Bachelor thesis
Computer Science


Radboud University

# Transducers over G-sets and their properties

*Author:*
Bas Hofmans
s4204336

*Supervisor/First assessor:*
Dr. J. C. Rot
jrot@cs.ru.nl

*Second assessor:*
Prof. Dr. J. H. Geuvers
h.geuvers@cs.ru.nl

23rd January 2019

**Abstract**

In this thesis we want to show and explore the possibilities and properties of automata and, in particular, transducers, over G-sets. Until now no theory has been available for such transducers. This means that it has not been possible to assess which properties these kinds of transducers might have. In this thesis we introduce a definition of transducers over G-sets and then analyse their properties. We find that transducers can be normalised showing that the length of transitions the transducer has is not important, and that the set of languages recognised by G-NFAs is closed under transduction.

# Contents

# 1 Introduction

Transducers provide a huge amount of possibilities for manipulating languages. They allow us to transform words and languages into other words and language with very simple operations. Classically, these transducers are modelled as a variation of a finite automaton and are therefore limited in the same way that classic automata are. This means that they cannot be applied to any kind of infinite alphabets.

Bojańczyk, Klin and Lasota, 2014 describes theory for automata over G-sets and nominal sets. These automata are able to work with G-sets allowing infinite alphabets, state spaces, and transition relations. Adding structure to infinite sets allows these automata to be described and represented finitely, allowing us to work with these infinite structures intuitively and elegantly. While they discuss a lot of theory about and properties of these kinds of automata this theory is not extended to providing theory for transducers over these same sets.

In this thesis we add to the theory by introducing transducers over G-sets in a way such that they relate closely to non-deterministic automata over G-sets. Such G-transducers are then able to compute transductions from words and languages over one G-set alphabet to another. In particular we provide a formal definition of this kind of transducer. We then analyse some of the characteristics of these transducers where we specifically look at the closure properties of the languages recognised by non-deterministic automata over G-sets and show that these are indeed closed under the application of transduction.

Section 2 discusses some of the preliminaries relevant to this thesis including theory about classic automata and transducers. Section 3 then discusses existing theory about both deterministic and non-deterministic automata over G-sets. Following this section 4 provides a definition of G-transducers such that they can be constructed and provides a method of normalising these transducers. Section 6 then discusses the closure properties of non-deterministic automata over G-sets and shows these are closed under G-transduction. Section 7 then discusses related work and concludes.

# 2 Preliminaries

In this section we present the preliminaries for this thesis. This starts with a short description of the notation used for formal languages, this is followed by a description of certain types of classic automata. We discuss both deterministic and non-deterministic finite automata as presented, among many others, in Hopcroft, Motwani and Ullman, 2006. Then we discuss the classic finite transducers as they are described in Shallit, 2008.

## 2.1 Languages

We give a short recap of the notation that is used in this thesis for languages and related matters. We describe languages as a set of words over an alphabet. Given an alphabet $\Sigma$ we define the set of all words $\Sigma^*$ as defined by the Kleene star such that $\Sigma^*$ contains all finite-length words consisting of arbitrary elements from $\Sigma$ including the empty word given by $\varepsilon$. A language over $\Sigma$ is then given by a subset of all words $L \subseteq \Sigma^*$.

We also define a function which can be applied to (regular)expressions to retrieve the language generated by that (regular)expression. For any regular expression $r$ the language generated by that regular expression is given by $[\![r]\!] \subseteq \Sigma^*$. In the same fashion we define a function which is applied to automata to retrieve the language recognised by that automaton. For an automaton $A$ the language recognised by that automaton is given by $[\![A]\!] \subseteq \Sigma^*$.

For sets in general, we use the notation $\binom{S}{n}$ for the set of subsets of a set $S$ containing $n$ elements. Similarly, the notation $S^n$ is used to represent the set of all $n$-tuples consisting of elements of $S$.

## 2.2 Classic automata

Here we will briefly consider two classic types of automata over finite alphabets, deterministic finite automata (DFAs) and non-deterministic finite automata (NFAs). These provide the basis for our later consideration of DFAs and NFAs over G-sets. We then consider the classic notion of transducer over finite alphabets.

In general automata are machines consisting of states and transitions which recognise languages over finite alphabets by recognising the words in them. All these automata read words by taking transitions from one state to another dependant on the next letter read from the inputted word.

In the case of these classic automata, deterministic and non-deterministic automata are able to recognise the same set of languages. These are also exactly those languages for which a regular expression exists and these are appropriately named the regular languages.

### 2.2.1 Deterministic finite automata

**Definition 2.1.** A deterministic finite automaton (DFA) is a tuple $A = (Q, \Sigma, I, F, \delta)$ consisting of:

- A finite set of states $Q$,
- A finite input alphabet $\Sigma$,
- An initial state $q_0$,
- A finite set of final states $F \subseteq Q$,
- A transition function $\delta : Q \times \Sigma \to Q$.

The transition functions defines the transitions present in the automaton, if $\delta(q, a) = q'$ then there is a transition from $q$ to $q'$ along which $a$ is read in the automaton. This transition function thus represents the possibilities to take single steps in the automaton. We extend this to a multi-step transition function $\delta^* : Q \times \Sigma^* \to Q$ which tells us in which state we end up if a word is read from some state. We define it as:

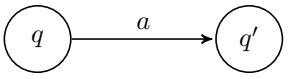
$$\delta^*(q, \varepsilon) = q \qquad \delta^*(q, ax) = \delta^*(\delta(q, a), x)$$

with $a \in \Sigma$ and $x \in \Sigma^*$. The automaton then recognises those words which end in final state when read from the initial state $q_0$. This means that the language of an automaton $A$ is given by:

$$[\![A]\!] = \{w \mid \delta^*(q_0, w) \in F\}$$

Like other automata DFAs recognise words by reading them from the input. More precisely a DFA reads words from input and takes a transition from one state to another whenever a letter from the input is read. This process starts in some initial state and a word is recognised by the automaton if this process ends in a final state when the word is completely read.

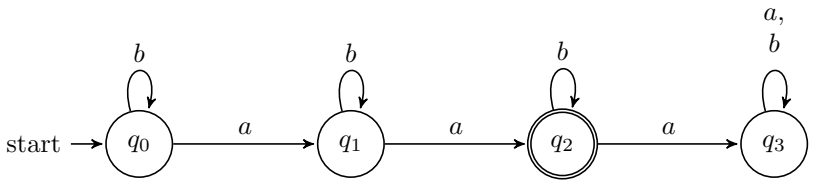
Such automata can be represented graphically, a transition from some state $q$ to another state $q'$ along which the letter $a$ is read would be represented by:




States which are initial are indicated with a start arrow, final states are indicated with double border lines, there are represented by:


In which state $q$ is initial and state $r$ is final.

**Example 2.1.** We give an example of a simple DFA $A$ over the alphabet $\{a, b\}$ the goal of this automaton is to recognise any word which contains the letter $a$ exactly two times. This automaton is given by:




It is easy to recognise that the letter $a$ needs to occur exactly twice in a word for the word to be recognised by this automaton. If it occurs fewer, the only accepting state, state $q_2$ is never reached. If it contains occurs more then the computation will be stuck in state $q_3$ which is not accepting. Thus the automaton does what it is designed to do: $[\![A]\!] = \{w \in \Sigma^* \mid a \text{ occurs twice in } w\}$.

### 2.2.2 Non-deterministic finite automata

Non-deterministic finite automata (NFA) resemble DFAs and are both defined and represented very similarly. The only addition is that these automata allow non-determinism in their transitions. This means that it is allowed to have any amount of possible transitions from a state when a certain letter is read.

**Definition 2.2.** A non-deterministic finite automaton (NFA) is a tuple $A = (Q, \Sigma, I, F, \delta)$ consisting of:

- A finite set of states $Q$,
- A finite input alphabet $\Sigma$,
- A finite set of initial states $I \subseteq Q$,
- A finite set of final states $F \subseteq Q$,
- A transition relation $\delta \subseteq Q \times \Sigma \times Q$.

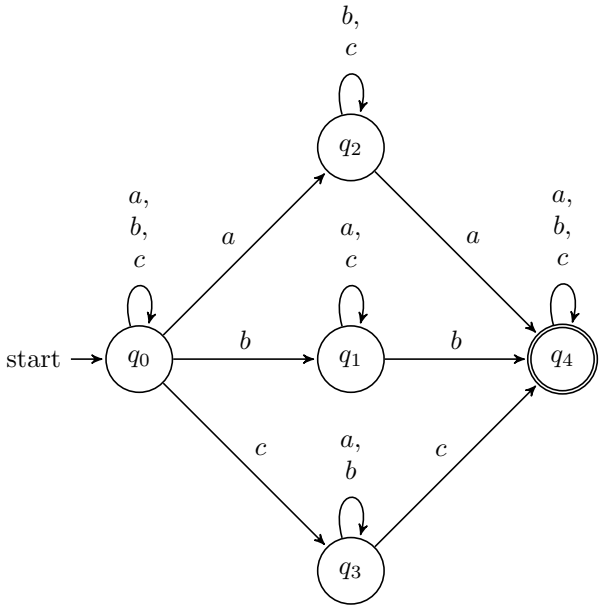
In this case a tuple $(q, a, q')$ is in the transition relation $\delta$ if there is a transition from $q$ to $q'$ in the automaton along which $a$ is read. Once again this relation can be extended to a multi-step transition relation $\delta^* \subseteq Q \times \Sigma^* \times Q$ as the least relation such that:

$$\frac{(q, w, q') \in \delta}{(q, w, q') \in \delta^*} \qquad \frac{(q, v, q') \in \delta^* \quad (q'', w, q''') \in \delta^* \quad q' = q''}{(q, vw, q''') \in \delta^*}$$

A word is then recognised by the automata if there is a possible path from an initial state to a final state along which the word is read. The language of a NFA $A$ is then given by:

$$[\![A]\!] = \begin{array}{l} \{w \mid (q, w, q') \in \delta^* \text{ for some } q \in I \text{ and } q' \in F\} \\ \cup \{\varepsilon \mid \text{ there is } q \text{ such that } q \in I \text{ and } q \in F\} \end{array}$$

**Example 2.2.** We give an example of a simple NFA. The non-determinism allowed in NFAs allows them to recognise certain language more intuitively than DFAs can. While every language recognised by a NFA can also be recognised by a DFA making such an automaton is much easier when using NFAs. Consider a language over the alphabet $\{a, b, c\}$ which contains all words in which some letter occurs more than once. A NFA which recognises this language is given by:




This recognises the right language because whenever a letter occurs at least two times in a word the automaton can loop in $q_0$ until the first occurrence of the word and take the transition to another state at this first occurrence. The non-determinism is useful in this case because it allows the automaton to stay in state $q_0$ until the letter it knows to occur twice occurs for the first time.

### 2.2.3 Transducers

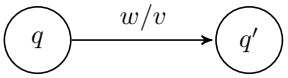
Transducers are a variation of non-deterministic automata. Rather than recognising whether a word is contained in the language of the automata, transducers write output when reading input and effectively transform words and languages. Here we present a definition of transducers closely following the definition presented in Shallit, 2008.

**Definition 2.3.** A *transducer* is a tuple $T = (Q, \Sigma, \Gamma, I, F, \delta)$ consisting of:

- A finite set of states $Q$,

- A finite input alphabet $\Sigma$,
- A finite output alphabet $\Gamma$,
- A finite set of initial states $I \subseteq Q$,
- A finite set of final states $F \subseteq Q$,
- A finite transition relation $\delta \subseteq Q \times \Sigma^* \times \Gamma^* \times Q$.

Transitions in $\delta$ are tuples $(q, w, v, q')$ such that there is a transition from state $q$ to state $q'$ along which $w$ is read from the input and $v$ is read from the output. Such a transition of a transducer is represented by:




The single-step transition relation $\delta$ can then be extended to a multi-step transition relation $\delta^*$ by making $\delta^*$ the least relation such that:

$$\frac{(q, w, v, q') \in \delta}{(q, w, v, q') \in \delta^*}$$

$$\frac{(q, w, v, q') \in \delta^* \quad (q'', x, y, q''') \in \delta^* \quad q' = q''}{(q, wx, vy, q''') \in \delta^*}$$

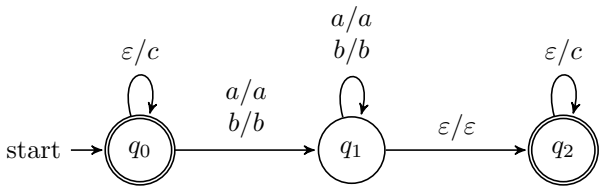
We can then say that the transducer computes a transduction $T$ which we define as a function $T : \Sigma^* \to \mathcal{P}(\Gamma)$ over words $w \in \Sigma^*$ which is given by:

$$T(w) = \begin{array}{l} \{v \mid (q, w, v, q') \in \delta^* \text{ for some } q \in I \text{ and } q' \in F\} \\ \cup \{\varepsilon \mid w = \varepsilon \text{ and there is } q \in Q \text{ such that } q \in I \text{ and } q \in F\} \end{array}$$

We then extend this to a transduction $\bar{T}$ for languages over $\Sigma$ which we can also define as a function $\bar{T} : \mathcal{P}(\Sigma) \to \mathcal{P}(\Gamma)$ which for a language $L \in \mathcal{P}(\Sigma)$ is given by:

$$\bar{T} = \bigcup_{w \in L} T(w)$$

**Example 2.3.** Let us consider a basic transducer over input alphabet $\Sigma = \{a, b\}$ and output alphabet $\Gamma = \{a, b, c\}$ the goal of which is to add a number of $c$'s in front of a word and after a word. For this purpose we make the following transducer:



Then this transducer achieves the goal as the transducer can loop in the initial state while printing $c$'s without reading any input, then has to read the entire word before going to a new accepting state without printing or reading anything where it can once again print any amount of $c$'s without reading input.

# 3 G-Automata

In this section we illustrate the existing theory used to describe G-sets and how they functions as well as the theory of automata over G-sets. We do this following the way in which it is presented in Bojańczyk et al., 2014.

## 3.1 G-sets

We are going to consider languages and automata over possibly infinite alphabets $\Sigma$ involving some countably infinite set of data values $\mathbb{D}$. Values in the collection of data atoms cannot be accessed directly by an automaton, they can be accessed through some structure imposed on the collection of data values. We consider the structure of equality of the set of data atoms such that for any two elements $d, e \in \mathbb{D}$ we can evaluate whether $d = e$ or $d \neq e$. This means that the only thing any automaton can do with atoms is compare whether are they are equal or not.

The set of data values $\mathbb{D}$ is then considered together with a group of bijections over $\mathbb{D}$ called $G$. Here, $G$ is the set of all bijections over $\mathbb{D}$ that is given by:

$$G = \{\pi : \mathbb{D} \to \mathbb{D} \mid \pi \text{ is a bijection}\}$$

paired with the binary operation (written infix) $\cdot : G \times G \to G$ given by composition to combine elements of $G$. Composition works such that for all bijections $f, g \in G$ the composition of two bijections applied to some value $x \in \mathbb{D}$ is $(f \cdot g)(x) = f(g(x))$. This satisfies all the necessary group axioms. It satisfies closure as the composition of two bijections is still a bijection. Furthermore it is associative as function composition is associative. It has a neutral element $e$ in the identity function. Every bijection $\pi$ has an inverse bijection $\pi^{-1}$ such that $\pi \cdot \pi^{-1} = e$. A pair of a set of data atoms paired with a certain group is called a data symmetry. Our imposed structure of equality on the set of data atoms and the choice of the group of all bijections over $\mathbb{D}$ means that we operate in the *equality* symmetry, other symmetries are possible but will not be considered in this thesis, see Bojańczyk et al., 2014 for more discussion on this subject.

### 3.1.1 Group action

We define a (right) group action which is a function $\cdot : X \times G \to X$, a group action must satisfy two axioms namely that for all elements $x \in X$, $\pi, \sigma \in G$, and the neutral element $e$ of $G$:

$$x \cdot e = x \qquad x \cdot (\pi\sigma) = (x \cdot \pi) \cdot \sigma$$

A set $X$ paired with an action is called a G-set. In this thesis we will consider the following actions:

- For any $x \in \mathbb{D}$ and $\pi \in G$, $x \cdot \pi = \pi(x)$,
- For elements of the powerset of a G-set $T$, $S \in \mathcal{P}(T)$ it is defined as $S \cdot \pi = \{x \cdot \pi \mid x \in S\}$,
- For product types over G-sets the action is applied to the elements,
- On lists of elements of a G-set, such as words over alphabets, the action is applied to the elements of the list,
- For elements of all sets $S$ not covered by any of these the action functions as the trivial action such that for all $x \in S$ and $\pi \in G$ it is defined as: $x \cdot \pi = x$.

### 3.1.2 Orbits

When considering automata over infinite alphabets we will consider a type of finiteness called orbit-finiteness. The orbit of an element $x$ for some G-set $X$ is the set of all elements $x$ maps to when the group action is applied to it with some element $\pi \in G$. Precisely this means the orbit of $x$ is given by:

$$\{x \cdot \pi \mid \pi \in G\}$$

This also means that it is often the case that elements have the same orbit. We consider the amount of orbits a certain set has to be the amount of different orbit its elements have. Take as an example the set of data values $\mathbb{D}$, this is a single-orbit set as each of its elements will have the same orbit as each $x \in \mathbb{D}$ maps to the same set of elements when all bijections over $\mathbb{D}$ are applied to it. A set is then considered to be orbit-finite if the amount of orbits it has is finite.

### 3.1.3 Equivariant subsets, relations, and functions

We introduce the concept of equivariance on sets, relations, and functions. A subset of a G-set or a relation over a G-set is considered to be equivariant if it is closed under application of an arbitrary bijection from $G$ by the group action. Intuitively this means that this set treats each data value equally with respect to equality of values or that if the set contains a certain data values in some way it contains all other data values in that same way as well.

We can give a more exact definition of this. Suppose X is a G-set, a subset $Y \subseteq X$ is then equivariant if the application of the group action leaves the set intact for all permutations $\pi \in G$, meaning that $Y \cdot \pi = Y$. Extending this to relations gives that a relation $R \subseteq X \times Y$ is equivariant if it is left intact by point-wise application of the group action on the Cartesian product $X \times Y$, that is, if $(x, y) \in R$ implies $(x \cdot \pi, y \cdot \pi) \in R$. A function is considered to be equivariant if the relation it corresponds to is equivariant. This means that a function $f$ is equivariant if $f(x) \cdot \pi = f(x \cdot \pi)$.

We will now introduce a number of lemmas with regard to equivariance which will later be used in the main proofs of the thesis.

**Lemma 3.1.** *If sets $S \subseteq X$ and $T \subseteq Y$ are equivariant subsets of G-sets $X$ and $Y$, then the Cartesian product $S \times T \subseteq X \times Y$ is equivariant as well.*

*Proof.* In order to show $S \times T$ equivariant we need to prove that $(S \times T) \cdot \pi = S \times T$ for all $\pi \in G$, we know $S \times T = \{(s,t) \mid s \in S, t \in T)\}$.

We then have:

$$\begin{aligned}(S \times T) \cdot \pi &= \{(s,t) \cdot \pi \mid s \in S, t \in T\} \\ &= \{(s \cdot \pi, t \cdot \pi) \mid s \in S, t \in T\} \\ &= \{(s,t) \mid s \in S \cdot \pi, t \in T \cdot \pi\} \\ &= \{(s,t) \mid s \in S, t \in T\} \qquad \text{(by equivariance of } S \text{ and } T)\end{aligned}$$

As this is equal to $S \times T$ we know that $S \times T$ is equivariant. □

The next lemma shows that the image of application of an equivariant function with respect to an equivariant subset of a G-set is equivariant.

**Lemma 3.2.** *Given an equivariant function $f$ and an equivariant subset of some G-set $S$ the image of $f$ applied to all elements of $S$, $\{f(s) \mid s \in S\}$ is equivariant.*

*Proof.* Suppose we have a set $S$ and a function $f$ such that $S$ and $f$ are equivariant. Then the image of $f$ with respect to $S$ is given by

$$\{f(s) \mid s \in S\}$$

To show this is equivariant we must show that

$$\{f(s) \mid s \in S\} \cdot \pi = \{f(s) \mid s \in S\}$$

holds for all $\pi \in G$, this holds as:

$$\begin{aligned}\{f(s) \mid s \in S\} \cdot \pi &= \{f(s) \cdot \pi \mid s \in S\} \\ &= \{f(s \cdot \pi) \mid s \in S\} \qquad \text{(by equivariance of } f) \\ &= \{f(s) \mid s \in S \cdot \pi\} \\ &= \{f(s) \mid s \in S\} \qquad \text{(by equivariance of } S)\end{aligned}$$

Which is what we had to show. □

**Lemma 3.3.** *The union of the elements of an equivariant set of sets is equivariant.*

*Proof.* Let $S$ be an equivariant set of sets, then the union of its elements is given by:

$$\bigcup_{s \in S} s$$

To show this is equivariant we need to show that:

$$(\bigcup_{s \in S} s) \cdot \pi = \bigcup_{s \in S} s$$

For all $\pi \in G$, this follows from:

$$\begin{aligned}(\bigcup_{s \in S} s) \cdot \pi &= \bigcup_{s \in S} (s \cdot \pi) \\ &= \bigcup_{s \in S \cdot \pi} s \\ &= \bigcup_{s \in S} s && \text{(by equivariance of S)}\end{aligned}$$

Which shows that the union of the elements of $S$ is equivariant. □

## 3.2 G-Automata

G-Automata are, as their name suggests, automata over G-sets. They clearly distinguish themselves from classic automata in that they allow infinite alphabets as long as they are G-sets and allow for an orbit-finite state space instead of a finite one.

### 3.2.1 Deterministic G-Automata

**Definition 3.1.** We define a deterministic G-Automata over a (possibly infinite) alphabet $\Sigma$ which needs to be a G-set as a tuple $(Q, \Sigma, \delta, I, F)$ with:

- An orbit-finite G-set of states $Q$,
- A G-set $\Sigma$ containing the alphabet,
- An equivariant single-step transition function $\delta\colon Q \times \Sigma \to Q$,
- An equivariant singleton set $I \subseteq Q$ containing initial state $q_i \in Q$,
- An equivariant set of accepting states $F \subseteq Q$.

We extend the single-step transition function $\delta$ to a multi-step transition function $\delta^* : Q \times \Sigma^* \to Q$ where $(q, \varepsilon) \mapsto q$ and for $w \neq \varepsilon$ we have $(q_0, w) \mapsto q_n$ if and only if there is a sequence $(q_0, a_0), (q_1, a_1), \ldots, (q_n, a_n)$ such that $w = a_0 a_1 a_2 \ldots a_n$ and $\delta(q_i, a_i) = q_{i+1}$ for all $i \in \mathbb{N}$ with $0 \leq i < n$.

A word $w$ is then accepted by the automaton if and only if $\delta^*(q_0, w) = q_f$ for some $q_f \in F$ or if $w = \varepsilon$ and $q_0 \in F$. The language of a given automaton is then given by:

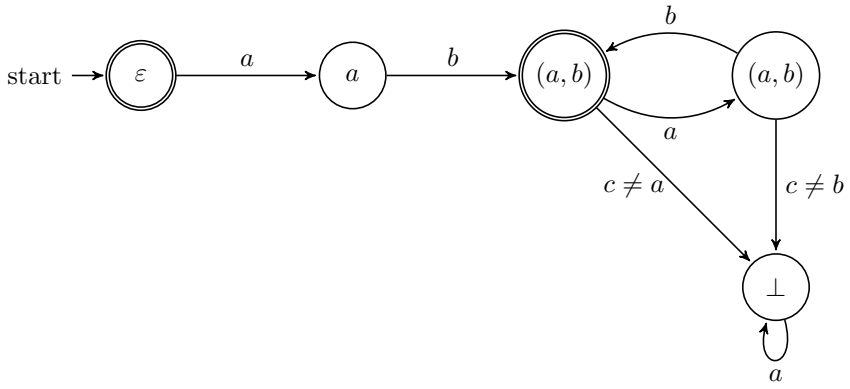
$$\{w \mid \delta^*(q_0, w) \in F\}$$

.

**Example 3.1.** Let us consider an automaton for a language $L$ in the equality symmetry over the set of atoms $\mathbb{D}$. let $L$ be defined by:

$$L = \bigcup_{a,b \in \mathbb{D}} [\![(ab)^*]\!]$$

Then an automaton recognising $L$ is given by:




Then the set of states $Q$ is given by:

- The initial state given by $\varepsilon$,
- The set of states in which the first letter is stored given by a with $a \in \mathbb{D}$,
- The set of states in which two letters have been stored and where the word should be accepted given by $(\mathbb{D}^2, 1)$,
- The set of states in which two letters have been stored and where the word should not be accepted given by $(\mathbb{D}^2, 0)$,
- The sink state given by $\perp$.

Thus $Q$ is defined by

$$Q = \{\varepsilon, \perp\} \cup \mathbb{D} \cup (\mathbb{D}^2, 1) \cup (\mathbb{D}^2, 0)$$

This set of states has two singleton orbits $\varepsilon$ and $\perp$ and three infinite orbits $\mathbb{D}$, $(\mathbb{D}^2, 1)$, $(\mathbb{D}^2, 0)$. The initial state $q_i$ is then given by $\varepsilon$ and the set of final states F given by:

$$F = \{\varepsilon\} \cup (\mathbb{D}^2, 1)$$

The transition function $\delta$ is then given by:

$$\delta(\varepsilon, a) = a$$

$$\delta(a, b) = ((a, b), 1)$$

$$\delta(((a, b), 1), c) = \begin{cases} ((a, b), 0) \text{ if } c = a \\ \bot \text{ otherwise} \end{cases}$$

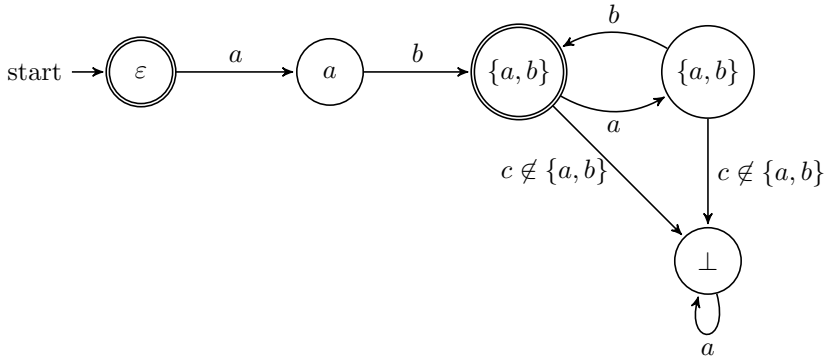
$$\delta(((a, b), 0), c) = \begin{cases} ((a, b), 1) \text{ if } c = b \\ \bot \text{ otherwise} \end{cases}$$

$$\delta(\bot, a) = \bot$$

**Example 3.2.** Let us consider an automaton for a language $L$ over the alphabet $\Sigma$ given by the set of atoms $\mathbb{D}$. let $L$ be defined as:

$$L = \bigcup_{a,b \in \mathbb{D}} [\![(1 + (ab(aa + ab + ba + bb)^*)]\!]$$

Then an automaton recognising $L$ is given by:




Then the set of states $Q$ is given by:

- The initial state given by $\varepsilon$
- The set of states in which the first letter is stored given by $a$ with $a \in \mathbb{D}$
- The set of states in which two letters have been stored and where the word should be accepted given by $(\binom{\mathbb{D}}{2}, 1)$ if the first two letters are not equal and by $(\binom{\mathbb{D}}{1}, 1)$ if they are.
- The set of states in which two letters have been stored in where the word should not be accepted give by $(\binom{\mathbb{D}}{2}, 0)$ if the first two letters are not equal and by $(\binom{\mathbb{D}}{1}, 0)$ if they are.
- The sink state given by $\bot$

Thus $Q$ is defined by

$$Q = \{\varepsilon, \bot\} \cup \mathbb{D} \cup \left(\binom{\mathbb{D}}{2}, 1\right) \cup \left(\binom{\mathbb{D}}{1}, 1\right) \cup \left(\binom{\mathbb{D}}{2}, 0\right) \left(\binom{\mathbb{D}}{1}, 0\right)$$

This set of states has two singleton orbits $\varepsilon$ and $\bot$ and five infinite orbits $\mathbb{D}$, $(\binom{\mathbb{D}}{2}, 1)$, $(\binom{\mathbb{D}}{1}, 1)$, $(\binom{\mathbb{D}}{2}, 0)$, and $(\binom{\mathbb{D}}{1}, 0)$.

With the initial state $q_0$ given by $\varepsilon$ and the set of final states F given by:

$$F = \{\varepsilon\} \cup \left(\binom{\mathbb{D}}{2}, 1\right) \cup \left(\binom{\mathbb{D}}{1}, 1\right)$$

The transition function $\delta$ is then given by:

$$\delta(\varepsilon, a) = a$$

$$\delta(a, b) = \begin{cases} (\{a, b\}, 1) \text{ if } a \neq b \\ (\{a\}, 1) \text{ otherwise} \end{cases}$$

$$\delta((\{a, b\}, 1), c) = \begin{cases} (\{a, b\}, 0) \text{ if } c \in \{a, b\} \\ \bot \text{ otherwise} \end{cases}$$

$$\delta((\{a\}, 1), b) = \begin{cases} (\{a\}, 0) \text{ if } b = a \\ \bot \text{ otherwise} \end{cases}$$

$$\delta((\{a, b\}, 0), c) = \begin{cases} (\{a, b\}, 1) \text{ if } c \in \{a, b\} \\ \bot \text{ otherwise} \end{cases}$$

$$\delta((\{a\}, 0), b) = \begin{cases} (\{a\}, 1) \text{ if } b = a \\ \bot \text{ otherwise} \end{cases}$$

$$\delta(\bot, a) = \bot$$

### 3.2.2 Non-deterministic G-automata

Like deterministic G-automata we can define non-deterministic G-automata which allow non-determinism in the transitions in a similar fashion.

**Definition 3.2.** A non-deterministic G-automata is defined by a tuple $(Q, \Sigma, \delta, I, F)$ consisting of:

- An orbit-finite G-set of states $Q$,
- A G-set $\Sigma$ containing the alphabet,
- An equivariant transition relation $\delta \subseteq Q \times \mathbb{D} \times Q$,

- An equivariant orbit finite set of initial states $I \subseteq Q$,
- An equivariant orbit finite set of accepting states $F \subseteq Q$.

We can now extend the transition relation $\delta$ to a multi-step transition relation $\delta^* \subseteq Q \times \Sigma^* \times Q$ which is defined as the least relation such that:

$$\frac{(q, w, q') \in \delta}{(q, w, q') \in \delta^*}$$

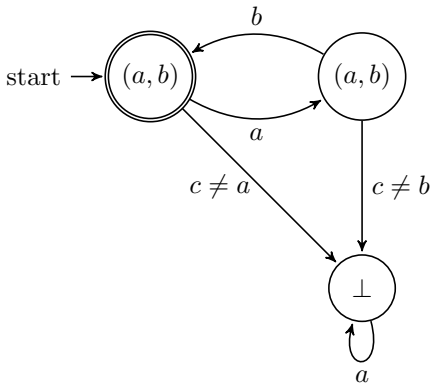
$$\frac{(q, w, q') \in \delta \quad (q'', v, q''') \in \delta \quad q' = q''}{(q, wv, q''') \in \delta^*}$$

A word $w$ is then accepted by a given automaton $A = (Q, \delta, I, F)$ if and only if $(q, w, q') \in \delta^*$ for some $q \in I$ and $q' \in F$ or if $w = \varepsilon$ and there is some $q \in Q$ for which both $q \in I$ and $q \in F$ hold. Equivalently a word is accepted if a sequence $(q_0, a_0, q_1), \ldots, (q_{n-1}, a_{n-1}, q_n)$ exists such that $w = a_0 a_1 \ldots a_{n-1}$, that for all tuples $(q_i, a_i, q_{i+1})$ it holds that $(q_i, a_i, q_{i+1}) \in \delta$, that $q_0 \in I$, and that $q_n \in F$. Intuitively this means that a words is accepted if there is a path from an initial state to a final state along which $w$ is read.

**Example 3.3.** Consider the language from example 3.1 given by

$$L = \bigcup_{a,b \in \mathbb{D}} [\![(ab)^*]\!]$$

Using non-determinism the automaton given in example 3.1 can be simplified to the following:




The set of states $Q$ is then given by:

- A set of accepting states in which two characters are being stored given by: $(\mathbb{D}^2, 1)$,
- A set of non-accepting states in which two characters are being stored given by: $(\mathbb{D}^2, 0)$,

- A sink state given by: $\perp$.

Thus:

$$Q = \{\perp\} \cup (\mathbb{D}^2, 1) \cup (\mathbb{D}^2, 0)$$

For this automaton the set of initial states $I$ is then equal to the set of accepting states $F$ and is given by the orbit-finite set of states $(\mathbb{D}^2, 1)$.

The transition relation for the automaton consists of

$$\begin{aligned}\delta = &\{(((a,b),1),a,((a,b),0)) \mid a,b \in \mathbb{D}\} \\ &\cup \{(((a,b),1),c,\perp) \mid a,b,c \in \mathbb{D} \wedge c \neq a\} \\ &\cup \{(((a,b),0),b,((a,b),1)) \mid a,b \in \mathbb{D}\} \\ &\cup \{(((a,b),0),c,\perp) \mid a,b,c \in \mathbb{D} \wedge c \neq b\} \\ &\cup \{(\perp,a,\perp) \mid a \in \mathbb{D}\}\end{aligned}$$

### 3.2.3 G-NFAs and $\varepsilon$ transitions.

Note that in these G-NFAs it is not possible to take $\varepsilon$-transitions, that is, transitions in which no letter from the inputted word is read. We can define G-NFAs in which this is allowed as G-NFA-$\varepsilon$. A G-NFA-$\varepsilon$ is a G-NFA in which the transition relation $\delta$ becomes a relation over $Q \times (\Sigma \cup \{\varepsilon\}) \times Q$. A $\delta^*$ can be constructed from this $\delta$ in the same way it normally would be constructed for a G-NFA.

The distinction between G-NFA which allows $\varepsilon$ transitions and those which do not is not very meaningful as they recognise the same set of languages. To show this we show how to construct a G-NFA which recognises the same language as any given G-NFA-$\varepsilon$. Note that this is the same method that is used to turn a regular NFA-$\varepsilon$ into a NFA. The $\varepsilon$-*closure* of a state represents all states which can be reached from that states taking only $\varepsilon$ transitions. We define the $\varepsilon$-*closure* of a state $q \in Q$ as all states $q_n \in Q$ such that a path $(q,\varepsilon,q_1),(q_1,\varepsilon,q_2),\ldots,(q_{n-1},\varepsilon,q_n)$ exists such that all transitions in the path are in $\delta$. For a given G-NFA-$\varepsilon$ $G = (Q, I, F, \delta)$ we can then create a G-NFA $G' = (Q, I, F', \delta')$ such that $G$ and $G'$ recognise the same language. We make:

$$\begin{aligned}\delta' = \; &\{(q,v,q') \mid v \in \Sigma, (q,v,q') \in \delta\} \\ &\cup \{(q,v,q') \mid v \in \Sigma, q'' \in \varepsilon\text{-}closure(q), (q'',v,q') \in \delta\}\end{aligned}$$

and

$$F' = F \cup \{q \mid q_f \in \varepsilon\text{-}closure(q) \text{ for some } q_f \in F\}$$

Using a proof in the same way as used for the classic case any G-NFA-$\varepsilon$ and its corresponding G-NFA then recognise the same language (Bojańczyk et al., 2014).

### 3.2.4 Myhill-Nerode theorem

In this section we will use the Myhill-Nerode theorem applied to G-automata to show that deterministic and non-deterministic G-Automata recognise a different set of languages. Myhill-Nerode equivalence is given with regard to a language $L$ over an alphabet $\Sigma$ by a relation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$ such that two words $w$ and $v$ are related by $\equiv_L$ if and only if their left quotients with regard to $L$ are the same. This is equivalent to saying that $w$ and $v$ are related if for all $u \in \Sigma^*$ we have $wu \in L$ if and only if $vu \in L$ (Shallit, 2008).

The Myhill-Nerode theorem then states that for an orbit finite G-set $A$ and a G-language $L \subseteq A^*$ the following conditions are equivalent (Bojańczyk et al., 2014):

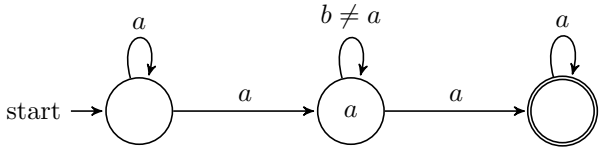
- The set of Myhill-Nerode equivalence classes $L/\equiv_L$ is orbit finite,
- Language $L$ is recognised by a deterministic orbit-finite G-automaton.

Using the Myhill-Nerode theorem we can show that in the context of G-automata deterministic automatons do not recognise the same set of languages as non-deterministic automata.

**Example 3.4.** Consider the following language where $\Sigma$ is equal to $\mathbb{D}$

$$L = \bigcup_{u,v,w \in \Sigma^*} \bigcup_{a \in \Sigma} [\![uavaw]\!]$$

It is easy to see a non-deterministic automaton which recognises this language such as:




However, we can use the Myhill-Nerode theorem to show that there is no deterministic G-automaton which recognises $L$. We do this by showing that the set of Myhill-Nerode equivalence classes $L/\equiv_L$ is not orbit-finite.

*Proof.* For all words $w$ and $w'$ such that $w, w' \in L$ it holds that $w \equiv_L w'$:

- For all $v \in \mathbb{D}^*$ we know $wv \in L$ and $w'v \in L$ holds, thus for all $v \in \mathbb{D}^*$ we have that $wv \in L$ if and only if $w'v \in L$ also holds.

For all words $w$ and $w'$ such that $w, w' \notin L$ we have that $w \equiv_L w'$ if and only if $w$ and $w'$ contain the same unique atoms:

- If $w$ and $w'$ contain the same unique atoms given by some set $C \subset \mathbb{D}$ then for all $v$ both $wv \in L$ and $w'v \in L$ only hold if $v$ contains at least one atom in $C$.

- If $w$ and $w'$ do not contain the same unique atoms there must either be an atom which is in $w$ but which is not in $w'$ or there is an atom which is in $w'$ but is not in $w$:
  - Assume without loss of generality that an atom $a$ which is in $w$ but not in $w'$ exists, then it follows that $wa \in L$ but $w'a \notin L$, thus as there is a $v$ for which $wv \in L$ if and only if $w'v \in L$ does not hold $w \equiv_L w'$ does not hold either.

Thus there is one equivalence class containing all words $w \in L$ and an equivalence class for each combination of unique atoms containing all those words $w \notin L$ for that combination of atoms.

$L/\equiv_L$ is not finite:

- For each combination of unique atoms, words $w \notin L$ exist, those words in which all the atoms occur only once.
- As there are infinitely many combinations of different unique atoms and each of these combinations results in a nonempty equivalence class the set of Myhill-Nerode equivalence classes of $L$ is infinite.

$L/\equiv_L$ is not orbit finite:

- For a given equivalence class of words which are not in $L$, all elements have the same length:
  - If the length of two words differ, they either have different unique elements, meaning they would not be related by $\equiv_L$, or at least one of the words has duplicate elements, meaning it would in fact be in $L$.
- Two equivalence classes converge to a single orbit if and only if their elements have the same length, as length is conserved through permutations.
- For all $x \in \mathbb{N}$ there are equivalence classes in $L/\equiv_L$ such that its elements are of length $x$
- as these all converge to different orbits, there are infinite orbits.

As $L/\equiv_L$ is not orbit finite, the Myhill-Nerode theorem tells us that there is no deterministic G-automaton which recognises $L$. □

Thus as there is a non-deterministic G-automaton which recognises $L$ but no deterministic G-automaton which recognises $L$, we can conclude that deterministic and non-deterministic G-automata recognise a different set of languages.

Because of this difference between deterministic and non-deterministic G-Automata, we cannot determine properties over these different kinds of G-automata by determining those properties for a single kind. As such we will focus on defining transducers which relate to G-NFAs and their languages and focus on the properties which relate to those.

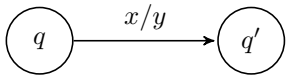
# 4 G-transducers

In this section we will provide a definition of transducers over G-sets. This definition is largely based on the definition of classic transducers and the definition of G-NFAs given in the preliminaries. We will then use this definition to explore the properties of this kind of transducer.

**Definition 4.1.** A G-transducer $T$ is a tuple $(Q, \Sigma, \Gamma, I, F, \delta)$ with:

- An orbit-finite G-set of states $Q$,
- An input alphabet $\Sigma$ being a G-set,
- An output alphabet $\Gamma$ being a G-set,
- An equivariant orbit-finite set of initial states $I \subseteq Q$,
- An equivariant orbit-finite set of accepting states $F \subseteq Q$,
- An equivariant single-step transition relation $\delta \subseteq Q \times \Sigma^* \times \Gamma^* \times Q$.

The transition relation $\delta$ contains tuples $(q, x, y, q')$ such that if a word $x$ is read in state $q$ then $y$ is written and the next state is $q'$. A transition $(q, x, y, q') \in \delta$ is represented by:




We can extend the single-step transition relation $\delta$ to a multi-step transition relation $\delta^* \subseteq Q \times \Sigma^* \times \Gamma^* \times Q$. We define $\delta^*$ as the least relation such that:

$$\frac{(q_1, w, v, q_2) \in \delta}{(q_1, w, v, q_2) \in \delta^*}$$

$$\frac{(q_1, w, v, q_2) \in \delta^* \quad (q_3, x, y, q_4) \in \delta^* \quad q_2 = q_3}{(q_1, wx, vy, q_4) \in \delta^*}$$

Equivalently, we can say that $(q_i, w, v, q_f) \in \delta^*$ if and only if a sequence $(q_0, a_0, b_0, q_1), (q_1, a_1, b_1, q_2), \ldots, (q_{n-1}, a_{n-1}, b_{n-1}, q_n)$ exists such that all tuples in the sequence are in $\delta$, $q_0 \in I$, $q_n \in F, a_0, a_1, \ldots, a_{n-1} \in \Sigma^*, b_0, b_1, \ldots, b_{n-1} \in \Gamma^*$ where $w = a_0 a_1 \ldots a_{n-1}$ and $v = b_0 b_1 \ldots b_{n-1}$. Intuitively, this means that a tuple $(q_i, w, v, q_f)$ is in $\delta^*$ if there is a possible path in the automaton from an initial state to a final state such that $w$ is read on this path and $v$ is outputted on this path.

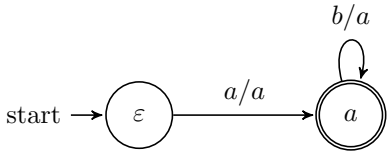
For a transducer $T = (Q, \Sigma, \Gamma, I, F, S)$ we define a function which computes a transduction for words over the input alphabet to languages over the output alphabet. The function $T : \Sigma^* \to \mathcal{P}(\Gamma^*)$ for a word $w \in \Sigma^*$ is defined as:

$$T(w) = \begin{array}{l} \{v \mid (q, w, v, q') \in \delta^* \text{ for some } q \in I \text{ and } q' \in F\} \\ \cup \{\varepsilon \mid w = \varepsilon \text{ and there is } q \in Q \text{ such that } q \in I \text{ and } q \in F\} \end{array}$$

We can extend this to a function $\bar{T} : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Gamma^*)$ which computes a transduction for languages over the input alphabet to languages over the output alphabet. For a language $L \subseteq \Sigma^*$ the function is defined as

$$\bar{T}(L) = \bigcup_{w \in L} T(w)$$

**Example 4.1.** Consider a simple transducer $T$ where input and output alphabets $\Sigma$ and $\Gamma$ are both equal to $\mathbb{D}$:




This transducer replaces all letters in a word by the first letter of the word. Its set of states is given by:

$$Q = \{\varepsilon\} \cup \mathbb{D}$$

This consists of two orbits given by $\varepsilon$ the singleton orbit representing the initial state and $\mathbb{D}$ as the infinite orbit representing the set of states in which the first character read is stored. The sets of initial and accepting states are given by the single orbit sets:

$$I = \{\varepsilon\} \qquad F = \mathbb{D}$$

The transition relation $\delta$ for this transducer is given by:

$$\begin{aligned} \delta = &\{(\varepsilon, a, a, a) \mid a \in \mathbb{D}\} \\ &\cup \{(a, b, a, a) \mid a, b \in \mathbb{D}\} \end{aligned}$$

We know that $\Sigma$, $\Gamma$, and $F$ are equivariant as they are equal to $\mathbb{D}$. Because $\varepsilon \cdot \pi = \varepsilon$ for all $\pi \in G$ we know that $I$ is equivariant as well.

We still need to show that $\delta$ is equivariant for this to be a valid transducer, informally this means that the transducer treats all values in the input language equally and is essentially indifferent to which exact value is being read.

We do this by showing that $\delta \cdot \pi = \delta$ for all $\pi \in G$.

Note that

$$\{(\varepsilon, a, a, a) \mid a \in \mathbb{D}\} \cdot \pi = \{(\varepsilon \cdot \pi, a \cdot \pi, a \cdot \pi, a \cdot \pi) \mid a \in \mathbb{D}\} = \{(\varepsilon, a, a, a) \mid a \in \mathbb{D}\}$$

as $\varepsilon \cdot \pi = \varepsilon$ and $a \cdot \pi \in \mathbb{D}$ for all $a \in \mathbb{D}$ and that

$$\{(a, b, a, a) \mid a, b \in \mathbb{D}\} \cdot \pi = \{(a \cdot \pi, b \cdot \pi, a \cdot \pi, a \cdot \pi) \mid a, b \in \mathbb{D}\} = \{(a, b, a, a) \mid a, b \in \mathbb{D}\}$$

as $a \cdot \pi \in \mathbb{D}$ and $b \cdot \pi \in \mathbb{D}$ for all $a, b \in \mathbb{D}$. From this it follows that:

$$\begin{aligned}\delta \cdot \pi &= (\{(\varepsilon, a, a, a) \mid a \in \mathbb{D}\} \cup \{(a, b, a, a) \mid a, b \in \mathbb{D}\}) \cdot \pi \\ &= \{(\varepsilon, a, a, a) \mid a \in \mathbb{D}\} \cdot \pi \cup \{(a, b, a, a) \mid a, b \in \mathbb{D}\} \cdot \pi \\ &= \{(\varepsilon, a, a, a) \mid a \in \mathbb{D}\} \cup \{(a, b, a, a) \mid a, b \in \mathbb{D}\} \\ &= \delta\end{aligned}$$

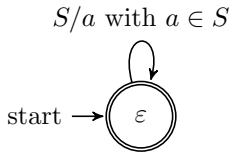
Thus $\delta$ is equivariant.

As we have shown that $\Sigma$, $\Gamma$, $I$, $F$, and $\delta$ are all equivariant, and that $Q$, $I$, and $F$ are all orbit finite sets $T$ is a valid transducer.

**Example 4.2.** We can also consider transducers over alphabets which are not equal to $\mathbb{D}$. Let us consider a transducer where the input alphabet is given by $\Sigma = \binom{\mathbb{D}}{3}$ and the output alphabet $\Gamma$ is equal to $\mathbb{D}$. Note that $\Sigma$ is an equivariant set as the set of all subsets of $\mathbb{D}$ with three elements is closed under permutation.

The goal of this transducer is to output all words that result from the combining the letters in each subsequent set in different ways such that the first letter of each word is from the first set in the input word, the second letter is from the second set and so on.

In order to achieve this we make the following transducer:




Where all of $Q$, $I$, and $F$, are given by the single-orbit set $\{\varepsilon\}$. These are all equivariant as $\varepsilon \cdot \pi = \varepsilon$ for all $\pi \in G$. $\delta$ is then given by:

$$\delta = \{(\varepsilon, S, a, \varepsilon \mid S \in \binom{\mathbb{D}}{3}, a \in S\}$$

We then still need to prove that $\delta$ is equivariant, we can do this by showing that $\delta = \delta \cdot \pi$ for all $\pi \in G$.

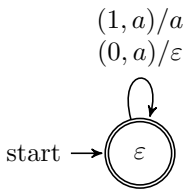
$$\begin{aligned}\delta \cdot \pi &= \{(\varepsilon, S, a, \varepsilon \mid b \in \binom{\mathbb{D}}{3}, a \in S\} \cdot \pi \\ &= \{(\varepsilon \cdot \pi, S \cdot \pi, a \cdot \pi, \varepsilon \cdot \pi \mid S \in \binom{\mathbb{D}}{3}, a \in S\} \\ &= \{(\varepsilon, S \cdot \pi, a \cdot \pi, \varepsilon \mid S \in \binom{\mathbb{D}}{3}, a \in S\} \\ &= \{(\varepsilon, S, a, \varepsilon \mid S \in \binom{\mathbb{D}}{3}, a \in S\} \qquad \text{(by equivariance of } \tbinom{\mathbb{D}}{3}\text{)} \\ &= \delta\end{aligned}$$

Thus $\delta$ is equivariant. $\square$

Following this we know $T$ is a valid transducer.

**Example 4.3.** Similarly it is also possible to define transducers over composite alphabets. Let us consider a transducer with $\Sigma = \{0,1\} \times \mathbb{D}$ and $\Gamma = \mathbb{D}$. Note that $\Sigma$ is equivariant as $\{0,1\} \cdot \pi = \{0 \cdot \pi, 1 \cdot \pi\} = \{0,1\}$ for all $\pi \in G$ and as the Cartesian product preserves equivariance following lemma 3.1.

Let us then consider a transducer $T = (Q, \Sigma, \Gamma, I, F, \delta)$ where whether a letter read from input is written to output depends on whether the value from $\{0,1\}$ is one or zero. To achieve this we make the following automaton:




Here, $Q$, $I$, and $F$ are all given by the single orbit equivariant set $\{\varepsilon\}$ and the transition relation is given by:

$$\delta = \{(\varepsilon, (1,a), a, \varepsilon) \mid a \in \mathbb{D}\} \cup \{(\varepsilon, (0,a), \varepsilon, \varepsilon) \mid a \in \mathbb{D}\}$$

This can easily be seen to be equivariant making T a valid transducer for this purpose.

We will now prove some lemmas about transducers showing a few of their properties. First we show that any valid transducer with an equivariant transition relation $\delta$ also has an equivariant multi-step transition relation $\delta^*$. We then show that the functions which compute transductions over words and languages are equivariant as well.

**Lemma 4.1.** *If $\delta$ is equivariant, then $\delta^*$ is equivariant.*

*Proof.* To prove that $\delta^*$ is equivariant, we need to show that $\delta^* \cdot \pi = \delta^*$ for all $\pi \in G$ which we do by showing that $x \cdot \pi \in \delta^*$ for all $x \in \delta^*$ and $\pi \in G$. In turn, we show this by induction on the structure of $\delta^*$:

For all $x \in \delta^*$, we know that either $x \in \delta$ or that $x = (q_1, wx, vy, q_4)$ for some $q_1, w, x, v, y$, and $q_4$ and that some $(q_1, w, v, q_2) \in \delta^*$ and $(q_3, x, y, q_4) \in \delta^*$ exist such that $q_2 = q_3$.

- If $x \in \delta$, by equivariance of $\delta$, we know that $x \cdot \pi \in \delta$ and, as $\delta \subseteq \delta^*$, $x \cdot \pi \in \delta^*$ for all $\pi \in G$.
- If $x = (q_1, wx, vy, q_4)$ for some $q_1, w, x, v, y$, and $q_4$ where some $(q_1, w, v, q_2) \in \delta^*$ and $(q_3, x, y, q_4) \in \delta^*$ exist such that $q_2 = q_3$. As an induction hypothesis we may then assume that $(q_1, w, v, q_2) \cdot \pi \in \delta^*$ for all $\pi \in G$ and that $(q_3, x, y, q_4) \cdot \pi \in \delta^*$ for all $\pi \in G$.

We need to show that $(q_1, wx, vy, q_4) \cdot \pi \in \delta^*$ for all $\pi \in G$

$$\begin{aligned}(q_1, wx, vy, q_4) \cdot \pi &= (q_1 \cdot \pi, wx \cdot \pi, vy \cdot \pi, q_4 \cdot \pi) \\ &= (q_1 \cdot \pi, (w \cdot \pi)(x \cdot \pi), (v \cdot \pi)(y \cdot \pi), q_4 \cdot \pi)\end{aligned}$$

From the induction hypothesis we know that $(q_1 \cdot \pi, w \cdot \pi, v \cdot \pi, q_2 \cdot \pi) \in \delta^*$ and that $(q_3 \cdot \pi, x \cdot \pi, y \cdot \pi, q_4 \cdot \pi) \in \delta^*$. We also know that $q_2 = q_3$, from which it also follows that $q_2 \cdot \pi = q_3 \cdot \pi$. Following this the second rule of $\delta^*$ tells us that

$$(q_1 \cdot \pi, (w \cdot \pi)(x \cdot \pi), (v \cdot \pi)(y \cdot \pi), q_4 \cdot \pi) \in \delta^*$$

Thus for all $x \in \delta^*$ it holds that $x \cdot \pi \in \delta^*$ for all $\pi \in G$ and as such $\delta^*$ is equivariant. □

**Lemma 4.2.** *The transducing function $T : \Sigma^* \to \mathcal{P}(\Gamma^*)$ is equivariant.*

*Proof.* The function $T$ is equivariant if for all $w \in \Sigma^*$ and $\pi \in G$ it holds that $T(w) \cdot \pi = T(w \cdot \pi)$.

We know that

$$T(w) = \begin{array}{l}\{v \mid (q, w, v, q') \in \delta^* \text{ for some } q \in I \text{ and } q' \in F\} \\ \cup\{\varepsilon \mid w = \varepsilon \text{ and there is } q \in Q \text{ such that } q \in I \text{ and } q \in F\}\end{array}$$

As set union preserves equivariance we show this for both sets in the union separately. So we need to show that

$$\begin{aligned}&\{v \mid (q_i, w, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\} \cdot \pi \\ &= \{v \mid (q_i, w \cdot \pi, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\}\end{aligned}$$

and

$$\begin{aligned}&\{\varepsilon \mid w = \varepsilon \text{ and there is } q \in Q \text{ such that } q \in I \text{ and } q \in F\} \cdot \pi \\ &= \{\varepsilon \mid w \cdot \pi = \varepsilon \text{ and there is } q \in Q \text{ such that } q \in I \text{ and } q \in F\}\end{aligned}$$

For the first we know:

$$\begin{aligned}&\{v \mid (q_i, w, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\} \cdot \pi \\ &= \{v \cdot \pi \mid (q_i, w, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\}\end{aligned}$$

By equivariance of $\delta^*$ we know that $(q_i, w, v, q_f) \in \delta^*$ implies $(q_i, w, v, q_f) \cdot \pi$, which equals $(q_i \cdot \pi, w \cdot \pi, v \cdot \pi, q_f \cdot \pi)$ is in $\delta^*$. By equivariance of $I$ and $F$ we also know that $q_i \in I$ implies $q_i \cdot \pi \in I$ and $q_f \in F$ implies $q_f \cdot \pi \in F$.

Thus for each $v \cdot \pi \in \{v \mid (q_i, w, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\} \cdot \pi$ there is $(q_i \cdot \pi, w \cdot \pi, v \cdot \pi, q_f \cdot \pi)$ such that $(q_i \cdot \pi, w \cdot \pi, v \cdot \pi, q_f \cdot \pi) \in \delta^*$, $q_i \cdot \pi \in I$, and $q_f \cdot \pi \in F$.

$$\{v \mid (q_i, w \cdot \pi, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\}$$

And thus for each $v \cdot \pi \in \{v \mid (q_i, w, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\} \cdot \pi$, we know $v \cdot \pi \in \{v \mid (q_i, w \cdot \pi, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\}$ from which we know that $\{v \mid (q_i, w, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\} \cdot \pi \subseteq \{v \mid (q_i, w \cdot \pi, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\}$.

Conversely, for each $v \in \{v \mid (q_i, w \cdot \pi, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\}$ there is a tuple $(q_i, w \cdot \pi, v, q_f) \in \delta^*$ with $q_i \in I$ and $q_f \in F$. By equivariance of $\delta^*$, $I$, and $F$, and given $\pi^{-1}$ the inverse of $\pi$ we know that

$$(q_i \cdot \pi^{-1}, w \cdot \pi \cdot \pi^{-1}, v \cdot \pi^{-1}, q_f \cdot \pi^{-1}) = (q_i \cdot \pi^{-1}, w, v \cdot \pi^{-1}, q_f \cdot \pi^{-1})$$

is also in $\delta^*$ with $q_i \cdot \pi^{-1} \in I$ and $q_f \cdot \pi^{-1} \in F$.

From this it follows that all these $v \cdot \pi^{-1}$ are in $\{v \mid (q_i, w, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\}$ and thus that all these $v = v \cdot \pi^{-1} \cdot \pi$ are in $\{v \mid (q_i, w, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\} \cdot \pi$. This means that $\{v \mid (q_i, w \cdot \pi, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\} \subseteq \{v \mid (q_i, w, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\} \cdot \pi$. We then know

$$\begin{aligned}&\{v \mid (q_i, w, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\} \cdot \pi \\ &= \{v \mid (q_i, w \cdot \pi, v, q_f) \in \delta^* \text{ for some } q_i \in I \text{ and } q_f \in F\}\end{aligned}$$

For the second part we know:

$$\begin{aligned}&\{\varepsilon \mid w = \varepsilon \text{ and there is } q \in Q \text{ such that } q \in I \text{ and } q \in F\} \cdot \pi \\ &= \{\varepsilon \cdot \pi \mid w = \varepsilon \text{ and there is } q \in Q \text{ such that } q \in I \text{ and } q \in F\} \\ &= \{\varepsilon \mid w = \varepsilon \text{ and there is } q \in Q \text{ such that } q \in I \text{ and } q \in F\} \quad (\text{as } \varepsilon \cdot \pi = \varepsilon)\end{aligned}$$

Which shows that

$$\begin{aligned}&\{\varepsilon \mid w = \varepsilon \text{ and there is } q \in Q \text{ such that } q \in I \text{ and } q \in F\} \cdot \pi \\ &= \{\varepsilon \mid w = \varepsilon \text{ and there is } q \in Q \text{ such that } q \in I \text{ and } q \in F\}\end{aligned}$$

Because we have shown both sides and the property is preserved by set union, the function computing transductions is equivariant. □

**Lemma 4.3.** *The transducing function over languages $\bar{T}$ is equivariant.*

*Proof.* To show $\bar{T} : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Gamma^*)$ is equivariant we must show that for all $L \in \mathcal{P}(\Sigma^*)$ and $\pi \in G$ we have that $\bar{T}(L) \cdot \pi = \bar{T}(L \cdot \pi)$ holds.

$$
\begin{aligned}
\bar{T}(L) \cdot \pi &= (\bigcup_{w \in L} T(w)) \cdot \pi \\
&= \bigcup_{w \in L} T(w) \cdot \pi \\
&= \bigcup_{w \in L} T(w \cdot \pi) && \text{(by equivariance of } T\text{)} \\
&= \bigcup_{w \in L \cdot \pi} T(w) = \bar{T}(L \cdot \pi)
\end{aligned}
$$

Thus $\bar{T}$ is equivariant. □

This definition of G-transducers will be used in the following sections to consider some of the properties of these kinds of transducers.

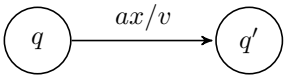
# 5 Normalising G-transducers

In this section we will show that we can normalise transducers, we do this because it shows that allowing transition of words longer than one letter does not affect the power of the transducer and because we use this to show a closure property of the set of languages recognised by a G-NFA in the following section.

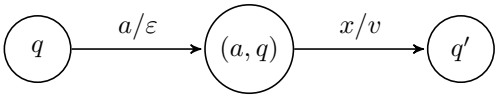
**Definition 5.1.** A *normal* G-transducer is a transducer such that for all $(q, w, v, q') \in \delta$ we have that $|w| \leq 1$ and $|v| \leq 1$.

We show that it is possible for us to transform any arbitrary transducer to a normal transducer. Given an arbitrary transducer $T = (Q, \Sigma, \Gamma, I, F, \delta)$ we create a normal transducer $T_{norm} = (Q_{norm}, \Sigma, \Gamma, I, F, \delta_{norm})$. We do this by constructing an intermediate transducer $T' = (Q', \Sigma, \Gamma, I, F, \delta')$ with a modified transition relation $\delta'$ and set of states $Q'$ by doing the following for every transition $(q, w, v, q')$ in $\delta$, starting with an empty $\delta'$ and $Q'$.

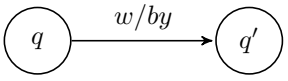
If $|w| \leq 1$ and $|v| \leq 1$ then the transition is not changed and $\delta' \leftarrow \delta' \cup \{(q, w, v, q')\}$ and $Q' \leftarrow Q' \cup \{q, q'\}$. If $|w| > 1$ we split the transition, we know we can write $w$ as $ax$ for some $a \in \Sigma$ and $x \in \Sigma^*$. We then split the transition such that $(q, ax, v, q')$ is replaced by $(q, a, \varepsilon, (a, q))$ and $((a, q), x, v, q')$. Then $\delta' \leftarrow \delta' \cup \{(q, a, \varepsilon, (a, q)), ((a, q), x, v, q')\}$ and $Q' \leftarrow Q' \cup \{q, (a, q), q'\}$, making sure that $(a, q)$ is not already in $Q'$. This means that:



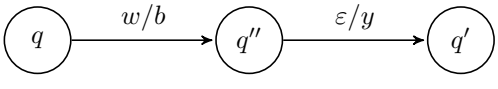

Becomes




Similarly, if $|w| \leq 1$ but $|v| > 1$ we also want to split the transition into two. We can write $v$ as $by$ for some $b \in \Gamma$ and $y \in \Gamma^*$. We then split the transition such that we replace $(q, w, by, q')$ by $(q, w, b, q'')$ and $(q'', \varepsilon, y, q')$ where $q''$ is an arbitrary new state. Then $\delta' \leftarrow \delta' \cup \{(q, w, b, q''), (q'', \varepsilon, y, q')\}$ and $Q' \leftarrow Q' \cup \{q, q', q''\}$. This means that:




Becomes




This process is applied to the resulting $T'$ repeatedly until $|w| \leq 1$ and

$|v| \leq 1$ hold for all transitions $(q, w, v, q')$ in the resulting $\delta'$. In the process we assume that whenever we add a new state $q''$ that it is an arbitrary new state such that if new states are added for two states $q_1$ and $q_2$ that $q_1'' \cdot \pi = q_2''$ if and only if $q_1 \cdot \pi q_2$. We furthermore assume that when a state $(a, q)$ is added that it does not yet exist in the set of states and would otherwise be altered in such a way that it is a new state.

We can also represent a step of this process by means of two functions we called

$$norm_\delta : Q \times \Sigma^* \times \Gamma^* \times Q \to \mathcal{P}(Q' \times \Sigma^* \times \Gamma^* \times Q')$$

and

$$norm_Q : Q \to Q'$$

which can be applied to the elements of $\delta$ and $Q$ respectively to compute this step. These functions are defined as:

$$norm_\delta((q, w, v, q')) = \begin{cases} \{(q, w, v, q')\} & \text{if } |w| \leq 1 \text{ and } |v| \leq 1 \\ \{(q, a, \varepsilon, (a, q)), ((a, q), x, v, q')\} & \text{if } |w| > 1 \text{ and } w = ax \text{ for some } a \in \Sigma \text{ and } x \in \Sigma^* \\ \{(q, w, b, q''), (q'', \varepsilon, y, q')\} & \text{if } |w| \leq 1, |v| > 1 \text{ and } v = by \text{ for some } b \in \Gamma \text{ and } y \in \Gamma^* \end{cases}$$

for $norm_\delta$ applied to transitions $(q, w, v, q')$ in $\delta$ and as:

$$\begin{array}{rl} & \{q \mid (q, w, v, q') \in \delta \text{ or } (q', w, v, q) \in \delta \text{ for some } w, v, q'\} \\ \cup & \{(a, q) \mid (q, w, v, q') \in \delta \text{ for some } w, v, q' \text{ such that } |w| > 1 \\ & \text{and } w = ax \text{ for some } a \in \Sigma \text{ and } x \in \Sigma^*\} \\ \cup & \{q'' \mid (q, w, v, q') \in \delta \text{ for some } w, v, q' \\ & \text{such that } |w| \leq 1 \text{ and } |v| > 1\} \end{array}$$

$norm_Q(q) =$

for states $q$ in $Q$. When applied to the entire set of transitions or states we define them as:

$$norm_\delta(\delta) = \bigcup_{(q,w,v,q') \in \delta} norm_\delta((q, w, v, q') \qquad norm_Q(Q) = \bigcup_{q \in Q} norm_Q(q)$$

A step of the normalisation process is then computed by

$$T' = (norm_Q(Q), \Sigma, \Gamma, I, F, norm_\delta(\delta))$$

which is then computed repeatedly from resulting $T'$ until a normal automaton is achieved.

We now want to prove that this process is valid. This means that we need to show that this operation preserves equivariance such that the resulting transducer is still a valid transducer. We also need to show that the resulting transducer computes the same transduction.

**Lemma 5.1.** *Normalisation of a transducer $T$ to $T_{norm}$ preserves equivariance of $\delta$ such that $\delta_{norm}$ is equivariant.*

*Proof.* To show this, we need to show that, given a transducer $T = (Q, \Sigma, \Gamma, I, F, \delta)$, the $\delta_{norm}$ in normalised transducer $T_{norm}$ is still equivariant. $T_{norm}$ arises from repeatedly applying a step of the normalisation procedure to intermediate $T'$, thus if one step of the procedure preserves equivariance, meaning that $\delta'$ is equivariant, so does the entire procedure.

One step of the normalisation procedure consists of applying $norm_\delta$ to all the elements of $\delta$ and $norm_Q$ to all elements of $Q$. If we can show $norm_\delta$ to be equivariant, then lemmas 3.2 and 3.3 tell us that applying one step of this procedure to equivariant $\delta$ results in equivariant $\delta'$.

If $norm_\delta$ is applied to a transition $(q, w, v, q')$ then there are three possibilities:

If $norm_\delta((q, w, v, q')) = \{(q, w, v, q')\}$ because $|w| \leq 1$ and $|v| \leq 1$ then

$$\begin{aligned} norm_\delta((q, w, v, q')) \cdot \pi &= \{(q, w, v, q')\} \cdot \pi \\ &= \{(q, w, v, q') \cdot \pi\} \\ &= \{(q \cdot \pi, w \cdot \pi, v \cdot \pi, q' \cdot \pi)\} \\ &= norm_\delta((q \cdot \pi, w \cdot \pi, v \cdot \pi, q' \cdot \pi)) \\ &= norm_\delta((q, w, v, q') \cdot \pi) \end{aligned}$$

Otherwise, if $norm_\delta((q, w, v, q')) = \{(q, a, \varepsilon, (a, q)), ((a, q), x, v, q')\}$ where $w = ax$ and $a \in \Sigma$ and $x \in \Sigma^*$ such that $|w| > 1$ then

$$\begin{aligned} norm_\delta((q, ax, v, q')) \cdot \pi &= \{(q, a, \varepsilon, (a, q)), ((a, q), x, v, q')\} \cdot \pi \\ &= \{(q, a, \varepsilon, (a, q)) \cdot \pi, ((a, q), x, v, q') \cdot \pi\} \\ &= \{(q \cdot \pi, a \cdot \pi, \varepsilon, (a, q) \cdot \pi), ((a, q) \cdot \pi, x \cdot \pi, v \cdot \pi, q' \cdot \pi)\} \\ &= norm_\delta((q \cdot \pi, (a \cdot \pi)(x \cdot \pi), v \cdot \pi, q' \cdot \pi)) \\ &= norm_\delta((q \cdot \pi, ax \cdot \pi, v \cdot \pi, q' \cdot \pi)) \\ &= norm_\delta((q, ax, v, q') \cdot \pi) \end{aligned}$$

The last option is that $norm_\delta((q, w, v, q')) = \{(q, w, b, q''), (q'', \varepsilon, y, q')\}$ where $v = by$ and $b \in \Gamma$ and $y \in \Gamma^*$, then

$$\begin{aligned} norm_\delta((q, w, by, q')) \cdot \pi &= \{(q, w, b, q''), (q'', \varepsilon, y, q')\} \cdot \pi \\ &= \{(q, w, b, q'') \cdot \pi, (q'', \varepsilon, y, q') \cdot \pi\} \\ &= \{(q \cdot \pi, w \cdot \pi, b \cdot \pi, q'' \cdot \pi), (q'' \cdot \pi, \varepsilon, y \cdot \pi, q' \cdot \pi)\} \\ &= norm_\delta((q \cdot \pi, w \cdot \pi, (b \cdot \pi)(y \cdot \pi), q' \cdot \pi)) \\ &= norm_\delta((q \cdot \pi, w \cdot \pi, by \cdot \pi, q' \cdot \pi)) \\ &= norm_\delta((q, w, by, q') \cdot \pi) \end{aligned}$$

This means that for all three possibilities

$$norm_\delta((q, w, v, q')) \cdot \pi = norm_\delta((q, w, v, q') \cdot \pi)$$

holds which means that $norm_\delta$ is equivariant. From this we can conclude that performing one step of the normalisation procedure preserves equivariance. As all steps are the same this means that the entire procedure preserves equivariance. □

We will now show that the normalisation operation preserves the transduction computed by the transducer such that the normalised transducer computes the same transduction as the original.

**Lemma 5.2.** *If a step of the normalisation operation has been applied to get a $\delta'$ and $Q'$ from some $\delta$ and $Q$ then for all $q, q' \in Q$ it holds that $(q, w, v, q') \in \delta'^*$ if and only if $(q, w, v, q') \in \delta^*$.*

*Proof.* We first want to prove the property that if $(q, w, v, q') \in \delta^*$ with $q, q' \in Q$, then $(q, w, v, q') \in \delta'^*$, we do this by means of induction on the structure of $\delta^*$.

In the base case we assume that $(q, w, v, q') \in \delta$. In this case there are three possibilities: either $(q, w, v, q') \in \delta'$, $w = ax$ for some $a \in \Sigma$ and $x \in \Sigma^*$ and $(q, a, \varepsilon, (a, q)) \in \delta'$ and $((a, q), x, v, q') \in \delta'$, or $v = by$ for some $b \in \Gamma$ and $y \in \Gamma^*$ and $(q, w, b, q'') \in \delta'$ and $(q'', \varepsilon, y, q') \in \delta'$.

- If $(q, w, v, q') \in \delta'$ then it directly follows that $(q, w, v, q') \in \delta'^*$.
- If $w = ax$ for some $a \in \Sigma$ and $x \in \Sigma^*$ and $(q, a, \varepsilon, (a, q)) \in \delta'$ and $((a, q), x, v, q') \in \delta'$ then $(q, a, \varepsilon, (a, q)) \in \delta'^*$ and $((a, q), x, v, q') \in \delta'^*$, as $(a, q) = (a, q)$ we then know that $(q, ax, v, q') \in \delta'^*$, and as $ax = w$ this means $(q, w, v, q') \in \delta'^*$.
- If $v = by$ for some $b \in \Gamma$ and $y \in \Gamma^*$ and $(q, w, b, q'') \in \delta'$ and $(q'', \varepsilon, y, q') \in \delta'$, then $(q, w, b, q'') \in \delta'^*$ and $(q'', \varepsilon, y, q') \in \delta'^*$, as $q'' = q''$ this implies $(q, w, by, q') \in \delta'^*$. Then as $by = v$ this means that $(q, w, v, q') \in \delta'^*$.

In the inductive case we assume that there are $(q, r, s, q'') \in \delta^*$ and $(q''', x, y, q') \in \delta^*$ with $q'' = q'''$, $rx = w$ and $sy = v$ such that $(q, rx, sy, q') \in \delta^*$. We assume that the property holds for these premises and thus that $(q, r, s, q'') \in \delta'^*$ and $(q''', x, y, q') \in \delta'^*$. From this $(q, rx, sy, q') \in \delta'^*$ directly follows. As $rx = w$ and $sy = v$ this means $(q, w, v, q') \in \delta'^*$.

Thus we know that in all cases that $(q, w, v, q') \in \delta^*$ it follows that $(q, w, v, q') \in \delta'^*$. We then still need to show the property that if $(q, w, v, q') \in \delta'^*$ with $q, q' \in Q$ then $(q, w, v, q') \in \delta^*$ follows. We show this by induction on the structure of $\delta'^*$.

In the base case we assume $(q, w, v, q') \in \delta'$ with $q, q' \in Q$ such that $(q, w, v, q') \in \delta'^*$. In this case as $q, q' \in Q$ this transition was taken from $\delta$

directly without being split. Thus $(q, w, v, q') \in \delta$. From this we immediately know that $(q, w, v, q') \in \delta^*$.

In the inductive case we assume there are $(q, r, s, q'') \in \delta'^*$ and $(q''', x, y, q') \in \delta'^*$ with $q'' = q'''$, $rx = w$, and $sy = v$. We then assume that the property holds for the premises such that $(q, r, s, q'') \in \delta'^*$ implies $(q, r, s, q'') \in \delta^*$ if $q, q'' \in Q$ and that $(q''', x, y, q') \in \delta'^*$ implies $(q''', x, y, q') \in \delta^*$ if $q''', q' \in Q$. We first make the distinction whether $q''$ and thus $q'''$ is in $Q$.

- If $q''$ and $q'''$ are in $Q$ the we know that $(q, r, s, q'') \in \delta^*$ and $(q''', x, y, q') \in \delta^*$. From this and the fact that $q'' = q'''$ we can then immediately conclude that $(q, rx, sy, q') \in \delta^*$.
- If $q''$ and $q'''$ are not in $Q$ then the two transitions $(q, r, s, q'')$ and $(q''', x, y, q')$ must be the result of splitting some transition originally in $\delta$. If this is the case then the transition $(q, rx, sy, q')$ must have been in $\delta$. From this we can then easily conclude that $(q, rx, sy, q') \in \delta^*$.

We then know for $q, q' \in Q$ that if $(q, w, v, q') \in \delta^*$ then $(q, w, v, q') \in \delta'^*$ and vice-versa, so we can conclude for $q, q' \in Q$ that $(q, w, v, q') \in \delta^*$ if and only if $(q, w, v, q') \in \delta'^*$. □

We can then easily show that the new transducer computes the same transduction as the original.

**Theorem 5.1.** *If a step of the normalisation procedure has been applied to get a transducer $T'$ from some original transducer $T$, they compute the same transduction. This means that for all $w \in \Sigma^*$, $v \in T(w)$ if and only if $v \in T'(w)$.*

*Proof.* We know that

$$T(w) = \begin{array}{l} \{v \mid (q, w, v, q') \in \delta^* \text{ for some } q \in I \text{ and } q' \in F\} \\ \cup \{\varepsilon \mid w = \varepsilon \text{ and there is } q \in Q \text{ such that } q \in I \text{ and } q \in F\} \end{array}$$

and that

$$T'(w) = \begin{array}{l} \{v \mid (q, w, v, q') \in \delta'^* \text{ for some } q \in I \text{ and } q' \in F\} \\ \cup \{\varepsilon \mid w = \varepsilon \text{ and there is } q \in Q' \text{ such that } q \in I \text{ and } q \in F\} \end{array}$$

As $I$ and $F$ are both subsets of $Q$ and are shared as sets of initial and final states by $T$ and $T'$. We also know that for all $q, q' \in Q$ it holds that $(q, w, v, q') \in \delta^*$ if and only if $(q, w, v, q') \in \delta'^*$. From this we can then easily conclude that $T(w)$ and $T'(w)$ must be the same set. □

As one step of the normalisation process preserves the transduction that is computed and because all steps are the same the complete normalisation process also trivially preserves the computed transduction which is what we needed to show.

As we have shown that the transducer resulting from the normalisation procedure is valid and that the normalisation procedure preserves the computed transduction we can conclude that we can successfully normalise transducers. In turn, this means that the allowed lengths of the words in the transitions of a transducers does not affect the ability transducers to compute certain transductions.

# 6 Closure of G-NFA languages

In this section we will show the main result of the thesis, that being that we will show a closure property of the set of languages recognised by G-NFAs. In particular we will show that this set of languages is closed under the application of a G-transduction. Formally, this means that if a language $L$ is recognised by a G-NFA and is acted upon by a G-transducer $T$ then the resulting language $\bar{T}(L)$ is also recognised by some G-NFA. We show this by giving a procedure which constructs a new G-NFA which recognises $\bar{T}(L)$ given a G-NFA which recognises language $L$ and a transducer $T$.

Suppose we have a G-NFA $A = (Q_a, I_a, F_a, \delta_a)$ which recognises a language over some alphabet $\Sigma$ and a G-transducer $T$ which has been normalised to a transducer $T_{norm} = (Q_t, \Sigma, \Gamma, I_t, F_t, \delta_t)$. We can then make a automaton which recognises $T([\![A]\!])$ by first combining $A$ and $T_{norm}$ into an intermediate transducer $AT$ and then simplifying $AT$ into a G-NFA $B$ which recognises $T([\![A]\!])$. We want to make the intermediate transducer $AT$ such that it outputs those words which are outputted along some accepting path in $T$ such that the corresponding input word is also accepted by $A$. To achieve this we use a product construction such that

$$AT = ((Q_a \times Q_t), \Sigma, \Gamma, (I_a \times I_t), (F_a \times F_t), \delta_{at})$$

Each state of $AT$ represents a state from $Q_a$ and a state from $Q_t$, each initial state of $AT$ represents a pair of state such that both states are initial in their respective automata and each final state of $AT$ represents a pair of states such that both states are final in their respective automata. We then want to construct the transition relation of $AT$ such that a transition between two states $(q_a, q_t)$ and $(q'_a, q'_t)$ along which $w$ is read and $v$ is outputted is in the transition relation $\delta_{at}$ if there is such a transition $q_t$ and $q'_t$ in $\delta_t$ and if there is a transition between $q_a$ and $q'_a$ in $\delta_a$ along which $w$ is read. Additionally, as $\varepsilon$ cannot be read along transitions in $\delta$, a transition between $(q_a, q_t)$ and $(q_a, q'_t)$ along which $\varepsilon$ is read and $v$ is written is in $\delta_{at}$ if there is such a transition between $q_t$ and $q'_t$ in $\delta_t$. Note that while words longer than a single letter also cannot be read along transitions in $\delta_a$, but this does not need to be considered as the transducer $T$ has been normalised. In all, this means that we get $\delta_{at}$ such that:

$$\begin{aligned} \delta_{at} = \quad & \{((q_a, q_t), w, v, (q'_a, q'_t)) \mid (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\} \\ \cup \quad & \{((q_a, q_t), \varepsilon, v, (q_a, q'_t)) \mid (q_t, \varepsilon, v, q'_t) \in \delta_t\} \end{aligned}$$

We will now introduce a lemma which shows the way in which $\delta_a^*$, $\delta_t^*$ and $\delta_{at}^*$ relate based on this definition. This will be used in the later proof showing the correctness of this procedure.

**Lemma 6.1.** *If* $(q_a, w, q'_a) \in \delta_a^*$ *and* $(q_t, w, v, q'_t) \in \delta_t^*$ *then* $((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta_{at}^*$.

*Proof.* If $(q_t, w, v, q'_t) \in \delta_t^*$ then there is a sequence of one or more tuples

$$(r_0, b_0, c_0, r_1), (r_1, b_1, c_1, r_2), \ldots, (r_{n-1}, b_{n-1}, c_{n-1}, r_n)$$

such that $r_0 = q_t$, $r_n = q'_t$, $w = b_0 b_1 \ldots b_{n-1}$ and $v = c_0 c_1 \ldots c_{n-1}$ and that all these tuples are in $\delta_t$. As $\delta_t$ is the transition relation of a normalised transducer we know that $|b_i| \leq 1$ for all $b_i$ in one of the tuples of the sequence. Similarly, if $(r_a, w, r'_a) \in \delta_a^*$ then there is a sequence of tuples $(q_0, a_0, q_1), (q_1, a_1, q_2), \ldots, (q_{m-1}, a_{m-1}, q_m)$ such that $q_0 = q_a$, $q_m = q'_a$, $w = a_0 a_1 \ldots a_{m-1}$ and that all these tuples are in $\delta_a$.

For all the tuples $(r_i, b_i, c_i, r_{i+1})$ in the sequence which makes $(q_t, w, v, q'_t)$ we know that either $b_i = \varepsilon$ or that $b_i = x$ for some $x \in \Sigma$. If $b_i = \varepsilon$ then because $(r_i, \varepsilon, c_i, r_{i+1}) \in \delta_t$ it follows that $((q_a, r_i), \varepsilon, c_i, (q_a, r_{i+1})) \in \delta_{at}$. If $b_i = x$ for some $x \in \Sigma$ then $b_i$ represents some letter of $w$ being read. Then there is also a tuple $(q_i, a_i, q_{i+1})$ in the sequence representing $(q_a, w, q'_a)$ such that $a_i = b_i$ and that $a_i$ being read in this transition represents $b_i$ being read. If this is the case then because $(q_i, a_i, q_{i+1}) \in \delta_a$, $(r_i, b_i, c_i, r_{i+1}) \in \delta_t$, and $a_i = b_i$ we know that $((q_i, r_i), b_i, c_i, ((q_{i+1}, r_{i+1})) \in \delta_{at}$.

Following this we have that for every tuple $(r_i, b_i, c_i, r_{i+1})$ in the sequence which makes $(q_t, w, v, q'_t)$ there is $((q_j, r_i), b_i, c_i, (q_k, r_i)) \in \delta_{at}$ where $k = j$ if $b_i = \varepsilon$ and $k = j + 1$ otherwise. Then using the structure of $\delta^*$ of G-transducers we know that we can combine these tuples such that a tuple $((q_0, r_0), w, v, (q_m, r_n))$ is in $\delta_{at}^*$. As $q_0 = q_a$, $q_m = q'_a$, $r_0 = q_t$, and $r_n = q'_t$ this means $((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta_{at}^*$. □

We then also show the converse of this lemma as this will also be necessary in the later proofs.

**Lemma 6.2.** *If $((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta_{at}^*$, then either $(q_a, w, q'_a) \in \delta_a^*$ and $(q_t, w, v, q'_t) \in \delta_t^*$ or $w = \varepsilon$, $q_a = q'_a$, and $(q_t, w, v, q'_t) \in \delta_t^*$.*

*Proof.* If $((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta_{at}^*$ then the structure of $\delta_{at}^*$ tells us that either $((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta_{at}$ or there are two tuples $((q_a, q_t), r, s, (q''_a, q''_t)) \in \delta_{at}^*$ and $((q'''_a, q'''_t), x, y, (q'_a, q'_t)) \in \delta_{at}^*$ where $(q''_a, q''_t) = (q'''_a, q'''_t)$, $rx = w$, and $sy = v$. We prove this lemma by induction on the structure of $\delta_{at}^*$.

In the base case we assume that $((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta_{at}$. The way $\delta_{at}$ is defined then directly tells us that either $(q_t, w, v, q'_t) \in \delta_t$ and $(q_a, w, q'_a) \in \delta_a$ hold or that $w = \varepsilon$, $q_a = q'_a$, and $(q_t, \varepsilon, v, q'_t) \in \delta_t$ all hold. These two possibilities directly correspond to the two possibilities in the consequent of the implication in the lemma.

In the inductive case we assume that there are two tuples $((q_a, q_t), r, s, (q''_a, q''_t)) \in \delta_{at}^*$ and $((q'''_a, q'''_t), x, y, (q'_a, q'_t)) \in \delta_{at}^*$ where $(q''_a, q''_t) = (q'''_a, q'''_t)$, $rx = w$, and $sy = v$. We then assume that the implication in the lemma holds for these two tuples.

We then have to make the distinction whether $r$ and/or $x$ are equal to $\varepsilon$. If neither $r$ nor $x$ is equal to $\varepsilon$ then we know by induction that $(q_a, r, q_a'') \in \delta_a^*$ and $(q_t, r, s, q_t'') \in \delta_t^*$ as well as that $(q_a''', x, q_a') \in \delta_a^*$ and $(q_t''', x, y, q_t') \in \delta_t^*$ with $q_a'' = q_a'''$ and $q_t'' = q_t'''$ from which we can easily conclude that $(q_a, rx, q_a') \in \delta_a^*$ and $(q_t, rx, sy, q_t') \in \delta_t^*$.

As the cases in which either $r$ or $x$ is equal to $\varepsilon$ are very similar we then assume without loss of generality that $r = \varepsilon$ but $x \neq \varepsilon$. Then by induction we know that $q_a = q_a''$ and $(q_t, \varepsilon, s, q_t'') \in \delta_t^*$ and that $(q_a''', x, q_a') \in \delta_a^*$ and $(q_t''', x, y, q_t') \in \delta_t^*$. As this means that $q_a = q_a'''$ this means that $(q_a, x, q_a') \in \delta_a^*$ and as $q_t'' = q_t'''$ we can also easily conclude that $(q_t, x, sy, q_t') \in \delta_t^*$.

We then assume that both $r = \varepsilon$ and $x = \varepsilon$. Then by induction we know that $q_a = q_a''$ and $(q_t, \varepsilon, s, q_t') \in \delta_t^*$ and that $q_a''' = q_a'$ and $(q_t''', \varepsilon, y, q_t') \in \delta_t^*$. We can then easily conclude that $q_a = q_a'$, that $(q_t, \varepsilon, sy, q_t') \in \delta_t^*$ as $q_t'' = q_t'''$ and that $w = \varepsilon$ as $w = rx = \varepsilon$.

This means that the lemma holds in all cases and is therefore valid. □

Continuing with the procedure, this is then simplified to a G-NFA-$\varepsilon$:

$$B_\varepsilon = ((Q_a \times Q_t), (I_a \times I_t), (F_a \times F_t), \delta_{b\varepsilon})$$

by filling $\delta_{b\varepsilon}$ with transitions $((q_a, q_t), v, (q_a', q_t'))$ if there is a transition $((q_a, q_t), w, v, (q_a', q_t'))$ in $\delta_{at}$ for some $w \in \Sigma^*$, so:

$$\delta_{b\varepsilon} = \{((q_a, q_t), v, (q_a', q_t')) \mid ((q_a, q_t), w, v, (q_a', q_t')) \in \delta_{at}\}$$

$B_\varepsilon$ can then be simplified to a G-NFA by eliminating the $\varepsilon$ transitions as shown possible in section 3.2.3.

We still need to show that $AT$ and $B_\varepsilon$ are a valid transducer and G-NFA, respectively. The sets of states, initial states, and final states of the automata are given by $(Q_a \times Q_t)$, $(I_a \times I_t)$, and $(F_a \times F_t)$ in both cases. Because $A$ and $T$ are a valid G-NFA and transducer we know that: $Q_a$, $Q_t$, $I_a$, $I_t$, $F_a$, and $F_t$ are all orbit finite equivariant sets. As the Cartesian product preserves orbit finiteness in the equality symmetry we know that $(Q_a \times Q_t)$, $(I_a \times I_t)$, and $(F_a \times F_t)$ are all also orbit finite. Furthermore following lemma 3.1 we know the Cartesian product preserves equivariance of sets. We then still need to show that $\delta_{at}$ and $\delta_{b\varepsilon}$ are equivariant relations.

**Lemma 6.3.** *Given an automaton $A$ and transducer $T$ with transition relations $\delta_a$ and $\delta_t$, the transition relation $\delta_{at}$ of the combined transducer $AT$ and the transition $\delta_{b\varepsilon}$ of the simplified automaton $B_\varepsilon$ are equivariant.*

*Proof.* Recall that:

$$\begin{aligned} \delta_{at} = \quad & \{((q_a, q_t), w, v, (q_a', q_t')) \mid (q_a, w, q_a') \in \delta_a, (q_t, w, v, q_t') \in \delta_t\} \\ \cup \quad & \{((q_a, q_t), \varepsilon, v, (q_a, q_t')) \mid (q_t, \varepsilon, v, q_t') \in \delta_t\} \end{aligned}$$

Because set union preserves equivariance we can show $\delta_{at}$ equivariant by showing that

$$\{((q_a, q_t), w, v, (q'_a, q'_t)) \mid (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\}$$

and

$$\{((q_a, q_t), \varepsilon, v, (q_a, q'_t)) \mid (q_t, \varepsilon, v, q'_t) \in \delta_t\}$$

are equivariant. So we need to show that

$$\begin{aligned}&\{((q_a, q_t), w, v, (q'_a, q'_t)) \mid (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\} \cdot \pi \\ &= \{((q_a, q_t), w, v, (q'_a, q'_t)) \mid (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\}\end{aligned}$$

and

$$\begin{aligned}&\{((q_a, q_t), \varepsilon, v, (q_a, q'_t)) \mid (q_t, \varepsilon, v, q'_t) \in \delta_t\} \cdot \pi \\ &= \{((q_a, q_t), \varepsilon, v, (q_a, q'_t)) \mid (q_t, \varepsilon, v, q'_t) \in \delta_t\}\end{aligned}$$

For the first we know that:

$$\begin{aligned}&\{((q_a, q_t), w, v, (q'_a, q'_t)) \mid (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\} \cdot \pi \\ &= \{((q_a, q_t), w, v, (q'_a, q'_t)) \cdot \pi \mid (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\} \\ &= \{((q_a, q_t) \cdot \pi, w \cdot \pi, v \cdot \pi, (q'_a, q'_t) \cdot \pi) \mid (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\} \\ &= \{((q_a \cdot \pi, q_t \cdot \pi), w \cdot \pi, v \cdot \pi, (q'_a \cdot \pi, q'_t \cdot \pi)) \mid (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\}\end{aligned}$$

By equivariance of $\delta_a$ and $\delta_t$ we know that if $(q_a, w, q'_a) \in \delta_a$ then $(q_a \cdot \pi, w \cdot \pi, q'_a \cdot \pi) \in \delta_a$ and if $(q_t, w, v, q'_t) \in \delta_t$ then $(q_t \cdot \pi, w \cdot \pi, v \cdot \pi, q'_t \cdot \pi) \in \delta_t$. Because of this we know that if $(q_a, w, q'_a) \in \delta_a$ and $(q_t, w, v, q'_t) \in \delta_t$ then consequently

$$((q_a \cdot \pi, q_t \cdot \pi), w \cdot \pi, v \cdot \pi, (q'_a \cdot \pi, q'_t \cdot \pi)) \in \begin{array}{l}\{((q_a, q_t), w, v, (q'_a, q'_t)) \mid \\ (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\}\end{array}$$

$$\text{and } ((q_a \cdot \pi, q_t \cdot \pi), w \cdot \pi, v \cdot \pi, (q'_a \cdot \pi, q'_t \cdot \pi)) \in \begin{array}{l}\{((q_a, q_t), w, v, (q'_a, q'_t)) \mid \\ (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\} \cdot \pi\end{array}$$

Following this we can conclude that

$$\begin{aligned}&\{((q_a, q_t), w, v, (q'_a, q'_t)) \mid (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\} \cdot \pi \\ \subseteq &\{((q_a, q_t), w, v, (q'_a, q'_t)) \mid (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\}\end{aligned}$$

We also know that if

$$((q_a \cdot \pi, q_t \cdot \pi), w \cdot \pi, v \cdot \pi, (q'_a \cdot \pi, q'_t \cdot \pi)) \in \begin{array}{l}\{((q_a \cdot \pi, q_t \cdot \pi), w \cdot \pi, v \cdot \pi, (q'_a \cdot \pi, q'_t \cdot \pi)) \mid \\ (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\}\end{array}$$

then by equivariance of $\delta_a$ and $\delta_t$ we know that $(q_a \cdot \pi, w \cdot \pi, q'_a \cdot \pi) \in \delta_a$ and $(q_t \cdot \pi, w \cdot \pi, v \cdot \pi, q'_t \cdot \pi) \in \delta_t$ from which we can directly conclude that

$$((q_a \cdot \pi, q_t \cdot \pi), w \cdot \pi, v \cdot \pi, (q'_a \cdot \pi, q'_t \cdot \pi)) \in \begin{array}{l}\{((q_a, q_t), w, v, (q'_a, q'_t)) \mid \\ (q_a, w, q'_a) \in \delta_a, (q_t, w, v, q'_t) \in \delta_t\}\end{array}$$

and thus

$$\begin{aligned}&\{((q_a,q_t),w,v,(q'_a,q'_t)) \mid (q_a,w,q'_a)\in\delta_a,(q_t,w,v,q'_t)\in\delta_t\}\\ \subseteq&\{((q_a,q_t),w,v,(q'_a,q'_t)) \mid (q_a,w,q'_a)\in\delta_a,(q_t,w,v,q'_t)\in\delta_t\}\cdot\pi\ \}\end{aligned}$$

As such we know that

$$\begin{aligned}&\{((q_a,q_t),w,v,(q'_a,q'_t)) \mid (q_a,w,q'_a)\in\delta_a,(q_t,w,v,q'_t)\in\delta_t\}\\ =&\{((q_a,q_t),w,v,(q'_a,q'_t)) \mid (q_a,w,q'_a)\in\delta_a,(q_t,w,v,q'_t)\in\delta_t\}\cdot\pi\ \}\end{aligned}$$

For the second we know that

$$\begin{aligned}&\{((q_a,q_t),\varepsilon,v,(q_a,q'_t)) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}\cdot\pi\\ =&\{((q_a,q_t)\cdot\pi,\varepsilon,v\cdot\pi,(q_a,q'_t)\cdot\pi) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}\\ =&\{((q_a\cdot\pi,q_t\cdot\pi),\varepsilon,v\cdot\pi,(q_a\cdot\pi,q'_t\cdot\pi)) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}\end{aligned}$$

By equivariance of $\delta_t$ we know that if $(q_t,\varepsilon,v,q'_t)\in\delta_t$ that also $(q_t\cdot\pi,\varepsilon,v\cdot\pi,q'_t\cdot\pi)\in\delta_t$ and, using equivariance of $Q_a$, also that

$$((q_a\cdot\pi,q_t\cdot\pi),\varepsilon,v\cdot\pi,(q_a\cdot\pi,q'_t\cdot\pi))\in\{((q_a,q_t),\varepsilon,v,(q_a,q'_t)) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}$$

from which we can conclude that

$$\begin{aligned}&\{((q_a,q_t),\varepsilon,v,(q_a,q'_t)) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}\cdot\pi\\ \subseteq&\{((q_a,q_t),\varepsilon,v,(q_a,q'_t)) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}\end{aligned}$$

Furthermore, assume $((q_a,q_t),\varepsilon,v,(q_a,q'_t))\in\{((q_a,q_t),\varepsilon,v,(q_a,q'_t)) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}$, then $(q_t,\varepsilon,v,q'_t)\in\delta_t$. Then there must be some $q''_t$, $v'$, and $q'''_t$ such that $q''_t\cdot\pi^{-1}=q_t$, $v'\cdot\pi^{-1}=v$, and $q'''_t\cdot\pi^{-1}=q'_t$. By equivariance of $\delta_t$ we know that $(q''_t,\varepsilon,v',q'''_t)\in\delta_t$ from which it follows that

$$\begin{aligned}&((q_a,q''_t\cdot\pi),\varepsilon,v\cdot\pi,(q_a,q'''_t\cdot\pi))\in\{((q_a,q_t),\varepsilon,v,(q_a,q'_t)) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}\cdot\pi\\ =&((q_a,q_t),\varepsilon,v,(q_a,q'_t))\in\{((q_a,q_t),\varepsilon,v,(q_a,q'_t)) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}\cdot\pi\end{aligned}$$

Which shows that

$$\begin{aligned}&\{((q_a,q_t),\varepsilon,v,(q_a,q'_t)) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}\\ \subseteq&\{((q_a,q_t),\varepsilon,v,(q_a,q'_t)) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}\cdot\pi\end{aligned}$$

And thus we can conclude that

$$\begin{aligned}&\{((q_a,q_t),\varepsilon,v,(q_a,q'_t)) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}\cdot\pi\\ =&\{((q_a,q_t),\varepsilon,v,(q_a,q'_t)) \mid (q_t,\varepsilon,v,q'_t)\in\delta_t\}\end{aligned}$$

As we have shows that both parts of the union that composes $\delta_{at}$ are equivariant we also know that $\delta_{at}$ is equivariant.

We then still need to show that $\delta_{b\varepsilon}$ is equivariant. Recall that

$$\delta_{b\varepsilon} = \{((q_a, q_t), v, (q'_a, q'_t)) \mid ((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta_{at}\}$$

Then

$$\begin{aligned}\delta_{b\varepsilon} &= \{((q_a, q_t), v, (q'_a, q'_t)) \mid ((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta_{at}\} \cdot \pi \\ &= \{((q_a, q_t), v, (q'_a, q'_t)) \cdot \pi \mid ((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta_{at}\} \\ &= \{((q_a, q_t) \cdot \pi, v \cdot \pi, (q'_a, q'_t) \cdot \pi) \mid ((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta_{at}\}\end{aligned}$$

We know that if $((q_a, q_t), v, (q'_a, q'_t)) \in \delta_{b\varepsilon}$ then $((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta_{at}$ for some $w$. By equivariance of $\delta_{at}$ we then know that $((q_a, q_t) \cdot \pi, w \cdot \pi, v \cdot \pi, (q'_a, q'_t) \cdot \pi) \in \delta_{at}$ and thus that also $((q_a, q_t) \cdot \pi, v \cdot \pi, (q'_a, q'_t) \cdot \pi) \in \delta_{b\varepsilon}$. This shows that $\delta_{b\varepsilon} \cdot \pi \subseteq \delta_{b\varepsilon}$. We also know that if $((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta_{at}$ then by equivariance of $\delta_{at}$ also $((q_a, q_t) \cdot \pi^{-1}, w \cdot \pi^{-1}, v \cdot \pi^{-1}, (q'_a, q'_t) \cdot \pi^{-1}) \in \delta_{at}$, which in turn implies that $((q_a, q_t) \cdot \pi^{-1} \cdot \pi, w \cdot \pi^{-1} \cdot \pi, v \cdot \pi^{-1} \cdot \pi, (q'_a, q'_t) \cdot \pi^{-1} \cdot \pi) \in \delta_{at}$ and thus that $((q_a, q_t)v, (q'_a, q'_t)) \in \delta_{b\varepsilon}$. This then shows that $\delta_{b\varepsilon} \subseteq \delta_{b\varepsilon}$. Because of this we know that $\delta_{b\varepsilon} = \delta_{b\varepsilon}$ which shows that $\delta_{b\varepsilon}$ is equivariant. □

We have been able to show that $\delta_{at}$ and $\delta_{b\varepsilon}$ are equivariant, we know that the set of states is an orbit-finite G-set for both automata, and we know that the sets of initial states and final states of both automata are both orbit finite and equivariant. From this we conclude that $AT$ and $B_\varepsilon$ are a valid transducer and G-NFA-$\varepsilon$.

We then need to prove that this construction is correct, that is, the resulting automaton $B_\varepsilon$ actually recognises the intended languages language, that being the languages resulting from computing transduction $T$ over language $[\![A]\!]$. In this case that means that for all $v \in \Gamma^*$, we have $v \in [\![B]\!]$ if and only if $v \in T([\![A]\!])$.

**Theorem 6.1.** *The construction is correct, that is, the new automaton $B$ recognises the language given by the transduction of $T$ applied to the language recognised by automaton $A$. This means that $v \in [\![B]\!]$ if and only if $v \in T([\![A]\!])$.*

*Proof.* Suppose that $v \in T([\![A]\!])$. Then there is a $w \in [\![A]\!]$ such that $v \in T(w)$. If $w \in [\![A]\!]$ then $(q_a, w, q'_a) \in \delta_a^*$ for $w \neq \varepsilon$ and some $q_a \in I_a$ and $q'_a \in F_a$ or $w = \varepsilon$ and there is some $q_a \in Q_a$ with $q_a \in I_a$ and $q_a \in F_a$. Also if $v \in T(w)$ then we know that either $(q_t, w, v, q'_t) \in \delta_t^*$ for some $q_t \in I_t$ and $q'_t \in F_t$ or $w = \varepsilon$ and $v = \varepsilon$ and there is some $q_t \in Q_t$ with $q_t \in I_t$ and $q_t \in F_t$. We make a distinction on whether $w = \varepsilon$ or not.

If $w = \varepsilon$ and $w \in [\![A]\!]$, then there is some $q_a \in Q_a$ such that $q_a \in I_a$ and $q_a \in F_a$. Then if $v \in T(w)$ we know that there is either $(q_t, \varepsilon, v, q'_t) \in \delta_t^*$ for some $q_t \in I_t$ and $q'_t \in F_t$ or $v = \varepsilon$ and there is some $q_t \in Q_t$ with $q_t \in I_t$ and $q_t \in F_t$. If $(q_t, \varepsilon, v, q'_t) \in \delta_t^*$ then we easily know that $((q, q_t), \varepsilon, v, (q, q'_t)) \in \delta_{at}^*$ for all $q \in Q_a$. Thus in particular it also holds for $q = q_a$ from which we know that

$((q_a, q_t), \varepsilon, v, (q_a, q'_t)) \in \delta^*_{at}$. From this it follows that $((q_a, q_t), v, (q_a, q'_t)) \in \delta^*_{b\varepsilon}$. As $q_a \in I_a$ and $q_t \in I_t$ we know that $(q_a, q_t) \in (I_a \times I_t)$, similarly because $q_a \in F_a$ and $q'_t \in F_t$ we know that $(q_a, q'_t) \in (F_a \times F_t)$. This tells us that $v \in [\![B_\varepsilon]\!]$.

If $v = \varepsilon$ and there is some $q_t \in Q_t$ with $q_t \in I_t$ and $q_t \in F_t$. Then $v \in [\![B_\varepsilon]\!]$ as we know that $(q_a, q_t) \in (I_a \times I_t)$ and $(q_a, q_t) \in (F_a \times F_t)$.

If $w \neq \varepsilon$ and $v \in T(w)$ the we know that $(q_a, w, q'_a) \in \delta^*_a$ and $(q_t, w, v, q'_t) \in \delta^*_t$, lemma 6.1 then tells us that $((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta^*_{at}$. From this it follows that $((q_a, q_t), v, (q'_a, q'_t)) \in \delta^*_{b\varepsilon}$. As $q_a \in I_a$ and $q_t \in I_t$ we know that $(q_a, q_t) \in (I_a \times I_t)$, similarly because $q'_a \in F_a$ and $q'_t \in F_t$ we know that $(q'_a, q'_t) \in (F_a \times F_t)$. Following this the definition of $B_\varepsilon$ tells us that as $((q_a, q_t), v, (q'_a, q'_t)) \in \delta^*_{b\varepsilon}$, $(q_a, q_t) \in (I_a \times I_t)$, and $(q'_a, q'_t) \in (F_a \times F_t)$ that $v$ is recognised by $B_\varepsilon$ and thus that $v \in [\![B_\varepsilon]\!]$. Thus for all possibilities we have that $v \in T([\![A]\!])$ implies $v \in [\![B_\varepsilon]\!]$. As $B$ and $B_\varepsilon$ recognise the same language we thus have $v \in T([\![A]\!])$ implies $v \in [\![B]\!]$.

Suppose that $v \in [\![B]\!]$. This also implies that $v \in [\![B_\varepsilon]\!]$ from which we know that either there is some $((q_a, q_t), v, (q'_a, q'_t)) \in \delta^*_{b\varepsilon}$ with $(q_a, q_t) \in (I_a \times I_t)$ and $(q'_a, q'_t) \in (F_a \times F_t)$ or $v = \varepsilon$ and there is a pair $(q_a, q_t) \in (Q_a \times Q_t)$ such that $(q_a, q_t) \in (I_a \times I_t)$ and $(q_a, q_t) \in (F_a \times F_t)$. In the first case there must be some $w$ such that $((q_a, q_t), w, v, (q'_a, q'_t)) \in \delta^*_{at}$. From $(q_a, q_t) \in (I_a \times I_t)$ we know that $q_a \in I_a$ and $q_t \in I_t$ and similarly from $(q'_a, q'_t) \in (F_a \times F_t)$ we know that $q'_a \in F_a$ and $q'_t \in F_t$. Lemma 6.2 then tells us that either $(q_a, w, q'_a) \in \delta^*_a$ and $(q_t, w, v, q'_t) \in \delta^*_t$ or that $w = \varepsilon$, that $q_a = q'_a$, and that $(q_t, \varepsilon, v, q'_t) \in \delta^*_t$.

If $(q_a, w, q'_a) \in \delta^*_a$ and $(q_t, w, v, q'_t) \in \delta^*_t$ we know because of $(q_a, w, q'_a) \in \delta^*_a$ with $q_a \in I_a$ and $q'_a \in F_a$ that $w$ is recognised by $A$ and thus $w \in [\![A]\!]$. Furthermore, as $(q_t, w, v, q'_t) \in \delta^*_t$ with $q_t \in I_t$ and $q'_t \in F_t$ we know that $v \in T(w)$. Else if $w = \varepsilon$, $q_a = q'_a$ and $(q_t, \varepsilon, v, q'_t) \in \delta^*_t$, we know that because $q_a = q'_a$, $q_a \in I_a$ and $q'_a \in F_a$ that $\varepsilon \in [\![A]\!]$ and thus that $w \in [\![A]\!]$. From $(q_t, \varepsilon, v, q'_t) \in \delta^*_t$ with $q_t \in I_t$ and $q'_t \in F_t$ that $v \in T(\varepsilon)$ and thus that $v \in T(w)$.

In the other case $v = \varepsilon$ and there is a pair $(q_a, q_t) \in (Q_a \times Q_t)$ such that $(q_a, q_t) \in (I_a \times I_t)$ and $(q_a, q_t) \in (F_a \times F_t)$. Then $\varepsilon \in [\![A]\!]$ as we have $q_a \in I_a$ and $q_a \in F_a$ and $\varepsilon \in T(\varepsilon)$ as we have $q_t \in I_t$ and $q_t \in F_t$.

Thus in all cases if $v \in [\![B]\!]$ then there is some $w \in [\![A]\!]$ such that $v \in T(w)$, which means that $v \in [\![B]\!]$ implies $v \in T([\![A]\!])$.

Thus as $v \in T([\![A]\!])$ implies $v \in [\![B]\!]$ and $v \in [\![B]\!]$ implies $v \in T([\![A]\!])$ we know that $T([\![A]\!]) = [\![B]\!]$. □

We have shown that for every G-NFA $A$ and every G-transducer $T$ that there is a valid G-NFA which recognises the language given by using $\bar{T}$ to compute a transduction over the language recognised by $A$. The fact that this is possible shows that the set of languages recognised by G-NFAs is closed with regard to this kind of transduction.

# 7 Related work and conclusions

In this thesis we have introduced a definition for transducers over G-sets. We then showed a number of properties of these kinds of transducers: we showed that it is possible to normalise transducers such there are no transitions remaining in which words longer than one letter are either read or written, showing that allowing these kinds of transitions has no effect on which transductions can be computed. We were further able to show our main result which is that the set of languages that is recognised by G-NFAs is closed under the type of transduction that we describe.

These results are the same as in the classical case but the methods shown are largely different. The normalisation procedure described in this thesis is new and differs from the classic case because the states added when splitting a transition need to contain the values that are being read in some way. The procedure used to show the closure property is similar to a procedure which can be used in the classic case, but differs due to the infinite state spaces. Some of the proofs shown also clearly differ from the classic case. Firstly the equivariance proofs shown are not needed in the classic case and are thus new. Furthermore the proofs used to show the correctness of the normalisation procedure are new as the method differs so greatly from the one used in the classic case.

Work related to this kind of transducer is limited. Work on automata over G-sets exists and are often continuations of the work presented in Bojańczyk et al., 2014, but no other work on transducers over these kinds of sets really exists. Other models which explore transducers over infinite collections of data values exist, such as the streaming transducer introduced in Alur and Černý, 2011. The streaming transducer model requires the input alphabet to contain some set of tag values, the model using G-sets introduced in this thesis requires no such set. The exact relation between the model introduced in this thesis and other models such as the model of streaming transducers could be explored in future work.

In this area some more properties of these kinds of transducers could be explored. For example the functions which these transducers are able to compute could be characterised in some way similar to how Nivat's theorem characterises those function which are computed by some finite state transducer (Shallit, 2008). Additionally a future opportunity is to make framework for different kinds of transducers, such as two-way transducers, over G-sets. Thus there are a lot of opportunities for future research in this area.

# References


Alur, R. & Černý, P. (2011). Streaming transducers for algorithmic verification of single-pass list-processing programs. *SIGPLAN Not. 46*(1), 599–610. doi:10.1145/1925844.1926454

Bojańczyk, M., Klin, B. & Lasota, S. (2014). Automata theory in nominal sets. *Logical Methods in Computer Science*, *10*(3).

Hopcroft, J. E., Motwani, R. & Ullman, J. D. (2006). *Introduction to automata theory, languages, and computation (3rd edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Shallit, J. (2008). *A second course in formal languages and automata theory*. Cambridge University Press. doi:10.1017/CBO9780511808876