# On the relation between the classes of Weighted Automata and Linear Cost Register Automata

S. Hitarth

**Abstract**

Weighted automata (WA) and linear cost register automata (LCRA) are two equivalent models of quantitative automata. LCRAs are deterministic finite-state automata with access to write-only registers that can be updated by combining other registers and constants using operations over a semiring. They give a different perspective for the class of functions computable by WA. For several natural classes of LCRA, we prove equivalence with well-known classes of WA, mostly by bounding the ambiguity of the latter.

Submitted in partial satisfaction of the requirements for the
Degree of Master of Science in Computer Science
**Chennai Mathematical Institute**


Supervisor: Laure Daviaud
Co-supervisor: Andrew Ryzhikov

**Acknowledgements**

It remains a mystery who reads the acknowledgement. Regardless, firstly, I would like to thank Laure and Andrew for their guidance and support. Laure and Andrew also went through the process of proof reading various sections very carefully (painfully?) and made numerous suggestions to improve the style of writing. Laure's enthusiastic personality and encouragement helped curb my pessimism.

I would like to thank Aiswarya, M. Praveen, B. Srivathsan, and M. Srivas for their wonderful courses in automata theory, logic, and verification, and all the people at CMI who taught me various things.

I am grateful to my friends, especially Soumo, and my family for always *trying* to motivate and wake me up! Finally, I thank Mikey, Gopi, and Keanu for their love which I definitely never deserved.

# Contents

# 1 Introduction

Classical automata are finite-state machines that either accept or reject a given word. Weighted automata (WA) are an extension of classical automata that, instead of having a binary behavior, associates a weight with each word. WA were introduced by Shützenberger [1]. A WA is defined over some semiring $\mathbb{S} = (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$, and each transition has an associated *weight*, which is a constant from the semiring. The multiplicative operation $\odot$ of the semiring is used to aggregate the weights along a run, and the additive operation $\oplus$ is used to aggregate values of distinct accepting runs of a word. WA and decidability of their properties have been extensively studied, e.g., in [2]. WA have found applications in digital image processing [3], natural language processing [4], etc.

Cost Register Automata (CRA) is a deterministic finite-state model with a finite set of *write-only*[1] registers that maps each word to a value. This model was introduced in [5] by Alur et al. The registers hold values from some semiring[2], and on each step, the CRA takes a transition and updates the values of the registers using *update expressions*. For each state, we have a *final expression* which is used to compute the output of the CRA. The update and final expression are syntactical expressions defined with the set of registers $\mathcal{X}$, the constants, and the binary operations of the semiring. The class of functions definable using a CRA depends upon the choice of semiring. In our discussion, while comparing WA and CRA, we always assume that they are over the same semiring $\mathbb{S} = (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$, unless otherwise stated.

It was shown in [5] that, in general, CRA are strictly more expressive than WA. For example, CRA over max-plus semiring can compute a function that is exponential in the length of the word, while a function computable by WA over the max-plus semiring is always bounded by a linear function. It is natural to find structural restrictions on the CRA and study the classes of functions definable by them. One such structural restriction for a transition of a CRA is being *copyless*, which means that each register can be used to update at most one register for this transition.

Copyless Cost Register Automata (CCRA) are CRA where every transition is copyless. This class was proposed in [5], and the results related to its expressiveness are discussed in [6]. It was shown in [5] that the class of functions computed by CCRA is a strict subset of functions definable by WA. Alur et al. conjectured in [5] that the equivalence problem could be decidable for CCRA, which was disproved in [7]. A fragment of CCRA, called *bounded alternation copyless* (BAC) CRA, was introduced in [6]. This class is closed under reversal and regular look ahead [6].

Linear Cost Register Automata (LCRA) are CRA in which all the update and final expressions are linear, i.e., the expressions do not allow to multiply

[1] Write-only condition means that the updates cannot depend upon the value held by registers

[2] The general CRA uses the so-called *cost models*, and the class of cost functions definable using a CRA depends upon the grammar, the domain, and the semantics used to evaluate the expressions. In this work, we restrict the domain to be a semiring

two registers, and set registers to a constant. It was shown in [5] that LCRA are equally expressive as WA. LCRA with *resets* additionally allow the update expressions to be of the form $x := c$. This does not increase the expressiveness of LCRA, since for each such update we can use a separate register initialized with a constant, but could help to define more natural classes of functions definable by LCRA.

LCRA give us a different perspective to reason about the functions that are definable using WA. They could also be used to define new natural classes of functions which cannot be defined via WA. In this work, we discuss various natural subclasses of LCRA obtained by further restricting the update expressions and the final expressions. We also discuss the relations between the classes of functions computable by these subclasses of LCRA and the classes of functions computable by various types of WA.

Additive Cost Register Automata (ACRA), defined in [8], are CRA where both the update and final expressions are of the form $x+c$, where $c \in \mathbb{Z}$. . In [8], it was shown that over tropical semirings, ACRA are equivalent to unambiguous automata. A generalization of ACRA, which we call *plusfree* LCRA, over arbitrary semiring has been discussed in [9]. In a plusfree LCRA, the update expressions are of the form $x := y \odot c$, but the final expressions are allowed to be an unrestricted linear expression.

## Outline

The organization of the thesis is as follows:

- In Section 2 we define the LCRA and WA formally and introduce some notation that would be used later in the results.

- In Section 3.1 we prove, for the sake of completeness, that LCRA and WA are equally expressive. We also define various classes of functions computable by WA and LCRA and discuss various relations between these classes.

  We restrict the final expressions of plusfree LCRA to use at most $k$ distinct registers, and the class of functions computable by such LCRA is denoted by $\text{LCRA}(\odot)_k$. We shall prove in Lemma 3.2 that this class is a subset of functions definable using $k$-ambiguous WA, over all semirings. For tropical semirings, these classes coincide [9].

  We impose the copyless restriction on LCRA, and define $\text{LCRA}_k^c$ as the class of functions computable by copyless LCRA with $k$ registers. This class is equal to the class of functions definable by $k$-sequential WA, over all semirings. We prove this result formally in Lemma 3.4.

- In Section 4, we will discuss a specific class of LCRA called *diagonal* LCRA. We show that the class of functions computable by diagonal LCRA is the intersection of finitely sequential and unambiguous functions

# 2 Preliminaries

This section introduces the definitions related to *weighted automata* and *cost register automata*.

## 2.1 Semirings

A semiring $\mathbb{S} = (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$ is a set equipped with two binary operations $\oplus$ and $\odot$, where $(S, \oplus, \mathbb{0})$ is a **commutative monoid** with *identity* $\mathbb{0}$, $(S, \odot, \mathbb{0})$ is a **monoid** with *identity* $\mathbb{1}$, and $\odot$ (left and right) distributes over $\oplus$, and $\mathbb{0}$ is an *absorbing element* for $\odot$ [3].

A semiring is said to be *commutative* when the multiplication ($\odot$) is commutative. A semiring is said to be **idempotent** if $x \oplus x = x$ for all $x \in S$. An important semiring that will be used extensively in our discussion is the *max-plus semiring* $\mathbb{N}_{\max} = (\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$. The max-plus semiring, boolean semiring, etc., are idempotent.

We define a matrix $M$ over $\mathbb{S}$ whose rows and columns are indexed by arbitrary finite sets $A$ and $B$, respectively, as a a function $M : A \times B \to S$, and we write $M \in S^{A \times B}$.

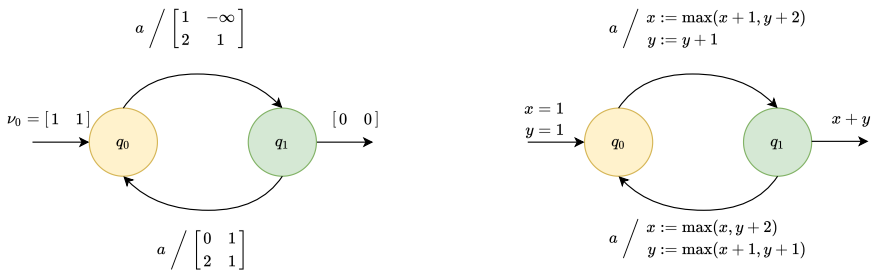
- We define the sum of $M_1, M_2 \in S^{A \times B}$ as $M_1 \oplus M_2 = M \in S^{A \times B}$ such that $M(p, q) = M_1(p, q) \oplus M_2(p, q)$.
- We define the product of $M_1 \in S^{A \times B}$ and $M_2 \in S^{B \times C}$ as a matrix $M_1 \odot M_2 = M \in S^{A \times C}$ with $M(p, r) = \bigoplus_{q \in B}(M_1(p, q) \odot M_2(q, r))$.
- The *identity* matrix $I \in S^{A \times A}$, for some set $A$, is a matrix with $I(p, q) = \mathbb{1}$ if $p = q$ and $I(p, q) = \mathbb{0}$ if $p \neq q$ where $p, q \in A$
- The *zero* matrix $0_A \in S^{A \times A}$, for some set $A$, is a matrix $0_A(p, q) = \mathbb{0}$ for all $p, q \in A$

## 2.2 Cost Register Automata

A cost register automaton is a deterministic finite automaton with write-only registers that are updated on each transition using expressions built from some predefined grammar. The registers hold values from a semiring, and they are updated using the expressions that involve operations on values of registers and elements from the semiring and combined using the semiring operations . We present an equivalent definition of LCRA that uses matrices which is helpful for our discussion.

**Definition 2.1.** A Linear Cost Register Automata (LCRA) over the semiring $\mathbb{S} = (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$ is a tuple $\mathcal{A} = (Q, \Sigma, \mathcal{X}, q_0, \delta, \nu_0, \mu)$ where $Q$ is a finite set of states, $\Sigma$ is an alphabet, $\mathcal{X}$ is a finite set of registers, $q_0 \in Q$ is the initial state, $\delta : (Q \times \Sigma) \to Q \times S^{\mathcal{X}^2}$ is the transition function, $\nu_0 \in S^{\mathcal{X}}$ is the initial valuation, and $\mu : Q \to S^{\mathcal{X}}$ is the output function.

[3] $\mathbb{0}$ is said to be an *absorbing element* for $\odot$ if for all $s \in S, s \odot \mathbb{0} = \mathbb{0} \odot s = 0$




(a) Matrix representation

(b) Expression representation

Figure 1: Two equivalent representations of a CRA over the semiring $\mathbb{N}_{\max}$ and unary alphabet $\{a\}$

A configuration of an LCRA is a pair $(q, \nu) \in Q \times S^{\mathcal{X}}$ . The initial configuration is $(q_0, \nu_0)$. If the current configuration is $(q, \nu)$, and $\delta(q, a) = (q', M)$ where $M \in S^{\mathcal{X}^2}$ is an **update matrix**, then after reading a letter $a \in \Sigma$ the next configuration is defined as $(q', \nu')$ where $\nu' = \nu \odot M$ . We denote this transition by $q \xrightarrow{a,M} q'$, and the update of a register $y \in \mathcal{A}$ for this transition is given by the expression $y := \bigoplus_{x \in \mathcal{X}} (x \odot M(x, y))$. A **run** of the LCRA $\mathcal{A}$ over the word $w = a_1 a_2 \dots a_n$ is a sequence of transitions $\rho = q_0 \xrightarrow{a_1,M_1} q_1 \xrightarrow{a_2,M_2} q_2 \dots \xrightarrow{a_n,M_n} q_n$, where for each $1 \leq i \leq n, \delta(q_{i-1}, a_i) = (q_i, M_i)$ and $M_i \neq 0_{\mathcal{X}}$.

The output of $\mathcal{A}$ over $w$ is defined as $v = \nu_0 \odot M_1 \odot M_2 \cdots M_n \odot \mu(q_n)$ where $\nu_0$ and $\mu(q_n)$ are interpreted as row and column vector, respectively. If there is no run for a word, then we associate the output $\mathbb{0}$ to it. The LCRA $\mathcal{A}$ computes a function $[\![\mathcal{A}]\!] : \Sigma^* \to S$ that maps each word $w$ to the output of $\mathcal{A}$ over $w$.

The *expressiveness* of LCRA can restricted by putting contraints on the update and final expressions.

## 2.3 Weighted Automata

A weighted automaton is a finite state automaton where transitions have associated weights from a semiring $S$. It associates a *weight* (or *value*) from $S$ to each word $w \in \Sigma^*$.

**Definition 2.2.** A weighted automaton (WA) over the alphabet $\Sigma$ and the semiring $\mathbb{S} = (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$ is a tuple $W = (Q, \Sigma, T, \lambda, \gamma)$ where $Q$ is a finite set of states, $T$ is the transition function $T : (Q \times \Sigma \times Q) \to S$, and $\lambda, \gamma : Q \to S$ are the initial and final functions, respectively.
If $W$ is a WA and $p, q \in Q$, $a \in \Sigma$, and $\delta(p, a, q) = s \in S \setminus \{\mathbb{0}\}$, then we denote this by the notation $p \xrightarrow{a,s} q$. A **run** $\rho$ of $\mathcal{W}$ over the word $w = a_1 a_2 \dots a_n$ is a sequence of transitions $q_0 \xrightarrow{a_1,s_1} q_1 \xrightarrow{a_2,s_2} q_2 \dots \xrightarrow{a_n,s_n} q_n$, where for each $1 \leq i \leq n$ we have $T(q_{i-1}, a_i, q_i) = s_i \neq \mathbb{0}$. The *weight* of the run $\rho$ is defined

as $Wt(\rho) = \lambda(q_0) \odot (\odot_{i=1}^{n} s_i) \odot \gamma(q_n)$. A run is called *accepting* if it's weight is non-zero.

The *weight* of $W$ over a word $w$ is defined as the sum ($\oplus$) of weights of all the runs of $W$ over the word $w$, and if there is no run of $W$ over $w$, then we define the weight to be $\mathbb{0}$.

A weighted automaton *computes* a functions $[\![W]\!] : \Sigma^* \to S$. This function maps each word $w$ to the *weight of* $W$ *over the word* $w$.

An equivalent definition of weighted automata using matrices is described in the following.

**Definition 2.3.** A weighted automaton over a semiring $\mathbb{S} = (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$ is a tuple $(Q, \Sigma, \lambda, \mu, \gamma)$ such that $\lambda \in S^{\{1\} \times Q}$, $\gamma \in S^{Q \times \{1\}}$ and $\mu : (\Sigma, \cdot, \{\epsilon\}) \to (S^{Q \times Q}, \cdot, I)$ is a monoid homomorphism [10].

The *weight* of a word $w = w_1 w_2 \dots w_n$ is defined as $[\![W]\!](w) = \lambda \odot \mu(w) \odot \gamma = \lambda \odot (\mu(w_1) \odot \mu(w_2) \dots \mu(w_n)) \odot \gamma$.

We call this representation of a weighted automaton a *linear representation*.

A state $q$ is *initial* if $\lambda(q) \neq \mathbb{0}$, and *final* if $\gamma(q) \neq \mathbb{0}$. A state $q$ is said to be *reachable* from state $q'$ (or state $q'$ *can reach* the state $q$) if there is at least one run starting from $q'$ that ends at $q$ for some word $w$. A state is ***useful*** if it is reachable from some initial state and can also reach some final state. An automaton is ***trimmed*** if all its states are reachable and useful.

# 3 Relation between classes of WA and LCRA

In this section we will define various classes of functions from strings to values via structural restrictions on weighted automata and LCRA, and study the relation between them. We shall also show that non-deterministic WA and LCRA define the same class of functions, i.e., they are equally expressive

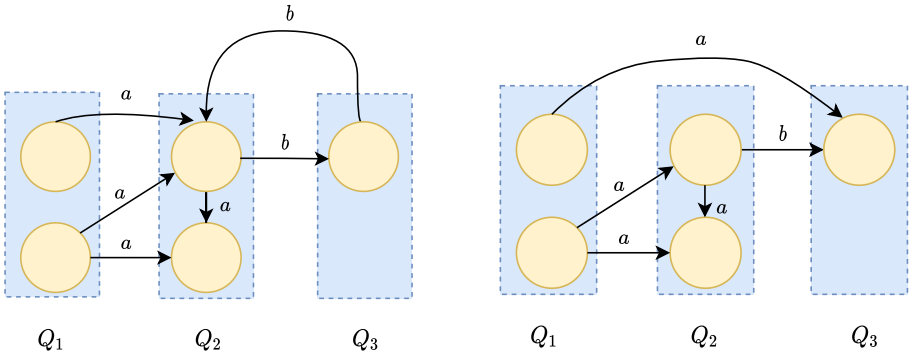
## 3.1 LCRA and WA are equally expressive

It has been proved that the class of functions computable by LCRA is equal to that of weighted automata [5]. We prove this result in the Lemma 3.1, for the sake of completeness.

We introduce some notations that will be useful in proving the main result of this section. For $a, b \in \mathbb{Z}$, we define $[a, b] = \{n \in \mathbb{Z} \mid a \leq n \leq b\}$. Let $W = (Q, T, \lambda, \gamma)$ be a weighted automaton. We define the **running weight** of a run $\rho := q_0 \xrightarrow{a_1, m_1} q_1 \xrightarrow{a_2, m_2} q_2 \dots \xrightarrow{a_n, m_n} q_n$ as $Wt_r(\rho) = \lambda(q_0) \odot \bigodot_{i=1}^{n} m_i$. Note that multiplying the running weight of any run with final weight of its last state, denoted by $end(\rho)$, will give us the weight of the run. Let us denote the set of all accepting runs of a word by $\mathcal{R}(w)$.

A weighted automaton $W = (Q, \lambda, T, \gamma)$ is said to be **layered** with layers $Q_1, Q_2, \dots, Q_m$ if the following conditions are met:

- $Q_i \cap Q_j = \emptyset$ for $i \neq j$




(a) $W_1$ (Layered) (b) $W_2$ (Not layered)

Figure 2: The automata $W_1$ is layered with layers $Q_1, Q_2$ and $Q_3$. The automata $W_2$ is *not* layered with layers $Q_1, Q_2$ and $Q_3$ because the two states in layer $Q_1$ have transition to states belonging to different layer on reading $a$

- $Q = \cup_i Q_i$
- If $T(q, a, p) \neq \mathbb{0}$, $T(q', a, p') \neq \mathbb{0}$ and $q, q' \in Q_i$, then $p, p' \in Q_j$, for some $j \in [1, m]$.

The weighted automaton we will construct in Lemma 3.1 from an LCRA is a layered weighted automaton.

In a layered weighted automaton, on reading a word $w$, all the runs end in the same layer, say $Q_w$, and we say that automaton reaches the layer $Q_w$ on reading $w$. We define the **running weight of the layer** $Q_i$ on reading a word $w$, as a function $Wt_r[Q_i, w] : Q_i \mapsto S$ such that $Wt_r[Q_i, w](q) = \bigoplus_{\rho, end(\rho)=q} Wt_r(\rho)$.

**Lemma 3.1.** *The class of functions computed by weighted automata over a semiring $\mathbb{S}$ is equal to the class of functions computed by LCRA over $\mathbb{S}$.*

*Proof.* ($\Rightarrow$)

Let $W = (Q, \Sigma, \lambda, \mu, \gamma)$ be a weighted automaton in *linear representation*. We construct the corresponding LCRA $\mathcal{A} = (\{\mathbf{q}\}, \Sigma, \mathcal{X}, \mathbf{q}, \delta', \nu_0, \mu')$.

- $\mathcal{X} = \{x_q \mid q \in Q\}$
- $\delta'(\mathbf{q}, a) = (\mathbf{q}, M_a)$ for all $a \in \Sigma$ where $M_a \in S^{\mathcal{X}^2}$ defined as $M_a(x_q, x_{q'}) = \mu(a)(q, q')$ for all $x_q, x_{q'} \in \mathcal{X}$
- $\nu_0(x_q) = \lambda(1, q)$ for all $q \in Q$
- $\mu'(\mathbf{q})(x_q) = \gamma(q, 1)$ for all $q \in Q$

We shall show that $[\![\mathcal{A}]\!] = [\![W]\!]$. Consider a word $w = a_1 a_2 \dots a_n$. The run of $\mathcal{A}$ over $w$ will be $\mathbf{q} \xrightarrow{a_1, M_{a_1}} \mathbf{q} \xrightarrow{a_2, M_{a_2}} \mathbf{q} \dots \xrightarrow{a_n, M_{a_n}} \mathbf{q}$, and the output of $\mathcal{A}$ over $w$ will be $[\![\mathcal{A}]\!](w) = \nu_0 \odot (M_{a_1} \odot M_{a_2} \dots M_{a_n}) \odot \mu'(q)$. Note that up to renaming

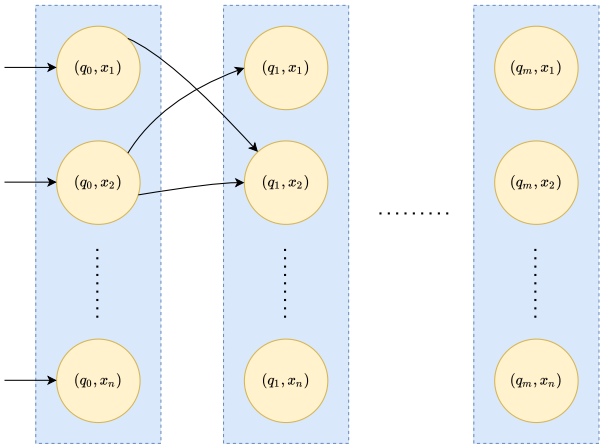



Figure 3: Illustration for the weighted automaton constructed equivalent to a CRA with $m$-states and $n$-registers. The *layers* are shown inside the dotted rectangles

of sets indexing the matrices, we have $M_a \equiv \mu(a)$ for all $a \in \Sigma$, $v_0 = \lambda$, and $\mu' = \gamma$, therefore $[\![\mathcal{A}]\!](w) = \lambda \odot (\mu(a_1) \odot \mu(a_2) \dots \mu(a_n)) \odot \gamma = [\![W]\!](w)$.

($\Leftarrow$) Consider an LCRA $\mathcal{A} = (Q, \Sigma, \mathcal{X}, q_0, \delta, \nu_0, \mu)$. The idea is to have a state $(q, x)$ corresponding to each state-register pair such that, if $\mathcal{A}$ reaches the state $q$ on reading $w$ with value of register $x = c$, then the sum of weights of all runs ending at the state $(q, x)$ on reading $w$, is equal to $c$. We construct an equivalent weighted automaton $W = (Q \times \mathcal{X}, \Sigma, \lambda, \delta', \gamma))$ with

- $\lambda((q_0, x)) = \nu_0(x)$ for all $x \in \mathcal{X}$ and $\lambda((q, x)) = \mathbb{0}$ for all $q \in Q, q \neq q_0, x \in \mathcal{X}$
- If $\delta(q, a) = (q', M)$, then $\delta'((q, x), a, (q', y)) = M(x, y)$ for all $x, y \in \mathcal{X}$, and $\delta'((q, x), a, (p, y)) = \mathbb{0}$ for all $p \in Q \setminus \{q'\}, x, y \in \mathcal{X}$
- $\gamma((q, x) = \mu(q)(x)$ for all $q \in Q, x \in \mathcal{X}$.

It can be readily verified that the constructed automaton is *layered* with the layers being $Q_i = \{(q_i, x) \mid x \in \mathcal{X}\}$. Recall that $Wt_r[Q_k, w](q_k, x)$ is equal to the sum of weights of all runs of $W$ over $w$ ending at the state $(q_k, x)$.

We claim that if $\mathcal{A}$ has the configuration $(q_k, \nu)$ on reading $w$, then $Wt_r[Q_k, w](q_k, x) = \nu(x)$ for all $x \in \mathcal{X}$. We prove the claim by induction on the length of the word $w$. The base case corresponds to $w = \epsilon$. The configuration of CRA $\mathcal{A}$ on reading $\epsilon$ will be $(q_0, v_0)$, and the $W$ will be in the layer $Q_0$. $Wt_r[Q_0, \epsilon](q_0, x)(x) = \lambda((q_0, x)) = \nu_0(x)$ for all $x \in \mathcal{X}$, by construction. Hence, the base case is established.

For the induction step, suppose the claim holds for all words $w$ of length up to $k$, and we need to prove that it holds for the word $wa$ with length $k + 1$.

On reading the word $w$, let the configuration of $\mathcal{A}$ be $(q_i, \nu)$ and $W$ be in the layer $Q_i$. Further, on reading next letter $a$, let the configuration of $\mathcal{A}$ changes to $(q_j, \nu')$ by taking the transition $q_i \xrightarrow{a,M} q_j$, where $\nu' = \nu \odot M$, and the $W$ be in the layer $Q_j$. By induction hypothesis, we assume that $Wt_r[Q_i, w](q_i, x) = \nu(x)$ for all $x \in \mathcal{X}$. All runs of $W$ over $w$ ends at some state of the form $(q_i, y)$ for some $y \in \mathcal{X}$, and each of those run can be extended to a run of $W$ over $wa$ that ends in the state $(q_j, x)$ for some $x \in \mathcal{X}$ if, and only if, $\delta((q, y), a, (q', x))$ is non-zero. Therefore, by using the distributive property of multiplication over addition, we show that for all $x \in \mathcal{X}$

$$\begin{aligned} Wt_r[Q_j, wa](q_j, x) &= \bigoplus_{y \in \mathcal{X}} (\, Wt_r[Q_i, w](q_i, y) \odot \delta((q_i, y), a, (q_j, x))) \\ &= \bigoplus_{y \in \mathcal{X}} Wt_r[Q_i, w](q_i, y) \odot M(x, y) \\ &= \bigoplus_{y \in \mathcal{X}} (\nu(y) \odot M(x, y)) \\ &= \nu'(x) \end{aligned}$$

Let, after reading the word $w$, the CRA $\mathcal{A}$ is in configuration $(q_i, \nu)$, and the weighted automata is in layer $Q_i$ with running weight of the layer $Wt_r[Q_i, w] = f$. The output of the CRA would be $\nu \odot \mu(q_i)$. The weight of $W$ over the word $w$ would be $[\![W]\!](w) = \bigoplus_{P \in Q_i} f(P) \odot \gamma(P) = \nu \odot \mu(q_i)$, as for all $P = (q_i, y) \in Q_i$, we have $f(q_i, y) = \nu(y)$ and $\gamma(P) = \mu(q_i)(y)$. Therefore, $[\![\mathcal{A}]\!](w) = [\![W]\!](w)$. □

## 3.2 Classes of Weighted automata

Given a weighted automaton, suppose that every word $w$ of length $n$ label at most $f(n)$ runs. The nature of the function $f(n)$ gives rise to different types of WA, and corresponding to each type, we get the class of functions which can be computed using that type of WA.

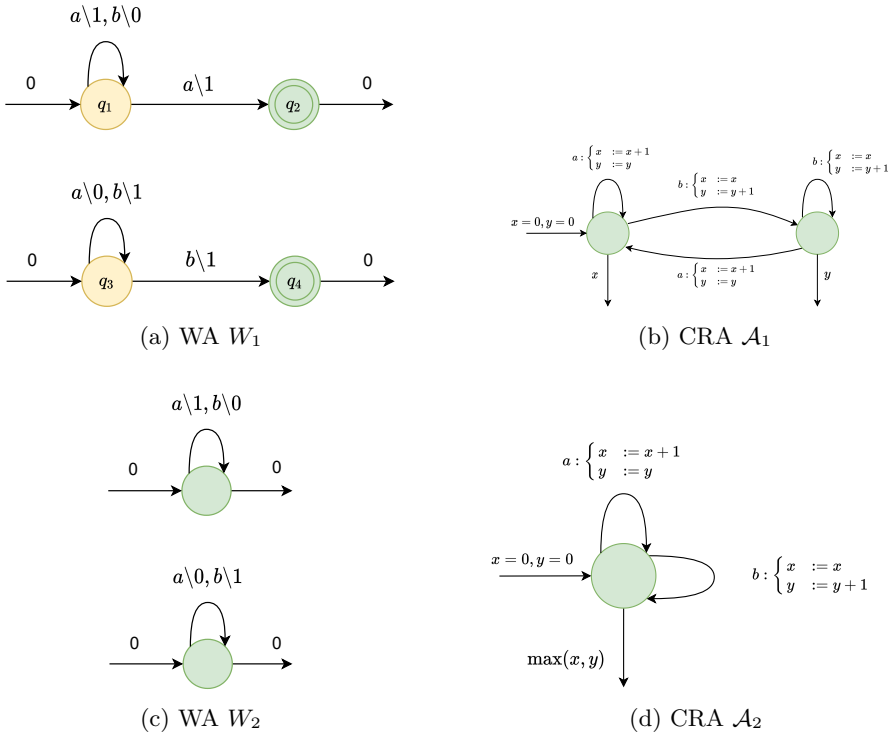
**Definition 3.1.** A weighted automaton $A$ is said to be *unambiguous* ($k$-ambiguous, polynomially ambiguous resp.) if the number of accepting run (with non-zero weight) of $A$ over any word $w$ is bounded by 1 ($k$, $p(|w|)$ for some polynomial $p$, resp.). It is called *finitely ambiguous* if it is $k$-ambiguous for some $k$.

The class of functions that are computable by unambiguous ($k$-ambiguous, finitely ambiguous, polynomially ambiguous resp.) weighted automata is denoted by *UNAMB* (*kAMB*, *FAMB*, *PAMB* resp.)

We will now define deterministic WA.

**Definition 3.2.** A WA is said to be *deterministic* (or *sequential*) if it has at most one initial state and at any state no two outgoing transitions have the same label.

Let $W_1, W_2, \ldots, W_k$ be deterministic weighted automata with disjoint state spaces. The union of $W_i$'s is the automaton formed by *putting $W_i$s together*.




(a) WA $W_1$ (b) CRA $\mathcal{A}_1$

(c) WA $W_2$ (d) CRA $\mathcal{A}_2$

Figure 4: Unambiguous WA $W_1$ and plusfree LCRA $\mathcal{A}_1$ compute the same function $f(wa) = |wa|_a$ and $f(wb) = |wb|_b$.
2-ambiguous WA $W_2$ and plusfree LCRA $\mathcal{A}_2$ compute the same function $f(w) = \max(|w|_a, |w|_b)$.

Formally, let $W_i = (Q_i, \lambda_i, T_i, \gamma_i)$ with $Q_i \cap Q_j = \emptyset \quad$ for $i \neq j \in [1, k]$, then we define the union of $W_i$'s as $W = (\cup_i W_i, \lambda, T, \gamma)$ with $\lambda(q_i) = \lambda_i(q_i)$ and $\gamma(q_i) = \gamma_i(q_i)$, for all $q_i \in Q_i$, and $T(q, a, q') = T_i(q, a, q')$ iff $q, q' \in Q_i$. It can be readily verified that $W$ computes the function $[\![W]\!](w) = \bigoplus_i ([\![W_i]\!](w))$

**Definition 3.3.** A WA is called $k$-sequential if it is union of $k$ sequential (i.e., deterministic) WA. It is called finitely sequential if it is $k$-sequential for some $k$. We denote the class of functions that are computable by $k$-sequential (finitely sequential resp.) automata by $kSEQ$ ($FSEQ$ resp.).

Unlike finite state automata, in general, a WA which is a union of $k$ deterministic WA may not have an equivalent deterministic WA. Few example of automata over the classes we defined are given in the Figure 4.

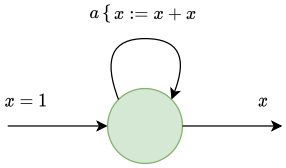



Figure 5: CRA over $\mathbb{N}_{\max}$ computing an exponential function $f_1$

## 3.3 Classes of LCRA

We define the *copyless* LCRA in the following.

**Definition 3.4.** An LCRA $\mathcal{A} = (Q, \mathcal{X}, q_0, \delta, \nu_0, \mu)$ is **copyless** if every update matrix $M$ of $\mathcal{A}$ has at most one nonzero entry in each row. We denote the class of functions that can be computed by a copyless LCRA by $\text{LCRA}^c$. The class of functions computed by copyless LCRA with $k$ registers is denoted by $\text{LCRA}^c_k$.

In terms of the update expressions, this class contains the LCRA that do not ***copy*** any register, i.e, for each transition, each register appears at most once in the right hand side of all the register update expressions.

**Definition 3.5.** An LCRA $\mathcal{A} = (Q, \mathcal{X}, q_0, \delta, \nu_0, \mu)$ is **plusfree** if for each transition, the update expression of each register $x$ is of the form $x := y \odot c$ or $x := \mathbb{0}$ (in matrix representation, every update matrix of $\mathcal{A}$ should have at most one nonzero entry in each column). We denote the class of functions computable by plusfree LCRA in which $\mu(q)$ has at most $k$ non-zero entry for all $q \in Q$ by $\text{LCRA}(\odot)_k$.

We define $\text{LCRA}(\odot)_\oplus = \bigcup_{k\in\mathbb{N}} \text{LCRA}(\odot)_k$. We denote $\text{LCRA}(\odot)_{(\oplus),1}$ by $\text{LCRA}(\odot)_{(\oplus)}$.

**Definition 3.6.** A LCRA is **diagonal** if every update matrix of $\mathcal{A}$ is diagonal matrix (or, the update expressions are of the form $x := x \odot c$). The class of functions computable by diagonal LCRA $\mathcal{A} = (Q, \mathcal{X}, q_0, \delta, \nu_0, \mu)$ in $\mu(q)$ has at most one non-zero entry, for all $q \in Q$, is denoted by $\text{LCRA}^c(\odot)$.

We shall prove that $\text{LCRA}(\odot)_k \subseteq kAMB$, $\text{LCRA}(\odot) = UNAMB$ and $\text{LCRA}^c_k = kSEQ$ in Theorem 3.2, 3.3 and 3.4, respectively. The second result was claimed in [9] for tropical semiring.

## 3.4 $\text{LCRA}(\odot)_k$ and $k$-ambiguous functions

For every function in $\text{LCRA}(\odot)_k$, we can find an equivalent weighted automaton that is at most $k$-ambiguous, this result is proved in the following theorem. The transformation used in the proof is the same as in Theorem 3.1.

**Theorem 3.2.** *The class $LCRA(\odot)_k \subseteq kAMB$.*

*Proof.* Let $f \in \text{LCRA}(\odot)_k$ be a function, and let $\mathcal{A} = (Q, \mathcal{X}, q_0, \delta, \nu_0, \mu)$ be a plusfree CRA computing $f$ with at most $k$ non-zero values in the vector $\mu(q)$

for each $q \in Q$. We use the same construction as described in Lemma 3.1 to construct an equivalent weighted automaton $W = (Q \times \mathcal{X}, \lambda, T, \gamma)$ that computes the function $f$. We assume that $W$ is trimmed with layers $Q_1, Q_2, \ldots, Q_n$, where layer $Q_i = \{q_i\} \times \mathcal{X}$. Note that for all $q_i, q_j \in Q$, $T((q_i, x), a, (q_j, y)) \neq \mathbb{0}$ iff $M(x, y) \neq \mathbb{0}$, where $\delta(q_i, a) = (q_j, M)$.

Let all the runs of $W$ over the word $w = a_1 a_2 \ldots a_l$ end in some state of the layer $Q_i$ . We claim that for each state $(q_i, z) \in Q_i$, there can exist at most one run that ends at $(q_i, z)$. Suppose, for the sake of contradiction, that there exist two distinct runs $\rho := (q_0, x_0) \xrightarrow{a_1} (q_1, x_1) \ldots \xrightarrow{a_l} (q_l, x_l)$ and $\rho' := (q_0, x'_0) \xrightarrow{a_1} (q_1, x'_1) \ldots \xrightarrow{a_l} (q_l, x'_l)$ of $W$ over the word $w$, where $x_l = x'_l = z$, and $q_l \in Q_i$. As the runs are distinct, there should exist $j \in [0, n-1]$ such that $x_j \neq x'_j$, and $x_{j+1} = x'_{j+1}$. Therefore, the CRA has the transitions $(q_j, x_j) \xrightarrow{a_{j+1}} (q_{j+1}, x_{j+1})$ and $(q_j, x'_j) \xrightarrow{a_{j+1}} (q_{j+1}, x_{j+1})$, and thus, $M(x_j, x_{j+1}), M(x'_j, x_{j+1}) \neq \mathbb{0}$, where $\delta(q_j, a_{j+1}) = (q_{j+1}, M)$. As this contradicts the fact that $M$ can have at most one nonzero value in each column, the claim is established.

Due to the restriction on the final function, in each layer $Q_i$, there will be at most $k$ states that are final state (i.e., with non-zero outgoing weights or $\gamma((q_i, y)) \neq \mathbb{0}$). As all runs of $W$ end in the same layer, and there is at at most one run for each state in the layer, there could be at most $k$ accepting run because there are at most $k$ final states in each layer. Therefore, the constructed weighted automata $W$ is at most $k$-ambiguous, so $f \in kAMB$. □

Now, we shall prove the following.

**Theorem 3.3.** *The class LCRA($\odot$) = UNAMB.*

*Proof.* From Lemma 3.2, we know that $\mathrm{LCRA}(\odot) = \mathrm{LCRA}(\odot)_1 \subseteq UNAMB$. It remains to prove that $UNAMB \subseteq \mathrm{LCRA}(\odot)$. Let $f \in UNAMB$ be any function, and $W = (Q, \lambda, T, \gamma)$ be an unambiguous weighted automaton computing $f$. We assume that $W$ is trimmed.

We construct an equivalent CRA $\mathcal{A} = (Q', \mathcal{X}, q_0, \delta, \nu_0, \mu)$ with

- $Q' = 2^Q$
- $\mathcal{X} = \{x_q \mid q \in Q\}$
- $q_0 = \{q \mid \lambda(q) \neq \mathbb{0}\}$
- $\delta(P, a) = (P', M)$ for all $a \in \Sigma$ where $P'$ is the set of all states that can be reached from some state in $P$ via transition labelled $a$ in the automaton $W$, and $M \in S^{\mathcal{X}^2}$ defined as $M(x_q, x_{q'}) = T(q, a, q')$ for all $q \in P, q' \in P'$
- $\nu_0(x_q) = \lambda(q)$ for all $x_q \in \mathcal{X}$
- $\mu(P)(x_q) = \gamma(q)$ if $q \in P$, else it is $\mathbb{0}$, for all $P \in 2^Q$

We remove all the states from $\mathcal{A}$ that are not *useful* to obtain the equivalent automaton $\mathcal{A}'$. By construction, $\mathcal{A}'$ reaches the state $P \in 2^Q$, after reading the

word $w$, if $P$ is exactly the set of states that $W$ can reach on reading $w$. We claim that for each transition $\delta'(P,a) = (P', M)$ in $\mathcal{A}'$ the matrix $M$ has at most one non-zero value in each column. If not, let $M(x_p, x_q) = \delta(p,a,q)$ and $M(x_{p'}, x_q) = \delta(p',a,q)$ be non-zero entry for the transition $\delta'(P,a) = (P', M)$), where $p, p' \in P$ and $q \in$. As $\mathcal{A}'$ is trimmed, by construction, there will exists a word $w$ that labels at least two runs in $W$ that can reach $p$ and $p'$. Consequently, the word $wa$ will have at least two distinct runs in $W$ that both reach $q$. Again, since $W$ is trimmed, there exists a word $waw'$ that extends the the two run from $q$ to reach some final state, or in other words, has two distinct accepting runs, but this contradicts the fact that $W$ is an unambiguous weighted automaton.

We now show that for any state $P$ in $\mathcal{A}'$, there could be only one $q \in P$ for which $\gamma(q) \neq \mathbb{0}$. Suppose, for the sake of contradiction, that there exists distinct $p, p' \in P$ such that $\gamma(p), \gamma(p') \neq \mathbb{0}$. By construction, there must exists a word $w$ that labels at least two accepting runs reaching $p$ and $p'$, but this contradicts that $W$ is unambiguous. Therefore, the vector $\mu(P)$ has at most one non-zero value for all $P$, and the CRA $\mathcal{A}'$ is plusfree with $\mu(q)$ being non-zero for at most one register, and computes the function $f$, so $f \in \text{LCRA}(\odot)$. □

## 3.5 Copyless LCRA and finitely sequential WA

We shall now prove the following Theorem.

**Theorem 3.4.** *The class $LCRA^c_k = kSEQ$.*

*Proof.* ($\Rightarrow$) Let $f \in \text{LCRA}^c_k$ be a function, and let $\mathcal{A} = (Q, \mathcal{X}, q_0, \delta, \nu_0, \mu)$ be the copyless LCRA with $k$ registers computing $f$. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$ and $\mathcal{Q} = Q \times \mathcal{X}$. We construct an equivalent weighted automaton $W = (\mathcal{Q}, \lambda, T, \gamma)$ using the construction described in Lemma 3.1. We define $U = \bigcup_{i=1}^{k} W_i$, where each $W_i = (\mathcal{Q} \times \{i\}, \lambda_i, T_i, \gamma_i)$ is identical to $W$ with possible exception to the initial function.

Formally, for all states $(q,x,i), (q',x',i) \in \mathcal{Q} \times \{i\}$, we have

- $\lambda_i((q_0, x_i, i)) = \lambda(q_0, x_i)$, and $\mathbb{0}$ elsewhere.
- $T_i((q,x,i), a, (q',x',i)) = T((q,x), a, (q',x'))$
- $\gamma_i((q,x,i)) = \gamma((q,x))$

It is readily checked that $U$ defines the same function $f$ . We will prove that each $W_i$ is sequential. Each state in $W_i$ has at most a single outgoing transition on each letter. Suppose not, for the sake of contradiction, then let $T_i((q,x,i), a, (p,y,i)), T_i((q,x,i), a, (p,y',i)) \neq \mathbb{0}$ where $y \neq y'$, and therefore, $M(x,y), M(x,y') \neq \mathbb{0}$ where $\delta(q,a) = (p,M)$, but this contradicts the assumption that each matrix $M$ has at most one non-zero value in each row. As each $W_i$ has a single initial state, $W_i$ is deterministic.

($\Leftarrow$) Let $f \in kSEQ$ be a function, and $U = \bigcup_{i=1}^{k} W_i$ be the finitely sequential weighted automaton computing $f$, where each $W_i = (Q_i, q_{0i}, \lambda_i, \delta_i, \gamma_i)$ is sequential. The idea is to construct a LCRA $\mathcal{A}$ that stores the running weight

of each $W_i$ over a word in a unique register $x_i$. Formally, $\mathcal{A} = (Q, \mathcal{X}, q_0, \delta, \nu_0, \mu)$ with

- $Q = Q_1 \times Q_2 \ldots \times Q_k$
- $\mathcal{X} = \{x_i \mid 1 \leq i \leq k\}$
- $\delta((q_1, q_2, \ldots q_k), a) = ((q'_1, q'_2, \ldots q'_k), M)$ where $\delta_i(q_i, a) = (q'_i, m_i)$ and $M$ is a diagonal matrix with $M(x_i, x_i) = m_i$
- $v_0(x_i) = \lambda_i(q_{0i})$
- $\mu((q_1, q_2, \ldots, q_k))(x_i) = (\gamma_i(q_i))$

Let the final configuration of $\mathcal{A}$ after reading $w$ be $((q_1, q_2, \ldots, q_n), \nu)$ and $\rho_i$ be the unique run of $W_i$ over $w$, then it follows directly from the construction, that $\nu(x_i)$ will hold exactly the running weight of $W_i$ over $w$, or $\nu(x_i) = Wt_r(\rho_i)$. We have

$$\begin{aligned}
[\![U]\!](w) &= \bigoplus_i (Wt_r(\rho_i) \odot \gamma_i(q_i)) \\
&= \bigoplus_i (\nu(x_i) \odot \gamma_i(q_i)) \\
&= \bigoplus_i (\nu(x_i) \odot \mu((q_1, q_2, \ldots, q_k))(x_i))) \\
&= \nu \odot \gamma((q_1, q_2, \ldots, q_k)) \\
&= [\![\mathcal{A}]\!](w)
\end{aligned}$$

□

Note that each matrix of the copyless $CRA$ constructed from a finitely sequential weighted automaton in Lemma 3.4 is a *diagonal matrix*, and hence we get the following corollary.

**Corollary 3.4.1.** *Let $\mathcal{A} \in LCRA^c_k$. We can construct an equivalent LCRA $\mathcal{B}$ with $k$ registers such that each update expression in the CRA is of the form $x = x \odot c$ (in matrix representation, each update matrix is a diagonal matrix).*

Another direct corollary of Lemma 3.4 is the following that has been mentioned in [9].

**Corollary 3.4.2.** *The class $LCRA^c = FSEQ$.*

The Figure 6 illustrates the relation we have proved so far. In general, all these classes are distinct.

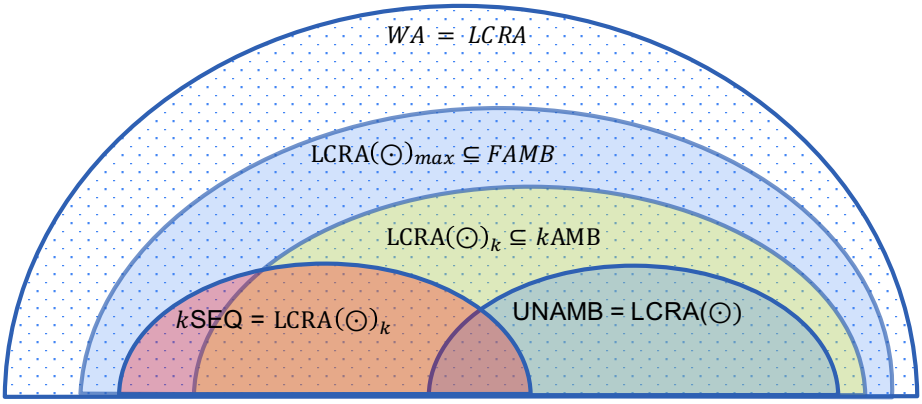



Figure 6: The illustration of relation between classes of LCRA and WA over an arbitrary semiring.

# 4 Relation between unambiguous finitely sequential WA and diagonal LCRA

In this section, we shall first prove that if a function $f \in FSEQ \cap UNAMB$, then the function is computable by a weighted automaton which is both, finitely sequential and unambiguous. We use this result to prove that the class $\text{LCRA}^c(\odot) = FSEQ \cap UNAMB$

We shall assume that all CRA and weighted automata in this section are, unless otherwise stated, over the semiring $\mathbb{N}_{\max}$. We have the Theorem 4.1 that proves that the class of functions computable by the diagonal LCRA with at most one non-zero value in final function for each state, i.e., $\text{LCRA}^c(\odot)$, is equal to $FSEQ \cap UNAMB$.

**Theorem 4.1.** *The class* $LCRA^c(\odot) = FSEQ \cap UNAMB$.

In subsection 4.1 and, we will study the Lemma that are necessary to prove this result. In the following section, we prove a few elementary results which are necessary for our proof.

## 4.1 Class of functions computable by WA over $\mathbb{Z}_{\max}$ is closed under difference

We define the negation of the function computed by an $\mathbb{Z}$-automata in the following definition. Note that we define $-(-\infty) = -\infty$.

**Definition 4.1.** Let $A$ be a $\mathbb{Z}$-automata. We define the **negation function** of $A$ as $[\![-A]\!](w) = -[\![A]\!](w)$.

We first prove in the following lemma shows that the negation function of a $\mathbb{Z}$-automaton is computable by a $\mathbb{Z}$-automaton.

**Lemma 4.2.** *Given a $\mathbb{Z}$-automaton $A = (Q, \lambda, T, \gamma)$, there exists a $\mathbb{Z}$-automaton $B$ that computes the negation function of $A$. Moreover, if $A$ is $k$-ambiguous, then so is $B$.*

*Proof.* Construct an $\mathbb{Z}$-automaton $B = (Q, \lambda', T, \gamma)$ with

- $\lambda'(q) = -\lambda(q)$ for all $q \in Q$
- $T'(p, a, q) = -T(p, a, q)$ for all $p, q \in Q$
- $\gamma(q) = -\gamma(q)$ for all $q \in Q$

It can be readily seen that $[\![B]\!] = [\![-A]\!]$, and if $A$ is $k$-ambiguous, then $B$ would also be $k$-ambiguous. □

We define the addition of the functions computed by the automata as follows.

**Definition 4.2.** Let $A$ and $B$ be two $\mathbb{Z}$-automata. We define the function $[\![C]\!] = [\![A]\!] + [\![B]\!]$ as $[\![C]\!](w) = [\![A]\!](w) + [\![B]\!](w)$.

It is obvious that $C(w)$ would be not equal to $-\infty$ iff $A(w)$ and $B(w)$, both, are not equal $-\infty$. In the following Lemma, we show that we can compute the addition $[\![A]\!] + [\![B]\!]$ with a $Z$-automaton.

**Lemma 4.3.** *Given two $\mathbb{Z}$-automata $A$ and $B$ over $\mathbb{N}_{\max}$, we can construct a $\mathbb{Z}$-automata $C$ that computes the function $[\![A]\!] + [\![B]\!]$. Moreover, if $A$ is unambiguous, and $B$ is $k$-ambiguous, then $C$ would also be $k$-ambiguous.*

*Proof.* Let $A = (Q_1, \lambda_1, T_1, \gamma_1)$ and $B = (Q_2, \lambda_2, T_2, \gamma_2)$. Construct $C = (Q, \lambda, T, \gamma)$ with

- $Q = Q_1 \times Q_2$
- $\lambda((q_1, q_2)) = \lambda_1(q_1) + \lambda_2(q_2)$, for all $q_1 \in Q_1, q_2 \in Q_2$
- $T((q_1, q_2), a, (q_1', q_2') = T_1(q_1, a, q_1') + T_2(q_2, a, q_2')$ for all $q_1 \in Q_1$, $q_2 \in Q_2$ and $a \in \Sigma$
- $\gamma((q_1, q_2)) = \gamma_1(q_1) + \gamma_2(q_2)$, for all $q_1 \in Q_1, q_2 \in Q_2$

Let $w = a_1 a_2 \dots a_n$, and $\rho = (q_0, q_0') \xrightarrow{a_1} (q_1, q_1') \dots \xrightarrow{a_n} (q_n, q_n')$ be any run, whether accepting or not, of $C$ over $w$. Let us define $\Pi_A(\rho) := q_0 \xrightarrow{a_1} q_1 \dots \xrightarrow{a_n} q_n$, and $\Pi_B(\rho) := q_0' \xrightarrow{a_1} q_1' \dots \xrightarrow{a_n} q_n'$. It readily follows from the definition of $C$ that $\Pi_A(\rho)$ and $\Pi_B(\rho)$ are runs of $A$ and $B$ over $w$ respectively, and $Wt(\Pi_A(\rho)) + Wt(\Pi_B(\rho)) = Wt(\rho)$. In addition, $\Pi_A(\rho)$ and $\Pi_B(\rho)$ are accepting runs of $A$ and $B$ respectively if, and only if, $\rho$ is an accepting run of $C$.

Let $\rho_1 = q_0 \xrightarrow{a_1} q_1 \dots \xrightarrow{a_n} q_n$ and $\rho_2 = q_0' \xrightarrow{a_1} q_1' \dots \xrightarrow{a_n} q_n'$ be any two runs of $A$ and $B$ over $w$ respectively. We define $\rho_1 \times \rho_2 := (q_0, q_0') \xrightarrow{a_1} (q_1, q_1') \dots \xrightarrow{a_n} (q_n, q_n')$.

Let $\rho$ be the accepting run of $A$ over $w$, and $\sigma_1, \sigma_2, \dots, \sigma_j$, where $j \leq k$, be all the accepting runs of $B$ over $w$. Then, for $i \in [1, j]$, $\rho \times \sigma_i$ should be

accepting run of $C$ over $w$ with weight $Wt(\rho) + Wt(\sigma_i)$. If there is any other accepting run of $C$ over $w$, then this will either contradict the assumption that $A$ is unambiguous, or $B$ has exactly $j$ accepting runs over $w$. As $B$ can have at most $k$ accepting runs over any word, therefore $C$ is $k$-ambiguous. Finally,

$$\begin{aligned}
[\![C]\!](w) &= \max_{i\in[1,j]} (Wt(\rho \times \sigma_i)) \\
&= \max_{i\in[1,j]} (Wt(\rho) + Wt(\sigma_i)) \\
&= Wt(\rho) + \max_{i\in[1,j]} (Wt(\sigma_i)) \\
&= [\![A]\!](w) + [\![B]\!](w)
\end{aligned}$$

□

We now define the difference of functions computed by two $\mathbb{Z}$-automata.

**Definition 4.3.** Let $A$ and $B$ be two $\mathbb{Z}$-automata. We define the function $[\![C]\!] = [\![B]\!] - [\![A]\!]$ as $[\![C]\!](w) = [\![B]\!](w) - [\![A]\!](w)$.

A direct corollary of Lemma 4.3 is following.

**Corollary 4.3.1.** *Given two $\mathbb{Z}$-automata $A$ and $B$, we can construct a $\mathbb{Z}$-automata $C$ that computes the function $[\![B]\!] - [\![A]\!]$. Moreover, if $A$ is unambiguous, and $B$ is $k$-ambiguous, then $C$ would be $k$-ambiguous.*

*Proof.* Use Lemma 4.2 to construct the $\mathbb{Z}$-automata $A'$ computing the function $-[\![A]\!]$, and then use Lemma 4.3 to construct the $k$-ambiguous $Z$-automata $C$ computing the addition of functions computed by $B$ and $A'$. □

## 4.2 $\mathbb{Z}$-automata which compute positive functions can be converted into $\mathbb{N}$-automaton over max-plus semiring

We introduce the notation used in the next result. Let $W = (Q, \lambda, T, \gamma)$ be a weighted automaton over the semiring $\mathbb{S} = (\mathbf{S}, \oplus, \odot, \mathbb{1}, \mathbb{0})$. We define the **internal weight** of a run $\rho := q_0 \xrightarrow{a_1,m_1} q_1 \xrightarrow{a_2,m_2} q_2 \dots \xrightarrow{a_n,m_n} q_n$ as $Wt_{in}(\rho) = \bigodot_{i=1}^{n} m_i$. Note that $q_0$ ($q_n$, resp.) need not be an initial (final, resp.) state.

In the following Lemma we prove a lower bound on the internal weight of any run over an $\mathbb{Z}$-automaton which computes a positive function.

**Lemma 4.4.** *Let $C$ be a trimmed $\mathbb{Z}$-automaton with $[\![C]\!] \geq 0$, There exists a number $M \in \mathbb{N}$ such that internal weight of any run of $C$ over any word $w$ cannot be smaller than $-M$.*

*Proof.* Let $V$ be the set of all weight on the transitions, initial weights, and final weights of $C$, and $v_{\max} = max(\{|v| \mid v \in V\})$, where $|v|$ denotes the absolute value of $v$. Let $M' = v_{\max} \cdot N$, where $\cdot$ denotes the usual multiplication for integers and $N = |Q|$. Let $M = 2 \cdot M'$.

We shall prove this claim by contradiction. Suppose there exists a run $\rho_w$ of $C$ over $w$ that starts at the state $q$ and ends at $p$, and $Wt_{in}(\rho_w) < -2M'$. Since $W$ is trimmed, there exists a word that label runs from some initial state $q_0$ to $q$ ($p$ to a final state $q_f$, resp.), let $u$ ($u'$, resp.) be smallest such word. It can be readily seen that the length of $u, u'$ are bounded by $N-1 = |Q|-1$, and hence the internal weight of run from $q_0$ to $q$ and $p$ to $q_f$ on reading $u$ and $u'$ are bounded by $v_{\max} \cdot (N-1)$. Therefore, there exists a run $\rho$ of $C$ over $uwu'$ which starts at initial state $q_0$ and ends at final state $q_f$ with internal weight $l' = Wt_{in}(u) + Wt_{in}(w) + Wt_{in}(u') < v_{\max} \cdot (N-1) - 2M' + v_{\max} \cdot (N-1) = -2 \cdot v_{\max}$. This would be an accepting run, and since initial and final weights are also bounded by $v_{\max}$, the weight of this run $Wt(\rho) < -2 \cdot v_{\max} + 2 \cdot v_{\max} = 0$. However, $C$ is unambiguous, so the weight of $C$ over $uwu'$ is $Wt(\rho) < 0$ which contradicts the fact that $[\![C]\!] \geq 0$. □

The following lemma shows that we can always assume that for any $f \in$ *UNAMB*, the initial and final weights of the unambiguous weighted automaton computing a function $f$ are either $\mathbb{0}$ or $\mathbb{1}$.

**Lemma 4.5.** *For every unambiguous weighted automaton $W = (Q, \lambda, T, \gamma)$ over $\mathbb{S} = (\mathbf{S}, \oplus, \odot, \mathbb{1}, \mathbb{0})$, there exists an equivalent unambiguous weighted automaton $U = (Q', \lambda', T', \gamma')$ with the initial and final weights being either $\mathbb{0}$ or $\mathbb{1}$ for all the states.*

*Proof.* Let $Q' = Q \times \{-1, 0, 1\}$. For all $q \in Q$, we define $\lambda'(q, -1) = \mathbb{1}$, $\lambda'(q, 0) = \lambda'(q, 1) = \mathbb{0}$, $\gamma'(q, -1) = \gamma'(q, 0) = \mathbb{0}$ and $\gamma'(q, 1) = \mathbb{1}$. For all $p, q \in Q$, we define

$$T'((p, x), a, (q, y)) = \begin{cases} T(p, a, q) \odot \lambda(q), & \text{if } x = -1, y = 0 \\ T(p, a, q), & \text{if } x = y = 0 \\ T(p, a, q) \odot \gamma(q), & \text{if } x = 0, y = 1 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

Note that any accepting run of $U$ over a word $w$ can only, and it must, reach a state $(q_f, 1) \in Q \times \{1\}$ at the end. For an accepting run $\rho := q_0 \xrightarrow{a_1} q_1 \dots q_{n-1} \xrightarrow{a_n} q_n$ of $W$ over $w = a_1 a_2 \dots a_n$, we can find find a unique accepting run $\rho' := (q_0, -1) \xrightarrow{a_1} (q_1, 0) \dots (q_{n-1}, 0) \xrightarrow{a_n} (q_n, 1)$ of $U$ over $w$, and vice versa. Therefore, $U$ is unambiguous. We have

$$\begin{aligned} Wt(\rho) &= \lambda(q_0) \odot \bigodot_{i=0}^{n-1} (T(q_i, a_{i+1}, q_{i+1})) \odot \gamma(q_n) \\ &= (\lambda(q_0) \odot (T(q_0, a_1, q_1)) \odot \bigodot_{i=1}^{n-2} (T(q_i, a_{i+1}, q_{i+1})) \odot (T(q_{n-1}, a_n, q_n) \odot \gamma(q_n)) \\ &= T((q, -1), a_1, (q_1, 0)) \odot \bigodot_{i=1}^{n-2} (T((q_i, 0), a_{i+1}, (q_{i+1}, 0))) \odot T((q_{n-1}, 0), a_n, (q_n, 1)) \\ &= Wt(\rho') \end{aligned}$$

Therefore, $[\![U]\!] = [\![W]\!]$. □

In the following lemma, we prove that if a nonnegative function $f$ is computable by $\mathbb{Z}$-automaton then $f$ is also computable by an $\mathbb{N}$-automaton. This result is also given in [2], but we give a different proof here.

**Lemma 4.6.** *Let $C$ be an unambiguous $\mathbb{Z}$-automaton with $[\![C]\!] \geq 0$. There exists an $\mathbb{N}$-automata $D$ such that $[\![D]\!] = [\![C]\!]$.*

*Proof.* We assume that $C$ is trimmed, and, using Lemma 4.5, that all the initial and final weights are either $-\infty$ or 0. Using Lemma 4.4, let $M \in \mathbb{N}$ be a number such that internal weight of any run of $C$ over $w$ $Wt_{in}(w) \geq -M$.

Let us define $C_i = Q \times \{i\}$, and $Q' = \bigcup_{i \in [-M,M]} C_i$. We construct an $\mathbb{N}$-automaton $D = (Q', \lambda', T', \gamma')$ over $\mathbb{N}_{\max}$ with

- For all $a \in \Sigma$ and $l, l' \in [M, M]$,

$$T'((q,l),a,(q',l')) = \begin{cases} 0, & \text{if } l' - l = T(q,a,q') \\ l + T(q,a,q') - l', & \text{if } l + T(q,a,q') > M = l' \\ -\infty, & \text{otherwise} \end{cases}$$

- For all $q \in Q, l \in [M, M]$, $\lambda'(q,l) = \lambda(q)$ if $l = 0$, otherwise $-\infty$
- For all $q \in Q, l \in [M, M]$, $\gamma'((q,l)) = \gamma(q) + l$ if $l \geq 0$, otherwise $-\infty$

We need to prove that $[\![D]\!] = [\![C]\!]$. Figure 7 may help understand the following argument. The intuition behind the construction is that we *store* the running weight of the run by moving from one copy of $C$ to another to avoid taking any transition with negative weight. The formal proof of the correctness of the construction is given below.

Let $w = a_1 a_2 \dots a_n$ be a word. Let $\rho = (q_0, c_0) \xrightarrow{a_1} (q_1, c_1), \dots, \xrightarrow{a_n} (q_n, c_n)$ be a run of $D$ over $w$, we define $\Pi_1(\rho) := q_0 \xrightarrow{a_1} q_1, \dots, \xrightarrow{a_n} q_n$. Let $R_C(w)$ and $R_D(w)$ be the set of all runs, that from some initial state [4], of $C$ and $D$ over $w$, respectively. We shall prove that for each run $\rho \in R_C(w)$ with $end(\rho) = q$, there exists a corresponding run $\rho' \in R_D(w)$ with $end(\rho') = (q, l)$ such that following hold:

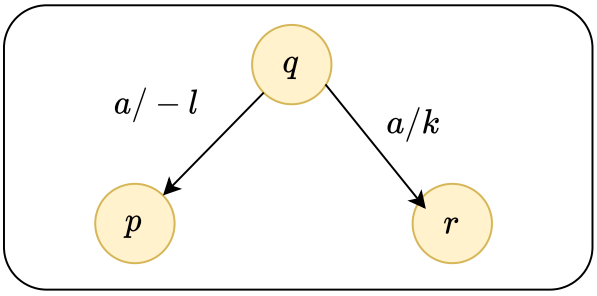
**P.1** $\rho = \Pi_1(\rho')$. In other words, the projection of run of $D$ over the first component is a run over $C$

**P.2** $Wt_r(\rho) = Wt_r(\rho') + l$

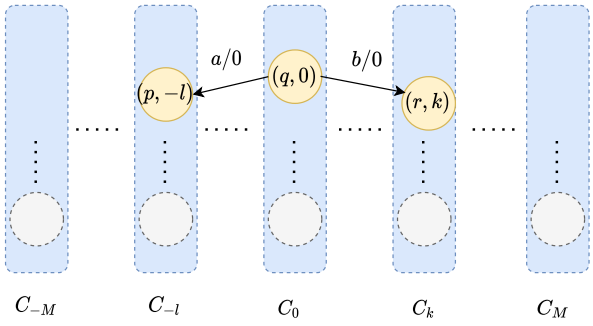
**P.3** $Wt_r(\rho') > 0$ then $l \geq 0$

We prove this by induction on length of the word. Note that $Wt_r(\rho) = Wt_{in}(\rho)$ for any run $\rho$ of $C$ or $D$ because the initial weights is 0 for all initial states in both $C$ and $D$. For the base case $\epsilon$, the run of $C$ with state $q \in Q$ corresponds to

[4] Each run considered in this proof starts from some initial state unless otherwise stated




(a) Automaton $C$




(b) Automaton $D$

Figure 7: Illustration of construction of $\mathbb{N}$-automaton $D$ for the $\mathbb{Z}$-automaton $C$. Corresponding to a transition with negative weight $-l$ in $C$, we move from $C_i$ to $C_{i-l}$ with 0 weight, and for positive weight $k$ in $C$, we move from $C_i$ to $C_{i+k}$ with weight 0 if $i + k \leq E$, otherwise to $C_E$ with weight $i + k - E$, while keeping track of the states in $C$

the run $D$ with initial state $(q,0) \in Q'$, and it can be readily seen that **P.1**, **P.2**, and **P.3** hold. Assume that the result holds for word $w$ of length upto $k$. We shall prove that the result holds for the word $wa$ of length $k+1$. Each run of $C$ ($D$, resp.) over $wa$ will be an extension[5] of some run $\rho$ in $R^C(w)$ ($\rho' \in R^D(w)$, resp.). Let $\rho \in R^C(w)$ and $\rho' \in R^D(w)$ be any two corresponding runs, where $end(\rho) = q$ and $end(\rho') = (q,l)$. We shall show that the we can extend $\rho$ with transition $q \xrightarrow{a} q'$ iff we can extend $\rho'_1$ of $\rho'$ with $(q,l) \xrightarrow{a} (q',l')$ such that $\rho_1$ and $\rho'_1$ satisfy **P.1** to **P.3**.
($\Rightarrow$) Let $d = T(q,a,q')$, and $p = l+d$. Let $l' = \min(p,M)$. We claim that we can extend the run $\rho'$ with the transition $(q,l) \xrightarrow{a} (q',l')$ with weight $\max(0,p-M))$ to get the run $\rho'_1$ of $D$ over $wa$. If we prove that $l' \in [-M,M]$, and **P.3** holds for $l'$, then it immediately follows from the construction that **P.1** and **P.2** hold. We will consider the following two cases.

**Case 1.** $Wt_r(\rho') = 0$: We have $Wt_r(\rho) = Wt_r(\rho') + l = l$. We have $Wt_r(\rho_1) = Wt_r(\rho) + d = l + d$, but since $Wt_r(\rho_1) \geq -M$, we have $p = l + d \geq -M$. Hence, $-M \leq p = l' \leq M$, or $l' \in [-M,M]$. If $Wt_r(\rho'_1) > 0$, then we must have taken the transition to a state of the form $(q,M)$, and hence $l' = M \geq 0$.

**Case 2.** $Wt_r(\rho') > 0$: It is clear from the construction that $Wt_r(\rho') > 0$ is only possible if a state of the form $(r,M)$ is visited during the run $\rho'$, and let $(r,M)$ be the last such state visited in $\rho'$. Let us divide the run $\rho' = (q_0,0) \rightsquigarrow_u (r,M) \rightsquigarrow_v (q,l)$ into $\sigma'_1 = (q_0,0) \rightsquigarrow_u (r,M)$ and $\sigma'_2 = (r,M) \rightsquigarrow_v (q,l)$. Thus, we have $Wt_r(\rho') = Wt_r(\sigma'_1) + Wt_{in}(\sigma'_2)$. In the run $\sigma'_2$, all transitions are of weight 0, and therefore, $Wt_{in}(\sigma'_2) = 0$. Using induction hypothesis, we can also divide $\rho$ into $\sigma_1 = q_0 \rightsquigarrow_u r$ and $\sigma_2 = r \rightsquigarrow_v q$, where $\sigma_1$ corresponds to $\sigma'_1$, and $Wt_r(\sigma_1) = Wt_r(\sigma'_1) + M$. Now, we have

$$\begin{aligned} Wt_r(\rho_1) &= Wt_r(\rho) + d \\ &= Wt_r(\rho') + l + d \\ &= Wt_r(\sigma'_1) + Wt_{in}(\sigma'_2) + l + d \\ &= Wt_r(\sigma_1) - M + l + d \end{aligned}$$

But, $Wt_r(\rho) = Wt_r(\sigma_1) + Wt_{in}(\sigma_2)$, therefore, $Wt_{in}(\sigma_2) = l + d - M$. But, from Lemma 4.4, $Wt_{in}(\sigma_2) \geq -M$, or $l + d - M \geq -M$ or $l + d \geq 0$. Let $l' = \min(p,M)$. Therefore, $l' \in [0,M]$.

($\Leftarrow$) Suppose, we can extend the run $\rho'$ with the transition $t := (q,l) \xrightarrow{a} (q',l')$ to get the run $\rho'_1$. By construction, $d = T(q,a,q') \neq -\infty$. Therefore, we can extend the run $\rho$ with transition $q \xrightarrow{a} q'$ to get the run $\rho_1$. It is straightforward that **P.1** holds. We consider the following two cases to prove that **P.2** and **P.3** hold.

[5]Let $\rho = q_0 \rightsquigarrow_w q$ be a run of automata $A$ over $w$ with $end(rho) = q$, and $t = q \xrightarrow{a} q'$ be a transition. The run $\rho' = q_0 \rightsquigarrow_w q \xrightarrow{a} q'$, which exactly the same as $\rho$ except it has an additional transition from the end state of $\rho$, is defined to be *extension* of $\rho$ with transition $t$

**Case 1.** $l' - l = d$: The weight of the transition $(q, l) \xrightarrow{a} (q', l')$ will be 0, in which case $Wt_r(\rho_1) = Wt_r(\rho) + d = Wt_r(\rho') + l + d = Wt_r(\rho') + l'$. Thus, **P.2** holds. If $Wt_r(\rho'_1) > 0$, then $Wt_r(\rho') > 0$ as weight of $t$ is 0.

Otherwise, by construction, the transition $t$ could only be taken if $l' = M$ and $l + d > M$. In that case,

$$\begin{aligned} Wt_r(\rho_1) &= Wt_r(\rho) + d \\ &= Wt_r(\rho') + l + d \\ &= Wt_r(\rho') + (l + d - M) + M \\ &= Wt_r(\rho'_1) + M \text{ , as weight of } t \text{ is } (l + d - M) \end{aligned}$$

Thus, **P.2** holds. It is left to prove that **P.3** holds. If $Wt_r(\rho'_1) = 0$, then there is nothing to prove. Otherwise, if $Wt_r(\rho'_1) > 0$, then $Wt_r(\rho') > 0$. Using the same argument as Case 2 in the other direction, we can prove that $l' \geq 0$.

**Case 2.** $l' = M$ and $l + d > M$: In this case, the weight of the transition is $l + d - M$, and hence the weight of the run

$$\begin{aligned} Wt_r(\rho'_1) + M &= Wt_r(\rho') + (l + d - M) + M \\ &= Wt_r(\rho) + d \\ &= Wt_r(\rho_1) \end{aligned}$$

Hence, **P.2** is established. Also, since $l' = M > 0$, therefore, **P.3** trivially holds.

To finish the proof of $[\![D]\!] = [\![C]\!]$, let $\rho \in R^C(w)$ and $\rho' \in R^D(w)$ be any two corresponding runs, where $end(\rho) = q$ and $end(\rho') = (q, l)$. If $\gamma(q) = -\infty$, then $\gamma'((q, l)) = -\infty$, and hence $Wt(\rho) = Wt(\rho') = -\infty$. Otherwise, $\gamma(q) = 0$. If $Wt_r(\rho') = 0$, then from **P.2**, $Wt_r(\rho) = l \geq 0$ as $[\![C]\!] \geq 0$, and if $Wt_r(\rho') > 0$, then from **P.3**, $l \geq 0$. Now, by construction, $\gamma((q, l)) = \gamma(q) + l = l$, and therefore, $Wt(\rho') = Wt_r(\rho') + \gamma((q, l)) = Wt_r(\rho) + l = Wt(\rho)$. Therefore, $\max_{\rho \in R^C(w)}(Wt(\rho)) = \max_{\rho' \in R^D(w)}(Wt(\rho'))$, or $[\![D]\!] = [\![C]\!]$. It can be readily observed that $D$ is unambiguous as each accepting run of $D$ corresponds to an accepting run of $C$, and $C$ is unambiguous. □

Any word in an unambiguous automaton $A$ over $\mathbb{N}_{\max}$ would have the weight 0 iff each transition in it's accepting run has the weight 0. The set of words which have the weight 0 over $A$ is regular.

**Lemma 4.7.** *Given an unambiguous automaton $A$ over $\mathbb{N}_{\max}$, the set then $R = \{w \mid [\![A]\!](w) = 0\}$ is a constructible regular set.*

*Proof.* Let $A = (Q, \lambda, T, \gamma)$. We construct another automata $A' = (Q, \lambda', T', \gamma')$ which is same as $A$ in structure but all the transitions, initial weights, and final weights which are not 0 are removed. Formally, for all $q, q' \in Q$, $\lambda'(q) = \lambda(q)$ if $\lambda(q) = 0$, else $-\infty$, $T'(q, a, q') = T(q, a, q')$ if $T(q, a, q') = 0$, else $-\infty$, and $\gamma'(q) = \gamma(q)$ if $\gamma(q) = 0$, else $-\infty$. It is clear that the weight of every word over $A'$ is either 0 or $-\infty$. We claim that for all words $w$, $[\![A]\!](w) = 0$ iff $[\![A']\!](w) = 0$.

As $A$ is unambiguous, $[\![A]\!](w) = 0 \Leftrightarrow$ there is a run $\rho$ of $A$ over $w$ with initial weight, weight of each transition, and final weight equal to $0 \Leftrightarrow$ there is an accepting run $\rho'$ of $A'$ over $w \Leftrightarrow [\![A']\!](w) = 0$.

As all the words accepted by $A'$ have weight 0, consider the finite automata (NFA) $B$ $(Q, I, \Delta, F)$ with $I = \{q \mid \lambda(q) = 0\}$, $\Delta(q, a) = \{q' \mid T(q, a, q') = 0\}$, and $\gamma(q) = \{q \mid \gamma(q) = 0\}$. It follows readily that $B$ accepts the set of words that have weight 0 in $A$. □

In the following Lemma, we prove that given a $k$-ambiguous weighted automaton $A$ and a DFA $B$, we can construct a $k$-ambiguous weighted automaton $C$ that outputs $A(w)$ if $w \in L(B)$, else does not accept $w$.

**Lemma 4.8.** *Let $A = (Q, \lambda, T, \gamma)$ be an $k$-ambiguous (sequential, resp.) automaton over the semiring $\mathbb{S} = (\mathbf{S}, \oplus, \odot, \mathbb{1}, \mathbb{0})$, and $R$ be a regular set. We can construct the $k$-ambiguous (sequential, resp.) automaton $C$ over $\mathbb{N}_{\max}$ such that $[\![C]\!](w) = [\![A]\!](w)$ if $w \in R$, else $\mathbb{0}$.*

*Proof.* Let $B = (P, p_0, \delta, F)$ be the DFA, with state space $P$, initial state $p_0$, transition function $\delta$, and final state set $F$, recognizing the language $R$. Construct $C = (R, \lambda', T', \gamma')$ with $Q' = Q \times P$ such that for all $q, q' \in Q, p, p' \in P$,

- $$\lambda'(q, p) = \begin{cases} \lambda(q), & \text{if } p = p_0 \\ \mathbb{0}, & \text{otherwise} \end{cases}$$
- $$\gamma'(q, p) = \begin{cases} \gamma(q), & \text{if } q \in F \\ \mathbb{0}, & \text{otherwise} \end{cases}$$
- $$T'((q, p), a, (q', p') \begin{cases} T(q, a, q'), & \text{if } p' = \delta(p, a) \\ \mathbb{0}, & \text{otherwise} \end{cases}$$

The projection of any run $\rho$ of $C$ over the first coordinate of the states will give a run $\rho'$ over $A$ with $Wt(\rho) = Wt(\rho')$. This immediately implies that if $A$ is unambiguous, then $C$ is unambiguous. Furthermore, the projection of any run of $C$ over the second coordinate of the state will give us a accepting run over $B$. Hence, the only words that can have an accepting run in $C$ must be in $R$, and the weight of these words will be same as that in $A$. Therefore, if $w \in R$, then $[\![C]\!](w) = [\![A]\!](w)$, otherwise $[\![C]\!](w) = 0$.

It is trivial to see that if $A$ is sequential, then $C$ would be sequential. □

## 4.3 Putting it all together...

First, we prove the following theorem using the lemmas that we discussed in subsection 4.1 and 4.2.

**Lemma 4.9.** *If the function $f \in FSEQ \cap UNAMB$, then there exists a automaton $C$ which is both finitely sequential and unambiguous.*

*Proof.* Let $A = \bigcup_{i=1}^{k} A_i$ and $B$ be finitely sequential and unambiguous weighted automata computing the function $f$. Since, $[\![B]\!](w) = \max_i ([\![A_i]\!](w))$, we have $[\![A_i]\!] \leq [\![B]\!]$. From Corollary 4.3.1, there exists an unambiguous automata $D_i$ over $\mathbb{N}_{\max}$ which computes $[\![B]\!] - [\![A_i]\!]$. Let $R_i = \{w \mid [\![B]\!](w) = [\![A_i]\!](w)\} = \{w \mid [\![D]\!](w) = 0\}$ Now, applying the Lemma 4.7, the set $R_i$ is regular. Let $R'_i = R_i \setminus \bigcup_{j>i} R_j$. Using Lemma 4.8, let $A'_i$ be a sequential automaton such that $[\![A'_i]\!](w) = [\![A_i]\!](w)$ if $w \in R'_i$, else $-\infty$. Consider the finitely sequential automaton $A' = \bigcup_{i=k}^{n}(A'_i)$.

We first show that $A'$ is unambiguous. Let us suppose, for the sake of contradiction, that there is a word $w$ that is accepted by both $A'_i$ and $A'_j$ for some $i < j \in [1, k]$, then $w \in R'_i$ and $w \in R'_j$, but $w \in R_i \implies w \in R_i$ and $w \notin R_j$, which is a contradiction.

Let $w$ by any word. We know that $[\![A]\!](w) = [\![B]\!](w)$. Since $A$ is finite union of sequential automata $A_i$, $w$ must be accepted by some $A_i$ with weight $[\![B]\!](w)$. Let $j$ be the largest index for which $[\![A_j]\!](w) = [\![B]\!](w)$. Therefore, $[\![A_l]\!](w) \neq [\![B]\!](w)$, or $w \notin R_l$ for all $l > j$. Since $R'_j = R_j \setminus \bigcup_{l>j} R_l$, by definition, $w \in R'_j$, hence, $[\![A']\!](w) = [\![A'_j]\!](w) = [\![A_j]\!](w) = [\![A]\!](w)$. For the other direction, if $w$ is accepted by $A'$, then it must be accepted by some $A'_j$, and hence by $A_j$ with $[\![A_j]\!](w) = [\![A]\!](w)$, or $[\![A']\!](w) = [\![A'_j]\!](w) = [\![A_j]\!](w) = [\![A]\!](w)$. Therefore, $[\![A']\!] = [\![A]\!]$. □

Now, using the Lemma 4.3, we shall finally prove the Theorem 4.1.

*Proof of Theorem 4.1.* ($\Rightarrow$) Let $f \in \mathrm{LCRA}^c(\odot)_1$, and let $\mathcal{A}$ be the CRA computing $f$ such that every update matrix is diagonal and there is at most one non-zero entry in $\mu(q)$, for every $q \in Q$. As $\mathcal{A}$ is copyless, thus $f \in FSEQ$. Moreover, since there is at most one non-zero entry in $\mu(q)$, for every $q \in Q$, and $\mathcal{A}$ is plusfree, we have $f \in UNAMB$. Therefore, $f \in FSEQ \cap UNAMB$.

($\Leftarrow$) Let $f \in FSEQ \cap UNAMB$. From Lemma 4.3, there exists an weighted automata $A$ that computes $f$ and is both finitely sequential and unambiguous. Let $A = \bigcup_{i\in[1,k]} A_i$, where each $A_i = (Q_i, q_{0i}, \lambda_i, \delta_i, \gamma_i)$ is sequential. Let $Q' = Q_1 \times Q_2 \ldots \times Q_k$. Using the construction described in Lemma 3.4, we construct an equivalent LCRA $\mathcal{A} = (Q, \mathcal{X}, q_0, \delta, \nu_0, \mu)$ such that each update matrix is diagonal. Let $\mathcal{A}' = (Q', \mathcal{X}, q_0, \delta, \nu_0, \mu)$ be the LCRA obtained by removing all the states in $\mathcal{A}$ which are *not* useful.

Suppose, for the sake of contradiction, that there exists a state $q$ such that $\mu(q)$ has more than one nonzero entry. Let $x_i, x_j \in \mathcal{X}$ such that $\mu(q)(x_i), \mu(q)(x_j) \neq -\infty$. By construction, after reading a word $w$ in $\mathcal{A}'$, $x_i$ and $x_j$ hold the running weight of run of $A_i$ and $A_j$ over $w$, respectively. Moreover, for any state $(q_1, q_2, \ldots, q_k) \in Q'$, $\mu(q)(x_i)$ is nonzero iff $q_i$ is a final state of $A_i$. Hence, there exists a word $w$ such that both $A_i$ and $A_j$ have a accepting run over $w$, which contradicts the fact that $A$ is unambiguous. Hence, $f \in \mathrm{LCRA}^c(\odot)_1$.; □

# 5 Conclusion

In this thesis, we discussed various classes of LCRA and their relation to the classes of weighted automata. In particular, we showed that, over all semirings, the copyless LCRA with $k$ registers compute the class $kSEQ$, plusfree LCRA compute the class which is subset of $FAMB$, and plusfree LCRA with further restriction on final expression is equivalent to $UNAMB$. We also discussed a few results which hold over tropical semirings.

We believe that $kAMB = \text{LCRA}(\odot)_k$ for idempotent semirings. The proof requires us to decompose a $kAMB$ WA into $k$ unambiguous WA, but the formal proof is left for further work.

A natural direction for future work is to find different restrictions on update matrices which give robust and meaningful classes of functions. Pumping Lemmas for various general classes of functions definable via WA are discussed in [11]. As LCRA are deterministic, another direction could be to come up with different versions of the pumping lemmas for various subclasses of LCRA that we discussed, which are, in some way, easier to apply and reason about than their WA counterparts.

The LCRA with resets are also equally expressive as WA. A third direction could be to study the structural restrictions on LCRA with reset.

# References


[1] M.P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2):245–270, 1961.

[2] Shaull Almagor, Udi Boker, and Orna Kupferman. What's decidable about weighted automata? In Tevfik Bultan and Pao-Ann Hsiung, editors, *Automated Technology for Verification and Analysis*, pages 482–491, Berlin, Heidelberg, 2011.

[3] Jürgen Albert and Jarkko Kari. Digital image compression. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, pages 453–479. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[4] Kevin Knight and Jonathan May. Applications of weighted automata in natural language processing. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, pages 571–596. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[5] Rajeev Alur, Loris DAntoni, Jyotirmoy Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 13–22, 2013.

[6] Filip Mazowiecki and Cristian Riveros. Copyless cost-register automata: Structure, expressiveness, and closure properties. *Journal of Computer and System Sciences*, 100:1–29, 2019.

[7] Shaull Almagor, Michaël Cadilhac, Filip Mazowiecki, and Guillermo Pérez. Weak cost register automata are still powerful. *International Journal of Foundations of Computer Science*, 31, 04 2018.

[8] Rajeev Alur and Mukund Raghothaman. Decision problems for additive regular functions. In Fedor V. Fomin, Rūsiņš Freivalds, Marta Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming*, pages 37–48, Berlin, Heidelberg, 2013.

[9] Laure Daviaud. Register complexity and determinisation of max-plus automata. *ACM SIGLOG News*, 7(2):4–14, April 2020.

[10] Manfred Droste and Paul Gastin. Weighted automata and weighted logics. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming*, pages 513–525, Berlin, Heidelberg, 2005.

[11] Agnishom Chattopadhyay, Filip Mazowiecki, Anca Muscholl, and Cristian Riveros. Pumping lemmas for weighted automata, 2021.