

Approximation of weighted automata with storage

Tobias Denkinger

Faculty of Computer Science

Technische Universität Dresden

01062 Dresden, Germany

tobias.denkinger@tu-dresden.de

2017-03-30

Abstract

We use a non-deterministic variant of storage types to develop a framework for the approximation of automata with storage. This framework is used to provide an automata-theoretic view on the approximation of multiple context-free languages.

Contents

1	Introduction	2
2	Preliminaries	2
3	Automata with non-deterministic storage	3
3.1	Non-deterministic storage types	3
3.2	Automata with non-deterministic storage	4
4	Approximation of automata with storage	7
4.1	Superset approximations	8
4.2	Subset approximations	10
4.3	Approximation of weighted automata with storage	11
5	Approximation of multiple context-free languages	13

1 Introduction

In the application of formal languages, approximation is a well-established concept (see Nederhof [Ned00a] for an overview). For a context-free grammar it is common (but not exclusive [Ned00b, CPV⁺06]) to construct a pushdown automaton and then approximate this automaton [KdT81, Pul86, LL87, BS90, PW91, Eva97, Joh98], e.g. by restricting the height of the pushdown. Automata with storage [Sco67, Eng86, Eng14] are a generalisation of pushdown automata and were studied in recent literature [HV15, VDH16]. Multiple context-free languages [SMFK91, VSWJ87] are among the language classes captured by automata with storage [Den16] and their approximation was studied recently [BL05, vC12]. By attaching weights to the transitions of an automaton with storage, we can model, for example, the *multiplicity* with which a word belongs to a language or the *cost* of recognising a word. The resulting devices are called *weighted* automata with storage [HV15, VDH16].

We develop a framework to study the approximation of weighted automata with arbitrary storage. To deal with non-determinism that arises due to approximation, we introduce weighted automata with non-deterministic storage (Sec. 3). Our storage differs from Engelfriet’s [Eng86, Eng14] in two aspects: As instructions we allow binary relations instead of partial functions and each transition is associated with a weight from a semiring. Inspired by the storage simulation of Hoare [Hoa72] (also studied by Engelfriet and Vogler [EV86]), we formalise the strategy with which a storage type is approximated (the *approximation strategy*) as a partial function (Sec. 4). In contrast to Engelfriet and Vogler [EV86], we do not utilise flowcharts in our constructions. We use our framework to provide an automata-based approach to the approximation of multiple context-free languages (Sec. 5).

2 Preliminaries

The set of $\{0, 1, 2, \dots\}$ is denoted by \mathbb{N} and $\mathbb{N} \setminus \{0\}$ is denoted by \mathbb{N}_+ . The set $\{1, \dots, k\}$ is denoted by $[k]$ for every $k \in \mathbb{N}$. Let A be a set. The *power set* of A is denoted by $\mathcal{P}(A)$. By $\exists!$ we denote the quantifier “there is exactly one”.

Let A , B , and C be sets and let $r \subseteq A \times B$ and $s \subseteq B \times C$ be binary relations. We denote $\{(b, a) \in B \times A \mid (a, b) \in r\}$ by r^{-1} , $\{b \in B \mid (a, b) \in r\}$ by $r(a)$ for every $a \in A$, and $\bigcup_{a \in A'} r(a)$ by $r(A')$ for every $A' \subseteq A$. The *sequential composition* of r and s is the binary relation

$$r ; s = \{(a, c) \in A \times C \mid \exists b \in B: ((a, b) \in r) \wedge ((b, c) \in s)\}.$$

We call r an *endorelation* (on A) if $A = B$. A *semiring* is an algebraic structure $(K, +, \cdot, 0, 1)$ where $(K, +, 0)$ is a commutative monoid, $(K, \cdot, 1)$ is a monoid, 0 is absorptive with respect to \cdot , and \cdot distributes over $+$. We say that K is *complete* if it has a sum operation $\sum_I: K^I \rightarrow K$ that extends $+$ for each countable set I [DKV09, Ch. 1]. Let \leq be a partial order on K . We say that K is *positively \leq -ordered* if \leq is preserved by $+$.

The *set of partial functions from A to B* is denoted by $A \dashrightarrow B$. The *set of (total) functions from A to B* is denoted by $A \rightarrow B$. Note that every total function is a partial function and that every partial function is a binary relation. Let $f: A \dashrightarrow B$ be a partial function. The *domain of f* and the *image of f* are defined by $\text{dom}(f) = \{a \in A \mid \exists b \in B: f(a) = b\}$ and $\text{img}(f) = \{b \in B \mid \exists a \in A: f(a) = b\}$, respectively. Abusing the notation, we may sometimes write $f(a) = \text{undefined}$ to denote that $a \notin \text{dom}(f)$.

3 Automata with non-deterministic storage

In addition to the finite state control, automata with storage allow the checking and manipulation of a so-called storage configuration that comes from a possibly infinite set. We propose a syntactic extension of automata with storage where the set of unary functions (the *instructions*) is replaced by a set of binary relations on the configurations.

3.1 Non-deterministic storage types

Definition 3.1. A *non-deterministic storage type* (short: *nd storage type*) is a tuple $S = (C, P, R, c_i)$ where C is a set (of *configurations*), $P \subseteq \mathcal{P}(C)$ (*predicates*), $R \subseteq \mathcal{P}(C \times C)$ (*instructions*), and $c_i \in C$ (*initial storage configuration*). \square

Consider an nd storage type $S = (C, P, R, c_i)$. If every element of R is a partial function, we call S *deterministic*. The definitions of “deterministic nd storage type” in this paper and “storage type” in previous literature [Eng86, Eng14, HV15, VDH16] coincide.

Example 3.2. The deterministic nd storage type Count models simple counting (Engelfriet [Eng86, Eng14, Def. 3.4]): $\text{Count} = (\mathbb{N}, \{\text{true}, \text{iszero}\}, \{\text{inc}, \text{dec}\}, 0)$ where $\text{true} = \mathbb{N}$, $\text{iszero} = \{0\}$, $\text{inc} = \{(n, n+1) \mid n \in \mathbb{N}\}$ and $\text{dec} = \text{inc}^{-1}$. \square

Example 3.3. The following deterministic nd storage type models a pushdown stack (similar to Engelfriet [Eng86, Eng14, Def. 3.2]): $\text{PD}_\Gamma = (\Gamma^*, P_{\text{pd}}, R_{\text{pd}}, \varepsilon)$ where Γ is a finite set (*pushdown symbols*); $P_{\text{pd}} = \{\Gamma^*, \text{bottom}\} \cup \{\text{top}_\gamma \mid \gamma \in \Gamma\}$ with

$$\text{bottom} = \{\varepsilon\} \quad \text{and} \quad \text{top}_\gamma = \{\gamma w \mid w \in \Gamma^*\}$$

for every $\gamma \in \Gamma$; and $R_{\text{pd}} = \{\text{stay}, \text{pop}\} \cup \{\text{push}_\gamma, \text{stay}_\gamma \mid \gamma \in \Gamma\}$ with

$$\begin{aligned} \text{stay} &= \{(w, w) \mid w \in \Gamma^*\}, & \text{pop} &= \{(\gamma w, w) \mid w \in \Gamma^*, \gamma \in \Gamma\}, \\ \text{push}_\gamma &= \{(w, \gamma w) \mid w \in \Gamma^*\}, & \text{stay}_\gamma &= \{(\gamma' w, \gamma w) \mid w \in \Gamma^*, \gamma' \in \Gamma\} \end{aligned}$$

for every $\gamma \in \Gamma$. \square

Example 3.4. The nd storage type Count' models counting, but does not distinguish different odd numbers: $\text{Count}' = (\{2n \mid n \in \mathbb{N}\} \cup \{o\}, \{\text{true}, \text{iszero}\}, \{\text{inc}'\}, 0)$ where $\text{true} = \{2n \mid n \in \mathbb{N}\} \cup \{o\}$, $\text{iszero} = \{0\}$, and $\text{inc}' = \{(2n, o), (o, 2n) \mid n \in \mathbb{N}\}$. Note that $(\text{inc}')^{-1} = \text{inc}'$ and that Count' is *not* deterministic. \square

We call a nd storage type $S = (C, P, R, c_i)$ *finitely non-deterministic* (short: *finitely nd*) if the set $r(c)$ is finite for every $r \in R$ and $c \in C$. The nd storage type Count' , for example, is not finitely nd since $\text{inc}'(o) = \{2n \mid n \in \mathbb{N}\}$ is not finite.

3.2 Automata with non-deterministic storage

Definition 3.5. Let $S = (C, P, R, c_i)$ be a finitely nd storage type and Σ be a finite set. An (S, Σ) -automaton is a tuple $\mathcal{M} = (Q, T, Q_i, Q_f)$ where Q is a finite set (of *states*), T is a finite subset of $Q \times (\Sigma \cup \{\varepsilon\}) \times P \times R \times Q$ (*transitions*), $Q_i \subseteq Q$ (*initial states*), and $Q_f \subseteq Q$ (*final states*). \square

Let $\mathcal{M} = (Q, T, Q_i, Q_f)$ be an (S, Σ) -automaton and $S = (C, P, R, c_i)$. A *configuration* of \mathcal{M} is an element of $Q \times C \times \Sigma^*$. For every $\tau = (q, v, p, r, q') \in T$, the *transition relation* of τ is the endorelation \vdash_τ on the set of configurations of \mathcal{M} that contains $(q, c, vw) \vdash_\tau (q', c', w)$ for every $w \in \Sigma^*$ and $(c, c') \in r$ with $c \in p$. The *run relation* of \mathcal{M} is $\vdash_{\mathcal{M}} = \bigcup_{\tau \in T} \vdash_\tau$. The transition relations are extended to sequences of transitions by setting $\vdash_{\tau_1 \dots \tau_k} = \vdash_{\tau_1}; \dots; \vdash_{\tau_k}$ for every $k \in \mathbb{N}$ and $\tau_1, \dots, \tau_k \in T$. The *set of runs* of \mathcal{M} is the set

$$R_{\mathcal{M}} = \{\theta \in T^* \mid \exists q, q' \in Q, c, c' \in C, w, w' \in \Sigma^*: (q, c, w) \vdash_\theta (q', c', w')\}. \quad (1)$$

Let $w \in \Sigma^*$. The *set of runs of \mathcal{M} on w* is

$$R_{\mathcal{M}}(w) = \{\theta \in T^* \mid \exists q \in Q_i, q' \in Q_f, c' \in C: (q, c_i, w) \vdash_\theta (q', c', \varepsilon)\}. \quad (2)$$

The *language accepted by \mathcal{M}* is the set $L(\mathcal{M}) = \{w \in \Sigma^* \mid R_{\mathcal{M}}(w) \neq \emptyset\}$.

Let S be a finitely nd storage type, Σ be a finite set, and $L \subseteq \Sigma^*$. We call L (S, Σ) -recognisable if there is an (S, Σ) -automaton \mathcal{M} with $L = L(\mathcal{M})$.

Proposition 3.6. For every finitely nd storage type S there is a deterministic nd storage type S_{det} such that for every finite set Σ the class of (S, Σ) -recognisable languages is equal to the class of (S_{det}, Σ) -recognisable languages.

Proof. Let $S = (C, P, R, c_i)$. Using a power set construction, we obtain the deterministic nd storage type $S_{\text{det}} = (P(C), P_{\text{det}}, R_{\text{det}}, \{c_i\})$ where $P_{\text{det}} = \{p_{\text{det}} \mid p \in P\}$ with $p_{\text{det}} = \{d \subseteq C \mid d \cap p \neq \emptyset\}$ for every $p \in P$, and $R_{\text{det}} = \{r_{\text{det}} \mid r \in R\}$ with $r_{\text{det}} = \{(d, r(d)) \mid d \subseteq C\}$ for every $r \in R$.

Let $\mathcal{M} = (Q, T, Q_i, Q_f)$ be an (S, Σ) -automaton and $\mathcal{M}' = (Q', T', Q'_i, Q'_f)$ be an (S_{det}, Σ) -automaton. We say that \mathcal{M} and \mathcal{M}' are *related* if $Q = Q'$, $Q_i = Q'_i$, $Q_f = Q'_f$, and $T' = T_{\text{det}} = \{t_{\text{det}} \mid t \in T\}$ where $t_{\text{det}} = (q, v, p_{\text{det}}, r_{\text{det}}, q')$ for each $t = (q, v, p, r, q') \in T$. Clearly, for every (S, Σ) -automaton there is an (S_{det}, Σ) -automaton such that both are related and vice versa.

Now consider the (S, Σ) -automaton $\mathcal{M} = (Q, T, Q_i, Q_f)$ and the (S_{det}, Σ) -automaton $\mathcal{M}' = (Q, T_{\text{det}}, Q_i, Q_f)$ that are related. We extend $(\cdot)_{\text{det}}: T \rightarrow T_{\text{det}}$ to a function $(\cdot)_{\text{det}}: T^* \rightarrow T_{\text{det}}^*$ by point-wise application. We show for every $\theta \in T^*$ by induction on the length of θ that

$$\begin{aligned} \forall q, q' \in Q, c, c' \in C, w, w' \in \Sigma^*: \\ (q, c, w) \vdash_\theta (q', c', w') \iff \forall d \ni c: \exists! d' \ni c': (q, d, w) \vdash_{\theta_{\text{det}}} (q', d', w') \end{aligned} \quad (3)$$

holds. The induction base is given by the sequence of length 0. For the induction step we assume that (3) holds for all sequences of length n . Let $q, q' \in Q, c, c' \in C, w, w' \in \Sigma^*, \theta \in T^n$, and $\tau \in T$. We distinguish two cases.

Case 1: $(q, c, w) \vdash_{\theta\tau} (q', c', w')$. Then there are $\bar{c} \in C, u \in \Sigma^*$, and $\tau = (\bar{q}, v, p, r, q') \in T$ with $w = uvw'$ and $(q, c, uvw') \vdash_{\theta} (\bar{q}, \bar{c}, vw') \vdash_{\tau} (q', c', w')$. By (3) we know that for every $d \ni c$ there is exactly one $\bar{d} \ni \bar{c}$ with $(q, d, uvw') \vdash_{\theta_{\text{det}}} (\bar{q}, \bar{d}, vw')$. It remains to be shown that $(\bar{q}, \bar{d}, vw') \vdash_{\tau_{\text{det}}} (q', d', w')$ for exactly one $d' \ni c'$. By construction, we know that $\tau_{\text{det}} = (\bar{q}, v, p_{\text{det}}, f_{\text{det}}, q')$. By definition of p_{det} , the fact that $\bar{c} \in p$, and $\bar{c} \in \bar{d}$, we have that $\bar{d} \in p_{\text{det}}$. Then $(\bar{d}, d') \in r_{\text{det}}$ by the definition of $\vdash_{\tau_{\text{det}}}$ and d' is unique by construction of r_{det} . By definition of r_{det} , the fact that $(\bar{c}, c') \in r$, and $\bar{c} \in \bar{d}$, we have that $c' \in d'$. Finally, τ_{det} clearly goes from state \bar{q} to state q' and reads v .

Case 2: $\neg(q, c, w) \vdash_{\theta\tau} (q', c', w')$. This can have two reasons.

Case 2.1: There is no $(\bar{q}, \bar{c}, \bar{w}) \in Q \times C \times \Sigma^*$ such that $(q, c, w) \vdash_{\theta} (\bar{q}, \bar{c}, \bar{w})$. Then it follows by (3) that for every $d \ni c$ there is no $\bar{d} \ni \bar{c}$ such that $(q, d, w) \vdash_{\theta_{\text{det}}} (\bar{q}, \bar{d}, \bar{w})$ and hence there is no $d' \ni c'$ such that $(c, d, w) \vdash_{(\theta\tau)_{\text{det}}} (c', d', w')$.

Case 2.2: There are $(\bar{q}, \bar{c}, u, v) \in Q \times C \times \Sigma^* \times (\Sigma \cup \{\varepsilon\})$ such that $w = uvw'$ and $(q, c, w) \vdash_{\theta} (\bar{q}, \bar{c}, vw')$, but for none of them exists a transition $\tau = (\bar{q}, v, p, r, q)$ such that $(\bar{q}, \bar{c}, vw') \vdash_{\tau} (q', c', w')$. Then by (3) we know that for every $d \ni c$ there is some $\bar{d} \ni \bar{c}$ such that $(q, d, uvw') \vdash_{\theta_{\text{det}}} (\bar{q}, \bar{d}, vw')$. It remains to be shown that there is no $d' \ni c'$ such that $(\bar{q}, \bar{c}, vw') \vdash_{\tau_{\text{det}}} (q', c', w')$. If such a d' were to exist, then there would exist a transition $\tau_{\text{det}} = (\bar{q}, v, p_{\text{det}}, r_{\text{det}}, q') \in T_{\text{det}}$ such that $\bar{d} \in p_{\text{det}}$ and $(\bar{d}, d') \in r_{\text{det}}$. Then by the construction there would be some $\bar{c} \in \bar{d}$ such that $\bar{c} \in p$ and $(\bar{c}, c') \in r$ which contradicts the assumption that there is no transition $\tau = (\bar{q}, v, p, r, q)$ such that $(\bar{q}, \bar{c}, vw') \vdash_{\tau} (q', c', w')$.

We then obtain $L(\mathcal{M}) = L(\mathcal{M}')$ from (3) and from $\{c_i\}$ being the initial storage configuration of \mathcal{M}' . ■

For practical reasons it might sometimes be preferable to avoid the construction of power sets. The proof presented here, however, only shows containment rather than equality for the considered language classes.

Proposition 3.7. For every finitely nd storage type S there is a deterministic nd storage type S' with the same set of configurations such that for every finite set Σ the class of (S, Σ) -recognisable languages is contained in the class of (S', Σ) -recognisable languages.

Proof. Let $S = (C, P, R, c_i)$ be a nd storage type. We construct the deterministic nd storage type $S' = (C, P, R', c_i)$ where R' is constructed as follows: Let $r \in R$ and $r(c)_1, \dots, r(c)_{m_{r,c}}$ be a fixed enumeration of the elements of $r(c)$ for every $c \in C$. Furthermore, let $k_r = \max\{|r(c)| \mid c \in C\}$. We define for each $i \in [k_r]$ an instruction r'_i by $r'_i(c) = r(c)_i$ if $i \leq m_{r,c}$ and $r'_i(c) = \text{undefined}$ otherwise. Let R' contain the instruction r'_i for every $r \in R$ and $i \in [k_r]$.

Now let $\mathcal{M} = (Q, T, Q_i, Q_f)$ be an (S, Σ) -automaton. We construct the (S', Σ) -automaton $\mathcal{M}' = (Q, T', Q_i, Q_f)$ where T' contains for every transition $t = (q, v, p, r, q') \in$

T and $i \in [k_r]$ the transition $t'_i = (q, v, p, r'_i, q')$. Then

$$\vdash_{\mathcal{M}} = \bigcup_{t \in T} \vdash_t = \bigcup_{t=(q,v,p,r,q') \in T} \bigcup_{i \in [k_r]} \vdash_{t'_i} = \bigcup_{t' \in T'} \vdash_{t'} = \vdash_{\mathcal{M}'}$$

and thus $L(\mathcal{M}) = L(\mathcal{M}')$. ■

The author conjectures that there are an nd storage type S , a finite set Σ , and an (S', Σ) -recognisable language L (where S' is defined as in the proof of Prop. 3.7) such that L is *not* (S, Σ) -recognisable.

Proposition 3.8. Let Σ be a finite set, $S = (C, P, R, c_i)$ be an nd storage type, and L be an (S, Σ) -recognisable language. If C is finite, then L is recognisable (by a finite state automaton).

Proof. We will use non-deterministic finite-state automata with extended transition function from Hopcroft and Ullman [HU79, Sec. 2.3] in a notation similar to that of automata with storage (we leave out the storage-related parts of the transitions).

Let $\mathcal{M} = (Q, T, Q_i, Q_f)$. We construct the fsa $\mathcal{M}' = (Q \times C, \Sigma, T', Q_i \times \{c_i\}, Q_f \times C)$ where $T' = \{((q, c), v, (q', c')) \mid (q, v, p, r, q') \in T, (c, c') \in r, c \in p\}$.

It remains to be shown that $L(\mathcal{M}) = L(\mathcal{M}')$. For this we first show by induction on the length of runs that

$$\begin{aligned} \forall q, q' \in Q, c, c' \in C, w, w' \in \Sigma^*: \\ (q, c, w) \vdash_{\mathcal{M}}^* (q', c', w') \iff ((q, c), w) \vdash_{\mathcal{M}'}^* ((q', c'), w'). \end{aligned} \quad (4)$$

The induction base is given by runs of length 0. For the induction step we assume that (4) holds for all runs of length n and let $(q, c, w) \vdash_{\mathcal{M}}^* (\bar{q}, \bar{c}, \bar{w})$ be witnessed by a run of length $n + 1$. We derive

$$\begin{aligned} (q, c, w) \vdash_{\mathcal{M}}^* (\bar{q}, \bar{c}, \bar{w}) \\ \iff \exists q' \in Q, c' \in C, w' \in \Sigma^*: (q, c, w) \vdash_{\mathcal{M}}^* (q', c', w') \vdash_{\mathcal{M}} (\bar{q}, \bar{c}, \bar{w}) \\ \iff \exists c' \in C, w' \in \Sigma^*, (q', v, p, r, \bar{q}) \in T: \\ (q, c, w) \vdash_{\mathcal{M}}^* (q', c', w') \wedge w' = v\bar{w} \wedge c' \in p \wedge \bar{c} \in r(c') \\ \iff \exists w' \in \Sigma^*, ((q', c'), v, (\bar{q}, \bar{c})) \in T': (q, c, w) \vdash_{\mathcal{M}}^* (q', c', w') \wedge w' = v\bar{w} \\ \hspace{15em} \text{(by construction of } \mathcal{M}') \\ \iff \exists w' \in \Sigma^*, ((q', c'), v, (\bar{q}, \bar{c})) \in T': ((q, c), w) \vdash_{\mathcal{M}'}^* ((q', c'), w') \wedge w' = v\bar{w} \text{ (by (4))} \\ \iff \exists q' \in Q, c' \in C, w' \in \Sigma^*: ((q, c), w) \vdash_{\mathcal{M}'}^* ((q', c'), w') \vdash_{\mathcal{M}'} ((\bar{q}, \bar{c}), \bar{w}) \\ \iff ((q, c), w) \vdash_{\mathcal{M}'}^* ((\bar{q}, \bar{c}), \bar{w}) \end{aligned}$$

For the languages we then derive

$$\begin{aligned} L(\mathcal{M}) &= \{w \in \Sigma^* \mid R_{\mathcal{M}}(w) \neq \emptyset\} \\ &= \{w \in \Sigma^* \mid \exists q \in Q_i, q' \in Q_f, c' \in C: (q, c_i, w) \vdash_{\mathcal{M}}^* (q', c', \varepsilon)\} \quad \text{(by (1))} \\ &= \{w \in \Sigma^* \mid \exists q \in Q_i, q' \in Q_f, c' \in C: ((q, c_i), w) \vdash_{\mathcal{M}'}^* ((q', c'), \varepsilon)\} \quad \text{(by (4))} \\ &= \{w \in \Sigma^* \mid R_{\mathcal{M}'}(w) \neq \emptyset\} \\ &= L(\mathcal{M}'). \end{aligned} \quad \blacksquare$$

4 Approximation of automata with storage

An approximation strategy maps an nd storage type to another nd storage type. It is specified in terms of storage configurations and naturally extended to predicates and instructions.

Definition 4.1. An *approximation strategy* is a partial function $A: C \dashrightarrow C'$ where C and C' are arbitrary sets. \square

Definition 4.2. Let $S = (C, P, R, c_i)$ be an nd storage type and $A: C \dashrightarrow C'$ be an approximation strategy. The *approximation of S with respect to A* is the nd storage type $S_{\approx A} = (C', P_{\approx A}, R_{\approx A}, A(c_i))$ where $P_{\approx A} = \{p_{\approx A} \mid p \in P\}$ with $p_{\approx A} = A(p)$ for every $p \in P$, and $R_{\approx A} = \{r_{\approx A} \mid r \in R\}$ with $r_{\approx A} = A^{-1}; r; A$ for every $r \in R$. \square

Let $S = (C, P, F, c_i)$ be a finitely nd storage type. We call an approximation strategy $A: C \dashrightarrow C'$ *S -proper* if $S_{\approx A}$ is finitely non-deterministic.

Example 4.3. Consider the approximation strategy $A_o: \mathbb{N} \rightarrow \{o\} \cup \{2n \mid n \in \mathbb{N}\}$ that assigns to every odd number the value o and to every even number the number itself. Then A_o is *not* Count-proper since $\text{inc}_{\approx A_o}(o) = \text{dec}_{\approx A_o}(o) = \{2n \mid n \in \mathbb{N}\}$ is not finite. Note that $\text{Count}_{\approx A_o} = \text{Count}'$ (cf. Ex. 3.4).

On the other hand, consider the approximation strategy $A_{eo}: \mathbb{N} \rightarrow \{e, o\}$ that returns o for every odd number and e otherwise. This approximation strategy is Count-proper since $\text{inc}_{\approx A_{eo}}(e) = \text{dec}_{\approx A_{eo}}(e) = \{o\}$ and $\text{inc}_{\approx A_{eo}}(o) = \text{dec}_{\approx A_{eo}}(o) = \{e\}$ are finite. \square

Example 4.4. Consider the $(\text{Count}, \{a, b\})$ -automaton $\mathcal{M} = ([3], T, \{1\}, \{3\})$ and its approximation $\mathcal{M}_{\approx A_{eo}} = ([3], T', \{1\}, \{3\})$ with

$$\begin{array}{ll} T: \tau_1 = (1, a, \text{true}, \text{inc}, 1) & T': \tau'_1 = (1, a, \text{true}', \text{toggle}, 1) \\ \tau_2 = (1, b, \text{true}, \text{dec}, 2) & \tau'_2 = (1, b, \text{true}', \text{toggle}, 2) \\ \tau_3 = (2, b, \text{true}, \text{dec}, 2) & \tau'_3 = (2, b, \text{true}', \text{toggle}, 2) \\ \tau_4 = (2, \varepsilon, \text{iszero}, \text{inc}, 3) & \tau'_4 = (2, \varepsilon, \text{iseven}, \text{toggle}, 3) \end{array}$$

where $\text{true}' = \text{true}_{\approx A_{eo}} = \{e, o\}$ and $\text{iseven} = \text{iszero}_{\approx A_{eo}} = \{e\}$ are the predicates of $\text{Count}_{\approx A_{eo}}$, and $\text{toggle} = \text{inc}_{\approx A_{eo}} = \text{dec}_{\approx A_{eo}} = \{(e, o), (o, e)\}$ is the instruction of $\text{Count}_{\approx A_{eo}}$. The word $aabb \in \{a, b\}^*$ can be recognised by both automata:

$$\begin{array}{l} (1, 0, aabb) \vdash_{\tau_1} (1, 1, abb) \vdash_{\tau_1} (1, 2, bb) \vdash_{\tau_2} (2, 1, b) \vdash_{\tau_3} (2, 0, \varepsilon) \vdash_{\tau_4} (3, 1, \varepsilon) \\ (1, e, aabb) \vdash_{\tau'_1} (1, o, abb) \vdash_{\tau'_1} (1, e, bb) \vdash_{\tau'_2} (2, o, b) \vdash_{\tau'_3} (2, e, \varepsilon) \vdash_{\tau'_4} (3, o, \varepsilon). \end{array}$$

On the other hand, the word bb can be recognised by $\mathcal{M}_{\approx A_{eo}}$ but not by \mathcal{M} :

$$(1, e, bb) \vdash_{\tau'_2} (2, o, b) \vdash_{\tau'_3} (2, e, \varepsilon) \vdash_{\tau'_4} (3, o, \varepsilon). \quad \square$$

Definition 4.5. Let $\mathcal{M} = (Q, T, Q_i, Q_f)$ be an (S, Σ) -automaton and A be an S -proper approximation strategy. The *approximation of \mathcal{M} with respect to A* is the $(S_{\approx A}, \Sigma)$ -automaton $\mathcal{M}_{\approx A} = (Q, T_{\approx A}, Q_i, Q_f)$ where $T_{\approx A} = \{\tau_{\approx A} \mid \tau \in T\}$ with $\tau_{\approx A} = (q, v, p_{\approx A}, r_{\approx A}, q')$ for every $\tau = (q, v, p, r, q') \in T$. \square

Observation 4.6. Let \mathcal{M} be an (S, Σ) -automaton with $S = (C, P, R, c_i)$, and $A_1: C \dashrightarrow \bar{C}$ and $A_2: \bar{C} \dashrightarrow C'$ be approximation strategies. If A_1 is S -proper and A_2 is $S_{\approx A_1}$ -proper, then $(\mathcal{M}_{\approx A_1})_{\approx A_2} = \mathcal{M}_{\approx A_1; A_2}$. \blacksquare

Definition 4.7. Let $A_1: C \dashrightarrow \bar{C}$ and $A_2: \bar{C} \dashrightarrow C'$ be approximation strategies. We call A_1 *finer than* A_2 , denoted by $A_1 \preceq A_2$, if there is a total approximation strategy A with $A_1; A = A_2$. We call A_1 *less partial than* A_2 , denoted by $A_1 \sqsubseteq A_2$, if there is an injective approximation strategy A with $A_1; A = A_2$. \square

Note that if $A_1 \preceq A_2$, we also know that A_1 and A_2 are *equally partial* (i.e. $A_1 \sqsubseteq A_2$ and A_2 is less partial than A_1). Similarly, if A_1 is less partial than A_2 , we know that A_1 and A_2 are *equally fine* (i.e. A_1 is finer than A_2 and A_2 is finer than A_1). Consequently, \preceq and \sqsubseteq are partial orders but not total orders.

4.1 Superset approximations

In this section we will show that total approximation strategies (i.e. total functions) lead to superset approximations.

Theorem 4.8. Let \mathcal{M} be an (S, Σ) -automaton and A be an S -proper total approximation strategy. Then $L(\mathcal{M}_{\approx A}) \supseteq L(\mathcal{M})$.

Proof. Let $\mathcal{M} = (Q, T, Q_i, Q_f)$ and $S = (C, P, R, c_i)$. We extend $(\cdot)_{\approx A}$ from transitions to sequences of transitions by point-wise application. We show for every $\theta \in T^*$ by induction on the length of θ that

$$\begin{aligned} \forall q, q' \in Q, c, c' \in C, w, w' \in \Sigma^*: \\ (q, c, w) \vdash_\theta (q', c', w') \implies (q, A(c), w) \vdash_{\theta_{\approx A}} (q', A(c'), w') \end{aligned} \quad (5)$$

holds. The induction base is given by the sequence of length 0. For the induction step we assume that (5) holds for all sequences of length n . Let $q, q' \in Q$, $c, c' \in C$, $w, w' \in \Sigma^*$, $\theta \in T^n$, and $\tau \in T$ such that $(q, c, w) \vdash_{\theta\tau} (q', c', w')$. Then there are $u \in \Sigma^*$, $v \in \Sigma \cup \{\varepsilon\}$, and a configuration $(\bar{q}, \bar{c}, vw') \in Q \times C \times \Sigma^*$ such that $w = uvw'$ and $(q, c, uvw') \vdash_\theta (\bar{q}, \bar{c}, vw') \vdash_\tau (q', c', w')$. Hence τ has the form (\bar{q}, v, p, r, q') for some $p \in P$ and $r \in R$ with $\bar{c} \in p$ and $(\bar{c}, c') \in r$. By (5) we have $(q, A(c), uvw') \vdash_{\theta_{\approx A}} (\bar{q}, A(\bar{c}), vw')$ and by Def. 4.5 we know that $\tau_{\approx A} = (\bar{q}, v, p_{\approx A}, r_{\approx A}, q') \in T_{\approx A}$. Note that $A(c') \in C'$ since A is a total function. Now from Def. 4.2 we immediately obtain that $A(\bar{c}) \in p_{\approx A}$ and $(A(\bar{c}), A(c')) \in r_{\approx A}$. Since the states and the read symbol (or ε) are taken over in $\tau_{\approx A}$, we therefore know that $(\bar{q}, A(\bar{c}), vw') \vdash_{\tau_{\approx A}} (q', A(c'), w')$.

The claim then follows from (5) and the definition of $\mathcal{M}_{\approx A}$. \blacksquare

Example 4.9. Recall \mathcal{M} and $\mathcal{M}_{\approx A_{eo}}$ from Ex. 4.4. Their recognised languages are $L(\mathcal{M}) = \{a^n b^n \mid n \in \mathbb{N}_+\}$ and $L(\mathcal{M}_{\approx A_{eo}}) = \{a^m b^n \mid m \in \mathbb{N}, n \in \mathbb{N}_+, m \equiv n \pmod{2}\}$. Thus $L(\mathcal{M}_{\approx A_{eo}})$ is a superset of $L(\mathcal{M})$. \square

Corollary 4.10. Let \mathcal{M} be an (S, Σ) -automaton, and A_1 and A_2 be S -proper approximation strategies. If A_1 is finer than A_2 , then $L(\mathcal{M}_{\approx A_1}) \subseteq L(\mathcal{M}_{\approx A_2})$.

Proof. Since A_1 is finer than A_2 , we know that there is a total approximation strategy A such that $A_1; A = A_2$. We obtain

$$L(\mathcal{M}_{\approx A_1}) \stackrel{(\text{Thm. 4.8})}{\subseteq} L((\mathcal{M}_{\approx A_1})_{\approx A}) \stackrel{(\text{Obs. 4.6})}{=} L(\mathcal{M}_{\approx A_1}; A) \stackrel{(\text{Def. 4.7})}{=} L(\mathcal{M}_{\approx A_2}). \quad \blacksquare$$

The following example shows four approximation strategies that occur in the literature. The first three approximation strategies approximate a context-free language by a recognisable language (taken from Nederhof [Ned00b, Sec. 7]). The forth approximation strategy approximates a context-free language by another context-free language. It is easy to see that the shown approximation strategies are total and thus lead to superset approximations.

Example 4.11. Let Γ be a finite set and $k \in \mathbb{N}_+$.

- (i) Evans [Eva97] proposed to map each pushdown to its top-most element. The same result is achieved by dropping condition 7 and 8 from Baker [Bak81]. This idea is expressed by the approximation strategy $A_{\text{top}}: \Gamma^* \rightarrow \Gamma \cup \{@\}$ with

$$A_{\text{top}}(\varepsilon) = @ \quad \text{and} \quad A_{\text{top}}(\gamma w) = \gamma \quad \text{for every } w \in \Gamma^* \text{ and } \gamma \in \Gamma$$

where $@$ is a new symbol that is not in Γ .

- (ii) Bermudez and Schimpf [BS90] proposed to map each pushdown to its top-most k elements. The following approximation strategy implements this idea:

$$A_{\text{top},k}: \Gamma^+ \rightarrow \{w \in \Gamma^+ \mid |w| \leq k\}$$

$$A_{\text{top},k}(w) = \begin{cases} w & \text{if } |w| \leq k \\ u & \text{if } w \text{ is of the form } uv \text{ for some } u \in \Gamma^k \text{ and } v \in \Gamma^+. \end{cases}$$

- (iii) Pereira and Wright [PW91] proposed to map each pushdown to one where no pushdown symbol occurs more than once. To achieve a unique approximation of each pushdown, we provide an approximation strategy that retains exactly the top-most (actually left-most) occurrence of each symbol: Consider

$$A_{\text{uniq}}: \Gamma^+ \rightarrow \text{Seq}_{\text{nr}}(\Gamma)$$

$$A_{\text{uniq}}(w) = \begin{cases} A_{\text{uniq}}(u\gamma w'\gamma v) & \text{if } w \text{ is of form } u\gamma w'\gamma v \text{ with } \gamma \in \Gamma \\ w & \text{otherwise} \end{cases}$$

where $\text{Seq}_{\text{nr}}(\Gamma)$ denotes the set of all sequences over Γ without repetition.

- (iv) In their coarse-to-fine parsing approach for context-free grammars (short: CFG), Charniak et al. [CPV⁺06] propose, given an equivalence relation \equiv on the set of non-terminals N of some CFG G , to construct a new CFG G' whose non-terminals are the equivalence classes of \equiv .¹ Let Σ be the terminal alphabet of G . Say that

¹Charniak et al. [CPV⁺06] actually considered probabilistic CFGs, but for the sake of simplicity we leave out the probabilities in this example.

$g: N \rightarrow N/\equiv$ is the function that assigns for a nonterminal of G its corresponding equivalence class; and let $g': (N \cup \Sigma)^* \rightarrow ((N/\equiv) \cup \Sigma)^*$ be an extension of $g \cup \{(\sigma, \sigma) \mid \sigma \in \Sigma\}$. Then g' is $\text{PD}_{N \cup \Sigma}$ -proper and $L(\mathcal{M}_{\approx g'}) = L(G')$ where \mathcal{M} is the $(\text{PD}_{N \cup \Sigma}, \Sigma)$ -automaton obtained from G by the usual construction [HU79, Thm. 5.3]. \square

4.2 Subset approximations

In this section we will show that injective approximation strategies lead to a subset approximation, this is proved by a variation of the proof of Thm. 4.8.

Theorem 4.12. Let \mathcal{M} be an (S, Σ) -automaton and A be an S -proper injective approximation strategy. Then $L(\mathcal{M}_{\approx A}) \subseteq L(\mathcal{M})$.

Proof. Let $\mathcal{M} = (Q, T, Q_i, Q_f)$ and $S = (C, P, R, c_i)$. We show for every $\theta \in T^*$ by induction on the length of θ that

$$\begin{aligned} \forall q, q' \in Q, c, c' \in \text{img}(A), w, w' \in \Sigma^*: \\ (q, c, w) \vdash_{\theta \approx A} (q', c', w') \implies (q, A^{-1}(c), w) \vdash_{\theta} (q', A^{-1}(c'), w') \end{aligned} \quad (6)$$

holds. The induction base is given by the sequence of length 0. For the induction step we assume that (6) holds for all sequences of length n . Let $q, q' \in Q$, $c, c' \in \text{img}(A)$, $w, w' \in \Sigma^*$, $\theta \in T^n$, and $\tau \in T$ such that $(q, c, w) \vdash_{\theta \approx A \tau \approx A} (q', c', w')$. Then there are $u \in \Sigma^*$, $v \in \Sigma \cup \{\varepsilon\}$, $\bar{q} \in Q$, and $\bar{c} \in \text{img}(A)$ such that $w = uvw'$ and $(q, c, uvw') \vdash_{\theta \approx A} (\bar{q}, \bar{c}, vw') \vdash_{\tau \approx A} (q', c', w')$. Hence $\tau \approx A$ has the form $(\bar{q}, v, p \approx A, r \approx A, q')$ for some $p \in P$ and $r \in R$ such that $\bar{c} \in p \approx A$ and $(\bar{c}, c') \in r \approx A$. By (6) we have $(q, A^{-1}(c), uvw') \vdash_{\theta} (\bar{q}, A^{-1}(\bar{c}), vw')$ and by Def. 4.5 we know that $\tau \approx A = (\bar{q}, v, p, f, q') \in T$. Note that $A^{-1}(c')$ is uniquely defined since $c \in \text{img}(A)$ and A is injective. Now from Def. 4.2 we immediately obtain that $A^{-1}(\bar{c}) \in p$ and $(A^{-1}(\bar{c}), A^{-1}(c')) \in r$. Since the states and the read symbol (or ε) are taken over in $\tau \approx A$, we therefore know that $(\bar{q}, A^{-1}(\bar{c}), vw') \vdash_{\tau \approx A} (q', A^{-1}(c'), w')$.

The claim then follows from (6) and the definition of $\mathcal{M}_{\approx A}$. \blacksquare

Corollary 4.13. Let \mathcal{M} be an (S, Σ) -automaton and A_1 and A_2 be S -proper approximation strategies. If A_1 is less partial than A_2 , then $L(\mathcal{M}_{\approx A_1}) \supseteq L(\mathcal{M}_{\approx A_2})$.

Proof. Since A_1 is less partial than A_2 , we know that there is a injective approximation strategy A such that $A_1; A = A_2$. We obtain

$$L(\mathcal{M}_{\approx A_1}) \stackrel{(\text{Thm. 4.12})}{\supseteq} L((\mathcal{M}_{\approx A_1})_{\approx A}) \stackrel{(\text{Obs. 4.6})}{=} L(\mathcal{M}_{\approx A_1; A}) \stackrel{(\text{Def. 4.7})}{=} L(\mathcal{M}_{\approx A_2}). \quad \blacksquare$$

The following example approximates a context-free language with a recognisable language (taken from Nederhof [Ned00b, Sec. 7]). It is easy to see that the shown approximation strategy is injective and thus leads to subset approximations.

Example 4.14. Let Γ be a finite set and $k \in \mathbb{N}_+$. Krauwer and des Tombe [KdT81], Pulman [Pul86], and Langendoen and Langsam [LL87] proposed to disallow stacks of height greater than k . This can be achieved by a partial identity:

$$A_{\text{bd},k}: \Gamma^+ \dashrightarrow \{w \in \Gamma \mid |w| \leq k\} \quad A_{\text{bd},k}(w) = \begin{cases} w & \text{if } |w| \leq k \\ \text{undefined} & \text{otherwise.} \end{cases} \quad \square$$

4.3 Approximation of weighted automata with storage

Definition 4.15. Let Σ be a finite set, S be a finitely nd storage type, and K be a complete semiring. An (S, Σ, K) -automaton is a tuple $\mathcal{M} = (Q, T, Q_i, Q_f, \delta)$ where (Q, T, Q_i, Q_f) is an (S, Σ) -automaton and $\delta: T \rightarrow K$ (*transition weights*). We sometimes denote (Q, T, Q_i, Q_f) by \mathcal{M}_{uw} . \square

Consider the (S, Σ, K) -automaton $\mathcal{M} = (Q, T, Q_i, Q_f, \delta)$. The *configurations* of \mathcal{M} , the *run relation* of \mathcal{M} , and the *set of runs* of \mathcal{M} on w for every $w \in \Sigma^*$ are the same as for \mathcal{M}_{uw} . The *weight* of θ in \mathcal{M} is the value $\text{wt}_{\mathcal{M}}(\theta) = \delta(\tau_1) \cdot \dots \cdot \delta(\tau_k)$ for every $\theta = \tau_1 \cdots \tau_k$ with $\tau_1, \dots, \tau_k \in T$. The *weighted language induced by \mathcal{M}* is the function $\llbracket \mathcal{M} \rrbracket: \Sigma^* \rightarrow K$ where for every $w \in \Sigma^*$:

$$\llbracket \mathcal{M} \rrbracket(w) = \sum_{\theta \in R_{\mathcal{M}}(w)} \text{wt}_{\mathcal{M}}(\theta). \quad (7)$$

Let S be a finitely nd storage type, Σ be a finite set, K be a semiring, and $r: \Sigma^* \rightarrow K$. We call r (S, Σ, K) -recognisable if there is an (S, Σ, K) -automaton \mathcal{M} with $r = \llbracket \mathcal{M} \rrbracket$.

We extend Prop. 3.6 to the weighted case, using $(\cdot)_{\text{det}}$ as defined in Prop. 3.6.

Proposition 4.16. The classes of (S, Σ, K) -recognisable and of $(S_{\text{det}}, \Sigma, K)$ -recognisable languages are the same for every finitely nd storage type S , finite set Σ , and semiring K .

Proof. Let $\mathcal{M} = (Q, T, Q_i, Q_f, \delta)$ and $\mathcal{M}' = (Q', T', Q'_i, Q'_f, \delta')$ be an (S, Σ, K) -automaton and an $(S_{\text{det}}, \Sigma, K)$ -automaton, respectively. We call \mathcal{M} and \mathcal{M}' *related* if \mathcal{M}_{uw} and \mathcal{M}'_{uw} are related, and $\delta'(\tau_{\text{det}}) = \delta(\tau)$ for every $\tau \in T$. Note that $(\cdot)_{\text{det}}$ is a bijection between T and T_{det} . Clearly, for every (S, Σ, K) -automaton \mathcal{M} there is an $(S_{\text{det}}, \Sigma, K)$ -automaton \mathcal{M}' such that \mathcal{M} and \mathcal{M}' are related and vice versa. It remains to be shown that $\llbracket \mathcal{M} \rrbracket = \llbracket \mathcal{M}' \rrbracket$. For every $w \in \Sigma^*$, we derive

$$\begin{aligned} \llbracket \mathcal{M} \rrbracket(w) &= \sum_{\theta \in R_{\mathcal{M}}(w)} \text{wt}_{\mathcal{M}}(\theta) && \text{(by (7))} \\ &= \sum_{\theta \in R_{\mathcal{M}}(w)} \text{wt}_{\mathcal{M}'}(\theta_{\text{det}}) && \text{(by Def. of } \delta_{\text{det}}) \\ &= \sum_{\theta' \in R_{\mathcal{M}'}(w)} \text{wt}_{\mathcal{M}'}(\theta') && \text{(by } (\cdot)_{\text{det}} \text{ being a bijection and (3))} \\ &= \llbracket \mathcal{M}' \rrbracket(w) && \text{(by (7))} \quad \blacksquare \end{aligned}$$

Definition 4.17. Let $\mathcal{M} = (Q, T, Q_i, Q_f, \delta)$ be an (S, Σ, K) -automaton, and A an S -proper approximation strategy. The *approximation of \mathcal{M} with respect to A* is the $(S_{\approx A}, \Sigma, K)$ -automaton $\mathcal{M}_{\approx A} = (Q, T_{\approx A}, Q_i, Q_f, \delta_{\approx A})$ where $S_{\approx A}$ and $T_{\approx A}$ are defined as in Def. 4.5, and $\delta_{\approx A}(\tau') = \sum_{\tau \in T: \tau_{\approx A} = \tau'} \delta(\tau)$ for every $\tau' \in T_{\approx A}$. \square

Lemma 4.18. Let \mathcal{M} be an (S, Σ, K) -automaton, A be an S -proper approximation strategy, \leq be a partial order on K , and K be positively \leq -ordered.

- (i) If A is total, then $\text{wt}_{\mathcal{M}_{\approx A}}(\theta') \geq \sum_{\theta \in R_{\mathcal{M}}: \theta \approx A = \theta'} \text{wt}_{\mathcal{M}}(\theta)$ for every $\theta' \in R_{\mathcal{M}_{\approx A}}$.
- (ii) If A is injective, then $\text{wt}_{\mathcal{M}_{\approx A}}(\theta') \leq \sum_{\theta \in R_{\mathcal{M}}: \theta \approx A = \theta'} \text{wt}_{\mathcal{M}}(\theta)$ for every $\theta' \in R_{\mathcal{M}_{\approx A}}$.

Proof. We prove (i) by induction on the length of θ' . For $\theta' = \varepsilon$, we derive

$$\text{wt}_{\mathcal{M}_{\approx A}}(\varepsilon) = 1 \geq 1 = \text{wt}_{\mathcal{M}}(\varepsilon) = \sum_{\theta \in R_{\mathcal{M}}: \theta \approx A = \varepsilon} \text{wt}_{\mathcal{M}}(\theta)$$

For $\theta'\tau' \in R_{\mathcal{M}_{\approx A}}$ with $\tau' \in T_{\approx A}$, we derive

$$\begin{aligned} \text{wt}_{\mathcal{M}_{\approx A}}(\theta'\tau') &= \text{wt}_{\mathcal{M}_{\approx A}}(\theta') \cdot \delta_{\approx A}(\tau') \\ &\geq \left(\sum_{\theta \in R_{\mathcal{M}}: \theta \approx A = \theta'} \text{wt}_{\mathcal{M}}(\theta) \right) \cdot \delta_{\approx A}(\tau') && \text{(by induction hypothesis)} \\ &= \left(\sum_{\theta \in R_{\mathcal{M}}: \theta \approx A = \theta'} \text{wt}_{\mathcal{M}}(\theta) \right) \cdot \left(\sum_{\tau \in T: \tau \approx A = \tau'} \delta(\tau) \right) && \text{(by Def. 4.17)} \\ &= \sum_{\theta \in R_{\mathcal{M}}, \tau \in T: (\theta \approx A = \theta') \wedge (\tau \approx A = \tau')} \text{wt}_{\mathcal{M}}(\theta) \cdot \delta(\tau) && \text{(by distributivity of } K) \\ &\geq \sum_{\theta \in R_{\mathcal{M}}: (\tau \in T) \wedge ((\theta\tau) \approx A = \theta'\tau')} \text{wt}_{\mathcal{M}}(\theta) \cdot \delta(\tau) && \text{(by (5) and } K \text{ being positively } \leq\text{-ordered)} \\ &= \sum_{\bar{\theta} \in R_{\mathcal{M}}: (\bar{\theta} \approx A = \theta'\tau')} \text{wt}_{\mathcal{M}}(\bar{\theta}) && \text{(by Def. 4.17)} \end{aligned}$$

The proof for (ii) can be obtained from the proof for (i) by replacing every occurrence of \geq by \leq and the use of (5) by (6). \blacksquare

Theorem 4.19. Let \mathcal{M} be an (S, Σ, K) -automaton, A be an S -proper approximation strategy, and \leq be a partial order on K , and K be positively \leq -ordered.

- (i) If A is total, then $\llbracket \mathcal{M}_{\approx A} \rrbracket(w) \geq \llbracket \mathcal{M} \rrbracket(w)$ for every $w \in \Sigma^*$.
- (ii) If A is injective, then $\llbracket \mathcal{M}_{\approx A} \rrbracket(w) \leq \llbracket \mathcal{M} \rrbracket(w)$ for every $w \in \Sigma^*$.

Proof. **ad (i):** For every $w \in \Sigma^*$, we derive

$$\begin{aligned} \llbracket \mathcal{M}_{\approx A} \rrbracket(w) &= \sum_{\theta' \in R_{\mathcal{M}_{\approx A}}(w)} \text{wt}_{\mathcal{M}_{\approx A}}(\theta') && \text{(by (7))} \\ &\geq \sum_{\theta' \in R_{\mathcal{M}_{\approx A}}(w)} \sum_{\theta \in R_{\mathcal{M}}: \theta \approx A = \theta'} \text{wt}_{\mathcal{M}}(\theta) && \text{(by Lem. 4.18 (i))} \\ &= \sum_{\theta' \in R_{\mathcal{M}_{\approx A}}(w)} \sum_{\theta \in R_{\mathcal{M}}(w): \theta \approx A = \theta'} \text{wt}_{\mathcal{M}}(\theta) && \text{(by Def. 4.5)} \\ &\geq \sum_{\theta \in R_{\mathcal{M}}(w)} \text{wt}_{\mathcal{M}}(\theta) && \text{(by } (\dagger)) \\ &= \llbracket \mathcal{M} \rrbracket(w) && \text{(by (7))} \end{aligned}$$

For (\dagger) , we argue that, since A is total, we know that for every $\theta \in R_{\mathcal{M}}(w)$, the addend $\text{wt}_{\mathcal{M}}(\theta)$ occurs at least once on the left side of the inequality and exactly once on the right side. Hence “ \geq ” is justified.

ad (ii): For every $w \in \Sigma^*$, we derive

$$\begin{aligned}
\llbracket \mathcal{M}_{\approx A} \rrbracket(w) &= \sum_{\theta' \in R_{\mathcal{M}_{\approx A}}(w)} \text{wt}_{\mathcal{M}_{\approx A}}(\theta') && \text{(by (7))} \\
&\leq \sum_{\theta' \in R_{\mathcal{M}_{\approx A}}(w)} \sum_{\theta \in R_{\mathcal{M}} : \theta \approx_A \theta'} \text{wt}_{\mathcal{M}}(\theta) && \text{(by Lem. 4.18 (ii))} \\
&= \sum_{\theta' \in R_{\mathcal{M}_{\approx A}}(w)} \sum_{\theta \in R_{\mathcal{M}}(w) : \theta \approx_A \theta'} \text{wt}_{\mathcal{M}}(\theta) && \text{(by Def. 4.5)} \\
&\leq \sum_{\theta \in R_{\mathcal{M}}(w)} \text{wt}_{\mathcal{M}}(\theta) && \text{(by (\dagger))} \\
&= \llbracket \mathcal{M} \rrbracket(w) && \text{(by (7))}
\end{aligned}$$

For (\dagger), we argue that since A is injective, we know that for every $\theta \in R_{\mathcal{M}}(w)$, the addend $\text{wt}_{\mathcal{M}}(\theta)$ occurs at most once on the left side of the inequality and exactly once on the right side. Hence “ \leq ” is justified. \blacksquare

5 Approximation of multiple context-free languages

In Exs. 4.11 and 4.14 we recall some approximation strategies for pushdown automata from the literature. Due to the equivalence of pushdown automata and context-free grammars [HU79, Thms. 5.3 and 5.4], those approximation strategies can be used for the approximation of context-free languages. The framework presented in this paper together with the automata characterisation of multiple context-free languages [Den16, Thm. 18] allows an automata-theoretic view on the approximation of multiple context-free languages. The automata characterisation uses an excursion-restricted form of automata with tree-stack storage.²

Definition 5.1. Let Γ be a finite set. The *tree-stack storage over Γ* is the deterministic nd storage type $\text{TreeStack}_{\Gamma} = (\text{TS}_{\Gamma}, P_{\text{ts}}, R_{\text{ts}}, c_{\text{i,ts}})$ where

- TS_{Γ} is the set of tuples $\langle \xi, \rho \rangle$ where $\xi : \mathbb{N}_+^* \dashrightarrow \Gamma \cup \{\text{@}\}$, $\text{dom}(\xi)$ is finite and prefix-closed,³ $\rho \in \text{dom}(\xi)$, and $\xi(\rho') = \text{@}$ iff $\rho' = \varepsilon$;
- $c_{\text{i,ts}} = \langle \{(\varepsilon, \text{@})\}, \varepsilon \rangle$;
- $P_{\text{ts}} = \{\text{TS}_{\Gamma}, \text{bottom}\} \cup \{\text{top}_{\gamma} \mid \gamma \in \Gamma\}$ with $\text{bottom} = \{\langle \xi, \rho \rangle \in \text{TS}_{\Gamma} \mid \rho = \varepsilon\}$ and $\text{top}_{\gamma} = \{\langle \xi, \rho \rangle \in \text{TS}_{\Gamma} \mid \xi(\rho) = \gamma\}$ for every $\gamma \in \Gamma$; and
- $R_{\text{ts}} = \{\text{down}\} \cup \{\text{up}_n, \text{push}_{n,\gamma} \mid n \in \mathbb{N}, \gamma \in \Gamma\}$ with
 - $\text{down} = \{(\langle \xi, \rho n \rangle, \langle \xi, \rho \rangle) \mid \langle \xi, \rho \rangle \in \text{TS}_{\Gamma}, n \in \mathbb{N}_+, \rho n \in \text{dom}(\xi)\}$,
 - $\text{up}_n = \{(\langle \xi, \rho \rangle, \langle \xi, \rho n \rangle) \mid \langle \xi, \rho \rangle \in \text{TS}_{\Gamma}, \rho n \in \text{dom}(\xi)\}$, and
 - $\text{push}_{n,\gamma} = \{(\langle \xi, \rho \rangle, \langle \xi \cup \{(\rho n, \gamma)\}, \rho n \rangle) \mid \langle \xi, \rho \rangle \in \text{TS}_{\Gamma}, \rho n \notin \text{dom}(\xi)\}$
for every $n \in \mathbb{N}_+$ and $\gamma \in \Gamma$. \square

²See Denkinger [Den16] for details on the automata characterisation.

³A set $D \subseteq \mathbb{N}_+^*$ is *prefix closed* if for each $w \in D$, every prefix of w is also in D .

Example 5.2. Let $\Sigma = \{a, b, c\}$. Consider the $(\text{TreeStack}_{\{*, \#\}}, \Sigma)$ -automaton $\mathcal{M} = ([4], T, \{1\}, \{4\})$ where T contains the transitions

$$\begin{array}{lll} (1, a, \text{TS}_\Gamma, \text{push}_{1,*}, 1) & (2, b, \text{top}_*, \text{down}, 2) & (3, c, \text{top}_*, \text{up}_1, 3) \\ (1, \varepsilon, \text{TS}_\Gamma, \text{push}_{1,\#}, 2) & (2, \varepsilon, \text{bottom}, \text{up}_1, 3) & (3, \varepsilon, \text{top}_\#, \text{down}, 4) \\ (2, \varepsilon, \text{top}_\#, \text{down}, 2) \end{array}$$

and note that $L(\mathcal{M}) = \{a^n b^n c^n \mid n \in \mathbb{N}\}$. \square

We present two approximation strategies for multiple context-free languages from the literature, using the framework of this paper.

Example 5.3. Let Γ be a finite set.

- (i) Van Cranenburgh [vC12, Sec. 4] observed that the idea of Ex. 4.11 (iv) also applies to multiple context-free grammars (short: MCFG). The idea can be applied to tree-stack automata similarly to the way it was applied to pushdown automata in Ex. 4.11 (iv). The resulting nd storage type is still a tree-stack storage.
- (ii) Burden and Ljunglöf [BL05, Sec. 4] and van Cranenburgh [vC12, Sec. 4] proposed to split each production of a given MCFG into multiple productions, each of fan-out 1. Since the resulting grammar is of fan-out 1, it produces a context-free language and can be recognised by a pushdown automaton. The corresponding approximation strategy in our framework is

$$A_{\text{cf}, \Gamma}: \text{TS}_\Gamma \rightarrow \Gamma^*, \quad A_{\text{cf}, \Gamma}((\xi, n_1 \cdots n_k)) = \xi(n_1 \cdots n_k) \cdots \xi(n_1 n_2) \xi(n_1)$$

for every $(\xi, n_1 \cdots n_k) \in \text{TS}_\Gamma$ with $n_1, \dots, n_k \in \mathbb{N}_+$. The resulting nd storage type is pushdown storage. \square

Example 5.4. Let us consider the $(\text{TreeStack}_{\{*, \#\}}, \Sigma)$ -automaton \mathcal{M} from Ex. 5.2. The transitions of the $((\text{TreeStack}_{\{*, \#\}})_{\approx A_{\text{cf}, \Gamma}}, \Sigma)$ -automaton $\mathcal{M}_{\approx A_{\text{cf}, \Gamma}}$ (cf. Ex. 5.3) and the $((\text{TreeStack}_{\{*, \#\}})_{\approx A_{\text{cf}, \Gamma}; A_{\text{top}}}, \Sigma)$ -automaton $\mathcal{M}_{\approx A_{\text{cf}, \Gamma}; A_{\text{top}}}$ (cf. also Ex. 4.11) are shown below.

transitions of $\mathcal{M}_{\approx A_{\text{cf}, \Gamma}}$:	transitions of $\mathcal{M}_{\approx A_{\text{cf}, \Gamma}; A_{\text{top}}}$:
$(1, a, \Gamma^*, \text{push}_*, 1)$	$(1, a, \Gamma_{\text{@}}, \{(\gamma, *) \mid \gamma \in \Gamma_{\text{@}}\}, 1)$
$(1, \varepsilon, \Gamma^*, \text{push}_\#, 2)$	$(1, \varepsilon, \Gamma_{\text{@}}, \{(\gamma, \#) \mid \gamma \in \Gamma_{\text{@}}\}, 2)$
$(2, \varepsilon, \text{top}_\#, \text{pop}, 2)$	$(2, \varepsilon, \{\#\}, \{(\gamma, \gamma') \mid \gamma, \gamma' \in \Gamma_{\text{@}}\}, 2)$
$(2, b, \text{top}_*, \text{pop}, 2)$	$(2, b, \{*\}, \{(\gamma, \gamma') \mid \gamma, \gamma' \in \Gamma_{\text{@}}\}, 2)$
$(2, \varepsilon, \text{bottom}, \text{push}_* \cup \text{push}_\#, 3)$	$(2, \varepsilon, \{\text{@}\}, \{(\gamma, \gamma') \mid \gamma \in \Gamma_{\text{@}}, \gamma' \in \Gamma\}, 3)$
$(3, c, \text{top}_*, \text{push}_* \cup \text{push}_\#, 3)$	$(3, c, \{*\}, \{(\gamma, \gamma') \mid \gamma \in \Gamma_{\text{@}}, \gamma' \in \Gamma\}, 3)$
$(3, \varepsilon, \text{top}_\#, \text{pop}, 4)$	$(3, \varepsilon, \{\#\}, \{(\gamma, \gamma') \mid \gamma, \gamma' \in \Gamma_{\text{@}}\}, 4)$

Note that $\mathcal{M}_{\approx A_{\text{cf}, \Gamma}; A_{\text{top}}}$ has finitely many storage configurations, and thus its language is recognisable by a finite state automaton (by Prop. 3.8). \square

References

- [Bak81] T. P. Baker. Extending lookahead for LR parsers. *Journal of Computer and System Sciences*, 22(2):243–259, 1981. DOI: 10.1016/0022-0000(81)90030-1.
- [BL05] H. Burden and P. Ljunglöf. Parsing linear context-free rewriting systems. In H. Bunt, R. Malouf, and A. Lavie, editors, *Proceedings of the 9th International Workshop on Parsing Technology (IWPT 2005)*, pages 11–17, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL: <http://dl.acm.org/citation.cfm?id=1654494.1654496>.
- [BS90] M. E. Bermudez and K. M. Schimpf. Practical arbitrary lookahead LR parsing. *Journal of Computer and System Sciences*, 41(2):230–250, 1990. DOI: 10.1016/0022-0000(90)90037-1.
- [CPV⁺06] E. Charniak, M. Pozar, T. Vu, M. Johnson, M. Elsner, J. Austerweil, D. Ellis, I. Haxton, C. Hill, R. Shrivaths, and J. Moore. Multilevel coarse-to-fine PCFG parsing. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (NAACL HLT)*. Association for Computational Linguistics, 2006. DOI: 10.3115/1220835.1220857.
- [Den16] T. Denking. An automata characterisation for multiple context-free languages. In S. Brlek and C. Reutenauer, editors, *Developments in Language Theory (DLT 2016)*, pages 138–150. Springer Berlin Heidelberg, 2016. DOI: 10.1007/978-3-662-53132-7_12.
- [DKV09] M. Droste, W. Kuich, and H. Vogler. *Handbook of weighted automata*. Springer, 2009. DOI: 10.1007/978-3-642-01492-5.
- [Eng86] J. Engelfriet. Context-free grammars with storage. Technical Report I86-11, Leiden University, 1986.
- [Eng14] J. Engelfriet. Context-free grammars with storage. *Computing Research Repository*, 2014. arXiv:1408.0683 [cs.FL].
- [EV86] J. Engelfriet and H. Vogler. Pushdown machines for the macro tree transducer. *Theoretical Computer Science*, 42(0):251–368, 1986. DOI: 10.1016/0304-3975(86)90052-6.
- [Eva97] E. G. Evans. Approximating context-free grammars with a finite-state calculus. In *Proceedings of the 8th Conference of the European chapter of the Association for Computational Linguistics (EACL 1997)*. Association for Computational Linguistics, 1997. DOI: 10.3115/979617.979675.
- [Hoa72] C. A. R. Hoare. Proof of correctness of data representations. *Acta Informatica*, 1(4), 1972. DOI: 10.1007/bf00289507.

- [HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1st edition, 1979.
- [HV15] L. Herrmann and H. Vogler. A Chomsky-Schützenberger theorem for weighted automata with storage. In A. Maletti, editor, *Proceedings of the 6th International Conference on Algebraic Informatics (CAI 2015)*, volume 9270, pages 90–102. Springer International Publishing, 2015. DOI: 10.1007/978-3-319-23021-4_11.
- [Joh98] M. Johnson. Finite-state approximation of constraint-based grammars using left-corner grammar transforms. In *Proceedings of the 17th International Conference on Computational Linguistics*. Association for Computational Linguistics, 1998. DOI: 10.3115/980451.980948.
- [KdT81] S. Krauwer and L. des Tombe. Transducers and grammars as theories of language. *Theoretical Linguistics*, 8(1-3), 1981. DOI: 10.1515/thli.1981.8.1-3.173.
- [LL87] D. T. Langendoen and Y. Langsam. On the design of finite transducers for parsing phrase-structure languages. *Mathematics of Language*, pages 191–235, 1987.
- [Ned00a] M.-J. Nederhof. Practical experiments with regular approximation of context-free languages. *Computational Linguistics*, 26(1):17–44, 2000. DOI: 10.1162/089120100561610.
- [Ned00b] M.-J. Nederhof. Regular approximation of CFLs: a grammatical view. In H. Bunt and A. Nijholt, editors, *Advances in Probabilistic and other Parsing Technologies*, pages 221–241. Springer, 2000. DOI: 10.1007/978-94-015-9470-7_12.
- [Pul86] S. G. Pulman. Grammars, parsers, and memory limitations. *Language and Cognitive Processes*, 1(3):197–225, 1986. DOI: 10.1080/01690968608407061.
- [PW91] F. C. N. Pereira and R. N. Wright. Finite-state approximation of phrase structure grammars. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics (ACL’91)*. Association for Computational Linguistics, 1991. DOI: 10.3115/981344.981376.
- [Sco67] D. Scott. Some definitional suggestions for automata theory. *Journal of Computer and System Sciences*, 1(2):187–212, 1967. DOI: 10.1016/S0022-0000(67)80014-X.
- [SMFK91] H. Seki, T. Matsumura, M. Fujii, and T. Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229, 1991. DOI: 10.1016/0304-3975(91)90374-B.

- [vC12] A. van Cranenburgh. Efficient parsing with linear context-free rewriting systems. In W. Daelemans, editor, *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*, pages 460–470, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL: <http://dl.acm.org/citation.cfm?id=2380816.2380873>.
- [VDH16] H. Vogler, M. Droste, and L. Herrmann. A weighted mso logic with storage behaviour and its büchichi-elgot-trakhtenbrot theorem. In A.-H. Dediu, J. Janoušek, C. Martín-Vide, and B. Truthe, editors, *Proceedings of the 10th International Conference on Language and Automata Theory and Applications (LATA 2016)*, pages 127–139. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-30000-9_10.
- [VSWJ87] K. Vijay-Shanker, D. J. Weir, and A. K. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting on Association for Computational Linguistics (ACL 1997)*, pages 104–111, Stroudsburg, PA, USA, 1987. Association for Computational Linguistics. DOI: 10.3115/981175.981190.