A Chomsky-Schützenberger Theorem for Weighted Automata with Storage

Masterarbeit

vorgelegt von

Luisa Herrmann

bearbeitet vom 15. April 2015
bis zum 23. September 2015

Verantwortlicher Hochschullehrer:
Prof. Dr.-Ing. habil. Heiko Vogler
Zweitgutachter:
Prof. Dr. Christel Baier

Technische Universität Dresden
Fakultät Informatik
Institut für Theoretische Informatik
Lehrstuhl für Grundlagen der Programmierung

Aufgabenstellung für die Masterarbeit

„A Chomsky-Schützenberger Theorem for Weighted Automata with Storage“

Technische Universität Dresden
Fakultät Informatik

31. März 2015

Chomsky-Schützenberger-Theorem Das klassische Theorem von Chomsky und Schützenberger (CS-Theorem) [CS63] besagt, dass jede kontextfreie Sprache als ein homomorphes Bild des Schnitts einer Dyck-Sprache mit einer regulären Sprache dargestellt werden kann. Während in diesem Theorem die Menge Y der Symbole, die in der Dyck-Sprache auftauchen, von einer gegebenen kontextfreien Grammatik abhängt, kann Y alternativ durch einen Homomorphismus g in ein Alphabet bestehend aus nur zwei Symbolen kodiert werden. Somit erhält man das folgende CS-Theorem [Har78, Thm.10.4.3]: jede kontextfreie Sprache L kann mithilfe von zwei Homomorphismen g und h sowie einer regulären Sprache R in der Form $L = h(g^{-1}(D_2) \cap R)$ dargestellt werden, dabei bezeichnet D_2 die Dyck-Sprache über einem Alphabet mit zwei Symbolen. Diese alternative Darstellung bezeichnen wir in Folge als CS-Theorem.

Weiterhin wurde in [CS63] ein spezieller Aspekt von Gewichtung betrachtet: Jedes Wort einer Sprache wird mit der Anzahl seiner Ableitungen assoziiert. Später wurden allgemeinere Gewichtungsstrukturen untersucht, so dass CS Theoreme für gewichtete kontextfreie Sprachen über Semiringen [SS78] sowie Valuierungsmonoiden mit Identität [DV13] erhalten wurden.

Automaten mit Speicher Die Theorie von Automaten mit Speicher basiert auf Ideen von Dana Scott [Sco67], sie wurde durch Joost Engelfriet [Eng86] auf kontextfreie Grammatiken mit Speicher erweitert. Ein *Automat mit Speicher* besteht aus einer endlichen Zustandskontrolle sowie einem Speichertyp – eine Menge von Konfigurationen zusammen mit Prädikaten und Funktionen. Zu jedem Zeitpunkt einer Berechnung des Automaten befindet sich der Speicher in genau einer Konfiguration, welche für eine Transition durch ein Prädikat geprüft und durch eine Funktion auf eine Folgekonfiguration geändert wird. Ein Beispiel für einen solchen Speichertyp sind iterierte Pushdowns [Mas76].

Aufgaben Die Studentin soll Automaten mit Speicher formal einführen und auf *gewichtete Automaten mit Speicher über Valuierungsmonoiden mit Identität* (kurz: (S, Σ, K)-Automaten) erweitern. Ziel der Arbeit ist es, für diese Automaten ein Chomsky-Schützenberger-Resultat zu entwickeln. Außerdem sollen die (S, Σ, K)-Automaten, instanziiert für den Speichertyp 1-iterierter Pushdown, mit den gewichteten Pushdown-Automaten in [DV13] verglichen werden. Schließlich soll das CS-Theorem von [DV13] mit dem entsprechenden CS-Theorem dieser Arbeit zumindest informell verglichen und anhand eines Beispiels illustriert werden.

Form Die Arbeit muss den üblichen Standards wie folgt genügen: Die Arbeit muss in sich abgeschlossen sein und alle nötigen Definitionen und Referenzen enthalten. Die Struktur der Arbeit muss klar erkennbar sein, und der Leser soll gut durch die Arbeit geführt werden. Die Darstellung aller Begriffe und Verfahren soll mathematisch formal fundiert sein. Für jeden wichtigen Begriff sollen Beispiele angegeben werden, ebenso für die Abläufe und Konstruktionen der beschriebenen Verfahren. Wo es angemessen ist, sollen Illustrationen die Darstellung vervollständigen. Schließlich sollen alle Lemmata und Sätze möglichst lückenlos bewiesen werden. Die Beweise sollen leicht nachvollziehbar dokumentiert sein.

Student:	Luisa Herrmann
Studiengang:	Master Informatik
Matrikelnummer:	3658849
Verantwortlicher Hochschullehrer:	Prof. Dr.-Ing. habil. Heiko Vogler
Beginn am:	15.04.2015
Einzureichen am:	23.09.2015

Dresden, 31. März 2015

Unterschrift von Heiko Vogler

Unterschrift von Luisa Herrmann

Contents

1 Introduction

In 1963, Noam Chomsky and Marcel-Paul Schützenberger introduced a well-known characterization of context-free languages. The classical Chomsky-Schützenberger theorem [CS63, Proposition 2] (CS theorem) states that each context-free language L is the homomorphic image of the intersection of a Dyck-language D_n and a regular language R, i.e.,

$$L = h(D_n \cap R)$$

for some homomorphism h. The Dyck language is the language of all the well-bracketed expressions over some parenthesis alphabet Y. In the classical CS theorem, the number n of parenthesis pairs in Y depends on the given context-free grammar. In [Har78, Theorem 10.4.3] an alternative formulation is given where this dependency can be avoided by coding Y with a second homomorphism into a two-letter alphabet. This leads to the following theorem: each context-free language L can be represented in the form

$$L = h(g^{-1}(D_2) \cap R)$$

for some homomorphisms h and g and a regular language R.

In recent years, this result could be extended for several classes of languages. Meanwhile, there are CS theorems for string languages generated by tree-adjoining grammars [Wei88], multiple context-free languages [YKS10], yield images of simple context-free tree languages [Kan14], and indexed languages [FV15]. Also in natural language processing, this result has been applied – it is used for example in [Hul09] to specify a parser for context-free languages.

Already in the classical CS theorem a first aspect of weighting was considered. Thereby, each word of a context-free language is associated with the number of its derivations. In general, a weighted language is a function $r : \Sigma^* \to K$ which maps words over an alphabet Σ to values from some weight algebra K. Also for the weighted case a number of CS results could be obtained. In [SS78] the CS theorem was shown for weighted context-free languages over commutative semirings. In [DV13] this result was generalized to weighted context-free languages over unital valuation monoids, called quantitative context-free languages. Recently, the CS theorem has been proved for weighted multiple context-free languages over complete commutative strong bimonoids [Den15].

The aim of this work is to develop a CS theorem for the class of weighted languages recognizable by K-weighted automata with storage, where K is an arbitrary unital valuation monoid. An automaton with storage S [Eng86; GG70; Sco67][1] is a one-way nondeterministic finite-state automaton with an additional storage of type S; a successful computation starts with the initial state and an initial configuration of S; in each transition the automaton can test the current storage configuration and apply an instruction to it.

We extend the concept of automata with storage to that of K-weighted automata with storage where K is a unital valuation monoid. This weight structure was introduced in [DM10; DM11; DV13] for modelling a possibility to compute weights in a global manner, e.g., to describe the average consumption of technical systems.

Then our main result states the following (cf. Thm. 7.6). Let $r : \Sigma^* \to K$ be recognizable by some K-weighted automaton over some storage type S. Then there are a regular language R, a finite set Ω of pairs (each consisting of a predicate and an instruction), a configuration c of S, a letter-to-letter morphism g, and a (weighted) alphabetic morphism h such that

$$r = h(g^{-1}(\mathrm{B}(\Omega, c)) \cap R)$$

where $\mathrm{B}(\Omega, c)$ is the set of all Ω-behaviours of c. An Ω-behaviour can be understood as a valid storage protocol.

On the way to this result, we proceed as follows:

After introducing some elementary notions, we recall in Section 3.1 and 3.2 the concept of storage and automata with storage. As examples of storage types we consider the trivial storage type, the (1-iterated) pushdown as well as the n-iterated pushdown, whereby the corresponding automata recognize the classes of recognizable languages, context-free languages and n-iterated pushdown languages, respectively. Furthermore, we show how to express M-automata [Kam07], where M is a (multiplicative) monoid, in a straightforward way as automata with storage by introducing a new storage type $\mathrm{MON}(M)$ (cf. Example 3.2.1).

In Section 3.3 we extend automata with storage to weighted automata with storage, where we use an arbitrary unital valuation monoid K as weight algebra ((S, Σ, K)-automata). Each transition of an automaton is assigned a value of K; the weight of a computation results from applying the valuation function of K to the weights of the transitions of the computation.

In [DV13] K-weighted pushdown automata (K-WPDA), where K is again a unital valuation monoid, were used to represent K-weighted context-free languages. In Chapter 4 we compare these automata formally with our automaton model instantiated with the 1-iterated pushdown as storage type ($(\mathrm{P}^1, \Sigma, K)$-automata). Thereby, we have to handle among others the following differences: In contrast to K-WPDAs, $(\mathrm{P}^1, \Sigma, K)$-automata must not have an empty pushdown. Furthermore, $(\mathrm{P}^1, \Sigma, K)$-automata accept a word with an arbitrary final storage configuration, whereas K-WPDA only recognize with empty pushdown. Nevertheless, we can prove that K-WPDA over some alphabet Σ are expressively equivalent to $(\mathrm{P}^1, \Sigma, K)$-automata.

Our CS theorem for (S, Σ, K)-automata is presented in Chapter 7 and can be obtained by the following two steps: In Chapter 5 we separate the weights from such an automaton. That

[1] If we cite notions or definitions from [Eng86], then we always refer to the version of 2014, which is available on arXiv.

means, we decompose an (S, Σ, K)-automata into an unweighted (S, Δ)-automaton as well as a weighted alphabetic morphism. Then we proceed in Chapter 6 with separating the storage from an unweighted (S, Δ)-automaton.

Finally, in Chapter 8 we compare our CS result, instantiated with the 1-iterated pushdown, informally with the CS theorem of [DV13]. Although the proof of the CS theorem in [DV13] follows a different way, we hint that their result follows from our instantiated theorem.

Note that this thesis is based on [HV15] and expands this paper. However, the results of Chapter 4 and Chapter 8 are entirely new.

2 Preliminaries

2.1 Notations and notions

This section introduces some elementary definitions and conventions.

We assume $\mathbb{N}$ to be the set of *natural numbers*, i.e. the set of all nonnegative integers including zero. The set of all nonnegative real numbers is denoted by $\mathbb{R}_{\geq 0}$. Furthermore, by $[n]$ we denote the set $\{1,2,\ldots,n\}$ for every $n \in \mathbb{N}$. Thus, $[0] = \emptyset$. For each finite and nonempty subset A of $\mathbb{N}$ we denote by $\max(A)$ the greatest element of A. We use as a convention that $\max(\emptyset) = 0$.

Let A be a set. The *(finite) power set of* A, denoted by $\mathcal{P}(A)$ (respectively $\mathcal{P}_{\text{fin}}(A)$), is the set of all (finite) subsets of A. The *cardinality* of A will be denoted by $|A|$. We define the *set of words* of length $n \in \mathbb{N}$ over A as $A^n = \{a_1 \ldots a_n \mid a_1, \ldots, a_n \in A\}$ and for a word $w = a_1 \ldots a_n \in A^n$ and an index $i \in [n]$ we set $w(i) = a_i$. The set of words over A is defined as

$$A^* = \bigcup\nolimits_{n\in\mathbb{N}} A^n$$

and for any word $w \in A^*$, $|w|$ is the length of w. The word of length 0 is denoted by ε and is called the *empty word*. Moreover, for every $a \in A$, we denote by $|w|_a$ the number of as in w.

In the following let A, B and C be sets.
We denote the identity mapping on A by id_A. Presuming a function $f : A \to B$, for every subset $C \subseteq B$, we define its *preimage under* f as $f^{-1}(C) = \{a \in A \mid f(a) \in C\}$. Furthermore, we denote by $\mathrm{im}(f)$ the set $\{b \in B \mid \exists a \in A\colon f(a) = b\}$.

Given two sets A and I, we define a *family of elements* in A indexed by I, as a function f from I to A. Instead of f we write $(a_i \mid i \in I)$, where $a_i = f(i)$ for all indices $i \in I$. If f maps I to elements of the power set of A, $(A_i \mid i \in I)$ is called an *indexed family of sets*.

We fix a countably infinite set Λ and call its elements symbols. We call each finite subset Σ of Λ an *alphabet*. A *language over* Σ is a set $L \subseteq \Sigma^*$.

Let Σ be an alphabet and A, $B \subseteq \Sigma^*$. We define the *language concatenation* of A and B by $AB = \{ab \mid a \in A, b \in B\}$. Note that if A consists of only one element, say $A = \{a\}$, we write aB instead of AB, and analogously for B.

In the rest of this work, we let Σ and Δ denote alphabets unless specified otherwise.

2.2 Unital valuation monoids

The concept of valuation monoid was introduced in [DM10; DM11] and extended in [DV13] to unital valuation monoid. Motivated by the standpoint of modeling quantitative aspects of technical systems such as average consumption of some resource, there is a need for the ability to calculate weights in a global manner. Instead of semirings, that only handle local calculations, unital valuation monoids allow this global type of computations.

First, recall the notion of a monoid, which is an algebraic structure consisting of a set together with an associative binary operation and an identity element.

Definition 2.1. A *monoid* is a triple $(K, \cdot, 1)$, where K is a set (*carrier set*), $1 \in K$, and $\cdot \colon K \times K \to K$ is a mapping such that for all $a, b, c \in K$

- $(a \cdot b) \cdot c = a \cdot (b \cdot c)$, and
- $1 \cdot a = a \cdot 1 = a$.

A monoid $(K, \cdot, 1)$ is called *commutative* if its operation is commutative, i.e., for every $a, b \in K$ we have that $a \cdot b = b \cdot a$. Commutative monoids are often denoted $(K, +, 0)$ instead. □

In various parts of this work we can not ensure the index sets of sums to be finite. For this case we recall the notion of completeness (see, for example, [Eil74]).

Definition 2.2. A commutative monoid $(K, +, 0)$ is *complete* if it has an infinitary sum operation $\sum_I \colon K^I \to K$ for any index set I such that

- $\sum_{i \in \emptyset} a_i = 0$,
- $\sum_{i \in \{k\}} a_i = a_k$,
- $\sum_{i \in \{j,k\}} a_i = a_j + a_k$ for $j \neq k$, and
- $\sum_{j \in J} \left(\sum_{i \in I_j} a_i \right) = \sum_{i \in I} a_i$ if $\bigcup_{j \in J} I_j = I$ and $I_j \cap I_k = \emptyset$ for $j \neq k$. □

Now we extend the notion of a monoid by a valuation function as well as a unit 1 and obtain the definition of a unital valuation monoid.

Definition 2.3. A *unital valuation monoid* is a tuple $(K, +, \mathrm{val}, 0, 1)$ such that

- $(K, +, 0)$ is a commutative monoid,
- $\mathrm{val} \colon K^* \to K$ is a mapping (*valuation function*) such that for every $i \in [n]$, $a, a_1, \ldots, a_n \in K$,
  (i) $\mathrm{val}(a) = a$,
  (ii) $\mathrm{val}(a_1 \ldots a_n) = 0$ whenever $a_i = 0$,

(iii) $\mathrm{val}(a_1 \dots a_{i-1}\, 1\, a_{i+1} \dots a_n) = \mathrm{val}(a_1 \dots a_{i-1}\, a_{i+1} \dots a_n)$, and

(iv) $\mathrm{val}(\varepsilon) = 1$.

We call a unital valuation monoid $(K, +, \mathrm{val}, 0, 1)$ *complete* if $(K, +, 0)$ has this property. □

In the rest of this work, we let K denote an arbitrary unital valuation monoid $(K, +, \mathrm{val}, 0, 1)$ unless specified otherwise.

Even in the case that the carrier set of a unital valuation monoid K is denoted by, say, L, we will often refer to L by K.

There are a number of examples for unital valuation monoids. The valuation function can easily capture local, binary operations and therefore, each semiring can be simulated by a unital valuation monoid. Recall that a semiring is a structure $(K, +, \cdot, 0, 1)$, where $(K, +, 0)$ is a commutative monoid, $(K, \cdot, 1)$ is a monoid, multiplication distributes over addition, and $a \cdot 0 = 0 \cdot a = 0$ for every $a \in K$.

Example 2.4. Let $(K, +, \cdot, 0, 1)$ be a semiring. Then the structure $(K, +, \mathrm{val}, 0, 1)$, where for every $n \in \mathbb{N}$, $d_1, \dots, d_n \in K$

$$\mathrm{val}(d_1 \dots d_n) = d_1 \cdot \ldots \cdot d_n,$$

is a unital valuation monoid. In particular, we consider the *Boolean semiring* $\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$, which can be simulated by the *Boolean unital valuation monoid* $\mathbb{B} = (\{0, 1\}, \vee, \mathrm{val}, 0, 1)$, where for every $n \in \mathbb{N}$, $b_1, \dots, b_n \in \{0, 1\}$ we have $\mathrm{val}(b_1 \dots b_n) = b_1 \wedge \ldots \wedge b_n$. □

To show the ability of unital valuation monoids for calculating weights in a global manner, we define such a structure for discounting [CDH08]. The idea behind this concept is to control, depending on a discounting factor, whether the beginning or ending part of an argument word has stronger influence on the result obtained by a valuation function.

Example 2.5. Let $\lambda \in \mathbb{R}_{\geq 0}$ and $\tilde{\mathbb{R}} = \mathbb{R}_{\geq 0} \cup \{-\infty\}$. We define the unital valuation monoid $\mathcal{K}^{\lambda}_{\mathrm{disc}} = (\tilde{\mathbb{R}}, \max, \mathrm{val}^{\lambda}_{\mathrm{disc}}, -\infty, 0)$[1] such that for $n \in \mathbb{N}$, $a_1, \dots, a_n \in \tilde{\mathbb{R}}$

$$\mathrm{val}^{\lambda}_{\mathrm{disc}}(a_1 \dots a_n) = \lambda^0 a_1 + \ldots + \lambda^{n-1} a_n.$$

If we now choose $\lambda = 0.5$, we obtain as an example

$$\begin{aligned} \mathrm{val}^{\lambda}_{\mathrm{disc}}(2\ 1\ 2\ 0) &= \lambda^0 2 + \lambda^1 1 + \lambda^2 2 + \lambda^3 0 \\ &= (0.5)^0 2 + (0.5)^1 1 + (0.5)^2 2 + (0.5)^3 0 \\ &= 3\,. \end{aligned}$$ □

[1] Note that in this case max is a binary operation on $\tilde{\mathbb{R}}$. However, we also use the name max since this causes no confusion.

2.3 Weighted languages

In this section we define the notion of weighted languages and related concepts. We start with a simple example of an unweighted language that we will use frequently in the course of this work.

Example 2.6. Let $\Sigma = \{a, b\}$. The set $L = \{w \in \Sigma^* \mid |w|_a = |w|_b\}$ with words consisting of the same number of as and bs is a language over Σ. Then the word $aabbba$ is an element of L. □

Definition 2.7. A *K-weighted language (over Σ)* is a mapping of the form $r \colon \Sigma^* \to K$. We denote the set of all mappings of this type by $K\langle\langle\Sigma^*\rangle\rangle$. □

We note that we will not write the application $r(w)$ as (r, w), although the latter is customary in the theory of formal power series. In the sequel we write weighted language instead of K-weighted language if K is clear from the context.

Example 2.8. Recall the unital valuation monoid $\mathcal{K}^{\lambda}_{\text{disc}} = (\tilde{\mathbb{R}}, \max, \text{val}^{\lambda}_{\text{disc}}, -\infty, 0)$ from Example 2.5 for $\lambda = 0.5$ and let $\Sigma = \{a, b\}$. We define the weighted language $r \colon \Sigma^* \to \mathcal{K}^{\lambda}_{\text{disc}}$ with

$$r(w_1 \dots w_n) = \begin{cases} \tilde{\text{wt}}(w_1)\lambda^0 + \ldots + \tilde{\text{wt}}(w_n)\lambda^{n-1} & \text{if } |w_1 \dots w_n|_a = |w_1 \dots w_n|_b, \\ -\infty & \text{otherwise} \end{cases}$$

for every $n \in \mathbb{N}$, $w_1, \dots, w_n \in \Sigma$, where $\tilde{\text{wt}} \colon \Sigma \to \mathcal{K}^{\lambda}_{\text{disc}}$ with $\tilde{\text{wt}}(a) = 2$ and $\tilde{\text{wt}}(b) = 1$. Now consider the word $aabbba \in \Sigma^*$. We have that

$$\begin{aligned} r(aabbba) &= \tilde{\text{wt}}(a)\lambda^0 + \tilde{\text{wt}}(a)\lambda^1 + \tilde{\text{wt}}(b)\lambda^2 + \tilde{\text{wt}}(b)\lambda^3 + \tilde{\text{wt}}(b)\lambda^4 + \tilde{\text{wt}}(a)\lambda^5 \\ &= 2\lambda^0 + 2\lambda^1 + 1\lambda^2 + 1\lambda^3 + 1\lambda^4 + 2\lambda^5 \\ &= 3.5\ . \end{aligned}$$ □

At some places in this work, the K-weighted language r will itself be the result of the application of a mapping g to some element v, i.e., $r = g(v)$. Then we will write $g(v)(w)$ instead of $\big(g(v)\big)(w)$.

Definition 2.9. Let $r \in K\langle\langle\Sigma^*\rangle\rangle$. The *support of* r is the set $\{w \in \Sigma^* \mid r(w) \neq 0\}$, denoted by $\text{supp}(r)$. □

Each $L \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ determines the set $\text{supp}(L) \subseteq \Sigma^*$. Vice versa, each set $L \subseteq \Sigma^*$ determines the $\mathbb{B}$-weighted language $\chi_L \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ with $\chi_L(w) = 1$ if and only if $w \in L$. Hence, for every $L \subseteq \Sigma^*$, we have $\text{supp}(\chi_L) = L$; and for every $L \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ we have $\chi_{\text{supp}(L)} = L$. Thus, in the sequel we will not distinguish between these two points of view.

Next we define the notion of local finiteness. The intuition behind this concept is the following. A family of K-weighted languages over Σ is locally finite if each word over Σ is only in the support of a finite number of these languages.

Definition 2.10. Let I be an enumerable set. A family $(r_i \mid i \in I)$ of K-weighted languages $r_i \in K\langle\langle\Sigma^*\rangle\rangle$ is *locally finite* if for each $w \in \Sigma^*$ the set

$$I_w = \{i \in I \mid r_i(w) \neq 0\}$$

is finite. In this case or if K is complete, we define $\sum_{i\in I} r_i \in K\langle\langle\Sigma^*\rangle\rangle$ by letting $\big(\sum_{i\in I} r_i\big)(w) = \sum_{i\in I_w} r_i(w)$ for every $w \in \Sigma^*$. □

3 Weighted automata with storage

Due to the large number of upcoming new automata models in the 60s of the previous century, Dana Scott advocated in [Sco67] a homogeneous point of view on sequential programs working on machines. There, a program is a flowchart over some sets of predicate symbols and of (partial) function symbols, and a machine consists of a memory set and the interpretation of the predicate and function symbols as predicates and functions on the memory set. This viewpoint promoted the theory of abstract families of automata and their relationship to abstract families of languages (cf. e.g., [GGH69]). We take up this concept and call it finite-state automata with storage, where the finite-state automata correspond to sequential programs and storages correspond to machines. We present this concept in the style of [Eng86] (cf. [EV86; EV88] for further investigations). There, Joost Engelfriet has generalized Scott's idea from sequential programs to recursive programs, the latter being represented as context-free grammars. From that point of view, we consider regular grammars with storage in the present work. Moreover, we add weights to the transitions of the automaton, where the weights are taken from some unital valuation monoid.

3.1 Storage types

We recall the definition of storage type from [Eng86; Sco67] as a set of configurations together with predicates and instructions, which can test and modify a storage configuration, respectively. However, we use a slight modification. In contrast to the definition in [Eng86], which interprets predicate and instruction symbols with a meaning function, we immediately define sets of predicates and instruction mappings. Furthermore, motivated by the viewpoint of context-free grammars with storage as transducers, in [Eng86] an arbitrary set of input symbols and a set of encodings is defined. Thereby each encoding is a partial function from input symbols to configurations, due to the fact that different transducers could interpret the input symbols in a different way. Since we neglect this view, we use a set of input configurations instead.

Definition 3.1. A *storage type* is a tuple $S = (C, P, F, C_0)$, where C is a set (*configurations*), P is a set of total functions each having the type $p\colon C \to \{\text{true}, \text{false}\}$ (*predicates*), F is a set of partial functions each having the type $f : C \to C$ (*instructions*), and $C_0 \subseteq C$ (*initial configurations*). ◻

Throughout this work we let S denote an arbitrary storage type (C, P, F, C_0) unless specified otherwise.

In the following we give the definitions of two specific storage types that we use often in the course of this work. We start with the trivial storage type with one configuration c which satisfies the only predicate $\mathrm{p}_{\mathrm{true}}$ and is mapped by the instruction f_{id} to itself:

Definition 3.2. Let c be an arbitrary but fixed symbol. The *trivial storage type* is the storage type $\mathrm{TRIV} = (\{\mathrm{c}\}, \{\mathrm{p}_{\mathrm{true}}\}, \{\mathrm{f}_{\mathrm{id}}\}, \{\mathrm{c}\})$, where $\mathrm{p}_{\mathrm{true}}(\mathrm{c}) = \mathrm{true}$ and $\mathrm{f}_{\mathrm{id}}(\mathrm{c}) = \mathrm{c}$. □

Next we recall the pushdown operator P from [Eng86, Definition 5.1] and [EV86, Definition 3.28]: if S is a storage type, then $\mathrm{P}(S)$ is another storage type of which the configurations have the form of a pushdown; one part of each cell contains a configuration of S:

Definition 3.3. Let Γ be a fixed infinite set of *pushdown symbols*. Furthermore, let $S = (C, P, F, C_0)$ be a storage type. The *pushdown of S* is the storage type $\mathrm{P}(S) = (C', P', F', C'_0)$ where

- $C' = (\Gamma \times C)^+$ and $C'_0 = \{(\gamma_0, c_0) \mid \gamma_0 \in \Gamma, c_0 \in C_0\}$,
- $P' = \{\mathrm{bottom}\} \cup \{(\mathrm{top} = \gamma) \mid \gamma \in \Gamma\} \cup \{\mathrm{test}(p) \mid p \in P\}$ such that for every $(\delta, c) \in \Gamma \times C$ and $\alpha \in (\Gamma \times C)^*$ we have

$$\begin{aligned} \mathrm{bottom}\big((\delta, c)\alpha\big) &= \mathrm{true} \text{ if and only if } \alpha = \varepsilon \\ (\mathrm{top} = \gamma)\big((\delta, c)\alpha\big) &= \mathrm{true} \text{ if and only if } \gamma = \delta \\ \mathrm{test}(p)\big((\delta, c)\alpha\big) &= p(c) \end{aligned}$$

- $F' = \{\mathrm{pop}\} \cup \{\mathrm{stay}(\gamma) \mid \gamma \in \Gamma\} \cup \{\mathrm{push}(\gamma, f) \mid \gamma \in \Gamma, f \in F\}$ such that for every $(\delta, c) \in \Gamma \times C$ and $\alpha \in (\Gamma \times C)^*$ we have

$$\begin{aligned} \mathrm{pop}\big((\delta, c)\alpha\big) &= \alpha \text{ if and only if } \alpha \neq \varepsilon \\ \mathrm{stay}(\gamma)\big((\delta, c)\alpha\big) &= (\gamma, c)\alpha \\ \mathrm{push}(\gamma, f)\big((\delta, c)\alpha\big) &= (\gamma, f(c))(\delta, c)\alpha \text{ if } f(c) \text{ is defined} \end{aligned}$$

  and undefined in all other situations. □

The intuition behind the storage type pushdown of S is the following. A configuration consists of a nonempty sequence of pushdown cells. In contrast to classical pushdowns, each cell has two parts – one part containing a pushdown symbol and one part with a configuration of S.

Such a configuration can be tested by the three kinds of predicates defined above. The predicate bottom is true on a configuration consisting of only one pushdown cell. If the topmost pushdown symbol equals γ, then the predicate $\mathrm{top} = \gamma$ is satisfied. Furthermore,

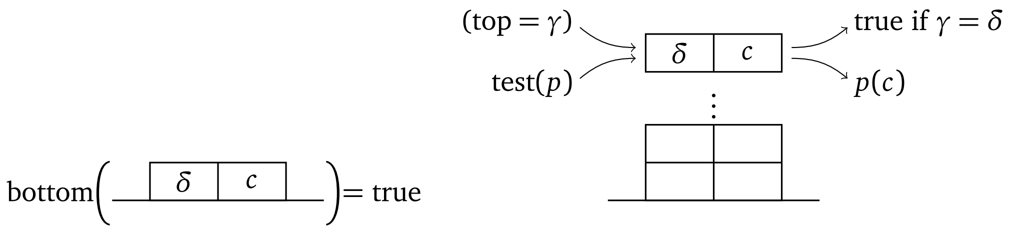


Figure 3.1: An illustration of the predicates bottom, top $= \gamma$, and test(p) applied to a configuration of P(S) for some storage type S.

test(p) equals the predicate p applied to the topmost configuration of S. This is illustrated in Figure 3.1.

Additionally, there are three possibilities to modify a pushdown configuration. By the instruction pop, the topmost cell is removed from a pushdown with at east two cells. The function stay(γ) replaces the topmost pushdown symbol by the symbol γ. By the instruction push(γ, f) a new pushdown cell is added. This cell contains as pushdown symbol γ and as configuration f applied to the configuration of the pushdown cell lying underneath. For further illustration see Figure 3.2.

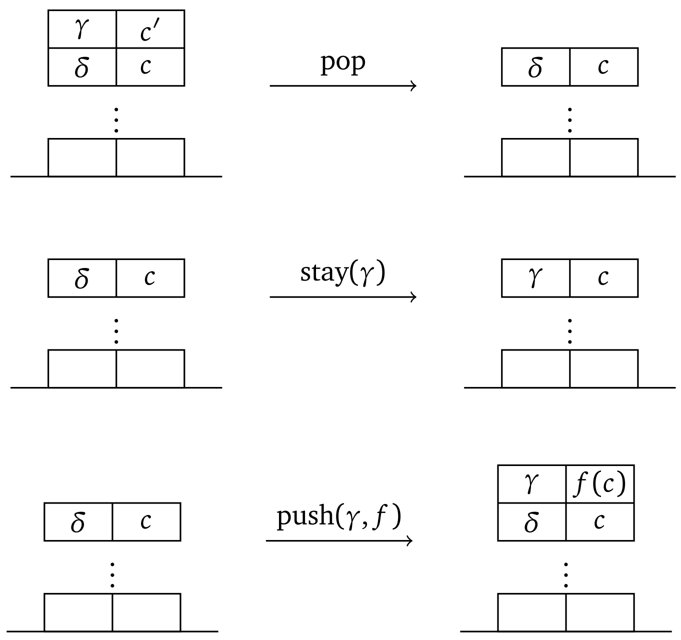


Figure 3.2: An illustration of the instructions pop, stay(γ), and push(γ, f) applied to a configuration of P(S) for some storage type S.

Intuitively, P(TRIV) corresponds to the usual pushdown storage, except that there is no empty pushdown. We can also apply P more than once to TRIV.

Definition 3.4. Let $n \geq 0$. The *n-iterated pushdown storage*, denoted by P^n, is defined inductively as follows: $\mathrm{P}^0 = \mathrm{TRIV}$ and if $n \geq 1$, then $\mathrm{P}^n = \mathrm{P}(\mathrm{P}^{n-1})$. □

Example 3.5. As an example we consider the 2-iterated pushdown storage P^2. A configuration of this storage type consists of an outer pushdown which contains in the configuration part of each pushdown cell an inner pushdown of type P^1. For an illustration of such a configuration consider Figure 3.3.

The inner pushdown of the topmost pushdown cell is accessible by “nested” predicates and instructions. Thus, for example by test(top $= \delta$) it can be tested if the topmost symbol of the inner pushdown on top of the outer pushdown is δ. Furthermore, by push(γ, pop) a new cell is pushed to the outer pushdown. This cell contains as pushdown symbol γ and as configuration the pushdown of the lying underneath pushdown cell, where the topmost cell is popped.

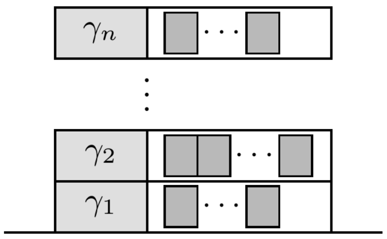


Figure 3.3: An illustration of a configuration of P^2.

□

3.2 Automata with storage

Now we combine the well-known formalism of finite-state automata with the concept of storage. The transitions of an automaton are enriched by predicates and instructions from a storage type which control the derivation together with the automaton's state control. Such an automaton has additionally as part of its internal state a configuration of its underlying storage type. A transition can be applied if its predicate succeeds on this configuration and the instruction is defined on it. With execution of such a transition its instruction is applied to the current configuration and the result is used for the next transition.

Recall that Σ is an alphabet and $S = (C, P, F, C_0)$ is a storage type.

Definition 3.6. An (S, Σ)-*automaton* is a tuple $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T)$, where Q is a finite set (*states*), Σ is an alphabet (*terminal symbols*), $c_0 \in C_0$ (*initial configuration*), $q_0 \in Q$ (*initial state*), $Q_f \subseteq Q$ (*final states*), and

$$T \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times P \times Q \times F$$

is a finite set (*transitions*). If $T \subseteq Q \times \Sigma \times P \times Q \times F$, then we call $\mathcal{A}$ *ε-free*.

An *S-automaton* is an (S, Σ)-automaton for some alphabet Σ. ☐

Whereas the configuration of a finite-state automaton consists of its current state and a remaining partial word that has to be read, we extend this notion in the context of automata with storage by a storage configuration.

Definition 3.7. Let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T)$ be an (S, Σ)-automaton. Then the set of *$\mathcal{A}$-configurations* is the set $Q \times \Sigma^* \times C$. ☐

The computation relation of $\mathcal{A}$ is the binary relation on the set of $\mathcal{A}$-configurations defined as follows.

Definition 3.8. Let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T)$ be an (S, Σ)-automaton and let $\tau = (q, x, p, q', f)$ be in T. We define the binary relation $\vdash^\tau$ on the set of $\mathcal{A}$-configurations such that for every $w \in \Sigma^*$ and $c \in C$ we have

$$(q, xw, c) \vdash^\tau (q', w, f(c))$$

if and only if $p(c)$ is true and $f(c)$ is defined. The *computation relation of $\mathcal{A}$* is the binary relation $\vdash = \bigcup_{\tau \in T} \vdash^\tau$. ☐

Then we can define the language recognized by an automaton with storage. [1]

Definition 3.9. Let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T)$ be an (S, Σ)-automaton. The *language recognized by $\mathcal{A}$* is the set

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid (q_0, w, c_0) \vdash^* (q_f, \varepsilon, c) \text{ for some } q_f \in Q_f, c \in C\}.$$ ☐

[1] Note that we write $\vdash^*$ for the reflexive, transitive closure of $\vdash$.

Definition 3.10. Let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T)$ be an (S, Σ)-automaton. A *computation* is a sequence $\theta = \tau_1 \dots \tau_n$ of transitions $\tau_i \in T$, $i \in [n]$, such that there are $\mathcal{A}$-configurations $c_0, \dots, c_n$ with $c_{i-1} \vdash^{\tau_i} c_i$, $i \in [n]$, for some $n \in \mathbb{N}$. We abbreviate this computation by $c_0 \vdash^{\theta} c_n$.

Let $q \in Q$, $w \in \Sigma^*$, and $c \in C$. A *q-computation on w and c* is a computation θ such that $(q, w, c) \vdash^{\theta} (q_f, \varepsilon, c')$ for some $q_f \in Q_f$, $c' \in C$. We denote the set of all q-computations on w and c by $\Theta_{\mathcal{A}}(q, w, c)$. Furthermore, we denote the set of all q_0-computations on w and c_0 by $\Theta_{\mathcal{A}}(w)$. □

The notion of computations of some (S, Σ)-automaton $\mathcal{A}$ gives us the possibility of defining the language recognized by $\mathcal{A}$ in an alternative way as the set

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \Theta_{\mathcal{A}}(w) \neq \emptyset\} \ .$$

Definition 3.11. Let $L \subseteq \Sigma^*$. Then L is (S, Σ)-*recognizable* if there is an (S, Σ)-automaton $\mathcal{A}$ with $L(\mathcal{A}) = L$. □

Now, after introducing the definitions for the syntax and semantics of automata with storage, it is time to illustrate this concept with an example.

Example 3.12. Let $\Sigma = \{a, b\}$. We show how to construct a (P^1, Σ)-automaton $\mathcal{A}$ which recognizes the context-free language $\{w \in \Sigma^* \mid |w|_a = |w|_b\}$. Recall that $\mathrm{P}^1 = \mathrm{P}(\mathrm{TRIV})$ with $\mathrm{TRIV} = (\{\mathrm{c}\}, \{\mathrm{p_{true}}\}, \{\mathrm{f_{id}}\}, \{\mathrm{c}\})$ and Γ is a fixed infinite set of pushdown symbols.

Now let $\mathcal{A} = (\{q, f\}, \Sigma, (\$, \mathrm{c}), q, \{f\}, T)$ with $\$ \in \Gamma$ and T contains the following transitions, where A, $B \in \Gamma$ are two symbols distinct from \$:

$$\begin{aligned} \tau_1 &= (q, a, \mathrm{bottom}, q, \mathrm{push}(B, f_{\mathrm{id}})), & \tau_4 &= (q, b, \mathrm{bottom}, q, \mathrm{push}(A, f_{\mathrm{id}})),\\ \tau_2 &= (q, a, \mathrm{top} = A, q, \mathrm{pop}), & \tau_5 &= (q, b, \mathrm{top} = B, q, \mathrm{pop}),\\ \tau_3 &= (q, a, \mathrm{top} = B, q, \mathrm{push}(B, f_{\mathrm{id}})), & \tau_6 &= (q, b, \mathrm{top} = A, q, \mathrm{push}(A, f_{\mathrm{id}})),\end{aligned}$$

$$\tau_7 = (q, \varepsilon, \mathrm{bottom}, f, \mathrm{stay}(\$)).$$

To illustrate how this automaton works, we consider the behaviour of $\mathcal{A}$ on the word $w = aabbba$. It is easy to see that $\mathcal{A}$ recognizes w with the computation $\tau_1\tau_3\tau_5\tau_5\tau_4\tau_2\tau_7 \in \Theta_{\mathcal{A}}(w)$. Starting with the initial storage configuration $(\$, \mathrm{c})$, $\mathcal{A}$ reads two times the symbol a and pushes thus two Bs on the pushdown. Intuitively, $\mathcal{A}$ counts with the number of Bs on the pushdown how much more as than bs were read. Each of this Bs is popped while reading a symbol b. After the prefix $aabb$ of w has been read the storage of $\mathcal{A}$ has its initial configuration again. Now another b is read. At this point, the storage “switches” from counting as to counting bs by pushing A. With the last symbol a this topmost A is popped. By testing bottom, the automaton checks in the last transition τ_7 if the number of as equals the number of bs that have been read and changes into the final state. This behaviour of the storage of $\mathcal{A}$ is illustrated in Figure 3.4. □

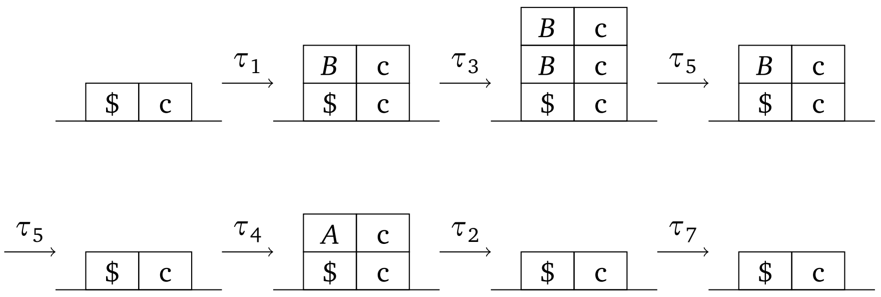


Figure 3.4: The behaviour of the storage of $\mathcal{A}$ during the computation $\tau_1\tau_3\tau_5\tau_5\tau_4\tau_2\tau_7$ on the word $aabbba$.

Given a word $w \in \Sigma^*$ there might be more than one possibility for an (S,Σ)-automaton $\mathcal{A}$ to recognize w. This observation leads to the concept of ambiguity.

Definition 3.13. Let $\mathcal{A} = (Q,\Sigma,c_0,q_0,Q_f,T)$ be an (S,Σ)-automaton. We say that $\mathcal{A}$ is *ambiguous* if there is a $w \in \Sigma^*$ such that $|\Theta_{\mathcal{A}}(w)| \geq 2$. Otherwise $\mathcal{A}$ is *unambiguous*. □

3.2.1 Instantiations of (S,Σ)-automata

In this subsection we want to consider a few instantiations of (S,Σ,K)-automata and the resulting classes of languages.

Iterated pushdown languages

If S is the trivial storage type, i.e., $S = \mathrm{TRIV}$, then the third component of each $\mathcal{A}$-configuration of a (TRIV,Σ)-automaton $\mathcal{A}$ is c and obviously the storage has no influence on $L(\mathcal{A})$. Thus, (TRIV,Σ)-automata are expressively equivalent to finite state automata over Σ.

Observation 3.14. Let $L \subseteq \Sigma^*$. Then L is recognizable by a finite state automaton if and only if L is (TRIV,Σ)-recognizable.

Moreover, we have that P^1-automata are essentially pushdown automata. The only difference lies in the fact that P^1-automata cannot have an empty pushdown. This causes no problem and can be handled appropriately (see, for example, Chapter 4).

Observation 3.15. Let $L \subseteq \Sigma^*$. Then L is recognizable by a pushdown automaton if and only if L is (P^1,Σ)-recognizable.

For each $n \geq 1$, (P^n,Σ)-automata correspond to n-iterated pushdown automata of [DG86; Eng83; Mas74; Mas76].

Observation 3.16. Let $L \subseteq \Sigma^*$. Then L is an n-iterated pushdown language if and only if L is (P^n,Σ)-recognizable for some $n \geq 1$.

Embedding of M-automata

As another example we show how to embed the concept of M-automata [Kam07], where M is a monoid, into the setting of automata with storage. First let us recall briefly the concept of M-automata from [Kam07].

Definition 3.17. Let $(M, \cdot, 1)$ be a monoid and Σ an alphabet. An *M-automaton (over Σ)* is a tuple $\mathcal{A} = (Q, \Sigma, q_0, Q_f, T)$ where Q is a finite set (*states*), $q_0 \in Q$ (*initial state*), $F \subseteq Q$ (*final states*), and T is a finite set (*transitions*); each transition has the form (q, x, q', m) where $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, and $m \in M$. □

Let $\mathcal{A} = (Q, \Sigma, q_0, Q_f, T)$ be an M-automaton for some monoid M for the rest of this section.

Definition 3.18. The *set of configurations of $\mathcal{A}$* is the set $Q \times \Sigma^* \times M$. For every transition $\tau = (q, x, q', m)$ in T we define the binary relation $\vdash^\tau \subseteq (Q \times \Sigma^* \times M) \times (Q \times \Sigma^* \times M)$ such that for every $w \in \Sigma^*$ and $m' \in M$, we have $(q, xw, m') \vdash^\tau (q', w, m' \cdot m)$. □

Definition 3.19. The *computation relation of $\mathcal{A}$* is the binary relation $\vdash = \bigcup_{\tau \in T} \vdash^\tau$. In the same way as for (S, Σ)-automata we define the concept of a computation θ. Then, for each $w \in \Sigma^*$, the set of *computations of $\mathcal{A}$ on w* is the set

$$\Theta_{\mathcal{A}}(w) = \{\theta \mid (q_0, w, 1) \vdash^\theta (q_f, \varepsilon, 1) \text{ for some } q_f \in Q_f\}. \qquad \square$$

Definition 3.20. The *language recognized by $\mathcal{A}$* is the set

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \Theta_{\mathcal{A}}(w) \neq \emptyset\}. \qquad \square$$

Example 3.21. Let $\Sigma = \{a, b\}$. We will show how to construct an M-automaton $\mathcal{A}$ which recognizes the context-free language $\{w \in \Sigma^* \mid |w|_a = |w|_b\}$. For this let $M = (\mathbb{Z}, +, 0)$, where $\mathbb{Z}$ is the set of integers. We define the M-automaton $\mathcal{A} = (\{q\}, \Sigma, q, \{q\}, T)$ with the set of transitions

$$T = \{\tau_1 = (q, a, q, 1), \tau_2 = (q, b, q, \text{-}1)\}.$$

Now consider the behaviour of $\mathcal{A}$ on the word $w = aabbba$. It is easy to see that $\mathcal{A}$ recognizes w with the computation

$$\begin{aligned}(q, aabbba, 0) \vdash^{\tau_1} (q, abbba, 0+1) \vdash^{\tau_1} (q, bbba, 1+1) \vdash^{\tau_2} (q, bba, 2 + (\text{-}1)) \\ \vdash^{\tau_2} (q, ba, 1 + (\text{-}1)) \vdash^{\tau_2} (q, a, 0 + (\text{-}1)) \vdash^{\tau_1} (q, \varepsilon, (\text{-}1) + 1)\ .\end{aligned} \qquad \square$$

For the embedding of M-automata into automata with storage, we first define a new storage type.

Definition 3.22. Let $(M, \cdot, 1)$ be a monoid. We define the storage type *monoid M*, denoted by $\mathrm{MON}(M)$, by (C, P, F, C_0), where

- $C = M$ and $C_0 = \{1\}$,
- $P = \{\text{true?}\} \cup \{1?\}$ where $\text{true?}(m) = \text{true}$, and $1?(m) = \text{true}$ if and only if $m = 1$,
- $F = \{[m] \mid m \in M\}$ where $[m]\colon M \to M$ is defined by $[m](m') = m' \cdot m$ for every $m,\, m' \in M$. □

Now we can show that each language $L \subseteq \Sigma^*$ which is recognizable by an M-automaton is also recognizable by a $\text{MON}(M)$-automaton over Σ.

Lemma 3.23. Let $L \subseteq \Sigma^*$ be a languages. If L is recognizable by an M-automaton over Σ, then L is $(\text{MON}(M), \Sigma)$-recognizable.

Proof. Let $\mathcal{A} = (Q, \Sigma, q_0, Q_f, T)$ be an M-automaton. Then we construct the $(\text{MON}(M), \Sigma)$-automaton $\mathcal{B} = (Q', \Sigma, 1, q_0, \{q_f\}, T')$ where $Q' = Q \cup \{q_f\}$ with some new state $q_f \notin Q$ and T' is defined as follows:

- If $(q, x, q', m) \in T$, then $(q, x, \text{true?}, q', [m]) \in T'$, and
- for each $q \in Q_f$, the transition $(q, \varepsilon, 1?, q_f, [1])$ is in T'.

Let $w \in \Sigma^*$ such that $w = u_1 \dots u_n$ for some $n \in \mathbb{N}$ and with $u_i \in \Sigma$ for $i \in [n]$. Let $\theta = \tau_1 \dots \tau_m$ with $\tau_1, \dots, \tau_m \in T$ be a computation in $\Theta_{\mathcal{A}}(w)$ for some $m \geq n$. Then we construct the q_0-computation $\theta' = \tau'_1 \dots \tau'_{m+1}$, $\tau'_1, \dots, \tau'_{m+1} \in T'$, in $\Theta_{\mathcal{B}}(w)$ as follows:

- Let $1 \leq i \leq m$. If $\tau_i = (q_{i-1}, x_i, q_i, m_i)$, then $\tau'_i = (q_{i-1}, x_i, \text{true?}, q_i, [m_i])$.
- Furthermore, $\tau'_{m+1} = (q_m, \varepsilon, 1?, q_f, [1])$.

Conversely, for every q_0-computation $\theta' = \tau'_1 \dots \tau'_{m+1}$ in $\Theta_{\mathcal{B}}(w)$, by definition of T', there is a uniquely determined computation $\theta = \tau_1 \dots \tau_m$ in $\Theta_{\mathcal{A}}(w)$ such that θ' is the computation constructed above. Thus, for each $w \in \Sigma^*$, the set $\Theta_{\mathcal{A}}(w)$ of computations of $\mathcal{A}$ on w and the set $\Theta_{\mathcal{B}}(w)$ of q_0-computations on w and 1 are in a one-to-one correspondence, and hence

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \Theta_{\mathcal{A}}(w) \neq \emptyset\} = \{w \in \Sigma^* \mid \Theta_{\mathcal{B}}(w) \neq \emptyset\} = L(\mathcal{B}). \qquad \blacksquare$$

Example 3.24. Recall the M-automaton $\mathcal{A} = (\{q\}, \Sigma, q, \{q\}, T)$ as well as the monoid $M = (\mathbb{Z}, +, 0)$ from Example 3.21. We construct an $(\text{MON}(M), \Sigma)$-automaton $\mathcal{B}$ such that $L(\mathcal{A}) = L(\mathcal{B})$. For this we define $\mathcal{B} = (\{q, f\}, \Sigma, 0, q, \{f\}, T')$ with the set of transitions

$$T' = \{\tau_1 = (q, a, \text{true?}, q, [1]), \tau_2 = (q, b, \text{true?}, q, [\text{-}1]), \tau_f = (q, \varepsilon, 0?, q, [0])\}.$$

Now consider the behaviour of $\mathcal{B}$ on the word $w = aabbba$. It is easy to see that $\mathcal{B}$ recognizes w with the computation $\tau_1\tau_1\tau_2\tau_2\tau_2\tau_1\tau_f$. □

3.3 Weighted automata with storage

Next we define the weighted version of (S,Σ)-automata. For this recall that $(K,+,\mathrm{val},0,1)$ is a unital valuation monoid. As usual, we combine an unweighted (S,Σ)-automaton with a weight function, which assigns to each transition a value from K. The line of our definitions follows the definition of weighted pushdown automata in [DV13].

Definition 3.25. An (S,Σ)-*automaton with weights in* K is a tuple $\mathcal{A} = (Q,\Sigma,c_0,q_0,Q_f,T,\mathrm{wt})$ where (Q,Σ,c_0,q_0,Q_f,T) is an (S,Σ)-automaton (*underlying* (S,Σ)-*automaton*) and $\mathrm{wt}\colon T \to K$ is a mapping (*weight assignment*).

If the underlying (S,Σ)-automaton is ε-free, then we call $\mathcal{A}$ *ε-free*. □

Note that if we write in the following "the transition τ is of weight a", then we mean $\mathrm{wt}(\tau) = a$.

The computations of $\mathcal{A}$ are those of its underlying (S,Σ)-automaton, and the sets $\Theta_{\mathcal{A}}(q,w,c)$ and $\Theta_{\mathcal{A}}(w)$, where $q \in Q$, $w \in \Sigma^*$ and $c \in C$, are defined analogously. Additionally, the weights that are assigned to the transitions of a computation, are evaluated by the valuation function of K.

Definition 3.26. Let $\mathcal{A} = (Q,\Sigma,c_0,q_0,Q_f,T,\mathrm{wt})$ be an (S,Σ)-automaton with weights in K. Furthermore, let $\theta = \tau_1 \dots \tau_n$ be a computation of $\mathcal{A}$ for some $n \in \mathbb{N}$, $\tau_1,\dots,\tau_n \in T$. The *weight of* θ is the element $\mathrm{wt}(\theta)$ in K defined by

$$\mathrm{wt}(\theta) = \mathrm{val}(\mathrm{wt}(\tau_1)\dots\mathrm{wt}(\tau_n)) \ . \qquad \square$$

To determine the weight of a word $w \in \Sigma^*$, we have to consider all computations on w in $\mathcal{A}$. Therefore we have to require some finiteness properties for $\mathcal{A}$, namely, that there are only finitely many computations on w for the case that K is not complete.

Definition 3.27. An *(S,Σ,K)-automaton* is an (S,Σ)-automaton $\mathcal{A}$ with weights in K such that

(i) $\Theta_{\mathcal{A}}(w)$ is finite for every $w \in \Sigma^*$, or

(ii) K is complete. □

Then we can define that an (S,Σ,K)-automaton $\mathcal{A}$ associates to every word $w \in \Sigma^*$ the sum of the weights of all computations on w.

Definition 3.28. Let $\mathcal{A}$ be an (S,Σ,K)-automaton. The *weighted language recognized by* $\mathcal{A}$ is the K-weighted language $\|\mathcal{A}\|\colon \Sigma^* \to K$ defined for every $w \in \Sigma^*$ by

$$\|\mathcal{A}\|(w) = \textstyle\sum_{\theta \in \Theta_{\mathcal{A}}(w)} \mathrm{wt}(\theta) \ . \qquad \square$$

Definition 3.29. A weighted language $r\colon \Sigma^* \to K$ is (S,Σ,K)-*recognizable* if there is an (S,Σ,K)-automaton $\mathcal{A}$ such that $r = \|\mathcal{A}\|$. □

Definition 3.30. Two (S,Σ,K)-automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are *equivalent* if $\|\mathcal{A}_1\| = \|\mathcal{A}_2\|$. □

At this point we want to take up the (S,Σ)-automaton from Example 3.12 and expand it by weights. So far this automaton recognized words over Σ with the same number of as and bs. Now additionally the discounting of Example 2.5 is applied on these words. Intuitively the thereby obtained weighted language orders words of the same length by the positions of the contained symbols. A word is assigned a greater value the more as are located at the beginning of the word. Words with an unequal number of as and bs are mapped to $-\infty$.

Example 3.31. Recall the unital valuation monoid $\mathcal{K}^{\lambda}_{\text{disc}} = (\tilde{\mathbb{R}}, \max, \text{val}^{\lambda}_{\text{disc}}, -\infty, 0)$ from Example 2.5 for $\lambda = 0.5$ and let $\Sigma = \{a, b\}$. We define the $(\mathrm{P}^1, \Sigma, \mathcal{K}^{\lambda}_{\text{disc}})$-automaton $\mathcal{A} = (\{q, f\}, \Sigma, (\$, \mathrm{c}), q, \{f\}, T, \text{wt})$ with $\$ \in \Gamma$ and T contains the following transitions, where $A, B \in \Gamma$ are two symbols distinct from \$:

$$\begin{aligned}
\tau_1 &= (q, a, \text{bottom}, q, \text{push}(B, f_{\text{id}})), & \tau_4 &= (q, b, \text{bottom}, q, \text{push}(A, f_{\text{id}})),\\
\tau_2 &= (q, a, \text{top} = A, q, \text{pop}), & \tau_5 &= (q, b, \text{top} = B, q, \text{pop}),\\
\tau_3 &= (q, a, \text{top} = B, q, \text{push}(B, f_{\text{id}})), & \tau_6 &= (q, b, \text{top} = A, q, \text{push}(A, f_{\text{id}})),
\end{aligned}$$

$$\tau_7 = (q, \varepsilon, \text{bottom}, f, \text{stay}(\$)).$$

Moreover, for each $\tau \in T$ of the form (q, x, p, q', f), $\text{wt}(\tau)$ is defined by

$$\text{wt}(\tau) = \begin{cases} 2 & \text{if } x = a,\\ 1 & \text{if } x = b, \text{ and}\\ 0 & \text{otherwise .}\end{cases}$$

Now, again consider the computation $\theta = \tau_1\tau_3\tau_5\tau_5\tau_4\tau_2\tau_7$ of $\mathcal{A}$ on the word $w = aabbba$. We have that

$$\begin{aligned}
\text{wt}(\theta) &= \text{val}^{\lambda}_{\text{disc}}(\text{wt}(\tau_1)\,\text{wt}(\tau_3)\,\text{wt}(\tau_5)\,\text{wt}(\tau_5)\,\text{wt}(\tau_4)\,\text{wt}(\tau_2)\,\text{wt}(\tau_7))\\
&= \text{val}^{\lambda}_{\text{disc}}(2211120)\\
&= 2\lambda^0 + 2\lambda^1 + 1\lambda^2 + 1\lambda^3 + 1\lambda^4 + 2\lambda^5 + 0\lambda^6\\
&= 3.5\ .
\end{aligned}$$

Note that also ε is in the support of $\mathcal{A}$ since $\tau_7 \in \Theta_{\mathcal{A}}(\varepsilon)$ and $\text{wt}(\tau_7) = 0$, which is the 1 element of $\mathcal{K}^{\lambda}_{\text{disc}}$.

It is easy to see that $\|\mathcal{A}\|$ equals the weighted language $r\colon \Sigma^* \to \mathcal{K}^{\lambda}_{\text{disc}}$ with

$$r(w_1 \dots w_n) = \begin{cases} \tilde{\text{wt}}(w_1)\lambda^0 + \ldots + \tilde{\text{wt}}(w_n)\lambda^{n-1} & \text{if } |w_1 \dots w_n|_a = |w_1 \dots w_n|_b,\\ -\infty & \text{otherwise,}\end{cases}$$

for every $n \in \mathbb{N}$, $w_1, \ldots, w_n \in \Sigma$, where $\tilde{\text{wt}}\colon \Sigma \to \mathcal{K}^{\lambda}_{\text{disc}}$ with $\tilde{\text{wt}}(a) = 2$ and $\tilde{\text{wt}}(b) = 1$. □

3.3.1 Instantiations of (S, Σ, K)-automata

Again we want to consider a few instantiations of (S, Σ, K)-automata and the resulting classes of languages.

Each $(S, \Sigma, \mathbb{B})$-automaton $\mathcal{A}$ can be considered as an (S, Σ)-automaton which recognizes $\mathrm{supp}(\|\mathcal{A}\|)$; note that $\mathbb{B}$ can be extended to a complete unital valuation monoid.

If S is the trivial storage type, i.e., $S = \mathrm{TRIV}$, then the third component of each $\mathcal{A}$-configuration is c and obviously the storage has no influence on $\|\mathcal{A}\|$. Thus, we drop S from all the denotations and simply write (Σ, K)-automaton instead of $(\mathrm{TRIV}, \Sigma, K)$-automaton. It is obvious that, apart from ε-moves, (Σ, K)-automata are same as weighted finite automata over Σ and the valuation monoid K (as, e.g., in [DM10; DM11; DV13]).

Observation 3.32. Let $r : \Sigma^* \to K$. Then r is recognizable by a weighted finite state automaton over Σ and K if and only if r is $(\mathrm{TRIV}, \Sigma, K)$-recognizable.

The $(\mathrm{P}^1, \Sigma, K)$-automata are essentially the same as weighted pushdown automata over Σ and K [DV13]. For this we refer to Theorem 4.9 in the next chapter.

Furthermore, we also want to consider the iterated pushdown storage type in the weighted case. For this we introduce the class of weighted iterated pushdown languages.

Definition 3.33. For $n \geq 0$, a *weighted n-iterated pushdown language over Σ and K* is a $(\mathrm{P}^n, \Sigma, K)$-recognizable weighted language. □

4 Comparison of weighted pushdown automata and $(\mathrm{P}^1, \Sigma, K)$-automata

Let us now compare weighted pushdown automata (WPDA) defined in [DV13] with $(\mathrm{P}^1, \Sigma, K)$-automata. For this we first recall the concept of WPDA over Σ and K, but in a form which uses the definitions and notions of the current work. Therefore, we define a pushdown storage type, which is able to simulate the pushdown part of a WPDA and afterwards, we give a slightly modified semantics for our automaton model.

Recall that Γ is a fixed infinite set of pushdown symbols.

Definition 4.1. The *pushdown storage type* is the storage type $\mathrm{PD} = (C, P, F, C_0)$ where

- $C = \Gamma^*$ and $C_0 = \Gamma$,
- $P = \{\gamma? \mid \gamma \in \Gamma\}$ such that for every $\alpha \in \Gamma^*$ we have
$$\gamma?(\alpha) = \begin{cases} \text{true} & \text{iff } \alpha = \gamma\alpha' \text{ for some } \alpha' \in \Gamma^* \\ \text{false} & \text{otherwise} \end{cases}$$
- $F = \{\pi! \mid \pi \in \Gamma^*\}$ such that for every $\alpha \in \Gamma^*$ we have
$$\pi!(\alpha) = \pi\alpha' \quad \text{if } \alpha = \gamma\alpha' \text{ for some } \gamma \in \Gamma \text{ and } \alpha' \in \Gamma^*$$
and undefined in all other situations. □

Note that the instructions of PD cover all instructions of P^1: the instruction $\varepsilon!$ equals pop, the instruction $\gamma!$ for some $\gamma \in \Gamma$ equals $\mathrm{stay}(\gamma)$, and the instruction $\delta\gamma!$ with $\delta, \gamma \in \Gamma$ covers $\mathrm{push}(\delta, f_{\mathrm{id}})$ assuming that γ is the current topmost pushdown symbol.

As the weighted pushdown automata defined in [DV13] accept with empty pushdown, we introduce the language recognized by a (PD, Σ, K)-automaton with empty pushdown. For this we introduce the concept of (q, ε)-computations, which ensure that a computation ends with the storage configuration ε.

Definition 4.2. Let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T, \mathrm{wt})$ be a (PD, Σ, K)-automaton. Moreover, let $q \in Q, w \in \Sigma^*$ and $c \in \Gamma^*$. A (q, ε)-*computation on w and c* is a computation θ such that
$$(q, w, c) \vdash^{\theta} (q_f, \varepsilon, \varepsilon)$$

for some $q_f \in Q_f$.

We denote the set of all (q, ε)-computations on w and c by $\Theta^{\varepsilon}_{\mathcal{A}}(q, w, c)$. Furthermore, we denote the set of all (q_0, ε)-computations on w and c_0 by $\Theta^{\varepsilon}_{\mathcal{A}}(w)$. □

Definition 4.3. Let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T, \mathrm{wt})$ be a (PD, Σ, K)-automaton. The *weighted language recognized by $\mathcal{A}$ with empty pushdown* is the K-weighted language $\|\mathcal{A}\|_\varepsilon \colon \Sigma^* \to K$ defined for every $w \in \Sigma^*$ by

$$\|\mathcal{A}\|_\varepsilon(w) = \sum_{\theta \in \Theta^{\varepsilon}_{\mathcal{A}}(w)} \mathrm{wt}(\theta) \ .$$ □

Note that the sum is well-defined, because $\mathcal{A}$ is a (PD, Σ, K)-automaton and $\Theta^{\varepsilon}_{\mathcal{A}}(w) \subseteq \Theta_{\mathcal{A}}(w)$.

Definition 4.4. Let $r \colon \Sigma^* \to K$ be a weighted language. Then r is *recognizable by a* (PD, Σ, K)*-automaton with empty pushdown* if there is a (PD, Σ, K)-automaton $\mathcal{A}$ such that $r = \|\mathcal{A}\|_\varepsilon$. □

It is easy to see that (PD, Σ, K)-automata which recognize with empty pushdown are the same as WPDA over Σ and K, since the storage type PD captures the storage behaviour of WPDA and our semantics in Definition 4.3 is exactly the semantics of WPDA defined in [DV13, page 208]. This leads to the following observation.

Observation 4.5. Let $r \colon \Sigma^* \to K$ be a weighted language. Then r is recognizable by a (PD, Σ, K)-automaton with empty pushdown if and only if r is the quantitative behaviour of a WPDA as defined in [DV13].

Now, before we compare (PD, Σ, K)-automata with $(\mathrm{P}^1, \Sigma, K)$-automata, we need a possibility to collate the weights of computations of such two automata without applying the valuation function val of K. For this we use the fact that all sequences over K which differ only in additional 1 elements are mapped by val to the same value from K.

Definition 4.6. Let $\mathcal{A}_1 = (Q_1, \Sigma, c_{0,1}, q_{0,1}, Q_{f,1}, T_1, \mathrm{wt}_1)$ be a $(\mathrm{P}^1, \Sigma, K)$-automaton and let $\mathcal{A}_2 = (Q_2, \Sigma, c_{0,2}, q_{0,2}, Q_{f,2}, T_2, \mathrm{wt}_2)$ be a (PD, Σ, K)-automaton. We define the mapping $\overline{\mathrm{wt}}_{\mathcal{A}_1,\mathcal{A}_2} \colon T_1 \cup T_2 \to K \cup \{\varepsilon\}$ such that for every $\tau \in T_1 \cup T_2$

$$\overline{\mathrm{wt}}_{\mathcal{A}_1,\mathcal{A}_2}(\tau) = \begin{cases} \mathrm{wt}_1(\tau) & \text{if } \tau \in T_1 \text{ and } \mathrm{wt}_1(\tau) \neq 1 \\ \mathrm{wt}_2(\tau) & \text{if } \tau \in T_2 \text{ and } \mathrm{wt}_2(\tau) \neq 1 \\ \varepsilon & \text{otherwise.} \end{cases}$$

This mapping is well-defined since $T_1 \cap T_2 = \emptyset$. Note that we write $\overline{\mathrm{wt}}$ instead of $\overline{\mathrm{wt}}_{\mathcal{A}_1,\mathcal{A}_2}$ if $\mathcal{A}_1$ and $\mathcal{A}_2$ are clear from the context. This mapping can be extended to a mapping $\overline{\mathrm{wt}}' \colon T_1^* \cup T_2^* \to K^*$ such that $\overline{\mathrm{wt}}'(\varepsilon) = \varepsilon$, and for every $n \in \mathbb{N}$, $\tau_1 \dots \tau_n \in T_1^* \cup T_2^*$ with $\tau_1, \dots, \tau_n \in T_1 \cup T_2$ we have $\overline{\mathrm{wt}}'(\tau_1 \dots \tau_n) = \overline{\mathrm{wt}}(\tau_1) \dots \overline{\mathrm{wt}}(\tau_n)$. In the following, we identify $\overline{\mathrm{wt}}$ and $\overline{\mathrm{wt}}'$. □

The next lemma proves that each two computations which result in the same sequence under $\overline{\mathrm{wt}}$ are mapped to the same weight in K by the valuation function.

Lemma 4.7. Let $\mathcal{A}_1$ be a $(\mathrm{P}^1, \Sigma, K)$-automaton and let $\mathcal{A}_2$ be a (PD, Σ, K)-automaton. Furthermore, let $w \in \Sigma^*$, $\theta_1 \in \Theta_{\mathcal{A}_1}(w)$, and $\theta_2 \in \Theta^{\varepsilon}_{\mathcal{A}_2}(w)$. If $\overline{\mathrm{wt}}(\theta_1) = \overline{\mathrm{wt}}(\theta_2)$, then $\mathrm{wt}_1(\theta_1) = \mathrm{wt}_2(\theta_2)$.

Proof. Let $\mathcal{A}_1 = (Q_1, \Sigma, c_{0,1}, q_{0,1}, Q_{f,1}, T_1, \mathrm{wt}_1)$ be a $(\mathrm{P}^1, \Sigma, K)$-automaton and let $\mathcal{A}_2 = (Q_2, \Sigma, c_{0,2}, q_{0,2}, Q_{f,2}, T_2, \mathrm{wt}_2)$ be a (PD, Σ, K)-automaton. Furthermore, let $w \in \Sigma^*$, $n, m \in \mathbb{N}$, $\theta_1 = \tau_1 \dots \tau_n \in \Theta_{\mathcal{A}_1}(w)$, $\tau_1, \dots, \tau_n \in T_1$, and $\theta_2 = \tau'_1 \dots \tau'_m \in \Theta^{\varepsilon}_{\mathcal{A}_2}(w)$, $\tau'_1, \dots, \tau'_m \in T_2$, such that $\overline{\mathrm{wt}}(\theta_1) = \overline{\mathrm{wt}}(\theta_2)$. Then

$$\begin{aligned}
\mathrm{wt}_1(\theta_1) &= \mathrm{val}(\mathrm{wt}_1(\tau_1) \dots \mathrm{wt}_1(\tau_n)) && \\
&= \mathrm{val}(\overline{\mathrm{wt}}(\tau_1 \dots \tau_n)) && (*)\\
&= \mathrm{val}(\overline{\mathrm{wt}}(\tau'_1 \dots \tau'_m)) && (\text{since } \overline{\mathrm{wt}}(\theta_1) = \overline{\mathrm{wt}}(\theta_2))\\
&= \mathrm{val}(\mathrm{wt}_2(\tau'_1) \dots \mathrm{wt}_2(\tau'_m)) && (*)\\
&= \mathrm{wt}_2(\theta_2),
\end{aligned}$$

where $(*)$ holds by property (iii) in the definition of unital valuation monoids, which states that each “1” can be ignored. ■

If we compare non-empty pushdown configurations from P^1 and PD, we can observe that the only difference lies in the fact that configurations from P^1 consist of a second component. Consider for example

$$(\gamma_1, \mathrm{c})(\gamma_2, \mathrm{c})(\gamma_3, \mathrm{c}) \text{ from } \mathrm{P}^1 \quad \text{and} \quad \gamma_1\gamma_2\gamma_3 \text{ from PD}$$

for γ_1, γ_2, $\gamma_3 \in \Gamma$. It is easy to see that this second component gives a configuration no additional information since it consists always of the only configuration c of TRIV. But on the contrary in some situations it is easier to prove a property, if we neglect this component. For this reason we introduce the following abbreviation.

Definition 4.8. Let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T, \mathrm{wt})$ be a $(\mathrm{P}^1, \Sigma, K)$-automaton. Furthermore, let $c_0 = (\gamma_0, \mathrm{c})$ for some $\gamma_0 \in \Gamma$, and let $w \in \Sigma^*$ and $q \in Q$. For every $n \geq 1$, $\eta = \gamma_1 \dots \gamma_n$ with $\gamma_1, \dots, \gamma_n \in \Gamma$ we denote by $\overline{\Theta}_{\mathcal{A}}(q, w, \eta)$ the set $\Theta_{\mathcal{A}}(q, w, \eta')$, where $\eta' = (\gamma_1, \mathrm{c}) \dots (\gamma_n, \mathrm{c})$.

Trivially, we have that $\overline{\Theta}_{\mathcal{A}}(q, w, \gamma_0) = \Theta_{\mathcal{A}}(q, w, c_0)$. □

The main issue of this chapter is to prove that (PD, Σ, K)-automata accepting with empty pushdown and $(\mathrm{P}^1, \Sigma, K)$-automata are expressively equivalent. This is stated by the following theorem.

Theorem 4.9. Let $r \colon \Sigma^* \to K$ be a weighted language. Then r is recognizable by a (PD, Σ, K)-automaton with empty pushdown if and only if r is $(\mathrm{P}^1, \Sigma, K)$-recognizable.

This result follows from the subsequent Lemmas 4.10 and 4.25 which we will prove in Section 4.1 and 4.2, respectively.

4.1 $(\mathrm{P}^1, \Sigma, K)$-automata can be simulated by (PD, Σ, K)-automata

The aim of this section is to show formally that each $(\mathrm{P}^1, \Sigma, K)$-automaton can be simulated by some (PD, Σ, K)-automaton accepting with empty storage. That means we will show that for each $(\mathrm{P}^1, \Sigma, K)$-automaton $\mathcal{A}$ there exists a (PD, Σ, K)-automaton $\mathcal{B}$ such that $\|\mathcal{A}\| = \|\mathcal{B}\|_\varepsilon$, which is stated by the following lemma.

Lemma 4.10. Let $r\colon \Sigma^* \to K$ be a weighted language. If r is $(\mathrm{P}^1, \Sigma, K)$-recognizable, then r is (PD, Σ, K)-recognizable with empty pushdown.

To prove this Lemma, we proceed in several steps, about which we would like to give a brief overview here. First, we will introduce a normal form for $(\mathrm{P}^1, \Sigma, K)$-automata, which we call test normal form. Based on a $(\mathrm{P}^1, \Sigma, K)$-automaton $\mathcal{A}$ in test normal form we will subsequently define a (PD, Σ, K)-automaton $\mathcal{B}$ that is induced by $\mathcal{A}$. To show that $\|\mathcal{A}\| = \|\mathcal{B}\|_\varepsilon$, we will indicate a bijection between the computations of $\mathcal{A}$ and $\mathcal{B}$ by specifying a function f between these two sets and proving that f is surjective, injective, and weight preserving.

Recall that $\mathrm{P}^1 = \mathrm{P}(\mathrm{TRIV})$ with $\mathrm{TRIV} = (\{\mathrm{c}\}, \{\mathrm{p}_{\mathrm{true}}\}, \{\mathrm{f}_{\mathrm{id}}\}, \{\mathrm{c}\})$. Since the only predicate $\mathrm{p}_{\mathrm{true}}$ of TRIV is true on c, we have that the predicate $\mathrm{test}(\mathrm{p}_{\mathrm{true}})$ is successful on each storage configuration of P^1. Therefore, $\mathrm{test}(\mathrm{p}_{\mathrm{true}})$ is superfluous in the context of $(\mathrm{P}^1, \Sigma, K)$-automata as each transition containing this predicate can be replaced by transitions which test each possible top of the pushdown. For this reason, we can introduce a normal form for $(\mathrm{P}^1, \Sigma, K)$-automata in which the predicate $\mathrm{test}(\mathrm{p}_{\mathrm{true}})$ is avoided.

Definition 4.11. Let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T, \mathrm{wt})$ be a $(\mathrm{P}^1, \Sigma, K)$-automaton. We say that $\mathcal{A}$ is in *test normal form* if for every transition $(q, x, p, q', f) \in T$ we have that $p \neq \mathrm{test}(\mathrm{p}_{\mathrm{true}})$. □

Lemma 4.12. For every $(\mathrm{P}^1, \Sigma, K)$-automaton $\mathcal{A}$ there is a $(\mathrm{P}^1, \Sigma, K)$-automaton $\mathcal{A}'$ in test normal form such that $\|\mathcal{A}\| = \|\mathcal{A}'\|$.

Proof. Let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T, \mathrm{wt})$ be a $(\mathrm{P}^1, \Sigma, K)$-automaton. Recall that $c_0 = (\gamma_0, \mathrm{c})$ for some $\gamma_0 \in \Gamma$ and let $\Gamma_{\mathcal{A}} = \{\gamma_0\} \cup X$, where X is the finite set of pushdown symbols occurring in transitions of $\mathcal{A}$. We construct a $(\mathrm{P}^1, \Sigma, K)$-automaton $\mathcal{A}'$ which has no transitions containing $\mathrm{test}(\mathrm{p}_{\mathrm{true}})$ as predicate and such that $\|\mathcal{A}'\| = \|\mathcal{A}\|$. For this we define $\mathcal{A}' = (Q, \Sigma, c_0, q_0, Q_f, T_{\mathcal{A}'}, \mathrm{wt}_{\mathcal{A}'})$ as follows. Let $\tau = (q, x, p, q', f)$ be in T. If $p \neq \mathrm{test}(\mathrm{p}_{\mathrm{true}})$, then τ is in $T_{\mathcal{A}'}$ and $\mathrm{wt}_{\mathcal{A}'}(\tau) = \mathrm{wt}(\tau)$. If $p = \mathrm{test}(\mathrm{p}_{\mathrm{true}})$, then for each $\gamma \in \Gamma_{\mathcal{A}}$ the transition $\tau' = (q, x, \mathrm{top} = \gamma, q', f)$ is in $T_{\mathcal{A}'}$ and $\mathrm{wt}_{\mathcal{A}'}(\tau') = \mathrm{wt}(\tau)$.

As the predicate $\mathrm{test}(\mathrm{p}_{\mathrm{true}})$ is true on every configuration of P^1 and we copy transitions containing this predicate for every possible topmost pushdown symbol, it is easy to see that $\|\mathcal{A}'\| = \|\mathcal{A}\|$. ■

Let $\mathcal{A}$ be a $(\mathrm{P}^1, \Sigma, K)$-automaton in test normal form. In a second step we construct a (PD, Σ, K)-automaton $\mathcal{B}$ that is induced by $\mathcal{A}$. Therefore we have to take care of differences between (P^1,Σ,K)-automata and (PD,Σ,K)-automata concerning the handling of pop on a pushdown which contains exactly one symbol. Since P^1 does not have ε as configuration, any attempt of a (P^1,Σ,K)-automaton to apply pop to a single-symbol pushdown blocks the computation. In contrast, a (PD,Σ,K)-automaton can very well apply a pop in such a situation, yielding the empty pushdown (and then, e.g. accept the string on final state). Thus, when simulating $\mathcal{A}$, the automaton $\mathcal{B}$ also has to block its computation whenever $\mathcal{A}$ applies a pop to a single-symbol pushdown. For this purpose, we insert a new symbol # at the very bottom of the pushdown whereby $\mathcal{B}$ can detect the mentioned situation. Moreover, as $\mathcal{B}$ accepts with empty pushdown, the pushdown must be emptied when $\mathcal{A}$ accepts.

Definition 4.13. Let $\mathcal{A}=(Q,\Sigma,c_0,q_0,Q_f,T,\mathrm{wt})$ be a (P^1,Σ,K)-automaton in test normal form. Recall that $c_0=(\gamma_0,\mathrm{c})$ for some $\gamma_0\in\Gamma$. Furthermore, let $\Gamma_\mathcal{A}=\{\gamma_0\}\cup X$, where X is the finite set of pushdown symbols occurring in transitions of $\mathcal{A}$ and let $\#\in\Gamma$ be a symbol such that $\Gamma_\mathcal{A}\cap\{\#\}=\emptyset$.

Then we define $\mathcal{B}=(Q_\mathcal{B},\Sigma,\#,q_{0,\mathcal{B}},\{f_\mathcal{B}\},T_\mathcal{B},\mathrm{wt}_\mathcal{B})$ as the (PD,Σ,K)-*automaton induced by* $\mathcal{A}$, where $Q_\mathcal{B}=Q\cup\bar{Q}\cup Q_1\cup Q_2\cup\{q_{0,\mathcal{B}},f_\mathcal{B}\}$ with some new states $q_{0,\mathcal{B}},f_\mathcal{B}\notin Q$, as well as $Q_1=(Q\times\Gamma_\mathcal{A})$, $Q_2=(Q\times\Gamma_\mathcal{A}\times\Gamma_\mathcal{A})$, and $\bar{Q}=\{\bar{q}\mid q\in Q\}$. The set $T_\mathcal{B}$ of transitions is defined as follows:

- The transition $\tau=(q_{0,\mathcal{B}},\varepsilon,\#?,q_0,\gamma_0\#!)$ is in $T_\mathcal{B}$ and $\mathrm{wt}_\mathcal{B}(\tau)=1$.
- Let $\tau=(q,x,\mathrm{top}=\gamma,q',f)$ be in T.
  - If $f=\mathrm{pop}$, then $\tau_\mathcal{B}=(q,x,\gamma?,\overline{q'},\varepsilon!)$ is in $T_\mathcal{B}$ and $\mathrm{wt}_\mathcal{B}(\tau_\mathcal{B})=\mathrm{wt}(\tau)$. Furthermore, for every $\delta\in\Gamma_\mathcal{A}$, the transition $(\overline{q'},\varepsilon,\delta?,q',\delta!)$ with weight 1 is in $T_\mathcal{B}$.
  - If $f=\mathrm{push}(\delta,f_{\mathrm{id}})$ for some $\delta\in\Gamma_\mathcal{A}$, then $\tau_\mathcal{B}=(q,x,\gamma?,q',\delta\gamma!)$ is in $T_\mathcal{B}$ and $\mathrm{wt}_\mathcal{B}(\tau_\mathcal{B})=\mathrm{wt}(\tau)$.
  - If $f=\mathrm{stay}(\delta)$ for some $\delta\in\Gamma_\mathcal{A}$, then $\tau_\mathcal{B}=(q,x,\gamma?,q',\delta!)$ is in $T_\mathcal{B}$ and $\mathrm{wt}_\mathcal{B}(\tau_\mathcal{B})=\mathrm{wt}(\tau)$.
- Let $\tau=(q,x,\mathrm{bottom},q',f)$ be in T.
  - If $f=\mathrm{push}(\delta,f_{\mathrm{id}})$ for some $\delta\in\Gamma_\mathcal{A}$, then for every $\gamma\in\Gamma_\mathcal{A}$ the transition $\tau_\mathcal{B}=(q,x,\gamma?,(q',\gamma,\delta),\varepsilon!)$ is in $T_\mathcal{B}$ and $\mathrm{wt}_\mathcal{B}(\tau_\mathcal{B})=\mathrm{wt}(\tau)$. Furthermore, the transition $((q',\gamma,\delta),\varepsilon,\#?,q',\delta\gamma\#!)$ with weight 1 is in $T_\mathcal{B}$.
  - If $f=\mathrm{stay}(\delta)$ for some $\delta\in\Gamma_\mathcal{A}$, then the transition $\tau_\mathcal{B}=(q,x,\gamma?,(q',\delta),\varepsilon!)$ is in $T_\mathcal{B}$ for every $\gamma\in\Gamma_\mathcal{A}$ and $\mathrm{wt}_\mathcal{B}(\tau_\mathcal{B})=\mathrm{wt}(\tau)$. Furthermore, the transition $((q',\delta),\varepsilon,\#?,q',\delta\#!)$ with weight 1 is in $T_\mathcal{B}$.
- For every $q\in Q_f$ and $\delta\in\Gamma_\mathcal{A}\cup\{\#\}$ the transition $(q,\varepsilon,\delta?,f_\mathcal{B},\varepsilon!)$ with weight 1 is in $T_\mathcal{B}$. Furthermore the transition $(f_\mathcal{B},\varepsilon,\delta?,f_\mathcal{B},\varepsilon!)$ with weight 1 is in $T_\mathcal{B}$. □

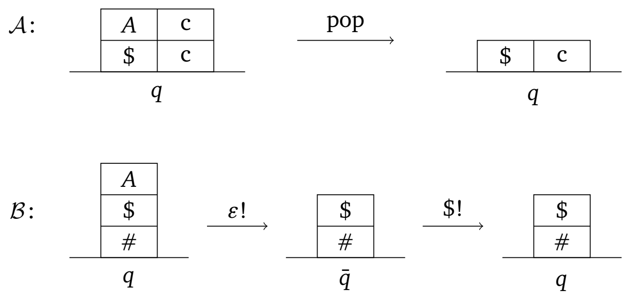


Figure 4.1: Pushdown behaviour during the simulation of a transition $\tau = (q, a, \mathrm{top} = A, q, \mathrm{pop})$ of the $(\mathrm{P}^1, \Sigma, \mathcal{K}^\lambda_{\mathrm{disc}})$-automaton $\mathcal{A}$ by the $(\mathrm{PD}, \Sigma, \mathcal{K}^\lambda_{\mathrm{disc}})$-automaton $\mathcal{B}$ which is induced by $\mathcal{A}$.

Example 4.14. Recall the $(\mathrm{P}^1, \Sigma, \mathcal{K}^\lambda_{\mathrm{disc}})$-automaton $\mathcal{A} = (\{q, f\}, \Sigma, (\$, \mathrm{c}), q, \{f\}, T, \mathrm{wt})$ from Example 3.31. Clearly, $\mathcal{A}$ is in test normal form and $\Gamma_\mathcal{A} = \{B, A, \$\}$. Then the $(\mathrm{PD}, \Sigma, \mathcal{K}^\lambda_{\mathrm{disc}})$-automaton induced by $\mathcal{A}$ is the automaton $\mathcal{B} = (Q_\mathcal{B}, \Sigma, \#, q_{0,\mathcal{B}}, \{f_\mathcal{B}\}, T_\mathcal{B}, \mathrm{wt}_\mathcal{B})$, where

$$\begin{aligned} Q_\mathcal{B} = {} & \{q, f\} \cup \{\bar{q}, \bar{f}\} \cup \{q_{0,\mathcal{B}}, f_\mathcal{B}\} \\ & \cup \{(q,B), (q,A), (q,\$), (f,B), (f,A), (f,\$)\} \\ & \cup \{(q,B,B), (q,A,B), (q,\$,B), (q,B,A), (q,A,A), (q,\$,A), \\ & \quad (q,B,\$), (q,A,\$), (q,\$,\$), (f,B,B), (f,A,B), (f,\$,B), \\ & \quad (f,B,A), (f,A,A), (f,\$,A), (f,B,\$), (f,A,\$), (f,\$,\$)\}. \end{aligned}$$

Furthermore, we show based on different transitions from T, how the transitions in $T_\mathcal{B}$ and the weight assignment are constructed. First, consider the transition

$$\tau = (q, a, \mathrm{top} = A, q, \mathrm{pop}) \in T$$

and recall that $\mathrm{wt}(\tau) = 2$. To check that there is at least one symbol left on the pushdown after applying a pop, the transition

$$(q, a, A?, \bar{q}, \varepsilon!)$$

with weight 2 is in $T_\mathcal{B}$, which switches into the auxiliary state $\bar{q}$. This state $\bar{q}$ can be left by one of the transitions

$$(\bar{q}, \varepsilon, B?, q, B!),\ (\bar{q}, \varepsilon, A?, q, A!),\ \text{and } (\bar{q}, \varepsilon, \$?, q, \$!),$$

which are additionally in $T_\mathcal{B}$, each with weight 0.[1] The simulation of τ is illustrated in Figure 4.1.

[1] Recall that 0 is the 1 element of $\mathcal{K}^\lambda_{\mathrm{disc}}$.

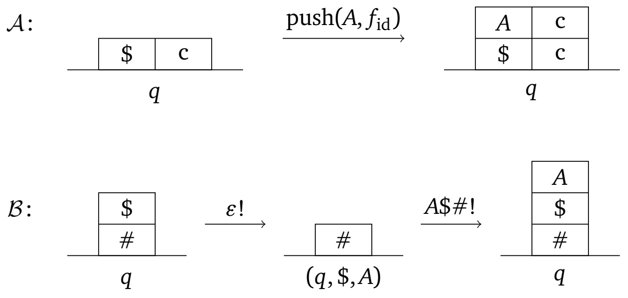


Figure 4.2: Pushdown behaviour during the simulation of a transition $\tau = (q, b, \mathrm{bottom}, q, \mathrm{push}(A, f_{\mathrm{id}}))$ of the $(\mathrm{P}^1, \Sigma, \mathcal{K}^{\lambda}_{\mathrm{disc}})$-automaton $\mathcal{A}$ by the $(\mathrm{PD}, \Sigma, \mathcal{K}^{\lambda}_{\mathrm{disc}})$-automaton $\mathcal{B}$ which is induced by $\mathcal{A}$.

As a second example, we consider the transition

$$\tau = (q, b, \mathrm{bottom}, q, \mathrm{push}(A, f_{\mathrm{id}})) \in T$$

and recall that $\mathrm{wt}(\tau) = 1$. In order to test bottom before pushing the symbol A, the transitions

$$(q, b, B?, (q,B,A), \varepsilon!),\ (q, b, A?, (q,A,A), \varepsilon!),\ \text{and } (q, b, \$?, (q,\$,A), \varepsilon!),$$

each with weight 1, are in $T_{\mathcal{B}}$. These transitions remove the topmost pushdown symbol and save it, together with the target state of τ and the symbol to be pushed, in their target state. Now the symbol #, which simulates the bottom of the pushdown, can be tested. This is done by one of the transitions

$$((q,B,A), \varepsilon, \#?, q, AB\#!),\ ((q,A,A), \varepsilon, \#?, q, AA\#!),\ \text{and } ((q,\$,A), \varepsilon, \#?, q, A\$\#!),$$

which are additionally in $T_{\mathcal{B}}$, each with weight 0. If this test is successful, the saved symbols are pushed and the automaton switches into the target state of τ. For an illustration of this simulation see Figure 4.2.

The remaining transitions of $T_{\mathcal{B}}$ are constructed in a similar way and as described in Definition 4.13. $\square$

For the rest of this section let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T, \mathrm{wt})$ *be a* $(\mathrm{P}^1, \Sigma, K)$*-automaton in test normal form and let* $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, \#, q_{0,\mathcal{B}}, \{f_{\mathcal{B}}\}, T_{\mathcal{B}}, \mathrm{wt}_{\mathcal{B}})$ *be the* (PD, Σ, K)*-automaton induced by* $\mathcal{A}$*.*

To show the correctness of this construction, i.e., to prove that $\|\mathcal{A}\| = \|\mathcal{B}\|_{\varepsilon}$, we want to indicate a bijection between the computations of $\mathcal{A}$ and $\mathcal{B}$. For this, we define a function π, which maps sequences of transitions over T to sets of sequences of transition over $T_{\mathcal{B}}$. This "nondeterminism" (which means that one sequence is mapped to several sequences) arises since a constructed transition not only depends on one transition in the preimage of π, but also on the current configuration of the pushdown. But we will show afterwards that π nevertheless will be helpful to define a function between computations of $\mathcal{A}$ and $\mathcal{B}$.

Definition 4.15. We define a mapping $\pi\colon T \to \mathcal{P}(T_{\mathcal{B}}^*)$ for every $\tau \in T$ as follows:

- if $\tau = (q, x, \mathrm{top} = \gamma, q', \mathrm{pop})$, then $\pi(\tau) = \{(q, x, \gamma?, \overline{q'}, \varepsilon!)(\overline{q'}, \varepsilon, \delta?, q', \delta!) \mid \delta \in \Gamma_{\mathcal{A}}\}$
- if $\tau = (q, x, \mathrm{top} = \gamma, q', \mathrm{push}(\delta, f_{\mathrm{id}}))$, then $\pi(\tau) = \{(q, x, \gamma?, q', \delta\gamma!)\}$
- if $\tau = (q, x, \mathrm{top} = \gamma, q', \mathrm{stay}(\delta))$, then $\pi(\tau) = \{(q, x, \gamma?, q', \delta!)\}$
- if $\tau = (q, x, \mathrm{bottom}, q', \mathrm{push}(\delta, f_{\mathrm{id}}))$, then $\pi(\tau) = \{(q, x, \gamma?, (q', \gamma, \delta), \varepsilon!)((q', \gamma, \delta), \varepsilon, \#?, q', \delta\gamma\#!) \mid \gamma \in \Gamma_{\mathcal{A}}\}$
- if $\tau = (q, x, \mathrm{bottom}, q', \mathrm{stay}(\delta)$, then $\pi(\tau) = \{(q, x, \gamma?, (q', \delta), \varepsilon!)((q', \delta), \varepsilon, \#?, q', \delta\#!) \mid \gamma \in \Gamma_{\mathcal{A}}\}$.

We can extend π to a mapping $\pi'\colon T^* \to \mathcal{P}(T_{\mathcal{B}}^*)$ by defining $\pi'(\varepsilon) = \{\varepsilon\}$ and $\pi'(\tau_1 \dots \tau_n) = \pi(\tau_1) \dots \pi(\tau_n)$, for $n \geq 1$, $\tau_1, \dots, \tau_n \in T$. In the following we identify π and π'. □

If we now look at π more detailed, we notice two facts. There are transitions of $\mathcal{B}$ that are not in the image of π, such as the only transition of $\mathcal{B}$ which starts in an initial state as well as transitions that reach the final state of $\mathcal{B}$. For this reason, computations of $\mathcal{A}$ can only be mapped to parts of computations of $\mathcal{B}$, but not to complete computations[2]. Furthermore, it does not suffice to restrict the domain of π to computations of $\mathcal{A}$ to avoid sequences that are no parts of computations of $\mathcal{B}$ in its image. These two facts are illustrated in the following example.

Example 4.16. Recall the $(\mathrm{P}^1, \Sigma, \mathcal{K}_{\mathrm{disc}}^{\lambda})$-automaton $\mathcal{A} = (\{q, f\}, \Sigma, (\$, c), q, \{f\}, T, \mathrm{wt})$ from Example 3.31 and the $(\mathrm{PD}, \Sigma, \mathcal{K}_{\mathrm{disc}}^{\lambda})$-automaton $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, \#, q_{0,\mathcal{B}}, \{f_{\mathcal{B}}\}, T_{\mathcal{B}}, \mathrm{wt}_{\mathcal{B}})$ induced by $\mathcal{A}$ from Example 4.14.

Now consider the computation

$$\theta = (q, a, \mathrm{bottom}, q, \mathrm{push}(B, f_{\mathrm{id}}))(q, b, \mathrm{top} = B, q, \mathrm{pop})(q, \varepsilon, \mathrm{bottom}, f, \mathrm{stay}(\$))$$

in $\Theta_{\mathcal{A}}(ab)$. It is easy to see that the sequence

$$\begin{aligned}\theta' = {} & (q, a, \$?, (q, \$, B), \varepsilon!)((q, \$, B), \varepsilon, \#?, q, B\$\#!)(q, b, B?, \bar{q}, \varepsilon!) \\ & (\bar{q}, \varepsilon, A?, q, A!)(q, \varepsilon, \$?, (f, \$), \varepsilon!)((f, \$), \varepsilon, \#?, f, \$!)\end{aligned}$$

[2] Here, for the sake of brevity, we say that θ is *mapped* to θ' if $\theta' \in \pi(\theta)$, and the *image* of π is $\bigcup_{\theta} \pi(\theta)$.

is in $\pi(\theta)$ but is neither a computation nor a part of a computation of $\mathcal{B}$ (for example, the predicate $\mathcal{A}$? in the fourth transition cannot be successful). Also the sequence

$$\theta'' = (q,a,\$?,(q,\$,B),\varepsilon!)((q,\$,B),\varepsilon,\#?,q,B\$\#!)(q,b,B?,\bar{q},\varepsilon!)$$
$$(\bar{q},\varepsilon,\$?,q,\$!)(q,\varepsilon,\$?,(f,\$),\varepsilon!)((f,\$),\varepsilon,\#?,f,\$!)$$

in $\pi(\theta)$ is no computation of $\mathcal{B}$, but it is a part of a computation. By adding the transition $\tau_0 = (q_{0,\mathcal{B}},\varepsilon,\#?,q,\$\#!)$ at the beginning and the transitions $\tau_1 = (f,\varepsilon,\$?,f_{\mathcal{B}},\varepsilon!)$ and $\tau_2 = (f_{\mathcal{B}},\varepsilon,\#?,f_{\mathcal{B}},\varepsilon!)$ at the end of θ'', we obtain that $\tau_0\theta''\tau_1\tau_2 \in \Theta^{\varepsilon}_{\mathcal{B}}(ab)$. □

To complete parts of computations in the image of π to computations from $\mathcal{B}$ we introduce the concept of extensions.

Definition 4.17. Let $\theta \in T^*$,

$$\vartheta_1 = \{(q,\varepsilon,\delta?,f_{\mathcal{B}},\varepsilon!) \in T_{\mathcal{B}} \mid q \in Q_f, \delta \in \Gamma_{\mathcal{A}}\},$$
$$\vartheta_2 = \{(f_{\mathcal{B}},\varepsilon,\delta?,f_{\mathcal{B}},\varepsilon!) \in T_{\mathcal{B}} \mid \delta \in \Gamma_{\mathcal{A}}\},$$

and $\tau_0 = (q_{0,\mathcal{B}},\varepsilon,\#?,q_0,\gamma_0\#!)$. We say that a sequence $\theta' \in T^*_{\mathcal{B}}$ *extends* θ if $\theta' \in \pi(\theta)\vartheta_1\vartheta_2^*$ and τ_0-*extends* θ if $\theta' \in \tau_0\pi(\theta)\vartheta_1\vartheta_2^*$.[3] □

Example 4.18. Recall the $(\mathrm{PD},\Sigma,\mathcal{K}^{\lambda}_{\mathrm{disc}})$-automaton $\mathcal{B} = (Q_{\mathcal{B}},\Sigma,\#,q_{0,\mathcal{B}},\{f_{\mathcal{B}}\},T_{\mathcal{B}},\mathrm{wt}_{\mathcal{B}})$ from Example 4.14 and the computation θ as well as the sequence θ'' from Example 4.16. It is easy to see that the sequence

$$(q_{0,\mathcal{B}},\varepsilon,\#?,q,\$\#!)\theta''(f,\varepsilon,A?,f_{\mathcal{B}},\varepsilon!)(f_{\mathcal{B}},\varepsilon,\#?,f_{\mathcal{B}},\varepsilon!)$$

τ_0-extends θ but is no computation of $\mathcal{B}$ since the predicate A? in the last but one transition is not successful. □

As the previous example has shown, not all extensions of a q-computation θ from $\mathcal{A}$ have to be computations (or parts of computations) of $\mathcal{B}$. But we will prove now, roughly speaking, that under all extensions of θ there is exactly one extension θ' which forms a (q,ε)-computation of $\mathcal{B}$ with the same initial automaton configuration and, furthermore, θ and θ' are mapped by $\overline{\mathrm{wt}}$ to the same sequence of values from K.

Lemma 4.19. Let $q \in Q$, $w \in \Sigma^*$, and $\eta \in \Gamma^+$. If θ is a q-computation in $\overline{\Theta}_{\mathcal{A}}(q,w,\eta)$, then there is exactly one (q,ε)-computation θ' in $\Theta^{\varepsilon}_{\mathcal{B}}(q,w,\eta\#)$ s.t. θ' extends θ. Moreover, $\overline{\mathrm{wt}}(\theta) = \overline{\mathrm{wt}}(\theta')$.

Proof. Let $q \in Q$, $w \in \Sigma^*$, $\eta \in \Gamma^+$, and $\theta \in \overline{\Theta}_{\mathcal{A}}(q,w,\eta)$. We prove this statement by induction on the length of θ.

[3] Recall that we abbreviate the language concatenation $\{a\}B$ by aB.

First, let $\theta = \varepsilon$. As $\theta \in \overline{\Theta}_{\mathcal{A}}(q, w, \eta)$, this means $q \in Q_f$ and $w = \varepsilon$. Moreover, $\pi(\theta) = \{\varepsilon\}$. It remains to show that there is exactly one $\theta' \in \vartheta_1\vartheta_2^*$ such that $\theta' \in \Theta^{\varepsilon}_{\mathcal{B}}(q, w, \eta\#)$. For this, assume that $\eta = \gamma_1 \dots \gamma_n$ for some $n \geq 1$ and $\gamma_1, \dots, \gamma_n \in \Gamma_{\mathcal{A}}$. Then θ' is uniquely given by $\theta' = (q, \varepsilon, \gamma_1?, f_{\mathcal{B}}, \varepsilon!)(f_{\mathcal{B}}, \varepsilon, \gamma_2?, f_{\mathcal{B}}, \varepsilon!) \dots (f_{\mathcal{B}}, \varepsilon, \gamma_n?, f_{\mathcal{B}}, \varepsilon!)(f_{\mathcal{B}}, \varepsilon, \#?, f_{\mathcal{B}}, \varepsilon!) \in \vartheta_1\vartheta_2^*$.

By construction, $\mathrm{wt}_{\mathcal{B}}((q, \varepsilon, \gamma_1?, f_{\mathcal{B}}, \varepsilon!)) = 1$, $\mathrm{wt}_{\mathcal{B}}((f_{\mathcal{B}}, \varepsilon, \gamma_i?, f_{\mathcal{B}}, \varepsilon!)) = 1$ for $1 < i \leq n$, and $\mathrm{wt}_{\mathcal{B}}((f_{\mathcal{B}}, \varepsilon, \#?, f_{\mathcal{B}}, \varepsilon!)) = 1$. Therefore, $\overline{\mathrm{wt}}(\theta') = \varepsilon = \overline{\mathrm{wt}}(\theta)$.

Now, let $\theta = \tau\theta_1$ for $\tau \in T$, $\theta_1 \in T^*$. In the following, we distinguish five cases of τ.

Case 1: Let $\tau = (q, x, \mathrm{top} = \gamma, q', \mathrm{pop})$ for some $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, and $\gamma \in \Gamma_{\mathcal{A}}$. Then $w = xw'$ for some $w' \in \Sigma^*$ and $\eta = \gamma\eta'$ for some $\eta' \in \Gamma^*_{\mathcal{A}}$. Since $\theta \in \overline{\Theta}_{\mathcal{A}}(q, w, \eta)$ it follows that $\theta_1 \in \overline{\Theta}_{\mathcal{A}}(q', w', \eta')$. By induction hypothesis there is exactly one $\theta'_1 \in \Theta^{\varepsilon}_{\mathcal{B}}(q', w', \eta'\#)$ such that θ'_1 extends θ_1.

It remains to show that there is exactly one $\tau' \in \pi(\tau)$ with $\tau'\theta'_1 \in \Theta^{\varepsilon}_{\mathcal{B}}(q, w, \eta\#)$. Since τ occurs in a computation and pops a symbol, we can assume that η consists of more than one symbol and therefore $\eta' = \delta\eta''$ for some $\delta \in \Gamma_{\mathcal{A}}$, $\eta'' \in \Gamma^*_{\mathcal{A}}$. Then τ' is uniquely given by $\tau' = (q, x, \gamma?, \overline{q'}, \varepsilon!)(\overline{q'}, \varepsilon, \delta?, q', \delta!)$ since only this sequence is compatible with the assumed pushdown configuration in the computation.

Furthermore, we have that

$$\overline{\mathrm{wt}}(\tau\theta_1) = \overline{\mathrm{wt}}(\tau)\overline{\mathrm{wt}}(\theta_1) \overset{\text{(IH)}}{=} \overline{\mathrm{wt}}(\tau)\overline{\mathrm{wt}}(\theta'_1) \overset{(*)}{=} \overline{\mathrm{wt}}(\tau')\overline{\mathrm{wt}}(\theta'_1) = \overline{\mathrm{wt}}(\tau'\theta'_1),$$

where $(*)$ holds since $\mathrm{wt}(\tau) = \mathrm{wt}_{\mathcal{B}}((q, x, \gamma?, \overline{q'}, \varepsilon!))$ and $\mathrm{wt}_{\mathcal{B}}((\overline{q'}, \varepsilon, \delta?, q', \delta!)) = 1$ by construction.

Case 2: Let $\tau = (q, x, \mathrm{top} = \gamma, q', \mathrm{push}(\delta, f_{\mathrm{id}}))$ for some q, $q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, and $\gamma, \delta \in \Gamma_{\mathcal{A}}$. Then $w = xw'$ for some $w' \in \Sigma^*$ and since $\theta \in \overline{\Theta}_{\mathcal{A}}(q, w, \eta)$ it follows that $\theta_1 \in \overline{\Theta}_{\mathcal{A}}(q', w', \delta\eta)$. By induction hypothesis there is exactly one $\theta'_1 \in \Theta^{\varepsilon}_{\mathcal{B}}(q', w', \delta\eta\#)$ such that θ'_1 extends θ_1.

It remains to show that there is exactly one $\tau' \in \pi(\tau)$ with $\tau'\theta'_1 \in \Theta^{\varepsilon}_{\mathcal{B}}(q, w, \eta\#)$. Since $\pi(\tau) = \{(q, x, \gamma?, q', \delta\gamma!)\}$, this is uniquely determined.

Furthermore, we have that

$$\overline{\mathrm{wt}}(\tau\theta_1) = \overline{\mathrm{wt}}(\tau)\overline{\mathrm{wt}}(\theta_1) \overset{\text{(IH)}}{=} \overline{\mathrm{wt}}(\tau)\overline{\mathrm{wt}}(\theta'_1) \overset{(*)}{=} \overline{\mathrm{wt}}(\tau')\overline{\mathrm{wt}}(\theta'_1) = \overline{\mathrm{wt}}(\tau'\theta'_1),$$

where $(*)$ holds since $\mathrm{wt}(\tau) = \mathrm{wt}_{\mathcal{B}}(\tau')$ by construction.

Case 3: Let $\tau = (q, x, \mathrm{top} = \gamma, q', \mathrm{stay}(\delta))$ for some $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, and $\gamma, \delta \in \Gamma_{\mathcal{A}}$. Then $w = xw'$ for some $w' \in \Sigma^*$ and $\eta = \gamma\eta'$ for some $\eta' \in \Gamma^*_{\mathcal{A}}$. Since $\theta \in \overline{\Theta}_{\mathcal{A}}(q, w, \eta)$ it follows that $\theta_1 \in \overline{\Theta}_{\mathcal{A}}(q', w', \delta\eta')$. By induction hypothesis there is exactly one $\theta'_1 \in \Theta^{\varepsilon}_{\mathcal{B}}(q', w', \delta\eta'\#)$ such that θ'_1 extends θ_1.

It remains to show that there is exactly one $\tau' \in \pi(\tau)$ with $\tau'\theta_1' \in \Theta^{\varepsilon}_{\mathcal{B}}(q,w,\eta\#)$. Since $\pi(\tau) = \{(q,x,\gamma?,q',\delta!)\}$, this is uniquely determined.

Furthermore, we have that

$$\overline{\mathrm{wt}}(\tau\theta_1) = \overline{\mathrm{wt}}(\tau)\overline{\mathrm{wt}}(\theta_1) \overset{(\mathrm{IH})}{=} \overline{\mathrm{wt}}(\tau)\overline{\mathrm{wt}}(\theta_1') \overset{(*)}{=} \overline{\mathrm{wt}}(\tau')\overline{\mathrm{wt}}(\theta_1') = \overline{\mathrm{wt}}(\tau'\theta_1'),$$

where $(*)$ holds since $\mathrm{wt}(\tau) = \mathrm{wt}_{\mathcal{B}}(\tau')$ by construction.

Case 4: Let $\tau = (q,x,\mathrm{bottom},q',\mathrm{push}(\delta,f_{\mathrm{id}}))$ for some $q,q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, and $\delta \in \Gamma_{\mathcal{A}}$. Then $w = xw'$ for some $w' \in \Sigma^*$. Since $\theta \in \overline{\Theta}_{\mathcal{A}}(q,w,\eta)$ it follows that $\theta_1 \in \overline{\Theta}_{\mathcal{A}}(q',w',\delta\eta)$. By induction hypothesis there is exactly one $\theta_1' \in \Theta^{\varepsilon}_{\mathcal{B}}(q',w',\delta\eta\#)$ such that θ_1' extends θ_1.

It remains to show that there is exactly one $\tau' \in \pi(\tau)$ with $\tau'\theta_1' \in \Theta^{\varepsilon}_{\mathcal{B}}(q,w,\eta\#)$. Since τ occurs in a computation and tests bottom, we can assume that there is only one symbol left on the pushdown and therefore $\eta = \gamma$ for some $\gamma \in \Gamma_{\mathcal{A}}$. Then τ' is uniquely given by $\tau' = (q,x,\gamma?,(q',\gamma,\delta),\varepsilon!)((q',\gamma,\delta),\varepsilon,\#?,q',\delta\gamma\#!)$ since only this sequence is compatible with the assumed pushdown configuration in the computation.

Furthermore, we have that

$$\overline{\mathrm{wt}}(\tau\theta_1) = \overline{\mathrm{wt}}(\tau)\overline{\mathrm{wt}}(\theta_1) \overset{(\mathrm{IH})}{=} \overline{\mathrm{wt}}(\tau)\overline{\mathrm{wt}}(\theta_1') \overset{(*)}{=} \overline{\mathrm{wt}}(\tau')\overline{\mathrm{wt}}(\theta_1') = \overline{\mathrm{wt}}(\tau'\theta_1'),$$

where $(*)$ holds since, by construction, we have that $\mathrm{wt}(\tau) = \mathrm{wt}_{\mathcal{B}}((q,x,\gamma?,(q',\gamma,\delta),\varepsilon!))$ and $\mathrm{wt}_{\mathcal{B}}(((q',\gamma,\delta),\varepsilon,\#?,q',\delta\gamma\#!)) = 1$.

Case 5: Let $\tau = (q,x,\mathrm{bottom},q',\mathrm{stay}(\delta))$ for some $q,q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, and $\delta \in \Gamma_{\mathcal{A}}$. Then $w = xw'$ for some $w' \in \Sigma^*$ and $\eta = \gamma$ for some $\gamma \in \Gamma_{\mathcal{A}}$. Since $\theta \in \overline{\Theta}_{\mathcal{A}}(q,w,\eta)$ it follows that $\theta_1 \in \overline{\Theta}_{\mathcal{A}}(q',w',\delta)$. By induction hypothesis there is exactly one $\theta_1' \in \Theta^{\varepsilon}_{\mathcal{B}}(q',w',\delta\#)$ such that θ_1' extends θ_1.

It remains to show that there is exactly one $\tau' \in \pi(\tau)$ with $\tau'\theta_1' \in \Theta^{\varepsilon}_{\mathcal{B}}(q,w,\eta\#)$. By the same argumentation as in Case 4, τ' is uniquely given by
$\tau' = (q,x,\gamma?,(q',\delta),\varepsilon!)((q',\delta),\varepsilon,\#?,q',\delta\#!)$ since only this sequence is compatible with the assumed pushdown configuration in the computation.

Furthermore, we have that

$$\overline{\mathrm{wt}}(\tau\theta_1) = \overline{\mathrm{wt}}(\tau)\overline{\mathrm{wt}}(\theta_1) \overset{(\mathrm{IH})}{=} \overline{\mathrm{wt}}(\tau)\overline{\mathrm{wt}}(\theta_1') \overset{(*)}{=} \overline{\mathrm{wt}}(\tau')\overline{\mathrm{wt}}(\theta_1') = \overline{\mathrm{wt}}(\tau'\theta_1'),$$

where $(*)$ holds since, by construction, we have that $\mathrm{wt}(\tau) = \mathrm{wt}_{\mathcal{B}}((q,x,\gamma?,(q',\delta),\varepsilon!))$ and $\mathrm{wt}_{\mathcal{B}}(((q',\delta),\varepsilon,\#?,q',\delta\#!)) = 1$. ■

Now, where we have seen that each q-computation of $\mathcal{A}$ is extended by exactly one "compatible" (q,ε)-computation of $\mathcal{B}$, we can define the following function.

Definition 4.20. For every $w \in \Sigma^*$ let

$$f_w \colon \Theta_{\mathcal{A}}(w) \to \Theta^{\varepsilon}_{\mathcal{B}}(w)$$

be the function which maps each $\theta \in \Theta_{\mathcal{A}}(w)$ to the uniquely determined computation $\theta' \in \Theta^{\varepsilon}_{\mathcal{B}}(w)$ that τ_0-extends θ. □

Note that this function is indeed well-defined by Lemma 4.19 and since τ_0 is the uniquely determined transition $(q_{0,\mathcal{B}}, \varepsilon, \#?, q_0, \gamma_0\#!)$. It remains to show that, for every $w \in \Sigma^*$, f_w is injective, surjective, and preserves the weight of a computation in K.

Let us start with proving the injectivity of f_w for each $w \in \Sigma^*$.

Lemma 4.21. For every $w \in \Sigma^*$ it holds that f_w is injective.

Proof. Let $t \in T^*_{\mathcal{B}}$. We show the lemma by proving the following property for t:

Let $t_1, t_2 \in T^*$. If $t \in \pi(t_1) \cap \pi(t_2)$, then $t_1 = t_2$.

We prove this statement by induction on the length of t.

As induction basis let $|t| = 0$. Then $t = \varepsilon$ and it follows that $t_1 = t_2 = \varepsilon$.

Now let $|t| = n + 1$ for some $n \geq 0$ and assume that the property holds for every word $u \in T^*_{\mathcal{B}}$ with $|u| \leq n$. Let $t = \tau t'$ for some $\tau \in T_{\mathcal{B}}$, $t' \in T^*_{\mathcal{B}}$, and let $t_1, t_2 \in T^*$ such that $\tau t' \in \pi(t_1) \cap \pi(t_2)$. In the following, we distinguish four cases of τ.

Case 1: Let $\tau = (q, x, \gamma?, q', \eta!)$ for some $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, $\gamma \in \Gamma_{\mathcal{A}}$, and $\eta \in \Gamma^+_{\mathcal{A}}$. Then $t_1 = \tau_1 t'_1$ and $t_2 = \tau_2 t'_2$ for some $\tau_1, \tau_2 \in T$ such that $\tau \in \pi(\tau_1) \cap \pi(\tau_2)$ and, by definition of π, $\tau_1 = \tau_2$ since all pieces of information of τ_1 and τ_2 are coded into τ.

Also, it follows that $t' \in \pi(t'_1) \cap \pi(t'_2)$ and, by the induction hypothesis, $t'_1 = t'_2$. Thus, $t_1 = t_2$.

Case 2: Let $\tau = (q, x, \gamma?, \overline{q'}, \varepsilon!)$ for some $q \in Q$, $\overline{q'} \in \overline{Q}$, $x \in \Sigma \cup \{\varepsilon\}$, and $\gamma \in \Gamma_{\mathcal{A}}$. Then we must have that $t' = \tau' t''$ with $\tau' = (\overline{q'}, \varepsilon, \delta?, q', \varepsilon)$ for some $\delta \in \Gamma_{\mathcal{A}}$ as τ only occurs followed by such a second transition in the image of π.

Furthermore, $t_1 = \tau_1 t'_1$ and $t_2 = \tau_2 t'_2$ for some $\tau_1, \tau_2 \in T$ such that $\tau\tau' \in \pi(\tau_1) \cap \pi(\tau_2)$. By definition of π, we have that $\tau_1 = \tau_2$.

Also, it follows that $t'' \in \pi(t'_1) \cap \pi(t'_2)$ and, by the induction hypothesis, $t'_1 = t'_2$. Thus, $t_1 = t_2$.

Case 3: Let $\tau = (q, x, \gamma?, (q', \delta), \varepsilon!)$ for some $q \in Q$, $(q', \delta) \in Q_1$, $x \in \Sigma \cup \{\varepsilon\}$, and $\gamma \in \Gamma_{\mathcal{A}}$. Then we must have that $t' = \tau' t''$ with $\tau' = ((q', \delta), \varepsilon, \#?, q', \delta\#!)$ as τ only occurs followed by this second transition in the image of π.

Furthermore, $t_1 = \tau_1 t_1'$ and $t_2 = \tau_2 t_2'$ for some $\tau_1, \tau_2 \in T$ such that $\tau\tau' \in \pi(\tau_1) \cap \pi(\tau_2)$. By definition of π, we have that $\tau_1 = \tau_2$.

Also, it follows that $t'' \in \pi(t_1') \cap \pi(t_2')$ and, by the induction hypothesis, $t_1' = t_2'$. Thus, $t_1 = t_2$.

Case 4: Let $\tau = (q, x, \gamma?, (q', \gamma, \delta), \varepsilon!)$ for some $q \in Q$, $(q', \gamma, \delta) \in Q_2$, $x \in \Sigma \cup \{\varepsilon\}$, and $\gamma \in \Gamma_{\mathcal{A}}$. Then we must have that $t' = \tau' t''$ with $\tau' = ((q', \gamma, \delta), \varepsilon, \#?, q', \delta\gamma\#!)$ as τ only occurs followed by this second transition in the image of π.

Furthermore, $t_1 = \tau_1 t_1'$ and $t_2 = \tau_2 t_2'$ for some $\tau_1, \tau_2 \in T$ such that $\tau\tau' \in \pi(\tau_1) \cap \pi(\tau_2)$. By definition of π, we have that $\tau_1 = \tau_2$.

Also, it follows that $t'' \in \pi(t_1') \cap \pi(t_2')$ and, by the induction hypothesis, $t_1' = t_2'$. Thus, $t_1 = t_2$.

Note that we need not to consider other forms of τ, since only the mentioned cases are in the image of π.

Now let $w \in \Sigma^*$. We prove by contraposition that f_w is injective. For this, we consider two computations $\theta_1, \theta_2 \in \Theta_{\mathcal{A}}(w)$ such that $f_w(\theta_1) = f_w(\theta_2)$. By definition of f_w it follows that $\tau_0\pi(\theta_1)\vartheta_1\vartheta_2^* \cap \tau_0\pi(\theta_2)\vartheta_1\vartheta_2^* \neq \emptyset$. Then $\pi(\theta_1) \cap \pi(\theta_2) \neq \emptyset$ as the sequences in the image of π and in $\vartheta_1\vartheta_2^*$ have no symbols in common. Therefore, there is a $t \in T_{\mathcal{B}}^*$ with $t \in \pi(\theta_1) \cap \pi(\theta_2)$ and it follows that $\theta_1 = \theta_2$. Thus, f_w is injective. ■

For the next lemma, we need to analyse the structure of computations of $\mathcal{B}$ and obtain the following observation:

Observation 4.22. For every $n \in \mathbb{N}$, $w \in \Sigma^*$, and $\theta \in \Theta_{\mathcal{B}}^{\varepsilon}(w)$ with $\theta = \tau_1 \ldots \tau_n$, $\tau_1, \ldots, \tau_n \in T_{\mathcal{B}}$, there exists a $k \in [n]$ such that $\tau_k = (q, \varepsilon, \delta?, f_{\mathcal{B}}, \varepsilon!)$ for some $q \in Q$ and for each i with $k < i \leq n$ we have $\tau_i = (f_{\mathcal{B}}, \varepsilon, \delta_i?, f_{\mathcal{B}}, \varepsilon!)$ for some $\delta_i \in \Gamma_{\mathcal{A}} \cup \{\#\}$. Moreover, $\tau_1 = (q_{0,\mathcal{B}}, \varepsilon, \#?, q_0, \gamma_0\#!)$, and we have that

(i) for every $2 \leq i \leq k-1$, $q \in Q$ and $\delta \in \Gamma_{\mathcal{A}} \cup \{\#\}$ it holds that $\tau_i \neq (q, \varepsilon, \delta?, f_{\mathcal{B}}, \varepsilon!)$ and $\tau_i \neq (f_{\mathcal{B}}, \varepsilon, \delta?, f_{\mathcal{B}}, \varepsilon!)$,

(ii) for every $2 \leq i \leq k-2$, $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, $\gamma \in \Gamma_{\mathcal{A}}$, $\overline{q'} \in \overline{Q}$ it holds that $\tau_i = (q, x, \gamma?, \overline{q'}, \varepsilon!)$ iff there is a $\delta \in \Gamma_{\mathcal{A}}$ such that $\tau_{i+1} = (\overline{q'}, \varepsilon, \delta?, q', \delta!)$,

(iii) for every $2 \leq i \leq k-2$, $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, $\gamma, \delta \in \Gamma_{\mathcal{A}}$, and $(q', \gamma, \delta) \in Q_2$ it holds that $\tau_i = (q, \varepsilon, \gamma?, (q', \gamma, \delta), \varepsilon!)$ iff $\tau_{i+1} = ((q', \gamma, \delta), x, \#?, q', \delta\gamma\#!)$, and

(iv) for every $2 \leq i \leq k-2$, $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, γ, $\delta \in \Gamma_{\mathcal{A}}$, and $(q', \delta) \in Q_1$ it holds that $\tau_i = (q, \varepsilon, \gamma?, (q', \delta), \varepsilon!)$ iff $\tau_{i+1} = ((q', \delta), x, \#?, q', \delta\#!)$.

Now we can prove the surjectivity of f_w for every $w \in \Sigma^*$.

Lemma 4.23. For every $w \in \Sigma^*$ it holds that f_w is surjective.

Proof. Let $\theta' \in T_{\mathcal{B}}^*$. We show this lemma by proving the following property of θ' for all $n \in \mathbb{N}$:

Let $q \in Q$, $w \in \Sigma^*$, and $\eta \in \Gamma^+$. If θ' is a (q, ε)-computation in $\Theta_{\mathcal{B}}^{\varepsilon}(q, w, \eta\#)$ of length n, then there is a q-computation θ in $\overline{\Theta}_{\mathcal{A}}(q, w, \eta)$ such that θ' extends θ.

We prove this statement by induction on the length n of θ'. As there is no (q, ε)-computation with length lower than 2 in $\Theta_{\mathcal{B}}^{\varepsilon}(q, w, \eta\#)$, the induction base is trivially true.

Now, let $|\theta'| = n + 1$ with $n \geq 1$ and $\theta' \in \Theta_{\mathcal{B}}^{\varepsilon}(q, w, \eta\#)$. By Observation 4.22 we have that θ' is of the form $\tau_1 \ldots \tau_{k-1}\tau_k\tau_{k+1} \ldots \tau_{n+1}$ for some $k \in [n + 1]$ and $\tau_1, \ldots, \tau_{n+1} \in T_{\mathcal{B}}$, where $\tau_k = (q, \varepsilon, \delta?, f_{\mathcal{B}}, \varepsilon!)$ for some $q \in Q$, $\delta \in \Gamma_{\mathcal{A}}$.

First, assume that $k = 1$. Then

$$\theta' = (q, \varepsilon, \delta_1?, f_{\mathcal{B}}, \varepsilon!)(f_{\mathcal{B}}, \varepsilon, \delta_2?, f_{\mathcal{B}}, \varepsilon!) \ldots (f_{\mathcal{B}}, \varepsilon, \delta_n?, f_{\mathcal{B}}, \varepsilon!)(f_{\mathcal{B}}, \varepsilon, \#?, f_{\mathcal{B}}, \varepsilon!)$$

for some $\delta_1, \ldots, \delta_n \in \Gamma_{\mathcal{A}}$. Then $q \in Q_f$, $w = \varepsilon$, $\eta = \delta_1 \ldots \delta_n$ and $\theta = \varepsilon$. Thus, $\theta \in \overline{\Theta}_{\mathcal{A}}(q, w, \eta)$ and θ' extends θ.

Now, let $k > 1$ and $\theta' = \tau'\theta'_1$ for some $\tau' \in T_{\mathcal{B}}$. Assume that the property holds for every word $u \in T_{\mathcal{B}}^*$ with $|u| \leq n$. In the following, we distinguish five cases of τ'.

Case 1: Let $\tau' = (q, x, \gamma?, q', \delta\gamma!)$ for some $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, and $\delta, \gamma \in \Gamma_{\mathcal{A}}$. Then $w = xw'$ for some $w' \in \Sigma^*$ and $\eta = \gamma\eta'$ for some $\eta' \in \Gamma_{\mathcal{A}}^*$. Since $\theta' \in \Theta_{\mathcal{B}}^{\varepsilon}(q, w, \eta\#)$ it follows that $\theta'_1 \in \Theta_{\mathcal{B}}^{\varepsilon}(q', w', \delta\eta\#)$. By the induction hypothesis there is a $\theta_1 \in \overline{\Theta}_{\mathcal{A}}(q', w', \delta\eta)$ such that θ'_1 extends θ_1.

Then let $\theta = \tau\theta_1$ with $\tau = (q, x, \mathrm{top} = \gamma, q', \mathrm{push}(\delta, f_{\mathrm{id}}))$. Thus, $\theta \in \overline{\Theta}_{\mathcal{A}}(q, w, \eta)$ and by definition of π it is clear that θ' extends θ.

Case 2: Let $\tau' = (q, x, \gamma?, q', \delta!)$ for some $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, and $\delta, \gamma \in \Gamma_{\mathcal{A}}$. Then $w = xw'$ for some $w' \in \Sigma^*$ and $\eta = \gamma\eta'$ for some $\eta' \in \Gamma_{\mathcal{A}}^*$. Since $\theta' \in \Theta_{\mathcal{B}}^{\varepsilon}(q, w, \eta\#)$ it follows that $\theta'_1 \in \Theta_{\mathcal{B}}^{\varepsilon}(q', w', \delta\eta'\#)$. By the induction hypothesis there is a $\theta_1 \in \overline{\Theta}_{\mathcal{A}}(q', w', \delta\eta')$ such that θ'_1 extends θ_1.

Then let $\theta = \tau\theta_1$ with $\tau = (q, x, \mathrm{top} = \gamma, q', \mathrm{stay}(\delta))$. Thus, $\theta \in \overline{\Theta}_{\mathcal{A}}(q, w, \eta)$ and by definition of π it is clear that θ' extends θ.

Case 3: Let $\tau = (q, x, \gamma?, \overline{q'}, \varepsilon!)$ for some $q \in Q$, $\overline{q'} \in \overline{Q}$, $x \in \Sigma \cup \{\varepsilon\}$, and $\gamma \in \Gamma_{\mathcal{A}}$. By Observation 4.22 we have that θ'_1 is of the form $(\overline{q'}, \varepsilon, \delta?, q', \delta!)\theta'_2$ for some $\delta \in \Gamma_{\mathcal{A}}$ and $\theta'_2 \in T_{\mathcal{B}}^*$. Then $\eta = \gamma\delta\eta'$ for some $\eta' \in \Gamma_{\mathcal{A}}^*$ and $w = xw'$ for some $w' \in \Sigma^*$. Since $\theta' \in \Theta_{\mathcal{B}}^{\varepsilon}(q, w, \eta\#)$ it follows that $\theta'_2 \in \Theta_{\mathcal{B}}^{\varepsilon}(q', w', \delta\eta'\#)$. By the induction hypothesis there is a $\theta_1 \in \overline{\Theta}_{\mathcal{A}}(q', w', \delta\eta')$ such that θ'_2 extends θ_1.

Then let $\theta=\tau\theta_1$ with $\tau=(q,x,\mathrm{top}=\gamma,q',\mathrm{pop})$. Thus, $\theta\in\overline{\Theta}_{\mathcal{A}}(q,w,\eta)$ and by definition of π it is clear that θ' extends θ.

Case 4: Let $\tau=(q,x,\gamma?,(q',\gamma,\delta),\varepsilon!)$ for some $q\in Q$, $(q',\gamma,\delta)\in Q_2$, $x\in\Sigma\cup\{\varepsilon\}$, and $\gamma\in\Gamma_{\mathcal{A}}$. By Observation 4.22 θ_1' is of the form $((q',\gamma,\delta),\varepsilon,\#?,q',\delta\gamma\#!)\theta_2'$ for some $\theta_2'\in T_{\mathcal{B}}^*$. Then $\eta=\gamma$ and $w=xw'$ for some $w'\in\Sigma^*$. Since $\theta'\in\Theta_{\mathcal{B}}^{\varepsilon}(q,w,\eta\#)$ it follows that $\theta_2'\in\Theta_{\mathcal{B}}^{\varepsilon}(q',w',\delta\eta\#)$. By the induction hypothesis there is a $\theta_1\in\overline{\Theta}_{\mathcal{A}}(q',w',\delta\eta)$ such that θ_2' extends θ_1.

Then let $\theta=\tau\theta_1$ with $\tau=(q,x,\mathrm{bottom}?,q',\mathrm{push}(\delta,f_{\mathrm{id}}))$. Thus, $\theta\in\overline{\Theta}_{\mathcal{A}}(q,w,\eta)$ and by definition of π it is clear that θ' extends θ.

Case 5: Let $\tau=(q,x,\gamma?,(q',\delta),\varepsilon!)$ for some $q\in Q$, $(q',\delta)\in Q_1$, $x\in\Sigma\cup\{\varepsilon\}$, and $\gamma\in\Gamma_{\mathcal{A}}$. By Observation 4.22 we must have that θ_1' is of the form $((q',\delta),\varepsilon,\#?,q',\delta\#!)\theta_2'$ for some $\theta_2'\in T_{\mathcal{B}}^*$. Then $\eta=\gamma$ and $w=xw'$ for some $w'\in\Sigma^*$. Since $\theta'\in\Theta_{\mathcal{B}}^{\varepsilon}(q,w,\eta\#)$ it follows that $\theta_2'\in\Theta_{\mathcal{B}}^{\varepsilon}(q',w',\delta\#)$. By the induction hypothesis there is a $\theta_1\in\overline{\Theta}_{\mathcal{A}}(q',w',\delta)$ such that θ_2' extends θ_1.

Then let $\theta=\tau\theta_1$ with $\tau=(q,x,\mathrm{bottom}?,q',\mathrm{stay}(\delta))$. Thus, $\theta\in\overline{\Theta}_{\mathcal{A}}(q,w,\eta)$ and by definition of π it is clear that θ' extends θ.

Now let $w\in\Sigma^*$ and $\theta'\in\Theta_{\mathcal{B}}^{\varepsilon}(w)$. By the construction of $\mathcal{B}$ we must have that θ' is of the form $(q_{0,\mathcal{B}},\varepsilon,\#?,q_0,\gamma_0\#!)\theta_1'$ where θ_1' is a (q_0,ε)-computation in $\Theta_{\mathcal{B}}^{\varepsilon}(q_0,w,\gamma_0\#)$. Then, by the statement just shown, there is a q_0-computation $\theta\in\overline{\Theta}_{\mathcal{A}}(q_0,w,\gamma_0)$ such that θ_1' extends θ. It follows that θ' τ_0-extends θ and therefore, $f_w(\theta)=\theta'$. Thus, f_w is surjective. ■

As a last step, before proving the main result of this section, we will show that f_w preserves the weight of each computation in its domain for every $w\in\Sigma^*$.

Lemma 4.24. For every $w\in\Sigma^*$, $\theta\in\Theta_{\mathcal{A}}(w)$ it holds that $\mathrm{wt}(\theta)=\mathrm{wt}_{\mathcal{B}}(f_w(\theta))$.

Proof. Let $w\in\Sigma^*$ and $\theta\in\Theta_{\mathcal{A}}(q_0,w,c_0)$. Furthermore, recall that $c_0=(\gamma_0,\mathrm{c})$ for some $\gamma_0\in\Gamma$ and $\tau_0=(q_{0,\mathcal{B}},\varepsilon,\#?,q_0,\gamma_0\#!)$. By Lemma 4.19 there is exactly one (q_0,ε)-computation $\theta'\in\Theta_{\mathcal{B}}^{\varepsilon}(q_0,w,\gamma_0\#)$ such that θ' extends θ and $\overline{\mathrm{wt}}(\theta')=\overline{\mathrm{wt}}(\theta)$. Therefore, $f_w(\theta)=\tau_0\theta'$. Since $\mathrm{wt}_{\mathcal{B}}(\tau_0)=1$, it follows that $\overline{\mathrm{wt}}(\theta')=\overline{\mathrm{wt}}(\tau_0\theta')$. By Lemma 4.7 it holds that $\mathrm{wt}(\theta)=\mathrm{wt}_{\mathcal{B}}(\tau_0\theta')$. Thus, $\mathrm{wt}(\theta)=\mathrm{wt}_{\mathcal{B}}(f_w(\theta))$. ■

With these results we can now prove the main issue of this section – Lemma 4.10, which stated that each (P^1,Σ,K)-recognizable language is also (PD,Σ,K)-recognizable with empty pushdown.

Proof of Lemma 4.10. Let $\mathcal{A}=(Q,\Sigma,c_0,q_0,Q_f,T,\mathrm{wt})$ be a (P^1,Σ,K)-automaton. By Lemma 4.12 we can assume that $\mathcal{A}$ is in test normal form. Then we construct a (PD,Σ,K)-automaton $\mathcal{B}=(Q_{\mathcal{B}},\Sigma,\#,q_{0,\mathcal{B}},\{f_{\mathcal{B}}\},T_{\mathcal{B}},\mathrm{wt}_{\mathcal{B}})$ such that $\mathcal{B}$ is the automaton induced by $\mathcal{A}$. Let, for every

$w \in \Sigma^*$, $f_w \colon \Theta_{\mathcal{A}}(w) \to \Theta^{\varepsilon}_{\mathcal{B}}(w)$ as in Definition 4.20. By Lemma 4.21 and 4.23 this function is injective and surjective. Moreover, by 4.24 for every $w \in \Sigma^*$, $\theta \in \Theta_{\mathcal{A}}(w)$ it holds that $\mathrm{wt}(\theta) = \mathrm{wt}_{\mathcal{B}}(f_w(\theta))$. Thus, for every $w \in \Sigma^*$ we have that $\Theta_{\mathcal{A}}(w)$ and $\Theta^{\varepsilon}_{\mathcal{B}}(w)$ are in a one-to-one correspondence. Therefore, for every $w \in \Sigma^*$ we obtain

$$\|\mathcal{A}\|(w) = \sum_{\theta \in \Theta_{\mathcal{A}}(w)} \mathrm{wt}(\theta) = \sum_{\theta \in \Theta_{\mathcal{A}}(w)} \mathrm{wt}_{\mathcal{B}}(f_w(\theta)) = \sum_{\theta' \in \Theta^{\varepsilon}_{\mathcal{B}}(w)} \mathrm{wt}_{\mathcal{B}}(\theta') = \|\mathcal{B}\|_{\varepsilon}(w).$$

Thus, $\|\mathcal{A}\| = \|\mathcal{B}\|_{\varepsilon}$. ■

4.2 (PD,Σ,K)-automata can be simulated by (P^1,Σ,K)-automata

The aim of this section is to show formally that each (PD,Σ,K)-automaton accepting with empty pushdown can be simulated by some (P^1,Σ,K)-automaton. That means we will show that for each (PD,Σ,K)-automaton $\mathcal{A}$ there exists a (P^1,Σ,K)-automaton $\mathcal{B}$ such that $\|\mathcal{A}\|_\varepsilon = \|\mathcal{B}\|$, which is stated by the following lemma.

Lemma 4.25. Let $r\colon \Sigma^* \to K$ be a weighted language. If r is (PD,Σ,K)-recognizable with empty pushdown, then r is (P^1,Σ,K)-recognizable.

To prove this lemma, we proceed as follows. In a first step, we give a definition of the (P^1,Σ,K)-automaton $\mathcal{B}$ which is induced by some (PD,Σ,K)-automaton $\mathcal{A}$. Afterwards, we define for every word $w \in \Sigma^*$ a function between computations on w of $\mathcal{A}$ and $\mathcal{B}$, and show that this function is injective, surjective and weight-preserving.

For the construction we have to take care of a few differences between (PD,Σ,K)-automata and (P^1,Σ,K)-automata. Since $\mathcal{A}$ can push in one step several symbols to the pushdown, we have to divide transitions which contain such an instruction into a sequence of transitions in $\mathcal{B}$. As $\mathcal{B}$ must not have an empty pushdown, it carries the additional symbol # at the bottom of the pushdown. Moreover, since $\mathcal{A}$ accepts with empty pushdown, $\mathcal{B}$ has to test bottom before switching into a final state.

Definition 4.26. Let $\mathcal{A} = (Q,\Sigma,c_0,q_0,Q_f,T,\mathrm{wt})$ be a (PD,Σ,K)-automaton. Furthermore, let $\Gamma_\mathcal{A}$ be the finite set of pushdown symbols occurring in transitions of $\mathcal{A}$ and let # be a symbol such that $\Gamma_\mathcal{A} \cap \{\#\} = \emptyset$. The (P^1,Σ,K)*-automaton induced by* $\mathcal{A}$ is the automaton $\mathcal{B} = (Q_\mathcal{B},\Sigma,c_{0,\mathcal{B}},q_{0,\mathcal{B}},\{f\},T_\mathcal{B},\mathrm{wt}_\mathcal{B})$ with some elements $q_{0,\mathcal{B}}, f \notin Q$ such that

- $Q_\mathcal{B} = \{q_{0,\mathcal{B}}, f\} \cup Q \cup Q_\Pi$, where $Q_\Pi = (Q \times \Pi \times [k])$, $\Pi = \{\pi \in \Gamma_\mathcal{A}^* \mid \text{there is a } \tau \in T \text{ containing } \pi! \text{ as instruction and } |\pi| \geq 2\}$, and $k = \max(\{|\pi| \mid \pi \in \Pi\})$,[4]
- $c_{0,\mathcal{B}} = (\#, \mathrm{c})$.

Moreover, $T_\mathcal{B}$ and $\mathrm{wt}_\mathcal{B}$ are defined as follows:

- The transition $(q_{0,\mathcal{B}}, \varepsilon, \mathrm{bottom}, q_0, \mathrm{push}(c_0, f_{\mathrm{id}}))$ with weight 1 is in $T_\mathcal{B}$.
- Let $\tau = (q,x,\gamma?,q',\pi!)$ be in T.
  - If $\pi = \varepsilon$, then the transition $\tau' = (q,x,\mathrm{top}=\gamma,q',\mathrm{pop})$ is in $T_\mathcal{B}$ and $\mathrm{wt}_\mathcal{B}(\tau') = \mathrm{wt}(\tau)$.
  - If $\pi = \delta$ for some $\delta \in \Gamma_\mathcal{A}$, then the transition $\tau' = (q,x,\mathrm{top}=\gamma,q',\mathrm{stay}(\delta))$ is in $T_\mathcal{B}$ and $\mathrm{wt}_\mathcal{B}(\tau') = \mathrm{wt}(\tau)$.

[4] Recall that we use as convention $\max(\emptyset) = 0$.

– If $\pi = \delta_1 \dots \delta_n$ with $n > 1, \delta_i \in \Gamma_{\mathcal{A}}, i \in [n]$, then the transition $\tau' = (q, x, \text{top} = \gamma, (q', \pi, n), \text{stay}(\delta_n))$ with $\text{wt}_{\mathcal{B}}(\tau') = \text{wt}(\tau)$ is in $T_{\mathcal{B}}$. Additionally, the transitions $((q', \pi, i), \varepsilon, \text{top} = \delta_i, (q', \pi, i-1), \text{push}(\delta_{i-1}, f_{\text{id}}))$ for $2 < i \leq n$ and the transition $((q', \pi, 2), \varepsilon, \text{top} = \delta_2, q', \text{push}(\delta_1, f_{\text{id}}))$, each with weight 1, are in $T_{\mathcal{B}}$.

– For every $q \in Q_f$ the transition $(q, \varepsilon, \text{bottom}, f, \text{stay}(\#))$ with weight 1 is in $T_{\mathcal{B}}$. □

To demonstrate this construction, for the sake of brevity and clarity, we introduce a new example. Intuitively, the now considered automaton discounts all words over $\Sigma = \{a, b\}$ which are of the form $a^n b a^{2n}$ for some $n \geq 0$ in the unital valuation monoid $\mathcal{K}^{\lambda}_{\text{disc}}$, the remaining words over Σ are mapped to $-\infty$.

Example 4.27. Let Σ be the alphabet $\Sigma = \{a, b\}$. We consider the $(\mathrm{PD}, \Sigma, \mathcal{K}^{\lambda}_{\text{disc}})$-automaton $\mathcal{A} = (\{q, f\}, \Sigma, B, q, \{f\}, T, \text{wt})$, where the set T contains the following transitions:

$\tau_1 = (q, a, B?, q, BAA!)$ and $\text{wt}(\tau_1) = 2$, $\quad \tau_2 = (q, a, A?, q, \varepsilon!)$ and $\text{wt}(\tau_2) = 2$,

$\tau_3 = (q, b, B?, q, A!)$ and $\text{wt}(\tau_3) = 1$, $\quad \tau_4 = (q, \varepsilon, A?, f, \varepsilon!)$ and $\text{wt}(\tau_4) = 0$.

By applying Definition 4.26 we obtain as the $(\mathrm{P}^1, \Sigma, K)$-automaton induced by $\mathcal{A}$ the automaton $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, c_{0,\mathcal{B}}, q_{0,\mathcal{B}}, \{f_{\mathcal{B}}\}, T_{\mathcal{B}}, \text{wt}_{\mathcal{B}})$ with some elements $q_{0,\mathcal{B}}$, $f_{\mathcal{B}} \notin \{q, f\}$ and where

- the initial configuration $c_{0,\mathcal{B}} = (\#, \mathrm{c})$ for $\# \notin \{A, B\}$, and
- $Q_{\mathcal{B}} = \{q_{0,\mathcal{B}}, f_{\mathcal{B}}\} \cup \{q, f\} \cup$
  $\{(q, BAA, 3), (q, BAA, 2), (q, BAA, 1), (f, BAA, 3), (f, BAA, 2), (f, BAA, 1)\}$.

Moreover, we have that $T_{\mathcal{B}}$ contains the following transitions which we want to explain shortly. For this consider the transition

$$\tau_1 = (q, a, B?, q, BAA!) \in T$$

which replaces the topmost pushdown symbol B by the symbol A and pushes additionally BA. Recall that $\text{wt}(\tau_1) = 2$. This pushdown behaviour can be simulated in $\mathcal{B}$ with a stay instruction followed by two push instructions. For that the transition

$$(q, a, \text{top} = B, (q, BAA, 3), \text{stay}(A))$$

with weight 2 as well as the transitions

$$((q, BAA, 3), \varepsilon, \text{top} = A, (q, BAA, 2), \text{push}(A, f_{\text{id}}))$$

and

$$((q, BAA, 2), \varepsilon, \text{top} = A, q, \text{push}(B, f_{\text{id}})),$$

both with weight 0, are in $T_{\mathcal{B}}$. For an illustration of the simulation of the pushdown behaviour of $\mathcal{A}$ by $\mathcal{B}$ in this concrete case see Figure 4.3.

The remaining transitions of $T_{\mathcal{B}}$ are constructed as described in Definition 4.26. □

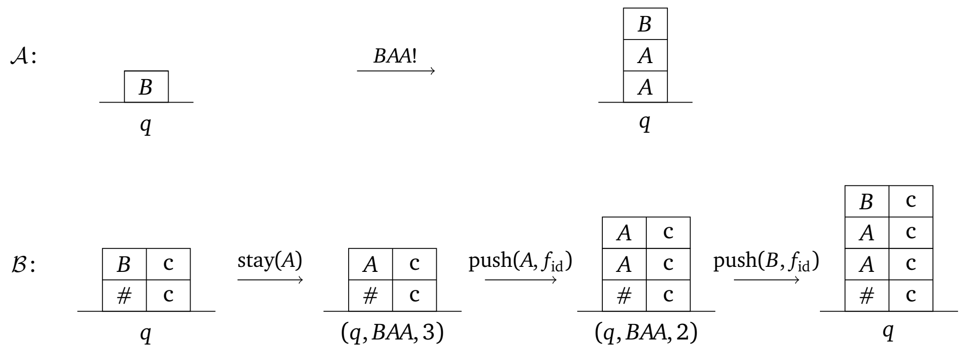


Figure 4.3: Simulation of a transition $\tau = (q, a, B?, q, BAA!)$ of the automaton $\mathcal{A}$ by the automaton $\mathcal{B}$.

For the remaining section let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T, \mathrm{wt})$ *be a* (PD, Σ, K)*-automaton and let* $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, c_{0,\mathcal{B}}, q_{0,\mathcal{B}}, \{f\}, T_{\mathcal{B}}, \mathrm{wt}_{\mathcal{B}})$ *be the* $(\mathrm{P}^1, \Sigma, K)$*-automaton induced by* $\mathcal{A}$.

Now we want to show that the weighted language recognized by $\mathcal{A}$ with empty pushdown is exactly the weighted language recognized by $\mathcal{B}$. This can be achieved in a way similar to the proof structure of Lemma 4.10. We will indicate a mapping φ between sequences of transitions from $\mathcal{A}$ and $\mathcal{B}$, respectively. From this mapping we can then derive a bijection between computations of $\mathcal{A}$ and computations of $\mathcal{B}$. In contrast to the function π in Section 4.1 there appears no “nondeterminism” in φ – from each sequence of transitions from $\mathcal{A}$ one can derive exactly one sequence of transitions from $\mathcal{B}$.

Definition 4.28. We define a mapping $\varphi\colon T \to T_{\mathcal{B}}^*$ for every $\tau \in T$ as follows:

- if $\tau = (q, x, \gamma?, q', \varepsilon!)$, then $\varphi(\tau) = (q, x, \mathrm{top} = \gamma, q', \mathrm{pop})$,
- if $\tau = (q, x, \gamma?, q', \delta!)$ for some $\delta \in \Gamma_{\mathcal{A}}$, then $\varphi(\tau) = (q, x, \mathrm{top} = \gamma, q', \mathrm{stay}(\delta))$, and
- if $\tau = (q, x, \gamma?, q', \pi!)$ for some $n \geq 2$, $\pi = \delta_1 \dots \delta_n$ with $\delta_i \in \Gamma_{\mathcal{A}}$, $i \in [n]$, then

$$\begin{aligned}\varphi(\tau) = (q, x, \mathrm{top} = \gamma, (q', \pi, n), \mathrm{stay}(\delta_n))&\\ ((q', \pi, n), \varepsilon, \mathrm{top} = \delta_n, (q', \pi, n-1), \mathrm{push}(\delta_{n-1}, f_{\mathrm{id}}))&\\ \vdots\qquad&\\ ((q', \pi, 2), \varepsilon, \mathrm{top} = \delta_2, q', \mathrm{push}(\delta_1, f_{\mathrm{id}})).&\end{aligned}$$

Then φ can be extended to a mapping $\varphi'\colon T^* \to T_{\mathcal{B}}^*$ such that $\varphi'(\varepsilon) = \varepsilon$ and $\varphi'(\tau_1 \dots \tau_n) = \varphi(\tau_1) \dots \varphi(\tau_n)$ for $n \geq 1$, $\tau_1, \dots, \tau_n \in T$. In the following we identify φ and φ'. □

Similar to π in the image of φ certain transitions of $\mathcal{B}$ are missing. For an example of this situation, regard the unique transition starting with the initial state of $\mathcal{B}$. To consider nevertheless computations from $\mathcal{B}$ we introduce the notion of extensions similar to Definition 4.17.

Definition 4.29. Let $\theta \in T^*$, $\vartheta_f = \{(q, \varepsilon, \text{bottom}, f, \text{stay}(\#)) \in T_{\mathcal{B}} \mid q \in Q_f\}$, and let $\tau_0 = (q_{0,\mathcal{B}}, \varepsilon, \text{bottom}, q_0, \text{push}(c_0, f_{\text{id}}))$. We say that a sequence $\theta' \in T_{\mathcal{B}}^*$ *extends* θ if $\theta' \in \varphi(\theta)\vartheta_f$ and τ_0-*extends* θ if $\theta' \in \tau_0\varphi(\theta)\vartheta_f$. □

Then we can prove the following lemma.

Lemma 4.30. Let $q \in Q$, $w \in \Sigma^*$, and $\eta \in \Gamma^*$. If θ is a (q, ε)-computation in $\Theta_{\mathcal{A}}^{\varepsilon}(q, w, \eta)$, then there is exactly one q-computation θ' in $\overline{\Theta}_{\mathcal{B}}(q, w, \eta\#)$ s.t. θ' extends θ. Moreover, $\overline{\text{wt}}(\theta) = \overline{\text{wt}}(\theta')$.

Proof. Let $q \in Q$, $w \in \Sigma^*$, $\eta \in \Gamma^*$, and $\theta \in \Theta_{\mathcal{A}}^{\varepsilon}(q, w, \eta)$. We prove this statement by induction on the length of θ.

First, let $\theta = \varepsilon$. As $\theta \in \Theta_{\mathcal{A}}^{\varepsilon}(q, w, \eta)$, this means $q \in Q_f$, $w = \varepsilon$ and $\eta = \varepsilon$. Moreover, $\varphi(\theta) = \varepsilon$. It remains to show that there is exactly one $\theta' \in \vartheta_f$ such that $\theta' \in \overline{\Theta}_{\mathcal{B}}(q, w, \eta\#)$. This is is uniquely given by $\theta' = (q, \varepsilon, \text{bottom}, f, \text{stay}(\#))$.

By construction, $\text{wt}_{\mathcal{B}}((q, \varepsilon, \text{bottom}, f, \text{stay}(\#))) = 1$. Therefore, $\overline{\text{wt}}(\theta') = \varepsilon = \overline{\text{wt}}(\theta)$.

Now, let $\theta = \tau\theta_1$ for some $\tau \in T$ and $\theta_1 \in T^*$. In the following, we consider a case distinction on τ.

Case 1: Let $\tau = (q, x, \gamma?, q', \varepsilon!)$ for some $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, and $\gamma \in \Gamma_{\mathcal{A}}$. Then $w = xw'$ for some $w' \in \Sigma^*$ and $\eta = \gamma\eta'$ for some $\eta' \in \Gamma_{\mathcal{A}}^*$. Since $\theta \in \Theta_{\mathcal{A}}^{\varepsilon}(q, w, \eta)$ it follows that $\theta_1 \in \Theta_{\mathcal{A}}^{\varepsilon}(q', w', \eta')$. By the induction hypothesis there is exactly one $\theta_1' \in \overline{\Theta}_{\mathcal{B}}(q', w', \eta'\#)$ such that θ_1' extends θ_1. Then τ' is uniquely determined by $\tau' = \varphi(\tau) = (q, x, \text{top} = \gamma, q', \text{pop})$ and $\tau'\theta_1' \in \overline{\Theta}_{\mathcal{B}}(q, w, \eta\#)$.

Furthermore, we have that

$$\overline{\text{wt}}(\tau\theta_1) = \overline{\text{wt}}(\tau)\overline{\text{wt}}(\theta_1) \overset{\text{(IH)}}{=} \overline{\text{wt}}(\tau)\overline{\text{wt}}(\theta_1') \overset{(*)}{=} \overline{\text{wt}}(\tau')\overline{\text{wt}}(\theta_1') = \overline{\text{wt}}(\tau'\theta_1'),$$

where $(*)$ holds since $\text{wt}(\tau) = \text{wt}_{\mathcal{B}}(\tau')$ by construction.

Case 2: Let $\tau = (q, x, \gamma?, q', \delta!)$ for some $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, and γ, $\delta \in \Gamma_{\mathcal{A}}$. Then $w = xw'$ for some $w' \in \Sigma^*$ and $\eta = \gamma\eta'$ for some $\eta' \in \Gamma_{\mathcal{A}}^*$. Since $\theta \in \Theta_{\mathcal{A}}^{\varepsilon}(q, w, \eta)$ it follows that $\theta_1 \in \Theta_{\mathcal{A}}^{\varepsilon}(q', w', \delta\eta')$. By the induction hypothesis there is exactly one $\theta_1' \in \overline{\Theta}_{\mathcal{B}}(q', w', \delta\eta'\#)$ such that θ_1' extends θ_1. Then τ' is uniquely determined by $\tau' = \varphi(\tau) = (q, x, \text{top} = \gamma, q', \text{stay}(\delta))$ and $\tau'\theta_1' \in \overline{\Theta}_{\mathcal{B}}(q, w, \eta\#)$.

Furthermore, we have that

$$\overline{\text{wt}}(\tau\theta_1) = \overline{\text{wt}}(\tau)\overline{\text{wt}}(\theta_1) \overset{\text{(IH)}}{=} \overline{\text{wt}}(\tau)\overline{\text{wt}}(\theta_1') \overset{(*)}{=} \overline{\text{wt}}(\tau')\overline{\text{wt}}(\theta_1') = \overline{\text{wt}}(\tau'\theta_1'),$$

where $(*)$ holds since $\mathrm{wt}(\tau) = \mathrm{wt}_{\mathcal{B}}(\tau')$ by construction.

Case 3: Let $\tau = (q, x, \gamma?, q', \pi!)$ for some $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, $\gamma \in \Gamma_{\mathcal{A}}$, and for some $n \geq 2$, $\pi = \delta_1 \dots \delta_n$ with $\delta_i \in \Gamma_{\mathcal{A}}$, $i \in [n]$. Then $w = xw'$ for some $w' \in \Sigma^*$ and $\eta = \gamma\eta'$ for some $\eta' \in \Gamma_{\mathcal{A}}^*$. Since $\theta \in \Theta^{\varepsilon}_{\mathcal{A}}(q, w, \eta)$ it follows that $\theta_1 \in \Theta^{\varepsilon}_{\mathcal{A}}(q', w', \pi\eta')$. By the induction hypothesis there is exactly one $\theta_1' \in \overline{\Theta}_{\mathcal{B}}(q', w', \pi\eta'\#)$ such that θ_1' extends θ_1. Then

$$\begin{aligned}\tau' = \varphi(\tau) = &(q, x, \mathrm{top} = \gamma, (q', \pi, n), \mathrm{stay}(\delta_n))\\ &((q', \pi, n), \varepsilon, \mathrm{top} = \delta_n, (q', \pi, n-1), \mathrm{push}(\delta_{n-1}, f_{\mathrm{id}}))\\ &\quad\vdots\\ &((q', \pi, 2), \varepsilon, \mathrm{top} = \delta_2, q', \mathrm{push}(\delta_1, f_{\mathrm{id}}))\end{aligned}$$

and $\tau'\theta_1' \in \overline{\Theta}_{\mathcal{B}}(q, w, \eta\#)$.

Furthermore, we have that

$$\overline{\mathrm{wt}}(\tau\theta_1) = \overline{\mathrm{wt}}(\tau)\overline{\mathrm{wt}}(\theta_1) \overset{\text{(IH)}}{=} \overline{\mathrm{wt}}(\tau)\overline{\mathrm{wt}}(\theta_1') \overset{(*)}{=} \overline{\mathrm{wt}}(\tau')\overline{\mathrm{wt}}(\theta_1') = \overline{\mathrm{wt}}(\tau'\theta_1'),$$

where $(*)$ holds since by construction $\mathrm{wt}(\tau) = \mathrm{wt}_{\mathcal{B}}((q, x, \mathrm{top} = \gamma, (q', \pi, n), \mathrm{stay}(\delta_n)))$ and the remaining transitions of τ' are of weight 1. ■

As a consequence of Lemma 4.30 we can now define for every word $w \in \Sigma^*$ a function g_w which assigns to each computation from $\Theta^{\varepsilon}_{\mathcal{A}}(w)$ a computation from $\Theta_{\mathcal{B}}(w)$.

Definition 4.31. For every $w \in \Sigma^*$ let

$$g_w : \Theta^{\varepsilon}_{\mathcal{A}}(w) \to \Theta_{\mathcal{B}}(w)$$

be the function which maps each $\theta \in \Theta^{\varepsilon}_{\mathcal{A}}(w)$ to the uniquely determined computation $\theta' \in \Theta_{\mathcal{B}}(w)$ that τ_0-extends θ. □

Note that this function is indeed well defined by Lemma 4.30 and since τ_0 is the uniquely determined transition $(q_{0,\mathcal{B}}, \varepsilon, \mathrm{bottom}, q_0, \mathrm{push}(c_0, f_{\mathrm{id}}))$. It remains to show that, for every $w \in \Sigma^*$, g_w is injective, surjective, and preserves the weight of a computation in K.

Let us start with proving the injectivity of g_w for each $w \in \Sigma^*$.

Lemma 4.32. For every $w \in \Sigma^*$ it holds that g_w is injective.

Proof. Let $t \in T_{\mathcal{B}}^*$. We show the lemma by proving the following property of t:

Let $t_1, t_2 \in T^*$. If $t = \varphi(t_1) = \varphi(t_2)$, then $t_1 = t_2$.

We prove this statement by induction on the length of t.

As induction basis let $|t| = 0$. Then $t = \varepsilon$ and it follows that $t_1 = t_2 = \varepsilon$.

Now let $|t| = n + 1$ for some $n \geq 0$ and assume that the property holds for every word $u \in T_{\mathcal{B}}^*$ with $|u| \leq n$. Let $t = \tau t'$ for some $\tau \in T_{\mathcal{B}}$, $t' \in T_{\mathcal{B}}^*$, and let $t_1, t_2 \in T^*$ such that $\tau t' = \varphi(t_1) = \varphi(t_2)$. In the following, we distinguish two cases of τ.

Case 1: Let $\tau = (q, x, \mathrm{top} = \gamma, q', f)$ for some $q, q' \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, $\gamma \in \Gamma_{\mathcal{A}}$, and $f \in \{\mathrm{pop}\} \cup \{\mathrm{stay}(\delta) \mid \delta \in \Gamma_{\mathcal{A}}\}$. Then $t_1 = \tau_1 t_1'$ and $t_2 = \tau_2 t_2'$ for some $\tau_1, \tau_2 \in T$ such that $\varphi(\tau_1) = \varphi(\tau_2) = \tau$ and, by definition of π, $\tau_1 = \tau_2$ since all pieces of information of τ_1 and τ_2 are coded into τ.

Also, it follows that $t' = \varphi(t_1') = \varphi(t_2')$ and, by induction hypothesis, $t_1' = t_2'$. Thus, $t_1 = t_2$.

Case 2: Let $\tau = (q, x, \mathrm{top} = \gamma, (q', \pi, n), \mathrm{stay}(\delta_n))$ for some $n \geq 2$, $q \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, $\gamma \in \Gamma_{\mathcal{A}}$, $\pi = \delta_1 \ldots \delta_n$ with $\delta_1, \ldots, \delta_n \in \Gamma_{\mathcal{A}}$, and $(q', \pi, n) \in Q_\Pi$. Then we must have that $t' = \rho_n \ldots \rho_2 t''$ with

$$\begin{aligned}
\rho_n &= ((q', \pi, n), \varepsilon, \mathrm{top} = \delta_n, (q', \pi, n-1), \mathrm{push}(\delta_{n-1}, f_{\mathrm{id}})),\\
&\quad\vdots\\
\rho_2 &= ((q', \pi, 2), \varepsilon, \mathrm{top} = \delta_2, q', \mathrm{push}(\delta_1, f_{\mathrm{id}})),
\end{aligned}$$

since τ only occurs followed by these transitions in the image of φ.

Furthermore, $t_1 = \tau_1 t_1'$ and $t_2 = \tau_2 t_2'$ for some $\tau_1, \tau_2 \in T$ such that $\varphi(\tau_1) = \varphi(\tau_2) = \tau \rho_n \ldots \rho_2$. By definition of φ, we have that $\tau_1 = \tau_2$.

Also, it follows that $t'' = \varphi(t_1') = \varphi(t_2')$ and, by induction hypothesis, $t_1' = t_2'$. Thus, $t_1 = t_2$.

Note that we need not to consider other forms of τ, since only the mentioned cases are in the image of φ.

Now let $w \in \Sigma^*$. We prove by contraposition that g_w is injective. For this, we consider two computations $\theta_1, \theta_2 \in \Theta_{\mathcal{A}}^{\varepsilon}(w)$ such that $g_w(\theta_1) = g_w(\theta_2)$. By definition of g_w it follows that $\tau_0 \varphi(\theta_1) \vartheta_f \cap \tau_0 \varphi(\theta_2) \vartheta_f \neq \emptyset$. Then $\varphi(\theta_1) = \varphi(\theta_2)$. Therefore, there is a $t \in T_{\mathcal{B}}^*$ with $t = \varphi(\theta_1) = \varphi(\theta_2)$ and it follows that $\theta_1 = \theta_2$. Thus, g_w is injective. ■

For the next lemma, we need to analyse the structure of computations of $\mathcal{B}$ and obtain the following observation:

Observation 4.33. Let $n \in \mathbb{N}$, $w \in \Sigma^*$, and $\theta \in \Theta_{\mathcal{B}}(w)$ with $\theta = \tau_1 \ldots \tau_n$, where $\tau_1, \ldots, \tau_n \in T_{\mathcal{B}}$. For every $m \in \mathbb{N}$, $i \in [n - m]$, $q \in Q$, $x \in \Sigma \cup \{\varepsilon\}$, $\gamma \in \Gamma_{\mathcal{A}}$, $\pi = \delta_1 \ldots \delta_m$ with $\delta_1, \ldots, \delta_m \in$

$\Gamma_{\mathcal{A}}$, and $(q',\pi,2),\dots,(q',\pi,m)\in Q_\Pi$ we have that

$$\begin{aligned} & \tau_i = (q,x,\mathrm{top}=\gamma,(q',\pi,m),\mathrm{stay}(\delta_m)) \\ \text{iff}\quad & \tau_{i+1} = ((q',\pi,m),\varepsilon,\mathrm{top}=\delta_m,(q',\pi,m-1),\mathrm{push}(\delta_{m-1},f_{\mathrm{id}})) \\ & \vdots \\ \text{iff}\quad & \tau_{i+m-1} = ((q',\pi,2),\varepsilon,\mathrm{top}=\delta_2,q',\mathrm{push}(\delta_1,f_{\mathrm{id}})). \end{aligned}$$

Now we can proceed with proving the surjectivity of g_w for every $w\in\Sigma^*$.

Lemma 4.34. For every $w\in\Sigma^*$ it holds that g_w is surjective.

Proof. Let $\theta'\in T_{\mathcal{B}}^*$. We show this lemma by proving the following property of θ' for all $n\in\mathbb{N}$:

Let $q\in Q$, $w\in\Sigma^*$, and $\eta\in\Gamma^*$. If θ' is a q-computation in $\overline{\Theta}_{\mathcal{B}}(q,w,\eta\#)$ of length n, then there is a (q,ε)-computation θ in $\Theta^{\varepsilon}_{\mathcal{A}}(q,w,\eta)$ such that θ' extends θ.

We prove this statement by induction on the length n of θ'. As there is no q-computation with length 0 in $\overline{\Theta}_{\mathcal{B}}(q,w,\eta\#)$, let $n=1$ and $\theta'=(q,\varepsilon,\mathrm{bottom},f,\mathrm{stay}(\#))$. Then $q\in Q_f$, $w=\varepsilon$, $\eta=\varepsilon$, and $\theta=\varepsilon$. Thus, $\theta\in\Theta^{\varepsilon}_{\mathcal{A}}(q,w,\eta)$ and θ' extends θ.

Now, let $|\theta'|=n+1$ for some $n\geq 1$ and $\theta'=\tau'\theta'_1$ for some $\tau'\in T_{\mathcal{B}}$ and such that $\theta'\in\overline{\Theta}_{\mathcal{B}}(q,w,\eta\#)$. Assume that the property holds for every word $u\in T_{\mathcal{B}}^*$ with $|u|\leq n$. In the following, we distinguish three cases of τ'.

Case 1: Let $\tau'=(q,x,\mathrm{top}=\gamma,q',\mathrm{pop})$ for some $q,q'\in Q$, $x\in\Sigma\cup\{\varepsilon\}$, and $\gamma\in\Gamma_{\mathcal{A}}$. Then $w=xw'$ for some $w'\in\Sigma^*$ and $\eta=\gamma\eta'$ for some $\eta'\in\Gamma_{\mathcal{A}}^*$. Since $\theta'\in\overline{\Theta}_{\mathcal{B}}(q,w,\eta\#)$ it follows that $\theta'_1\in\overline{\Theta}_{\mathcal{B}}(q',w',\eta'\#)$. By induction hypothesis there is a $\theta_1\in\Theta^{\varepsilon}_{\mathcal{A}}(q',w',\eta')$ such that θ'_1 extends θ_1.

Then let $\theta=\tau\theta_1$ with $\tau=(q,x,\gamma?,q',\varepsilon!)$. Thus, $\theta\in\Theta^{\varepsilon}_{\mathcal{A}}(q,w,\eta)$ and by definition of φ it is clear that θ' extends θ.

Case 2: Let $\tau'=(q,x,\mathrm{top}=\gamma,q',\mathrm{stay}(\delta))$ for some $q,q'\in Q$, $x\in\Sigma\cup\{\varepsilon\}$, and $\gamma,\delta\in\Gamma_{\mathcal{A}}$. Then $w=xw'$ for some $w'\in\Sigma^*$ and $\eta=\gamma\eta'$ for some $\eta'\in\Gamma_{\mathcal{A}}^*$. Since $\theta'\in\overline{\Theta}_{\mathcal{B}}(q,w,\eta\#)$ it follows that $\theta'_1\in\overline{\Theta}_{\mathcal{B}}(q',w',\delta\eta'\#)$. By induction hypothesis there is a $\theta_1\in\Theta^{\varepsilon}_{\mathcal{A}}(q',w',\delta\eta')$ such that θ'_1 extends θ_1.

Then let $\theta=\tau\theta_1$ with $\tau=(q,x,\gamma?,q',\delta!)$. Thus, $\theta\in\Theta^{\varepsilon}_{\mathcal{A}}(q,w,\eta)$ and by definition of φ it is clear that θ' extends θ.

Case 3: Let $\tau=(q,x,\mathrm{top}=\gamma,(q',\pi,n),\mathrm{stay}(\delta_n))$ for some $n\geq 2$, $q\in Q$, $x\in\Sigma\cup\{\varepsilon\}$, $\gamma\in\Gamma_{\mathcal{A}}$, $\pi=\delta_1\dots\delta_n$ with $\delta_1,\dots,\delta_n\in\Gamma_{\mathcal{A}}$, and $(q',\pi,n)\in Q_\Pi$. By Observation 4.33 we must have that θ_1' is of the form

$$\begin{gathered}((q', \pi, n), \varepsilon, \mathrm{top} = \delta_n, (q', \pi, n-1), \mathrm{push}(\delta_{n-1}, f_{\mathrm{id}}))\\ \vdots \\ ((q', \pi, 2), \varepsilon, \mathrm{top} = \delta_2, q', \mathrm{push}(\delta_1, f_{\mathrm{id}}))\theta_2'\end{gathered}$$

for some $\theta_2' \in T_{\mathcal{B}}^*$. Furthermore, $\eta = \gamma\eta'$ for some $\eta' \in \Gamma_{\mathcal{A}}^*$ and $w = xw'$ for $w' \in \Sigma^*$. Since $\theta' \in \overline{\Theta}_{\mathcal{B}}(q, w, \eta\#)$ it follows that $\theta_2' \in \overline{\Theta}_{\mathcal{B}}(q', w', \pi\eta'\#)$. By induction hypothesis there is a $\theta_1 \in \Theta_{\mathcal{A}}^{\varepsilon}(q', w', \pi\eta')$ such that θ_2' extends θ_1.

Then let $\theta = \tau\theta_1$ with $\tau = (q, x, \gamma?, q', \pi!)$. Thus, $\theta \in \Theta_{\mathcal{A}}^{\varepsilon}(q, w, \eta)$ and by definition of φ it is clear that θ' extends θ.

Now let $w \in \Sigma^*$ and $\theta' \in \Theta_{\mathcal{B}}(w)$. By the construction of $\mathcal{B}$ we must have that θ' is of the form $(q_{0,\mathcal{B}}, \varepsilon, \mathrm{bottom}, q_0, \mathrm{push}(c_0, f_{\mathrm{id}}))\theta_1'$ where θ_1' is a q_0-computation in $\overline{\Theta}_{\mathcal{B}}(q_0, w, c_0\#)$. Then, by the statement just shown, there is a (q_0, ε)-computation $\theta \in \Theta_{\mathcal{A}}^{\varepsilon}(q_0, w, c_0)$ such that θ_1' extends θ. It follows that θ' τ_0-extends θ and therefore, $g_w(\theta) = \theta'$. Thus, g_w is surjective. ■

As a last step, before proving the main result of this section, we will show that g_w preserves the weight of each computation in its domain for every $w \in \Sigma^*$.

Lemma 4.35. For every $w \in \Sigma^*$, $\theta \in \Theta_{\mathcal{A}}^{\varepsilon}(w)$ it holds that $\mathrm{wt}(\theta) = \mathrm{wt}_{\mathcal{B}}(g_w(\theta))$.

Proof. Let $w \in \Sigma^*$ and $\theta \in \Theta_{\mathcal{A}}^{\varepsilon}(q_0, w, c_0)$. Recall that $\tau_0 = (q_{0,\mathcal{B}}, \varepsilon, \mathrm{bottom}, q_0, \mathrm{push}(c_0, f_{\mathrm{id}}))$. By Lemma 4.30 there is exactly one q_0-computation $\theta' \in \overline{\Theta}_{\mathcal{B}}(q_0, w, c_0\#)$ such that θ' extends θ and $\overline{\mathrm{wt}}(\theta') = \overline{\mathrm{wt}}(\theta)$. Therefore, $g_w(\theta) = \tau_0\theta'$. Since $\mathrm{wt}_{\mathcal{B}}(\tau_0) = 1$, it follows that $\overline{\mathrm{wt}}(\theta') = \overline{\mathrm{wt}}(\tau_0\theta')$. By Lemma 4.7 it holds that $\mathrm{wt}(\theta) = \mathrm{wt}_{\mathcal{B}}(\tau_0\theta')$. Thus, $\mathrm{wt}(\theta) = \mathrm{wt}_{\mathcal{B}}(g_w(\theta))$. ■

With these results we can now prove the main issue of this section – Lemma 4.25 which stated that each weighted language that is (PD, Σ, K)-recognizable with empty pushdown is also $(\mathrm{P}^1, \Sigma, K)$-recognizable.

Proof of Lemma 4.25. Let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T, \mathrm{wt})$ be a (PD, Σ, K)-automaton. We construct a $(\mathrm{P}^1, \Sigma, K)$-automaton $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, c_{0,\mathcal{B}}, q_{0,\mathcal{B}}, \{f\}, T_{\mathcal{B}}, \mathrm{wt}_{\mathcal{B}})$ such that $\mathcal{B}$ is the automaton induced by $\mathcal{A}$. Let, for every $w \in \Sigma^*$, $g_w \colon \Theta_{\mathcal{A}}^{\varepsilon}(w) \to \Theta_{\mathcal{B}}(w)$ as in Definition 4.31. By Lemma 4.32 and 4.34 this function is injective and surjective. Moreover, by Lemma 4.35 for every $w \in \Sigma^*$ and $\theta \in \Theta_{\mathcal{A}}^{\varepsilon}(w)$ it holds that $\mathrm{wt}(\theta) = \mathrm{wt}_{\mathcal{B}}(g_w(\theta))$. Thus, for every $w \in \Sigma^*$ we have that $\Theta_{\mathcal{A}}^{\varepsilon}(w)$ and $\Theta_{\mathcal{B}}(w)$ are in a one-to-one correspondence. Therefore, for every $w \in \Sigma^*$ we obtain

$$\|\mathcal{A}\|_{\varepsilon}(w) = \sum_{\theta \in \Theta_{\mathcal{A}}^{\varepsilon}(w)} \mathrm{wt}(\theta) = \sum_{\theta \in \Theta_{\mathcal{A}}^{\varepsilon}(w)} \mathrm{wt}_{\mathcal{B}}(g_w(\theta)) = \sum_{\theta' \in \Theta_{\mathcal{B}}(w)} \mathrm{wt}_{\mathcal{B}}(\theta') = \|\mathcal{B}\|(w).$$

Thus, $\|\mathcal{A}\|_{\varepsilon} = \|\mathcal{B}\|$. ■

5 Separating the weights from an (S,Σ,K)-automaton

In this chapter we will characterize the weighted language recognized by an (S,Σ,K)-automaton as the image of the language recognized by an unweighted (S,Δ)-automaton under a weighted alphabetic morphism. For this, we first recall from [DV13] the concept of (weighted) alphabetic morphism.

5.1 Monomes and alphabetic morphisms

In this section we define monomes, alphabetic morphisms and related concepts. Weighted languages can be considered as formal power series – each weighted language is a (possibly infinite) sum of monomials. In this sense, monomes[1] are the "simplest" series, which associate a non zero value to at most one word.

Definition 5.1. A mapping $r\colon \Sigma^* \to K$ is called a *monome* if $\mathrm{supp}(r)$ is empty or a singleton. If $\mathrm{supp}(r) = \{w\}$, then we also write $r(w).w$ instead of r. We let $K[\Sigma \cup \{\varepsilon\}]$ denote the set of all monomes with support in $\Sigma \cup \{\varepsilon\}$. □

Definition 5.2. Let Δ be an alphabet and let $h\colon \Delta \to K[\Sigma \cup \{\varepsilon\}]$ be a mapping. Then the *alphabetic morphism induced by h* is the mapping $h'\colon \Delta^* \to K\langle\langle\Sigma^*\rangle\rangle$ such that for every $n \geq 0$, $\delta_1,\dots,\delta_n \in \Delta$ with $h(\delta_i) = a_i.y_i$, $i \in [n]$, we have $h'(\delta_1\dots\delta_n) = \mathrm{val}(a_1\dots a_n).y_1\dots y_n$. □

Note that $h'(v)$ is a monome for every $v \in \Delta^*$, and $h'(\varepsilon) = 1.\varepsilon$. If $L \subseteq \Delta^*$ such that the family $(h'(v) \mid v \in L)$ is locally finite or if K is complete, we let $h'(L) = \sum_{v\in L} h'(v)$. Note that whenever we write $h'(L)$ we require that $(h'(v) \mid v \in L)$ is locally finite in the case that K is not complete. In the sequel we will use the following convention. If we write "alphabetic morphism $h\colon \Delta \to K[\Sigma \cup \{\varepsilon\}]$", then we mean the alphabetic morphism induced by h. Furthermore, if $K = \mathbb{B}$ then sometimes in this work we write "$h(a) = b$" instead of "$h(a) = 1.b$".

Next we define a special case of alphabetic morphisms.

[1] Note that we write in the following *monome* instead of *monomial* since this name has become established in the literature on unital valuation monoids.

Definition 5.3. Let $K = \mathbb{B}$ and let $h\colon \Delta \to K[\Sigma \cup \{\varepsilon\}]$. If for every $\delta \in \Delta$ the support of $h(\delta)$ is $\{\sigma\}$ for some $\sigma \in \Sigma$, then we call h' a *letter-to-letter morphism*. □

Note that in this case the alphabetic morphism induced by h has the property that for every $v \in \Delta^*$, $\operatorname{supp}(h'(v))$ contains at most one element and if $\operatorname{supp}(h'(v)) = \{w\}$ for some $w \in \Sigma^*$, then the lengths of w and v are equal.

If the alphabetic morphism induced by $h\colon \Delta \to K[\Sigma \cup \{\varepsilon\}]$ is a letter-to-letter morphism, then we often write $h\colon \Delta \to K[\Sigma]$.

5.2 Separating the weights

The aim of this section is to prove the following theorem.

Theorem 5.4. For every $r \in K\langle\langle\Sigma^*\rangle\rangle$ the following two statements are equivalent:

(i) r is (S,Σ,K)-recognizable.

(ii) There are an alphabet Δ, an unambiguous ε-free (S,Δ)-automaton $\mathcal{B}$, and an alphabetic morphism $h : \Delta \to K[\Sigma \cup \{\varepsilon\}]$ such that $r = h(L(\mathcal{B}))$.

This theorem follows directly from Lemmas 5.5 and 5.7. The first one generalizes [DV13, Lemma 3].

Lemma 5.5. Let $r \in K\langle\langle\Sigma^*\rangle\rangle$. If r is (S,Σ,K)-recognizable, then there are an alphabet Δ, an unambiguous ε-free (S,Δ)-automaton $\mathcal{B}$, and an alphabetic morphism $h : \Delta \to K[\Sigma \cup \{\varepsilon\}]$ such that $r = h(L(\mathcal{B}))$.

Proof. Let $\mathcal{A} = (Q,\Sigma,c_0,q_0,Q_f,T,\mathrm{wt})$ be an (S,Σ,K)-automaton. We construct the (S,T)-automaton $\mathcal{B} = (Q,T,c_0,q_0,Q_f,T')$ and the mapping $h\colon T \to K[\Sigma \cup \{\varepsilon\}]$ such that, if $\tau = (q,x,p,q',f)$ is in T, then (q,τ,p,q',f) is in T' and we define $h(\tau) = \mathrm{wt}(\tau).x$. Obviously, $\mathcal{B}$ is unambiguous and ε-free.

Now let $w \in \Sigma^*$ and $\theta = \tau_1 \dots \tau_n \in \Theta_{\mathcal{A}}(w)$ for some $n \in \mathbb{N}$, $\tau_1, \dots \tau_n \in T$. By definition of h, we have that $h(\theta) = \mathrm{val}(\mathrm{wt}(\tau_1)\dots\mathrm{wt}(\tau_n)).w$. Hence $\mathrm{wt}(\theta) = h(\theta)(w)$. Also, by definition of (S,Σ,K)-automata, the set $\Theta_{\mathcal{A}}(w)$ is finite if K is not complete. Thus the family $(h(\theta) \mid \theta \in L(\mathcal{B}))$ is locally finite if K is not complete. Then, for every $w \in \Sigma^*$, we have

$$\begin{aligned}\|\mathcal{A}\|(w) &= \textstyle\sum_{\theta\in\Theta_{\mathcal{A}}(w)} \mathrm{wt}(\theta) = \sum_{\theta\in\Theta_{\mathcal{A}}(w)} h(\theta)(w)\\ &\overset{(*)}{=} \textstyle\sum_{\theta\in L(\mathcal{B})} h(\theta)(w) = \big(\sum_{\theta\in L(\mathcal{B})} h(\theta)\big)(w) = h(L(\mathcal{B}))(w),\end{aligned}$$

where $(*)$ holds because for every $\theta \in L(\mathcal{B})$ with $\theta \notin \Theta_{\mathcal{A}}(w)$, we have $h(\theta)(w) = 0$ and due to the fact that $\sum_{\theta\in L(\mathcal{B}),\ \theta\notin\Theta_{\mathcal{A}}(w)} 0 = 0$. Thus $\|\mathcal{A}\| = h(L(\mathcal{B}))$. ■

Example 5.6. Recall the unital valuation monoid $\mathcal{K}^{\lambda}_{\mathrm{disc}} = (\tilde{\mathbb{R}}, \max, \mathrm{val}^{\lambda}_{\mathrm{disc}}, -\infty, 0)$ from Example 2.5 for $\lambda = 0.5$ and let $\mathcal{A} = (\{q, f\}, \Sigma, (\$, c), q, \{f\}, T, \mathrm{wt})$ be the $(\mathrm{P}^1, \Sigma, \mathcal{K}^{\lambda}_{\mathrm{disc}})$-automaton from Example 3.31 with $\Sigma = \{a, b\}$. The application of the construction yields the (P^1, T)-automaton $\mathcal{B} = (\{q, f\}, T, (\$, c), q, \{f\}, T_{\mathcal{B}})$ with the set $T_{\mathcal{B}}$ of transitions

$$\begin{aligned}
&(q, \tau_1, \mathrm{bottom}, q, \mathrm{push}(B, f_{\mathrm{id}})), && (q, \tau_4, \mathrm{bottom}, q, \mathrm{push}(A, f_{\mathrm{id}})),\\
&(q, \tau_2, \mathrm{top} = A, q, \mathrm{pop}), && (q, \tau_5, \mathrm{top} = B, q, \mathrm{pop}),\\
&(q, \tau_3, \mathrm{top} = B, q, \mathrm{push}(B, f_{\mathrm{id}})), && (q, \tau_6, \mathrm{top} = A, q, \mathrm{push}(A, f_{\mathrm{id}})),
\end{aligned}$$

$$(q, \tau_7, \mathrm{bottom}, f, \mathrm{stay}(\$)),$$

and the alphabetic morphism $h\colon T \to \mathcal{K}^{\lambda}_{\mathrm{disc}}[\Sigma \cup \{\varepsilon\}]$ such that

$$\begin{aligned}
h(\tau_1) &= 2.a & h(\tau_4) &= 1.b\\
h(\tau_2) &= 2.a & h(\tau_5) &= 1.b\\
h(\tau_3) &= 2.a & h(\tau_6) &= 1.b\\
& & h(\tau_7) &= 0.\varepsilon\ .
\end{aligned}$$

Again, we consider the computation $\theta = \tau_1\tau_3\tau_5\tau_5\tau_4\tau_2\tau_7$ of $\mathcal{A}$ on the word $w = aabbba$ (see Example 3.31). It is easy to see that θ can be recognized by $\mathcal{B}$ and, moreover,

$$\begin{aligned}
h(\tau_1\tau_3\tau_5\tau_5\tau_4\tau_2\tau_7) &= \mathrm{val}^{\lambda}_{\mathrm{disc}}(2211120)\ .\ w\\
&= (2\lambda^0 + 2\lambda^1 + 1\lambda^2 + 1\lambda^3 + 1\lambda^4 + 2\lambda^5 + 0\lambda^6)\ .\ w\\
&= 3.5\ .\ w\ .
\end{aligned}$$

Hence, $\|\mathcal{A}\|(w) = \mathrm{wt}(\theta) = h(\theta)(w)$ as $\mathcal{A}$ is unambiguous. □

In the next lemma we will prove the converse of Lemma 5.5 also for unambiguous (S, Δ)-automata which are not ε-free. We have to change the construction of the corresponding [DV13, Lemma 4].

Lemma 5.7. For every alphabet Δ, unambiguous (S, Δ)-automaton $\mathcal{B}$, and alphabetic morphism $h : \Delta \to K[\Sigma \cup \{\varepsilon\}]$ the weighted language $h(L(\mathcal{B}))$ is (S, Σ, K)-recognizable.

Proof. Let $\mathcal{B} = (Q, \Delta, c_0, q_0, Q_f, T)$ be an unambiguous (S, Δ)-automaton and let $h\colon \Delta \to K[\Sigma \cup \{\varepsilon\}]$ be an alphabetic morphism. Moreover, recall our assumption that the family $(h(v) \mid v \in L(\mathcal{B}))$ is locally finite if K is not complete. We will construct an (S, Σ, K)-automaton $\mathcal{A}$ such that $\|\mathcal{A}\| = h(L(\mathcal{B}))$.

The following construction employs a similar technique of coding the preimage of h into the set of states as in [DV13, Lemma 4] in order to handle non-injectivity of h appropriately. However, we have to modify the construction slightly, because of the following reason. In each derivation of the automaton constructed by [DV13], the target state of a transition is enriched by the preimage of the symbol that is read by the subsequent transition. Hence, in the beginning of each derivation an ε-transition is required which guesses a preimage of the first input symbol; the pushdown remains unchanged. As arbitrary storage types in general do not have an identity instruction, we cannot use this construction. However, we change it as follows: In each transition of our constructed automaton the target state encodes a preimage of the symbol which is read by this transition; therefore we do not need an additional transition.

We construct the (S,Σ,K)-automaton $\mathcal{A} = (Q', \Sigma, c_0, q'_0, Q'_f, T', \mathrm{wt})$ where $Q' = \{q'_0\} \cup (\Delta \cup \{\varepsilon\}) \times Q$ with some element q'_0 with $q'_0 \notin (\Delta \cup \{\varepsilon\}) \times Q$, $Q'_f = (\Delta \cup \{\varepsilon\}) \times Q_f$, and T' and wt are defined as follows.

- Let (q_0, x, p, q, f) be in T and $x \in \Delta \cup \{\varepsilon\}$.
  - If $x \in \Delta$ and $h(x) = a.y$, then the rule $(q'_0, y, p, (x,q), f)$ is in T', and its weight is a.
  - If $x = \varepsilon$, then the rule $(q'_0, \varepsilon, p, (\varepsilon, q), f)$ is in T', and its weight is 1.
- Let (q, x, p, q', f) be in T and $x \in \Delta \cup \{\varepsilon\}$. Moreover, let $x' \in \Delta \cup \{\varepsilon\}$ be arbitrary.
  - If $x \in \Delta$ and $h(x) = a.y$, then the rule $((x',q), y, p, (x,q'), f)$ is in T', and its weight is a.
  - If $x = \varepsilon$, then the rule $((x',q), \varepsilon, p, (\varepsilon, q'), f)$ is in T', and its weight is 1.

Let $w \in \Sigma^*$. First, let $v \in \Delta^*$ with $h(v) = z.w$ for some $z \in K$. We write $v = \delta_1 \dots \delta_n \in \Delta^*$ with $n \geq 0$ and $\delta_i \in \Delta$, $i \in [n]$. Let $h(\delta_i) = a_i.y_i$ for every $1 \leq i \leq n$. Thus $h(v) = \mathrm{val}(a_1 \dots a_n).y_1 \dots y_n$ and $w = y_1 \dots y_n$.

For $m \geq n$ let $\theta = \tau_1 \dots \tau_m$ be a q_0-computation in $\Theta_{\mathcal{B}}(v)$ with $\tau_1, \dots, \tau_m \in T$. For $1 \leq i \leq m$, let x_i be the second component of τ_i, so $x_i \in \Delta \cup \{\varepsilon\}$, and $v = x_1 \dots x_m$. Then we construct the q'_0-computation $\theta' = \tau'_1 \dots \tau'_m$ in $\Theta_{\mathcal{A}}(y_1 \dots y_n)$ with $\tau'_1, \dots, \tau'_m \in T'$ as follows:

- If $\tau_1 = (q_0, x_1, p_1, q_1, f_1)$, then we let $\tau'_1 = (q'_0, y', p_1, (x_1, q_1), f_1)$, where $y' = y$ if $x_1 \in \Delta$ and $h(x_1) = a.y$, and $y' = \varepsilon$ if $x_1 = \varepsilon$.
- If $1 < i \leq m$ and $\tau_i = (q_{i-1}, x_i, p_i, q_i, f_i)$, then we let $\tau'_i = ((x_{i-1}, q_{i-1}), y', p_i, (x_i, q_i), f_i)$, where $y' = y$ if $x_i \in \Delta$ and $h(x_i) = a.y$, and $y' = \varepsilon$ if $x_i = \varepsilon$.

Note that if $x_i \in \Delta$ and $h(x_i) = a.y$, then $\mathrm{wt}(\tau'_i) = a$, and if $x_i = \varepsilon$, then $\mathrm{wt}(\tau'_i) = 1$ for each $1 \leq i \leq m$ by definition of wt. Consequently

$$h(v)(w) = \mathrm{val}(a_1 \dots a_n) = \mathrm{val}(\mathrm{wt}(\tau'_1) \dots \mathrm{wt}(\tau'_m)) = \mathrm{wt}(\theta').$$

Conversely, for every q'_0-computation $\theta' = \tau'_1 \dots \tau'_m$ in $\Theta_{\mathcal{A}}(w)$ by definition of T' there are a uniquely determined $v \in \Delta^*$ and a uniquely determined q_0-computation $\theta = \tau_1 \dots \tau_m$ in

$\Theta_{\mathcal{B}}(v)$ such that θ' is the computation constructed above. Hence, for every $v \in \Delta^*$ and $w \in \Sigma^*$, if $h(v) = z.w$ for some $z \in K$, then $\Theta_{\mathcal{B}}(v)$ and $\Theta_{\mathcal{A}}(w)$ are in a one-to-one correspondence.

So, for every $w \in \Sigma^*$ we obtain

$$h(L(\mathcal{B}))(w) = \sum_{v \in L(\mathcal{B})} h(v)(w) = \sum_{\substack{v \in L(\mathcal{B}):\\ h(v)(w) \neq 0}} h(v)(w)\,.$$

Since $\mathcal{B}$ is unambiguous we can continue with

$$\sum_{\substack{v \in L(\mathcal{B}):\\ h(v)(w) \neq 0}} h(v)(w) = \sum_{\substack{v \in L(\mathcal{B}), \theta \in \Theta_{\mathcal{B}}(v):\\ h(v)(w) \neq 0}} \mathrm{wt}(\theta')\,.$$

Since there is a one-to-one correspondence between $\Theta_{\mathcal{B}}(v)$ and $\Theta_{\mathcal{A}}(w)$ we can continue with

$$\sum_{\substack{v \in L(\mathcal{B}), \theta \in \Theta_{\mathcal{B}}(v):\\ h(v)(w) \neq 0}} \mathrm{wt}(\theta') = \sum_{\theta' \in \Theta_{\mathcal{A}}(w)} \mathrm{wt}(\theta') = \|\mathcal{A}\|(w).$$

Thus $h(L(\mathcal{B})) = \|\mathcal{A}\|$. ■

Example 5.8. Recall the (P^1, T)-automaton $\mathcal{B} = (\{q, f\}, T, (\$, c), q, \{f\}, T_{\mathcal{B}})$ as well as the alphabetic morphism $h\colon T \to \mathcal{K}^{\lambda}_{\mathrm{disc}}[\Sigma \cup \{\varepsilon\}]$ from Example 5.6 with $\lambda = 0.5$. We construct the $(\mathrm{P}^1, \Sigma, \mathcal{K}^{\lambda}_{\mathrm{disc}})$-automaton $\mathcal{A} = (Q, \Sigma, (\$, c), \{q'_0\}, Q_f, T', \mathrm{wt})$, where $Q = \{q'_0\} \cup (T \cup \{\varepsilon\}) \times \{q, f\}$ and $Q_f = (T \cup \{\varepsilon\}) \times \{f\}$. Furthermore, we show by the transition

$$(q, \tau_1, \mathrm{bottom}, q, \mathrm{push}(B, f_{\mathrm{id}})) \in T_{\mathcal{B}}$$

how the transitions in T' are constructed. Note that $h(\tau_1) = 2.a$. Since q is the initial state of $\mathcal{B}$, we have that the transition

$$\tau = (q'_0, a, \mathrm{bottom}, (\tau_1, q), \mathrm{push}(B, f_{\mathrm{id}})) \in T'$$

and $\mathrm{wt}(\tau) = 2$. Additionally, the transitions

$$\begin{aligned}
&((\tau_1, q), a, \mathrm{bottom}, (\tau_1, q), \mathrm{push}(B, f_{\mathrm{id}})), && ((\tau_5, q), a, \mathrm{bottom}, (\tau_1, q), \mathrm{push}(B, f_{\mathrm{id}})),\\
&((\tau_2, q), a, \mathrm{bottom}, (\tau_1, q), \mathrm{push}(B, f_{\mathrm{id}})), && ((\tau_6, q), a, \mathrm{bottom}, (\tau_1, q), \mathrm{push}(B, f_{\mathrm{id}})),\\
&((\tau_3, q), a, \mathrm{bottom}, (\tau_1, q), \mathrm{push}(B, f_{\mathrm{id}})), && ((\tau_7, q), a, \mathrm{bottom}, (\tau_1, q), \mathrm{push}(B, f_{\mathrm{id}})),\\
&((\tau_4, q), a, \mathrm{bottom}, (\tau_1, q), \mathrm{push}(B, f_{\mathrm{id}})), && ((\varepsilon, q), a, \mathrm{bottom}, (\tau_1, q), \mathrm{push}(B, f_{\mathrm{id}})),
\end{aligned}$$

are in T', each with weight 2. The same procedure constructs the other transitions of T' from the remaining transitions of $T_{\mathcal{B}}$. □

6 Separating the storage from an (S, Δ)-automaton

In this chapter we will characterize the language recognized by an ε-free (S, Δ)-automaton $\mathcal{B}$ as the image of the set of behaviours of the initial configuration of $\mathcal{B}$ under a simple transducer mapping. Our proof follows closely the technique in the proof of [EV86, Theorem 3.26].

Before we get to our main theorem, we first want to introduce, respectively repeat, some concepts needed for this.

6.1 Ω-behaviour and a-transducers

Let c_0 be the initial configuration of $\mathcal{B}$ and θ a computation of $\mathcal{B}$, i.e., $\theta \in \Theta_{\mathcal{B}}(q_0, w, c_0)$ for some $w \in \Delta^*$. By dropping from θ all references to states and to the input, a sequence of pairs remains where each pair consists of a predicate and an instruction. Such a sequence is called a behaviour of c_0.

Definition 6.1. Let $S = (C, P, F, C_0)$ be a storage type and let Ω be a finite subset of $P \times F$. Moreover, let $c \in C$ and $v = (p_1, f_1) \dots (p_n, f_n) \in \Omega^*$ for some $n \in \mathbb{N}$. We say that v is an *Ω-behaviour of c* if for every i with $i \in [n]$ we have

(i) $p_i(c') = \text{true}$ and

(ii) $f_i(c')$ is defined,

where $c' = f_{i-1}(\dots f_1(c)\dots)$. Note that $c' = c$ for $i = 1$. We denote the set of all Ω-behaviours of c by $\mathrm{B}(\Omega, c)$. □

We note that each behaviour of c is a path in the approximation of c as defined in [EV86, Definition 3.23].

Example 6.2. Recall the (P^1, T)-automaton $\mathcal{B} = (\{q, f\}, T, (\$, \mathrm{c}), q, \{f\}, T_{\mathcal{B}})$ from Example 5.6. Let Ω be the set containing all tuples $(p, f) \in P \times F$ such that for some q_1, $q_2 \in \{q, f\}$, $x \in T$ there is a transition $(q_1, x, p, q_2, f) \in T_{\mathcal{B}}$. As an example, we have $(\text{bottom}, \text{push}(B, f_{\text{id}})) \in \Omega$ since the transition $(q, \tau_1, \text{bottom}, q, \text{push}(B, f_{\text{id}}))$ is in $T_{\mathcal{B}}$.

Now consider the computation

$$\theta = (q, \tau_1, \text{bottom}, q, \text{push}(B, f_{\text{id}}))(q, \tau_5, \text{top} = B, q, \text{pop})(q, \tau_7, \text{bottom}, f, \text{stay}(\$))$$

from $\mathcal{B}$. By dropping from θ all references to states and to the input, we obtain the sequence

$$\omega = (\text{bottom}, \text{push}(B, f_{\text{id}}))(\text{top} = B, \text{pop})(\text{bottom}, \text{stay}(\$)).$$

Since bottom$((\$, \text{c})) =$ true, push$(B, f_{\text{id}})((\$, \text{c}))$ is defined, $(\text{top} = B)((B, \text{c})(\$, \text{c})) =$ true and so on, we have that $\omega \in \mathrm{B}(\Omega, (\$, \text{c}))$. □

Now we briefly recall the concept of a-transducers from [GG70].

Definition 6.3. An *a-transducer (from Ω to Δ)* is a machine $\mathcal{M} = (Q, \Omega, \Delta, \delta, q_0, Q_f)$, where Q, Ω, and Δ are alphabets (of *states*, *input symbols*, and *output symbols*, respectively), $q_0 \in Q$ (*initial state*), $Q_f \subseteq Q$ (*final states*), and δ is a finite subset of $Q \times \Omega^* \times Q \times \Delta^*$ (*transitions*). We say that $\mathcal{M}$ is a *simple transducer* if $\delta \subseteq Q \times \Omega \times Q \times \Delta$. □

For the rest of this section, let $\mathcal{M} = (Q, \Omega, \Delta, \delta, q_0, Q_f)$ be an a-transducer.

For some transition $(q, x, q', y) \in \delta$ we call q its source state, x its input word (respectively input symbol if $\mathcal{M}$ is a simple transducer), q' the target state, and y the output word (respectively output symbol if $\mathcal{M}$ is a simple transducer).

Definition 6.4. The *computation relation* of $\mathcal{M}$ is the binary relation $\vdash_{\mathcal{M}}$ on the set $Q \times \Omega^* \times \Delta^*$ defined as follows: let $(q, ww', v) \vdash_{\mathcal{M}} (q', w', vv')$ for every $(q, w, q', v') \in \delta$, $w' \in \Omega^*$, and $v \in \Delta^*$. □

Definition 6.5. The *mapping induced by $\mathcal{M}$*, also denoted by $\mathcal{M}$, is the mapping $\mathcal{M} \colon \Omega^* \to \mathcal{P}(\Delta^*)$ defined by

$$\mathcal{M}(w) = \{v \in \Delta^* \mid (q_0, w, \varepsilon) \vdash_{\mathcal{M}}^* (q, \varepsilon, v),\ q \in Q_f\}.$$

If $\mathcal{M}$ is a simple transducer, then $\mathcal{M}(w)$ is finite for every w. For every $L \subseteq \Omega^*$ we define $\mathcal{M}(L) = \bigcup_{v \in L} \mathcal{M}(v)$. □

6.2 Separating the storage

Our aim of this section is to prove the following theorem.

Theorem 6.6. Let $S = (C, P, F, C_0)$ be a storage type. Moreover, let $L \subseteq \Delta^*$. Then the following are equivalent:

(1) L is recognizable by some ε-free (S, Δ)-automaton.

(2) There are $c \in C$, a finite set $\Omega \subseteq P \times F$, and a simple transducer $\mathcal{M}$ from Ω to Δ such that $L = \mathcal{M}(\mathrm{B}(\Omega, c))$.

We note that (1)⇒(2) of Theorem 6.6 is similar to [GG70, Lemma 2.3] (after decomposing the simple transducer $\mathcal{M}$ according to Theorem 7.1).

For the proof of this theorem, we define the concept of relatedness between an ε-free (S, Δ)-automaton $\mathcal{B}$ and a simple transducer $\mathcal{M}$ with the following intention: $\mathcal{B}$ allows a computation

$$(q_0, x_1, p_1, q_1, f_1)(q_1, x_2, p_2, q_2, f_2) \ldots (q_{n-1}, x_n, p_n, q_n, f_n)$$

for some states $q_1, \ldots, q_{n-1}$ if and only if

$$(q_0, (p_1, f_1) \ldots (p_n, f_n), \varepsilon) \vdash^*_{\mathcal{M}} (q_n, \varepsilon, x_1 \ldots x_n)$$

and $(p_1, f_1) \ldots (p_n, f_n) \in \mathrm{B}(\Omega, c_0)$.

That is, while reading a behaviour of the initial configuration of $\mathcal{B}$, the simple transducer $\mathcal{M}$ produces a string which is recognized by $\mathcal{B}$.

Definition 6.7. Let $\mathcal{B} = (Q, \Delta, c, q_0, Q_f, T)$ be an ε-free (S, Δ)-automaton and let $\mathcal{M} = (Q', \Omega, \Delta', \delta, q'_0, Q'_f)$ be a simple transducer. Then $\mathcal{B}$ *is related to* $\mathcal{M}$ if

- $Q = Q'$, $q_0 = q'_0$, $Q_f = Q'_f$,
- $\Delta = \Delta'$ and Ω is the set of all pairs (p, f) such that T contains a transition of the form (q, x, p, q', f) for some q, $q' \in Q$, and $x \in \Delta$, and
- for every $q, q' \in Q$, $x \in \Delta$, $p \in P$, and $f \in F$ we have: $(q, x, p, q', f) \in T$ if and only if $(q, (p, f), q', x) \in \delta$. □

Example 6.8. Recall the (P^1, T)-automaton $\mathcal{B} = (\{q, f\}, T, (\$, \mathrm{c}), q, \{f\}, T_{\mathcal{B}})$ from Example 5.6. Clearly, $\mathcal{B}$ is ε-free. In the further we denote the initial configuration $(\$, \mathrm{c})$ of $\mathcal{B}$ by c. The simple transducer $\mathcal{M}$, such that $\mathcal{B}$ is related to $\mathcal{M}$, is the transducer $\mathcal{M} = (\{q, f\}, \Omega, T, \delta, q, \{f\})$, where

$$\begin{aligned}\Omega = \{&(\mathrm{bottom}, \mathrm{push}(B, f_{\mathrm{id}})), (\mathrm{bottom}, \mathrm{push}(A, f_{\mathrm{id}})), (\mathrm{bottom}, \mathrm{stay}(\$)),\\ &(\mathrm{top} = B, \mathrm{push}(B, f_{\mathrm{id}})), (\mathrm{top} = B, \mathrm{pop}), (\mathrm{top} = A, \mathrm{push}(A, f_{\mathrm{id}})),\\ &(\mathrm{top} = A, \mathrm{pop})\},\end{aligned}$$

and δ consists of the transitions

$$\begin{array}{ll} (q, (\mathrm{bottom}, \mathrm{push}(B, f_{\mathrm{id}})), q, \tau_1), & (q, (\mathrm{bottom}, \mathrm{push}(A, f_{\mathrm{id}})), q, \tau_4),\\ (q, (\mathrm{top} = A, \mathrm{pop}), q, \tau_2), & (q, (\mathrm{top} = B, \mathrm{pop}), q, \tau_5),\\ (q, (\mathrm{top} = B, \mathrm{push}(B, f_{\mathrm{id}})), q, \tau_3), & (q, (\mathrm{top} = A, \mathrm{push}(A, f_{\mathrm{id}})), q, \tau_6), \end{array}$$
$$(q, (\mathrm{bottom}, \mathrm{stay}(\$)), f, \tau_7).$$

Now consider the computation

$$(q,\tau_1,\text{bottom},q,\text{push}(B,f_{\text{id}}))(q,\tau_5,\text{top}=B,q,\text{pop})(q,\tau_7,\text{bottom},f,\text{stay}(\$))$$

of $\mathcal{B}$, recognizing the word $w=\tau_1\tau_5\tau_7 \in L(\mathcal{B})$. The corresponding computation of $\mathcal{M}$ is

$$(q,(\text{bottom}\,\text{push}(B,f_{\text{id}})),q,\tau_1)(q,(\text{top}=B,\text{pop}),q,\tau_5)(q,(\text{bottom},\text{stay}(\$)),f,\tau_7),$$

which translates the string $(\text{bottom},\text{push}(B,f_{\text{id}}))(\text{top}=B,\text{pop})(\text{bottom},\text{stay}(\$)) \in \text{B}(\Omega,c)$ into w. Thus, w is also in $\mathcal{M}(\text{B}(\Omega,c))$. □

Now we can state the following lemma.

Lemma 6.9. Let $\mathcal{B}$ be an ε-free (S,Δ)-automaton with initial configuration c and let $\mathcal{M}$ be a simple transducer from Ω to Δ. If $\mathcal{B}$ is related to $\mathcal{M}$, then $L(\mathcal{B})=\mathcal{M}(\text{B}(\Omega,c))$.

Proof. Let $\mathcal{B}=(Q,\Delta,c,q_0,Q_f,T)$ and $\mathcal{M}=(Q,\Omega,\Delta,\delta,q_0,Q_f)$.

First we prove that $L(\mathcal{B})\subseteq\mathcal{M}(\text{B}(\Omega,c))$. Let $v\in L(\mathcal{B})$. Then $v=x_1...x_n$ for some $n\geq 0$ and $x_i\in\Delta$ for every $1\leq i\leq n$. Moreover, there is a q_0-computation θ in $\Theta_{\mathcal{B}}(v)$ with $\theta=\tau_1...\tau_n$ such that $\tau_i\in T$, where $\tau_1=(q_0,x_1,p_1,q_1,f_1)$, for every $2\leq i\leq n$ we have $\tau_i=(q_{i-1},x_i,p_i,q_i,f_i)$, and $q_n\in Q_f$.

Since $\mathcal{B}$ is related to $\mathcal{M}$, we have $(q_{i-1},(p_i,f_i),q_i,x_i)\in\delta$ for every $1\leq i\leq n$. Hence $(q_0,w,\varepsilon)\vdash^*_{\mathcal{M}}(q_n,\varepsilon,x_1\dots x_n)$ with $w=(p_1,f_1)\dots(p_n,f_n)$. Since $w\in\text{B}(\Omega,c)$ is a behaviour of c, $v=x_1\dots x_n$, and $q_n\in Q_f$, we obtain that $v\in\mathcal{M}(\text{B}(\Omega,c))$.

Next we prove that $\mathcal{M}(\text{B}(\Omega,c))\subseteq L(\mathcal{B})$. Let $v\in\mathcal{M}(\text{B}(\Omega,c))$ with $v=x_1...x_n$ for some $n\geq 0$ and $x_i\in\Delta$ for every $1\leq i\leq n$. Then there is a behaviour $w\in\text{B}(\Omega,c)$ of c such that $v\in\mathcal{M}(w)$. Then there are $(p_i,f_i)\in\Omega$ with $1\leq i\leq n$ such that $w=(p_1,f_1)\dots(p_n,f_n)$. Moreover, there are $q_0,\dots,q_n\in Q$ such that $(q_0,(p_1,f_1),q_1,x_1)\in\delta$, for every $2\leq i\leq n$ we have $(q_{i-1},(p_i,f_i),q_i,x_i)\in\delta$, and $q_n\in Q_f$.

Since $\mathcal{B}$ is related to $\mathcal{M}$, we have $\tau_i=(q_{i-1},x_i,p_i,q_i,f_i)\in T$ for $1\leq i\leq n$. Since $w\in\text{B}(\Omega,c)$, q_0 is the initial state of $\mathcal{B}$, and $q_n\in Q_f$, we have that $\tau_1\dots\tau_n\in\Theta_{\mathcal{B}}(v)$ and thus $v\in L(\mathcal{B})$. ■

By means of Lemma 6.9 we can now prove our main theorem of this chapter.

Proof of Theorem 6.6. (1) ⇒ (2): Let L be a language recognizable by some ε-free (S,Δ)-automaton $\mathcal{B}=(Q,\Delta,c,q_0,Q_f,T)$. Let Ω be the set of all pairs (p,f) such that T contains a transition of the form (q,x,p,q',f) for some $q,q'\in Q$, and $x\in\Delta$. We construct the simple transducer $\mathcal{M}=(Q,\Omega,\Delta,\delta,q_0,Q_f)$ by defining $(q,(p,f),q',x)\in\delta$ if and only if $(q,x,p,q',f)\in T$ for every $q,q'\in Q$, $x\in\Delta$, and $(p,f)\in\Omega$. Clearly, $\mathcal{B}$ is related to $\mathcal{M}$ and thus, by Lemma 6.9, we have that $L(\mathcal{B})=\mathcal{M}(\text{B}(\Omega,c))$.

(2) ⇒ (1): Let $c\in C$, Ω a finite subset of $P\times F$, and $\mathcal{M}=(Q,\Omega,\Delta,\delta,q_0,Q_f)$ a simple transducer. First we reduce $\mathcal{M}$ to the simple transducer $\mathcal{M}'=(Q,\Omega',\Delta,\delta,q_0,Q_f)$ where

Ω' is the set of all pairs (p,f) such that $(q,(p,f),q',x) \in \delta$ for some $q,q' \in Q$ and $x \in \Delta$. Obviously, $\delta \subseteq Q \times \Omega' \times Q \times \Delta$ and $\mathcal{M}(\mathrm{B}(\Omega,c)) = \mathcal{M}'(\mathrm{B}(\Omega',c))$.

Next we construct the ε-free (S,Δ)-automaton $\mathcal{B} = (Q,\Delta,c,q_0,Q_f,T)$ by defining $T = \{(q,x,p,q',f) \mid (q,(p,f),q',x) \in \delta\}$. Since $\mathcal{B}$ is related to $\mathcal{M}'$, by Lemma 6.9 we have that $L(\mathcal{B}) = \mathcal{M}'(\mathrm{B}(\Omega',c)) = \mathcal{M}(\mathrm{B}(\Omega,c))$. ■

7 Chomsky-Schützenberger characterization of weighted automata with storage

In this chapter we want to give the proof of our Chomsky-Schützenberger theorem for weighted automata with storage. We obtain this theorem by separating the weights and separating the storage according to Theorem 5.4 and Theorem 6.6, respectively, from an (S, Σ, K)-automaton. Then the resulting simple transducer can be decomposed further. Therefore, we first recall a decomposition result for simple transducers [GG69, proof of Theorem 2.1].

Theorem 7.1. Let Ω be an alphabet, let $L \subseteq \Omega^*$, and let $\mathcal{M}: \Omega^* \to \mathcal{P}_{\text{fin}}(\Delta^*)$ be induced by a simple transducer $\mathcal{M}$. Then there are an alphabet Φ, two letter-to-letter morphisms $h_1: \Phi \to \mathbb{B}[\Omega]$ and $h_2: \Phi \to \mathbb{B}[\Delta]$, and a regular language $R \subseteq \Phi^*$ such that $\mathcal{M}(L) = h_2(h_1^{-1}(L) \cap R)$.

To illustrate our result afterwards by way of example, let us briefly recall the construction in the proof of [GG69, Theorem 2.1] at this point.
Let $L \subseteq \Omega^*$ and let $\mathcal{M} = (Q, \Omega, \Delta, \delta, q_0, Q_f)$ be a simple transducer. We construct the alphabet $\Phi \subseteq Q \times \Omega \times Q \times \Delta$ such that $(q, a, q', b) \in \Phi$ if and only if $(q, a, q', b) \in \delta$. Then we define two letter-to-letter morphisms $h_1: \Phi \to \mathbb{B}[\Omega]$ and $h_2: \Phi \to \mathbb{B}[\Delta]$ such that for all $(q, a, q', b) \in \Phi$

$$h_1(q, a, q', b) = 1.a \quad \text{and} \quad h_2(q, a, q', b) = 1.b\ .$$

Furthermore, let $R \subseteq \Phi^*$ be the set[1]

$$\{(p_0, a_1, p_1, b_1)(p_1, a_2, p_2, b_2) \dots (p_{n-1}, a_n, p_n, b_n) \in \Phi^* \mid n \geq 0, p_0 = q_0, p_n \in Q_f\}.$$

Obviously, R is regular and $\mathcal{M}(L) = h_2(h_1^{-1}(L) \cap R)$.

Example 7.2. Let $\mathcal{M} = (\{q, f\}, \Omega, T, \delta, q, \{f\})$ be the simple transducer from Example 6.8. Then let $\Phi = \delta$ and define $h_1: \Phi \to \mathbb{B}[\Omega]$, $h_2: \Phi \to \mathbb{B}[T]$, and R as in the construction above. We want to give a few examples to illustrate the resulting formalisms. For this consider the transition $(q, (\text{bottom}, \text{push}(B, f_{\text{id}})), q, \tau_1) \in \delta$. This transition is mapped by h_1 to its input symbol,

$$h_1(q, (\text{bottom}, \text{push}(B, f_{\text{id}})), q, \tau_1) = 1.(\text{bottom}, \text{push}(B, f_{\text{id}})),$$

[1] Note that, in contrast to [GG69], we also allow ε as an element of R for the case that the initial state of the simple transducer is also a final state.

and by h_2 to its output symbol,

$$h_2(q,(\text{bottom},\text{push}(B,f_{\text{id}})),q,\tau_1) = 1.\tau_1\ .$$

Moreover, we have that $(q,(\text{bottom},\text{push}(B,f_{\text{id}})),q,\tau_1) \notin R$ since $q \notin Q_f$. For an element of R consider the word

$$\theta = (q,(\text{bottom},\text{push}(B,f_{\text{id}})),q,\tau_1)(q,(\text{top}=B,\text{pop}),q,\tau_5)(q,(\text{bottom},\text{stay}(\#)),f,\tau_7),$$

where the source state of the first symbol is the initial state of $\mathcal{M}$, the target state of the last symbol is a final state of M, and the target state of each symbol equals the source state of its subsequent symbol. □

Next we show that a letter-to-letter morphism $h_2\colon \Phi \to \mathbb{B}[\Delta]$ and an alphabetic morphism $h\colon \Delta \to K[\Sigma \cup \{\varepsilon\}]$ can be combined smoothly. For this recall that for every $x \in \Phi$ we have $|\operatorname{supp}(h_2(x))| = 1$.

Definition 7.3. Let $h_2\colon \Phi \to \mathbb{B}[\Delta]$ be a letter-to-letter morphism and let $h\colon \Delta \to K[\Sigma \cup \{\varepsilon\}]$ be an alphabetic morphism. Then we define the alphabetic morphism $(h \circ h_2)\colon \Phi \to K[\Sigma \cup \{\varepsilon\}]$ for every $x \in \Phi$ by

$$(h \circ h_2)(x) = h(\delta) \quad \text{if} \quad h_2(x) = 1.\delta$$

for some $\delta \in \Delta$. □

Example 7.4. Recall the alphabetic morphism $h\colon T \to \mathcal{K}^{\lambda}_{\text{disc}}[\Sigma \cup \{\varepsilon\}]$ from Example 5.6 for $\lambda = 0.5$ as well as the letter-to-letter morphism $h_2\colon \Phi \to \mathbb{B}[T]$ from Example 7.2. Moreover, consider the symbol $(q,(\text{bottom},\text{push}(B,f_{\text{id}})),\tau_1,q) \in \Phi$. Since

$$h_2(q,(\text{bottom},\text{push}(B,f_{\text{id}})),q,\tau_1) = 1.\tau_1 \quad \text{and} \quad h(\tau_1) = 2.a,$$

we have that

$$(h \circ h_2)(q,(\text{bottom},\text{push}(B,f_{\text{id}})),\tau_1,q) = 2.a\ .$$

Moreover, for a more complex example consider the string

$$\theta = (q,(\text{bottom},\text{push}(B,f_{\text{id}})),q,\tau_1)(q,(\text{top}=B,\text{pop}),q,\tau_5)(q,(\text{bottom},\text{stay}(\#)),f,\tau_7)$$

from Example 7.2. With the same argumentation we have that

$$(h \circ h_2)(\theta) = \text{val}(210).ab\varepsilon = 2.5.ab\ .$$ □

Lemma 7.5. Let $h_2\colon \Phi \to \mathbb{B}[\Delta]$ be a letter-to-letter morphism and let $h\colon \Delta \to K[\Sigma \cup \{\varepsilon\}]$ be an alphabetic morphism. Moreover, let $H \subseteq \Phi^*$ be a language. If $(h(v) \mid v \in h_2(H))$ is locally finite, then $((h \circ h_2)(w) \mid w \in H)$ is locally finite.

$$\begin{tikzcd}
 & & \cap R \subseteq \Phi^* \arrow[dr, "h_2"] \arrow[ddr, bend left=60, "h'"] & \\
B(\Omega, c) \subseteq \Omega^* \arrow[urr, "h_1^{-1}"] \arrow[rrr, "\mathcal{M}"] & & & \Delta^* \supseteq L(\mathcal{B}) \arrow[d, "h"] \\
 & & & r \in K\langle\!\langle \Sigma^* \rangle\!\rangle
\end{tikzcd}$$

Figure 7.1: An illustration of the proof of Theorem 7.6.

Proof. Let $u \in \Sigma^*$. By assumption, we have that $\{v \in h_2(H) \mid u \in \operatorname{supp}(h(v))\}$ is finite; let us denote this set by C_u. Since h_2 is letter-to-letter, we have that $\{y \in H \mid v \in h_2(y)\}$ is finite for each $v \in h_2(H)$. Then we have:

$$|\{w \in H \mid u \in \operatorname{supp}((h \circ h_2)(w)\}| = \sum_{v \in C_u} |\{y \in H \mid v \in h_2(y)\}|.$$

Hence, $\{w \in H \mid u \in \operatorname{supp}((h \circ h_2)(w)\}$ is finite. ■

Now we can prove the CS theorem for (S, Σ, K)-automata. In Figure 7.1 we illustrate the proof of this theorem.

Theorem 7.6. Let $S = (C, P, F, C_0)$ be a storage type, Σ an alphabet, and K a unital valuation monoid. If $r \in K\langle\!\langle \Sigma^* \rangle\!\rangle$ is (S, Σ, K)-recognizable, then there are

- an alphabet Φ and a regular language $R \subseteq \Phi^*$,
- a finite set $\Omega \subseteq P \times F$ and a configuration $c \in C$,
- a letter-to-letter morphism $h_1 \colon \Phi \to \mathbb{B}[\Omega]$, and
- an alphabetic morphism $h' \colon \Phi \to K[\Sigma \cup \{\varepsilon\}]$

such that $r = h'(h_1^{-1}(B(\Omega, c)) \cap R)$.

Proof. By Theorem 5.4 there are an alphabet Δ, an ε-free (S, Δ)-automaton $\mathcal{B}$, and an alphabetic morphism $h \colon \Delta \to K[\Sigma \cup \{\varepsilon\}]$ such that $r = h(L(\mathcal{B}))$. Since $\mathcal{B}$ is ε-free, $\Theta_{\mathcal{B}}(w)$ is finite for every $w \in \Delta^*$, and if K is not complete then $(h(v) \mid v \in L(\mathcal{B}))$ is locally finite.

According to Theorem 6.6, there are $c \in C$, a finite set $\Omega \subseteq P \times F$, and a simple transducer $\mathcal{M}$ from Ω to Δ such that $L(\mathcal{B}) = \mathcal{M}(B(\Omega, c))$.

Due to Theorem 7.1, there are an alphabet Φ, two letter-to-letter morphisms $h_1 \colon \Phi \to \mathbb{B}[\Omega]$ and $h_2 \colon \Phi \to \mathbb{B}[\Delta]$, as well as a regular language $R \subseteq \Phi^*$ such that $\mathcal{M}(B(\Omega, c)) = h_2(h_1^{-1}(B(\Omega, c)) \cap R)$. Let us denote the language $h_1^{-1}(B(\Omega, c)) \cap R$ by H. Thus $L(\mathcal{B}) = h_2(H)$.

Since $(h(v) \mid v \in L(\mathcal{B}))$ is locally finite if K is not complete, we have by Lemma 7.5 that also $((h \circ h_2)(w) \mid w \in H)$ is locally finite if K is not complete. Thus $r = (h \circ h_2)(h_1^{-1}(\mathrm{B}(\Omega, c)) \cap R)$ and we can take $h' = (h \circ h_2)$. ■

Now, by the combination of previous examples, we can illustrate the decomposition of Theorem 7.6.

Example 7.7. The aim of this example is to illustrate the decomposition of a computation of an (S, Σ, K)-automaton into the components according to Theorem 7.6. For this recall the unital valuation monoid $\mathcal{K}^{\lambda}_{\mathrm{disc}} = (\bar{\mathbb{R}}, \max, \mathrm{val}^{\lambda}_{\mathrm{disc}}, -\infty, 0)$ from Example 2.5 for $\lambda = 0.5$ and let $\mathcal{A} = (\{q, f\}, \Sigma, (\$, \mathrm{c}), q, \{f\}, T, \mathrm{wt})$ be the $(\mathrm{P}^1, \Sigma, \mathcal{K}^{\lambda}_{\mathrm{disc}})$-automaton from Example 3.31 with $\Sigma = \{a, b\}$. Furthermore, recall

- the alphabet Φ and the regular language $R \subseteq \Phi^*$ from Example 7.2,
- the alphabet Ω and the configuration $c = (\$, \mathrm{c})$ from Example 6.8,
- the letter-to-letter morphism $h_1 \colon \Phi \to \mathbb{B}[\Omega]$ from Example 7.2, and
- the alphabetic morphism $(h \circ h_2) \colon \Phi \to K[\Sigma \cup \{\varepsilon\}]$ from Example 7.4 that we denote in the following by h'.

Now we want to illustrate how the word $ab \in \Sigma^*$ is treated. Consider the string

$$\omega = (\mathrm{bottom}, \mathrm{push}(B, f_{\mathrm{id}}))(\mathrm{top} = B, \mathrm{pop})(\mathrm{bottom}, \mathrm{stay}(\$)) \in \Omega^*.$$

As argued in Example 6.2 we have that $\omega \in \mathrm{B}(\Omega, c)$. The corresponding string under the inverse of h_1 is

$$\theta = (q, (\mathrm{bottom}, \mathrm{push}(B, f_{\mathrm{id}})), q, \tau_1)(q, (\mathrm{top} = B, \mathrm{pop}), q, \tau_5)(q, (\mathrm{bottom}, \mathrm{stay}(\#)), f, \tau_7),$$

which is also in R as explained in Example 7.2. It was illustrated in Example 7.4 that

$$h'(\theta) = 2.5.ab$$

and, therefore, $2.5.ab \in h'(h_1^{-1}(\mathrm{B}(\Omega, c)) \cap R)$. On the other hand, we have that

$$\theta' = (q, a, \mathrm{bottom}, q, \mathrm{push}(B, f_{\mathrm{id}}))(q, b, \mathrm{top} = B, q, \mathrm{pop})(q, \varepsilon, \mathrm{bottom}, f, \mathrm{stay}(\$))$$

is in $\Theta_{\mathcal{A}}(ab)$ and $\mathrm{wt}(\theta') = 2.5.ab$. Since $\mathcal{A}$ is unambiguous, we therefore also have $\|\mathcal{A}\|(ab) = 2.5$. □

7.1 Instantiations of the Chomsky-Schützenberger theorem

Finally we instantiate the storage type S and the unital valuation monoid K in Theorem 7.6 in several ways and obtain the CS theorem for the corresponding class of (S, Σ, K)-recognizable languages.

(1) Let $S = \mathrm{P}^1$ and let $K = \mathbb{B}$. The we obtain an alternative CS theorem for context-free languages. In fact, we conjecture that the original CS theorem as formulated in [Har78] follows from this instantiation of our result, compare to Chapter 8. However, we will postpone the formal investigation of this conjecture until further work.

(2) Let $S = \mathrm{P}^1$ and let K be an arbitrary unital valuation monoid. The we obtain an alternative CS theorem for quantitative context-free languages [DV13]. We will compare these two theorems in Chapter 8.

(3) Let $S = \mathrm{P}^n$ and let $K = \mathbb{B}$. The we obtain a CS theorem for n-iterated pushdown languages.

(4) Let $S = \mathrm{P}^n$ and let K be an arbitrary unital valuation monoid. The we obtain a CS theorem for the K-weighted n-iterated pushdown languages from Definition 3.33.

(5) $S = \mathrm{MON}(M)$ for some monoid M and let $K = \mathbb{B}$. We obtain a CS theorem for M-automata [Kam07].

8 Comparison of two Chomsky-Schützenberger theorems

Now that we have developed a CS theorem for weighted automata with storage and showed that $(\mathrm{P}^1, \Sigma, K)$-automata are expressively equivalent to the weighted pushdown automata (WPDA) of [DV13], in this chapter we want to continue with a comparison of Chomsky-Schützenberger characterizations for these two formalisms.

Also in [DV13], a CS theorem for languages recognizable by WPDAs was proved. In the following, we want to compare this result with our CS result, instantiated for $(\mathrm{P}^1, \Sigma, K)$-recognizable languages.

In contrast to the previous work, this chapter is intended to provide an informal overview. Instead of formal proofs we will use examples and illustrations to demonstrate similarities and differences between the two CS results.

First, we want to recall a few notions and definitions that we will need in the course of this chapter.

In the further we refer to the class of languages recognized by WPDAs as the class of weighted context-free languages. A second formalism which generates exactly this class of languages and is used in [DV13] are weighted context-free grammars (over unital valuation monoids).

Definition 8.1. A *context-free grammar* (CFG) is a tuple $G = (N, \Sigma, Z, P)$ where N is a finite set (*nonterminals*), Σ is an alphabet (*terminals*), $Z \in N$ (*initial nonterminal*), and P is a finite set with elements of the form $A \to \xi$, with $A \in N$ and $\xi \in (N \cup \Sigma)^*$ (*productions*). □

The leftmost derivation relation $\Rightarrow$ of a CFG G is defined in the usual way, and we write $\Rightarrow^p$ if the production $p \in P$ is used in the derivation step. Furthermore, a derivation of G is a sequence $d = p_1 \dots p_n$, $n \geq 0$, of productions of P such that there are $\xi_0, \dots, \xi_n \in (N \cup \Sigma)^*$ with $\xi_0 \Rightarrow^{p_1} \xi_1 \Rightarrow^{p_2} \dots \Rightarrow^{p_n} \xi_n$ in a leftmost manner. The set of all derivations starting with the initial nonterminal Z and resulting in a word $w \in \Sigma^*$ is denoted by $D(w)$. We say that G is unambiguous if $|D(w)| \leq 1$ for each $w \in \Sigma^*$. Then we can define the language generated by G as follows:

Definition 8.2. Let G be a CFG. The *language generated by* G is the set

$$L(G) = \{w \in \Sigma^* \mid D(w) \neq \emptyset\}. \qquad \square$$

We refer to the class of languages generated by context-free grammars as the class of context-free languages.

A context-free grammar can be extended to a weighted context-free grammar by assigning a weight, taken from a unital valuation monoid K, to each production of P.

Definition 8.3. Let K be a unital valuation monoid. A *K-weighted context-free grammar* (K-WCFG) is a tuple $G = (N, \Sigma, Z, P, \mathrm{wt})$ where (N, Σ, Z, P) is a CFG and $\mathrm{wt}\colon P \to K$ is a mapping (*weight assignment*). Moreover, it must be the case that $D(w)$ is finite for every $w \in \Sigma^*$ or that K is complete. $\square$

Each derivation $d = p_1 \dots p_n$, $n \geq 0$, is assigned a weight $\mathrm{wt}(d) = \mathrm{val}(\mathrm{wt}(p_1) \dots \mathrm{wt}(p_n))$ from K. Then the grammar G associates to every word $w \in \Sigma^*$ the sum of the weights of all derivations of w:

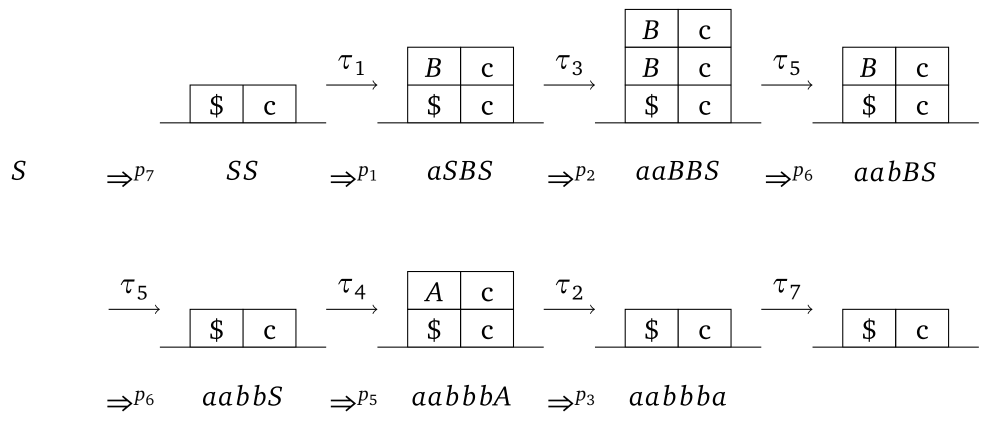
Definition 8.4. Let K be a unital valuation monoid and let $G = (N, \Sigma, Z, P, \mathrm{wt})$ be a K-WCFG. The *weighted language of* G is the K-weighted language $\|G\|\colon \Sigma^* \to K$ defined for every $w \in \Sigma^*$ by

$$\|G\|(w) = \sum_{d \in D(w)} \mathrm{wt}(d). \qquad \square$$

It was shown in [DV13, Theorem 1] that, for every alphabet Σ and unital valuation monoid K, the weighted languages generated by K-WCFGs over Σ are exactly the weighted languages recognized by WPDAs over Σ and K. Furthermore, we stated in Theorem 4.9 that the latter languages in turn are the same as the weighted languages recognized by $(\mathrm{P}^1, \Sigma, K)$-automata. To show the similarity between K-WCFGs over Σ and our formalism of $(\mathrm{P}^1, \Sigma, K)$-automata we consider the following example.

Example 8.5. Recall the unital valuation monoid $\mathcal{K}^\lambda_{\mathrm{disc}} = (\tilde{\mathbb{R}}, \max, \mathrm{val}^\lambda_{\mathrm{disc}}, -\infty, 0)$ from Example 2.5 for $\lambda = 0.5$ and let $\mathcal{A} = (\{q, f\}, \Sigma, (\$, c), q, \{f\}, T, \mathrm{wt})$ be the $(\mathrm{P}^1, \Sigma, \mathcal{K}^\lambda_{\mathrm{disc}})$-automaton from Example 3.31 with $\Sigma = \{a, b\}$. We construct a $\mathcal{K}^\lambda_{\mathrm{disc}}$-WCFG G such that $\|G\| = \|\mathcal{A}\|$. For this let $G = (\{S, A, B\}, \Sigma, S, P, \mathrm{wt}_G)$, where P contains the productions

$$\begin{aligned} p_1 &= (S \to aSB) & p_4 &= (S \to bSA) \\ p_2 &= (S \to aB) & p_5 &= (S \to bA) \\ p_3 &= (A \to a) & p_6 &= (B \to b) \\ & & p_7 &= (S \to SS) \end{aligned}$$



$S \Rightarrow_{p_7} SS \Rightarrow_{p_1} aSBS \Rightarrow_{p_2} aaBBS \Rightarrow_{p_6} aabBS$
$\Rightarrow_{p_6} aabbS \Rightarrow_{p_5} aabbbA \Rightarrow_{p_3} aabbba$
τ_1
τ_3
τ_5
τ_5
τ_4
τ_2
τ_7

Figure 8.1: The derivation of the word $w = aabbba$ by G in comparison with the behaviour of the pushdown of $\mathcal{A}$, while recognizing w with the computation $\tau_1\tau_3\tau_5\tau_5\tau_4\tau_2\tau_7$ from Example 3.31.

and

$$\mathrm{wt}_G(p) = \begin{cases} 2 & \text{if } p \in \{p_1, p_2, p_3\} \\ 1 & \text{if } p \in \{p_4, p_5, p_6\} \\ 0 & \text{if } p = p_7 \end{cases} .$$

The idea how to generate words with the same number of as and bs is very similar to the computation concept of $\mathcal{A}$. To see this, consider as an example the production/transition

$$p_1 = (S \to aSB), \qquad \tau_1 = (q, a, \text{bottom}, q, \text{push}(B, f_{\text{id}}))$$

from P and T, respectively. In p_1, starting from the nonterminal S, the terminal a is generated and then S is called again. Additionally, the nonterminal B ensures that also another b will be derived. In τ_1, this necessary b is counted by the symbol B on the pushdown.

The elimination of B is then realized by the production/transition

$$p_6 = (B \to b), \qquad \tau_5 = (q, b, \text{top} = B, q, \text{pop})$$

from P and T, respectively.

The main difference between G and $\mathcal{A}$ lies in the fact that G keeps track of the symbols to be generated with nonterminals while $\mathcal{A}$ realizes this counting by its pushdown. By this reason, G has not to implement the "switching mechanism" of the pushdown from B to A if a symbol difference is negative (compare with Example 3.31). This phenomenon is captured by the production $S \to SS$ as follows. If a word $w \in L(G)$ can be split into words $w_1, \ldots, w_n$, $n \geq 2$, such that $w = w_1 \ldots w_n$ and $w_i \in L(G)$ for every $i \in [n]$, each of this partial words is generated by a separate nonterminal S.

To understand the derivation $p_7p_1p_2p_6p_6p_5p_3$ of the word $w = aabbba$ by G in comparison with the behaviour of the pushdown of $\mathcal{A}$, during the recognition of w with the computation

$\tau_1 \dots \tau_7$ from Example 3.31, see Figure 8.1. It is clearly visible that G and $\mathcal{A}$ are counting the same number of As and Bs.

Since each production which generates an a is of weight 2, each production which generates a b is of weight 1, and $S \to SS$ is of weight 0, it is easy to see that $\|G\|(w) = 3.5$ and $\|G\| = \|\mathcal{A}\|$. □

A particular context-free language consists of all well-bracketed words over some alphabet Y and is also known as Dyck language.

Definition 8.6. Let Y be an alphabet and let $\overline{Y} = \{\overline{y} \mid y \in Y\}$. The *Dyck language over* Y, denoted by D_Y, is the smallest set D such that

- $\varepsilon \in D$,
- for all $v, w \in D$ we have $vw \in D$, and
- for all $w \in D$ and $y \in Y$ we have $yw\overline{y} \in D$.

We will often denote $y \in Y$ by $(_y$ and $\overline{y} \in \overline{Y}$ by $)_y$. □

Now we want to recall the Chomsky-Schützenberger theorem for weighted context-free languages of [DV13]. Note that we only consider the two essential parts of the original theorem that we need for our comparison. Moreover, the version in [DV13] was formulated as an equivalence instead of an implication. However, we only need one direction. Then, the CS result of [DV13] can be formulated as follows:

Theorem 8.7. (cf. [DV13, Theorem 2]) Let K be a unital valuation monoid and $r \in K\langle\langle\Sigma^*\rangle\rangle$. If there is a K-WCFG G such that $\|G\| = r$, then there are an alphabet Y, a recognizable language H over $Y \cup \overline{Y}$, and an alphabetic morphism $g' \colon Y \cup \overline{Y} \to K[\Sigma \cup \{\varepsilon\}]$ such that $r = g'(D_Y \cap H)$.

In the following we will briefly describe the structure of this theorem and the idea of the associated proof in comparison with the proof of our CS theorem, which are both illustrated in Figure 8.2.

Given a weighted context-free grammar, in a first step it is decomposed into an alphabetic morphism and an unweighted context-free grammar, similar to our Lemma 5.5. This part we will investigate first. Therefore, recall the following lemma:

Lemma 8.8. ([DV13, Lemma 3]) Let K be a unital valuation monoid and $r \in K\langle\langle\Sigma^*\rangle\rangle$. If there is a K-WCFG G such that $\|G\| = r$, then there are an alphabet Δ, an unambiguous CFG G' over Δ, and an alphabetic morphism $g : \Delta \to K[\Sigma \cup \{\varepsilon\}]$ such that $r = g(L(G'))$.

For the rest of this section let $G = (N, \Sigma, Z, P, \mathrm{wt})$ be a K-WCFG over Σ and let $\mathcal{A} = (Q, \Sigma, c_0, q_0, Q_f, T, \mathrm{wt})$ be a $(\mathrm{P}^1, \Sigma, K)$-automaton.

$$\begin{tikzcd}
 & \cap R \subseteq \Phi^* \arrow[dr, "h_2"] \arrow[ddr, bend left, "h'"] & \\
\mathrm{B}(\Omega, c) \subseteq \Omega^* \arrow[ur, "h_1^{-1}"] \arrow[rr, "\mathcal{M}"] & & \Delta^* \supseteq L(\mathcal{B}) \arrow[d, "h"] \\
 & & r \in K\langle\langle\Sigma^*\rangle\rangle
\end{tikzcd}$$

$$\begin{tikzcd}
D_Y \cap H \subseteq (Y \cup \overline{Y})^* \arrow[dr, "g_2"] \arrow[ddr, bend left, "g'"] & \\
 & \Delta^* \supseteq L(G') \arrow[d, "g"] \\
 & r \in K\langle\langle\Sigma^*\rangle\rangle
\end{tikzcd}$$

Figure 8.2: A comparison of the proofs of Theorem 7.6 (left) and Theorem 8.7 (right).

This lemma states that G can be decomposed into an alphabetic morphism g and an unambiguous CFG G' such that the weighted language of G equals g applied to the language of G'. This part is very similar to Lemma 5.5, which is used in our proof of Theorem 7.6. That means the construction of g and G' coincides with our construction of an unweighted (P^1, Δ)-automaton $\mathcal{B}$ as well as the alphabetic morphism $h\colon \Delta \to K[\Sigma \cup \{\varepsilon\}]$ based on the $(\mathrm{P}^1, \Sigma, K)$-automaton $\mathcal{A}$ in the proof of Lemma 5.5 except of the fact that instead of an unambiguous CFG G' as the unweighted formalism, an unambiguous and ε-free automaton with pushdown storage is used in the latter. This relationship will be examined more closely in Example 8.9.

Example 8.9. Recall the unital valuation monoid $\mathcal{K}^{\lambda}_{\text{disc}} = (\tilde{\mathbb{R}}, \max, \mathrm{val}^{\lambda}_{\text{disc}}, -\infty, 0)$ from Example 2.5 for $\lambda = 0.5$ and let $G = (\{S, A, B\}, \Sigma, S, P, \mathrm{wt}_G)$ be the $\mathcal{K}^{\lambda}_{\text{disc}}$-WCFG from Example 8.5. The aim of this example is to illustrate the construction of an alphabetic morphism g as well as an unambiguous CFG G' as in Lemma 8.8 on the basis of G. Furthermore, the relationship to the proof of Lemma 5.5 will be made clear.

For this it is required that the initial grammar is in head normal form, i.e., each production is of the form $A \to xB_1 \ldots B_k$, $k \in \mathbb{N}$, where A, $B_1, \ldots, B_k$ are nonterminals and x is a terminal or ε. In [DV13] it was shown that each K-WCFG can be transformed accordingly. Clearly, the example grammar G is in head normal form.

The idea behind the construction is the following: for each production $p = A \to xB_1 \ldots B_k \in P$ the resulting CFG G' has a production $A \to pB_1 \ldots B_k$ and we define $g(p) = \mathrm{wt}_G(p).x$.

The application of the construction on G yields the CFG $G' = (\{S, A, B\}, P, S, P')$ with the set P' of productions

$$\begin{aligned}
&p'_1 = (S \to p_1 SB), && p'_3 = (A \to p_3), && p'_5 = (S \to p_5 A),\\
&p'_2 = (S \to p_2 B), && p'_4 = (S \to p_4 SA), && p'_6 = (B \to p_6),\\
& && && p'_7 = (S \to p_7 SS),
\end{aligned}$$

and the alphabetic morphism $g : P \to \mathcal{K}^{\lambda}_{\text{disc}}[\Sigma \cup \{\varepsilon\}]$ such that

$$
\begin{array}{lll}
g(p_1) = 2.a & g(p_3) = 2.a & g(p_5) = 1.b \\
g(p_2) = 2.a & g(p_4) = 1.b & g(p_6) = 1.b \\
 & & g(p_7) = 0.\varepsilon\ .
\end{array}
$$

In fact, except for syntactical differences of the formalisms used, the construction is the same as the construction in the proof of Lemma 5.5, instantiated with the storage type P^1. For this, compare with Example 5.6.

By transforming the initial WCFG first into head normal form, it is quite close to an automaton – in every production exactly one symbol or ε is generated. In the next step then this symbol or ε in a production is replaced by the production itself. The same is performed with the transition of the automaton $\mathcal{A}$ in Example 5.6, compare

$$\frac{p_1 = S \to aSB}{p_1' = S \to p_1 SB} \qquad \text{and} \qquad \frac{\tau_1 = (q, a, \text{bottom}, q, \text{push}(B, f_{\text{id}}))}{\tau_1' = (q, \tau_1, \text{bottom}, q, \text{push}(B, f_{\text{id}}))}\ .$$

These new symbols are mapped in both constructions by an alphabetic morphism to a monome $w.a$, where a is the original symbol and w is the weight of the given production or transition, compare

$$g(p_1) = 2.a \qquad \text{and} \qquad h(\tau_1) = 2.a\ ,$$

where h is the alphabetic morphism constructed in Example 5.6.

In Lemma 5.5 it is additionally required that the constructed automaton is ε-free. Indeed, also the grammar resulting from the construction presented above derives in each production exactly one symbol.

Now consider the word $ab \in \Sigma^*$. It is easy to see that this string is recognized by $\mathcal{A}$ with the computation θ and generated by G with the derivation d, where

$$\theta = (q, a, \text{bottom}, q, \text{push}(B, f_{\text{id}}))(q, b, \text{top} = B, q, \text{pop})(q, \varepsilon, \text{bottom}, f, \text{stay}(\$)),$$

$$d = (S \to aB)(B \to b).$$

The computation corresponding to θ according to the proof of Lemma 5.5 of the automaton $\mathcal{B}$, that is constructed in Example 5.6 from the automaton $\mathcal{A}$, is the computation

$$\theta' = (q, \tau_1, \text{bottom}, q, \text{push}(B, f_{\text{id}}))(q, \tau_5, \text{top} = B, q, \text{pop})(q, \tau_7, \text{bottom}, f, \text{stay}(\$)),$$

recognizing the string $\theta = \tau_1\tau_5\tau_7 \in T^*$ (recall that the transitions of θ are used as symbols in the transitions of θ'). In a similar way, the derivation of G' corresponding to d is the derivation

$$d' = (S \to p_2 B)(B \to p_6),$$

generating $d = p_2 p_6 \in P^*$. Moreover, we have that

$$h(\theta') = 2.5.ab \qquad \text{and} \qquad g(d') = 2.5.ab\ .$$

□

Now, where we have seen that the first part of the proofs of Theorem 7.6 and Theorem 8.7 are very similar, we want to compare the remaining parts. These include differences in the overview of Figure 8.1 and, in fact, two different approaches were used. In the proof of our CS theorem the unweighted (P^1, Δ)-automaton $\mathcal{B}$ is decomposed into a storage part, represented by a set $B(\Omega, c)$ of behaviours of P^1 and a configuration c, and a simple transducer $\mathcal{M}$. This transducer can be decomposed further into two letter-to-letter morphisms h_1 and h_2 and a regular language R such that $L(\mathcal{B}) = h_2(h_1^{-1}(B(\Omega, c)) \cap R)$. The language R ranges over the transitions of $\mathcal{M}$ as symbols.

In Theorem 8.7, however, another kind of decomposition is used. Since in the first part of the proof, by Lemma 8.8, an unweighted CFG G over some alphabet Δ is obtained, the original Chomsky-Schützenberger theorem can be applied to obtain a Dyck language D_Y over some parenthesis alphabet Y, a regular language $H \subseteq (Y \cup \bar{Y})^*$ and an alphabetic morphism $g_2\colon Y \cup \bar{Y} \to \mathbb{B}[\Delta \cup \{\varepsilon\}]$ such that $L(G) = g_2(D_Y \cap H)$.[1]

Finally, in both the proof of Theorem 7.6 and the proof of Theorem 8.7, the alphabetic morphisms h and h_2 as well as g and g_2 are composed, respectively.

Now if we look at the decomposition and the illustration in Figure 8.1, one could suspect, that $h_1^{-1}(B(\Omega, c))$ in the left diagram corresponds in some way to the Dyck language D_Y in the right diagram. To show that this is not the case, we want to investigate the CS decomposition of G by an example.

Example 8.10. The aim of this example is to illustrate the Chomsky-Schützenberger decomposition of a context-free language over Δ, represented by a context-free grammar G', into a Dyck language over some parenthesis alphabet Y, a regular language H over $Y \cup \bar{Y}$ and an alphabetic morphism $g_2\colon Y \cup \bar{Y} \to \mathbb{B}[\Delta \cup \{\varepsilon\}]$.

Note that we do not want to explain the construction formally. This is folklore and can be looked up, for example, in [ABB97].

Recall the context-free grammar $G' = (\{S,A,B\}, P, S, P')$ from Example 8.9. As parentheses we use the nonterminals and terminals of G', i.e., we have

$$Y = \{\, (_x \mid x \in \{S,A,B\} \cup \{p_1, \dots, p_7\}\}, \text{ and}$$

$$\bar{Y} = \{\,)_x \mid x \in \{S,A,B\} \cup \{p_1, \dots, p_7\}\}.$$

Furthermore, we define the alphabetic morphism $g_2\colon (Y \cup \bar{Y}) \to \mathbb{B}[P \cup \{\varepsilon\}]$ such that for every $a \in Y \cup \bar{Y}$

$$g_2(a) = \begin{cases} \varepsilon & \text{if } a = (_x \text{ with } x \in \{A, S, B\} \text{ or } a \in \bar{Y}, \text{ and} \\ x & \text{if } a = (_x \text{ with } x \in \{p_1 \dots p_7\}\ . \end{cases}$$

[1] Note that in [DV13] the alphabetic morphism was considered in an unweighted manner as $g_2\colon Y \cup \bar{Y} \to \Delta \cup \{\varepsilon\}$. However, this is equivalent to our notation.

Now consider the word $w = (_S(_{p_1})_{p_1})_S$. Clearly, w is in D_Y and $h(w) = p_1$. But p_1 is not in $L(G')$, which can be checked by a regular language H. This language encodes the order of nonterminal calls in its words, which depends on the productions of G'. For this, each production $p = (A \to xB_1 \dots B_n)$ (note that we can assume head normal form) is translated into a sequence $)_A(_{B_n}\dots(_{B_1}(_a)_a$ in the case of $x = a$ for some $a \in \Delta$ and $)_A(_{B_n}\dots(_{B_1}$ otherwise. The regular language H contains all words which can be built up from these sequences. Thereby, $)_A$ represents the call of the nonterminal from the left side of p and $(_{B_n}\dots(_{B_1}$ are intended to make sure that the nonterminals $B_1, \dots, B_n$ are called in the further derivation. As we consider leftmost derivations, the order of the nonterminals has to be inverted, since the nonterminal that stands on the rightmost position of the sequence is called first. In this context, the intersection with D_Y ensures that the rightmost nonterminal of such a sequence is called next by checking that the corresponding parenthesis is closed in the following sequence.

Consider as an example the production

$$S \to p_1SB$$

from P'. For the nonterminal S a parenthesis has to be closed, represented by $)_S$, since this nonterminal has been called by the production. The nonterminals S and B now can be called next, represented by opening parenthesis $(_S$ and $(_B$. As we consider a leftmost derivation, their order has to be inverted such that $(_S$ is right of $(_B$. Finally, the generated nonterminal p_1 is encoded by $(_{p_1})_{p_1}$ and we obtain the sequence $)_S(_B(_S(_{p_1})_{p_1}$. The set of all sequences obtained from the productions of G' in this way is denoted by R_1. Additionally, there must be an opening bracket for the call of the initial nonterminal S and therefore, H is the set

$$H = (_SR_1^* \,.$$

Clearly, H is regular. □

Now, where we have seen how the CFG G' can be decomposed according to the original CS theorem, we can compare the resulting components with the constituents obtained from our decomposition of an (P^1, Δ)-automaton $\mathcal{B}$.

First, it is noticeable the alphabets of the regular languages R and H and, therefore, also the domain of the mappings h_2 and g_2 differ. While Φ contains as symbols transitions of the simple transducer $\mathcal{M}$, the parenthesis alphabet Y in $Y \cup \bar{Y}$ is built up from nonterminals and terminals of G'.

As this difference is due to construction, which is left open in the decomposition of G' in [DV13], we do not want to restrict our comparison to these dissimilar alphabets. Instead we will show next, based on an Example, that the words in $h_1^{-1}(B(\Omega, c))$ differ structurally from Dyck words.

Example 8.11. The aim of this example is to compare the Chomsky-Schützenberger decomposition of a context-free grammar G' as in Example 8.10 with the decomposition of some

ε-free (P^1, Δ)-automaton $\mathcal{B}$ according to Theorem 6.6 and 7.1. In particular, it is intended to clarify how the words in $h_1^{-1}(B(\Omega, c))$ differ structurally from words in D_Y. For this, recall

- the (P^1, T)-automaton $\mathcal{B} = (\{q, f\}, T, (\$, c), q, \{f\}, T_{\mathcal{B}})$ and the alphabetic morphism $h\colon T \to \mathcal{K}^{\lambda}_{\mathrm{disc}}[\Sigma \cup \{\varepsilon\}]$ from Example 5.6,
- the alphabet Ω and the configuration $c \in C$ from Example 6.8, and
- the alphabet Φ, the regular language $R \subseteq \Phi^*$, and the letter-to letter morphisms $h_1\colon \Phi \to \mathbb{B}[\Omega]$ and $h_2\colon \Phi \to \mathbb{B}[T]$ from Example 7.2.

Furthermore, recall

- the CFG $G' = (\{S, A, B\}, P, S, P')$ from Example 8.9, and
- the alphabet $Y \cup \bar{Y}$, the regular language $H \subseteq Y \cup \bar{Y}$, and the alphabetic morphism $g_2\colon (Y \cup \bar{Y}) \to \mathbb{B}[P \cup \{\varepsilon\}]$ from Example 8.10.

Now consider the computation

$$\theta = (q, \tau_1, \mathrm{bottom}, q, \mathrm{push}(B, f_{\mathrm{id}}))(q, \tau_5, \mathrm{top} = B, q, \mathrm{pop})(q, \tau_7, \mathrm{bottom}, f, \mathrm{stay}(\$))$$

recognizing the word $\tau = \tau_1\tau_5\tau_7 \in T^*$, and the derivation

$$d = (S \to p_2 B)(B \to p_6)$$

generating the word $p = p_2 p_6 \in P^*$, which are compared in Example 8.9.

The element from the set $h_1^{-1}(B(\Omega, c)) \cap R$, which is mapped by h_2 to τ, is the word

$$(q, (\mathrm{bottom}, \mathrm{push}(B, f_{\mathrm{id}})), q, \tau_1)(q, (\mathrm{top} = B, \mathrm{pop}), q, \tau_5)(q, (\mathrm{bottom}, \mathrm{stay}(\$)), f, \tau_7),$$

that is denoted in the following by w_1. Note that also $w_1 \in h_1^{-1}(B(\Omega, c))$. Recall that each symbol of Φ is a transition from some simple transducer $\mathcal{M}$.

On the other hand, the element from $D_Y \cap H$, which is mapped by g_2 to p, is the word

$$w_2 = (_S)_S(_B(_{p_2})_{p_2})_B(_{p_6})_{p_6}.$$

Now let us compare w_1 and w_2. Apart from the different alphabets, we have that w_1 is not well-bracketed, which is easy to see by the fact that w_1 consists of 3 symbols. This stands in contrast to w_2, which is a Dyck word. In the further we want to give some reasons for this difference.

One problem is that it is not clear how to partition Φ into some parenthesis alphabets Φ_1 and $\overline{\Phi}_1$, to compare $h_1^{-1}(B(\Omega, c))$ with D_Y. In $Y \cup \bar{Y}$ each opening parenthesis from Y can be assigned to exactly one closing parenthesis from $\bar{Y}$ – for example $(_S$ is assigned to $)_S$. As one approach we could take transitions of $\mathcal{M}$, which contain in their input symbols instructions pushing some element B, as opening parentheses. As corresponding closing parentheses we could then assign those transitions, which pop this element in their input symbol. Thus, for example, to the opening parenthesis

$$(q,(\text{bottom},\text{push}(B,f_{\text{id}})),q,\tau_1)$$

the closing parenthesis

$$(q,(\text{top}=B,\text{pop}),q,\tau_5)$$

could be assigned. However, the symbol B could also be pushed combined with another predicate – we might as well choose

$$(q,(\text{top}=B,\text{push}(B,f_{\text{id}})),q,\tau_6)$$

as opening parenthesis. Furthermore, we could imagine that there is an additional transition with translates the input symbol $(\text{top}=B,\text{push}(B,f_{\text{id}}))$ into another output symbol τ_8. Thus, the assignment of multiple opening parentheses to one closing parenthesis, or vice versa, would be possible. We call this difference in the following **D1**.

Additionally, in the context of this approach it is not clear how to handle a transition with a stay instruction. Such a transition could represent both – an opening parenthesis since a new symbol is put an the pushdown, and a closing parenthesis as the previous topmost pushdown symbol is removed. In the following we refer to this difference as **D2**.

A third difference is not visible in our example but has to be considered as well. In general, the pushdown has not to be “emptied” in the end of a computation of some (P^1,Δ)-automaton, which means that more than one pushdown cell can be left on the pushdown. This then is also reflected in words of the corresponding set $h_1^{-1}(B(\Omega,c))$ since not all opened parenthesis have to be closed. We call this difference in the following **D3**.

□

In addition to the differences **D1** to **D3** we want to mention an important similarity. Both the words in $h_1^{-1}(B(\Omega,c))\cap R$ and the words in $D_Y\cap H$ encode the computations, respectively derivations, of $\mathcal{B}$, respectively G'. This leads to the question whether a transformation of $\mathcal{B}$ into some equivalent automaton, avoiding the mentioned problems, as for example stay instructions, is possible. This would result in a better comparability of the sets obtained by the respective decomposition.

In the first instance, this question can be answered positively. Thus, for example the problem **D2** can be avoided by simulating a $\text{stay}(B)$ by $\text{pop};\text{push}(B,f_{\text{id}})$ as described in [Eng86]. Moreover it was shown in Chapter 4 that each (P^1,Δ)-automaton $\mathcal{B}$ can be transformed into some equivalent automaton $\mathcal{B}'$ that empties its pushdown in the end of each computation (apart from one last pushdown cell). This could help to obviate **D3**. However, in both constructions mentioned the ε-freeness required in Theorem 6.6 is not preserved.

Also the difference **D1** could probably, at least partially, be avoided since in general for pushdown automata no test of the topmost pushdown symbol before a push is needed. This information can be encoded into the state behaviour, compare to [Vog14, algorithm in Section 2.1.2]. However, to realize that the topmost pushdown symbol is only tested before a pop, we would have to introduce a new predicate and therefore change the storage type P^1. Moreover, it is not clear how to handle the mentioned nondeterminism in transitions.

Despite all the differences we claim that Theorem 8.7 follows from our Theorem 7.6. We will not give a formal proof for this claim in this thesis. However, we want to hint at the idea of a possible construction by the following example.

Example 8.12. The aim of this example is to indicate how the decomposition in Theorem 8.7 follows from the constructions used in the proof of Theorem 7.6. As we have already shown the similarity of the weight separation in both theorems, we will restrict ourselves in the following to the unweighted parts. That means we will illustrate, given an alphabet Ω and an configuration $c \in C$, an alphabet Φ, a regular language $R \subseteq \Phi^*$ and two letter-to-letter morphisms h_1 and h_2 as in Theorem 7.6, how to construct an alphabet $Y \cup \bar{Y}$, a regular language $H \subseteq (Y \cup \bar{Y})^*$ and an alphabetic morphism g_2 such that

$$h_2(h_1^{-1}(B(\Omega, c)) \cap R) = g_2(D_Y \cap H).$$

For this, recall

- the alphabet Ω and the initial configuration $c \in C$ from Example 6.8, and
- the alphabet Φ, the regular language $R \subseteq \Phi^*$, and the letter-to letter morphisms $h_1 \colon \Phi \to \mathbb{B}[\Omega]$ and $h_2 \colon \Phi \to \mathbb{B}[T]$ from Example 7.2.

Moreover, recall that the symbols of Φ are transitions of some simple transducer $\mathcal{M}$.

The elements of the alphabet $Y \cup \bar{Y}$ are built up from the states, pushdown symbols in the input symbols, and the output symbols of the transitions of $\mathcal{M}$ occurring in the symbols of Φ. Additionally, we need a parenthesis pair $\{(_b,)_b\}$ to represent the predicate bottom.[2] This leads to the sets

$$Y = \{(_q, (_f, (_A, (_B, (_\$, (_{\tau_1}, \dots, (_{\tau_7}\} \cup \{(_b\}, \text{ and}$$

$$\bar{Y} = \{)_q,)_f,)_A,)_B,)_\$,)_{\tau_1}, \dots,)_{\tau_7}\} \cup \{)_b\}.$$

The idea behind the construction of the regular language H is very similar to the construction in Example 8.10. Each element of Φ is mapped to a sequence of parentheses with the same intention as in Example 8.10, but additionally also the input symbols of each transition, which are elements from Ω, have to be associated to parentheses. For this, the predicates and instructions are handled differently as follows: each predicate $\mathrm{top} = B$ is represented by

[2] Note that this could in general cause problems (since the predicate bottom can be simulated by a predicate $\mathrm{top} = \gamma$ if γ is the current bottom symbol) and has to be handled appropriately in a formal construction. However, since in our particular example it suffices to use these additional parentheses, we will not go into more detail.

Table 8.1: Construction of R_1 from Φ.

Φ	R_1
$(q,(\text{bottom},\text{push}(B,f_{\text{id}})),q,\tau_1)$	$)_q)_b(_b(_B(_q(_{\tau_1})_{\tau_1}$
$(q,(\text{top}=A,\text{pop},),q,\tau_2)$	$)_q)_A(_q(_{\tau_2})_{\tau_2}$
$(q,(\text{top}=B,\text{push}(B,f_{\text{id}})),q,\tau_3)$	$)_q)_B(_B(_B(_q(_{\tau_3})_{\tau_3}$
$(q,(\text{bottom},\text{push}(A,f_{\text{id}})),q,\tau_4)$	$)_q)_b(_b(_A(_q(_{\tau_4})_{\tau_4}$
$(q,(\text{top}=B,\text{pop},),q,\tau_5)$	$)_q)_B(_q(_{\tau_5})_{\tau_5}$
$(q,(\text{top}=A,\text{push}(A,f_{\text{id}})),q,\tau_6)$	$)_q)_A(_A(_A(_q(_{\tau_6})_{\tau_6}$
$(q,(\text{bottom},\text{stay}(\$)),f,\tau_7)$	$)_q)_b(_\$(_f(_{\tau_7})_{\tau_7}$

the parenthesis $)_B$ and the predicate bottom is represented by $)_b$. On the other hand, each instruction push(B,f_{id}) is assigned to a parenthesis sequence $(_A(_B$, where A is the pushdown symbol tested by the preceding predicate (or $(_b(_B$ if bottom is tested), and each instruction stay($) is represented by $(_\$$. This leads to the following set R_1 as given in Table 8.1 constructed from Φ. Note that in this table each parenthesis sequence of R_1 stands right of the element of Φ from which it is constructed.

Since each computation of $\mathcal{M}$ (and therefore, each word of R) starts with the initial state q and the predicate bottom in the first transition, we have to add $(_b(_q$ at the beginning of each word over R_1. Furthermore, we add $)_f)_\$$ at the end of each word over R_1 since each computation of $\mathcal{M}$ (respectively each word of R) ends up with a transition containing the instruction stay($) and the final state f. Therefore, we obtain the regular language

$$H = (_b(_q\, R_1^*\,)_f)_\$.$$

Furthermore, we define the alphabetic morphism $g_2\colon (Y \cup \bar{Y}) \to \mathbb{B}[T \cup \{\varepsilon\}]$ such that for every $a \in Y \cup \bar{Y}$

$$g_2(a) = \begin{cases} \varepsilon & \text{if } a = (_x \text{ with } x \in \{A,B,\$,b\} \text{ or } a \in \bar{Y}, \text{ and} \\ x & \text{if } a = (_x \text{ with } x \in \{\tau_1 \dots \tau_7\}\ . \end{cases}$$

In order to understand the close relationship between strings in $h_1^{-1}(B(\Omega,c)) \cap R$ and $D_Y \cap H$, consider the string θ from Example 7.2 together with the string $w \in D_Y \cap H$, whose construction from θ we described above.

$$\theta = (q,(\text{bottom},\text{push}(B,f_{\text{id}})),q,\tau_1)(q,(\text{top}=B,\text{pop}),q,\tau_5)(q,(\text{bottom},\text{stay}(\#)),f,\tau_7)$$

$$w = \boxed{(_b(_q}\ \boxed{)_q)_b(_b(_B(_q(_{\tau_1})_{\tau_1}}\ \boxed{)_q)_B(_q(_{\tau_5})_{\tau_5}}\ \boxed{)_q)_b(_\$(_f(_{\tau_7})_{\tau_7}}\ \boxed{)_f)_\$}$$

The three parenthesis sequences in the middle of w, which are coloured with light gray, are the elements in R_1 constructed from the transitions of θ. Additionally, the sequences $(_b(_q$ and $)_f)_\$$ in dark gray are added as described above to obtain a well-bracketed word.

Furthermore, we have that

$$h_2(\theta) = \tau_1\tau_5\tau_7 = g_2(w). \qquad \square$$

9 Conclusion

In this work we introduced automata with storage and extended this concept to weighted automata with storage over unital valuation monoids as weight algebra.

We proved formally, that the weighted pushdown automata of [DV13] are expressively equivalent to weighted automata with 1-iterated pushdown storage.

The main result of this work is a Chomsky-Schützenberger theorem for weighted automata with storage. We obtained this by separating first the weight and secondly the storage from an (S, Σ, K)-automaton.

Finally, we compared our CS result, instantiated with the 1-iterated pushdown storage, informally with the CS theorem of [DV13].

Bibliography

[ABB97] J.-M. Autebert, J. Berstel, and L. Boasson. Context-free Languages and Pushdown Automata. In: *Handbook of Formal Languages*. Ed. by G Rozenberg and A Salomaa. Vol. 1. Springer-Verlag, 1997, 111–174.

[CDH08] K Chatterjee, L Doyen, and Th. Henzinger. Quantitative languages. In: *Proceedings of the 17th International Conference on Computer Science Logic (CSL 2008)*. Ed. by M Kaminski and S Martini. Vol. 5213. Lecture Notes in Computer Science. Springer-Verlag, 2008, 385–400.

[CS63] N. Chomsky and M.P. Schützenberger. The algebraic theory of context-free languages. In: *Computer Programming and Formal Systems*. North-Holland, Amsterdam, 1963, 118–161.

[Den15] T. Denkinger. A Chomsky-Schützenberger representation for weighted multiple context-free languages. In: *The 12th International Conference on Finite-State Methods and Natural Language Processing (FSMNLP 2015)*. accepted for publication. 2015.

[DG86] W. Damm and A. Goerdt. An automata-theoretical characterization of the OI-hierarchy. *Information and Control* 71 (1986), 1–32.

[DM10] M. Droste and I. Meinecke. Describing average- and longtime-behavior by weighted MSO logics. In: *Mathematical Foundations of Computer Science (MFCS)*. LNCS 6281. Springer, 2010, 537–548.

[DM11] M. Droste and I. Meinecke. Weighted automata and regular expressions over valuation monoids. *International Journal of Foundations of Computer Science* 22(8) (2011), 1829–1844.

[DV13] M. Droste and H. Vogler. The Chomsky-Schützenberger theorem for quantitative context-free languages. In: *Proceedings of Developments in Language Theory 2013*. Vol. 7907. Lecture Notes in Computer Science. see also: *International Journal of Foundations of Computer Science* 25(8):955–969, 2014. Springer-Verlag, 2013, 203–214.

[Eil74] S. Eilenberg. *Automata, Languages, and Machines – Volume A*. Vol. 59. Pure and Applied Mathematics. Academic Press, 1974.

[Eng83] J. Engelfriet. Iterated pushdown automata and complexity classes. In: *Proceedings of the fifteenth annual ACM Symposium on Theory of Computing (STOC)*. ACM New York, 1983, 365–373.

[Eng86] J. Engelfriet. *Context–free grammars with storage*. Tech. rep. 86-11. see also: arXiv:1408.0683 [cs.FL], 2014. University of Leiden, 1986.

[EV86] J. Engelfriet and H. Vogler. Pushdown machines for the macro tree transducer. *Theoretical Computer Science* 42.3 (1986), 251–368.

[EV88] J. Engelfriet and H. Vogler. High level tree transducers and iterated pushdown tree transducers. *Acta Informatica* 26 (1988), 131–192.

[FV15] S. Fratani and E.M. Voundy. Dyck-based characterizations of Indexed Languages. published on arXiv http://arxiv.org/abs/1409.6112. 2015.

[GG69] S. Ginsburg and S.A. Greibach. Abstract Families of Languages. *Memoirs of the American Mathematical Society* 87 (1969), 1–32.

[GG70] S. Ginsburg and S.A. Greibach. Principal AFL. *Journal of Computer and System Sciences* 4 (1970), 308–338.

[GGH69] S. Ginsburg, S.A. Greibach, and J. Hopcroft. *Studies in Abstract Families of Languages*. 87. American Mathematical Society, 1969, pp. 1–51.

[Har78] M.A. Harrison. *Introduction to Formal Language Theory*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1978. ISBN: 0201029553.

[Hul09] M. Hulden. Parsing CFGs and PCFGs with a Chomsky-Schützenberger representation. In: *Human Language Technology. Challenges for Computer Science and Linguistics*. Ed. by Z. Vetulani. Vol. 6562. Lecture Notes in Computer Science. Springer, 2009, 151–160.

[HV15] Luisa Herrmann and Heiko Vogler. A Chomsky-Schützenberger Theorem for Weighted Automata with Storage. In: *Algebraic Informatics*. Ed. by Andreas Maletti. Vol. 9270. Lecture Notes in Computer Science. Springer International Publishing, 2015, 115–127. DOI: 10.1007/978-3-319-23021-4_11.

[Kam07] M. Kambites. Formal languages and groups as memory. arXiv:math/0601061v2 [math.GR] 19 Oct 2007. 2007.

[Kan14] M. Kanazawa. Multidimensional trees and a Chomsky-Schützenberger-Weir representation theorem for simple context-free tree grammars. *Journal of Logic and Computation* (2014).

[Mas74] A.N. Maslov. The hierarchy of indexed languages of an arbitrary level. *Soviet Mathematics Doklady* 15 (1974), 1170–1174.

[Mas76] A.N. Maslov. Multilevel stack automata. *Problems of Information Transmission* 12 (1976), 38–42.

[Sco67] D. Scott. Some definitional suggestions for automata theory. *Journal of Compututer and System Sciences* 1 (1967), 187–212.

[SS78] A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science, Springer-Verlag, 1978.

[Vog14] Heiko Vogler. *Algorithmen und Datenstrukturen; Vorlesungsskript 13. Auflage*. 2014.

[Wei88] D.J. Weir. Characterizing Mildly Context-Sensitive Grammar Formalisms. PhD thesis. University of Pennsylvania, 1988.

[YKS10] R. Yoshinaka, Y. Kaji, and H. Seki. Chomsky-Schützenberger-type characterization of multiple context-free languages. In: *Language and Automata Theory and Applications (LATA)*. Vol. 6031. Lecture Notes in Computer Science. Springer-Verlag, 2010, 596–607.