# GO IMPLEMENTATION OF UP-TO TECHNIQUES FOR EQUIVALENCE OF WEIGHTED LANGUAGES

**Alessandro Cheli**
Undergraduate Student
Department of Computer Science
Università di Pisa
Pisa, PI 56127
a.cheli6@studenti.unipi.it

September 25, 2020

**ABSTRACT**

Weighted automata generalize non-deterministic automata by adding a quantity expressing the weight (or probability) of the execution of each transition. In this work we propose an implementation of two algorithms for computing language equivalence in finite state linear weighted automata (LWAs). The first, a linear partition refinement algorithm, calculates the linear weighted bisimulation for any given LWA, the second algorithm checks the equivalence of two vectors (states) for a given weighted automata, and is used to verify the results of the first algorithm. We then provide and compare runtime results.

***K*eywords** First keyword · Second keyword · More

## 1 Introduction

In [1], up-to techniques are defined for weighted systems over arbitrary semirings, while in [2], up-to techniques are studied under a more abstract coalgebraic perspective. For the sake of simplicity of the implementation, we focus only on weighted automata over the field of real numbers $\mathbb{R}$. Real numbers are implemented with double precision floating point numbers, known as the `float64` type in the Go programming language.

## 2 Preliminaries and Notation

**Definition 2.1.** A *weighted automaton* over a field $\mathbb{K}$ and an alphabet $A$ is a triple $(X, o, t)$ such that $X$ is a finite set of states, $t = (t_a : X \to K)_{a \in A}$ is a set of transition functions indexed over the symbols of the alphabet and $o : X \to \mathbb{K}$ is the output function. The transition functions $t_a$ are represented as $\mathbb{K}^{X \times X}$ matrices. $o \in \mathbb{K}^X$ is represented as a row vector. $t_a(v)$ denotes the vector obtained by multiplying the matrix $t_a$ by the vector $v \in \mathbb{K}^X$. $o(v)$ denotes the scalar $s \in \mathbb{K}^X$ obtained by multiplying the row vector $o$ by the column vector $v \in \mathbb{K}^X$. $A^*$ is the set of all words over $A$. $\epsilon$ is the empty word and $aw$ is the concatenation of a letter $a$ to the word $w \in A^*$. A weighted language is a function $\psi : A^* \to \mathbb{K}$. A function mapping each state vector into its accepted language, $[\![\cdot]\!] : \mathbb{K}^X \to K^{A^*}$ is defined as follows for every weighted automaton:

$$\forall v \in \mathbb{K}^X, a \in A, w \in A^* \qquad [\![v]\!](\epsilon) = o(v) \qquad [\![v]\!](aw) = [\![t_a(v)]\!](w)$$

Two vectors $v_1, v_2 \in \mathbb{K}^X$ are called language equivalent $v_1 \sim v_2 \iff [\![v_1]\!] = [\![v_2]\!]$. One can extend the notion of language equivalence to states rather than for vectors by assigning to each state $x \in X$ the corresponding unit vector $e_x \in \mathbb{K}^X$. When given an initial state $i$ for a weighted automaton, the language of the automaton can be defined as $[\![i]\!]$.

**Definition 2.2.** A binary relation $R \subseteq X \times Y$ between two sets $X, Y$ is a subset of the cartesian product of the sets. A relation is called *homogeneous* or an *endorelation* if it is a binary relation over $X$ and itself: $R \subseteq X \times X$. In such case, it is simply called a binary relation over $X$. An *equivalence relation* is a binary relation that is reflexive, symmetric and transitive.

An equivalence relation which is compatible with all the operations of the algebraic structure on which it is defined on, is called a *congruence relation*. Compatibility with the algebraic structure operations means that algebraic operations applied on equivalent elements will still yield equivalent elements.

**Definition 2.3.** The congruence closure $c(R)$ of a relation R is the smallest congruence relation $R'$ such that $R \subseteq R'$

**Definition 2.4.**

## 3 Implementation

The algorithms and data structures for this paper are implemented in the Go programming language. This implementation makes use of the *Gonum* library for numerical computations. We only import the Gonum libraries for matrices and linear algebra and visual plotting of samples and functions.

See Section **??**.

### 3.1 Headings: second level

#### 3.1.1 Headings: third level

**Paragraph**

## 4 Examples of citations, figures, tables, references

[**?**, **?**] and see [**?**].

The documentation for `natbib` may be found at

`http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf`

Of note is the command `\citet`, which produces citations appropriate for use in inline text. For example,

```
\citet{hasselmo} investigated\dots
```

produces

> Hasselmo, et al. (1995) investigated…

`https://www.ctan.org/pkg/booktabs`

### 4.1 Figures

See Figure 1. Here is how you add footnotes. [1]

### 4.2 Tables

See awesome Table 1.

### 4.3 Lists

- Lorem ipsum dolor sit amet
- consectetur adipiscing elit.
- Aliquam dignissim blandit est, in dictum tortor gravida eget. In ac rutrum magna.

[1] Sample of the first footnote.

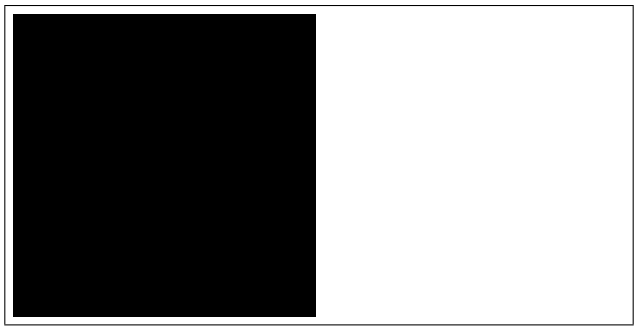

Figure 1: Sample figure caption.

Table 1: Sample table title

| | Part | |
|---|---|---|
| Name | Description | Size ($\mu$m) |
| Dendrite | Input terminal | $\sim$100 |
| Axon | Output terminal | $\sim$10 |
| Soma | Cell body | up to $10^6$ |

## References

[1] Filippo Bonchi, Barbara König, and Sebastian Küpper. Up-to techniques for weighted systems (extended version). *CoRR*, abs/1701.05001, 2017.

[2] Filippo Bonchi, Marcello Bonsangue, Michele Boreale, Jan Rutten, and Alexandra Silva. A coalgebraic perspective on linear weighted automata. *Information and Computation*, 211:77 – 105, 2012.