bar{d} \in D \text{ and } 1 \leq i \leq n\}$. Since each input symbol contributes at most t to each component of $\bar{x}$, it is not difficult to see that the length of the unary representation of $\bar{x}$ can be bounded by $n \cdot t \cdot |u| + n - 1$, which is linear in $|u|$ since n and t are fixed (the term $n-1$ represents the number of \$'s). Hence, we conclude that $\mathfrak{M}'$ is indeed an LBA. □

Note that the idea of the proof of Theorem 2 above applies to the case of Parikh-TM as well.

Corollary 3. *Each Parikh-TM is equivalent to a TM.*

Theorem 2 and Corollary 3 imply that we do not need to introduce new language classes in order to capture Parikh-LBA's and Parikh-TM's. In the following, we classify the Parikh-FA-recognizable and Parikh-PDA-recognizable languages in the Chomsky hierarchy. The results are summarized in Fig. 1, where each solid edge pointing to the right indicates a strict inclusion. For simplicity, we denote the language classes by the corresponding automaton classes.

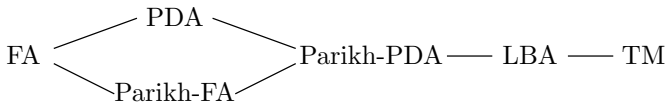

Fig. 1. Hierarchy of language classes

Clearly, any Parikh-FA can be simulated by a Parikh-PDA, which in turn can be simulated by a Parikh-LBA (and thus, by Theorem 2, also by an LBA). These facts justify the inclusion relations shown in Fig. 1. Moreover, these inclusions are strict. As noted before, the non-context-free language $\{a^k b^k c^k \mid k \geq 0\}$ is Parikh-FA-recognizable. On the other hand, one can show that the context-free language $\{ww^{\mathrm{R}} \mid w \in \{a, b\}^*\}$ is not Parikh-FA-recognizable[2]. The strictness

[2] Actually, the proof of this fact is merely a slight modification of the proof that the language $\{ww \mid w \in \{a, b\}^*\}$ is not Parikh-FA-recognizable, as presented in [10]. For further details, the reader is referred to [9, pages 31–33].

of the inclusion between Parikh-PDA's and LBA's follows from Proposition 4 below, which states that the Parikh image of any Parikh-PDA-recognizable language is semi-linear, and from the fact that there is a context-sensitive language whose Parikh image is not semi-linear; for instance, it is straightforward to show that the language $\{a^k b^{k^2} \mid k \geq 0\}$ is context sensitive, but its Parikh image is not semi-linear (see [9, pages 56 and 66–67]).

Proposition 4. *The Parikh image of any Parikh-PDA-recognizable language is effectively semi-linear.*

Proof (sketch). We use the extended Parikh mapping to 'simulate' the Parikh mapping by means of some additional dimensions.

Given a Parikh-PDA $(\mathfrak{A}, C)$ of dimension n over $\Sigma := \{a_1, \ldots, a_m\}$, we extend it to a new Parikh-PDA $(\mathfrak{B}, C')$ of dimension $n + m$ that recognizes the same language as $(\mathfrak{A}, C)$ and counts the number of occurrences of the input symbols by using the additional dimensions. Roughly speaking, the idea is to associate each transition in which a_i is involved with the unit vector $\bar{e}_i$. Then, the vector accumulated at the end of a computation contains the number of occurrences of the input symbols in the last m components. The set of such m-dimensional vectors can be extracted from the extended Parikh image of $L(\mathfrak{B})$ by using the projection functions. The semi-linearity of this set then follows from the semi-linearity of $L(\mathfrak{B})$ (Proposition 1) and the closure of semi-linear sets under the Boolean operations and the projection functions [5]. □

Corollary 5. *The class of Parikh-PDA-recognizable languages is properly contained in the class of context-sensitive languages.*

4 Monotonic-Counter Extensions of Transition Graphs

Let Σ be an alphabet. A *Σ-labeled (transition) graph* is a structure $G := (V, (E_a)_{a \in \Sigma})$, where V is the set of *vertices*, and $E_a \subseteq V \times V$, for each $a \in \Sigma$, is the set of *a-labeled edges* of G. The set of all edges of G is denoted by $E := \bigcup_{a \in \Sigma} E_a$.

Although the graph G under consideration might be infinite, we require that it has a finite representation; the set V of vertices is given by a regular language over an auxiliary alphabet, say Γ, and the set E_a of a-labeled edges, for each $a \in \Sigma$, is given by a binary relation over Γ^*, which is, more or less, definable by an automaton. Some well-known classes of such graphs, with respect to how the edge relations are defined, are *pushdown graphs*, *prefix-recognizable graphs*, *synchronized rational graphs*, and *rational graphs*, where the edge relations are defined by ε-free pushdown transitions, prefix rewriting rules, synchronized rational relations, and rational relations, respectively. Moreover, it is known that these classes constitute a strict increasing chain of inclusions. For an introduction to these graph classes, the reader is referred to [18].

Note that in the sequel we do not distinguish Σ-labeled graphs up to isomorphism; we consider two isomorphic Σ-labeled graphs to be the same. For convenience, we denote the class of finite graphs, pushdown graphs, prefix-recognizable

graphs, synchronized rational graphs, and rational graphs by $\mathcal{G}_{\mathrm{Fin}}$, $\mathcal{G}_{\mathrm{PD}}$, $\mathcal{G}_{\mathrm{PR}}$, $\mathcal{G}_{\mathrm{SR}}$, and $\mathcal{G}_{\mathrm{Rat}}$, respectively.

We apply the idea of Parikh automata as follows. First, we use an extended alphabet of the form $\Sigma \times D$, for some finite, nonempty set $D \subseteq \mathbb{N}^n$, $n \geq 1$, for the edge labeling, thereby obtaining a $(\Sigma \times D)$-labeled graph. If we now take into consideration the vectors that are *accumulated along a path* and code them as monotonic counters (represented by vectors of natural numbers) in the vertices, the vector component of any edge label can be reconstructed from the vectors of the vertices that are incident on this edge. Thus, we may omit the vector component of the edge labels. In this way, we get back to a Σ-labeled graph.

Definition 6. *Let Σ be an alphabet, and let D (called the* auxiliary set*) be a finite, nonempty subset of $\mathbb{N}^n$, $n \geq 1$. Let $G := (V, (E_{(a,\bar{d})})_{(a,\bar{d}) \in \Sigma \times D})$ be a $(\Sigma \times D)$-labeled graph. The Σ-*labeled monotonic-counter extension *(*of dimension *n*) of G (with respect to D) is the Σ-labeled graph $\widetilde{G} := (\widetilde{V}, (\widetilde{E}_a)_{a \in \Sigma})$ with $\widetilde{V} := V \times \mathbb{N}^n$ and*

$$\widetilde{E}_a := \{((\alpha, \bar{x}), (\beta, \bar{y})) \in \widetilde{V} \times \widetilde{V} \mid \exists \bar{d} \in D : (\alpha, \beta) \in E_{(a,\bar{d})} \text{ and } \bar{y} = \bar{x} + \bar{d}\} \ ,$$

for each $a \in \Sigma$. The set of all edges of $\widetilde{G}$ is denoted by $\widetilde{E} := \bigcup_{a \in \Sigma} \widetilde{E}_a$.

For a class $\mathcal{G}(\Sigma)$ of Σ-labeled graphs, the class $\mathcal{G}_{\mathrm{MC}}(\Sigma)$ precisely contains the Σ-labeled graphs in $\mathcal{G}(\Sigma)$ and their (Σ-labeled) monotonic-counter extensions. We will omit Σ whenever it is clear from the context. Occasionally, we refer to a graph that is obtained by a monotonic-counter extension simply as a monotonic-counter graph*.*

Example 7. The infinite two-dimensional grid (see [18, page 134]) can be captured as the following monotonic-counter graph. Let $\Sigma := \{a, b\}$ and $D := \{(1, 0), (0, 1)\}$. We define the $(\Sigma \times D)$-labeled finite graph G_{grid} depicted in Fig. 2 (left). Then, the monotonic-counter extension of G_{grid} is the graph $\widetilde{G}_{\mathrm{grid}} = (\widetilde{V}, \widetilde{E}_a, \widetilde{E}_b)$ with $\widetilde{V} := \{(\bullet, (i, j)) \mid i, j \in \mathbb{N}\}$, $\widetilde{E}_a := \{((\bullet, (i, j)), (\bullet, (i + 1, j))) \mid i, j \in \mathbb{N}\}$, and $\widetilde{E}_b := \{((\bullet, (i, j)), (\bullet, (i, j + 1))) \mid i, j \in \mathbb{N}\}$. The graph $\widetilde{G}_{\mathrm{grid}}$ is illustrated in Fig. 2 (right), where, for better readability, the symbol $\bullet$ has been omitted.

Starting from the class $\mathcal{G}_{\mathrm{Fin}}$ of finite graphs, we obtain the class $\mathcal{G}_{\mathrm{FMC}}$. Analogously, from the graph classes mentioned above we obtain the classes $\mathcal{G}_{\mathrm{PDMC}}$, $\mathcal{G}_{\mathrm{PRMC}}$, $\mathcal{G}_{\mathrm{SRMC}}$, and $\mathcal{G}_{\mathrm{RMC}}$. Moreover, these classes constitute an increasing chain of inclusions as the underlying graph classes also do.

By definition, we have $\mathcal{G} \subseteq \mathcal{G}_{\mathrm{MC}}$, for any class $\mathcal{G}$ of Σ-labeled graphs. For prefix-recognizable graphs, the converse does not hold; the infinite two-dimensional grid of Example 7 is known to have an undecidable monadic second-order theory and thus is not prefix-recognizable [18, Theorem 8 and Theorem 10]. However, the converse holds for synchronized rational graphs and for rational graphs. Towards showing this result, we briefly recall the definitions of synchronized rational and rational graphs. For further references, the reader is referred to [18, pages 133–135].

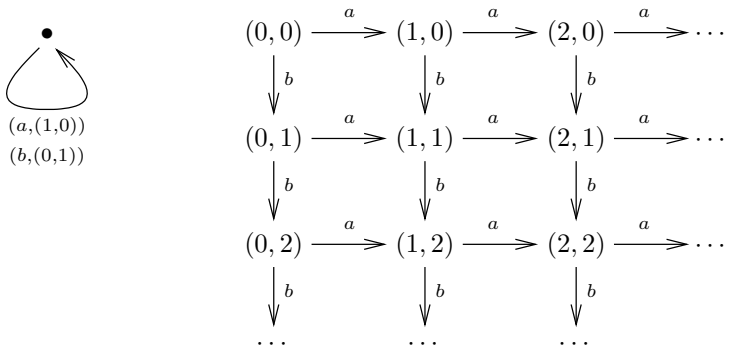

Fig. 2. The graph G_{grid} (left) and $\widetilde{G}_{\text{grid}}$ (right) of Example 7

Let Γ be an alphabet, and let $\diamond$ be a new symbol. For convenience, we write $\Gamma_\diamond$ for $\Gamma \cup \{\diamond\}$. For any words $\alpha = X_1 \cdots X_m$ and $\beta = Y_1 \cdots Y_n$ over Γ, we define $\alpha \hat{} \beta := \left[\begin{smallmatrix} X'_1 \\ Y'_1 \end{smallmatrix}\right] \cdots \left[\begin{smallmatrix} X'_k \\ Y'_k \end{smallmatrix}\right]$, a word of length $k := \max(m, n)$ over $\Gamma_\diamond \times \Gamma_\diamond$, where $X'_i := X_i$, for $i \leq m$, and $X'_i := \diamond$, otherwise; similarly, $Y'_j := Y_j$, for $j \leq n$, and $Y'_j := \diamond$, otherwise. We assign to a relation $R \subseteq \Gamma^* \times \Gamma^*$ the language

$$L_R := \{\alpha \hat{} \beta \mid (\alpha, \beta) \in R\} \subseteq (\Gamma_\diamond \times \Gamma_\diamond)^* \ .$$

The relation R is called *synchronized rational* if L_R is regular. Intuitively, a finite automaton recognizing L_R can be seen as a finite automaton with two one-way input tapes, each with its own input head, which may only be moved simultaneously. The special symbol $\diamond$ is needed to deal with the case where the two words under consideration are of different length.

The automaton characterization of synchronized rational relations carries over to rational relations; a relation $R \subseteq \Gamma^* \times \Gamma^*$ is called *rational* if L_R is recognized by a finite automaton with two one-way input tapes, each with its own input head, which may be moved independently from each other.

Let Σ be an alphabet. A *Σ-labeled* (*synchronized*) *rational graph* is a graph $G = (V, (E_a)_{a \in \Sigma})$ where $V \subseteq \Gamma^*$ is regular, for some alphabet Γ, and E_a is (synchronized) rational, for each $a \in \Sigma$.

Theorem 8. *The monotonic-counter extension of a synchronized rational graph is a synchronized rational graph.*

Proof. Let Σ be an alphabet and D be a finite, nonempty subset of $\mathbb{N}^n$, $n \geq 1$. Let $\widetilde{G} = (\widetilde{V}, (\widetilde{E}_a)_{a \in \Sigma})$ be the monotonic-counter extension of the synchronized rational graph $G = (V, (E_{(a,\bar{d})})_{(a,\bar{d}) \in \Sigma \times D})$, where $V \subseteq \Gamma^*$ is regular, for some alphabet Γ, and $E_{(a,\bar{d})}$ is synchronized rational, for each $(a, \bar{d}) \in \Sigma \times D$; that is, each $L_{E_{(a,\bar{d})}}$ is recognizable by a finite automaton with two one-way input tapes and simultaneously moving input heads, say by $\mathfrak{A}_{(a,\bar{d})}$. In order to show that $\widetilde{G}$

is synchronized rational, we need to find a synchronized rational graph that is isomorphic to $\widetilde{G}$.

First of all, we need to code the vertices of $\widetilde{G}$, each of which consists of a word over Γ and a vector of dimension n. Let $\Pi := \Gamma \cup \{|_1, \ldots, |_n\}$, where $|_1, \ldots, |_n$ are new symbols, which will be used to code vectors of dimension n as follows. Each vector $(x_1, \ldots, x_n) \in \mathbb{N}^n$ is represented by the word $f(x_1, \ldots, x_n) := |_1^{x_1} \cdots |_n^{x_n}$. Using this coding, we now define a coding function for the vertices of $\widetilde{G}$. Let $h\colon V \times \mathbb{N}^n \to (|_1^* \cdots |_n^*)V$ be defined by $h(\alpha, \bar{x}) := f(\bar{x})\alpha$, for each $\alpha \in V$ and $\bar{x} \in \mathbb{N}^n$. Clearly, both f and h are bijective, and thus, the graph $G' := (V', (E'_a)_{a\in\Sigma})$ with

$$V' := (|_1^* \cdots |_n^*)V \ ,$$
$$E'_a := \{(h(\alpha, \bar{x}), h(\beta, \bar{y})) \in V' \times V' \mid ((\alpha, \bar{x}), (\beta, \bar{y})) \in \widetilde{E}_a\}$$

is isomorphic to $\widetilde{G}$. The task is now to show that G' is synchronized rational.

Clearly, the set V' of vertices of G' is regular since V is regular. It remains to show that the edge relation E'_a is synchronized rational, for each $a \in \Sigma$. For each edge relation E'_a, we define a finite automaton $\mathfrak{A}'_a$ with two one-way input tapes and simultaneously moving input heads which works as follows. Let two vertices $f(\bar{x})\alpha$ and $f(\bar{y})\beta$ of G' be given.

- First, we guess a vector $\bar{d} \in D$.
- Then, we check whether $\bar{x}+\bar{d} = \bar{y}$. Note that, for this purpose, the automaton must be able to perform a kind of 'shifted reading' since the input heads may only be moved simultaneously on both input tapes. To illustrate this, consider the following example. Let $\alpha := ab$, $\beta := b$, $\bar{x} := (1, 2)$, and $\bar{y} := (3, 3)$. Then, we have $h(\alpha, \bar{x})\hat{}h(\beta, \bar{y}) = \left[\begin{smallmatrix}|_1\\|_1\end{smallmatrix}\right]\left[\begin{smallmatrix}|_2\\|_1\end{smallmatrix}\right]\left[\begin{smallmatrix}|_2\\|_1\end{smallmatrix}\right]\left[\begin{smallmatrix}a\\|_2\end{smallmatrix}\right]\left[\begin{smallmatrix}b\\|_2\end{smallmatrix}\right]\left[\begin{smallmatrix}\diamond\\|_2\end{smallmatrix}\right]\left[\begin{smallmatrix}\diamond\\a\end{smallmatrix}\right]$. In this example, while we are still working on the first component of the vectors (represented by sequences of $|_1$'s), we must already deal with the second component of the vectors (represented by sequences of $|_2$'s), and similarly, while we are still working with the second component of the vectors, we must already deal with the Γ-components. Since D and Γ are finite (and n is fixed), we need only use a finite memory. In particular, we must remember the *differences* between $\bar{x}$ and $\bar{y}$ (that is, the vector $\bar{d}$) in each component.
- Finally, after verifying that $\bar{x}+\bar{d} = \bar{y}$, we just simulate the automaton $\mathfrak{A}_{(a,\bar{d})}$, checking whether $(\alpha, \beta) \in E_{(a,\bar{d})}$.

It is now straightforward to see that $\mathfrak{A}'_a$ indeed recognizes $L_{E'_a}$. Thus, E'_a is synchronized rational, for each $a \in \Sigma$, and consequently, G' (and thus also $\widetilde{G}$) is indeed a synchronized rational graph. □

The proof idea above can also be applied to the monotonic-counter extensions of rational graphs; the construction of the automata for the edge relations is even easier since the input heads may be moved independently from each other.

Corollary 9. *The monotonic-counter extension of a rational graph is a rational graph.*

Consequently, the classes $\mathcal{G}_{\mathrm{SR}}$ and $\mathcal{G}_{\mathrm{Rat}}$ coincide with the classes $\mathcal{G}_{\mathrm{SRMC}}$ and $\mathcal{G}_{\mathrm{RMC}}$, respectively. The class $\mathcal{G}_{\mathrm{PRMC}}$ is contained in $\mathcal{G}_{\mathrm{SRMC}}$ (and thus also in $\mathcal{G}_{\mathrm{SR}}$) since $\mathcal{G}_{\mathrm{PR}}$ is contained in $\mathcal{G}_{\mathrm{SR}}$. Moreover, this inclusion is strict since $\mathcal{G}_{\mathrm{SR}}$ contains graphs for which the reachability problem is undecidable [18, Theorem 9] while $\mathcal{G}_{\mathrm{PRMC}}$ does not (see Sect. 5 below). Obviously, the class $\mathcal{G}_{\mathrm{PRMC}}$ in turn subsumes the classes $\mathcal{G}_{\mathrm{PR}}$ and $\mathcal{G}_{\mathrm{PDMC}}$. These inclusions are also strict; on the one hand, the infinite two-dimensional grid of Example 7 belongs to $\mathcal{G}_{\mathrm{PDMC}}$, but not to $\mathcal{G}_{\mathrm{PR}}$; on the other hand, since pushdown graphs are of bounded degree and since the auxiliary sets under consideration are always finite, the class $\mathcal{G}_{\mathrm{PDMC}}$ only contains graphs of bounded degree, which does not hold for $\mathcal{G}_{\mathrm{PR}}$ (see [18, pages 132–133]).

These (and some other) hierarchy results are summarized in Fig. 3, where each solid edge pointing to the right indicates a strict inclusion. For simplicity, we write, for instance, FMC instead of $\mathcal{G}_{\mathrm{FMC}}$. The proofs are straightforward and can be found in [9, pages 113–118].

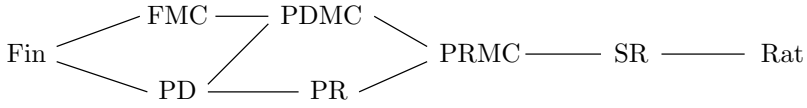

Fig. 3. Hierarchy of graph classes

5 Reachability Problem

A fundamental question in the field of model checking is the reachability problem. For prefix-recognizable graphs, this problem is decidable whereas it is undecidable for synchronized rational graphs. Regarding the former graphs, we show that the monotonic-counter extension preserves the decidability of the reachability problem.

To be precise, we consider the following reachability problem. Let Σ and Γ be alphabets, and let D be a finite, nonempty subset of $\mathbb{N}^n$, $n \geq 1$. Let G be a $(\Sigma \times D)$-labeled prefix-recognizable graph with regular set $V \subseteq \Gamma^*$ of vertices. Then, the reachability problem for $\widetilde{G}$, the monotonic-counter extension of G, is the question: "Given two regular sets $U, U' \subseteq V$ of vertices in G, and given two semi-linear sets $C, C' \subseteq \mathbb{N}^n$, are there vertices $(\alpha, \bar{x}) \in U \times C$ and $(\beta, \bar{y}) \in U' \times C'$ in $\widetilde{G}$ such that $(\beta, \bar{y})$ is reachable from $(\alpha, \bar{x})$?"

Towards our result, we will need the following lemma, stating that the traces[3] of prefix-recognizable graphs, with regular sets of initial and final vertices, are context-free. Actually, this lemma is a corollary of the fact that each prefix-recognizable graph is the ε-closure of the configuration graph of a pushdown automaton [17] (a *constructive* proof can be found in [12]).

[3] For the definition of the traces of transition graphs, the reader might consult [18].

Lemma 10 (Caucal [2]). *The traces of any Σ-labeled prefix-recognizable graph yield a context-free language over Σ, which is effectively constructible.*

In order to show the decidability of the reachability problem for the monotonic-counter extensions of prefix-recognizable graphs we reduce this problem to the emptiness problem for semi-linear sets, which is known to be decidable.

Theorem 11. *The reachability problem for the monotonic-counter extension of any prefix-recognizable graph is decidable.*

Proof. Let Σ be an alphabet and D be a finite, nonempty subset of $\mathbb{N}^n$, $n \geq 1$. Let $\widetilde{G}$ be the monotonic-counter extension of the prefix-recognizable graph $G = (V, (E_{(a,\bar{d})})_{(a,\bar{d})\in\Sigma\times D})$, where $V \subseteq \Gamma^*$ is regular, for some alphabet Γ. Now, let $U, U' \subseteq V$ be regular sets of vertices in G, and let $C, C' \subseteq \mathbb{N}^n$ be semi-linear sets. By Lemma 10, the traces of G, with U as the set of initial vertices and U' as the set of final vertices, yield *effectively* a context-free language L over $\Sigma \times D$. Then, by Parikh's theorem and Proposition 1, the extended Parikh image of L (denoted by $\widetilde{\Phi}(L)$) is effectively semi-linear.

It is straightforward to show that the following statements are equivalent:

1. There are some vertices $(\alpha, \bar{x}) \in U \times C$ and $(\beta, \bar{y}) \in U' \times C'$ in $\widetilde{G}$ such that $(\beta, \bar{y})$ is reachable from $(\alpha, \bar{x})$.
2. $(C + \widetilde{\Phi}(L)) \cap C' \neq \emptyset$.

By the effective closure of semi-linear sets under addition (this property is straightforward to verify) and intersection [5], the set $(C + \widetilde{\Phi}(L)) \cap C'$ is effectively semi-linear, and hence, the emptiness problem for this set is decidable. Consequently, it is decidable whether the first statement above holds, which in turn implies the decidability of the reachability problem for $\widetilde{G}$. □

6 Conclusions

We have drawn the boundary where the idea of Parikh automata, which corresponds to the extension of automata by monotonic counters, properly increases the language recognition power of the automaton classes of the Chomsky hierarchy. While this statement is true for finite automata and pushdown automata, it does not hold for linear-bounded automata and Turing machines. Likewise, we showed that adding monotonic counters to synchronized rational graphs does not go beyond the scope of synchronized rational graphs whereas for prefix-recognizable graphs one obtains a new class of transition graphs. Nevertheless, for the latter we showed that the reachability problem remains decidable.

A natural next step towards extending the present results, which is also a subject of our current work, is to consider the case of reversal-bounded counters [7] instead of monotonic counters. For this case, some technical preparations on the definitions are needed, in particular in order to connect the model of Parikh automata with the model of reversal-bounded counter machines carefully. For

the former model, for instance, a constraint on the values of the counters (via semi-linear sets) occurs only at the end of computations whereas for the latter model intermediate tests are allowed. A first contribution to this issue has been done by Klaedtke and Rueß [10], showing that (finite-state) Parikh automata and reversal-bounded counter machines are equivalent. Another direction is to extend the arithmetical conditions under consideration beyond the scope of semi-linear sets.

References

1. A. Bouajjani, P. Habermehl: Constrained properties, semi-linear systems, and Petri nets. In Proc. CONCUR 1996. LNCS 1119. Springer (1996) 481–497
2. D. Caucal: On infinite transition graphs having a decidable monadic theory. Theoretical Computer Science **290** (2003) 79–115
3. S. Dal Zilio, D. Lugiez: XML schema, tree logic and sheaves automata. In Proc. RTA 2003. LNCS 2706. Springer (2003) 246–263
4. Z. Dang, O.H. Ibarra, T. Bultan, R.A. Kemmerer, J. Su: Binary reachability analysis of discrete pushdown timed automata. In Proc. CAV 2000. LNCS 1855. Springer (2000) 69–84
5. S. Ginsburg, E.H. Spanier: Bounded ALGOL-like languages. Transactions of the American Mathematical Society **113** (1964) 333–368
6. S.A. Greibach: Remarks on blind and partially blind one-way multicounter machines. Theoretical Computer Science **7** (1978) 311–324
7. O.H. Ibarra: Reversal-bounded multicounter machines and their decision problems. Journal of the Association for Computing Machinery **25** (1978) 116–133
8. O.H. Ibarra, T. Bultan, J. Su: Reachability analysis for some models of infinite-state transition systems. In Proc. CONCUR 2000. LNCS 1877. Springer (2000) 183–198
9. W. Karianto: Parikh automata with pushdown stack. Diploma thesis, RWTH Aachen, Germany (2004)
10. F. Klaedtke, H. Rueß: Parikh automata and monadic second-order logics with linear cardinality constraints. Technical Report 177, Institute of Computer Science, Freiburg University, Germany (2002)
11. F. Klaedtke, H. Rueß: Monadic second-order logics with cardinalities. In Proc. ICALP 2003. LNCS 2719. Springer (2003) 681–696
12. M. Leucker: Prefix-recognizable graphs and monadic logic. In Automata, Logics, and Infinite Games. LNCS 2500. Springer (2002) 263–283
13. M.L. Minsky: Recursive unsolvability of Post's problem of 'tag' and other topics in theory of Turing machines. Annals of Mathematics **74** (1961) 437–455
14. R.J. Parikh: On context-free languages. Journal of the Association for Computing Machinery **13** (1966) 570–581
15. H. Seidl, T. Schwentick, A. Muscholl: Numerical document queries. In Proc. PODS 2003. ACM Press (2003) 155–166
16. H. Seidl, T. Schwentick, A. Muscholl, P. Habermehl: Counting in trees for free. In Proc. ICALP 2004. LNCS 3142. Springer (2004) 1136–1149
17. C. Stirling: Decidability of bisimulation equivalence for pushdown processes. Technical Report EDI-INF-RR-0005, School of Informatics, University of Edinburgh, Scotland (2000)
18. W. Thomas: A short introduction to infinite automata. In Proc. DLT 2001. LNCS 2295. Springer (2002) 130–144

Appendix

Proof of Proposition 4

Notations. Let $\bar{x} := (x_1, \ldots, x_n) \in \mathbb{N}^n$ and $\bar{y} := (y_1, \ldots, y_m) \in \mathbb{N}^m$. The *concatenation* of $\bar{x}$ and $\bar{y}$ is the vector defined by $\bar{x} \otimes \bar{y} := (x_1, \ldots, x_n, y_1, \ldots, y_m)$. For $A \subseteq \mathbb{N}^n$ and $B \subseteq \mathbb{N}^m$, we define $A \otimes B := \{\bar{x} \otimes \bar{y} \mid \bar{x} \in A \text{ and } \bar{y} \in B\}$.

For natural numbers $n \geq 2$ and $i = 1, \ldots, n$, we define the projection function $p_i^n \colon \mathbb{N}^n \to \mathbb{N}^{n-1}$ by $p_i^n(x_1, \ldots, x_n) := (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$, for each $(x_1, \ldots, x_n) \in \mathbb{N}^n$. For $A \subseteq \mathbb{N}^n$, we define $p_i^n(A) := \{p_i^n(\bar{x}) \mid \bar{x} \in A\}$.

Proposition. *The Parikh image of any Parikh-PDA-recognizable language is effectively semi-linear.*

Proof. The proof of this result is based on the fact that the Parikh mapping can somehow be 'simulated' by the extended Parikh mapping.

Let $(\mathfrak{A}, C)$ be a Parikh-PDA of dimension $n \geq 1$ over $\Sigma := \{a_1, \ldots, a_m\}$, where $\mathfrak{A} = (Q, \Sigma \times D, \Gamma, \delta, q_0, \bot, F)$.

We extend $(\mathfrak{A}, C)$ to a new Parikh-PDA $(\mathfrak{B}, C')$ of dimension $n + m$ that recognizes the same language as $(\mathfrak{A}, C)$ and counts the number of occurrences of the input symbols by using the additional dimensions. Roughly speaking, the idea is to associate each transition in which a_i is involved with the unit vector $\bar{e}_i$, for $i = 1, \ldots, m$.

Formally, we first define the auxiliary set

$$D' := D \otimes \{\bar{e}_1, \ldots, \bar{e}_m\} \ ,$$

where $\bar{e}_i$ is the i-th unit vector in $\mathbb{N}^m$, for $i = 1, \ldots, m$. We then define the pushdown automaton $\mathfrak{B} := (Q, \Sigma \times D', \Gamma, \delta', q_0, \bot, F)$, where the transition function δ' is given by

$$\delta'(q, \varepsilon, Z) := \delta(q, \varepsilon, Z)$$

and

$$\delta'(q, (a_i, \bar{d} \otimes \bar{e}_j), Z) := \begin{cases} \delta(q, (a_i, \bar{d}), Z) & \text{if } i = j, \\ \emptyset & \text{otherwise,} \end{cases}$$

for each $q \in Q$, $\bar{d} \in D$, $Z \in \Gamma$, and $i, j \in \{1, \ldots, m\}$.

Note that we may not put any constraints on the last m components of the dimensions, since these should only be used to count the number of occurrences of the input symbols. Thus, we define the constraint set to be $C' := C \otimes \mathbb{N}^m$, which is obviously semi-linear.

It should be clear from the construction that $L(\mathfrak{B}, C') = L(\mathfrak{A}, C)$.

To avoid confusion, in the following we will use $\widetilde{\Phi}_{\mathfrak{A}}$ to denote the extended Parikh mapping with respect to $\Sigma \times D$ and $\widetilde{\Phi}_{\mathfrak{B}}$ to denote the extended Parikh mapping with respect to $\Sigma \times D'$.

By construction, a vector in $\widetilde{\Phi}_{\mathfrak{B}}(L(\mathfrak{B}))$, in the first n components, contains a vector in $\widetilde{\Phi}_{\mathfrak{A}}(L(\mathfrak{A}))$ and, in the last m components, contains the number of

occurrences of the input symbols in a word that belongs to $L(\mathfrak{A})$. Still, $\widetilde{\Phi}_{\mathfrak{B}}(L(\mathfrak{B}))$ may contain vectors that are the extended Parikh images of the words whose Σ-projections do not belong to $L(\mathfrak{B}, C')$; we remove these vectors by intersecting $\widetilde{\Phi}_{\mathfrak{B}}(L(\mathfrak{B}))$ with C'. Then, we obtain $\Phi(L(\mathfrak{B}, C'))$, the Parikh image of $L(\mathfrak{B}, C')$ (and thus also of $L(\mathfrak{A}, C)$), by removing the first n components of $\widetilde{\Phi}_{\mathfrak{B}}(L(\mathfrak{B}))\cap C'$. Formally, we have

$$\Phi(L(\mathfrak{B}, C')) = p_1^{1+m}(p_1^{2+m}(\cdots(p_1^{n+m}(\widetilde{\Phi}_{\mathfrak{B}}(L(\mathfrak{B}))\cap C'))\cdots)) \ ,$$

where $p_1^{j+m}\colon \mathbb{N}^{j+m} \to \mathbb{N}^{j+m-1}$, for $j = 1, \ldots, n$, are the projection functions.

By Proposition 1, the set $\widetilde{\Phi}_{\mathfrak{B}}(L(\mathfrak{B}))$ is effectively semi-linear. Hence, by the fact that the class of semi-linear sets is effectively closed under intersection and under the projection functions [5], the set $\Phi(L(\mathfrak{B}, C'))$ is effectively semi-linear as well, whereupon we conclude that the Parikh image of $L(\mathfrak{A}, C)$ is effectively semi-linear. □

Additional Details for Theorem 11

We show that

$$\begin{aligned} &\text{there are some vertices } (\alpha, \bar{x}) \in U \times C \text{ and } (\beta, \bar{y}) \in U' \times C' \\ &\text{in } \widetilde{G} \text{ such that } (\beta, \bar{y}) \text{ is reachable from } (\alpha, \bar{x}) \end{aligned} \tag{1}$$

if and only if

$$(C + \widetilde{\Phi}(L)) \cap C' \neq \emptyset \ . \tag{2}$$

Proof. Suppose that the statement (1) holds. In other words, there is a Σ-labeled path in $\widetilde{G}$ from $(\alpha, \bar{x})$ to $(\beta, \bar{y})$. Formally, there are some edge labels $a_1, \ldots, a_m \in \Sigma$ and some vertices $(\alpha_0, \bar{x}_0), \ldots, (\alpha_m, \bar{x}_m)$ in $\widetilde{G}$, for some $m \geq 0$, such that

- $(\alpha_0, \bar{x}_0) = (\alpha, \bar{x})$,
- $((\alpha_{i-1}, \bar{x}_{i-1}), (\alpha_i, \bar{x}_i)) \in \widetilde{E}_{a_i}$, for $i = 1, \ldots, m$, and
- $(\alpha_m, \bar{x}_m) = (\beta, \bar{y})$.

Then, by the definition of $\widetilde{G}$, there are some vectors $\bar{d}_1, \ldots, \bar{d}_m \in D$ such that

$$(\alpha_{i-1}, \alpha_i) \in E_{(a_i, \bar{d}_i)} \quad \text{and} \quad \bar{x}_{i-1} + \bar{d}_i = \bar{x}_i \ , \tag{3}$$

for $i = 1, \ldots, m$. That is, there is a path in G from α to β, and the edge labels of this path constitute the word

$$w := (a_1, \bar{d}_1) \cdots (a_m, \bar{d}_m) \in (\Sigma \times D)^* \ .$$

Since $\alpha \in U$ and $\beta \in U'$, the word w is a trace in G that belongs to the language L. It follows that $\widetilde{\Phi}(w)$ belongs $\widetilde{\Phi}(L)$, and since $\bar{x}$ belongs to C, $\bar{x} + \widetilde{\Phi}(w)$ belongs to $C + \widetilde{\Phi}(L)$. Further, by the fact that $\bar{x} = \bar{x}_0$ and $\widetilde{\Phi}(w) = \bar{d}_1 + \cdots + \bar{d}_m$ and by (3), we obtain

$$\bar{x} + \widetilde{\Phi}(w) = \underbrace{\bar{x}_0 + \bar{d}_1}_{\bar{x}_1} + \bar{d}_2 + \bar{d}_3 + \cdots + \bar{d}_m = \underbrace{\bar{x}_1 + \bar{d}_2}_{\bar{x}_2} + \bar{d}_3 + \cdots + \bar{d}_m = \cdots = \bar{x}_m \ . \tag{4}$$

Since $\bar{x}_m = \bar{y}$ and $\bar{y}$ belongs to C', it follows that $\bar{x} + \widetilde{\Phi}(w)$ belongs to C'. Hence, we conclude that the set $(C + \widetilde{\Phi}(L)) \cap C'$ is not empty.

Conversely, suppose that $(C + \widetilde{\Phi}(L)) \cap C'$ is not empty. Then, there are some vectors $\bar{x} \in C$ and $\bar{y} \in C'$ and some word $w := (a_1, \bar{d}_1) \cdots (a_m, \bar{d}_m) \in L$ such that $\bar{x} + \widetilde{\Phi}(w) = \bar{y}$. Since $w \in L$, there is a path from some vertex $\alpha \in U$ to some vertex $\beta \in U'$, and the edge labels of this path constitute w. That is, there are some vertices $\alpha_0, \ldots, \alpha_m$ in G such that

- $\alpha_0 = \alpha$,
- $(\alpha_{i-1}, \alpha_i) \in E_{(a_i, \bar{d}_i)}$, for $i = 1, \ldots, m$, and
- $\alpha_m = \beta$.

We will construct a path in $\widetilde{G}$ from $(\alpha, \bar{x})$ to $(\beta, \bar{y})$, thus showing (1). Now, we define the vectors

$$\bar{x}_0 := \bar{x} \quad \text{and} \quad \bar{x}_i := \bar{x}_{i-1} + \bar{d}_i \ ,$$

for $i = 1, \ldots, m$. From this definition, it is not difficult to verify that $\bar{x}_m = \bar{y}$ (by using an analysis similar to (4) above). Further, by the definition of $\widetilde{G}$, we have

$$((\alpha_{i-1}, \bar{x}_{i-1}), (\alpha_i, \bar{x}_i)) \in \widetilde{E}_{a_i} \ ,$$

for each $i = 1, \ldots, m$. Then, the vertices $(\alpha_0, \bar{x}_0), \ldots, (\alpha_m, \bar{x}_m)$ in $\widetilde{G}$ constitute a path from $(\alpha, \bar{x})$ to $(\beta, \bar{y})$. □