Acta Informatica
DOI 10.1007/s00236-006-0027-7

ORIGINAL ARTICLE

# Representation and uniformization of algebraic transductions

**Stavros Konstantinidis · Nicolae Santean · Sheng Yu**

Received: 17 June 2006 / Accepted: 10 August 2006


**Abstract** This paper explores different means of representation for algebraic transductions, i.e., word relations realized by pushdown transducers. The relevance of this work lies more in its point of view rather than any particular result. We are aiming at giving specific techniques for obtaining, or perhaps explaining, decompositions of algebraic (and incidentally, rational) relations, relying solely on their "machine" definition rather than some complex algebraic apparatus. From this point of view, we are hoping to have demystified the heavy formalism employed in the present literature. Some of the novelties of our work are: the use of "stack languages" and "embeddings," which eliminate the need of arbitrary context-free languages in our characterizations, the study of uniformizations for algebraic transductions and the use of the so-called stack transductions for exposing the anatomy of pushdown transducers.

## 1 Introduction

One of the oldest devices designed to perform specific computations in a fairly automatic manner is, arguably, Pascal's Arithmetic Machine (seventeenth century). It is

This work was supported by the Natural Science and Engineering Research Council of Canada grants R220259 and OGP0041630.

S. Konstantinidis
Department of Mathematics and Computing Science, Saint Mary's University,
Halifax B3H 3C3, NS, Canada
e-mail: stavros@cs.smu.ca

N. Santean (✉)
David R. Cheriton School of Computer Science, University of Waterloo,
Waterloo N2L 3G1, ON, Canada
e-mail: nsantean@cs.uwaterloo.ca

S. Yu
Department of Computer Science, University of Western Ontario,
London N6A 5B8, ON, Canada
e-mail: syu@csd.uwo.ca

considered by many to be the first finite-state machine/automation resembling those known and used today ([13]). But it was not before the late 1940s when sequential and other machines become the principal subject of a stand-alone theory. Ever since, the interest in studying the computational power of such devices, and implicitly the nature of those sets computable by them, has steadily gown. Although automata theory has been considered (and still is, by some) a branch of formal language theory – for the reason that automata are ideal tools to specify languages as acceptors – their very nature is best captured when they are placed in an algebraic framework. The study of families of computable subsets of arbitrary monoids was first proposed by Eilenberg in [6–8], where two families of such entities were distinguished: rational and algebraic. Rational sets soon become favorite in the research community, due to their nice algebraic properties and elegant formalism. The same cannot be said about the algebraic family: algebraic transductions are seldom present in the contemporary literature, and their properties are sporadically captured by publications focussed on broader topics, such as [2], [1, Sect. 3.1.4] and [10, Sect. 3.5]. The very few sources that specifically address algebraic transductions include [9] and [4]. Nevertheless, it is in our perception that their existing properties make them powerful tools in practice, and careful design may alleviate some of their general difficulties. In this paper we address the very basics of algebraic transductions, namely their representations, in the spirit of the well-known saying: *"the beginning of wisdom is to call things by their right names"* (Confucius).

The paper is structured as following. In Sect. 2 we introduce Dyck languages, the so-called "stack languages" and embeddings, and we explore some of their properties. We also present variations on the Chomsky–Schützenberger and Nivat theorems, in whose proofs we apply some of the techniques used throughout the paper. In Sect. 3 we give a characterization of, and a matrix representation for, algebraic transductions. In Sect. 4 we tackle the problem of uniformizations for algebraic transductions. Although we did not succeed in finding an unambiguous uniformization for the general case, we have provided solutions for a few particular cases. Finally, in Sect. 5, we define and study the so-called stack transductions and we use them to look under-the-hood of pushdown transducers. Across the paper we use a few simple techniques, such as: the use of disjoint alphabets in order to deal with projections instead of arbitrary morphisms, or subsequently, the use of colorings in order to extend our results to the general case. Another efficient technique that we employed is to "stick together" different components of the labels of transitions in automata/transducers (input, stack and output labels) in order to first deal with finite automata, then put the results in terms of regular languages or rational relations and finally to convert everything back to the initial (algebraic) form. This method helped us produce characterizations for algebraic transductions without the use of heavy algebraic tools – which was our main goal.

## 2 Notions and notations

In this section we introduce Dyck languages, embeddings, and we derive a characterization of context-free languages similar to the well-known Chomsky–Schützenberger theorem (as found in [3, Sect. II, T. 3.10]). We also present a technique of characterizing transductions by means of projections and "colorings," that will further lead to general morphism characterizations.

We assume known basic notions of automata theory. Notation-wise, a finite transducer (FT) is denoted by $(Q, X, Y, \delta, q_0, F)$ and a pushdown transducer (PDT) by $(Q, X, Y, \Gamma, \delta, q_0, Z_0, F)$. $X$, $Y$ and $\Gamma$ are finite alphabets: input, output and stack, respectively. $Q$ is a finite set of states with $q_0$ denoting an initial state. For the pushdown machine, $Z_0 \in \Gamma$ represents a symbol present on the stack at the beginning of its computations. $\delta$ is the transition table (partial function) that governs the computational steps of either machine and takes one of the following forms:

- $\delta : Q \times \big(X \cup \{\varepsilon\}\big) \to \mathcal{PF}\big(Y^* \times Q\big)$, in the case of a FT,
- $\delta : Q \times \big(X \cup \{\varepsilon\}\big) \times \Gamma \to \mathcal{PF}\big(\Gamma^* \times Y^* \times Q\big)$, in the case of a PDT,

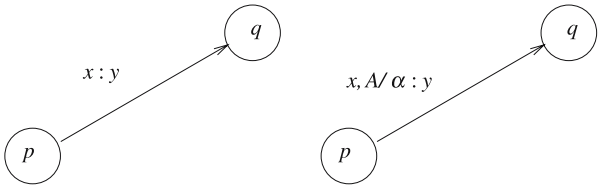
where $\varepsilon$ denotes the empty word and $\mathcal{PF}(A)$ stands for the family of finite subsets of a set $A$. Sometime we choose the notation $(p, x : y, q) \in \delta$ in order to denote a transition in a FT, from the state $p$ to the state $q$, with input $x$ and output $y$ (equivalent with saying that $(y, q) \in \delta(p, x)$). Also, we sometime consider "lazy" FT, that are finite transducers with transitions having words as input. For pushdown transducers, we sometime use the notation $(p, x, A/\alpha : y, q) \in \delta$ to denote a transition from state $p$ to state $q$ on input $x$, output $y$ and with the following stack operations: pop $A$ and push $\alpha$ (equivalent with saying that $(\alpha, y, q) \in \delta(p, x, A)$). The graphical representation of these transitions is depicted in Fig. 1. Pushdown automata (PDA) and nondeterministic finite automata (NFA) are defined similarly, with exception of the output component that is missing.

The transductions (word relations) realized by FT are called rational and their family is denoted by $Rat(X^* \times Y^*)$ and those realized by PDT are called algebraic and denoted by $Alg(X^* \times Y^*)$. As will become apparent later, the three acceptance criteria for PDT ("by final state," "by empty stack" and both) are equivalent (similar to PDA), and we will mostly use the acceptance criterion "by final state."

Let $\Gamma = \{z_1, \ldots, z_n\}$ be a finite alphabet and let $\hat{\Gamma} = \Gamma \cup \overline{\Gamma}$. We denote by $\sim$ the `restricted Dyck congruence` over $\hat{\Gamma}^*$, generated by

$$z_i \overline{z_i} \sim \varepsilon, \quad \forall i \in \{1, \ldots, n\}.$$

In other words, $\sim$ is the smallest congruence that verifies the above relation. The congruence is called "restricted" due to the fact that it does not necessarily imply that $\overline{z}_i z_i \sim \varepsilon, \forall i \in \{1, \ldots, n\}$.




in FT: $(p, x : y, q) \in \delta$ or: $(y, q) \in \delta(p, x)$

in PDT: $(p, x, A/\alpha : y, q) \in \delta$ or: $(\alpha, y, q) \in \delta(p, x, A)$

**Fig. 1** Transition notations for finite transducers (FT) and pushdown transducers (PDT)

The class of a word $w \in \hat{\Gamma}^*$ with respect to $\sim$ is denoted by $[w]$ and the canonical(surjective) homomorphism induced by $\sim$ is given by

$$p : \hat{\Gamma}^* \to \left(\hat{\Gamma}^*/\sim\right), \quad p(w) = [w].$$

The $n$-ary restricted Dyck language (over $\Gamma$) is the language

$$\mathcal{D}_n = (p^{-1} \circ p)(\varepsilon),$$

and consists of all the words in the class of $\varepsilon$.

**Definition 1** *The $n$-ary stack language (over $\Gamma$), is the language*

$$\mathcal{S}_n = (p^{-1} \circ p)(\Gamma^*).$$

In other words, $\mathcal{S}_n$ is the set of those words that are equivalent to some word with no symbol in $\overline{\Gamma}$. The motivation for the terminology "stack language" will become apparent by the end of this section.

Both $\mathcal{D}_n$ and $\mathcal{S}_n$ are context-free languages over $\hat{\Gamma}$ and $\mathcal{D}_n \subseteq \mathcal{S}_n$. Indeed, $\mathcal{D}_n$ is the language of "well-formed" words over $n$ pairs (types) of parentheses and $\mathcal{S}_n$ is the language of words $u_1 v_1 u_2 v_2 \cdots u_k v_k u_{k+1}$ where $u_j \in \Gamma^*$, $\forall j \in \{1, \ldots, k+1\}$ and $v_i \in \mathcal{D}_n$ $\forall i \in \{1, \ldots, k\}$. A context-free grammar for $\mathcal{S}_n$ may have the productions:

$$S \to US \mid VUS \mid V \mid \varepsilon, \quad U \to z_1 \mid \cdots \mid z_n,$$
$$V \to VV \mid z_1 V \overline{z_1} \mid \cdots \mid z_n V \overline{z_n} \mid z_1 \overline{z_1} \mid \cdots \mid z_n \overline{z_n},$$

where $\{S, U, V\}$ is the set of nonterminals with $S$ initial symbol. This grammar shows that $\mathcal{S}_n$ is not inherently ambiguous (the grammar is unambiguous). $\mathcal{D}_n$ and $\mathcal{S}_n$ will be denoted by $\mathcal{D}(\Gamma)$ and $\mathcal{S}(\Gamma)$ when we intend to emphasize the alphabet $\Gamma$ over which they are constructed.

**Definition 2** *Let $X$, $Y$ be two alphabets such that $Y \subseteq X$. The projection of $X^*$ onto $Y^*$ is the surjective homomorphism $\pi_Y : X^* \to Y^*$, given by $\pi_Y(x) = \varepsilon, \forall x \in X \setminus Y$ and $\pi_Y(y) = y, \forall y \in Y$.*

In other words, the projection of a word $w$ in $X^*$ is obtained by "erasing" all the symbols of $w$ that are not in $Y$. The projection is extended in a natural way to work on languages.

**Definition 3** *Let $U, V$ be two disjoint alphabets and $L \subseteq U^*$ be a nonempty language. The embedding of $V$ into $L$ is the largest (inclusionwise) language $\mathcal{E}(V, L) \subseteq (U \cup V)^*$ that satisfies:*

$$\pi_U\left(\mathcal{E}(V, L)\right) = L \cup \{\varepsilon\}.$$

*By convention, $\mathcal{E}(V, \emptyset) = \emptyset$.*

Properties:

1. If $u_1 \cdots u_k \in L$ with $u_1, \ldots, u_k \in U$, then

$$v_1 u_1 v_2 \cdots u_k v_{k+1} \in \mathcal{E}(V, L), \quad \forall v_1, \ldots, v_{k+1} \in V^*.$$

Aside of these words, $\mathcal{E}(V, L)$ contains solely the words of $V^*$.

2. $L \subseteq \mathcal{E}(V, L)$ and $\mathcal{E}(V, U^+) = \mathcal{E}(V, U^*) = (U \cup V)^*$.
3. If $L_1 \subseteq L_2$ then $\mathcal{E}(V, L_1) \subseteq \mathcal{E}(V, L_2)$.
4. $\mathcal{E}(V, L_1 \cup L_2) = \mathcal{E}(V, L_1) \cup \mathcal{E}(V, L_2)$, $\mathcal{E}(V, L_1 \cap L_2) = \mathcal{E}(V, L_1) \cap \mathcal{E}(V, L_2)$ and $\mathcal{E}(V, L_1 L_2) = \mathcal{E}(V, L_1)\mathcal{E}(V, L_2)$. Consequently, the application
$$\mathcal{E} : 2^{U^*} \to 2^{(U \cup V)^*}, \quad \text{given by} \quad L \to \mathcal{E}(V, L)$$
is a monoid morphism with respect to either union or intersection, and is a semigroup morphism with respect to concatenation.
5. If $L \neq \emptyset$ then
$$\mathcal{E}(V, L) = \pi_U^{-1}(L \cup \{\varepsilon\}).$$
Consequently, if $L$ is regular then $\mathcal{E}(V, L)$ is also regular and if $L$ is context-free then $\mathcal{E}(V, L)$ is context-free as well.

In the following we state a characterization of context-free languages similar to the well-known Chomsky–Schützenberger theorem. The result is essentially the same, the differences consisting of the fact that we will use regular languages in place of local regular languages, projections instead of alphabetic morphisms and "embeddings" in place of Dyck languages. The importance of this characterization consists of making the intuitive connection between the languages $\mathcal{D}(\Gamma)$ and $\mathcal{S}(\Gamma)$ on one hand, and the acceptance criteria for PDA: final state and empty stack versus final state, on the other.

**Theorem 1** *Let $X$ be an alphabet and $L \subseteq X^*$ be a nonempty language. The following statements are equivalent:*

(i) *$L$ is context-free.*

(ii) *There exists an alphabet $\Gamma$ with no symbols in common with $X$, and a regular language $R \subseteq (X \cup \hat{\Gamma})^*$ such that*
$$L = \pi_X\Big(R \cap \mathcal{E}\big(X, \mathcal{D}(\Gamma)\big)\Big)$$

(iii) *There exists an alphabet $\Gamma$ with no symbols in common with $X$, and a regular language $R \subseteq (X \cup \hat{\Gamma})^*$ such that*
$$L = \pi_X\Big(R \cap \mathcal{E}\big(X, \mathcal{S}(\Gamma)\big)\Big)$$

*Characterization* (ii) *corresponds to PDA's acceptance by final state and empty stack and characterization* (iii) *corresponds to PDA's acceptance by final state only.*

*Proof* We prove (i)⇔(iii), the equivalence (i)⇔(ii) being proven in a similar way. Considering the diagram in Fig. 2, we first notice that $\pi_{\hat{\Gamma}}^{-1}(\mathcal{S}(\Gamma)) = \mathcal{E}\big(X, \mathcal{S}(\Gamma)\big)$ – since the empty word is contained in $\mathcal{S}(\Gamma)$.

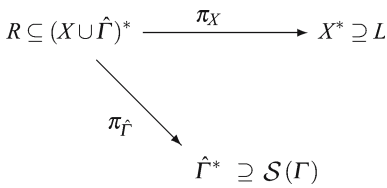



**Fig. 2** A characterization of context-free languages

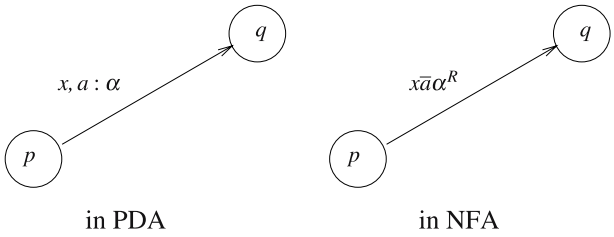



**Fig. 3** Transformation of a pushdown automaton (PDA) into a nondeterministic finite automaton (NFA)

(i) $\Rightarrow$ (iii). Suppose $L$ is a context-free language. Then there exists a PDA $A = (Q, X, \Gamma, \delta, q_0, Z_0, F)$ which accepts $L$ by final state. We first construct a "lazy" NFA (i.e., an NFA with words as transition labels) $A' = (Q \cup \{q_0'\}, X \cup \hat{\Gamma}, \delta', q_0', F)$ by adding a new state $q_0'$ and by setting

$$\delta'(q_0', Z_0) = \{q_0\},$$
$$\forall p \neq q_0' : \ \delta'(p, x\bar{a}\alpha^R) \ni q, \ \text{ if } \ (\alpha, q) \in \delta(p, x, a).$$

Recall that $\bar{a}$ denotes the symbol in $\overline{\Gamma}$ such that $a\bar{a} \sim \varepsilon$. This transformation is depicted in Fig. 3. It suffices now to denote by $R$ the regular language accepted by $A'$. One can observe that $A'$ simulates the computations of $A$, except the situations when $A$ rejects the input caused by a stack mismatch. In other words, $A'$ simulates $A$ on $R \cap \mathcal{E}\big(X, \mathcal{S}(\Gamma)\big)$. Projecting this intersection onto $X^*$ restores the input words accepted by $A$.

(iii) $\Rightarrow$ (i). Let $L = \pi_X\Big(R \cap \mathcal{E}\big(X, \mathcal{S}(\Gamma)\big)\Big)$ for some regular language $R \in (X \cup \hat{\Gamma})^*$. Recall that the family of context-free languages is closed under morphisms and that the intersection between a regular and a context-free language is context-free. The conclusion follows immediately from the fact that $\mathcal{E}\big(X, \mathcal{S}(\Gamma)\big)$ is context-free (Property 5). □

In the above theorem, the requirement that $\Gamma$ and $X$ are disjoint may be dropped, by replacing the projection with an arbitrary monoid homomorphism. For exemplification, we present this technique on a different result, this time in the context of rational transductions, hence anticipating the main purpose of this paper, that of the representation of algebraic transductions. Recall that a word transduction over two alphabets $X$ and $Y$ is simply a relation $\nu \in X^* \times Y^*$. By convenience, we view $\nu$ as a mapping from $X^*$ into $2^{Y^*}$, and we use the short notation $\nu : X^* \to Y^*$.

The First Factorization Theorem ([7, Sect. IX, T. 2.2.]) for rational relations, also known as Nivat's representation theorem for rational transductions ([12]), can be re-stated in terms of projections and identities as following:

**Theorem 2** *Let $X$ and $Y$ be disjoint alphabets. A transduction $\nu : X^* \to Y^*$ is rational if and only if there exists a regular language $R \subseteq (X \cup Y)^*$ such that*

$$\nu = \pi_Y \circ I_R \circ \pi_X^{-1},$$

*where $I_R$ is the identity on $R$ (see* Fig. 4).

For completeness, we give here a sketch of the proof.

*Proof* It suffices to observe that given a finite transducer $T$ realizing $\nu$, we can construct a finite automaton $A$ as following: $A$ has the same set of states (and initial and

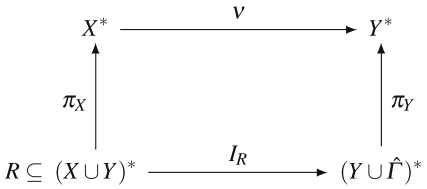



**Fig. 4** A factorization of rational transductions

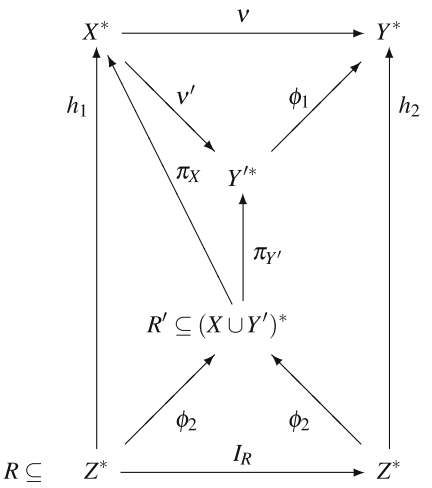



**Fig. 5** Nivat's characterization of rational transductions

final states) as $T$ and for every transition $(p, x, w, q)$ in $T$ with $p, q$ states, $x$ input word and $w$ output word, we associate a transition $(p, xw, q)$ in $A$. We denote by $R$ the language accepted by $A$ and is easy to observe that projecting any word $w \in R$ onto both $X^*$ and $Y^*$ leads to pairs of words in $\nu$. □

Notice that the requirement that $X$ and $Y$ are disjoint is a rather weak condition, since in the general case we can always perform a "coloring" (i.e., isomorphic transformation) and ensure this condition. This method leads us to the first factorization theorem in the general and original form. The process of coloring is depicted in Fig. 5 and is detailed as following:

1. If $X$ and $Y$ are not disjoint, map bijectively $Y$ into a new alphabet $Y'$ such that $Y' \cap X = \emptyset$. Extend this mapping to an isomorphism $\phi_1 : Y'^* \to Y^*$. We can trivially construct a transduction $\nu' : X^* \to Y'^*$ such that $\nu = \phi_1 \circ \nu'$.
2. We can now apply the theorem for $\nu'$, since $X$ and $Y'$ are disjoint. Then, there exists a regular language $R' \subseteq (X \cup Y')^*$ such that $\nu' = \pi_{Y'} \circ I_{R'} \circ \pi_X^{-1}$.
3. In order to replace $(X \cup Y')^*$ with an arbitrary free monoid, we perform a second coloring by choosing a new alphabet $Z$ bijectively mapped into $(X \cup Y')$. As before we extend this bijection to an isomorphism $\phi_2 : Z^* \to (X \cup Y')^*$. Then denote $R = \phi_2^{-1}(R')$, which is regular in $Z^*$.
4. Finally, set $h_1 = \pi_X \circ \phi_2$ and $h_2 = \phi_1 \circ \pi_{Y'} \circ \phi_2$. Then we obtained the following general result:

**Theorem 3** *A transduction $\nu : X^* \to Y^*$ is rational if and only if there exist an alphabet $Z$, two morphisms $h_1 : Z^* \to X^*$ and $h_2 : Z^* \to Y^*$, and a regular language $R \subseteq Z^*$ such that $\nu = h_2(R \cap h_1^{-1})$.*

The formula is interpreted as $\nu(u) = h_2(R \cap h_1^{-1}(u))$. Notice that the "if" part of this result is proven trivially, since $\nu$ becomes a composition of rational transductions (projections, identity over regular languages, morphisms and inverse morphisms are rational).

This technique can be applied to most results in this paper, that are stated for disjoint alphabets. It is in our belief that the natural way for obtaining transduction characterizations is to first obtain factorizations through "constants," i.e. through projections and identities over predefined languages, and then to generalize them to work over disjoint alphabets. In the following section we apply similar methods to characterize algebraic transductions.

Finally, let us recall an important property of rational relations, namely the "uniformization property." The result basically says that for any rational relation $\nu$, one can find a rational function that "chooses" for every word $u$ in the domain of $\nu$ a word in the image $\nu(u)$. To be more precise, we give the following definition and theorem:

**Definition 4** *Let $\nu : X^* \to Y^*$ be an arbitrary relation. An* `uniformization of` $\nu$ *is any function $f : X^* \to Y^*$ such that $dom(f) = dom(\nu)$ and $f(u) \in \nu(u), \forall u \in dom(\nu)$.*

**Theorem 4** (Rational Uniformization Theorem–[11]) *Any rational relation is uniformized by an unambiguous rational function.*

Since any rational function $f : X^* \to Y^*$ (which satisfies that either $f(\varepsilon) = \varepsilon$ or $f$ is undefined in $\varepsilon$) is unambiguous ([7, Sect. IX.8, T. 8.1]), the unambiguity property mentioned in the previous theorem is somehow redundant. However, a similar result for algebraic relations would not manifest such redundancy due to the existence of algebraic functions that are not unambiguous (for example, the identity over any inherently ambiguous context-free language). Notice that functionality of algebraic transductions is clearly an undecidable property – surprisingly, this matter was reported as an open problem in [5, Sect. 3.5.1, p. 116]. Indeed, given a context-free grammar, one can construct a pushdown transducer that, for each word generated by the grammar, it outputs its left derivation. Then the transducer is functional if and only if the grammar is unambiguous, a property known to be undecidable in general.

## 3 Algebraic transductions: characterization

In the previous section we have recalled a representation of rational transductions (word relations) by means of regular languages and projections/morphisms. A similar representation can be obtained for algebraic transductions, i.e., for transductions realized by pushdown transducers. Indeed, such a transduction can be expressed by means of morphisms and context-free languages, a characterization sometimes used as a definition:

**Definition 5** ([3, Sect. II.4, p. 71]) *A transduction $\tau : X^* \to Y^*$ is* `algebraic` *if there exist an alphabet $Z$, two morphisms $h_1 : Z^* \to X^*$, $h_2 : Z^* \to Y^*$, and a context-free language $L \subseteq Z^*$ such that*

$$\tau = h_2(L \cap h_1^{-1}).$$

In the following we present yet another two characterizations of algebraic transductions, by means of projections/morphisms and either regular languages or rational relations.

**Theorem 5** *Let $X, Y$ be disjoint alphabets and $\tau$ be a word relation from $X^*$ to $Y^*$. Then the following statements are equivalent:*

(i) *$\tau$ is algebraic.*

(ii) *There exists an alphabet $\Gamma$ (disjoint from $X$ and $Y$) and a regular language $R$ over $X \cup \hat{\Gamma} \cup Y$, such that*

$$\tau = \pi_Y\Big(\pi_X^{-1} \cap R \cap \mathcal{E}\big(X \cup Y, \mathcal{S}(\Gamma)\big)\Big),$$

*where $\pi_X$ is the projection of $(X \cup \hat{\Gamma} \cup Y)^*$ onto $X$.*

(iii) *There exist an alphabet $\Gamma$ (disjoint from $X$ and $Y$) and a rational transduction $\nu$ from $X^*$ to $\big(\hat{\Gamma} \cup Y\big)^*$ such that*

$$\tau = \pi_Y\Big(\nu \cap \mathcal{E}\big(Y, \mathcal{S}(\Gamma)\big)\Big),$$

*where $\pi_Y$ is the projection of $(\hat{\Gamma} \cup Y)^*$ onto $Y$.*

(iv) *There exist an alphabet $\Gamma$ (disjoint from $X$ and $Y$) and a rational transduction $\eta$ from $(X \cup \hat{\Gamma})^*$ to $Y^*$ such that*

$$\tau = \eta\Big(\pi_X^{-1} \cap \mathcal{E}\big(X, \mathcal{S}(\Gamma)\big)\Big),$$

*where $\pi_X$ is the projection of $(X \cup \hat{\Gamma})^*$ onto $X$.*

*Moreover, the theorem still holds if we replace $\mathcal{S}(\Gamma)$ by $\mathcal{D}(\Gamma)$.*

*Remark 1* To be more precise on the meaning of these notations, we interpret – for example – the relation used in (ii) as:

$$\tau(u) = \pi_y\Big(\pi_X^{-1}(u) \cap R \cap \mathcal{E}\big(X \cup Y, S(\Gamma)\big)\Big) \ ,$$

where we consider $\pi_y$ as the additive extension of the projection to subsets of its domain.

*Proof* (i) $\Leftrightarrow$ (ii). Figure 6 is a useful companion for this proof. Consider $\tau$ to be algebraic and let us prove (ii). Let $T = (Q, X, Y, \Gamma, \delta, q_0, Z_0, F)$ be a pushdown transducer which realizes $\tau$ by final state.

Construct the "lazy" NFA $A = (Q \cup \{q_0'\}, X \cup \hat{\Gamma} \cup Y, \delta', q_0', F)$ by adding a new state $q_0'$ and by defining the transition function as following:

$\delta(q_0', Z_0) = \{q_0\}$;
$\forall p \neq q_0' : \delta'(p, x\bar{a}\alpha^R w) \ni q$, if $(\alpha, w, q) \in \delta(p, x, a)$.

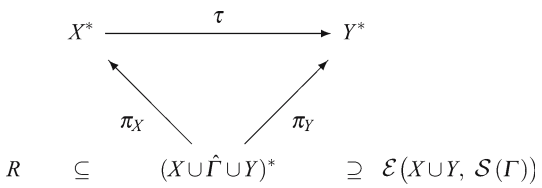



**Fig. 6** Characterization of algebraic transductions by regular languages

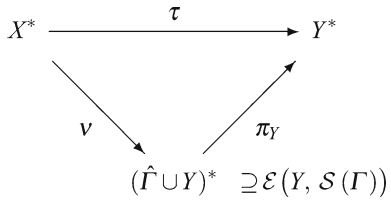



**Fig. 7** First characterization of algebric transductions by rational transductions

If we denote by $R$ the language accepted by $A'$, observe that $A'$ simulates the computations of $A$ on inputs in $R \cap \mathcal{E}(X \cup Y, \mathcal{S}(\Gamma))$, hence the conclusion follows shortly.

Vice-versa, consider that (ii) holds. Notice that $R \cap \mathcal{E}(X \cup Y, \mathcal{S}(\Gamma))$ is a context-free language. The fact that $\tau$ is algebraic follows immediately from the Nivat-like characterization of algebraic transductions (Definition 5).

We now prove (i)$\Leftrightarrow$(iii). We use the diagram in Fig. 7. Consider once again that $\tau$ is algebraic, realized by a pushdown transducer $T = (Q, X, Y, \Gamma, \delta, q_0, Z_0, F)$, by final state. Construct the finite transducer $T' = (Q \cup \{q_0'\}, X, \hat{\Gamma} \cup Y, \delta', q_0', F)$ with $q_0'$ new state and $\delta'$ given by

$\delta'(q_0', Z_0) = \{(\varepsilon, q_0)\}$;
$\forall p \neq q_0' : \delta'(p, x) \ni (\overline{a}\alpha^R w, q)$, if $(\alpha, w, q) \in \delta(p, x, a)$.

$T'$ simulates the computations of $T$ when its output is in $\mathcal{E}(Y, \mathcal{S}(\Gamma))$, whence the conclusion follows.

Vice-versa, suppose (iii) holds and consider the identity on $\mathcal{E}(Y, \mathcal{S}(\Gamma))$, given by

$$I_{\mathcal{E}(Y, \mathcal{S}(\Gamma))} = \left\{(u, u)/u \in \mathcal{E}(Y, \mathcal{S}(\Gamma))\right\}.$$

Since $\mathcal{E}(Y, \mathcal{S}(\Gamma))$ is context-free, trivially $I_{\mathcal{E}(Y, \mathcal{S}(\Gamma))}$ is an algebraic relation over $(\hat{\Gamma} \cup Y)^*$. It is also easy to notice that $\pi_Y \circ I_{\mathcal{E}(Y, \mathcal{S}(\Gamma))}$ is also an algebraic relation in $(\hat{\Gamma} \cup Y)^* \times Y^*$. Then $\tau$ is the composition of a rational relation $\nu$ and an algebraic relation $\pi_Y \circ I_{\mathcal{E}(Y, \mathcal{S}(\Gamma))}$, hence it is algebraic.

We prove now (i) $\Leftrightarrow$ (iv). The diagram in Fig. 8 is a companion of this proof. Considering $\tau$ to be algebraic, realized by the pushdown transducer $T = (Q, X, Y, \Gamma, \delta, q_0, Z_0, F)$ by final state, we construct a finite transducer $T' = (Q \cup \{q_0'\}, X \cup \hat{\Gamma}, Y, \delta', q_0', F)$ with the transition function given by

$\delta'(q_0', Z_0) = \{(\varepsilon, q_0)\}$
$\forall p \neq q_0' : \ \delta'(p, x\overline{a}\alpha^R) \ni (w, q)$ if $(\alpha, w, q) \in \delta(p, x, a)$.

If we denote by $\eta$ the rational transduction realized by $T'$, one observes that (iv) holds.

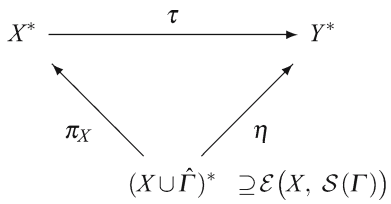



**Fig. 8** Second characterization of algebraic transductions by rational transductions

Conversely, if we consider that (iv) holds, it is easy to observe that $\pi_X^{-1} \cap \mathcal{E}\big(X, \mathcal{S}(\Gamma)\big)$ can be realized by a pushdown transducer, hence that $\tau$ is a composition of an algebraic and a rational transduction. It immediately follows that $\tau$ is algebraic. □

Using the coloring technique mentioned in the previous section, we obtain the following known characterizations of algebraic transductions - which cover the case of non-disjoint alphabets:

**Corollary 1** *Let $X$ and $Y$ be arbitrary languages (not necessarily disjoint). Then the following statements are true.*

(ii′) *$\tau : X^* \to Y^*$ is algebraic if and only if there exist an alphabet $Z$, a context-free language $L \in Z^*$ and two morphisms $h_1 : Z^* \to X^*$ and $h_2 : Z^* \to Y^*$ such that $\tau = h_2 \circ I_L \circ h_1^{-1}$.*

(iii′) *$\tau : X^* \to Y^*$ is algebraic if and only if there exist an alphabet $Z$, a context-free language $L \in Z^*$, a morphism $h : Z^* \to Y^*$ and a rational transduction $\nu : X^* \to Y^*$ such that $\tau = h \circ I_L \circ \nu$.*

Notice that this result is somehow stronger than (ii′), since morphisms are particular cases of rational transductions. The property captured by (iii′) is that the composition of a rational and an algebraic transduction is algebraic.

(iv′) *Similar to* (iii′)*, we can have a factorization $\tau = \nu \circ I_L \circ h^{-1}$.*

Observe that all characterizations are by means of only one "variable": in (ii) a regular language, in (iii) and (iv) a rational transduction. However, the classical characterizations (Corollary 1 (ii′), for example) use three variables (two morphisms and a context-free language, in this case). It is not clear which characterizations are more general; however it appears that Theorem 5 can be applied arguably more easily than Corollary 1, since it bears more information.

In Fig. 9 we detail the coloring used to obtain (ii′) of Corollary 1 from (i) of Theorem 5. Since we work with three disjoint alphabets ($X$, $\Gamma$, and $Y$), we require

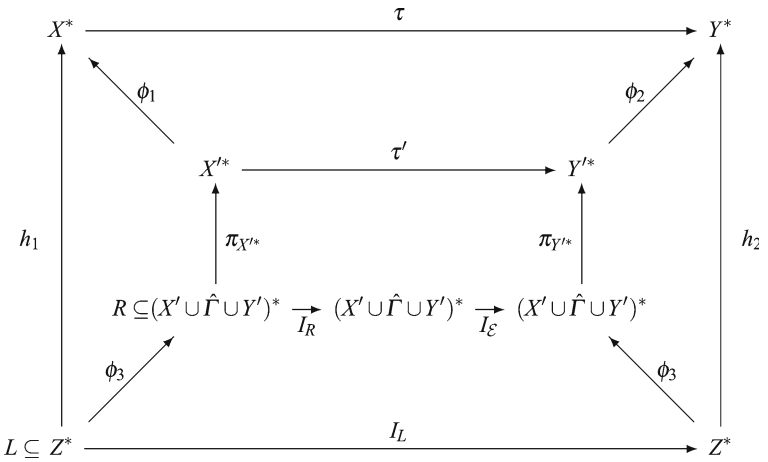



**Fig. 9** A coloring that infers (ii′) from (ii)

the coloring of both $X$ and $Y$. $\phi_1, \phi_2$ and $\phi_3$ are colorings (isomorphisms) and $R$ is a regular language given by (ii). We also have $L = \phi_3^{-1}(R)$, $h_1 = \phi_1 \circ \pi_{X'^*} \circ \phi_3$ and $h_2 = \phi_2 \circ \pi_{Y'^*} \circ \phi_3$ that are obviously homomorphisms which verify $\tau = h_2 \circ I_L \circ h_1^{-1}$.

To conclude this section, we give a "matrix representation" of algebraic transductions, based on the characterization of Theorem 5.

**Definition 6** *Let $X$ and $Y$ be two disjoint alphabets. An* `algebraic matrix representation` *(AMR, for short) from $X^*$ into $Y^*$ is a tuple $M = (\Gamma, Q, q_0, F, \mu)$ where*

- *$\Gamma$ is a* `stack alphabet`*, having no symbols in common with $X \cup Y$;*
- *$Q$ is a finite* `set of states`*;*
- *$q_0 \in Q$ is an* `initial state`*;*
- *$F \subseteq Q$ is a* `set of final states` *and*
- *$\mu$ is a semigroup morphism from $X^*$ into the multiplicative monoid of the semiring $\left[Rat(\hat{\Gamma} \cup Y)^*\right]^{Q \times Q}$,*

$$\mu : X^* \to \left[Rat(\hat{\Gamma} \cup Y)^*\right]^{Q \times Q}.$$

The transductions $|M_F|, |M_S| : X^* \to Y^*$ realized by $M$ are defined by

$$|M_F|(w) = \pi_Y\left(\bigcup_{q \in F} \mu(w)_{q_0 q} \cap \mathcal{E}\big(Y, \mathcal{S}(\Gamma)\big)\right),$$

$$|M_S|(w) = \pi_Y\left(\bigcup_{q \in Q} \mu(w)_{q_0 q} \cap \mathcal{E}\big(Y, \mathcal{D}(\Gamma)\big)\right).$$

$M_F$ will be called an "algebraic matrix representation by final state," and $M_S$ will be called an "algebraic matrix representation by final state and empty stack." Their corresponding families of transductions will be denoted by $\mathrm{AMR}_F$ and $\mathrm{AMR}_S$ respectively. In the following we state that both families of AMR are one with the family of algebraic transductions from $X^*$ into $Y^*$.

**Corollary 2** *Let $X$ and $Y$ be two disjoint alphabets and $\tau$ be a transduction from $X$ into $Y$. The following statements are equivalent:*

(i) *$\tau$ is algebraic.*
(ii) *There exists an $AMR_F$ $M$ such that $\tau =\mid M \mid$.*
(iii) *There exists an $AMR_S$ $M$ such that $\tau =\mid M \mid$.*

*In other words,*

$$\mathrm{AMR}_F = \mathrm{AMR}_S = Alg(X^* \times Y^*).$$

*Proof* This is a direct consequence of Theorem 5, affirmation (iii), in which we replace the rational transduction $\nu$ with its matrix representation (see [13, Sect. IV.1.5.b] for an introduction to matrix representations for rational relations). □

## 4 Uniformization of algebraic transductions

Our next purpose is to use the properties of rational transductions in order to produce an uniformization of algebraic transductions. Essentially, considering the factorization in Fig. 7, we investigate the situation when $\nu$ is replaced by its uniformization: we will see that in specific circumstances we indeed obtain an uniformization of $\tau$. However, we could not produce an uniformization for the most general case, and it is in our belief that this is probably not possible.

We first give a uniformization result that can be drawn directly from the existing properties of rational and algebraic transductions. In the following, by *AlgF* we denote the family of algebraic functions, and by *UAlgF* the family of unambiguous algebraic functions (i.e., of those functions realized by unambiguous pushdown transducers). Similarly, by *RatF* we denote functions realized by finite transducers.

**Proposition 1** *Any algebraic relation in Rat* $\circ$ *AlgF is uniformized by an algebraic function. Any algebraic relation in Rat* $\circ$ *UAlgF is uniformized by an unambiguous algebraic function.*

*Proof* Let $\tau : X^* \to Y^*$ be an algebraic transduction which accepts a factorization $\tau = \nu \circ \eta$ with $\nu : Z^* \to Y^*$ rational relation and $\eta : X^* \to Z^*$ algebraic function ($X, Y, Z$ are alphabets). By the uniformization theorem for rational transductions (Theorem 4), there exists a rational unambiguous function $\nu' : Z^* \to Y^*$ which uniformizes $\nu$. Then, $\nu' \circ \eta$ clearly uniformizes $\tau$, by the fact that $\eta$ is a function as well and its codomain has remained unchanged. If $\eta$ is unambiguous then $\nu' \circ \eta$ becomes unambiguous as well, since it is known (see, for example, [4]) that $RatF \circ UAlgF = UAlgF$. □

Let us give an intuitive description of the factorization of algebraic transductions depicted in Fig. 7 and reproduced (and interpreted) for convenience in Fig. 10, in terms of homomorphisms. The transducer for $\nu$ was constructed in such way, that it simulates the computations of the initial pushdown automaton in terms of state transitions triggered by the input and ignoring the correctness of stack operations. Then, the identity over a stack language does nothing more than filter out all those computations that are invalid from the stack point of view. Finally, the homomorphism in the last stage extracts from the remaining valid computations the corresponding output.

As mentioned in the introduction of this section, we investigate sufficient conditions in which the uniformization of $\nu$, as shown in the decomposition of $\tau$ in Fig. 10, triggers an uniformization of $\tau$.

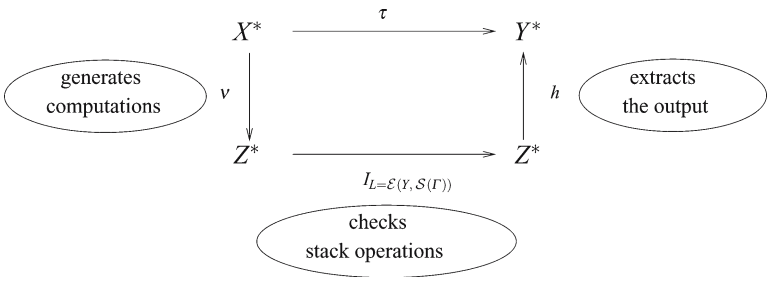



**Fig. 10** Factorization of algebraic transductions–an interpretation

**Definition 7** *Let $A$ be an arbitrary set and $\mathcal{D} = \{D_j\}_{j\in J}$ be an arbitrary family of sets. We say that $A$* `separates` *$\mathcal{D}$ if for any nonempty set $D_j \in \mathcal{D}$, either $D_j \subseteq A$ or $D_j \cap A = \emptyset$.*

Notice in this definition that if $D_1, D_2$ are two sets of $\mathcal{D}$ such that $D_1 \cap D_2 \neq \emptyset$ then they are either both included in $A$ or neither has common elements with $A$. Notice also that the condition in the previous definition applies only to nonempty sets of the family ($\emptyset$ always satisfies both relations).

**Theorem 6** *Let $\tau : X^* \to Y^*$ be an arbitrary algebraic transduction and $h \circ I_L \circ \nu$ be a factorization of $\tau$, with $L$ a context-free language over an alphabet $Z$, $\nu : X^* \to Z^*$ a rational relation, and $h : Z^* \to Y^*$ a monoid morphism.*

(i) *If $L$ separates $\{\nu(x)\}_{x\in X^*}$, then $\tau$ can be uniformized by an algebraic function.*
(ii) *If in addition $L$ is unambiguous, then $\tau$ is uniformized by an unambiguous algebraic function.*

*Proof* We uniformize $\nu$ by an unambiguous rational function $\nu'$ and prove that the new composition $\tau' = h \circ I_L \circ \nu'$ is an algebraic uniformization for $\tau$. It is clear that $\tau'$ is an algebraic function. It remains to prove that $dom(\tau) = dom(\tau')$. Assume by contrary that there exists a word $u \in dom(\tau) \setminus dom(\tau')$. Since the domain of $h$ is $Z^*$, it follows that $\nu'(u) \notin L$, as the only possible reason for which $u \notin dom(\tau')$. But since $L$ separates $\{\nu(x)\}_{x\in X^*}$, and since $\nu'(u) \in \nu(u)$, we infer that $\nu(u) \notin L$. This further implies that $u \notin dom(\tau)$ – a contradiction. Hence we have proven that $\tau'$ is an algebraic uniformization of $\tau$. For the second implication, we have that $I_L$ is in $UAlgF$, by the fact that $L$ is unambiguous. Then $\tau' \in RatF \circ UAlgF \circ RatF = UAlgF$. □

**Corollary 3** *Let $X$ and $Y$ be disjoint alphabets, and consider the factorization of $\tau$ as in Theorem 5* (iii): $\tau = \pi_Y\Big(\nu \cap \mathcal{E}\big(Y, \mathcal{S}(\Gamma)\big)\Big)$. *Then $\tau$ can be uniformized by an unambiguous algebraic function if $\mathcal{E}\big(Y, \mathcal{S}(\Gamma)\big)$ separates $\{\nu(x)\}_{x\in X^*}$.*

*Proof* We apply the previous result. It suffices to observe that $\mathcal{E}\big(Y, \mathcal{S}(\Gamma)\big)$ is an unambiguous context-free language. □

Let us interpret the conditions of Theorem 6/Corollary 3. Let $A$ be a PDA or PDT with acceptance by final state, and let $A'$ be the NFA obtained from $A$ by keeping only the input label/symbol for each transition (i.e., we simply discard all the stack operations and output symbols if dealing with a transducer). $A$ and $A'$ have exactly the same transition graph (except the labels): their edges and vertices are in bijective correspondence.

**Definition 8** *We say the $A$ has a* `fair stack` *if and only if for any word $w$ in the domain of $A$ the following property holds:*

> *for any successful computation path in $A'$ that recognizes $w$ there corresponds a successful computation path in $A$, that follows exactly the same transitions as in $A'$.*

In other words, if $w$ is in the domain of $A$ (i.e., it is accepted by $A$), then all the successful paths for $w$ in $A'$ and $A$ coincide (in $A$, the stack does not crash any of these paths ). Notice that the definition does not mention anything about the words that are not in the domain of $A$ (we are not interested in the rejected words).

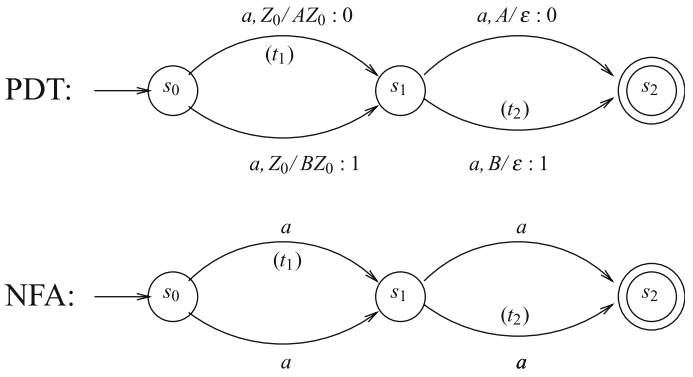



**Fig. 11** Pushdown transducer that has a uniformization and does not satisfy Theorem 6

**Lemma 1** *An algebraic transduction satisfies the conditions of Theorem* 6*/Corollary* 3 *if and only if it is realized by a PDT with fair stack.*

*Proof* The proof is straightforward. □

*Example 1* The transducer (PDT) in Fig. 11 represents the transduction $\tau = \{(aa, 00), (aa, 11)\}$, hence $aa$ is a word in its domain. For this word, the corresponding NFA (in the same figure) has a successful computation: $s_0t_1s_1t_2s_2$, where $t_1$ and $t_2$ are used to denote the exact transitions that are followed in the computation path (see the figure). However, this path is not successful in the PDT, since the transition $t_2$ attempts to pop a $B$ from the stack, and the previous transition $t_1$ has pushed an $A$ onto the stack. This implies that the transducer does not have a fair stack. Notice that despite this fact, the transducer accepts a trivial uniformization, a fact which shows that Theorem 6/Corollary 3 give a condition for uniformization that is sufficient but not necessary.

It is in our intention to continue the study of fair stack pushdown machines; however, this work is beyond the scope of the present paper. In particular, if we denote by *fs*CF the family of context-free languages that are accepted by *fs*PDA (PDA with fair stack), we leave as an open question whether

$$\mathrm{CF} = \mathit{fs}\mathrm{CF}$$

or not (we believe that the equality does not hold). Notice that if the equality holds, then Theorem 6 becomes an uniformization theorem for algebraic transductions in general. If the equality does not hold, then such uniformization still remains an open problem. Notice also that the "fair stack" property does not reduce pushdown machines to a trivial status; for example, there exist a PDA with fair stack for $\{a^ib^i/i \geq 0\}$ and for other CF languages that are not regular. In this matter, more needs to be investigated.

*Example 2* In the following, we give a simple example of obtaining an uniformization for an algebraic transduction–as application of Corollary 3. Let us consider the pushdown transducer $A$, given in Fig. 12. This transducer has a fair stack, hence we can use the constructive proof of Theorem 6 in order to algorithmically obtain an

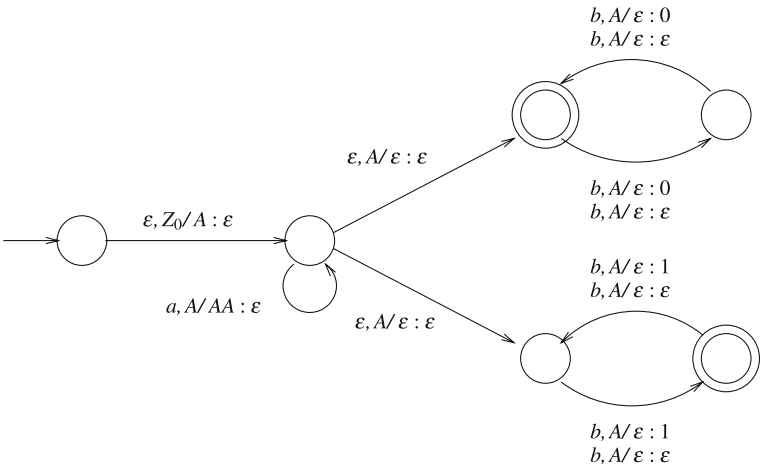



**Fig. 12** Pushdown transducer that satisfies the conditions of Theorem 6

uniformization for its corresponding transduction. Denote by $\tau$ the algebraic transduction realized by $A$, and observe that

$$\tau = \{(a^i b^j, 0^k)/0 \le k \le j \le i,\ \ j \text{ even}\} \ \cup \\ \cup\ \{(a^i b^j, 1^k)/0 \le k \le j \le i,\ \ j \text{ odd}\}\ .$$

We denote $X = \{a, b\}$, $Y = \{0, 1\}$, $\Gamma = \{Z_0, A\}$, $Z = X \cup Y \cup \hat{\Gamma}$, $L = \mathcal{E}(Y, \mathcal{S}(\Gamma))$, and $h = \pi_Y$. It remains to find $\nu$ in the factorization $\tau = h \circ I_L \circ \nu$ as in Theorem 6 and Fig. 10. By following the construction depicted in Fig. 15, we obtain the finite transducer in Fig. 13. This finite transducer represents the following rational relation:

$$\nu(a^i b^j) = \begin{cases} Z_0 \overline{Z_0} A (\overline{A} A A)^i \overline{A} \Big(\overline{A}(0+\varepsilon)\Big)^j, & \text{if } j \text{ even} \\ Z_0 \overline{Z_0} A (\overline{A} A A)^i \overline{A} \Big(\overline{A}(1+\varepsilon)\Big)^j, & \text{otherwise,} \end{cases}$$

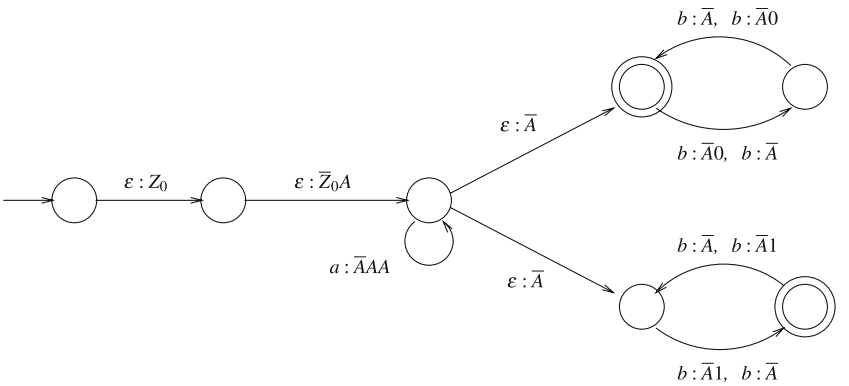



**Fig. 13** Finite transducer for the factorization corresponding to Fig. 12

whose domain is $\{a^i b^j / i, j \geq 0\}$. We further notice that

$$(I_L \circ \nu)(a^i b^j) = \begin{cases} \nu(a^i b^j), & \text{if } j \leq i \\ \emptyset, & \text{otherwise.} \end{cases}$$

Suppose we have the following uniformization for $\nu$:

$$\nu'(a^i b^j) = \begin{cases} Z_0 \overline{Z_0} A (\overline{A} A A)^i \overline{A} (\overline{A A} 0)^{\frac{j}{2}}, & \text{if } j \text{ even} \\ Z_0 \overline{Z_0} A (\overline{A} A A)^i \overline{A A} 1 (\overline{A A} 1)^{\frac{j-1}{2}}, & \text{if } j \text{ odd.} \end{cases}$$

It is easy to see that $\nu'$ indeed uniformizes $\nu$, since it is realized by a finite transducer obtained from the transducer for $\nu$ (Fig. 13) by eliminating several transitions in order to ensure functionality. Then, $\nu'$ induces an uniformization $\tau' = h \circ I_L \circ \nu'$ for $\tau$, given by

$$\tau' = \left\{ (a^i b^j, 0^{\frac{j}{2}}) / 0 \leq j \leq i, \ j \text{ even} \right\} \cup \left\{ (a^i b^j, 1^{\frac{j-1}{2}+1}) / 0 \leq j \leq i, \ j \text{ odd} \right\}.$$

An unambiguous transducer for $\tau'$ can be obtained by removing from the transducer $A$ in Fig. 12 the following transitions: in the upper cycle, the lower transition $(b, A/\varepsilon : 0)$ and upper transition $(b, A/\varepsilon : \varepsilon)$, and in the lower cycle, the lower transition $(b, A/\varepsilon : \varepsilon)$ and the upper transition $(b, A/\varepsilon : 1)$. One can check that the obtained transducer is unambiguous and therefore $\tau'$ is unambiguous.

**Corollary 4** *Let $\tau : X^* \to Y^*$ be an algebraic transduction realized by an unambiguous pushdown transducer with a fair stack. Then $\tau$ is functional and there exist an alphabet $Z$, a context free language $L \subseteq Z^*$, a morphism $h : Z^* \to Y^*$ and a rational function $\nu : X^* \to Z^*$ such that $\tau = h \circ I_L \circ \nu$.*

*Proof* The diagram in Fig. 14 is a rewriting of the diagram in Fig. 7. We invoke the construction of $\nu$ from $\tau$, as detailed in the proof of Theorem 5. If $\tau$ is given by a PDT $T$ with fair stack, we construct a finite transducer $T'$ which realizes $\tau$, as shown in Fig. 15. It is clear from the construction that, if $T$ is chosen to be unambiguous (any input word is the label of at most one successful computation in $T$) then $T'$ becomes unambiguous as well, hence $\nu$ is actually a rational function. Finally, $\tau$ is the composition of functions, hence it is functional.

Notice that if $T$ did not have a fair stack we could not reach this conclusion since $T'$ would not necessary be unambiguous. □

It remains for further investigation to find (if there exist) necessary and sufficient conditions such that an algebraic transduction is uniformized by an unambiguous algebraic function.

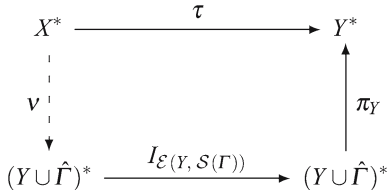



**Fig. 14** A companion for the proof of Corollary 4

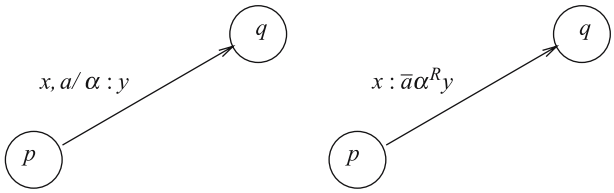



**Fig. 15** Transformation of a PDT into a FT

## 5 The anatomy of pushdown transducers

In the following we define a new family of algebraic transductions, prove that it is a proper subfamily and characterize it. We further use these transductions to investigate the algebraic mechanisms behind a pushdown transducer.

**Definition 9** *Let* $A = (Q, X, \Gamma, \delta, q_0, Z_0, F)$ *be a pushdown automaton. The* `stack transduction` *defined by A is the transduction* $s_A : X^* \to \Gamma^*$*, given by*

$$s_A(w) = \{\gamma / \exists q \in F \text{ s.t. } (q_0, w, Z_0) \vdash (q, \varepsilon, \gamma)\}$$

In other words, the stack transduction associates an accepted input word with the "stack residue" left at the end of the computation. Notice that, since there may be multiple successful computations for a given input, a stack transduction is indeed a relation and not necessarily a function.

In the following we sometime consider "lazy" PDA, i.e., transducers that have transitions with words as input labels, rather than letters (or $\varepsilon$). We may do so in the virtue of the following property: given a lazy PDA $A$, there exists an equivalent PDA $B$ (standard) such that $s_A = s_B$. Indeed, we may transform each transition in $A$ labeled with an input word of length greater than one into a chain of transitions (by adding new states) where we "keep the stack working" by artificially pushing and popping a special stack symbol.

We denote by *AlgS* the family of all stack transductions from $X^*$ to $\Gamma^*$. Then the following inclusion holds.

**Proposition 2**

$$AlgS \subset Alg,$$

*and the inclusion is strict.*

*Proof* For the inclusion, it suffices to remark that given a pushdown automaton, we can transform it into a pushdown transducer by adding output transitions from every final state (which becomes non-final) to a new final state with a loop which "dumps" the stack on an output tape.

To show that the inclusion is strict, we observe that the transduction $\tau = \{(a^i, b^i c^i) / i \geq 0\}$ is algebraic, but is not a stack transduction, as explained in the following.

A pushdown transducer for $\tau$ reads $a$'s from the input tape, and for each $a$ it pushes a corresponding $c$ onto the stack and it writes a $b$ on the output tape. When the input is exhausted, it "dumps" the content of the stack on the output tape, by an $\epsilon$-input loop. This construction shows that $\tau$ is algebraic.

Suppose there exists a PDA $A$ which realizes $\tau$; in other words, there exists a PDA which reads a number $i$ of $a$'s from the input tape and upon acceptance, its stack contains the word $b^i c^i$. We claim that this machine can be modified into a PDT that realizes the relation $\tau' = \{(a^i, a^i b^i c^i)/i \geq 0\}$. This will lead to a contradiction, due to the fact that $\tau'$ is not algebraic (by a well-known property of algebraic transducers, of mapping their domain into a context-free language). We take the PDA $A$, augment to it an output tape, and modify its transitions to write on the output tape the symbols read from the input. We add extra transitions from all final states, such that at the end of scanning, the transducer "dumps" the contents of its stack on the output tape. Therefore, while reading $a^i$ the transducer writes $a^i$ and stores $b^i c^i$ in its stack. Then, at the end of scanning it writes $b^i c^i$ (concatenating to the existing $a^i$) to the output tape. The outcome is a computation for $(a^i, a^i b^i c^i)$. □

*Remark 2* $AlgS \neq AlgS^{-1}$; in other words, the family of stack transductions is not closed under the "inverse" operator. Indeed, take for example the transduction $\{(a^i b^i, \varepsilon)/i \geq 0\}$. This is a stack transduction; however its inverse is not (we could not produce $a^i b^i$ on a stack, unless we had available a second stack–and then we would end up with a Turing machine). Nevertheless, $AlgS^{-1} \subseteq Alg$ since $AlgS \subseteq Alg$ and $Alg$ is closed under inverse.

Given the stack language $S(\Gamma)$ – as defined in the introduction – we define a transduction which "eliminates paired symbols". In other terms, the transducer will have $S(\Gamma)$ as its domain and for a given word $w \in S(\Gamma)$, it will output the unique word $\gamma \in \Gamma^*$ such that $w \sim \gamma$, where $\sim$ is the restricted Dyck congruence. This is certainly an algebraic function, and let us denote it by $\sigma : \hat{\Gamma}^* \to \Gamma^*$, given by

$$\sigma(w) = \gamma, \quad \text{where} \quad w \sim \gamma, \ \ \gamma \in \Gamma^*.$$

Notice that $\sigma$ itself is a stack transduction.

**Theorem 7** *A transduction $s : X^* \to \Gamma^*$ is a stack transduction if and only if there exists a rational transduction $\nu : X^* \to \hat{\Gamma}^*$ such that $s = \sigma \circ \nu$.*

*Proof* The diagram for this result is depicted in Fig. 16. For the "only if" part, we are given a PDA $A$ such that $s_A = s$ and we construct a finite transducer for $\nu$, similar to that in the proof of Theorem 5 (iii). For example, if $(p, x, a/\alpha, q)$ is a transition in $A$, in the transducer for $\nu$ we have a corresponding transition $(p, x : \bar{a}\alpha^R, q)$. Then, the transduction $\sigma$ will both filter out the words which are not in $S(\Gamma)$ and will produce (write on an output tape) the "stack residue", i.e., what remains in the stack of $A$ upon acceptance.

For the "if" part, we proceed as following.

1. Let $T$ be a transducer for $\nu$ with the initial state $q_0$. First, we augment to $\Gamma$ (and $\overline{\Gamma}$, respectively) the new symbol $Z_0$ (and $\overline{Z}_0$, respectively) and we add to $T$ a new

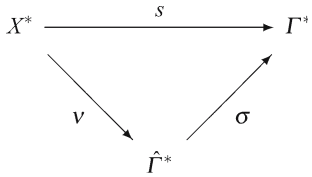



**Fig. 16** A characterization of stack transductions

initial state $q_0'$ and the transition $(q_0', \varepsilon : Z_0, q_0)$. This will ensure that all output words start with $Z_0$.

2. We modify each transition of $T$ in such way that any output label will start with a word in $\overline{\Gamma}$ (the new $\Gamma$). We do so in two steps (2 and 3). First, we modify all the transitions of $T$ as shown in Fig. 17.
3. Secondly, for each output label $\beta$ of every transition in $T$, we perform a factorization into $\beta = \beta_1 \dots \beta_k$ such that $\beta_i \in \overline{\Gamma}\Gamma^*$. In other words, every factor of $\beta$ starts with a symbol in $\overline{\Gamma}$ and that symbol is the only symbol in $\overline{\Gamma}$ which it contains. For convenience we denote each factor of $\beta$ as $\beta_i = \overline{a}_i\alpha_i$, to emphasize the letter in $\overline{\Gamma}$. We then add new states and transitions for each factor of $\beta$, as shown in Fig. 18. The newly obtained transducer realizes a transduction $\nu'$ with the same domain as $\nu$ and which differs from it only by the occurrences of $Z_0$ and $\overline{Z}_0$. Notice that this factorization is possible due to the previous transformation which ensures that each "$\beta$" starts with $\overline{Z}_0$.
4. Transform all transitions of the new finite transducer into pushdown transitions as following: a transition $(p, x, \overline{a}\alpha, q)$ becomes $(p, x, a/\alpha^R, q)$. In other words, if initially the transition was outputting $\overline{a}\alpha$ on an input $x$, it now pops $a$ from the stack and pushes $\alpha^R$.
5. We make each final state non-final. From all these states we add one extra transition to a new final state, that does nothing else but popping $Z_0$ from the stack (notice that previously, all computations were ending by pushing $Z_0$ onto the stack).

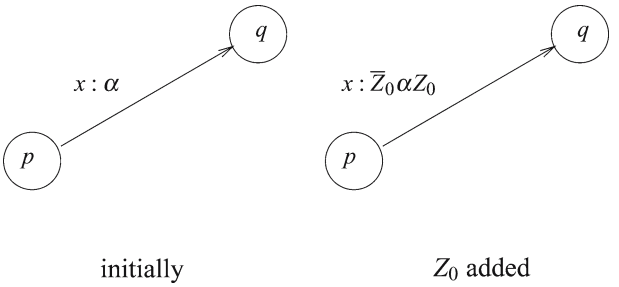



**Fig. 17** Addition of $Z_0$ and $\overline{Z}_0$ to all transitions

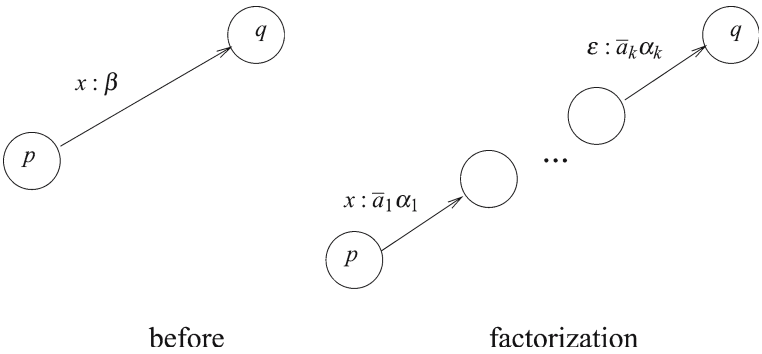



**Fig. 18** Factorization of output labels:$\beta = \overline{a}_1\alpha_1 \dots \overline{a}_k\alpha_k$

One can check now that the obtained PDA realizes $\sigma \circ \nu$ "in its stack," hence proving that $s$ is a stack transduction. □

There are still several questions about stack transductions left unexplored. For example, how are the languages of two pushdown automata which realize the same stack transduction related? We have also not investigated yet closure properties of stack transductions, and their composition with other types of transductions. Here, our main purpose is to "dissect" the pushdown automaton into its elementary components, among which stack transductions play a major role. Moreover, we can associate a stack transduction to every pushdown transducer: it is the stack transduction realized by its input automaton (the automaton obtained by ignoring the output).

Notationwise, for every pushdown transducer $T$, we denote by $T^{-1}$ the pushdown transducer obtained from $T$ by interchanging its input and output labels. For example, if in $T$ there exists a transition $(p, x, a/\alpha : y, q)$, in $T^{-1}$ there is a corresponding transition $(p, y, a/\alpha : x, q)$. We denote by $s_T$ the stack transduction of $T$ and by $s_{T^{-1}}$ the stack transduction of $T^{-1}$.

**Proposition 3** *For every pushdown transducer $T$, realizing the transduction $\tau$, the following relation holds:*

$$s_{T^{-1}} \circ \tau = s_T.$$

*Proof* The key point is to observe that every successful computation of $T$ produces a triplet $(u, \gamma, v)$, where $u$ is the input word, $v$ is the output word and $\gamma$ is the stack "residue." Moreover, $T^{-1}$ produces exactly the same triplets as $T$, with the first and the third component interchanged. The tuple $(u, \gamma)$ is in $s_T$ and the tuple $(v, \gamma)$ is in $s_{T^{-1}}$. Then, the conclusion follows shortly. □

In Fig. 19 we have related different transductions present in a pushdown transducer $T$:

- $\tau$ is the algebraic transduction realized by $T$;
- $\sigma$ is the transduction which eliminates matching stack symbols, as previously defined;
- $s_T$ is the stack transduction realized by $T$ and $s_{T^{-1}}$ is the stack transduction realized by the transducer $T^{-1}$ obtained from $T$ by interchanging the input with the output in every transition;
- $\nu_1$ is the rational transduction realized by a finite transducer obtained from $T$ as following: it has the same set of states (and same initial and final states) as $T$ and for every transition $(p, x, a/\alpha : y, q)$ in $T$ it has a transition $(p, x : \overline{a}\alpha^R, q)$ (it ignores the output of $T$)

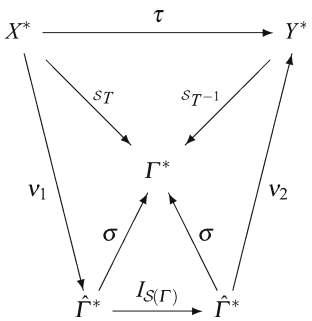



**Fig. 19** The anatomy of a pushdown transducer

- $\nu_2$ is the rational transduction realized by a finite transducer obtained from $T$ as following: it has the same set of states (and same initial and final states) as $T$ and for every transition $(p, x, a/\alpha : y, q)$ in $T$ it has a transition $(p, \overline{a}\alpha^R : y, q)$.

Then, the following relations hold (the diagram commutes):

- $\tau = \nu_2 \circ I_{\mathcal{S}(\Gamma)} \circ \nu_1$,
- $s_T = s_{T^{-1}} \circ \tau$,
- $\sigma \circ \nu_1 = s_T$ and $s_{T^{-1}} \circ \nu_2 = \sigma$.

## 6 Conclusion

In this paper we have studied several aspects of "context-freeness," as reflected in languages and transductions. We have given several representations for both context-free languages and algebraic transductions and we have succeeded in uniformizing the latter in particular cases. Finally, we have exposed "under the hood" of pushdown transducers and exhibited a special case of transductions inherently present in every pushdown machine: the stack transduction.

Perhaps one of the most significant points made in our work is to show that "heavy" algebraic results can be obtained by simply manipulating the transition labels of various machines. Consequently, we hope to have demystified a few classical results and to have produced several new, along with some useful techniques.

There are certainly several matters left for further work, and we believe, in general, that the entire topic of algebraic transductions deserves more vigorous attention from the research community. In direct relation to the results presented in this paper, we recall the open problem: it is still not clear how powerful the so-called "fair stack" pushdown automata and transducers are. Are they as powerful as general pushdown transducers? If yes, this would have several important consequences – one already mentioned. Otherwise, if they are proven to represent a strict subfamily of CF languages and algebraic transductions, then they deserve their own study. Another matter of further investigation is related to stack transductions, and a few suggestions for further work are presented previously. Finally, it is worth trying our techniques on other related topics – for example, we have already applied successfully these techniques on a study (pending) of length-preserving rational relations.

## References

1. Aho, A.V., Ullman, J.D.: The Theory of Parsing, Translation and Compiling, vol. 1. Prentice-Hall, Englewood Cliffs (1972)
2. Autebert, J.-M., Berstel, J., Boasson, L.: Context-free languages and pushdown automata. In: Salomaa, A., Rozenberg, G. (eds.) Handbook of Formal Languages, vol. 1, Word Language Grammar, pp. 111–174. Springer, Berlin Heidelberg New York (1997)
3. Berstel, J.: Transductions and Context-Free Languages. B. G. Teubner, Stuttgart (1979)
4. Choffrut, C., Culik, K.: Properties of finite and pushdown transducers. SIAM J Comput **12**(2): 300–315 (1983)
5. Cohen, A.: Program analysis and transformation: from the polytope model to formal languages. Thése de Doctorat de l'Universitè de Versailles, reported by Jean Berstel (1999)
6. Eilenberg, S.: Algèbre Catégorique et Théorie des Automates. Institut H. Poincaré, Université de Paris (1967)
7. Eilenberg, S.: Automata, Languages and Machines, vol. A. Academic, New York (1974)
8. Eilenberg, S.: Automata, Languages and Machines, vol. B. Academic, New York (1976)

9. Fliess, M.: Transductions algebriques. R.A.I.R.O., **R1**, 109–125 (1970)
10. Ginsburg, S.: The Mathematical Theory of Context-Free Languages. McGraw-Hill, New York (1966)
11. Kobayashi, K.: Classification of formal languages by functional binary transductions. Info Control, **15**(1), 95–109 (1969)
12. Nivat, M.: Transductions des langages de Chomsky. Ann Inst Fourier **18**, 339–456 (1968)
13. Sakarovitch, J.: Éléments de Théorie des Automates. Vuibert Informatique, Paris (2003)