



An algorithm for computing weighted *LTL* formulas in \bigcirc -normal form

Eleni Mandrali  

International Hellenic University, Greece

Abstract

We define a \bigcirc -normal form for *wLTL*-formulas over a subclass of ω -valuation monoids. We define a deterministic algorithm for determining, for every $\varphi \in \vee\text{-}t\text{-}RULTL$, an equivalent $\vee\text{-}t\text{-}RULTL$ -formula in \bigcirc -normal form in time $\mathcal{O}(|\varphi|^k |AP|)$. The output formula is obtained in the form of an array representation of its syntax tree.

2012 ACM Subject Classification Theory of Computation \rightarrow Logic

Keywords and phrases algorithms, weighted *LTL*, ω -valuation monoids, \bigcirc -normal form

Digital Object Identifier 10.4230/LIPIcs.2025.

1 Introduction

Weighted logics have been introduced by Droste and Gastin in [3], where a weighted MSO logic over semirings for finite words has been defined and has been proposed as a logical formalism that can describe a system's quantitative behavior. In [7] a weighted *LTL* over infinite words, with weights over subclasses of product ω -valuation monoids (the definition of product ω -valuation monoids was introduced in [5]), was presented. That logic was proposed in [8] as a specification language for modeling properties related to the way that quantitative characteristics of a system evolve over time. In fact, for any element k of the weight domain different from the neutral elements of the sum and product operations of the underlying weight structure, the semantics of formulas of a syntactic fragment of the logic express a notion of weighted safety with respect to threshold k . Further, these fragments of k -safe weighted *LTL* formulas are included to a larger syntactic fragment of the logic whose formulas can be effectively translated to equivalent weighted Büchi automata (wBa for short)[7], [8]. This result is of special interest, since for a family of the product ω -valuation monoids employed in [8], the quantitative language equivalence problem of wBa was proved decidable by generalizing a result of [2]. For the, proposed in [7], translation of weighted *LTL* formulas to wBa to be effective, a reduction of formulas with respect to the next operators, and the conjunction operators is necessary. The reduction with respect to the next operator produces a formula that is equivalent to the original one and also has the property that no until operator appears in the scope of a next operator in the formula.

Motivated by the need to study the complexity of translating weighted *LTL* formulas to wBa, and eventually the complexity of decision procedures that incorporate this translation (see for example [8]), we deal with the problem of defining a next normal form for weighted *LTL* formulas, and that of efficiently computing, given a weighted *LTL* formula, an equivalent one in next normal form. More specifically, in Section 4 we define that a weighted *LTL* formula is in next normal form if no operator different from the negation and the next operator appears in the scope of a next operator in the formula. We identify a quantitative necessary and sufficient condition that weighted *LTL* formulas must satisfy to be in next normal form. The condition requires that the sum of properly defined distances between specific operators in the syntax tree of the formula is equal to zero. These distances can be computed in time $\mathcal{O}(|\varphi|^k + |AP|)$. We further present, in Section 5, a deterministic



© Eleni Mandrali;

licensed under Creative Commons License CC-BY 4.0

1st International Workshop on Behavioural Metrics and Quantitative Logics.

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

recursive algorithm that given the array representation of the syntax tree of a formula of a fragment of our logic terminates and outputs the array representation of the syntax tree of an equivalent formula in next normal form. We prove that the time complexity of the algorithm is $\mathcal{O}\left(|\varphi|^k |AP|\right)$. The results presented in this paper are part of an ongoing work that studies the complexity of translating weighted *LTL* formulas to wBa via the construction presented in [7]. In this work, we use as weight domains for our weighted *LTL*, elements of a subclass of ω -valuation monoids, that capture the properties of a family of the product- ω -valuation monoids employed in [8]. We introduce the definition of the weight structure in Section 3.

2 Preliminaries

Let A be an alphabet, i.e., a finite non-empty set. As usually, we denote by A^* the set of all finite words over A and we let $A^+ = A^* \setminus \{\varepsilon\}$, where ε is the empty word. A finite word over A shall be denoted by $w = w(0)w(1)\dots w(n)$ where $w(i) \in A$ ($0 \leq i \leq n$). The prefix relation on A^* is a partial order defined in the following way: for every $u, w \in A^*$, $u \leq_{\text{pref}} w$ iff there exists $v \in A^*$ such that $w = uv$. A set $B \subseteq A^*$ is called prefix-closed if $w \in B$ implies $u \in B$ for every $u \leq_{\text{pref}} w$. For every $w \in A^*$ we shall denote by $|w|$ the number of letters of w . The set of all infinite sequences with elements in A , i.e., the set of all infinite words over A , is denoted by A^ω . An infinite word w over A is also denoted by $w = w(0)w(1)\dots$ where $w(i) \in A$ for all $i \geq 0$.

We shall denote by \mathbb{N} the set of non-negative integers, and by \mathbb{N}_+ the set $\mathbb{N} \setminus \{0\}$. An infinite ranked alphabet (see [9]) is a pair (Σ, rk_Σ) (simply denoted by Σ) where Σ is a possibly infinite set and $rk_\Sigma : \Sigma \rightarrow \mathbb{N}$ such that the set $\{n \in \mathbb{N} \mid \exists \sigma \in \Sigma, rk_\Sigma(\sigma) = n\}$ is finite. We set $\Sigma_k = \{\sigma \in \Sigma \mid rk_\Sigma(\sigma) = k\}$ for every $k \geq 0$. A finite tree t over Σ is defined as a partial mapping $t : \mathbb{N}_+^* \rightarrow \Sigma$ such that the domain of t , $dom(t)$, is a non-empty prefix closed set, and for every p of the domain of t , if $t(p) \in \Sigma_k$, $k \geq 0$, then for every $i \in \mathbb{N}_+$, $pi \in dom(t)$ iff $1 \leq i \leq k$. The set of positions of t , $Pos(t)$, is defined to be the domain of t . We will call ε the root of t , and we shall denote by T_Σ the set of all finite trees over Σ . Let now $t \in T_\Sigma$, and $p, p' \in Pos(t)$ such that $p \leq_{\text{pref}} p'$. The path of t from p to p' is the sequence $p, pp'(|p|), pp'(|p|)p'(|p|+1), \dots, p'$ of elements of $Pos(t)$. The length of the path from p to p' is defined to be equal to $|p'| - |p|$. For every $t \in T_\Sigma$ and $p \in Pos(t)$, the subtree $t|_p$ of t at p is defined as follows: $Pos(t|_p) = \{u \in \mathbb{N}_+^* \mid pu \in Pos(t)\}$, and $t|_p(u) = t(pu)$ for every $u \in Pos(t|_p)$. Next, for every $t, t' \in T_\Sigma$, and every $p \in Pos(t)$, we let $t[t']|_p$ denote the tree obtained by t if we substitute in t the subtree $t|_p$ by t' at position p .

Let now (Σ, rk_Σ) be an infinite ranked alphabet with $\max\{rk_\Sigma(\sigma) \mid \sigma \in \Sigma\} = 2$. Then, every element of T_Σ will be called a binary tree. Let t be a binary tree over (Σ, rk_Σ) . For every $p \in Pos(t)$ such that $t(p) \in \Sigma_2$, we shall call $p1, p2 \in Pos(t)$ the left, and right child of p , respectively. Similarly, for every $p \in Pos(t)$ such that $t(p) \in \Sigma_1$, we shall call $p1 \in Pos(t)$ the left child of p , and we shall say that p has no right child. We say that every $p \in Pos(t)$, such that $t(p) \in \Sigma_0$, has no children. Finally, for every $p \in Pos(t) \setminus \{\varepsilon\}$ the parent of p is defined to be $p(0)\dots p(|p|-2) \in Pos(t)$. The root ε of t has no parent.

We encode a binary tree t over (Σ, rk_Σ) using a four-row array [1] in the following way. The size of each row is equal to the number of positions of the binary tree. The i -indexed positions of the four rows will be used to represent position i of the binary tree (assuming that we have enumerated, starting from 0, the positions of the tree in a top-down, and from left to right way). The data that is stored on the i -indexed position of the first row, is the label of the tree-position i (i.e. the element of Σ assigned by t to tree-position i). The data

that is stored on the i -indexed position of the second row (resp. the third, the fourth row), is the array-position of the left child (resp. the right child, the parent) of tree-position i . For every $0 \leq i \leq \text{Card}(\text{Pos}(t)) - 1$ (where $\text{Card}(C)$ stands for the number of elements of a set C) we shall denote the fact that position i in the tree has no left child (resp. no right child, no parent) by writing $NULL$ in the i -indexed position of the corresponding row of the array. Given an array B we shall denote by $|B|$ its size, i.e., the number of its positions. In the sequel, we use binary trees over infinite ranked alphabets to represent the syntax of weighted LTL formulas.

3 Totally generalized (ω, \leq) -valuation monoids

Let C, K be sets. We denote by $B \subseteq_{fin} C$ the fact that B is a finite subset of C and we let $(C_{fin})^\omega = \bigcup_{B \subseteq_{fin} C} B^\omega$. An index set I of C is a subset of C . A family of elements of K

over the index set I , denoted by $(k_i)_{i \in I}$, is a mapping f from I to K where $k_i = f(i)$ for all $i \in I$. A monoid $(K, +, \mathbf{0})$ is an algebraic structure equipped with a non-empty set K and an associative additive operation $+$ with a zero element $\mathbf{0}$, i.e., $\mathbf{0} + k = k + \mathbf{0} = k$ for every $k \in K$. The monoid K is called commutative if $+$ is commutative. An ω -valuation function over K is a function $Val^\omega : (K_{fin})^\omega \rightarrow K$ such that for every $C \in K_{fin}$, and every $w \in C^\omega$, $Val^\omega(w) \in C$. Let K be a commutative monoid. K is called additively idempotent (or simply idempotent), if $k + k = k$ for every $k \in K$. We recall (cf. [6]) that idempotency gives rise to a natural partial order in K defined in the following way. Let $k, k' \in K$, then $k \leq k'$ iff $k' = k' + k$. Equivalently, it holds $k \leq k'$ iff $k' = k'' + k$ for some $k'' \in K$ (cf. [4]). Clearly, $\mathbf{0} \leq k$ for every $k \in K$. We shall call the natural order of K a total order, if $k \leq k'$, or $k' \leq k$ for all $k, k' \in K$. A commutative idempotent monoid K will be called ordered if the natural order induced by idempotency is a total order. We recall that a monoid $(K, +, \mathbf{0})$ is called complete if it is equipped, for every index set I , with an infinitary sum operation $\sum_I : K^I \rightarrow K$ such that for every family $(k_i)_{i \in I}$ of elements of K we have $\sum_{i \in \emptyset} k_i = \mathbf{0}$, $\sum_{i \in \{j\}} k_i = k_j$, $\sum_{i \in \{j, l\}} k_i = k_j + k_l$ for $j \neq l$, and

$$\sum_{j \in J} \left(\sum_{i \in I_j} k_i \right) = \sum_{i \in I} k_i, \text{ if } \bigcup_{j \in J} I_j = I \text{ and } I_j \cap I_{j'} = \emptyset \text{ for } j \neq j'.$$

We let now $(K, +, \mathbf{0})$ be a commutative idempotent ordered monoid with a maximum element denoted by $\mathbf{1}$, i.e., $k \leq \mathbf{1}$ for all $k \in K$. For every family $(k_i)_{i \in I}$ of elements of K , we denote by $\sup_{i \in I} (k_i)$, and $\inf_{i \in I} (k_i)$, the supremum and infimum respectively of $(k_i)_{i \in I}$ with respect to the natural order induced by idempotency. Clearly, every commutative idempotent ordered monoid $(K, +, \mathbf{0})$ with a maximum element is a complete monoid with $\sum (k_i)_{i \in I} = \sup (k_i)_{i \in I}$. We shall denote such a structure by $(K, \text{sup}, \mathbf{0})$. We present now the definition of (ω, \leq) -valuation monoids, and totally generalized- (ω, \leq) -valuation monoids. In fact, the definition of (ω, \leq) -valuation monoids describes a subclass of ω -valuation monoids introduced in [5].

► **Definition 1.** Let $(K, +, \mathbf{0})$ be a commutative idempotent ordered monoid with a maximum element. An (ω, \leq) -valuation monoid $(K, \text{sup}, Val^\omega, \mathbf{0})$ is the complete monoid $(K, \text{sup}, \mathbf{0})$ equipped with an ω -valuation function $Val^\omega : (K_{fin})^\omega \rightarrow K$ such that $Val^\omega(k_i)_{i \in \mathbb{N}} = \mathbf{0}$ whenever $k_i = \mathbf{0}$ for some $i \geq 0$. A totally generalized (ω, \leq) -valuation monoid (for short TG- (ω, \leq) -valuation monoid) $(K, \text{sup}, Val^\omega, \mathbf{0})$ is an (ω, \leq) -valuation monoid $(K, \text{sup}, Val^\omega, \mathbf{0})$ further equipped with the following properties: $Val^\omega(\mathbf{1}^\omega) = \mathbf{1}$, and for every $L \subseteq_{fin} K$, finite index sets I_j ($j \geq 0$) such that for all $j \geq 0$, it holds $k_{i_j} \in L \setminus \{\mathbf{0}, \mathbf{1}\}$ for all $i_j \in I_j$, or $k_{i_j} \in \{\mathbf{0}, \mathbf{1}\}$ for all $i_j \in I_j$, we have $Val^\omega \left(\sup_{i_j \in I_j} (k_{i_j}) \right)_{j \in \mathbb{N}} = \sup_{(i_j)_{j \in I_0 \times I_1 \times \dots}} (Val^\omega(k_{i_j})_{j \in \mathbb{N}})$.

We note that for every (ω, \leq) -valuation monoid, the structure $(K, \sup, \inf, Val^\omega, \mathbf{0}, \mathbf{1})$ is a totally generalized product- ω -valuation monoid (for short TGP- ω -valuation monoid). For the definition of TGP- ω -valuation monoids we refer the reader to Definition 6 in [8]. As in [8], in the rest of this paper we will consider TG- (ω, \leq) -valuation monoids $(K, \sup, Val^\omega, \mathbf{0})$ that further satisfy the following properties that state variants of neutrality of $\mathbf{1}$ with respect to Val^ω , and a notion of monotonicity of Val^ω . More specifically the following hold. For all $k, k_i \in K$ ($i \geq 1$) it holds $Val^\omega(\mathbf{1}, k_1, k_2, k_3, \dots) = Val^\omega(k_i)_{i \geq 1}$, (Property 2), and $k = Val^\omega(k, \mathbf{1}, \mathbf{1}, \mathbf{1}, \dots)$ (Property 3). Moreover, for all $k \in K$, and all $(k_i)_{i \geq 0} \in (K_{fin})^\omega$ with $k_i \geq k$ ($i \geq 0$) it holds $Val^\omega(k_0, k_1, k_2, \dots) \geq k$ (Property 4).

In the rest of this work, we will call TG- (ω, \leq) -valuation monoids that satisfy properties 2, 3, and 4 simply TG- (ω, \leq) -valuation monoids. Throughout the rest of the paper K will stand for a TG- (ω, \leq) -valuation monoid $(K, \sup, Val^\omega, \mathbf{0})$.

► **Example 2.** We let $K_2 = (\overline{\mathbb{Q}}, \sup, \limsup, -\infty)$ where $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{\infty, -\infty\}$, ∞ is the maximum element, and \limsup is an ω -valuation function from $(\overline{\mathbb{Q}}_{fin})^\omega$ to $\overline{\mathbb{Q}}$ defined in the following way: If there exists $i \geq 0$ with $d_i = -\infty$, then $\limsup((d_i)_{i \geq 0}) = -\infty$. If for all $i \geq 0$, $d_i = \infty$, then $\limsup((d_i)_{i \geq 0}) = \infty$. If $d_j \neq -\infty$ for all $j \geq 0$, and there exist infinitely many $i \geq 0$ with $d_i \neq \infty$, then $\limsup((d_i)_{i \geq 0}) = \inf_{i \geq 0} (\sup \{d_k \mid k \geq i, d_k \neq \infty\})$. Otherwise, $\limsup((d_i)_{i \geq 0}) = \sup \{d_i \mid i \geq 0 \text{ with } d_i \neq \infty\}$. K_2 is a TG- (ω, \leq) -valuation monoid.

4 Weighted *LTL* over TG- (ω, \leq) -valuation monoids

Let AP be a finite set of atomic propositions. As in [8], the syntax of the weighted *LTL* over AP and K is given by the grammar

$$\varphi ::= k \mid a \mid \neg a \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \bigcirc \varphi \mid \varphi U \psi \mid \Box \varphi$$

where $k \in K$, and $a \in AP$. We shall denote by $wLTL$ the class of all weighted *LTL* formulas over AP and K .

► **Definition 3.** The semantics of formulas $\varphi \in wLTL$ are represented as infinitary series $\|\varphi\| : (\mathcal{P}(AP))^\omega \rightarrow K$ which are inductively defined in the following way: For every $w \in (\mathcal{P}(AP))^\omega$ we let

$$\begin{aligned} & - \|\mathbf{k}\|(w) = k \\ & - \|a\|(w) = \begin{cases} \mathbf{1} & \text{if } a \in w(0) \\ \mathbf{0} & \text{otherwise} \end{cases} \quad - \|\neg a\|(w) = \begin{cases} \mathbf{1} & \text{if } a \notin w(0) \\ \mathbf{0} & \text{otherwise} \end{cases} \\ & - \|\varphi \vee \psi\|(w) = \sup(\|\varphi\|(w), \|\psi\|(w)) \quad - \|\varphi \wedge \psi\|(w) = \inf(\|\varphi\|(w), \|\psi\|(w)) \\ & - \|\bigcirc \varphi\|(w) = \|\varphi\|(w(1)w(2)\dots) \quad - \|\Box \varphi\|(w) = Val^\omega(\|\varphi\|(w(i)w(i+1)\dots))_{i \geq 0} \\ & - \|\varphi U \psi\|(w) = \sup_{i \geq 0} (Val^\omega(\|\varphi\|(w), \dots, \|\varphi\|(w(i-1)w(i)\dots), \|\psi\|(w(i)w(i+1)\dots), \mathbf{1}, \mathbf{1}, \dots)) \end{aligned}$$

The semantics of a $wLTL$ -formula φ over a TG- (ω, \leq) -valuation monoid $(K, \sup, Val^\omega, \mathbf{0})$ coincides with the semantics of φ over the TGP- ω -valuation monoid $(K, \sup, \inf, Val^\omega, \mathbf{0}, \mathbf{1})$. We recall that for $\varphi, \psi \in wLTL$, we say that φ, ψ are equivalent, and we denote it by $\varphi \equiv \psi$, if $\|\varphi\|(w) = \|\psi\|(w)$ for every $w \in (\mathcal{P}(AP))^\omega$.

Let $\varphi \in wLTL$. We let the size of φ , denoted by $|\varphi|$, be the number of operators that appear in φ , and $OP(\varphi)$ be the set of operators that appear in φ . We denote by $nest(\varphi)$ the maximal number of nesting operators that appear in φ , and by $Cl(\varphi)$, the closure of φ , i.e., the set that contains φ and all the subformulas of φ .

Moreover, if $\varphi \notin AP \cup K$, we define $EXOP(\varphi) \in \{\wedge, \vee, \neg, \Box, U, \bigcirc, null\}$ to be the most external operator in φ , i.e., the unique operator with the property that it has an appearance in φ that is not in the scope of any other operator in φ . Otherwise, we let $EXOP(\varphi) = null$. We also define $Left(\varphi), Right(\varphi) \in Cl(\varphi) \cup \{Empty\}$, by pointing out the following cases: If $\varphi \in K \cup AP$, then $Left(\varphi) = Right(\varphi) = Empty$. If $\varphi = \neg a$ with $a \in AP$, then $Left(\varphi) = a$, and $Right(\varphi) = Empty$. If $\varphi = \bigcirc\psi$ (resp. $\varphi = \Box\psi$) with $\psi \in wLTL$, then $Left(\varphi) = \psi$, $Right(\varphi) = Empty$. Finally, if $\varphi = \psi \wedge \xi$ (resp. $\varphi = \psi \vee \xi$, $\varphi = \psi U \xi$), with $\psi, \xi \in wLTL$, then $Left(\varphi) = \psi$, $Right(\varphi) = \xi$.

As in the case of classical *LTL*-formulas, we can represent every *wLTL*-formula φ by its syntax tree if we consider the infinite ranked alphabet $\{\wedge, \vee, \neg, \Box, U, \bigcirc\} \cup K \cup AP$, where the binary operators \wedge, \vee, U are of rank 2, the unary operators \bigcirc, \neg, \Box are of rank 1, and the elements of $K \cup AP$ are of rank 0. We will denote the syntax tree of a *wLTL*-formula φ by $T(\varphi)$. Syntax trees of *wLTL*-formulas are binary trees, and as such are amenable to an array encoding as described in Section 2. For a weighted *LTL* formula φ , we will denote by $Array(\varphi)$ the array representation of $T(\varphi)$. We can determine an algorithm that given a *wLTL*-formula as input string, and AP as input array, outputs $Array(\varphi)$ in time $\mathcal{O}(|\varphi|^3 + |AP|)$.

Let now $\varphi \in wLTL$, and $T(\varphi)$ be the corresponding syntax tree. For every $* \in \{U, \wedge, \vee, \Box, \neg, \bigcirc\}$, we define $dist(T(\varphi), *)$ as follows: If $* \in OP(\varphi) \setminus \{EXOP(\varphi)\}$, then we let $dist(T(\varphi), *)$ be the maximum length among the lengths of all paths of $T(\varphi)$ that lead from the root to a different from the root position labeled by a $*$ -operator. Otherwise, we let $dist(T(\varphi), *) = 0$. Now, for every $\varphi \in wLTL$, and every $* \in \{U, \wedge, \vee, \Box\}$, we define the $(\bigcirc, *)$ -maximal distance in φ , and denote it by $mdist(\varphi, \bigcirc, *)$, in the following way: if $\bigcirc \in OP(\varphi)$, then $mdist(\varphi, \bigcirc, *) = \max\{dist(T(\psi), *) \mid \psi \in Cl(\varphi) \text{ with } EXOP(\psi) = \bigcirc\}$, otherwise $mdist(\varphi, \bigcirc, *) = 0$. With standard arguments we can verify that for every $\varphi \in wLTL$, and every $* \in \{\wedge, \vee, U, \Box\}$, $mdist(\varphi, \bigcirc, *) = 0$ iff no $*$ operator appears in the scope of a \bigcirc -operator in φ .

► **Definition 4.** Let $\varphi \in wLTL$. We say that φ is in \bigcirc -normal form if no operator different from \bigcirc , and \neg appears in the scope of a \bigcirc -operator in φ .

► **Proposition 5.** Let $\varphi \in wLTL$. Then,

$$\varphi \text{ is in } \bigcirc\text{-normal form iff } \sum_{* \in \{\wedge, \vee, U, \Box\}} mdist(\varphi, \bigcirc, *) = 0.$$

Let $\varphi, \psi, \xi \in wLTL$ such that $\varphi = \bigcirc(\psi * \xi)$ where $* \in \{\wedge, \vee, U\}$ (resp. $\varphi = \bigcirc(\Box\psi)$, $\varphi = \bigcirc k$ where $k \in K$), we shall denote by $\varphi_{\equiv 1}$ the formula $(\bigcirc\psi) * (\bigcirc\xi)$ (resp. the formula $\Box(\bigcirc\psi)$, the formula k). It holds $\varphi \equiv \varphi_{\equiv 1}$. For every *wLTL*-formula φ we can effectively construct an equivalent one in \bigcirc -normal form by applying the aforementioned equivalences.

► **Example 6.** For $\varphi = \bigcirc((3 \wedge (aUb))U(2 \wedge (\bigcirc c))) \in wLTL$ over K_2 and $AP = \{a, b, c\}$, it holds $mdist(\varphi, \bigcirc, U) = \max\{dist(T(\varphi), U), dist(T(\bigcirc c), U)\} = \max\{3, 0\} = 3$. φ is not in \bigcirc -normal form. For $\psi = (3 \wedge ((\bigcirc a)U(\bigcirc b)))U(2 \wedge (\bigcirc(\bigcirc c)))$ it holds that $\psi \equiv \varphi$, and ψ is in \bigcirc -normal form. Moreover, for every $* \in \{\wedge, \vee, U, \Box\}$, we have $mdist(\psi, \bigcirc, *) = \max\{dist(T(\bigcirc a), *), dist(T(\bigcirc b), *), dist(T(\bigcirc(\bigcirc c)), *), dist(T(\bigcirc c), *)\} = 0$.

The syntactic boolean fragment of *wLTL*, denoted by *bLTL*, is given by the grammar $\varphi ::= \mathbf{0} \mid \mathbf{1} \mid a \mid \neg a \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi U \varphi \mid \Box\varphi$. A *wLTL*-formula of the form $k \wedge \varphi$, where $k \in K \setminus \{\mathbf{0}, \mathbf{1}\}$, and $\varphi \in bLTL$ will be called a monomial over AP and K . We shall denote by *mLTL* the class of all monomials over AP and K . We let the class of

XX:6 An algorithm for computing weighted LTL formulas in \bigcirc -normal form

restricted LTL -step formulas over AP and K , denoted by $r\text{-}stLTL$, be the least class of $wLTL$ -formulas inductively defined in the following way: $mLTL \subseteq r\text{-}stLTL$, and if $\varphi, \psi \in r\text{-}stLTL$, then $\varphi \vee \psi \in r\text{-}stLTL$. We note that every $\varphi \in r\text{-}stLTL$ can also be defined as a generalized disjunction of the form $\bigvee_{1 \leq i \leq n} (k_i \wedge \varphi_i)$ where $k_i \in K \setminus \{\mathbf{0}, \mathbf{1}\}$ and $\varphi_i \in bLTL$ for every $1 \leq i \leq n$ (see [8]).

► **Definition 7.** [8] We let the fragment of \vee -totally restricted U -nesting LTL -formulas over AP and K , denoted by $\vee\text{-}t\text{-}RULTL$, be the least class of formulas of $wLTL$ which is inductively defined in the following way

- $K \bigcup bLTL \bigcup r\text{-}stLTL \subseteq \vee\text{-}t\text{-}RULTL$,
- If $\varphi \in \vee\text{-}t\text{-}RULTL$, then $\bigcirc\varphi \in \vee\text{-}t\text{-}RULTL$,
- If $\varphi, \psi \in r\text{-}stLTL$, then $\varphi U \psi \in \vee\text{-}t\text{-}RULTL$,
- If $\varphi \in r\text{-}stLTL$, then $\Box\varphi \in \vee\text{-}t\text{-}RULTL$,
- If $\varphi, \psi \in r\text{-}stLTL$, or $\varphi = \lambda U \xi, \psi = \Box \zeta$ with $\lambda, \xi, \zeta \in r\text{-}stLTL$, then $\varphi \vee \psi, \psi \vee \varphi \in \vee\text{-}t\text{-}RULTL$,
- If $\varphi \in bLTL$, and $\psi \in r\text{-}stLTL$, or $\psi = \lambda U \xi$, or $\psi = \Box \zeta$ with $\lambda, \xi, \zeta \in r\text{-}stLTL$, then $\varphi \wedge \psi, \psi \wedge \varphi \in \vee\text{-}t\text{-}RULTL$.

The formulas φ, ψ of Example 6 are $\vee\text{-}t\text{-}RULTL$ -formulas.

5 An Algorithm for Reducing $wLTL$ -formulas to \bigcirc - normal form

In this section, we define a deterministic recursive algorithm that takes as an input the array representation of a formula $\varphi \in \vee\text{-}t\text{-}RULTL$, and array AP of atomic propositions, and outputs the array representation of an equivalent to φ , $\vee\text{-}t\text{-}RULTL$ formula $\varphi_{\bigcirc\text{-}re}$ in \bigcirc -normal form.

► **Definition 8.** (a) Let $\varphi \in wLTL$, and $\psi \in Cl(\varphi)$ such that $\psi = \bigcirc(\xi * \zeta)$, or $\psi = \bigcirc(\Box\xi)$, or $\psi = \bigcirc k$ where $*$ $\in \{\wedge, \vee, U\}$, $\xi, \zeta \in wLTL$, and $k \in K$. We let $[\varphi \leftarrow \psi_{\equiv_1}]$ denote the set of $wLTL$ -formulas defined in the following way:

$$[\varphi \leftarrow \psi_{\equiv_1}] = \left\{ \tilde{\varphi} \in wLTL \mid \begin{array}{l} \exists p \in Pos(T(\varphi)) \text{ such that} \\ T(\varphi)|_p = T(\psi), \text{ and } T(\tilde{\varphi}) = T(\varphi)[T(\psi_{\equiv_1})]_p \end{array} \right\}.$$

(b) We define the reduction relation $(wLTL, \equiv_1)$ as follows: for every $(\varphi, \tilde{\varphi}) \in wLTL \times wLTL$, $(\varphi, \tilde{\varphi}) \in \equiv_1$ iff $\tilde{\varphi} \in [\varphi \leftarrow \psi_{\equiv_1}]$ for some $\psi \in Cl(\varphi)$ such that $\psi = \bigcirc(\xi * \zeta)$, or $\psi = \bigcirc(\Box\xi)$, or $\psi = \bigcirc k$ where $*$ $\in \{\wedge, \vee, U\}$, $\xi, \zeta \in wLTL$, and $k \in K$.

We let \equiv_1^R (resp. \equiv_1^*) denote the reflexive (resp. transitive and reflexive) closure of \equiv_1 .

► **Example 9 (continued).** Let $\varphi = \bigcirc((3 \wedge (aUb)) U (2 \wedge (\bigcirc c))) \in wLTL$ over K_2 and $AP = \{a, b, c\}$. It holds $[\varphi \leftarrow \varphi_{\equiv_1}] = \{(\bigcirc(3 \wedge (aUb))) U (\bigcirc(2 \wedge (\bigcirc c)))\}$. We set $\tilde{\varphi} = (\bigcirc(3 \wedge (aUb))) U (\bigcirc(2 \wedge (\bigcirc c)))$. Then, for $\psi = \bigcirc(2 \wedge (\bigcirc c)) \in Cl(\tilde{\varphi})$, we have $[\tilde{\varphi} \leftarrow \psi_{\equiv_1}] = \{(\bigcirc(3 \wedge (aUb))) U ((\bigcirc 2) \wedge (\bigcirc(\bigcirc c)))\}$.

► **Lemma 10.** Let K be a TG -(ω, \leq)-valuation monoid and AP a finite set of atomic propositions. For every $(\varphi, \tilde{\varphi}) \in \equiv_1^*$ the following statements are true: (i) $nest(\varphi) \geq nest(\tilde{\varphi})$, (ii) for every $*$ $\in \{\wedge, \vee, U, \Box\}$, $mdist(\varphi, \bigcirc, *) \geq mdist(\tilde{\varphi}, \bigcirc, *)$, and (iii) $|\tilde{\varphi}| \leq |\varphi|^2 + |\varphi|$.

In the sequel, in the context of our pseudocode, for $\varphi, \psi \in wLTL$, the assignment $\varphi \leftarrow \psi$ stands for the definition of $Array(\varphi)$ and for setting it equal to $Array(\psi)$. We note that given $Array(\varphi)$ (of a $wLTL$ -formula φ), we can effectively determine $Array(Left(\varphi))$ (resp.

$Array(Right(\varphi))$ in time cubic in $|\varphi|$, $mdist(\varphi, \bigcirc, *)$ (where $*$ is a binary operator, or an always operator) in time polynomial in $|\varphi|$, and $Array(\star\varphi)$ (where \star is a next, or always operator) in time linear in $|\varphi|$. Finally, given $Array(\varphi)$, $Array(\psi)$ (of $wLTL$ -formulas φ, ψ), we can effectively determine $Array(\varphi \star \psi)$ (for a binary operator \star of weighted LTL) in time quadratic in $|\varphi| + |\psi|$.

► **Algorithm 1.** $\varphi_{\bigcirc-re}$

Input: $Array(\varphi)$ where $\varphi \in \mathcal{V}\text{-}t\text{-}RULTL$, array AP

Output: Array representation of equivalent to φ formula $\varphi_{\bigcirc-re} \in \mathcal{V}\text{-}t\text{-}RULTL$ in \bigcirc -normal form

```

1.  $MaxDistSum \leftarrow \sum_{* \in \{U, \wedge, \vee, \square\}} mdist(\varphi, \bigcirc, *)$ 
2. If  $MaxDistSum = 0$ 
3.   If  $EXOP(\varphi) = \bigcirc$  and  $EXOP(Left(\varphi)) = null$  and  $Left(\varphi) \notin AP$ 
4.      $\varphi_{\bigcirc-re} \leftarrow Left(\varphi)$ 
5.   else
6.      $\varphi_{\bigcirc-re} \leftarrow \varphi$ 
7.   end if
8. else
9.   If  $EXOP(\varphi) = \bigcirc$ 
10.    If  $EXOP(Left(\varphi)) = U$ 
11.       $\varphi_{\bigcirc-re} \leftarrow (\bigcirc(Left(Left(\varphi))))_{\bigcirc-re} U (\bigcirc(Right(Left(\varphi))))_{\bigcirc-re}$ 
12.    else if  $EXOP(Left(\varphi)) = \vee$ 
13.       $\varphi_{\bigcirc-re} \leftarrow (\bigcirc(Left(Left(\varphi))))_{\bigcirc-re} \vee (\bigcirc(Right(Left(\varphi))))_{\bigcirc-re}$ 
14.    else if  $EXOP(Left(\varphi)) = \wedge$ 
15.       $\varphi_{\bigcirc-re} \leftarrow (\bigcirc(Left(Left(\varphi))))_{\bigcirc-re} \wedge (\bigcirc(Right(Left(\varphi))))_{\bigcirc-re}$ 
16.    else if  $EXOP(Left(\varphi)) = \square$ 
17.       $\varphi_{\bigcirc-re} \leftarrow \square \left( (\bigcirc Left(Left(\varphi)))_{\bigcirc-re} \right)$ 
18.    else if  $EXOP(Left(\varphi)) = \bigcirc$ 
19.       $\varphi_{\bigcirc-re} \leftarrow \left( \bigcirc \left( (Left(\varphi))_{\bigcirc-re} \right) \right)_{\bigcirc-re}$ 
20.    end if
21.  else if  $EXOP(\varphi) = U$ 
22.     $\varphi_{\bigcirc-re} \leftarrow (Left(\varphi))_{\bigcirc-re} U (Right(\varphi))_{\bigcirc-re}$ 
23.  else if  $EXOP(\varphi) = \vee$ 
24.     $\varphi_{\bigcirc-re} \leftarrow (Left(\varphi))_{\bigcirc-re} \vee (Right(\varphi))_{\bigcirc-re}$ 
25.  else if  $EXOP(\varphi) = \wedge$ 
26.     $\varphi_{\bigcirc-re} \leftarrow (Left(\varphi))_{\bigcirc-re} \wedge (Right(\varphi))_{\bigcirc-re}$ 
27.  else if  $EXOP(\varphi) = \square$ 
28.     $\varphi_{\bigcirc-re} \leftarrow \square \left( (Left(\varphi))_{\bigcirc-re} \right)$ 
29.  end if
30. end if
31. return  $\varphi_{\bigcirc-re}$ 

```

► **Theorem 11.** (i) Let $\varphi \in \mathcal{V}\text{-}t\text{-}RULTL$. Given $Array(\varphi)$, and array AP as input in Algorithm $\varphi_{\bigcirc-re}$, the algorithm terminates, and outputs $Array(\varphi_{\bigcirc-re})$ such that $\varphi_{\bigcirc-re} \in \mathcal{V}\text{-}t\text{-}RULTL$ is in \bigcirc -normal form, $\varphi_{\bigcirc-re} \equiv \varphi$, and $(\varphi, \varphi_{\bigcirc-re}) \in \stackrel{\star}{\equiv}_1$.

(ii) Let $\varphi \in \mathcal{V}\text{-}t\text{-}RULTL$. $Array(\varphi_{\bigcirc-re})$ can be determined in time $\mathcal{O}(|\varphi|^k |AP|)$.

References

- 1 Panayiotis Bozaris. *Data Structures*. Tziola Publications, Thessaloniki, Greece, 3rd edition, 2022.
- 2 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. In Michael Kaminski and Simone Martini, editors, *Computer Science Logic, 22nd International Workshop, CSL 2008, 17th Annual Conference of the EACSL, Bertinoro, Italy, September 16-19, 2008. Proceedings*, volume 5213 of *Lecture Notes in Computer Science*, pages 385–400. Springer, 2008. doi:10.1007/978-3-540-87531-4_28.
- 3 Manfred Droste and Paul Gastin. Weighted automata and weighted logics. *Theor. Comp. Sci.*, 380(1-2):69–86, 2007.
- 4 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata, Monographs in Theoretical Computer Science. An EATCS Series*.
- 5 Manfred Droste and Ingmar Meinecke. Weighted automata and weighted MSO logics for average and long-time behaviors. *Inf. Comput.*, 220:44–59, 2012. URL: <https://doi.org/10.1016/j.ic.2012.10.001>, doi:10.1016/J.IC.2012.10.001.
- 6 Gunawardena Jeremy. An introduction to idempotency. *Cambridge University Press, Publications of the Newton Institute*, pages 1–49, 1998.
- 7 Eleni Mandrali. A translation of weighted LTL formulas to weighted Büchi automata over ω -valuation monoids. *Sci. Ann. Comput. Sci.*, 31(2):223–292, 2021. doi:10.7561/SACS.2021.2.223.
- 8 Eleni Mandrali. Describing weighted safety with weighted LTL over product ω -valuation monoids. *Sci. Ann. Comput. Sci.*, 33(2):93–157, 2023. doi:10.7561/SACS.2023.2.93.
- 9 Irini-Eleftheria Mens and George Rahonis. Variable tree automata over infinite ranked alphabets. In Franz Winkler, editor, *Algebraic Informatics - 4th International Conference, CAI 2011, Linz, Austria, June 21-24, 2011. Proceedings*, volume 6742 of *Lecture Notes in Computer Science*, pages 247–260. Springer, 2011. doi:10.1007/978-3-642-21493-6_16.