Weighted Automata and Logics on Hierarchical Structures and Graphs

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

DISSERTATION

zur Erlangung des akademischen Grades

DOCTOR RERUM NATURALIUM
(Dr. rer. nat.)

im Fachgebiet
Informatik

vorgelegt von
Dipl.-Math. Stefan Dück
geboren am 21. Juli 1988 in Leipzig

Die Annahme der Dissertation wurde empfohlen von

1. Prof. Dr. Manfred Droste (Leipzig)
2. Prof. Dr. Frank Drewes (Umeå)

Die Verleihung des akademischen Grades erfolgt mit
Bestehen der Verteidigung am 21.11.2017 mit dem
Gesamtprädikat *magna cum laude*.

Contents

Chapter 1

Introduction

Classical formal language theory goes back as far as 1956, to Noam Chomsky [Cho56]. It was originally introduced to model our natural spoken languages and to understand the structure and, consequently, the properties of languages. Nowadays, the use of formal languages also includes the definition and visualization of programming languages, and the examination and verification of algorithms and systems. Formal languages are not only used to analyze the computability and complexity of given problems and algorithms, they are also instrumental in proving the correct behavior of automated systems, e.g., to avoid that a flight guidance system navigates two airplanes too close to each other.

As we will see in the following, one ingredient of this success story is the comparatively easy definition of a formal language. However, ultimately, this vast field of applications is built upon a very well investigated and coherent theoretical basis. It is the goal of this dissertation to add to this theoretical foundation and to explore ways to make formal languages and their models more expressive. More specifically, we are interested in models which are able to handle *quantitative* information over very general structures, which we sketch in the following.

A formal *language* is a set of *words*, and every word is a concatenation of symbols over a given set, called *alphabet*. By enforcing certain rules or patterns into this set of words, we can describe different languages with varying complexity. Often, such a word is seen as a possible sequence of events and we ask how a given system or *automaton* reacts to one particular of all the possible chains of events. While this representation very intuitively models sequential events, the enforced linear structure of words ("one symbol after another") can also be limiting. For example, it may be essential to not only look at one possible sequence, but at a branching *tree* of possible outcomes. Another example can be found in programming languages, where the procedure *calls* and *returns* of a program induce a certain hierarchy. Such hierarchical information can naturally be embedded into a word with an additional hierarchical structure. In a more

general setting, we may be interested not only in modeling chronological events, but instead ask how a system reacts to a whole given environment. Often, such an environment can be naturally modeled as a graph instead of as a linear word. In this work, we consider automata over both types of structures; words with additional hierarchical information as well as general graphs.

Another price to pay for the power of classical formal language theory is that a language is purely qualitative: a word is either contained in a language or it is not. For example, we have no way of counting desirable or undesirable patterns in a word, and we have no way of assessing how close to or how far away a non-language word is from a language (e.g. whether it is only a simple typing error). Similarly, when a system reads a sequence of events or a complete situation given as graph, it answers only with "accepted" or "not accepted". Such a system can neither count nor calculate and when considering the input of the system as a possible solution, we can only measure it in terms of 0 or 1.

To tackle the restriction to qualitative statements, various models of quantitative systems have been studied in the literature. Weighted automata were introduced by Schützenberger [Sch61] over semirings like $(\mathbb{N}, +, \cdot, 0, 1)$ and soon developed a flourishing theory, cf. the books [BR88, Eil74, KS86, SS78] and the handbook [DKV09]. Such a system, for example, can not only tell whether an error occurred, but also how many of them. We can answer questions like "How often does an event appear?", "What is the cost of this solution?", or "What is the distance of this word to a given language?" (see e.g. [Moh03] for the last question).

Quantitative automata modeling the long-time average or discounted behavior of systems were investigated by Chatterjee, Doyen, and Henzinger [CDH08, CDH09]. The insight that many previous results can be extended to weight structures other than semirings motivated the introduction of *valuation monoids* [DM12], which formalize non-semiring behaviors like average or discounting but also incorporate classical semirings.

Another very powerful tool to describe languages is *monadic second order logic (MSO logic)*. In their seminal papers, Büchi, Elgot, and Trakhtenbrot [Büc60, Elg61, Tra61] independently proved that the languages recognizable by finite automata are exactly the languages definable by MSO logic. This fundamental result not only shows the decidability of MSO logic over words, but also led to multiple extensions covering e.g. finite and infinite trees [Don70, Rab69, TW68], traces [Tho90, DR95], pictures [GRST96], nested words [AM09], texts [HtP97], and graphs [Tho91]. Weighted logics were introduced by Droste and Gastin in [DG07]. There, the authors proved a respective connection between weighted automata and weighted logic for words. This in turn inspired several extensions e.g. to infinite words [DR06], trees [DV06, DGMM11], traces [Mei06], pictures [Fic11], nested words [Mat10b], texts [Mat10a], and pushdown automata with general storage types [VDH16].

However, notably, a general result for weighted automata and weighted logics covering graphs and linking the previous results remained, up to now, open.

In this thesis, we apply semirings and valuation monoids to define and characterize weighted automata and weighted logics which are able to read and rate structures with hierarchical information, and weighted automata and weighted logics operating on graphs. In the following, orientated on the structure of this dissertation, we give the background and the details of our approaches and specify our contributions.

In Chapter 2, we start with background on the theories used thesis. We recapitulate classical MSO logic and Büchi's theorem for words. We introduce *relational structures*, which can be employed to describe finite and infinite graphs. Graphs cover, in particular, words, trees, nested words, pictures, and many other structures. We restate a very useful and fundamental result of Hanf [Han65] connecting first order sentences of MSO logic with counting occurrences of local patterns. We conclude the chapter by studying the following extension of this result based on work of Bollig and Kuske [BK07] on the logic FO^{∞}. This first order logic features an additional existential quantifier that expresses that there are infinitely many positions satisfying a formula. In a context different from ours, namely for Muller message-passing automata, Bollig and Kuske were able to formulate an extended Ehrenfeucht-Fraïssé [Fra54, Ehr61] game to develop a Hanf-like theorem for this logic. We show how this result can be applied to infinite graphs to replace a sentence of FO^{∞} with an equivalent expression that counts occurrences of local patterns and is able to distinguish between a finite and an infinite number of occurrences.

In Chapter 3, we introduce the weights and weight structures that are used to assign quantitative values to relational structures. In addition to classical semirings, we define *valuation monoids*, a very general weight structure which incorporates average and discounted computations of weights. Valuation monoids are particularly useful in applying weights to infinite structures. Following the approach of Droste and Gastin [DG07] for words, we introduce a general weighted MSO-logic for relational structures (see also [Mat09]). We study different, suitable fragments of this weighted logic and show a transformation result between them, which provides us with a 'normal-form' of weighted formulas. Furthermore, this Transformation Theorem establishes a direct connection between the assumptions on our weight structure and the expressive power of the weighted MSO-logic over this weight structure.

In Chapter 4 and Chapter 5, we study weighted automata over hierarchical structures. In Chapter 4, we study *nested words*, which were introduced by Alur and Madhusudan [AM09], and are used to capture structures with both linear and hierarchical order like XML documents. Alur and Madhusudan also introduced automata and equivalent logics for both finite and infinite nested words, thus extending Büchi's theorem to nested words. They proved that the languages recognizable by nested word automata correspond to the

class of *visibly pushdown languages (VPL)* and have valuable closure properties comparable to those of *regular languages*. Here, using valuation monoids, we introduce and investigate weighted automata models and weighted MSO logics for infinite nested words. In the main result of Chapter 4, we show a Büchi-like result connecting weighted automata and weighted MSO logic for infinite nested words. Applying the logic Transformation Theorem of Chapter 3 to nested words, we obtain, depending on the assumptions on the valuation monoid, three characterizations of regular weighted nested word languages in the form of successively more expressive weighted logics.

In Chapter 5, we study (semiring-) *weighted operator precedence languages* (wOPL) for finite words. In the last years, renewed investigation of OPL led to the discovery of important properties of this class: OPL are closed with respect to all boolean operations and can be characterized, besides the original grammar family, in terms of an automata family and an MSO logic. Furthermore, they significantly generalize the class of VPL. This chapter can, therefore, also be seen as an extension of previous works considering finite nested words [Mat10b, DP14b]. We introduce *weighted operator precedence automata* (wOPA) and show how they are both strict extensions of OPA and weighted visibly pushdown automata. We prove a Nivat-like result which shows that quantitative OPL can be described by unweighted OPA and very particular weighted OPA. In a Büchi-like theorem, we show that weighted OPA are expressively equivalent to a weighted MSO-logic for OPL. Note that operator precedence automata feature pop operations which do not consume a symbol. We were able to prove that for arbitrary semirings, wOPA with weights at pop transitions are strictly more expressive than wOPA with only trivial weights at pop transitions. For commutative semirings, however, weights on pop transitions do not increase the expressive power of the automata.

In Chapter 6, we study weighted languages of finite and infinite graphs. We introduce a general model of weighted automata operating on finite graphs. This model provides a quantitative version of Thomas' unweighted model of graph acceptors [Tho91] and is a direct extension of both this unweighted model and the weighted automata investigated for various other structures, like words, trees, pictures, and nested words. We derive a Nivat theorem for weighted graph automata, which shows that their behaviors are precisely those obtainable from very particular weighted graph automata and unweighted graph acceptors. We also show that weighted MSO logic over graphs is expressively equivalent to weighted graph automata. As a consequence, we obtain Büchi-type equivalence results known from the recent literature for weighted automata and weighted logics on words, trees, pictures, and nested words. Establishing such a general result has been an open problem in weighted logics for some time.

Finally, we study unweighted and weighted automata for infinite graphs. This model features an acceptance condition which is able to distinguish infinitely many occurrences from finitely many ones. Using proof ideas from

Thomas [Tho96] and the result from Chapter 2, the latter relying on Bollig and Kuske's extension of first order logic [BK07], we show that graph acceptors and EMSO^{∞}-logic for infinite graphs are equally expressive. Using valuation monoids which naturally assign weights to infinite structures, we are then able to lift the Büchi-characterization of weighted graph automata to infinite graphs.

Note that although Chapter 6 is strongly connected to Chapter 4, the direct restriction of weighted graph automata to word, tree, and in particular nested word automata is shown only in the finite case and therefore does not cover infinite nested words. While we conjecture that such a direct connection can also be shown for infinite structures, this is one of the future problems to attack. Furthermore, graph automata can only read graphs with bounded vertex-degree, but the structures considered by operator precedence languages in Chapter 5 in general admit no such bound.

Chapter 2

Foundations

In the following, we introduce general concepts used throughout the whole thesis. We shortly recapitulate finite automata and Büchi's theorem for words. Then, we introduce relational structures, with a focus on graphs and special classes of graphs. We define classical MSO over relational structures. Finally, we show our first result, which is a corollary of an extension of Hanf's theorem by Bollig and Kuske [BK07]. This result will be used in Chapter 6 for infinite graphs.

We denote by $\mathbb{N}$ the natural numbers starting with 0. We set $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. We denote disjoint unions by $\uplus$.

2.1 Finite Word Automata and Büchi's Theorem

In its core, formal language theory is the study of patterns. A *word* is an abstract form of describing a sequence of possible events. We analyze structural rules of how these words are built up and how they form sets, called *languages*, often sharing certain characteristics. The four most established ways to formulate these rules and therefore to define languages are the following. We can describe a set of words by a *rational expression* – which describes the patterns directly –, by a finite state machine, an *automaton*, or a finite set of rules, a *regular grammar* – which are both a form of a build-up plan for the language –, and *logical sentences of an MSO logic* – which describe the rules in a formalism close to our natural language –. The classes of languages expressibly by these mechanism are usually called *rational*, *recognizable*, *regular*, or *definable* languages, respectively, and in fact describe the same class. In other words, all these different mechanisms are equally powerful.

The connection between MSO logic and automata is due to Büchi, Elgot, and Trakhtenbrot [Büc60, Elg61, Tra61] and has proven most fruitful in formal language theory. In the following, we shortly recapitulate the classical definitions and the Büchi-Elgot-Trakhtenbrot theorem, also called Büchi's theorem.

Formally, let Σ be a set and $n \in \mathbb{N}$. A *(finite) word* w over Σ is a sequence $w = a_1 a_2 ... a_n$ of symbols of Σ. We call n the *length* of w. If w has length 0, we call it the *empty word*, denoted by ε. We denote by Σ^* the set of all words over Σ, and by Σ^+ the set of all non-empty words over Σ. An infinite word $w = a_1 a_2 ...$ is an infinite sequence of Σ. We denote by Σ^ω the set of all infinite words over Σ. We call a subset L of Σ^* a *language* (of finite words) over Σ.

In the following, Σ will always be a finite set, called an alphabet.

Definition 2.1. A *finite (deterministic) automaton* over Σ is a quadruple $\mathcal{A} = (Q, q_I, \delta, F)$ consisting of

- a finite set of states Q,
- an initial state $q_I \in Q$,
- the transition function $\delta : Q \times \Sigma \to Q$,
- a set of accepting states $F \subseteq Q$.

A run r of a finite automaton $\mathcal{A}$ on a word $w = a_1 a_2 ... a_n$ is a sequence of states $r = (q_0, q_1, ..., q_n)$ such that $q_i = \delta(q_{i-1}, a_i)$ for all $0 < i \leq n$. A run is called *accepting* if $q_0 = q_I$ and $q_n \in F$. A word w is accepted by $\mathcal{A}$ if $\mathcal{A}$ has an accepting run on w. We define $L(\mathcal{A})$, the *language accepted by* $\mathcal{A}$ as the set of all words accepted by $\mathcal{A}$. A language L is called *recognizable* if there exists an automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L$.

Now, we introduce the second concept to describe languages, namely a classical *monadic second order logic* over words. We denote by $x, y, ...$ first-order variables ranging over positions and by $X, Y, ...$ second order variables ranging over sets of positions. We define the formulas of $\mathrm{MSO}(\Sigma^*)$ inductively by the following grammar

$$\varphi ::= \mathrm{Lab}_a(x) \mid x \leq y \mid x = y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \exists X.\varphi$$

where $a \in \Sigma$ and $\mathrm{Lab}_a(x)$ describes that a position of a word is labeled with the symbol a.

Then, the semantics, i.e., the satisfaction relation $\models$ is defined as usual, compare the full semantics for relational structures in Section 2.3. We call φ a *sentence* if φ contains no free variables. We define $L(\varphi)$, the *language of* φ as the set of all words w such that $w \models \varphi$. We call a language L *(MSO-) definable* if $L = L(\varphi)$ for some MSO-sentence φ. Then, the fundamental result of Büchi, Elgot, and Trakhtenbrot [Büc60, Elg61, Tra61] is the following.

Theorem 2.2. *A language of words is recognizable if and only if it is definable.*

2.2 Relational Structures and Graphs

In this section, we introduce the general concept of relational structures. Here, our focus is on graphs and special classes of graphs like words, trees, pictures,

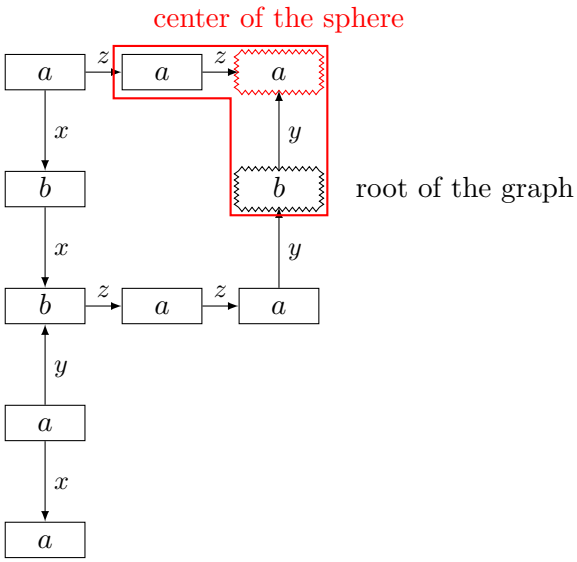


Figure 2.1: A graph of $\mathrm{pDG}_3(\{a,b\},\{x,y,z\})$ together with one of its 1-spheres.

nested words and a new structure that we call *precedence words*, which arises later from studying operator precedence languages.

Relational structures

A *(relational) signature* (σ, rk) consists of a set of relation symbols σ together with an *arity* for every symbol $\mathrm{rk} : \sigma \to \mathbb{N}$. We define a *$\sigma$-structure* (also *structure* or *relational structure* if the context is clear) as $s = (V(s), (R_s)_{R\in\sigma})$ consisting of a set $V(s)$, the *domain*, and a relation R_s of arity $\mathrm{rk}(R)$ for every $R \in \sigma$. We write V for $V(s)$ and R for R_s if the context is clear.

We say two σ-structures $s = (V(s), (R_s)_{R\in\sigma})$ and $s' = (V(s'), (R_{s'})_{R\in\sigma})$ are *isomorphic* if there exists a bijective map φ from $V(s)$ to $V(s')$ such that for all $R \in \sigma$ and for all $x_1, ..., x_{\mathrm{rk}(R)} \in V(s)$, we have $(x_1, ..., x_{\mathrm{rk}(R)}) \in R_s$ if and only if $(\varphi(x_1), ..., \varphi(x_{\mathrm{rk}(R)})) \in R_{s'}$. In this case, we call φ an *isomorphism*.

In the following, we distinguish σ-structures only up to isomorphism. Furthermore, we only consider structures s where $V(s)$ is a subset of the natural numbers $\mathbb{N}$. We let C^σ be a fixed class of such σ-structures.

Graphs

Following [Tho96] and [DD15], we define a *(directed, unpointed) graph* over two finite alphabets A and B as $G = (V, (\mathrm{Lab}_a)_{a\in A}, (E_b)_{b\in B})$, where V is a non-empty set of *vertices*, the sets Lab_a $(a \in A)$ form a partition of V, and the sets E_b $(b \in B)$ are pairwise disjoint irreflexive binary relations on V, called

edges. We denote by $E = \bigcup_{b \in B} E_b$ the set of all edges. Then the elements of A are the vertex labels, and the elements of B are the edge labels. Further, every graph is a σ-relational structure for $\sigma = \{\mathrm{Lab}_a \mid a \in A\} \cup \{E_b \mid b \in B\}$, $\mathrm{rk}(\mathrm{Lab}_a) = 1$ for all $a \in A$, and $\mathrm{rk}(E_b) = 2$ for all $b \in B$. We denote by $\mathrm{Lab}_G(v)$ the label of the vertex v of the graph G.

The *in-degree* (resp. *out-degree*) of a vertex v is the number of edges ending (resp. beginning) in v. The *degree* of v is the in-degree plus the out-degree of v. A graph is *bounded by* t if every vertex has a degree smaller than or equal to t. We denote by $\mathrm{DG}_t(A, B)$ the class of all finite directed graphs over A and B bounded by t.

We call a class of graphs *pointed* if every graph G of this class is *pointed with a vertex* v, i.e., it has a uniquely designated vertex $v \in V$. Formally, a *pointed graph* (G, v) is a graph G together with a unary relation *root* such that $\mathrm{root} = \{v\}$. We denote by $\mathrm{pDG}_t(A, B)$ the class of all finite, directed, and pointed graphs over A and B, bounded by t.

Let G be a graph G and r a natural number. For two vertices u and v, we say the *distance* of u and v is smaller than or equal to r, denoted by $\mathrm{dist}(u, v) \leq r$, if there exists a path $(u = u_0, u_1, ..., u_j = v)$ with $j \leq r$ and $(u_i, u_{i+1}) \in E$ or $(u_{i+1}, u_i) \in E$ for all $i < j$. If there exists no such path, we say the distance between u and v is infinite. A graph is *connected* if there exists no pair of vertices with an infinite distance. Note that here the distance and the connectivity of graphs do not take the orientation of edges into account.

We consider subgraphs around a vertex v of an unpointed graph G as follows. We denote by $\mathrm{sph}^r(G, v)$, the *r-sphere of G around the vertex v*, the unique subgraph of G pointed with v and consisting of all vertices with distance to v smaller than or equal to r, together with their edges. We call (H, v) an *r-tile* if $(H, v) = \mathrm{sph}^r(H, v)$.

Intuitively, subgraphs and r-tiles of a pointed graph (G, u) are defined analogously. However, in this case, we have to take care of the root u as follows. We call (H, v, w) a *pointed r-tile* if (H, v) is an r-tile and either w is an additional distinguished vertex of H or we write $w = \mathit{empty}$. Then, we denote by $\mathrm{sph}^r((G, u), v)$ the unique pointed r-tile (H, v, w) s.t. $(H, v) = \mathrm{sph}^r(G, v)$ and $w = u$ if $u \in \mathrm{sph}^r(G, v)$ and $w = \mathit{empty}$, otherwise.

Note that in $\mathrm{DG}_t(A, B)$, resp. $\mathrm{pDG}_t(A, B)$, there exist only finitely many pairwise non-isomorphic r-tiles, resp. pointed r-tiles. Given a tile $\tau = (H, v)$ or a pointed tile $\tau = (H, v, w)$, we say v is the *center* of τ. An example of a pointed graph together with one of its 1-spheres can be seen in Figure 2.1. We may omit the explicit root u of a graph and the radius r of a tile if the context is clear. While many of our results hold for both pointed and unpointed graphs, the pointing is crucial for some results and is useful for some of the considered valuation monoids, especially for infinite graphs.

We call a graph $G = (V, (\mathrm{Lab}_a)_{a \in A}, (E_b)_{b \in B})$ *infinite* if V is infinite. We denote by $\mathrm{pDG}_t^\infty(A, B)$ the class of all infinite, directed, and pointed graphs

Figure 2.2: The word $aabb$ as a labeled graph of $\mathrm{DG}_2(\{a,b\},\{1\})$ on the left. We omitted the edge labels which are all 1. As we distinguish relational structures only up to isomorphism, the actual vertex names $\{1,2,3,4\}$ can also be omitted, which results in the usual representation of this word on the right.

over A and B, bounded by t. Note that pointed r-tiles of infinite graphs are finite structures, and there still exist only finitely many non-isomorphic different pointed r-tiles since the degree of every considered infinite graph is bounded by t.

Special Classes of Graphs

In the following, we give examples of specific relational structures, namely words, trees, pictures, nested words, and a new structure we call *precedence words*. Note that all the following structures can be naturally introduced as finite and as infinite structures. Here, we choose the respective version that will be used throughout the thesis. Both finite and infinite version of these structures are special instances of pointed and connected graphs.

In contrast to graphs, the following structures are defined over only one alphabet. In our interpretation of these structures as graphs, this is the vertex alphabet A. We will see that the respective edge alphabet is either trivial or predefined by the structure itself (e.g. for pictures it is $\{1,2\}$ to distinguish between vertical and horizontal edges). To stress this difference, the following structures are introduced over a finite alphabet Σ instead of A.

Words

A word $w = a_1...a_n$ over Σ can also be regarded as a relational structure $(\{1,...,n\},(\mathrm{Lab}_a)_{a\in\Sigma},S)$, where s is the usual successor relation: $S = \{(i,i{+}1) \mid 1 \leq i \leq n-1\}$. That is, a word of length n is a graph in $\mathrm{DG}_2(\Sigma,\{1\})$ as seen in Figure 2.2 consisting of n labeled vertices together with unlabeled edges going from the left most position to the end of the word. If not given explicitly, the root of a word can be uniquely described by its first position, which is the only position without any incoming edges.

Sometimes words are introduced together with the order relation $\leq$ instead of the successor relation S; however, this does not fit our context as it would yield graphs without a bound on the vertex-degree.

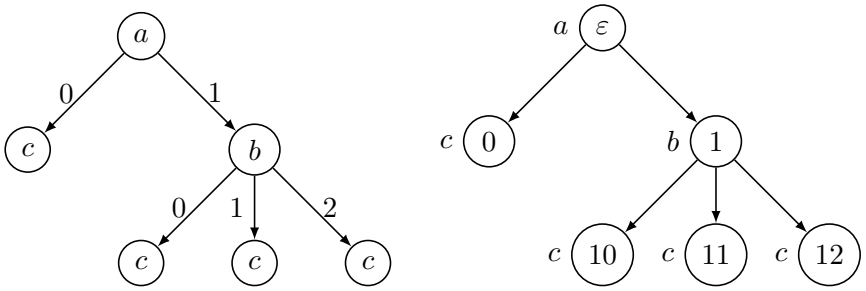


Figure 2.3: Left: A tree as a graph of $\mathrm{DG}_4(\Sigma, \{0,1,2\})$, $\Sigma = \{a, b, c\}$, with labels inside the nodes. Right: The same tree in a classical representation, with essential node names.

Trees

We introduce the set of *trees* as a special class of graphs as follow. A *directed path* from vertex u to vertex v, is a path $(u = u_0, u_1, ..., u_j = v)$ with $j > 0$ and $(u_i, u_{i+1}) \in E$ for all $i < j$. Then, a *tree* is a pointed graph (G, u) of $\mathrm{pDG}_t(A, B)$ such that the in-degree of u is 0, the out-degree of u is at most $t-1$ and for every vertex $v \neq u$, the in-degree of v is 1 and there exists a directed path from u to v.

As usual, in the following, we consider all our trees to be *ordered*, that is, the edge alphabet is given by $B = \{1, .., t-1\}$ and for every vertex v with out-degree d, the outgoing edges of v are uniquely labeled with $1, ..., d$.

Note that every word is a tree and as for words, we may omit the explicit root of a tree, as it is the uniquely defined vertex without incoming edges.

It is also noteworthy that as subsets of $\mathrm{DG}_t(A, B)$ our trees are rank bounded by $t-1$. Therefore, there exists an alphabet partitioning, which transforms every tree into a *ranked tree*, i.e., a tree over a *ranked alphabet* were every symbol has a predefined number of successors. In the context of tree automata, ranked trees are often defined over a ranked alphabet Σ with a non-empty, prefix-closed subset of $\mathbb{N}^*$ which also respects the rank of its labels. Then every such ranked tree is a graph of $\mathrm{DG}_{r+1}(\Sigma, \{1, ..., r\})$, where r is the maximum rank of Σ. See Figure 2.3 for an example of a tree represented as a graph and in a classical representation over a prefix-closed domain.

Pictures or Two-dimensional Words

Intuitively, a *two-dimensional word*, also called *picture* or *grid*, is a matrix over an alphabet Σ. More precisely, a picture is a relational structure $P = (\{1, ..., n\} \times \{1, ..., m\}, (\mathrm{Lab}_a)_{a \in \Sigma}, S_1, S_2)$, where $S_1 = \{((i,j),(i+1,j)) \mid 1 \leq i \leq n-1, 1 \leq j \leq m-1\}$ and $S_2 = \{((i,j),(i,j+1)) \mid 1 \leq i \leq n-1, 1 \leq j \leq m-1\}$. A picture can be seen as rooted in $(1,1)$ and every picture can be

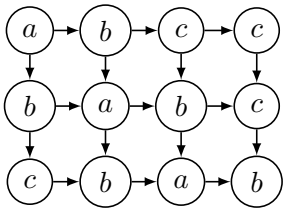


Figure 2.4: A picture of $\mathrm{DG}_4(\Sigma, \{1,2\})$, $\Sigma = \{a,b,c\}$, of *height* 3 and *width* 4. Every horizontal edge is implicitly marked with 1 and every vertical edge with 2.

depicted as a graph of $\mathrm{DG}_4(\Sigma, \{1,2\})$, see Figure 2.4 for an example.

Infinite Nested Words

Informally, an infinite nested word is an infinite word with an additional hierarchical structure. Formally, nested words and their regular languages, also known as *visibly pushdown languages*, were introduced by Alur and Madhusudan in [AM09]. They find their origins in *input-driven languages* [Meh80, vBV83, LST94]. While not given explicitly in the following, finite nested words can be defined analogously to the following.

Definition 2.3. A *matching relation* ν over $\mathbb{N}_+$ is a subset of $(\{-\infty\} \cup \mathbb{N}_+) \times (\mathbb{N}_+ \cup \{\infty\})$ such that:

(i) $\nu(i,j) \Rightarrow i < j$,

(ii) $\forall i \in \mathbb{N}_+ : |\{j : \nu(i,j)\}| \leq 1 \wedge \ |\{j : \nu(j,i)\}| \leq 1$,

(iii) $\nu(i,j) \wedge \nu(i',j') \wedge i < i' \Rightarrow j < i' \vee j > j'$,

(iv) $(-\infty, \infty) \notin \nu$.

An *infinite nested word*, also *nested ω-word* over Σ is a pair $nw = (w, \nu) = (a_1 a_2 ..., \nu)$ where $w = a_1 a_2 ...$ is an ω-word over Σ and ν is a matching relation over $\mathbb{N}_+$. We denote by $\mathrm{NW}^\omega(\Sigma)$ the set of all nested ω-words over Σ, and we call every subset of $\mathrm{NW}^\omega(\Sigma)$ a *language of nested ω-words*.

If $\nu(i,j)$ holds, we call i a *call position* and j a *return position*. In case of $j = \infty$, i is a *pending call* otherwise a *matched call*. In case of $i = -\infty$, j is a *pending return* otherwise a *matched return*. Note that similar to [AM09], condition (iii) ensures that nestings of calls and returns are either disjoint or hierarchical; in particular, a position cannot be both a call and a return. If a position i is neither call nor return, then we say i is an *internal*. If a (finite or infinite) nested word contains no pending calls and no pending returns, we call it *well-matched*.

Together with the usual successor function S, a finite nested word can be written as a relational structure $(\{1, ..., n\}, (\mathrm{Lab}_a)_{a \in \Sigma}, S, \nu)$ and defines a graph

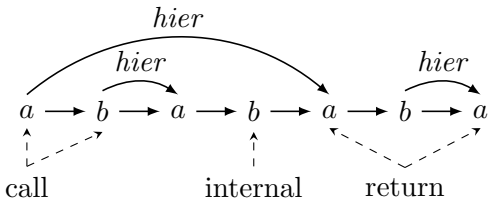


Figure 2.5: The graph representation of the finite nested word $(abababa, \{(1,5),(2,3),(6,7)\})$, which could also be written as $\langle a\langle ba\rangle ba\rangle\langle ba\rangle$. The linear edge labels *lin* were omitted.

of $\mathrm{DG}_3(\Sigma, \{lin, hier\})$, which distinguished between linear and hierarchical edges. Figure 2.5 depicts an example of a finite well-matched nested word. To handle pending edges, we need two more predicates, call and ret, describing also the calls, resp. returns without a matching position of $\mathbb{N}$. Then, a nested ω-word can be written as a relational structure $(\mathbb{N}, (\mathrm{Lab}_a)_{a\in\Sigma}, S, \mathrm{call}, \mathrm{ret}, \nu)$.

Precedence Words

In the following, we introduce *precedence words*, a structure which is closely related to nested words, but in particular features unbounded calls or returns at a position and is therefore not in any of the classes $\mathrm{DG}_t(A, B)$. Precedence words are closely related to the language class of *operator precedence languages* [CMM78], which was shown to be a strict extension of visibly pushdown languages [CM12] and still retains valuable closure properties. Furthermore, operator precedence languages can be characterized in terms of a grammar family, a class of pushdown automata, and an MSO-logic [LMPP15].

Precedence words and weighted operator precedence languages will be considered in detail in Chapter 5. They feature a hierarchical structure, which we call *chain relation* and denote by $\curvearrowright$. This new relation can be compared with the *nesting* or *matching relation* of [AM09], as it also is a non-crossing relation, going always forward and originating from additional information on the alphabet. However, it features significant differences: Instead of adding unary information to symbols, which partition the alphabet into three disjoint parts (calls, internals, and returns), we add a binary relation for every pair of symbols denoting their *precedence relation*. Therefore, in contrast to the nesting relation, the same symbol can be either call or return depending on its context. Furthermore, the same position can be part of multiple chain relations.

Following [CM12], we define an *OP alphabet* as a pair (Σ, M), where Σ is an alphabet and M, the *operator precedence matrix (OPM)*, is a $|\Sigma \cup \{\#\}|^2$ array describing for each ordered pair of symbols at most one precedence relation,

$M =$

	a	$+$	$\times$	$\#$
a		$\gtrdot$	$\gtrdot$	$\gtrdot$
$+$	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$
$\times$	$\doteq$			$\gtrdot$
$\#$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$

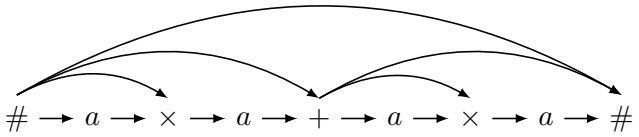


Figure 2.6: The graph representation of the precedence word $(a \times a + a \times a)$ over an OPM M over $\Sigma = \{a, +, \times\}$ together with its chain relation $\{(0,2),(4,6),(0,4),(4,8),(0,8)\}$). The edge labels were omitted.

that is, every entry of M is either $\lessdot$ (*yields precedence*), $\doteq$ (*equal in precedence*, $\gtrdot$ (*takes precedence*), or empty (no relation). Note that these relations are not required any order axioms.

Let $w = (a_1...a_n) \in \Sigma^+$ be a non-empty word and (Σ, M) an OP alphabet. We set $a_0 = a_{n+1} = \#$ and define a new relation $\curvearrowright$ on the set of all positions of $\#w\#$, inductively, as follows. Let $i, j \in \{0, 1, ..., n+1\}$, $i < j$. Then, we write $i \curvearrowright j$ if there exists a sequence of positions $k_1...k_m$ such that $i = k_1 < ... < k_m = j$, $a_{k_1} \lessdot a_{k_2} \doteq ... \doteq a_{k_{m-1}} \gtrdot a_{k_m}$, and either $k_s + 1 = k_{s+1}$ or $k_s \curvearrowright k_{s+1}$ for each $s \in \{1, ..., m-1\}$. In particular, $i \curvearrowright j$ holds if $a_i \lessdot a_{i+1} \doteq ... \doteq a_{j-1} \gtrdot a_j$.

We say w is a *precedence word* over (Σ, M) if w is not empty and for $\#w\#$ we have $0 \curvearrowright n+1$. By making the chain relation explicit, every precedence word induces a relational structure $(\{1, ..., n\}, (\mathrm{Lab}_a)_{a\in\Sigma}, S, \curvearrowright)$. An example for this representation of a precedence word together with its OPM can be seen in Figure 2.6.

2.3 MSO Logic on Relational Structures

Next, we introduce a classical *monadic second order logic* over relational structures. We denote by $x, y, ...$ first-order variables ranging over vertices and by $X, Y, ...$ second order variables ranging over sets of vertices. For a relation R, we write $R(x_1, ..., x_{\mathrm{rk}(R)})$ if $(x_1, ..., x_{\mathrm{rk}(R)}) \in R$. For a binary relation R, we may write xRy if $(x, y) \in R$. Given a signature σ, we define the formulas of $\mathrm{MSO}(\sigma)$ inductively by the following grammar

$$\varphi ::= R(x_1, ..., x_{\mathrm{rk}(R)}) \mid x = y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \exists X.\varphi$$

$$
\begin{array}{lll}
(s,\gamma) \models R(x_1,...,x_{\mathrm{rk}(R)}) & \text{iff} & R(\gamma(x_1),...,\gamma(x_{\mathrm{rk}(R)})) \\
(s,\gamma) \models x = y & \text{iff} & \gamma(x) = \gamma(y) \\
(s,\gamma) \models x \in X & \text{iff} & \gamma(x) \in \gamma(X) \\
(s,\gamma) \models x \vee y & \text{iff} & \gamma(x) \vee \gamma(y) \\
(s,\gamma) \models \neg\varphi & \text{iff} & (s,\gamma) \models \varphi \text{ does not hold} \\
(s,\gamma) \models \exists x.\varphi & \text{iff} & \text{there exists a } u \in V(s) \text{ s.t. } (s,\gamma[x \to u]) \models \varphi \\
(s,\gamma) \models \exists X.\varphi & \text{iff} & \text{there exists a } U \subseteq V(s) \text{ s.t. } (s,\gamma[X \to U]) \models \varphi
\end{array}
$$

Figure 2.7: Semantics of MSO(σ)

where $R \in \sigma$. Using common abbreviations, we define $\wedge$, $\rightarrow$, $\leftrightarrow$, $\forall x$, $\forall X$, and *true* (tautology). Notice that MSO logics can be both applied to finite and infinite structures. If the context is clear, we write MSO instead of MSO(σ).

An *FO-formula* is a formula of MSO without set quantifications, i.e. without using $\exists X$. An *EMSO-formula* is a formula of the form $\exists X_1 ... \exists X_k.\varphi$ where φ is an FO-formula.

We follow classical approaches for logics to define the semantics of formulas of MSO(σ). Let s be a σ-structure and $\varphi \in$ MSO(σ). Let free(φ) be the set of all free variables in φ, and let $\mathcal{V}$ be a finite set of variables containing free(φ). A $(\mathcal{V}, s)$-*assignment* γ is a function assigning to every first-order variable of $\mathcal{V}$ an element of $V(s)$ and to every second order variable a subset of $V(s)$. We define $\gamma[x \to v]$ as the $(\mathcal{V} \cup \{x\}, s)$-assignment mapping x to v and equaling γ everywhere else. The assignment $\gamma[X \to V]$ is defined analogously. Then, the satisfaction relation $(s,\gamma) \models \varphi$ for a relational structure s together with an assignment γ and an MSO-formula φ is defined inductively as seen in Figure 2.7.

Furthermore, we call φ a *sentence* if it contains no free variables. In this case, the assignment γ can be the empty function and then we write s instead of (s,γ). For a sentence $\varphi \in$ MSO, we define $L(\varphi)$, the *language of* φ as the class of all σ-structures s such that $s \models \varphi$. We call a class of σ-structures L *MSO-definable* (resp. *FO-definable)* if L equals $L(\varphi)$ for some MSO-sentence (resp. FO-sentence) φ.

Later, all our considered signatures will contain a unary labeling with an alphabet Σ. In this case, we can as usual encode every structure s together with an assignment γ as a structure s' over an extended alphabet $\Sigma' = \Sigma \times \{0,1\}^{\mathcal{V}}$. Then, we call a structure s' over Σ' *valid* if $s' = (s,\gamma)$ for a structure s and an assignment γ. Note that a structure is valid if and only if every first-order variable is assigned to exactly one position, which can be checked by an FO-formula.

Example 2.1. Following this approach, we get an MSO logic for all relational structures introduced in Section 2.2. As an example, we compare with the

grammar of $\mathrm{MSO}(\Sigma^*)$, the MSO logic for words over Σ given in Section 2.1, and $\mathrm{MSO}(\mathrm{DG}_t(A,B))$, the MSO logic for graphs over A and B (cf. [Tho96]):

$$\begin{array}{ll} \text{words} & \varphi ::= \mathrm{Lab}_a(x) \mid x \leq y \mid x = y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \exists X.\varphi \\ \text{graphs} & \varphi ::= \mathrm{Lab}_a(x) \mid E_b(x,y) \mid x = y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \exists X.\varphi \end{array}$$

where $a \in \Sigma$, resp. $a \in A$ and $b \in B$. ◇

2.4 Hanf's Theorem and Adding An Infinity Operator

In the following, we restate a Hanf-like result [Han65] of Bollig and Kuske [BK07] for the logic FO^∞. This logic features a first-order quantification $\exists^\infty x.\varphi$ to express that there exist infinitely many vertices satisfying φ. Using this result, we prove that for infinite graphs every FO^∞-sentence is equivalent to a boolean combination of formulas "$\mathrm{occ}(\tau) \geq n$" and formulas "$\mathrm{occ}(\tau) = \infty$", where τ are (possibly different) tiles and $n \in \mathbb{N}$ are (possibly different) natural numbers. This corollary will later be a cornerstone to prove that graph automata for infinite graphs are equivalent to the existential fragment of the MSO logic for graphs, see Theorem 6.48.

Following Bollig and Kuske, we define the logic $\mathrm{MSO}^\infty(\mathrm{DG}_t^\infty(A,B))$, short MSO^∞, by the following grammar

$$\varphi ::= \mathrm{Lab}_a(x) \mid E_b(x,y) \mid x = y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \exists^\infty x.\varphi \mid \exists X.\varphi$$

We denote by FO^∞, resp. EMSO^∞, the usual first-order, resp. existential fragment. Defining an *assignment* γ and an *update* $\gamma[x \to i]$ as usual, the satisfaction relation $\models$ is defined as before, together with $(G,\gamma) \models \exists^\infty x.\varphi$ iff $(G, \gamma[x \to v]) \models \varphi$ for infinitely many $v \in V$.

Using an extended Ehrenfeucht-Fraïssé game [Fra54, Ehr61], Bollig and Kuske succeeded in proving a Hanf-like result for these structures. Intuitively, Hanf's classical result applied to graphs says that for a given $k \in \mathbb{N}$ and a fixed maximal degree, there exist a sufficiently large radius r and a threshold h such that two graphs which cannot be distinguished by counting occurrences of r-tiles up to h, are also indistinguishable by any FO^∞-formula up to quantifier depth k.

Note that this result also can be formulated for relational structures by considering tiles of the *Gaifman graph* (that is, the graph that connects all pairs of vertices which are together in one relation). For a more detailed look, we have to assume the reader to be familiar with rudiments of classical model theory. For details see e.g. [Lib04] or [Tho97]. We denote by $\mathrm{FO}[k]$, resp. $\mathrm{FO}^\infty[k]$, the set of first-order formulas of quantifier rank at most k. We write $\mathfrak{A} \equiv_k \mathfrak{B}$ if the duplicator in an EF-Game can force the play started

in $(\mathfrak{A}, \mathfrak{B}, k)$ into a winning position. Bollig and Kuske [BK07] introduce an extended EF-game where spoiler can also choose to play on infinite subsets. We write $\mathfrak{A} \equiv_k^\infty \mathfrak{B}$ if duplicator wins such an extended EF-game starting in $(\mathfrak{A}, \mathfrak{B}, k)$.

Theorem 2.4. *Let $\mathfrak{A}$, $\mathfrak{B}$ be σ-structures and $k \in \mathbb{N}$.*

1. Ehrenfeucht-Fraïssé*: Then $\mathfrak{A}$ and $\mathfrak{B}$ agree on* $\mathrm{FO}[k]$ *if and only if* $\mathfrak{A} \equiv_k \mathfrak{B}$.
2. Bollig-Kuske*: Then $\mathfrak{A}$ and $\mathfrak{B}$ agree on* $\mathrm{FO}^\infty[k]$ *if and only if* $\mathfrak{A} \equiv_k^\infty \mathfrak{B}$.

Furthermore, given $r, h \in \mathbb{N}$ and two σ-structures $\mathfrak{A}$, $\mathfrak{B}$, we write $\mathfrak{A} \rightleftarrows_{r,h} \mathfrak{B}$ if for any r-tile, the number of occurrences of this tile in $\mathfrak{A}$ and $\mathfrak{B}$ is equal or both numbers are greater than or equal to h (cf. *threshold equivalence* [Tho97]). We write $\mathfrak{A} \rightleftarrows_{r,h}^\infty \mathfrak{B}$ for the respective notion where we also distinguish between "more than t" and "infinitely many". Then one formulation of Hanf's theorem and its extension including counting to infinity is the following.

Theorem 2.5.
1. Hanf*: For any $k, t \in \mathbb{N}$, there exist $r, h \in \mathbb{N}$ such that for all σ-structures $\mathfrak{A}$ and $\mathfrak{B}$ of degree at most t, it holds that $\mathfrak{A} \rightleftarrows_{r,h} \mathfrak{B}$ implies $\mathfrak{A} \equiv_k \mathfrak{B}$.*
2. Bollig-Kuske*: For any $k, t \in \mathbb{N}$, there exist $r, h \in \mathbb{N}$ such that for all σ-structures $\mathfrak{A}$ and $\mathfrak{B}$ of degree at most t, it holds that $\mathfrak{A} \rightleftarrows_{r,h}^\infty \mathfrak{B}$ implies $\mathfrak{A} \equiv_k^\infty \mathfrak{B}$.*

We apply the second result, which was originally developed in a different context, namely Muller message-passing automata, to $\mathrm{DG}_t(A, B)$ and $\mathrm{DG}_t^\infty(A, B)$ to get the following corollary.

A boolean combination of formulas of the form "$\mathrm{occ}(\tau) \geq n$", where n are natural numbers and τ are tiles, is called an *occurrence constraint*. We say G *satisfies* $\mathrm{occ}(\tau) \geq n$ if there exist at least n distinct vertices $v \in V$ such that $\mathrm{sph}^r(G, v)$ is isomorphic to τ. The semantics of "G *satisfies* Occ" are then defined in the usual way.

Note that neither of the following results is true for graphs of unbounded degree. Although, the first part of the following result was already stated by Thomas in [Tho91], we also reproduce a proof for the finite case which is then extendable to the infinite setting.

Corollary 2.6.
1. Thomas*: In the class $\mathrm{DG}_t(A, B)$, every FO-sentence φ is equivalent to an occurrence constraint.*
2. *In the class $\mathrm{DG}_t^\infty(A, B)$, every FO^∞-sentence φ is equivalent to an extended occurrence constraint.*

Proof. 1. Let φ be an FO-formula with quantifier depth k. Let r, h be the sufficiently large values of Theorem 2.5 and let $G \in \mathrm{DG}_t(A, B)$. We denote by

$[G]_{\text{occ}}$, resp. $[G]_{\text{FO}}$, the equivalence classes of G with respect to $\rightleftarrows_{r,h}$, resp. $\equiv_k$. Then Theorem 2.5 states that $[G]_{\text{occ}} \subseteq [G]_{\text{FO}}$. Therefore, we get

$$[G]_{\text{FO}} \subseteq \bigcup_{H \in [G]_{\text{FO}}} [H]_{\text{occ}} \subseteq \bigcup_{H \in [G]_{\text{FO}}} [H]_{\text{FO}} = [G]_{\text{FO}} \ . \tag{2.1}$$

Furthermore, we denote by $\text{Occ}_{r,h}$ the finite set of all logical non-equivalent occurrence constraints (with radius r and up to the threshold h). For a graph G, we define an occurrence constraint $\text{occ}_G \in \text{Occ}_{r,h}$, as follows

$$\text{occ}_G = \bigwedge_{\substack{\text{occ} \in \text{Occ}_{r,h} \\ G \models \text{occ}}} \text{occ} \wedge \bigwedge_{\substack{\text{occ} \in \text{Occ}_{r,h} \\ G \not\models \text{occ}}} \neg\, \text{occ} \ .$$

Then $H \models \text{occ}_G$ iff $G \rightleftarrows_{r,h} H$ iff $H \in [G]_{\text{occ}}$. Thus,

$$L(\text{occ}_G) = [G]_{\text{occ}} \ . \tag{2.2}$$

Now, we get the following

$$L(\varphi) = \bigcup_{G \in L(\varphi)} [G]_{\text{FO}} \overset{(2.1)}{=} \bigcup_{G \in L(\varphi)} \bigcup_{H \in [G]_{\text{FO}}} [H]_{\text{occ}} \ .$$

As there are only finitely many classes equivalence $[H]_{\text{occ}}$, there exists an $n \in \mathbb{N}$ and $H_1, ..., H_n \in \text{DG}_t(A, B)$ such that

$$\begin{aligned} L(\varphi) &= [H_1]_{\text{occ}} \cup ... \cup [H_n]_{\text{occ}} \\ &\overset{(2.2)}{=} L(\text{occ}_{H_1}) \cup ... \cup L(\text{occ}_{H_n}) \\ &= L(\text{occ}_{H_1} \vee ... \vee \text{occ}_{H_n}) \ . \end{aligned}$$

Thus, $\text{occ}_{H_1} \vee ... \vee \text{occ}_{H_n}$ is an occurrence constraint equivalent to φ.

2. Using respective notations, the second statement of Theorem 2.5 states $[G]^{\infty}_{\text{occ}} \subseteq [G]^{\infty}_{\text{FO}}$. Therefore, the respective statement of formula 2.1 holds, which shows

$$[G]^{\infty}_{\text{FO}} = \bigcup_{H \in [G]^{\infty}_{\text{FO}}} [H]_{\text{occ}} \ .$$

Furthermore, $\text{FO}^{\infty}[k]$ and $\text{Occ}^{\infty}_{r,h}$ are still finite (up to logical non-equivalence) and therefore also the other respective statements of part 1 still hold. Thus, the statement is proven analogously to 1. □

For pointed graphs of $\text{pDG}^{\infty}_t(A, B)$, we have to add a predicate *root* to the MSO logic describing the root of a graph. Apart from this, the proof stays the same for pointed infinite graphs.

Chapter 3

Quantitative Aspects

Weights are used to model quantitative aspects of systems. Classical systems either say yes or no, which can be seen as 1 or 0. Weighted systems can count higher than 1 or rate between 0 and 1. e.g. a classical system may say whether a pattern occurs in a string or not, while a weighted system can also say how often it occurs.

In this chapter, we introduce the weight structures that will be used in this thesis, namely semirings, valuation monoids, and product valuation monoids. We introduce general weighted MSO logics over these weight structures for relational structures, in particular for graphs. We introduce fragments of these logics used later and prove our first main result. This result is a transformation theorem showing how to, under appropriate assumptions on our weight structure, transform formulas into equivalent but restricted formulas.

3.1 Semirings

A semiring is a tuple $\mathbb{K} = (K, +, \cdot, 0, 1)$ such that $(K, +, 0)$ is a commutative monoid, $(K, \cdot, 1)$ is a monoid, and we have $(x + y) \cdot z = x \cdot z + y \cdot z$, $x \cdot (y + z) = x \cdot y + x \cdot z$, and $0 \cdot x = x \cdot 0 = 0$ for all $x, y, z \in K$. $\mathbb{K}$ is called *commutative* if $(K, \cdot, 1)$ is commutative.

Important examples of commutative semirings cover the Boolean semiring $\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$, the semiring of the natural numbers $\mathbb{N} = (\mathbb{N}, +, \cdot, 0, 1)$, and the tropical semirings $\mathbb{R}_{\max} = (\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)$ and $\mathbb{R}_{\min} = (\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$. Non-commutative semirings are given, e.g., by $n \times n$-matrices over semirings $\mathbb{K}$ with matrix addition and multiplication as usual ($n \geq 2$), or the semiring $(\mathcal{P}(\Sigma^*), \cup, \cdot, \emptyset, \{\varepsilon\})$ of languages over Σ.

We say $\mathbb{K}$ is *idempotent* if the addition is idempotent, i.e. $x + x = x$ for all $x \in K$. The semirings $\mathbb{B}$, $\mathbb{R}_{\max}$, and $\mathbb{R}_{\min}$ are idempotent. For a general introduction into the theory of semirings and extensive examples, see [DKV09, Gol99].

3.2 Valuation Monoids

In the following, we introduce a weight structure, called *valuation monoid* which, besides covering all commutative semirings, is able to model aspects like average, discounting, and long-time behaviors of automata. Originally, valuation monoids were introduced by Droste and Meinecke in [DM12] over words to formalize automata models investigated by Chatterjee, Doyen, and Henzinger [CDH08, CDH09]. Subsequently, valuation monoids were extended to, e.g., trees in [DGMM11] and to pictures in [BD15]. They also found application for timed automata [DP14a]. In the context of trees, another natural structure to generalize the semiring operations are multi-operator monoids. They were studied, e.g., in [FSV12] and shown to be closely related to tree-valuation monoids in [TO15] and [DGMM11]. Here, we introduce valuation monoids for general relational structures including graphs.

We introduce *D-labelings* to attach to every point of the domain of a structure a weight, as follows. Let D be a set and s a σ-structure. We call a mapping $\mathrm{Lab}_{s,D} : V(s) \to D$ a *labeling of s with D* or *D-labeling of s*. Every σ-structure together with such a labeling, $(s, \mathrm{Lab}_{s,D})$, is called a *D-labeled σ-structure*. We denote by C^σ_D the class of all D-labeled σ-structures of C^σ.

To handle infinite sums occurring for infinite structures later, we introduce *complete monoids* as follows. We call a monoid $(D, +, 0)$ *complete* if there exist infinitary sum operations $\sum_I : D^I \to D$ for every index set[1] I such that

- $\sum_{i\in\emptyset} d_i = 0$, $\sum_{i\in\{k\}} d_i = d_k$, $\sum_{i\in\{j,k\}} d_i = d_j + d_k$ for $j \neq k$,
- $\sum_{j\in J}(\sum_{i\in I_j} d_i) = \sum_{i\in I} d_i$ if $\bigcup_{j\in J} I_j = I$ and $I_j \cap I_k = \emptyset$ for $j \neq k$.

Note that in every complete monoid the operation $+$ is commutative.

Definition 3.1. A *C^σ-valuation monoid*, short σ-valuation monoid, is a tuple $\mathbb{D} = (D, +, \mathrm{Val}^\sigma, 0)$ consisting of a commutative monoid $(D, +, 0)$ together with a valuation function

$$\mathrm{Val}^\sigma : C^\sigma_D \to D$$

with $\mathrm{Val}^\sigma(s, \mathrm{Lab}_{s,D}) = 0$ if there exists at least one $v \in V(s)$ with $\mathrm{Lab}_{s,D}(v) = 0$. If C^σ contains structures with an infinite domain, we additionally require $(D, +, 0)$ to be complete.

We say $\mathbb{D}$ is *idempotent* if the addition is idempotent, i.e., $d + d = d$ for all $d \in D$.

As already noted by [DM12], in the case of finite words every semiring $(K, +, \cdot, 0)$ can be seen as a valuation monoid $(K, +, \prod, 0)$. For infinite words, the same is true for semirings which feature infinitary sum operations and countably infinite products satisfying natural axioms for these operations. Such semirings are called *totally complete semirings* [ÉK09].

[1] Note that these index sets may be uncountably infinite.

However, to embed an arbitrary semiring into a valuation monoid over other finite structures is only possible for structures with a defined order on their domain. Since in general our graphs have no predefined order, we additionally have to require the semiring to be commutative to embed it into a valuation monoid.

Additionally, and greatly motivated by them, valuation monoids cover structures using computations like average or discounting, as demonstrated by the following examples.

Example 3.1. Consider the class of finite graphs over their respective signature σ as in Section 2.2. Let $D = \mathbb{R} \cup \{-\infty\}$ and let $(G, \mathrm{Lab}_{G,D})$ be a graph of $\mathrm{DG}_t(A, B)$ together with a D-labeling. We denote by $|V|$ the number of vertices of G. Then, we define $\mathbb{D}_1 = (\mathbb{R} \cup \{-\infty\}, \sup, \mathrm{avg}, -\infty)$ where

$$\mathrm{avg}(G, \mathrm{Lab}_{G,D}) = \frac{1}{|V|} \sum_{v \in V} \mathrm{Lab}_{G,D}(v) \ .$$

Now, let $(G, u, \mathrm{Lab}_{G,D})$ be a pointed graph of $\mathrm{pDG}_t(A, B)$ together with a D-labeling. We let

$$\mathrm{reach}(G, u) = \max\{\mathrm{dist}(u, v) \mid v \in V, \mathrm{dist}(u, v) < \infty\} \ .$$

Then, for $0 < \lambda < 1$, we define $\mathbb{D}_2 = (\mathbb{R} \cup \{-\infty\}, \sup, \mathrm{disc}_\lambda, -\infty)$, where

$$\mathrm{disc}_\lambda(G, u, \mathrm{Lab}_{G,D}) = \sum_{r=0}^{\mathrm{reach}(G,u)} \sum_{\mathrm{dist}(u,v)=r} \lambda^r \, \mathrm{Lab}_{G,D}(v) \ .$$

Then $\mathbb{D}_1$ and $\mathbb{D}_2$ are two valuation monoids. ◇

Valuation monoids will later be used to define the behaviors of weighted automata. A weighted graph automaton will attach to every transition, and therefore to every vertex, a weight. Since every finite weighted automaton only uses finitely many different weights, the input on the valuation function will be a finite subset of D [2]. Therefore, by abuse of notation, the D-labeling can also be substituted by the vertex-labeling of the graph. The same holds true for weighted automata over special classes of graphs like words.

In the following, we fix a signature σ and may omit it if the context is clear. Furthermore, $\mathbb{D}$ will always refer to a σ-valuation monoid.

In contrast to classical semirings, a valuation monoid features no second binary operation. In the case that our underlying structure provides such an additional binary operation, usually a multiplication, we can naturally extend the definition of a valuation monoid to a *product valuation monoid* as follows.

[2] Since the graph can be arbitrarily large, the output of Val remains unbounded.

Definition 3.2. A *product σ-valuation monoid* (short *pv-monoid*) is a tuple $\mathbb{PD} = (D, +, \mathrm{Val}^\sigma, \diamond, 0, 1)$ consisting of a σ-valuation monoid $(D, +, \mathrm{Val}, 0)$ together with a constant 1 and an operation $\diamond : D^2 \to D$ satisfying $0 \diamond d = d \diamond 0 = 0$, $1 \diamond d = d \diamond 1 = d$ for all $d \in D$, and $\mathrm{Val}^\sigma(s, \mathrm{Lab}_{s,D}) = 1$ if $\mathrm{Lab}_{s,D}(v) = 1$ for all $v \in V(s)$.

As above, over finite structures, every commutative semiring $(K, +, \cdot, 0, 1)$ can be seen as a product valuation monoid $(K, +, \prod, \cdot, 0, 1)$. Over infinite structure, the same holds for totally complete semirings.

In the following, $\mathbb{PD}$ will always refer to a product σ-valuation monoid.

In comparison to a semiring, a product valuation monoid features much fewer axioms like associativity of the product or distributivity between its operations. Depending on the context and to compare pv-monoids to semirings, it is helpful to reintroduce some of these axioms. Later, we will see that the more we know about our valuation monoids, the stronger the characterization results are that we can formulate. This means that our results will cover the classical case of semirings, but we will also prove results for the most general case of a valuation monoid.

Furthermore, we want to stress that our motivating examples covering average and discounting fall under the majority of the following assumptions, cf. 3.2. With this motivation in mind, and inspired from [DM12] and [DD17], we define the following properties for a product valuation monoid $\mathbb{PD}$.

We call $\mathbb{PD}$ *associative* resp. *commutative* if $\diamond$ is associative resp. commutative.

We call $\mathbb{PD}$ *left-+-distributive* if for all $d \in D$, for any index set I, $(d_i)_{i \in I} \in D^I$ such that $\sum_{i \in I} d_i \in D$ and $\sum_{i \in I} (d \diamond d_i) \in D$, it holds that

$$d \diamond \sum_{i \in I} d_i = \sum_{i \in I} (d \diamond d_i) \ .$$

Then, *right-+-distributivity* is defined analogously. We call $\mathbb{PD}$ *+-distributive* if $\mathbb{PD}$ is left- and right-+-distributive.

We call $\mathbb{PD}$ *left*-Val-*distributive* if for all $d \in D$ and all D-labeled σ-structures $(s, \mathrm{Lab}_{s,D})$, we have

$$\begin{aligned} d \diamond \mathrm{Val}(s, \mathrm{Lab}_{s,D}) &= \mathrm{Val}(s, \mathrm{Lab}'_{s,D}) \ , \text{ where} \\ \mathrm{Lab}'_{s,D}(v) &= d \diamond \mathrm{Lab}_{s,D}(v) \quad \text{for all } v \in V(s) \ . \end{aligned}$$

We say C^σ has *minimal points* if there exists a well-defined function[3] f that assigns to every σ-structure $s = (V(s), (R_s)_{R \in \sigma})$ of C^σ an element $u \in V(s)$. In other words, we require the existence of a unique distinguished root for all

[3] As we distinguish σ-structures only up to isomorphism, the function f has to satisfy that for every two σ-structures s, s', and every isomorphism $\varphi : V(s) \to V(s')$, it holds that $\varphi(f(s)) = f(s')$.

structures of our class. In the context of graphs, we referred to this as *pointed graphs*. In particular, all examples of Section 2.2 except unpointed graphs have minimal points.

If C^σ has minimal points, we call $\mathbb{PD}$ *left-multiplicative* if for all $d \in D$ and all D-labeled σ-structures $(s, \mathrm{Lab}_{s,D})$ with minimal point $u \in V(s)$, the following holds

$$d \diamond \mathrm{Val}(s, \mathrm{Lab}_{s,D}) = \mathrm{Val}(s, \mathrm{Lab}'_{s,D}) \ , \text{ where}$$
$$\mathrm{Lab}'_{s,D}(v) = \begin{cases} d \diamond \mathrm{Lab}_{s,D}(v) & , \text{ if } v = u \\ \mathrm{Lab}_{s,D}(v) & , \text{ otherwise} \end{cases} \qquad \text{for all } v \in V \ .$$

We call $\mathbb{PD}$ *conditionally commutative* if the following holds. For all D-labeled σ-structures $(s, \mathrm{Lab}_{s,D})$ and $(s, \mathrm{Lab}'_{s,D})$ with $\mathrm{Lab}_{s,D}(w) \diamond \mathrm{Lab}'_{s,D}(w) = \mathrm{Lab}'_{s,D}(w) \diamond \mathrm{Lab}_{s,D}(w)$ for all $w \in V(s)$, we have

$$\mathrm{Val}(s, \mathrm{Lab}_{s,D}) \diamond \mathrm{Val}(s, \mathrm{Lab}'_{s,D}) = \mathrm{Val}(s, \mathrm{Lab}''_{s,D}) \ , \text{ where}$$
$$\mathrm{Lab}''_{s,D}(v) = \mathrm{Lab}_{s,D}(v) \diamond \mathrm{Lab}'_{s,D}(v) \quad \text{for all } v \in V(s) \ .$$

We call $\mathbb{PD}$ *left-distributive* if $\mathbb{PD}$ is left-$+$-distributive and, additionally, left-Val-distributive or left-multiplicative (assuming we have minimal points). If $\mathbb{PD}$ is $+$-distributive and associative, then $(D, +, \diamond, 0, 1)$ is a complete semiring and we call $(D, +, \mathrm{Val}, \diamond, 0, 1)$ a *σ-valuation semiring*. A *cc-σ-valuation semiring* is a σ-valuation semiring which is conditionally commutative and left-distributive.

Example 3.2. Similarly to [DM12], we can add a classical addition to the valuation monoids of Example 3.1 to obtain product valuation monoids as follows. We use labeled graphs and the operations avg() and $\mathrm{disc}_\lambda()$ as in Example 3.1. Then $\mathbb{PD}_1 = (\mathbb{R} \cup \{-\infty\}, \sup, \mathrm{avg}, +, -\infty, 0)$ is a left-Val-distributive cc-valuation semiring over finite graphs.

Furthermore, $\mathbb{PD}_2 = (\mathbb{R} \cup \{-\infty\}, \sup, \mathrm{disc}_\lambda, +, -\infty, 0)$ is a left-multiplicative cc-valuation semiring over pointed finite graphs. ◇

3.3 Weighted Logics

In this section, we introduce the weighted MSO logic and its different forms which will be used in this dissertation. Originally, this logic was introduced for words by Droste and Gastin [DG07] as a standalone generalization of the classical MSO logic. Here, we incorporate the distinction into an unweighted and a weighted part of this logic by Bollig and Gastin [BG09]. This distinction makes weighted MSO a natural extension of any classical MSO-logic. We define weighted MSO for general relational structures and over different weight structures. Then, we will derive its variants for nested words, precedence words, and graphs in the later chapters from this general definition.

We start with a version of this weighted logic, first studied in [GM15] that utilizes an 'if..then..else'-operator, $\beta?\varphi_1 : \varphi_2$, instead of a weighted product. The two advantages of this version are that it is directly applicable to valuation monoids and needs less restrictions in the later Büchi-like characterization of automata.

Definition 3.3. Given a signature σ and a σ-valuation monoid $\mathbb{D}$, we define the weighted logic $\mathrm{MSO}(\mathbb{D}, \sigma)$, short $\mathrm{MSO}(\mathbb{D})$, as

$$\beta ::= R(x_1, ..., x_{\mathrm{rk}(R)}) \mid x = y \mid x \in X \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta \mid \exists X.\beta$$
$$\varphi ::= d \mid \varphi \oplus \varphi \mid \beta?\varphi : \varphi \mid \bigoplus_x \varphi \mid \bigoplus_X \varphi \mid \mathrm{Val}_x\, \varphi$$

where $R \in \sigma$, $d \in D$; x, x_i, y are first order variables; and X is a second order variable.

If our underlying structure provides a second operation, e.g. a multiplication as in the case of semirings or product valuation monoid, we replace '$\beta?\varphi : \varphi$' in the second line with the weighted formula $\varphi \otimes \varphi$ which is evaluated using this product. If in this case, the multiplication also provides a neutral element 1, this allows to evaluate an unweighted formula β also as a weighted formula, by assigning the weight 1 if β holds and 0 otherwise. This leads to the following definition.

Definition 3.4. Given a signature σ and a product σ-valuation monoid $\mathbb{PD}$, we define the weighted MSO-logic $\mathrm{MSO}(\mathbb{PD}, \sigma)$, short $\mathrm{MSO}(\mathbb{PD})$ as

$$\beta ::= R(x_1, ..., x_{\mathrm{rk}(R)}) \mid x = y \mid x \in X \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta \mid \exists X.\beta$$
$$\varphi ::= \beta \mid d \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi \mid \bigoplus_x \varphi \mid \bigoplus_X \varphi \mid \mathrm{Val}_x\, \varphi$$

where $R \in \sigma$, $d \in D$; x, x_i, y are first order variables; and X is a second order variable.

Formulas of the fragment β are exactly the unweighted formulas of $\mathrm{MSO}(\sigma)$. Interpreted as formula of $\mathrm{MSO}(\mathbb{PD}, \sigma)$, we call β a *boolean formula*. If β does not contain $\exists X$ it is called *FO-boolean formula*. The formulas φ are called *weighted formulas*. For boolean formulas β, we use common abbreviations for $\wedge$, $\forall x.\beta$, $\forall X.\beta$, $\rightarrow$, and $\leftrightarrow$. In the special case that $\mathbb{PD}$ is a semiring, Val_x is usually denoted by $\prod_x$. Note that in [DG07], the weighted connectors were also denoted by $\vee$, $\wedge$, $\exists x$, $\exists X$, and $\forall x$. We employ this symbolic change to stress the quantitative evaluation of formulas.

Let s be a σ-structure and $\varphi \in \mathrm{MSO}(\mathbb{D})$ or $\varphi \in \mathrm{MSO}(\mathbb{PD})$. As in Section 2.3, let $\mathcal{V}$ be a finite set of variables containing $\mathrm{free}(\varphi)$. We denote by $\Gamma_{(\mathcal{V},s)}$ the set of all $(\mathcal{V}, s)$-assignments. Then the semantics of φ is defined as a function $[\![\varphi]\!]_{\mathcal{V}} : C^\sigma \times \Gamma_{(\mathcal{V},s)} \to D$ inductively as seen in Figure 3.1.

$$
\begin{array}{lcl}
[\![d]\!]_{\mathcal{V}}(s,\gamma) & = & d \quad \text{for all } d \in D \\
[\![\varphi \oplus \psi]\!]_{\mathcal{V}}(s,\gamma) & = & [\![\varphi]\!]_{\mathcal{V}}(s,\gamma) + [\![\psi]\!]_{\mathcal{V}}(s,\gamma) \\
[\![\beta ? \varphi : \psi]\!]_{\mathcal{V}}(s,\gamma) & = & \begin{cases} [\![\varphi]\!]_{\mathcal{V}}(s,\gamma) & , \text{ if } (s,\gamma) \models \beta \\ [\![\psi]\!]_{\mathcal{V}}(s,\gamma) & , \text{ otherwise} \end{cases} \\
[\![\beta]\!]_{\mathcal{V}}(s,\gamma) & = & \begin{cases} 1 & , \text{ if } (s,\gamma) \models \beta \\ 0 & , \text{ otherwise} \end{cases} \\
[\![\varphi \otimes \psi]\!]_{\mathcal{V}}(s,\gamma) & = & [\![\varphi]\!]_{\mathcal{V}}(s,\gamma) \diamond [\![\psi]\!]_{\mathcal{V}}(s,\gamma) \\
[\![\bigoplus_x \varphi]\!]_{\mathcal{V}}(s,\gamma) & = & \sum\limits_{i \in V(s)} [\![\varphi]\!]_{\mathcal{V} \cup \{x\}}(s, \gamma[x \to i]) \\
[\![\bigoplus_X \varphi]\!]_{\mathcal{V}}(s,\gamma) & = & \sum\limits_{I \subseteq V(s)} [\![\varphi]\!]_{\mathcal{V} \cup \{X\}}(s, \gamma[X \to I]) \\
[\![\mathrm{Val}_x \varphi]\!]_{\mathcal{V}}(s,\gamma) & = & \mathrm{Val}(s, \mathrm{Lab}_{s,D}) \quad , \text{ where} \\
 & & \mathrm{Lab}_{s,D}(v) = [\![\varphi]\!]_{\mathcal{V} \cup \{x\}}(s, \gamma[x \to v]) \text{ for all } v \in V(s)
\end{array}
$$

Figure 3.1: Semantics of MSO($\mathbb{D}$) and MSO($\mathbb{PD}$).

In particular, we have $[\![\beta ? \varphi : \psi]\!]_{\mathcal{V}} = [\![(\beta \otimes \varphi) \oplus (\neg\beta \otimes \psi)]\!]_{\mathcal{V}}$. Therefore, given a product valuation monoid $\mathbb{PD}$ with an underlying valuation monoid $\mathbb{D}$, we can consider MSO($\mathbb{D}$) as a restriction of MSO($\mathbb{PD}$).

We write $[\![\varphi]\!]$ for $[\![\varphi]\!]_{\mathrm{free}(\varphi)}$. The following lemma shows that for each finite set of variables containing $\mathrm{free}(\varphi)$ the semantics $[\![\varphi]\!]_{\mathcal{V}}$ are consistent with each other (cf. [DG07]).

Lemma 3.5. *Let $\varphi \in$ MSO($\mathbb{D}$) or $\varphi \in$ MSO($\mathbb{PD}$) and $\mathcal{V}$ be a finite set of variables with $\mathcal{V} \supseteq \mathrm{free}(\varphi)$. Then $[\![\varphi]\!]_{\mathcal{V}}(s,\gamma) = [\![\varphi]\!](s, \gamma|_{\mathrm{free}(\varphi)})$ for each $(s,\gamma) \in C^\sigma \times \Gamma_{(\mathcal{V},s)}$*

Proof. The statement is proved by a standard induction on the structure of φ analogously to Proposition 3.3 of [DG07]. □

3.4 Transformation Theorem

In the original proofs of Droste and Gastin of a Büchi theorem for weighted logics and weighted automata, the distributivity and commutativity of the semiring is crucial. In general, product valuation monoids do not enjoy these properties, which makes it very hard to model the general weighted product of the logic with an automaton. Apart from forgoing the product completely (like done with MSO($\mathbb{D}$) for valuation monoids), there are two possibilities to handle this difficulty. The first one is to restrict the use of the weighted product to only occur under specific circumstances. The second approach is to make some additional assumptions on the product valuation monoid to re-inject some form

of (left)-distributivity. For the second approach, we refer to the properties of product valuation monoids introduced in Section 3.2.

While it may not be surprising that these two approaches work on their own, in the following, we also study combinations of them. That is, we show that if our product valuation monoid enjoys, compared to a semiring, weaker forms of compatibility between its operations, then we only need a weak restriction of the weighted product of the logic. This is, e.g., the case for the product valuation monoids featuring average and discounting. More detailed, in the following, we prove a direct *Transformation Theorem* (Theorem 3.6), which, according to the assumptions on the product valuation monoid, transforms a formula into a restricted but semantically equivalent formula.

Our main application of this theorem can later be found in the connection between weighted automata and weighted logics, where it is used to answer the following question: Given a product valuation monoid with certain properties, how expressive can we make a logic such that it is still expressively equivalent to a weighted automata model over this product valuation monoid? In other words: Depending on the weight structure, how expressive are our weighted automata?

The formulation of the different properties for product valuation monoids and the specific fragments of the weighted logic is strongly inspired from [DM12]. Note however that here, we formulate the whole theory for relational structures, in particular for graphs and all its subclasses. Furthermore, Theorem 3.6 provides a direct transformation on the level of the weighted formulas, which is new even in the word case, since previous results were shown by multiple translations between weighted automata and weighted logic. A special case of the results in this section for nested words was published in [DD17].

In the following, we introduce the different restrictions on formulas of $\mathrm{MSO}(\mathbb{PD})$. We call a formula $\varphi \in \mathrm{MSO}(\mathbb{PD})$ *almost FO-boolean* if it is built up inductively from the following grammar

$$\varphi ::= d \mid \beta \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi$$

where $d \in D$ and β is an unweighted FO-formula.

Let $\varphi \in \mathrm{MSO}(\mathbb{PD})$ and let $\mathrm{const}(\varphi)$ be the set of all elements of D occurring in φ. We say two subsets $A, B \subseteq D$ *commute*, if $a \otimes b = b \otimes a$ for all $a \in A$ and for all $b \in B$. Then, we call φ

1. *strongly-$\otimes$-restricted* if for all its subformulas $\psi \otimes \theta$,
   - ψ and θ are almost FO-boolean,
   - ψ is an FO-boolean formula, or
   - θ is an FO-boolean formula.
2. *$\otimes$-restricted* if for all its subformulas $\psi \otimes \theta$,

- ψ is almost FO-boolean, or
- θ is an FO-boolean formula.

3. *weakly-$\otimes$-restricted* if for all its subformulas $\psi \otimes \theta$,
   - ψ is almost FO-boolean, or
   - $\mathrm{const}(\psi)$ and $\mathrm{const}(\theta)$ commute.
4. *Val-restricted* if for all its subformulas $\mathrm{Val}_x\, \psi$, ψ is almost FO-boolean.

Then, we call φ *restricted (resp. strongly- or weakly-restricted)* if φ is *Val-restricted*, all its unweighted subformulas β are EMSO-formula, and φ is $\otimes$-restricted (resp. strongly-$\otimes$- or weakly-$\otimes$-restricted). If additionally all unweighted subformulas β are FO-formulas, we call φ *FO-restricted (resp. strongly-FO- or weakly-FO-restricted)*.

Since every strongly-$\otimes$-restricted formula is also $\otimes$-restricted and every $\otimes$-restricted formula is also weakly-$\otimes$-restricted, this defines three consecutive fragments of $\mathrm{MSO}(\mathbb{PD})$ with increasing expressive power. Next, we show that under specific assumptions on our product valuation monoid (cf. Section 3.2), the different fragments collapse. These transformations give a strong correlation between the assumptions on our valuation monoid and the expressive power of the weighted logic.

Theorem 3.6. *(a) Let $\mathbb{PD}$ be left-distributive. Then we can transform every $\otimes$-restricted formula $\varphi \in \mathrm{MSO}(\mathbb{PD})$ into a strongly-$\otimes$-restricted formula $\varphi' \in \mathrm{MSO}(\mathbb{PD})$ with the same semantics.*

(b) Let $\mathbb{PD}$ be a cc-σ-valuation semiring. Then we can transform every weakly-$\otimes$-restricted formula $\varphi \in \mathrm{MSO}(\mathbb{PD})$ into a strongly-$\otimes$-restricted formula $\varphi' \in \mathrm{MSO}(\mathbb{PD})$ with the same semantics.

(c) If φ is Val-restricted in (a) or (b), we can construct φ' also Val-restricted. Also, if in φ all unweighted subformulas β are EMSO (resp. FO)-formula, then the same is true for φ'.

If $\mathbb{PD}$ is a cc-ω-valuation semiring, clearly almost FO-boolean formulas can be written as disjunctions of conjunctions of boolean formulas or constants from $\mathbb{PD}$. Therefore, the proof of Theorem 3.6 (b) shows the following corollary, which is a direct conversion on the level of formulas and also applicable to commutative semirings.

Corollary 3.7. *Let $\mathbb{PD}$ be a commutative cc-valuation semiring. Then for any formula $\varphi \in \mathrm{MSO}(\mathbb{PD})$, there exists a formula $\varphi' \in \mathrm{MSO}(\mathbb{PD})$ with $[\![\varphi']\!] = [\![\varphi]\!]$ such that for all subformulas of the form $\psi \otimes \theta$, each of ψ and θ is FO boolean or a constant.*

The proof of Theorem 3.6 consists of multiple steps. Its structure is the following.

First, we show that for a cc-valuation semiring, the commutativity on the level of a set $A \subseteq D$ lifts to the set of all values computable by A, 0, and 1 together with finite and infinite sums and applications of the operations $\diamond$ and Val. Then, we use an induction on the structure of φ and make a case distinction depending on our restrictions. The main idea is found in Lemma 3.10, where we show that every weighted product $\otimes$ (in previous works the weighted conjunction) can be 'pulled inwards' by our assumptions on the valuation monoid.

Both results together show that we can transform any formula φ into an equivalent formula φ' in which conjunctions only appear between unweighted FO-formulas and constants. Then φ' is strongly-$\otimes$-restricted. For (c), we can show that these transformations can be done without increasing the number of quantifiers or the quantifier depth, in particular the Val-quantifier. Thus, these transformations conserve the Val-restriction, EMSO-, and FO-restriction of the formula.

In the following, we give the detailed proof. We start with some intermediate results. Let $\mathbb{PD}$ be a cc-σ-valuation semiring, $A \subseteq D$ a subset, and $s = (V(s), (R^s)_{R\in\sigma})$ a σ-structure of C^σ. We abbreviate with $\mathrm{Val}\restriction_s$ the restriction of Val to all D-labeled structures which consist of s together with an arbitrary D-labeling. Then, we define $A^{cl(s)}$, the closure of A with respect to s, as the smallest subset of D which contains A, the constants 0, 1, and is closed under finite and infinite sums[4] and application of the operations $\diamond$ and $\mathrm{Val}\restriction_s$.

Lemma 3.8. *Let $\mathbb{PD}$ be a cc-σ-valuation semiring, $s = (V(s), (R^s)_{R\in\sigma})$ a σ-structure, and $A, B \subseteq D$ two subsets such that A and B commute. Then $A^{cl(s)}$ and $B^{cl(s)}$ also commute.*

Proof. Following the definition of $A^{cl(s)}$ (resp. $B^{cl(s)}$), we have to check the following three statements.

Firstly, finite (resp. infinite) sums and finite products of elements of A commute with finite and infinite sums and finite products of elements of B. Indeed, this follows directly from the $+$-distributivity and associativity of $\mathbb{PD}$.

Secondly, we show that if $b = \mathrm{Val}(s, \mathrm{Lab}_{s,D})$ and $a \in A$ commutes with $\mathrm{Lab}_{s,D}(v)$ for all $v \in V(s)$, then a and b commute. Indeed, if $\mathbb{PD}$ is left-Val-distributive, we have

$$a = a \diamond 1 = a \diamond \mathrm{Val}(s, \mathrm{Lab}^1_{s,D}) = \mathrm{Val}(s, \mathrm{Lab}'_{s,D}) \ ,$$

where $\mathrm{Lab}^1_{s,D}(v) = 1$ for all $v \in V(s)$ and $\mathrm{Lab}'_{s,D}(v) = a$ for all $v \in V(s)$. Note that we refer to different D-labeling of s but the structure s stays fixed. Hence, using that $\mathbb{PD}$ is conditionally commutative and denoting $\mathrm{Lab}''_{s,D}$ as the D-labeling of s with $\mathrm{Lab}''_{s,D}(v) = a \diamond \mathrm{Lab}_{s,D}(v) = \mathrm{Lab}_{s,D}(v) \diamond a$ for all $v \in V(s)$,

[4] Infinite sums are only included for structures with a possibly infinite domain

we get

$$\begin{aligned} a \diamond b &= \mathrm{Val}(s, \mathrm{Lab}'_{s,D}) \diamond \mathrm{Val}(s, \mathrm{Lab}_{s,D}) \\ &= \mathrm{Val}(s, \mathrm{Lab}''_{s,D}) \\ &= \mathrm{Val}(s, \mathrm{Lab}_{s,D}) \diamond \mathrm{Val}(s, \mathrm{Lab}'_{s,D}) \\ &= b \diamond a \ . \end{aligned}$$

If C^σ has minimal points, s is pointed with $u \in V(s)$, and $\mathbb{PD}$ is left-multiplicative, we have

$$a = a \diamond \mathrm{Val}(s, \mathrm{Lab}^1_{s,D}) = \mathrm{Val}(s, \mathrm{Lab}'''_{s,D}) \ ,$$

where $\mathrm{Lab}'''_{s,D}(u) = a$ and $\mathrm{Lab}'''_{s,D}(v) = 1$ for all $v \in V(s) \setminus u$. Hence, using that $\mathbb{PD}$ is conditionally commutative and denoting $\mathrm{Lab}''''_{s,D}$ as the D-labeling of s with $\mathrm{Lab}''''_{s,D}(u) = a \diamond \mathrm{Lab}_{s,D}(u) = \mathrm{Lab}_{s,D}(u) \diamond a$ and $\mathrm{Lab}''''_{s,D}(v) = 1 \diamond \mathrm{Lab}_{s,D}(v) = \mathrm{Lab}_{s,D}(v) = \mathrm{Lab}_{s,D}(v) \diamond 1$ for all $v \in V(s) \setminus u$, we get

$$\begin{aligned} a \diamond b &= \mathrm{Val}(s, \mathrm{Lab}'''_{s,D}) \diamond \mathrm{Val}(s, \mathrm{Lab}_{s,D}) \\ &= \mathrm{Val}(s, \mathrm{Lab}''''_{s,D}) \\ &= \mathrm{Val}(s, \mathrm{Lab}_{s,D}) \diamond \mathrm{Val}(s, \mathrm{Lab}'''_{s,D}) \\ &= b \diamond a \ . \end{aligned}$$

Finally, let $a = \mathrm{Val}(s, \mathrm{Lab}^a_{s,D})$, $b = \mathrm{Val}(s, \mathrm{Lab}^b_{s,D})$, and $\mathrm{Lab}^a_{s,D}(v)$ commute with $\mathrm{Lab}^b_{s,D}(v')$ for all $v, v' \in V(s)$. We use that $\mathbb{PD}$ is conditionally commutative, and denoting $\mathrm{Lab}^{ab}_{s,D}$ as the D-labeling of s with $\mathrm{Lab}^{ab}_{s,D}(v) = \mathrm{Lab}^a_{s,D}(v) \diamond \mathrm{Lab}^b_{s,D}(v) = \mathrm{Lab}^b_{s,D}(v) \diamond \mathrm{Lab}^a_{s,D}(v)$ for all $v \in V(s)$, we get

$$\begin{aligned} a \diamond b &= \mathrm{Val}(s, \mathrm{Lab}^a_{s,D}) \diamond \mathrm{Val}(s, \mathrm{Lab}^b_{s,D}) \\ &= \mathrm{Val}(s, \mathrm{Lab}^{ab}_{s,D}) \\ &= \mathrm{Val}(s, \mathrm{Lab}^b_{s,D}) \diamond \mathrm{Val}(s, \mathrm{Lab}^a_{s,D}) \\ &= b \diamond a \ . \end{aligned}$$

Now, the result follows by a straightforward induction. □

Lemma 3.9. *Let $\mathbb{PD}$ be a cc-σ-valuation semiring and let ψ and θ be formulas of* $\mathrm{MSO}(\mathbb{PD})$ *. Let* $\mathrm{const}(\psi)$ *and* $\mathrm{const}(\theta)$ *commute. Then*

$$[\![\psi \otimes \theta]\!] = [\![\theta \otimes \psi]\!] \ .$$

Proof. Let $A = \mathrm{const}(\psi)$, $B = \mathrm{const}(\theta)$, $\mathcal{V} = \mathrm{free}(\psi) \cup \mathrm{free}(\theta)$, and $(s, \gamma) \in C^\sigma \times \Gamma_{(\mathcal{V},s)}$. By induction on the structure of ψ (resp. θ), we obtain that all

values of $[\![\psi]\!]$ (resp. $[\![\theta]\!]$) belong to $A^{cl(s)}$ (resp. $B^{cl(s)}$). By Lemma 3.8, $A^{cl(s)}$ and $B^{cl(s)}$ commute. Thus, we get

$$\begin{aligned}[\![\psi \otimes \theta]\!]_{\mathcal{V}}(s,\gamma) &= [\![\psi]\!]_{\mathcal{V}}(s,\gamma) \diamond [\![\theta]\!]_{\mathcal{V}}(s,\gamma) \\ &= [\![\theta]\!]_{\mathcal{V}}(s,\gamma) \diamond [\![\psi]\!]_{\mathcal{V}}(s,\gamma) \\ &= [\![\theta \otimes \psi]\!]_{\mathcal{V}}(s,\gamma) \ . \end{aligned}$$ □

The following result resembles similar results of classical logic in a quantitative setting. Note, however, that our present quantitative setting is very general, as it contains semirings as well as average or discounted computations. Also, observe the difference between (*b*) and (*c*), where we use the formula

$$\min(x) \to \psi = \neg \min(x) \oplus (\min(x) \otimes \psi) \ .$$

If ψ is almost FO-boolean, then $\min(x) \to \psi$ is also almost FO-boolean. Clearly, statements dual to the following hold also if $\mathbb{PD}$ satisfies the appropriate versions of distributivity on the right.

Lemma 3.10. *Let ψ, ψ_1, θ_1, θ_2 be formulas of* $\mathrm{MSO}(\mathbb{PD})$.

(a) Let $\mathbb{PD}$ be left-$+$-distributive. Then

$$[\![\psi \otimes (\theta_1 \oplus \theta_2)]\!] = [\![(\psi \otimes \theta_1) \oplus (\psi \otimes \theta_2)]\!] \ .$$

Let ψ contain no x. Then

$$\begin{aligned}[\![\psi \otimes \textstyle\bigoplus_x \theta_1]\!] &= [\![\textstyle\bigoplus_x (\psi \otimes \theta_1)]\!] \\ [\![\psi \otimes \textstyle\bigoplus_X \theta_1]\!] &= [\![\textstyle\bigoplus_X (\psi \otimes \theta_1)]\!] \ . \end{aligned}$$

*(b) Let $\mathbb{PD}$ be left-*Val*-distributive, and let ψ contain no x. Then*

$$[\![\psi \otimes \mathrm{Val}_x\, \theta_1]\!] = [\![\mathrm{Val}_x(\psi \otimes \theta_1)]\!] \ .$$

(c) Let C^σ have minimal points, $\mathbb{PD}$ be left-multiplicative, and let ψ contain no x. Then

$$[\![\psi \otimes \mathrm{Val}_x\, \theta_1]\!] = [\![\mathrm{Val}_x((\min(x) \to \psi) \otimes \theta_1)]\!] \ .$$

(d) Let $\mathbb{PD}$ be a cc-σ-valuation semiring, and let $\mathrm{const}(\psi_1)$ *and* $\mathrm{const}(\theta_1)$ *commute. Then*

$$[\![\mathrm{Val}_x\, \psi_1 \otimes \mathrm{Val}_x\, \theta_1]\!] = [\![\mathrm{Val}_x(\psi_1 \otimes \theta_1)]\!] \ .$$

Proof. (*a*) For the first part, we put $\mathcal{V} = \text{free}(\psi) \cup \text{free}(\theta_1) \cup \text{free}(\theta_2)$. Since $\mathbb{PD}$ is left-+-distributive, we get by Lemma 3.5:

$$\begin{aligned}
[\![\psi \otimes (\theta_1 \oplus \theta_2)]\!]_{\mathcal{V}} &= [\![\psi]\!]_{\mathcal{V}} \diamond ([\![\theta_1]\!]_{\mathcal{V}} + [\![\theta_2]\!]_{\mathcal{V}}) \\
&= [\![\psi]\!]_{\mathcal{V}} \diamond [\![\theta_1]\!]_{\mathcal{V}} + [\![\psi]\!]_{\mathcal{V}} \diamond [\![\theta_2]\!]_{\mathcal{V}} \\
&= [\![\psi \otimes \theta_1]\!]_{\mathcal{V}} + [\![\psi \otimes \theta_2]\!]_{\mathcal{V}} \\
&= [\![(\psi \otimes \theta_1) \oplus (\psi \otimes \theta_2)]\!]_{\mathcal{V}} \ .
\end{aligned}$$

For the second part, we put $\mathcal{V} = \text{free}(\psi) \cup \text{free}(\bigoplus_x \theta)$ and use Lemma 3.5. We use the assumption that $\mathbb{PD}$ is left-+-distributive at equation * to get for each $(s, \gamma) \in C^\sigma \times \Gamma_{(\mathcal{V},s)}$

$$\begin{aligned}
[\![\psi \otimes \textstyle\bigoplus_x \theta_1]\!]_{\mathcal{V}}(s,\gamma) &= [\![\psi]\!]_{\mathcal{V}}(s,\gamma) \diamond \sum_{v \in V(s)} ([\![\theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v])) \\
&\overset{*}{=} \sum_{v \in V(s)} ([\![\psi]\!]_{\mathcal{V}}(s,\gamma) \diamond [\![\theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v])) \\
&= \sum_{v \in V(s)} ([\![\psi]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \diamond [\![\theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v])) \\
&= \sum_{v \in V(s)} ([\![\psi \otimes \theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v])) \\
&= [\![\textstyle\bigoplus_x (\psi \otimes \theta_1)]\!]_{\mathcal{V}}(s,\gamma) \ .
\end{aligned}$$

The statement for the $\bigoplus_X$-quantifier can be shown analogously.

(*b*) We put $\mathcal{V} = \text{free}(\psi) \cup \text{free}(\text{Val}_x \theta_1)$. Let $(s, \gamma) \in C^\sigma \times \Gamma_{(\mathcal{V},s)}$. To refer to different D-labelings of s, we define for $v \in V(s)$

$$\begin{aligned}
\text{Lab}^{\theta_1}_{s,D}(v) &= [\![\theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \ , \\
\text{Lab}^{\psi \text{ ld } \theta_1}_{s,D}(v) &= [\![\psi]\!]_{\mathcal{V}}(s,\gamma) \diamond \text{Lab}^{\theta_1}_{s,D}(v) \ , \\
\text{Lab}^{\psi\otimes\theta_1}_{s,D}(v) &= [\![\psi \otimes \theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \ .
\end{aligned}$$

It follows that for all $v \in V(s)$

$$\begin{aligned}
\text{Lab}^{\psi \text{ ld } \theta_1}_{s,D}(v) &= [\![\psi]\!]_{\mathcal{V}}(s,\gamma) \diamond \text{Lab}^{\theta_1}_{s,D}(v) \\
&= [\![\psi]\!]_{\mathcal{V}}(s,\gamma) \diamond [\![\theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \\
&= [\![\psi]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \diamond [\![\theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \\
&= [\![\psi \otimes \theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \\
&= \text{Lab}^{\psi\otimes\theta_1}_{s,D}(v) \ .
\end{aligned}$$

Using this equation at ** and the assumption that $\mathbb{PD}$ is left-Val-distributive

at equation *, we get

$$\begin{aligned}
[\![\psi \otimes \mathrm{Val}_x\,\theta_1]\!]_{\mathcal{V}}(s,\gamma) &= [\![\psi]\!]_{\mathcal{V}}(s,\gamma) \diamond \mathrm{Val}(s, \mathrm{Lab}_{s,D}^{\theta_1}) \\
&\overset{*}{=} \mathrm{Val}(s, \mathrm{Lab}_{s,D}^{\psi \text{ ld } \theta_1}) \\
&\overset{**}{=} \mathrm{Val}(s, \mathrm{Lab}_{s,D}^{\psi \otimes \theta_1}) \\
&= [\![\mathrm{Val}_x(\psi \otimes \theta_1)]\!]_{\mathcal{V}}(s,\gamma) \ .
\end{aligned}$$

(*c*) Let $s \in C^\sigma$ with minimal point $u \in V(s)$. Let $\mathcal{W} \supseteq \mathrm{free}(\psi) \cup \{x\}$ and let γ' be a $(\mathcal{W}, s)$-assignment. We note that

$$\begin{aligned}
&[\![\min(x) \to \psi]\!]_{\mathcal{W}}(s,\gamma') \\
&\quad = [\![\neg \min(x) \oplus (\min(x) \otimes \psi)]\!]_{\mathcal{W}}(s,\gamma') \\
&\quad = [\![\neg \min(x)]\!]_{\mathcal{W}}(s,\gamma') + ([\![\min(x)]\!]_{\mathcal{W}}(s,\gamma') \diamond [\![\psi]\!]_{\mathcal{W}}(s,\gamma')) \\
&\quad = \begin{cases} 0 + 1 \diamond [\![\psi]\!]_{\mathcal{W}}(s,\gamma') & , \text{ if } \min(\gamma'(x)) \text{ holds} \\ 1 + 0 \diamond [\![\psi]\!]_{\mathcal{W}}(s,\gamma') & , \text{ otherwise} \end{cases} \\
&\quad = \begin{cases} [\![\psi]\!]_{\mathcal{W}}(s,\gamma') & , \text{ if } \gamma'(x) = u \\ 1 & , \text{ otherwise } . \end{cases}
\end{aligned}$$

We put $\mathcal{V} = \mathrm{free}(\psi) \cup \mathrm{free}(\mathrm{Val}_x\,\theta_1)$ and let γ be a $(\mathcal{V}, s)$-assignment. Again, to refer to different D-labelings of s, we define for $v \in V(s)$

$$\begin{aligned}
\mathrm{Lab}_{s,D}^{\theta_1}(v) &= [\![\theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \ , \\
\mathrm{Lab}_{s,D}^{\psi \text{ lm } \theta_1}(v) &= \begin{cases} [\![\psi]\!]_{\mathcal{V}}(s,\gamma) \diamond \mathrm{Lab}_{s,D}^{\theta_1}(v) & , \text{ if } v = u \\ \mathrm{Lab}_{s,D}^{\theta_1}(v) & , \text{ otherwise} \end{cases} , \\
\mathrm{Lab}_{s,D}^{(\min(x)\to\psi)\otimes\theta_1} &= [\![(\min(x) \to \psi) \otimes \theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \ .
\end{aligned}$$

It follows that for all $v \in V(s)$

$$\begin{aligned}
\mathrm{Lab}_{s,D}^{\psi \text{ lm } \theta_1}(v) &= \begin{cases} [\![\psi]\!]_{\mathcal{V}}(s,\gamma) \diamond \mathrm{Lab}_{s,D}^{\theta_1}(v) & , \text{ if } v = u \\ \mathrm{Lab}_{s,D}^{\theta_1}(v) & , \text{ otherwise} \end{cases} \\
&= \begin{cases} [\![\psi \otimes \theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) & , \text{ if } v = u \\ [\![\theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) & , \text{ otherwise} \end{cases} \\
&= \begin{cases} [\![\psi \otimes \theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) & , \text{ if } \gamma(x) = u \\ [\![\theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) & , \text{ otherwise} \end{cases} \\
&= [\![(\min(x) \to \psi) \otimes \theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \\
&= \mathrm{Lab}_{s,D}^{(\min(x)\to\psi)\otimes\theta_1}(v) \ .
\end{aligned}$$

Using this equation at ** and the assumption that $\mathbb{PD}$ is left-multiplicative at

equation *, we get

$$\begin{aligned}
[\![\psi \otimes \mathrm{Val}_x\, \theta_1]\!]_{\mathcal{V}}(s,\gamma) &= [\![\psi]\!]_{\mathcal{V}}(s,\gamma) \diamond \mathrm{Val}(s, \mathrm{Lab}_{s,D}^{\theta_1}) \\
&\overset{*}{=} \mathrm{Val}(s, \mathrm{Lab}_{s,D}^{\psi \text{ lm } \theta_1}) \\
&\overset{**}{=} \mathrm{Val}(s, \mathrm{Lab}_{s,D}^{(\min(x)\to\psi)\otimes\theta_1}) \\
&= [\![\mathrm{Val}_x((\min(x) \to \psi) \otimes \theta_1)]\!]_{\mathcal{V}}(s,\gamma) \ .
\end{aligned}$$

(*d*) Since $\mathrm{const}(\psi_1)$ and $\mathrm{const}(\theta_1)$ commute, Lemma 3.8 implies that all values occurring at the evaluation of $[\![\psi_1]\!]$ and $[\![\theta_1]\!]$ commute too, which allows us to use the conditional commutativity of $\mathbb{PD}$. We put $\mathcal{V} = \mathrm{free}(\mathrm{Val}_x\, \psi_1) \cup \mathrm{free}(\mathrm{Val}_x\, \theta_1)$ and let $(s,\gamma) \in C^\sigma \times \Gamma_{(\mathcal{V},s)}$. Again, to refer to different D-labelings of s, we set for $v \in V(s)$

$$\begin{aligned}
\mathrm{Lab}_{s,D}^{\psi_1}(v) &= [\![\psi_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \ , \\
\mathrm{Lab}_{s,D}^{\theta_1}(v) &= [\![\theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \ , \\
\mathrm{Lab}_{s,D}^{\psi \text{ cc } \theta_1}(v) &= \mathrm{Lab}_{s,D}^{\psi_1}(v) \diamond \mathrm{Lab}_{s,D}^{\theta_1}(v) \\
&= [\![\psi_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \diamond [\![\theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \\
&= [\![\psi_1 \otimes \theta_1]\!]_{\mathcal{V}\cup\{x\}}(s, \gamma[x \to v]) \\
&= \mathrm{Lab}_{s,D}^{\psi_1\otimes\theta_1}(v) \ .
\end{aligned}$$

Using this equation at ** and the assumption that $\mathbb{PD}$ is conditionally commutative at equation *, we get

$$\begin{aligned}
[\![\mathrm{Val}_x\, \psi_1 \otimes \mathrm{Val}_x\, \theta_1]\!]_{\mathcal{V}}(s,\gamma) &= \mathrm{Val}(s, \mathrm{Lab}_{s,D}^{\psi_1}) \diamond \mathrm{Val}(s, \mathrm{Lab}_{s,D}^{\theta_1}) \\
&\overset{*}{=} \mathrm{Val}(s, \mathrm{Lab}_{s,D}^{\psi_1 \text{ cc } \theta_1}) \\
&\overset{**}{=} \mathrm{Val}(s, \mathrm{Lab}_{s,D}^{\psi_1\otimes\theta_1}) \\
&= [\![\mathrm{Val}_x(\psi_1 \otimes \theta_1)]\!]_{\mathcal{V}}(s,\gamma) \ . \qquad \square
\end{aligned}$$

We are now ready to conclude the proof of Theorem 3.6.

Proof of Theorem 3.6. We show the statements (*a*) and (*b*) by induction on the length of the formula φ and check the cases where φ is Val-restricted within (*a*) and (*b*). The base case for almost FO-boolean formulas is trivial. For the induction step, we have to check only the weighted product. Therefore, we may assume that $\varphi = \psi \otimes \theta$ and both ψ and θ are strongly-$\otimes$-restricted (and resp. Val-restricted). If θ is FO-boolean, we can put $\varphi' = \varphi$.

(*a*) We assume that ψ is almost FO-boolean. By an analysis of the structure of θ, we will construct a strongly-$\otimes$-restricted formula φ' with $[\![\varphi']\!] = [\![\psi \otimes \theta]\!] = [\![\varphi]\!]$.

Case 1: If θ is almost FO-boolean, then so is φ, hence $\varphi' = \varphi$ is strongly-$\otimes$-restricted (and resp. Val-restricted).

Case 2: Assume $\theta = \theta_1 \otimes \theta_2$ and θ is not almost FO-boolean. Since θ is strongly-$\otimes$-restricted θ_1 or θ_2 is FO-boolean. We may assume θ_1 to be FO-boolean. We put $\mathcal{V} = \text{free}(\psi) \cup \text{free}(\theta_1) \cup \text{free}(\theta_2)$. Now, we apply the induction hypothesis to $\psi \otimes \theta_2$ to obtain a strongly-$\otimes$-restricted formula φ_1 with $[\![\varphi_1]\!] = [\![\psi \otimes \theta_2]\!]$. Then $\varphi' = \theta_1 \otimes \varphi_1$ is strongly-$\otimes$-restricted and for each $(s, \gamma) \in C^\sigma \times \Gamma_{(\mathcal{V},s)}$ we have

$$\begin{aligned}
[\![\varphi]\!](s,\gamma) &= [\![\psi \otimes (\theta_1 \otimes \theta_2)]\!]_\mathcal{V}(s,\gamma) \\
&= \begin{cases} [\![\psi \otimes \theta_2]\!]_\mathcal{V}(s,\gamma) & , \text{ if } [\![\theta_1]\!]_\mathcal{V}(s,\gamma) = 1 \\ 0 & , \text{ otherwise} \end{cases} \\
&= [\![\theta_1 \otimes \varphi_1]\!]_\mathcal{V}(s,\gamma) = [\![\varphi']\!]_\mathcal{V}(s,\gamma) \ .
\end{aligned}$$

Case 3: Assume $\theta = \theta_1 \oplus \theta_2$. We put $\mathcal{V} = \text{free}(\psi) \cup \text{free}(\theta_1) \cup \text{free}(\theta_2)$ and apply Lemma 3.10 (a):

$$\begin{aligned}
[\![\varphi]\!] &= [\![\psi \otimes (\theta_1 \oplus \theta_2)]\!]_\mathcal{V} \\
&= [\![\psi \otimes \theta_1]\!]_\mathcal{V} + [\![\psi \otimes \theta_2]\!]_\mathcal{V} \ .
\end{aligned}$$

Since ψ is almost FO-boolean, we can apply the induction hypothesis to $\psi \otimes \theta_1$ and $\psi \otimes \theta_2$. Hence, there are strongly-$\otimes$-restricted (and resp. Val-restricted) formulas φ_1 and φ_2 such that $[\![\varphi_1]\!] = [\![\psi \otimes \theta_1]\!]$ and $[\![\varphi_2]\!] = [\![\psi \otimes \theta_2]\!]$. Then $\varphi_1 \oplus \varphi_2$ is strongly-$\otimes$-restricted (and resp. Val-restricted) and $[\![\varphi_1 \oplus \varphi_2]\!]_\mathcal{V} = [\![\varphi_1]\!]_\mathcal{V} + [\![\varphi_2]\!]_\mathcal{V} = [\![\varphi]\!]$.

Case 4: Assume $\theta = \bigoplus_x \theta_1$ and ψ contains no x. We apply Lemma 3.10 (a):

$$\begin{aligned}
[\![\varphi]\!] &= [\![\psi \otimes \textstyle\bigoplus_x \theta_1]\!] \\
&= [\![\textstyle\bigoplus_x (\psi \otimes \theta_1)]\!] \ .
\end{aligned}$$

We apply the induction hypothesis to $\psi \otimes \theta_1$ to obtain a strongly-$\otimes$-restricted formula φ_1 such that $[\![\varphi_1]\!] = [\![\psi \otimes \theta_1]\!]$. Then $\bigoplus_x \varphi_1$ is strongly-$\otimes$-restricted and $[\![\bigoplus_x \varphi_1]\!] = [\![\varphi]\!]$.

Case 4': Assume $\theta = \bigoplus_x \theta_1$ and ψ contains x. Let z be a variable not occurring in ψ. We rename the variable x to z at every occurrence in $\bigoplus_x \theta_1$ to get $\theta' = \bigoplus_z \theta'_1$ with $[\![\theta]\!] = [\![\theta']\!]$. Since ψ does not contain z, we can apply Case 4 to $\psi \otimes \theta'$.

Case 5: Assume $\theta = \bigoplus_X \theta_1$. This is done analogously to Cases 4 and 4'.

Case 6: Assume $\theta = \text{Val}_x \theta_1$. By renaming analogously to Case 4', we may assume that ψ does not contain x.

First, let $\mathbb{PD}$ be left-Val-distributive. Applying Lemma 3.10 (b), we get

$$\begin{aligned}
[\![\varphi]\!] &= [\![\psi \otimes \text{Val}_x \theta_1]\!] \\
&= [\![\text{Val}_x (\psi \otimes \theta_1)]\!] \ .
\end{aligned}$$

We apply the induction hypothesis to $\psi \otimes \theta_1$ to obtain a strongly-$\otimes$-restricted formula φ_1 such that $[\![\varphi_1]\!] = [\![\psi \otimes \theta_1]\!]$. Then $\mathrm{Val}_x\, \varphi_1$ is strongly-$\otimes$-restricted and $[\![\mathrm{Val}_x\, \varphi_1]\!] = [\![\varphi]\!]$. If φ is Val-restricted, θ_1 is almost FO-boolean. In this case, we can put directly $\varphi' = \mathrm{Val}_x(\psi \otimes \theta_1)$. Then φ' is strongly-$\otimes$-restricted and Val-restricted because ψ and θ_1 are almost FO-boolean formulas.

Now, let $\mathbb{PD}$ be left-multiplicative. Applying Lemma 3.10 (c) and using that the formula $\min(x)$ is FO-boolean, we get

$$\begin{aligned}[\![\varphi]\!] &= [\![\psi \otimes \mathrm{Val}_x\, \theta_1]\!] \\ &= [\![\mathrm{Val}_x((\min(x) \to \psi) \otimes \theta_1)]\!] \\ &= [\![\mathrm{Val}_x((\neg \min(x) \otimes \theta_1) \oplus (\min(x) \otimes \psi \otimes \theta_1))]\!] \ .\end{aligned}$$

We apply the induction hypothesis to $\psi \otimes \theta_1$ to obtain a strongly-$\otimes$-restricted formula φ_1 such that $[\![\varphi_1]\!] = [\![\psi \otimes \theta_1]\!]$. Then $\varphi' = \mathrm{Val}_x((\neg \min(x) \otimes \theta_1) \oplus (\min(x) \otimes \varphi_1))$ is strongly-$\otimes$-restricted since $\min(x)$ is FO-boolean. Furthermore, $[\![\varphi]\!] = [\![\varphi']\!]$.

If φ is Val-restricted, we can put directly $\varphi' = \mathrm{Val}_x((\min(x) \to \psi) \otimes \theta_1)$. Then φ' is strongly-$\otimes$-restricted and Val-restricted because $\min(x) \to \psi$ and θ_1 are almost FO-boolean formulas.

(b) Again, we may assume that $\varphi = \psi \otimes \theta$ where ψ and θ are strongly-$\otimes$-restricted (and resp. Val-restricted). In case ψ is almost FO-boolean, we can apply the argument of part (a) to obtain the claim. Therefore, we may assume that ψ is not almost FO-boolean and that $\mathrm{const}(\psi)$ and $\mathrm{const}(\theta)$ commute. If θ is almost FO-boolean, by Lemma 3.9 we have $[\![\psi \otimes \theta]\!] = [\![\theta \otimes \psi]\!]$, and we can apply part (a). Therefore, let now θ be not almost FO-boolean. In case that θ is a disjunction or a $\bigoplus$-quantification, we can proceed as in Case 3, Case 4 resp. Case 5 of part (a). If θ is a conjunction, we can proceed as in Case 2. If ψ is a disjunction, a $\bigoplus$-quantification or a conjunction, we observe that $[\![\psi \otimes \theta]\!] = [\![\theta \otimes \psi]\!]$ and apply the previous cases.

It remains to consider that both ψ and θ are Val-quantifications that is $\psi = \mathrm{Val}_x\, \psi_1$ and $\theta = \mathrm{Val}_y\, \theta_1$. Choose a variable z not occurring in ψ or θ. We rename all occurrences of x in ψ and of y in θ by z, i.e., we consider $\psi' = \mathrm{Val}_z\, \psi_1'$ and $\theta' = \mathrm{Val}_z\, \theta_1'$ where $[\![\psi_1']\!] = [\![\psi_1]\!]$ and $[\![\theta_1']\!] = [\![\theta_1]\!]$. So $[\![\varphi]\!] = [\![\psi' \otimes \theta']\!]$. Since ψ_1 and ψ_1' resp. θ_1 and θ_1' contain the same constants, $\mathrm{const}(\psi_1')$ and $\mathrm{const}(\theta_1')$ commute. Applying Lemma 3.10 (d) we get

$$\begin{aligned}[\![\varphi]\!] &= [\![\mathrm{Val}_z\, \psi_1' \otimes \mathrm{Val}_z\, \theta_1']\!] \\ &= [\![\mathrm{Val}_z(\psi_1' \otimes \theta_1')]\!] \ .\end{aligned}$$

Hence, we can apply the induction hypothesis to $\psi_1' \otimes \theta_1'$ to obtain a strongly-$\otimes$-restricted formula φ_1 such that $[\![\varphi_1]\!] = [\![\psi_1' \otimes \theta_1']\!]$. Then $\mathrm{Val}_z\, \varphi_1$ is strongly-$\otimes$-restricted and $[\![\mathrm{Val}_z\, \varphi_1]\!] = [\![\varphi]\!]$.

If φ is Val-restricted, we can put directly $\varphi' = \mathrm{Val}_z(\psi_1' \otimes \theta_1')$. Then φ' is strongly-$\otimes$-restricted and Val-restricted because ψ_1' and θ_1' are almost FO-boolean formulas. $\square$

Chapter 4

Weighted Automata for Infinite Nested Words

Nested words, introduced by Alur and Madhusudan [AM09], capture models with both a natural sequence of positions and a hierarchical nesting of these positions. Prominent examples include XML documents and executions of recursively structured programs. Automata on nested words, logical specifications, and corresponding languages of nested words have been intensively studied, see [AAB$^+$08, AM09, LMS04].

In this chapter, we will investigate quantitative nested word automata and suitable quantitative MSO logics. We will concentrate on infinite nested words, although our results also hold for finite nested words. We employ the stair Muller nested word automata of [AM09, LMS04] since these can be determinized without losing expressive power. As weight structures, we use valuation monoids and product valuation monoids (cf. Section 3.2 and [DM12]). These include infinite products as in totally complete semirings [DR06], but also computations of long-time averages or discountings of weights. As example for such a setting, we give the calculation of the long-time ratio of bracket-free positions in prefixes of an infinite nested word. We employ our Transformation Theorem 3.6 to show that under suitable assumptions on the product valuation monoid $\mathbb{PD}$, two resp. three versions of our weighted MSO logic have the same expressive power. In particular, if $\mathbb{PD}$ is commutative, then any weighted MSO-formula is equivalent to one in which conjunctions occur only between 'classical' boolean formulas and constants.

Furthermore, we show under suitable assumptions on the valuation monoid that our weighted MSO logics have the same expressive power as weighted nested word automata. These assumptions on the valuation monoid are satisfied by long-time average resp. discounted computations of weights; therefore our results apply to these settings. All our constructions of automata from formulas and vice versa are effective.

The results of this section were published in [DD17] and in an extended

abstract of this paper in [DD14]. One case of the theorem connecting weighted nested word automata and weighted MSO logics was part of [Düc13].

4.1 Infinite Nested Words

In this section, we describe classical unweighted automata and logics on infinite nested words (nested ω-words). As introduced in Section 2.2, a nested word (w, ν) is a word together with a non-crossing matching relation. We start with an automata model reading infinite nested words [AM09] with the appropriate acceptance condition to be determinizable [LMS04] and slightly adjusted to fit our notations.

Definition 4.1. A *deterministic stair Muller nested word automaton (sMNWA)* over Σ is a quadruple $\mathcal{A} = (Q, q_0, \delta, \mathfrak{F})$, where $\delta = (\delta_{\text{call}}, \delta_{\text{int}}, \delta_{\text{ret}})$, consisting of:

- a finite set of states Q,
- an initial state $q_0 \in Q$,
- a set $\mathfrak{F} \subseteq 2^Q$ of accepting sets of states,
- the transition functions $\delta_{\text{call}}, \delta_{\text{int}} : Q \times \Sigma \to Q$,
- the transition function $\delta_{\text{ret}} : Q \times Q \times \Sigma \to Q$.

A *run* r of the sMNWA $\mathcal{A}$ on the nested ω-word $nw = (a_1 a_2 ..., \nu)$ is an infinite sequence of states $r = (q_0, q_1, ...)$ where $q_i \in Q$ for each $i \in \mathbb{N}$ and q_0 is the initial state of $\mathcal{A}$ such that for each $i \in \mathbb{N}_+$ the following holds:

$$\begin{cases} \delta_{\text{call}}(q_{i-1}, a_i) = q_i & \text{, if } \nu(i,j) \text{ for some } j > i \\ \delta_{\text{int}}(q_{i-1}, a_i) = q_i & \text{, if } i \text{ is an internal} \\ \delta_{\text{ret}}(q_{i-1}, q_{j-1}, a_i) = q_i & \text{, if } \nu(j,i) \text{ for some } 1 \leq j < i \\ \delta_{\text{ret}}(q_{i-1}, q_0, a_i) = q_i & \text{, if } \nu(-\infty, i) \ . \end{cases}$$

We call $i \in \mathbb{N}$ a *top-level position* if there exist no positions $j, k \in \mathbb{N}$ with $j < i < k$ and $\nu(j,k)$. We define

$$Q^t_\infty(r) = \{q \in Q \mid q = q_i \text{ for infinitely many top-level positions } i\} \ .$$

A run r of an sMNWA is *accepted* if $Q^t_\infty(r) \in \mathfrak{F}$. An sMNWA $\mathcal{A}$ *accepts* the nested ω-word nw if there is an accepted run of $\mathcal{A}$ on nw. We call $L(\mathcal{A})$ the set of all accepted nested ω-words of $\mathcal{A}$. We call a language L of nested ω-words *regular* if there is an sMNWA $\mathcal{A}$ with $L(\mathcal{A}) = L$.

Alur and Madhusudan considered in [AM09] nondeterministic Büchi NWA and nondeterministic Muller NWA. They showed that the deterministic versions of these automata have strictly less expressive power than the nondeterministic automata. However, referring to Löding, Madhusudan, and Serre [LMS04], Alur

and Madhusudan stated that deterministic stair Muller NWA have the same expressive power as their nondeterministic versions as well as nondeterministic Büchi NWA. Moreover, they proved that the class of regular languages of nested ω-words is closed under union, intersection, and complement [AM09].

Definition 4.2. The monadic second order logic for infinite nested words, $\mathrm{MSO}(\mathrm{NW}(\Sigma))$, contains exactly all formulas φ which are given by the following syntax:

$$\varphi ::= \mathrm{Lab}_a(x) \mid \mathrm{call}(x) \mid \mathrm{ret}(x) \mid x \leq y \mid \nu(x,y) \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \exists X.\varphi$$

where $a \in \Sigma$, x, y are first order variables, and X is a second order variable.

The semantics of these formulas is given in a natural way, cf. [AM09]. The predicates $\mathrm{call}(x)$ and $\mathrm{ret}(x)$ are true if x is a call or return position, respectively. They are used to, in contrast to $\nu(x,y)$, additionally refer to pending calls or pending returns. If φ is sentence, then $L(\varphi) = \{nw \in \mathrm{NW}^\omega(\Sigma) \mid nw \models \varphi\}$ is *the language defined by* φ.

Theorem 4.3 (Alur, Madhusudan [AM09])**.** *Let L be a language of nested ω-words over Σ. Then L is regular if and only if L is definable by some $\mathrm{MSO}(\mathrm{NW}(\Sigma))$-sentence φ.*

4.2 Weighted Stair Muller Nested Word Automata

In this section, we introduce weighted versions of stair Muller nested word automata. As weight structures, we employ valuation monoids together with their properties, as introduced in Section 3.2, for infinite nested words (see also [DM12] for valuation monoids over words). We let D^ω comprise all infinite sequences of elements of D and recall the important definitions.

Definition 4.4. An *ω-valuation monoid* $\mathbb{D} = (D, +, \mathrm{Val}^\omega, 0)$ is a complete monoid $(D, +, 0)$ equipped with an *ω-valuation function* $\mathrm{Val}^\omega : D^\omega \to D$ with $\mathrm{Val}^\omega((d_i)_{i\in\mathbb{N}}) = 0$ if $d_i = 0$ for some $i \in \mathbb{N}$.

A *product ω-valuation monoid* $\mathbb{PD} = (D, +, \mathrm{Val}^\omega, \diamond, 0, 1)$ (short *ω-pv-monoid*) is an ω-valuation monoid $(D, +, \mathrm{Val}^\omega, 0)$ with a constant $1 \in D$ and an operation $\diamond : D^2 \to D$ satisfying $\mathrm{Val}^\omega(1^\omega) = 1$, $0 \diamond d = d \diamond 0 = 0$, and $1 \diamond d = d \diamond 1 = d$ for all $d \in D$.

Following [DM12], we give an example by adapting the product valuation monoids incorporating average and discounting to ω-words or nested ω-words as follows.

Example 4.1. We set $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$ and $-\infty + \infty = -\infty$. We let

$$\mathbb{PD}_1 = (D_1, +, \mathrm{Val}^\omega, \diamond, 0, 1) = (\bar{\mathbb{R}}, \sup, \lim\ \mathrm{avg}, +, -\infty, 0),$$

where $$\lim\ \mathrm{avg}((d_i)_{i\in\mathbb{N}}) = \liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} d_i \ .$$

Let $0 < \lambda < 1$ and $\bar{\mathbb{R}}_+ = \{x \in \bar{\mathbb{R}} \mid x \geq 0\} \cup \{-\infty\}$. We put

$$\mathbb{PD}_2 = (D_2, +, \mathrm{Val}^\omega, \diamond, 0, 1) = (\bar{\mathbb{R}}_+, \sup, \mathrm{disc}_\lambda, +, -\infty, 0),$$

where $$\mathrm{disc}_\lambda((d_i)_{i\in\mathbb{N}}) = \lim_{n\to\infty} \sum_{i=1}^{n} \lambda^{i-1} d_i \ .$$

Then $\mathbb{PD}_1$ is a left-$+$-distributive and left-Val^ω-distributive ω-pv-monoid but not conditionally commutative. Furthermore, $\mathbb{PD}_2$ is a left-multiplicative cc-ω-valuation semiring. $\diamond$

Definition 4.5. A *weighted stair Muller nested word automaton (wsMNWA)* $\mathcal{A} = (Q, I, \delta, \mathfrak{F})$, where $\delta = (\delta_{\mathrm{call}}, \delta_{\mathrm{int}}, \delta_{\mathrm{ret}})$, over the alphabet Σ and the ω-valuation monoid $(D, +, \mathrm{Val}^\omega, 0)$ consists of:

- a finite set of states Q,
- a set $I \subseteq Q$ of initial states,
- a set $\mathfrak{F} \subseteq 2^Q$ of accepting sets of states,
- the weight functions $\delta_{\mathrm{call}}, \delta_{\mathrm{int}} : Q \times \Sigma \times Q \to D$,
- the weight function $\delta_{\mathrm{ret}} : Q \times Q \times \Sigma \times Q \to D$.

A *run* r of the wsMNWA $\mathcal{A}$ on the nested ω-word $nw = (a_1 a_2 ..., \nu)$ is an infinite sequence of states $r = (q_0, q_1, ...)$. We denote by $\mathrm{wt}_\mathcal{A}(r, nw, i)$ the weight of the transition of r used at position $i \in \mathbb{N}_+$, defined as follows

$$\mathrm{wt}_\mathcal{A}(r, nw, i) = \begin{cases} \delta_{\mathrm{call}}(q_{i-1}, a_i, q_i) & , \text{ if } \nu(i,j) \text{ for some } j > i \\ \delta_{\mathrm{int}}(q_{i-1}, a_i, q_i) & , \text{ if } i \text{ is an internal} \\ \delta_{\mathrm{ret}}(q_{i-1}, q_{j-1}, a_i, q_i) & , \text{ if } \nu(j,i) \text{ for some } 1 \leq j < i \\ \delta_{\mathrm{ret}}(q_{i-1}, q_0, a_i, q_i) & , \text{ if } \nu(-\infty, i) \ . \end{cases} \tag{4.1}$$

Then, we define the *weight* $\mathrm{wt}_\mathcal{A}(r, nw)$ *of* r *on* nw by letting

$$\mathrm{wt}_\mathcal{A}(r, nw) = \mathrm{Val}^\omega((\mathrm{wt}_\mathcal{A}(r, nw, i))_{i\in\mathbb{N}_+}) \ .$$

We define top-level positions and the set $Q^t_\infty(r)$ as before. A run r is *accepted* if $q_0 \in I$ and $Q^t_\infty(r) \in \mathfrak{F}$. We denote by $acc(\mathcal{A})$ the set of all accepted runs in $\mathcal{A}$.

We define the *behavior of the automaton* $\mathcal{A}$ as the function $[\![\mathcal{A}]\!] : \mathrm{NW}^\omega(\Sigma) \to D$ given by (where as usual, empty sums are defined to be 0)

$$\begin{aligned}[\![\mathcal{A}]\!](nw) &= \sum_{r \in \mathrm{acc}(\mathcal{A})} \mathrm{wt}_{\mathcal{A}}(r, nw) \\ &= \sum_{r \in \mathrm{acc}(\mathcal{A})} \mathrm{Val}^\omega((\mathrm{wt}_{\mathcal{A}}(r, nw, i))_{i \in \mathbb{N}_+}) \ .\end{aligned}$$

We call every function $S : \mathrm{NW}^\omega(\Sigma) \to D$ a *nested ω-word series* (short: *series*). We say that a wsMNWA $\mathcal{A}$ *recognizes* or *accepts* a series S if $[\![\mathcal{A}]\!] = S$. We call a series S *regular* if there exists an automaton $\mathcal{A}$ accepting it.

Example 4.2. Within the following example, we call a position i of a nested ω-word $nw = (w, \nu)$ *bracket-free* if there are no positions $j, k \in (\mathbb{N}_+ \cup \{-\infty, \infty\})$ with $j < i < k$ and $\nu(j, k)$. This requirement for i is stronger than being a top-level position because j and k can be $-\infty$ or ∞, thus also preventing i from being in the scope of pending calls and pending returns. Only for well-matched nested ω-words, i.e., nested ω-words without pending edges, the two properties coincide.

We consider the series S assigning to every nested ω-word nw the greatest accumulation point of the ratio of bracket-free positions in finite prefixes of nw. To model S, we use the ω-valuation monoid $\mathbb{D} = (\bar{\mathbb{R}}, \sup, \limsup, -\infty)$. If we want to analyze this property for well-matched nested ω-words only, then the automaton $\mathcal{A}_1$ given in Figure 4.1 recognizes S. We denote the call transitions with $\langle\Sigma$ and the return transitions with $\Sigma\rangle/q$, where q has to be the state where the last open call was encountered. The weights 0 and 1 are given in brackets.

The automaton $\mathcal{A}_1$ starts in q_0 and the first position is always a top-level position. In a well-matched nested ω-word, every call of the form $\langle\Sigma$ that is read in q_0 is eventually answered by a return of the form $\Sigma\rangle/q_0$. It follows that we visit q_0 infinitely often at a top-level position. Then, by the definition of top-level positions, the internal transitions in q_0 and both the calls going from q_0 to q_1 and the returns going from q_1 to q_0 are top-level positions (and thus bracket-free). That is why we give these transitions the weight 1. All other transitions are not top-level and get the weight 0. Furthermore, if we eventually read no more calls, then q_1 is only visited finitely often, otherwise q_1 is also visited infinitely often at top-level positions (which are exactly the call positions going from q_0 to q_1). Therefore, the set of accepting sets of states of $\mathcal{A}_1$ has to be $\mathfrak{F}_1 = \{\{q_0\}, \{q_0, q_1\}\}$.

In the general case, including pending edges, the automaton $\mathcal{A}_2$ given in Figure 4.2 recognizes S. It works in part as the automaton $\mathcal{A}_1$ but has to handle pending calls and pending returns as follows: Let nw be a nested ω-word. As before, $\mathcal{A}_2$ is in q_0 whenever the previous and the next transition are bracket-free, that is, there are no calls open and in the remainder of nw there

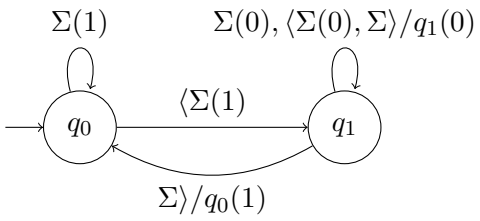


Figure 4.1: The wsMNWA $\mathcal{A}_1$ of Example 4.2 for well-matched nested ω-words with $\mathfrak{F}_1 = \{\{q_0\}, \{q_0, q_1\}\}$.

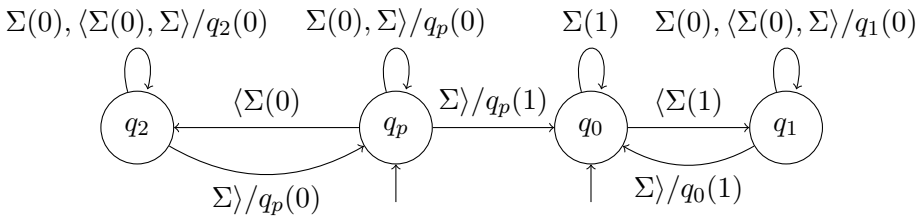


Figure 4.2: The wsMNWA $\mathcal{A}_2$ of Example 4.2 also reading nested ω-words with pending edges. The accepting set is given by $\mathfrak{F}_2 = \{\{q_p\}, \{q_2, q_p\}, \{q_0, q_1\}, \{q_0\}, \{q_1\}\}$.

are no pending returns. Also as before, the automaton is in q_1 whenever there are calls open and no pending returns left. The automaton is in q_p whenever there are no calls open but there are pending returns left. And finally, $\mathcal{A}_2$ is in q_2 whenever there are calls open and pending returns left.

At the start and whenever reading a pending return of nw, $\mathcal{A}_2$ guesses whether there is another pending return left in the remainder of nw. If yes, it stays in q_p. If not, it goes to q_0. The correctness of this guess is ensured by the fact that the automaton cannot read pending returns in q_0 and the following observation. If $\mathcal{A}_2$ stays in q_p, although there are no pending returns left, then this run will never reach q_0 and thus the weight of this run is 0, which is smaller than or equal to the weight of the run where we switched to q_0 at the last pending return.

If nw has infinitely many pending returns, then the automaton needs to have an accepting run (with weight 0) that never reaches q_0, thus, we add $\{q_p\}$ and $\{q_2, q_p\}$ to $\mathfrak{F}_2$. Note that in this case, nw can have no pending calls.

If nw has finitely many pending returns and at least a pending call, then $\mathcal{A}_2$ will stay forever in q_1 after reading the first pending call, thus, we also add $\{q_1\}$ to $\mathfrak{F}_2$. As this run yields the weight 0, due to the lim avg, we could similarly add $\{q_2\}$ to $\mathfrak{F}_2$. $\diamond$

4.3 Regularity of Valuation Monoids

Valuation monoids are a very general structure. In the following, we study a natural condition on the valuation function, called *regularity*, that will ensure some basic compatibility of the weighted logic and weighted automata over valuation monoids. Intuitively, an ω-valuation monoid $\mathbb{D}$ is called *regular* if all constant series of $\mathbb{D}$ are regular.

The principal notion of regularity of a weight structure can also be applied to other relational structures. Especially in the context of graphs, this opens some interesting questions even for semirings, see Section 6.2.4 for a related discussion.

Definition 4.6. An ω-valuation monoid $\mathbb{D}$ is called *wsMNWA-regular*, short *regular*, if for any non-empty alphabet Σ, we have: For each $d \in D$, there exists a wsMNWA $\mathcal{A}_d$ with $[\![\mathcal{A}_d]\!](nw) = d$ for all nested ω-words nw over Σ. An ω-pv-monoid $\mathbb{PD}$ is called *regular* if its underlying ω-valuation monoid is regular.

For example, the ω-pv-monoids of Example 4.1 incorporating computation of average and discounting are regular, which is shown as follows.

Analogously to Droste and Meinecke [DM12], we can prove that every left-distributive ω-pv-monoid (cf. Section 3.2) is regular. Indeed, if $\mathbb{PD}$ is left-Val-distributive, then for every d, we can construct a deterministic automaton $\mathcal{A}_d$ assigning the weight d at every position for every word. Hence, $[\![\mathcal{A}_d]\!] = \mathrm{Val}^\omega(d, d, ...) = d \diamond \mathrm{Val}^\omega(1, 1...) = d \diamond 1 = d$, using the axioms of the pv-monoid. If $\mathbb{PD}$ is left-multiplicative (note that nested words can be pointed with their first symbol being the root), then for every d, we can construct a deterministic automaton $\mathcal{A}_d$ assigning the weight d at the first position and 1 at every following position for every word. Therefore, we get $[\![\mathcal{A}_d]\!] = \mathrm{Val}^\omega(d, 1, 1, ...) = d \diamond \mathrm{Val}^\omega(1, 1, ...) = d \diamond 1 = d$.

Another general property yielding regularity is the following. We call $\mathbb{PD}$ ω-regular if for all $d \in D$, there exists an ultimately periodic sequence $(d_i)_{i\in\mathbb{N}}$ such that $d = \mathrm{Val}^\omega((d_i)_{i\in\mathbb{N}})$.

Proposition 4.7. *Every ω-regular product ω-valuation monoid $\mathbb{PD}$ is regular.*

Proof. For all d, we can construct an automaton $\mathcal{A}_d$ simulating the encountered weights of the sequence $(d_i)_{i\in\mathbb{N}}$ for every nested ω-word. The periodicity of this sequence ensures that we only need a finite number of states. □

In the following, we study closure properties of (nested ω-word-) series. As usual, we extend the operation $+$ and $\diamond$ to series $S, T : \mathrm{NW}^\omega(\Sigma) \to D$ by means of pointwise definitions as follows:

$$(S \star T)(nw) = S(nw) \star T(nw) \text{ for each } nw \in \mathrm{NW}^\omega(\Sigma), \star \in \{+, \diamond\} \ .$$

Proposition 4.8. *The sum of two regular series over* $\mathrm{NW}^\omega(\Sigma)$ *is again regular.*

Proof. We use a standard disjoint union of two wsMNWA accepting the given series to obtain a wsMNWA for the sum. □

Now, we introduce a very special class of series, the *regular step function*, which will play a critical role in our following proofs. Regular step function are series that only assume finitely many different values.

Let $\mathbb{PD}$ be an ω-pv-monoid. We let $d \in D$ also denote the constant series with value d, i.e., $[\![d]\!](nw) = d$ for each $nw \in \mathrm{NW}^\omega(\Sigma)$. For $L \subseteq \mathrm{NW}^\omega(\Sigma)$, we define the *characteristic series* $\mathbb{1}_L : \mathrm{NW}^\omega(\Sigma) \to D$ by letting $\mathbb{1}_L(nw) = 1$ if $nw \in L$, and $\mathbb{1}_L(nw) = 0$ otherwise. We call a series S a *regular step function* if

$$S = \sum_{i=1}^{k} d_i \diamond \mathbb{1}_{L_i} \ , \tag{4.2}$$

where L_i are regular languages of nested ω-words forming a partition of $\mathrm{NW}^\omega(\Sigma)$ and $d_i \in D$ for each $i \in \{1, ..., k\}$; so $S(nw) = d_i$ iff $nw \in L_i$, for each $i \in \{1, ..., k\}$.

In the following, we define automata constructions, which will be used in the proofs of this section. Let the following wsMNWA operate over Σ and an ω-pv-monoid $\mathbb{PD}$.

Definition 4.9. (a) Let $\mathcal{A} = (Q, q_0, \delta, \mathfrak{F})$ with $\delta = (\delta_{\mathrm{call}}, \delta_{\mathrm{int}}, \delta_{\mathrm{ret}})$ be a deterministic sMNWA. We define the wsMNWA $\mathcal{A}^{\mathbb{1}} = (Q, \{q_0\}, \delta^{\mathbb{1}}, \mathfrak{F})$ with $\delta^{\mathbb{1}} = (\delta^{\mathbb{1}}_{\mathrm{call}}, \delta^{\mathbb{1}}_{\mathrm{int}}, \delta^{\mathbb{1}}_{\mathrm{ret}})$ as follows

$$\delta^{\mathbb{1}}_{\mathrm{call}}(q_1, a, q_2) = \begin{cases} 1 & , \text{ if } \delta_{\mathrm{call}}(q_1, a) = q_2 \\ 0 & , \text{ otherwise } , \end{cases}$$

$$\delta^{\mathbb{1}}_{\mathrm{int}}(q_1, a, q_2) = \begin{cases} 1 & , \text{ if } \delta_{\mathrm{int}}(q_1, a) = q_2 \\ 0 & , \text{ otherwise } , \end{cases}$$

$$\delta^{\mathbb{1}}_{\mathrm{ret}}(q_1, q_2, a, q_3) = \begin{cases} 1 & , \text{ if } \delta_{\mathrm{ret}}(q_1, q_2, a) = q_3 \\ 0 & , \text{ otherwise } . \end{cases}$$

(b) Let $\mathcal{A} = (Q_\mathcal{A}, I_\mathcal{A}, \delta^\mathcal{A}, \mathfrak{F}_\mathcal{A})$ and $\mathcal{B} = (Q_\mathcal{B}, I_\mathcal{B}, \delta^\mathcal{B}, \mathfrak{F}_\mathcal{B})$ be two wsMNWA. We define the *product automaton* $\mathcal{P}$ of $\mathcal{A}$ and $\mathcal{B}$ as $\mathcal{P} = (Q_\mathcal{A} \times Q_\mathcal{B}, I_\mathcal{A} \times I_\mathcal{B}, \delta, \mathfrak{F})$ where $\delta = (\delta_{\mathrm{call}}, \delta_{\mathrm{int}}, \delta_{\mathrm{ret}})$ and set

$$\delta_{\mathrm{call}}((q_1, p_1), a, (q_2, p_2)) = \delta^\mathcal{A}_{\mathrm{call}}(q_1, a, q_2) \diamond \delta^\mathcal{B}_{\mathrm{call}}(p_1, a, p_2) \ ,$$

$$\delta_{\mathrm{int}}((q_1, p_1), a, (q_2, p_2)) = \delta^\mathcal{A}_{\mathrm{int}}(q_1, a, q_2) \diamond \delta^\mathcal{B}_{\mathrm{int}}(p_1, a, p_2) \ ,$$

$$\delta_{\mathrm{ret}}((q_1, p_1), (q_2, p_2), a, (q_3, p_3)) = \delta^\mathcal{A}_{\mathrm{ret}}(q_1, q_2, a, q_3) \diamond \delta^\mathcal{B}_{\mathrm{ret}}(p_1, p_2, a, p_3) \ .$$

Using the projections $\pi_{Q_\mathcal{A}} : Q_\mathcal{A} \times Q_\mathcal{B} \to Q_\mathcal{A}$ and $\pi_{Q_\mathcal{B}} : Q_\mathcal{A} \times Q_\mathcal{B} \to Q_\mathcal{B}$ of $Q_\mathcal{A} \times Q_\mathcal{B}$ on $Q_\mathcal{A}$ and $Q_\mathcal{B}$, respectively, we construct $\mathfrak{F}$ as follows:

$$\mathfrak{F} = \{F \mid \pi_{Q_\mathcal{A}}(F) \in \mathfrak{F}_\mathcal{A} \text{ and } \pi_{Q_\mathcal{B}}(F) \in \mathfrak{F}_\mathcal{B}\} \ .$$

Remark 1. (a) Since $\mathcal{A}$ is deterministic, we get for all $nw \in \mathrm{NW}^\omega(\Sigma)$:

$$[\![\mathcal{A}^{\mathbb{1}}]\!](nw) = \left\{ \begin{array}{ll} 1 & , \text{ if } nw \in L(\mathcal{A}) \\ 0 & , \text{ otherwise} \end{array} \right\} = \mathbb{1}_{L(\mathcal{A})}(nw) \ .$$

(b) The combination of the acceptance conditions (see [DR06] and [DM12]) is equivalent to forcing for all $k \geq 0, q_i \in Q_\mathcal{A}, p_i \in Q_\mathcal{B}$:

$$\{(q_1, p_1), ..., (q_k, p_k)\} \in \mathfrak{F} \Leftrightarrow \{q_1, ..., q_k\} \in \mathfrak{F}_\mathcal{A} \text{ and } \{p_1, ..., p_k\} \in \mathfrak{F}_\mathcal{B} \ .$$

Note that in the sets $\{q_1, ..., q_k\}$, resp. $\{p_1, ..., p_k\}$, some of the elements may be equal, hence these sets could have smaller size than k.

Proposition 4.10. *Let $\mathbb{PD}$ be a regular ω-pv-monoid. Then each regular step function $S : \mathrm{NW}^\omega(\Sigma) \to D$ is regular. Furthermore, the set of all regular step functions is closed under $+$ and $\diamond$.*

Proof. First, let $d \in D$ and $L \subseteq \mathrm{NW}^\omega(\Sigma)$ be a regular language. We show that $d \diamond \mathbb{1}_L$ is regular. Let $\mathcal{A} = (Q, q_0, \delta, \mathfrak{F})$ be a deterministic sMNWA with $L(\mathcal{A}) = L$. We construct the wsMNWA $\mathcal{A}^{\mathbb{1}}$ (definition above) with $[\![\mathcal{A}^{\mathbb{1}}]\!] = \mathbb{1}_{L(\mathcal{A})} = \mathbb{1}_L$. Since $\mathbb{PD}$ is regular, there exists a wsMNWA $\mathcal{C} = (Q_\mathcal{C}, I_\mathcal{C}, \delta_\mathcal{C}, \mathfrak{F}_\mathcal{C})$ with $[\![\mathcal{C}]\!](nw) = d$ for all $nw \in \mathrm{NW}^\omega(\Sigma)$. We construct the product automaton $\mathcal{P}$ of $\mathcal{A}^{\mathbb{1}}$ and $\mathcal{C}$. Then $\mathcal{A}^{\mathbb{1}}$ has at most one run on every nested ω-word. Furthermore, a run of $\mathcal{P}$ is accepted iff its projections are accepted both on $\mathcal{A}^{\mathbb{1}}$ and on $\mathcal{C}$. Therefore, the following holds

$$\begin{aligned} [\![\mathcal{P}]\!](nw) &= \left\{ \begin{array}{ll} d & , \text{ if } nw \in L(\mathcal{A}) \\ 0 & , \text{ otherwise} \end{array} \right. \\ &= d \diamond \mathbb{1}_L(nw) \ . \end{aligned}$$

Now, since any regular step function is a sum of series of the form $d \diamond \mathbb{1}_L$ with L regular, by the argument above and Proposition 4.8, S is regular.

For the second part, let $S = \sum_{i=1}^k d_i \diamond \mathbb{1}_{L_i}$ and $T = \sum_{j=1}^\ell d'_j \diamond \mathbb{1}_{L'_j}$ be regular step functions where (L_i) and (L'_j) are forming partitions. Then

$$S + T = \sum_{i=1}^k \sum_{j=1}^\ell (d_i + d'_j) \diamond \mathbb{1}_{L_i \cap L'_j} \ ,$$

$$S \diamond T = \sum_{i=1}^k \sum_{j=1}^\ell (d_i \diamond d'_j) \diamond \mathbb{1}_{L_i \cap L'_j} \ .$$

Since $(L_i \cap L'_j)$ is also a partition and regular nested ω-word languages are closed under intersection, $S + T$ and $S \diamond T$ are regular step functions. $\square$

Next, we show that regular series are closed under projections. Let $\mathbb{D}$ be an ω-valuation monoid. Consider a mapping $h : \Sigma \to \Gamma$ between two alphabets. We can extend h to a function $h : \mathrm{NW}^\omega(\Sigma) \to \mathrm{NW}^\omega(\Gamma)$ as follows. Given a nested ω-word $nw = (w, \nu) = (a_1 a_2 ..., \nu) \in \mathrm{NW}^\omega(\Sigma)$, we define $h(nw) = h((a_1 a_2 ..., \nu)) = (h(a_1)h(a_2)..., \nu)$. Let $S : \mathrm{NW}^\omega(\Sigma) \to D$ be a series. Then we define $h(S) : \mathrm{NW}^\omega(\Gamma) \to D$ for each $nv \in \mathrm{NW}^\omega(\Gamma)$ by

$$h(S)(nv) = \sum (S(nw) \mid nw \in \mathrm{NW}^\omega(\Sigma), h(nw) = nv) \ .$$

Proposition 4.11. *Let $\mathbb{D}$ be an ω-valuation monoid, $S : \mathrm{NW}^\omega(\Sigma) \to D$ regular, and $h : \Sigma \to \Gamma$. Then $h(S) : \mathrm{NW}^\omega(\Gamma) \to D$ is regular.*

Proof. We follow an idea of [DV12]. Let $\mathcal{A} = (Q, I, \delta, \mathfrak{F})$ be a wsMNWA over $\mathbb{D}$ and Σ with $[\![\mathcal{A}]\!] = S$ and $\delta = (\delta_{\mathrm{call}}, \delta_{\mathrm{int}}, \delta_{\mathrm{ret}})$. We construct the wsMNWA $\mathcal{A}' = (Q', I', \delta', \mathfrak{F}')$ over $\mathbb{D}$ and Γ as follows:

- $Q' = Q \times \Sigma$, $I' = I \times \{a_0\}$ for some fixed $a_0 \in \Sigma$,
- for all $k \in \mathbb{N}_+$ and for all $a_i \in \Sigma$ for $i \in \{1, ..., k\}$

$$\{(q_1, a_1), ..., (q_k, a_k)\} \in \mathfrak{F}' \Leftrightarrow \{q_1, ..., q_k\} \in \mathfrak{F} \ ,$$

- $\delta' = (\delta'_{\mathrm{call}}, \delta'_{\mathrm{int}}, \delta'_{\mathrm{ret}})$ and for every $b \in \Gamma$ and $(q, a), (q', a'), (q'', a'') \in Q'$, we define:

$$\delta'_{\mathrm{call}}((q, a), b, (q', a')) = \begin{cases} \delta_{\mathrm{call}}(q, a', q') & , \text{ if } b = h(a') \\ 0 & , \text{ otherwise } , \end{cases}$$

$$\delta'_{\mathrm{int}}((q, a), b, (q', a')) = \begin{cases} \delta_{\mathrm{int}}(q, a', q') & , \text{ if } b = h(a') \\ 0 & , \text{ otherwise } , \end{cases}$$

$$\delta'_{\mathrm{ret}}((q, a), (q', a'), b, (q'', a'')) = \begin{cases} \delta_{\mathrm{ret}}(q, q', a'', q'') & , \text{ if } b = h(a'') \\ 0 & , \text{ otherwise } . \end{cases}$$

Analogously to [DM12] and [DP14b], this implies that for every run $r = (q_0, q_1, ...)$ of $\mathcal{A}$ on nw, there exists exactly one run $r' = ((q_0, a_0), (q_1, a_1), ...)$ of $\mathcal{A}'$ on nv with $h(nw) = nv$ and $\mathrm{wt}_{\mathcal{A}}(r, nw) = \mathrm{wt}_{\mathcal{A}'}(r', nv)$. In contrast to [DM12] and [DP14b], we have to check the acceptance condition of stair Muller NWA. Since h is respecting the matching relation, nv has the same top-level positions as nw. Hence, r' is accepted if and only if $q_0 \in I$ and $Q^t_\infty(r) \in \mathfrak{F}'$, i.e., if and only if r is accepted. This yields $[\![\mathcal{A}']\!](nv) = h([\![\mathcal{A}]\!])(nv)$. Therefore, $h(S) = h([\![\mathcal{A}]\!]) = [\![\mathcal{A}']\!]$ is regular. □

4.4 Weighted MSO-Logic for Nested Words

In the following, we study the instance of the weighted MSO logic for product valuation monoids of Section 3.3 for nested ω-words. In this case, the underlying unweighted fragment is the MSO logic for nested words introduced by Alur and Madhusudan [AM09].

$$
\begin{aligned}
[\![d]\!]_{\mathcal{V}}(nw,\gamma) &= d \quad \text{for all } d \in D \\
[\![\beta]\!]_{\mathcal{V}}(nw,\gamma) &= \begin{cases} 1 & \text{, if } (nw,\gamma) \models \beta \\ 0 & \text{, otherwise} \end{cases} \\
[\![\varphi \oplus \psi]\!]_{\mathcal{V}}(nw,\gamma) &= [\![\varphi]\!]_{\mathcal{V}}(nw,\gamma) + [\![\psi]\!]_{\mathcal{V}}(nw,\gamma) \\
[\![\varphi \otimes \psi]\!]_{\mathcal{V}}(nw,\gamma) &= [\![\varphi]\!]_{\mathcal{V}}(nw,\gamma) \diamond [\![\psi]\!]_{\mathcal{V}}(nw,\gamma) \\
[\![\bigoplus_x \varphi]\!]_{\mathcal{V}}(nw,\gamma) &= \sum_{i \in \mathbb{N}_+} ([\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(nw,\gamma[x \to i])) \\
[\![\bigoplus_X \varphi]\!]_{\mathcal{V}}(nw,\gamma) &= \sum_{I \subseteq \mathbb{N}_+} ([\![\varphi]\!]_{\mathcal{V}\cup\{X\}}(nw,\gamma[X \to I])) \\
[\![\mathrm{Val}_x \varphi]\!]_{\mathcal{V}}(nw,\gamma) &= \mathrm{Val}^{\omega}(([\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(nw,\gamma[x \to i]))_{i \in \mathbb{N}_+})
\end{aligned}
$$

Figure 4.3: Semantics of MSO($\mathbb{PD}$) for nested ω-words.

Definition 4.12 (Syntax)**.** The weighted monadic second order logic for nested ω-words MSO($\mathbb{PD}, \mathrm{NW}(\Sigma)$) is given by the following syntax

$$
\begin{aligned}
\beta &::= \mathrm{Lab}_a(x) \mid \mathrm{call}(x) \mid \mathrm{ret}(x) \mid x \le y \mid \nu(x,y) \mid x \in X \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta \mid \exists X.\beta \\
\varphi &::= d \mid \beta \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi \mid \bigoplus_x \varphi \mid \bigoplus_X \varphi \mid \mathrm{Val}_x \varphi
\end{aligned}
$$

where $d \in D$, $a \in \Sigma$, and x, y, X are first resp. second order variables. Then, as before, we call such φ *weighted formulas* and such β *boolean formulas.*

The set of all positions of $nw \in \mathrm{NW}^{\omega}(\Sigma)$ is $\mathbb{N}_+$. We define *valid* nested ω-words (nw, γ) over $\Sigma_{\mathcal{V}}$, with an *assignment* γ of variables $\mathcal{V}$ containing free(φ) as in Section 2.3. Note that an encoding of a nested ω-word uses the same matching relation ν.

Then, the *semantics* of φ (cf. Section 3.3) is a function $[\![\varphi]\!]_{\mathcal{V}} : \mathrm{NW}^{\omega}(\Sigma_{\mathcal{V}}) \to D$ defined for valid (nw, γ) inductively as in Figure 4.3. For all not valid nested ω-words, $[\![\varphi]\!]_{\mathcal{V}}$ yields 0. For a sentence φ, we have $[\![\varphi]\!] : \mathrm{NW}^{\omega}(\Sigma) \to D$.

Clearly, every boolean formula $\beta \in \mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ can be interpreted as an unweighted MSO-formula $\psi \in \mathrm{MSO}(\mathrm{NW}(\Sigma))$ with $[\![\beta]\!] = \mathbb{1}_{L(\psi)}$ since $[\![\beta]\!]$ only yields the values 0 and 1.

Example 4.3. We continue Example 4.2, where we gave an automaton $\mathcal{A}_2$ assigning to every nested ω-word nw the greatest accumulation point of the ratio of bracket-free positions in finite prefixes of nw. We extend the valuation monoid $\mathbb{D}$ of Example 4.2 to the pv-monoid $\mathbb{PD}' = (\bar{\mathbb{R}}, \sup, \lim\,\mathrm{avg}, +, -\infty, 0)$

and define the following boolean formulas

$$\begin{aligned} \mathrm{pcall}(x) &= \mathrm{call}(x) \wedge \forall w.\neg\nu(x,w), \\ \mathrm{pret}(x) &= \mathrm{ret}(x) \wedge \forall u.\neg\nu(u,x), \\ x < y < z &= \neg(y \leq x) \wedge \neg(z \leq y), \\ \mathrm{toplevel}(y) &= \forall x \forall z.\neg(x < y < z \wedge \nu(x,z)) \ . \end{aligned}$$

Then we can characterize all bracket-free positions with the boolean formula

$$\mathrm{bfree}(y) = \mathrm{toplevel}(y) \wedge \forall x(\neg(x < y \wedge \mathrm{pcall}(x)) \wedge \neg(y < x \wedge \mathrm{pret}(x))) \ .$$

Finally, we define (using $0, 1 \in \bar{\mathbb{R}}$)

$$\varphi = \mathrm{Val}_y((\mathrm{bfree}(y) \otimes 1) \oplus 0) \ .$$

We can conclude that for all nested ω-words nw, $[\![\varphi]\!](nw)$ describes the greatest accumulation point of the ratio of bracket-free positions in finite prefixes of nw, thus $[\![\mathcal{A}_2]\!] = [\![\varphi]\!]$. Note that the automaton yields runs over the weights 0 and 1, but the corresponding neutral elements of $\mathbb{PD}'$ are $-\infty$ and 0 making the conversion using the real values 0 and 1 necessary. $\diamond$

Given $\varphi \in \mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ and a valid $(nw, \gamma) \in \mathrm{NW}^\omega(\Sigma_\mathcal{V})$, we refer with $(nw, \gamma{\restriction_{\mathrm{free}(\varphi)}}) \in \mathrm{NW}^\omega(\Sigma_{\mathrm{free}(\varphi)})$ to the correspondent nested ω-word where the assignment function is restricted to $\mathrm{free}(\varphi)$. Analogously to [DG07] and [DP14b], we can show by Lemma 3.5 that $[\![\varphi]\!]_\mathcal{V}(nw, \gamma) = [\![\varphi]\!](nw, \gamma{\restriction_{\mathrm{free}(\varphi)}})$ for each valid $(nw, \gamma) \in \mathrm{NW}^\omega(\Sigma_\mathcal{V})$. Furthermore, by a standard induction on the structure of φ analogously to Proposition 3.3 of [DG07], we can show the following.

Lemma 4.13. *Let $\varphi \in \mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ and let $\mathcal{V}$ be a finite set of variables with $\mathrm{free}(\varphi) \subseteq \mathcal{V}$. Then $[\![\varphi]\!]$ is regular iff $[\![\varphi]\!]_\mathcal{V}$ is regular.*

In order to obtain a Büchi-like theorem for weighted automata, it is necessary to restrict the weighted MSO logic (cf. [DG07]). Therefore, in the following, we introduce and study suitable fragments of $\mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ following our approach of Section 3.4. Since sMNWA can be determinized, in the following, we do not have to restrict the unweighted fragment to first order formulas as done in Section 3.4, but can take the full boolean fragment into account.

Definition 4.14. We call a formula φ *almost boolean* if it is built up inductively from the following grammar

$$\varphi ::= d \mid \beta \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi$$

where $d \in D$ and β is a boolean formula.

Proposition 4.15. *(a) If $\varphi \in \mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ is an almost boolean formula, then $[\![\varphi]\!]$ is a regular step function.*

(b) For every regular step function $S : \mathrm{NW}^\omega(\Sigma) \to D$, there exists an almost boolean sentence φ with $S = [\![\varphi]\!]$.

Proof. (a) We use induction on the structure of an almost boolean formula. If $\varphi = d \in D$, then $[\![\varphi]\!] = d = d \diamond \mathbb{1}_{\mathrm{NW}^\omega(\Sigma)}$ is a regular step function. If φ is boolean, we can interpret φ as unweighted formula with $[\![\varphi]\!] = \mathbb{1}_{L(\varphi)}$. Then $L(\varphi)$ is a regular language by Theorem 4.3. Hence, $[\![\varphi]\!]$ is a regular step function.

For the induction step, we assume that $[\![\varphi]\!]$ and $[\![\psi]\!]$ are regular step functions. Then $[\![\varphi \oplus \psi]\!]$ and $[\![\varphi \otimes \psi]\!]$ are regular step functions by Proposition 4.10.

(b) Let $S : \mathrm{NW}^\omega(\Sigma) \to D$ be a regular step function, so $S = \sum_{i=1}^k d_i \diamond \mathbb{1}_{L_i}$ with $d_i \in D$ and for each $i \in \{1, ..., k\}$, L_i is a regular language of nested ω-words and (L_i) is a partition of $\mathrm{NW}^\omega(\Sigma)$. By Theorem 4.3, for every language L_i, there exists an unweighted sentence ψ_i with $L(\psi_i) = L_i$. Then there exist weighted boolean sentences φ_i with $[\![\varphi_i]\!] = \mathbb{1}_{L(\psi_i)}$.

Then, $\varphi = \bigoplus_{i=1}^k (d_i \otimes \varphi_i)$ is our required sentence because the following holds:

$$\begin{aligned}
[\![\varphi]\!](nw) &= [\![\bigoplus_{i=1}^k (d_i \otimes \varphi_i)]\!](nw) \\
&= \sum_{i=1}^k ([\![d_i]\!](nw) \diamond [\![\varphi_i]\!](nw)) \\
&= \sum_{i=1}^k (d_i \diamond \mathbb{1}_{L_i}(nw)) = S(nw) \ . \qquad \square
\end{aligned}$$

As in Section 3.4, but using boolean formulas instead of FO-boolean formulas, we get the following fragments.

Definition 4.16. Let $\varphi \in \mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$. We denote by $\mathrm{const}(\varphi)$ the set of all elements of $\mathbb{PD}$ occurring in φ. We call φ

1. *strongly-$\otimes$-restricted* if for all subformulas $\psi \otimes \theta$ of φ:

   ψ and θ are almost boolean or ψ is boolean or θ is boolean.

2. *$\otimes$-restricted* if for all subformulas $\psi \otimes \theta$ of φ:

   ψ is almost boolean or θ is boolean.

3. *commutatively-$\otimes$-restricted* if for all subformulas $\psi \otimes \theta$ of φ:

   ψ is almost boolean or $\mathrm{const}(\psi)$ and $\mathrm{const}(\theta)$ commute.

4. Val-*restricted* if for all subformulas $\mathrm{Val}_x \psi$ of φ, ψ is almost boolean.

We call a formula of $\mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ *syntactically restricted* if it is both Val-restricted and strongly-$\otimes$-restricted. Note that every subformula of a syntactically restricted formula is syntactically restricted itself.

Now, similarly to Theorem 3.6, we get the following result, which shows that under suitable assumptions on the ω-pv-monoid $\mathbb{PD}$, particular classes of $\mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$-formulas have the same expressive power. Note that in the proof of Theorem 3.6 the restriction to FO-boolean instead of boolean formulas is not critical. A full proof of this theorem for nested words can be found in [DD17].

Theorem 4.17. *(a) Let $\mathbb{PD}$ be left-distributive and $\varphi \in \mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ be $\otimes$-restricted. Then there exists a strongly-$\otimes$-restricted formula $\varphi' \in \mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ with $[\![\varphi]\!] = [\![\varphi']\!]$. Moreover, if φ is additionally* Val*-restricted, then φ' can be chosen to be* Val*-restricted.*

(b) Let $\mathbb{PD}$ be a cc-ω-valuation semiring and let $\varphi \in \mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ be commutatively-$\otimes$-restricted. Then there exists a strongly-$\otimes$-restricted formula $\varphi' \in \mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ with $[\![\varphi]\!] = [\![\varphi']\!]$. Moreover, if φ is additionally Val*-restricted, then φ' can be chosen to be* Val*-restricted.*

Similarly to Corollary 3.7, we get the following corollary.

Corollary 4.18. *Let $\mathbb{PD}$ be a commutative cc-ω-valuation semiring and $\varphi \in \mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$. Then, there exists a formula $\varphi' \in \mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ with $[\![\varphi']\!] = [\![\varphi]\!]$ such that for all subformulas of the form $\psi \otimes \theta$, each of ψ and θ is boolean or a constant.*

Note that this also follows from a slightly modified proof of Theorem 4.23, but the Transformation Theorem gives direct and efficient conversions of the formulas.

4.5 Characterization of Recognizable Series

In this section, we give our main result of this chapter on the expressive equivalence of weighted stair Muller nested word automata and our different fragments of weighted MSO logic (Theorem 4.23). We begin with the required closure properties.

Lemma 4.19. *Let φ and ψ be two formulas of $\mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ such that $[\![\varphi]\!]$ and $[\![\psi]\!]$ are regular. Then $[\![\varphi \oplus \psi]\!]$ is regular.*

Proof. We put $\mathcal{V} = \mathrm{free}(\varphi) \cup \mathrm{free}(\psi)$. Then $[\![\varphi \oplus \psi]\!] = [\![\varphi]\!]_{\mathcal{V}} + [\![\psi]\!]_{\mathcal{V}}$ is regular by Proposition 4.8 and Lemma 4.13. □

Lemma 4.20. *Let $\mathbb{PD}$ be regular. Let $\varphi \otimes \psi$ be a subformula of a syntactically restricted formula θ of $\mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ such that $[\![\varphi]\!]$ and $[\![\psi]\!]$ are regular. Then $[\![\varphi \otimes \psi]\!]$ is regular.*

Proof. Since θ is strongly-$\otimes$-restricted, both φ and ψ are almost boolean or one of both formulas is boolean.

Case 1: Let us assume φ and ψ are almost boolean. Then the formula $\varphi \otimes \psi$ is also almost boolean. Hence, $[\![\varphi \otimes \psi]\!]$ is regular by Proposition 4.15 (a) and Proposition 4.10.

Case 2: Let φ be boolean. We have to construct a wsMNWA recognizing $\varphi \otimes \psi$. Since φ is boolean, we can interpret φ as an unweighted MSO-formula and by Theorem 4.3 there exists a deterministic sMNWA $\mathcal{A} = (Q, q_0, \delta, \mathfrak{F})$ with $L(\mathcal{A}) = L(\varphi)$. For this sMNWA $\mathcal{A}$, we construct the wsMNWA $\mathcal{A}^{\mathbb{1}}$ as specified in Definition 4.9, and we obtain

$$[\![\mathcal{A}^{\mathbb{1}}]\!] = \mathbb{1}_{L(\mathcal{A})} = \mathbb{1}_{L(\varphi)} = [\![\varphi]\!] \ .$$

Let $\mathcal{B}$ be a wsMNWA recognizing $[\![\psi]\!]$. Note that $\mathcal{A}^{\mathbb{1}}$ emerges from a deterministic sMNWA and therefore itself has not more than one run for every nested ω-word. Hence, analogously to the proof of Proposition 4.10, we construct the product automaton of $\mathcal{A}^{\mathbb{1}}$ and $\mathcal{B}$ recognizing $[\![\varphi \otimes \psi]\!]$.

Case 3: Let ψ be boolean. We can proceed analogously to Case 2. Alternatively, we could observe that $[\![\varphi \otimes \psi]\!] = [\![\psi \otimes \varphi]\!]$ since ψ is boolean and then apply Case 2. □

Lemma 4.21. *Let φ be a formula of* $\mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$ *such that $[\![\varphi]\!]$ is regular. Then $[\![\bigoplus_x \varphi]\!]$ and $[\![\bigoplus_X \varphi]\!]$ are regular.*

Proof. We follow [DP14b] and [DG07]. Let $\mathcal{X} \in \{x, X\}$ and $\mathcal{V} = \mathrm{free}(\bigoplus_{\mathcal{X}} \varphi)$. We define $\pi : \mathrm{NW}^\omega(\Sigma_{\mathcal{V} \cup \{\mathcal{X}\}}) \to \mathrm{NW}^\omega(\Sigma_{\mathcal{V}})$ by $\pi(nw, \gamma) = (nw, \gamma|_{\mathcal{V}})$ for any $(nw, \gamma) \in \mathrm{NW}^\omega(\Sigma_{\mathcal{V} \cup \{\mathcal{X}\}})$. Then for $(nw, \gamma) \in \mathrm{NW}^\omega(\Sigma_{\mathcal{V}})$, the following holds

$$\begin{aligned}
[\![\bigoplus_X \varphi]\!](nw, \gamma) &= \sum_{I \subseteq \mathbb{N}_+} [\![\varphi]\!]_{\mathcal{V} \cup \{X\}}(nw, \gamma[X \to I]) \\
&= \sum_{(nw, \gamma') \in \pi^{-1}(nw, \gamma)} [\![\varphi]\!]_{\mathcal{V} \cup \{X\}}(nw, \gamma') \\
&= \pi([\![\varphi]\!]_{\mathcal{V} \cup \{X\}})(nw, \gamma) \ .
\end{aligned}$$

Analogously, we show that $[\![\bigoplus_x \varphi]\!](nw, \gamma) = \pi([\![\varphi]\!]_{\mathcal{V} \cup \{x\}})(nw, \gamma)$ holds for all $(nw, \gamma) \in \mathrm{NW}^\omega(\Sigma_{\mathcal{V}})$. By Lemma 4.13, $[\![\varphi]\!]_{\mathcal{V} \cup \{\mathcal{X}\}}$ is regular because $\mathrm{free}(\varphi) \subseteq \mathcal{V} \cup \{\mathcal{X}\}$. Then $[\![\bigoplus_{\mathcal{X}} \varphi]\!]$ is regular by Proposition 4.11. □

Lemma 4.22 (Closure under restricted Val_x)**.** *Let φ be an almost boolean formula of* $\mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$*. Then $[\![\mathrm{Val}_x \varphi]\!]$ is regular.*

Proof. We use ideas of Droste and Gastin [DG07] and the extensions in [DM12] and [DP14b]. We define $\mathcal{V} = \mathrm{free}(\mathrm{Val}_x \varphi)$ and $\mathcal{W} = \mathrm{free}(\varphi) \cup \{x\}$. By Proposition 4.15, $[\![\varphi]\!]$ is a regular step function. Write $[\![\varphi]\!] = \sum_{j=1}^{k} d_j \diamond \mathbb{1}_{L_j}$ where

L_j is a regular language of nested ω-words for all $j \in \{1, ..., k\}$ and (L_j) is a partition of $\mathrm{NW}^\omega(\Sigma_\mathcal{W})$. By the semantics of the universal quantifier, we get

$$\begin{aligned}
[\![\mathrm{Val}_x\, \varphi]\!](nw, \gamma) &= \mathrm{Val}^\omega(([\![\varphi]\!]_\mathcal{W}(nw, \gamma[x \to i]))_{i \in \mathbb{N}_+}) \\
&= \mathrm{Val}^\omega((d_{g(i)})_{i \in \mathbb{N}_+}), \\
\text{where} \qquad g(i) &= \begin{cases} 1 & , \text{ if } (nw, \gamma[x \to i]) \in L_1 \\ \dots & \\ k & , \text{ if } (nw, \gamma[x \to i]) \in L_k \end{cases} , \text{ for all } i \in \mathbb{N}_+ \ . \qquad (4.3)
\end{aligned}$$

We use this property (4.3) and divide the proof into two parts: First, we encode the information to which language $(nw, \gamma[x \to i])$ belongs in a specially extended language $\tilde{L}$ and construct an MSO-formula for this language. Then by Theorem 4.3, we get an sMNWA recognizing $\tilde{L}$. In the second part, we add the weights d_i to this automaton and return to our original alphabet.

Part 1: We define the extended alphabet $\tilde{\Sigma} = \Sigma \times \{1, ..., n\}$, so:

$$\mathrm{NW}^\omega(\tilde{\Sigma}_\mathcal{V}) = \{(nw, g, \gamma) \mid (nw, \gamma) \in \mathrm{NW}^\omega(\Sigma_\mathcal{V}) \text{ and } g \in \{1, ..., n\}^{\mathbb{N}_+}\} \ .$$

We define the languages $\tilde{L}, \tilde{L}_j, \tilde{L}'_j \subseteq \mathrm{NW}^\omega(\tilde{\Sigma}_\mathcal{V})$ as follows:

$$\begin{aligned}
\tilde{L} &= \left\{ (nw, g, \gamma) \;\middle|\; \begin{array}{l} (nw, \gamma) \in \mathrm{NW}^\omega(\tilde{\Sigma}_\mathcal{V}) \text{ is valid and} \\ \text{for all } i \in \mathbb{N}_+,\ j \in \{1, ..., k\} : \\ \quad g(i) = j \Rightarrow (nw, \gamma[x \to i]) \in L_j \end{array} \right\} \ , \\
\tilde{L}_j &= \left\{ (nw, g, \gamma) \;\middle|\; \begin{array}{l} (nw, \gamma) \in \mathrm{NW}^\omega(\tilde{\Sigma}_\mathcal{V}) \text{ is valid and} \\ \text{for all } i \in \mathbb{N}_+ :\ g(i) = j \Rightarrow (nw, \gamma[x \to i]) \in L_j \end{array} \right\} \ , \\
\tilde{L}'_j &= \{\, (nw, g, \gamma) \mid \text{for all } i \in \mathbb{N}_+ :\ g(i) = j \Rightarrow (nw, \gamma[x \to i]) \in L_j \,\} \ .
\end{aligned}$$

Then $\tilde{L} = \bigcap_{j=1}^k \tilde{L}_j$. Hence, in order to show that $\tilde{L}$ is regular, it suffices to show that each $\tilde{L}_j$ is regular. By a standard procedure, compare [DG07], we obtain a formula $\tilde{\varphi}_j \in \mathrm{MSO}(\mathrm{NW}(\tilde{\Sigma}_\mathcal{V}))$ with $L(\tilde{\varphi}_j) = \tilde{L}'_j$. Therefore, by Theorem 4.3, $\tilde{L}'_j$ is regular. Since regular nested ω-word languages are closed under intersection, this language intersected with all valid (nw, g, γ), $\tilde{L}_j = \tilde{L}'_j \cap N_\mathcal{V}$, is regular too. So, $\tilde{L}$ is regular. Hence, there exists a deterministic sMNWA recognizing $\tilde{L}$.

Part 2: Let $\tilde{\mathcal{A}} = (Q, I, \tilde{\delta}, \mathfrak{F})$ be an sMNWA recognizing $\tilde{L}$. Now, we construct the wsMNWA $\mathcal{A} = (Q, I, \delta, \mathfrak{F})$ over $\Sigma_\mathcal{V}$ by adding to every transition of $\tilde{\mathcal{A}}$ with $g(i) = j$ the weight d_j.

That is, we keep the states, initial states, and sets of accepting states, and

for $\delta = (\delta_{\text{call}}, \delta_{\text{int}}, \delta_{\text{ret}})$ and all $q, q', p \in Q$ and $(a, j, s) \in \tilde{\Sigma}_{\mathcal{V}}$, we define

$$\delta_{\text{call}}(q, (a, s), q') = \begin{cases} d_j & , \text{ if } (q, (a, j, s), q') \in \tilde{\delta}_{\text{call}} \\ 0 & , \text{ otherwise } , \end{cases}$$

$$\delta_{\text{int}}(q, (a, s), q') = \begin{cases} d_j & , \text{ if } (q, (a, j, s), q') \in \tilde{\delta}_{\text{int}} \\ 0 & , \text{ otherwise } , \end{cases}$$

$$\delta_{\text{ret}}(q, p, (a, s), q') = \begin{cases} d_j & , \text{ if } (q, p, (a, j, s), q') \in \tilde{\delta}_{\text{ret}} \\ 0 & , \text{ otherwise } . \end{cases}$$

Since $\tilde{\mathcal{A}}$ is deterministic, for every $(nw, g, \gamma) \in \tilde{L}$, there exists exactly one accepted run $\tilde{r}$ of $\tilde{\mathcal{A}}$. On the other hand, for every $(nw, g, \gamma) \notin \tilde{L}$, there is no accepted run of $\tilde{\mathcal{A}}$. Since (L_j) is a partition of $\mathrm{NW}^\omega(\Sigma_{\mathcal{W}})$, for every $(nw, \gamma) \in \mathrm{NW}^\omega(\Sigma_{\mathcal{V}})$, there exists exactly one g with $(nw, g, \gamma) \in \tilde{L}$. Thus, every $(nw, \gamma) \in \mathrm{NW}^\omega(\Sigma_{\mathcal{V}})$ has exactly one run r of $\mathcal{A}$ determined by the run $\tilde{r}$ of (nw, g, γ) of $\tilde{\mathcal{A}}$. By definition of $\mathcal{A}$ and $\tilde{L}$, the following holds for all $i \in \mathbb{N}_+$

$$g(i) = j \Rightarrow \mathrm{wt}_{\mathcal{A}}(r, (nw, \gamma), i) = d_j \wedge (nw, \gamma[x \to i]) \in L_j \ .$$

By Formula (4.3), we obtain

$$[\![\varphi]\!]_{\mathcal{W}}(nw, \gamma[x \to i]) = d_j = \mathrm{wt}_{\mathcal{A}}(r, (nw, \gamma), i) \ .$$

Hence, for the behavior of the automaton $\mathcal{A}$ the following holds

$$\begin{aligned} [\![\mathcal{A}]\!](nw, \gamma) &= \sum_{r' \in \mathrm{acc}(\mathcal{A})} \mathrm{wt}_{\mathcal{A}}(r', (nw, \gamma)) \\ &= \mathrm{Val}^\omega((\mathrm{wt}_{\mathcal{A}}(r, (nw, \gamma), i))_{i \in \mathbb{N}_+}) \\ &= \mathrm{Val}^\omega(([\![\varphi]\!]_{\mathcal{W}}(nw, \gamma[x \to i]))_{i \in \mathbb{N}_+}) \\ &= [\![\mathrm{Val}_x \varphi]\!](nw, \gamma) \ . \end{aligned}$$

Thus, $\mathcal{A}$ recognizes $[\![\mathrm{Val}_x \varphi]\!]$. □

This provides us with the means to prove the following main theorem of this chapter.

Theorem 4.23. *Let $\mathbb{PD}$ be a regular ω-pv-monoid and $S : \mathrm{NW}^\omega(\Sigma) \to D$ a series.*

1. *The following are equivalent:*
   - *(i) S is regular.*
   - *(ii) $S = [\![\varphi]\!]$ for a syntactically restricted sentence φ of $\mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$.*
2. *Let $\mathbb{PD}$ be left-distributive. Then the following are equivalent:*
   - *(i) S is regular.*

(ii) $S = [\![\varphi]\!]$ *for a* Val*-restricted and* $\otimes$*-restricted sentence* φ *of* $\mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$.

3. *Let* $\mathbb{PD}$ *be cc-*ω*-valuation semiring. Then the following are equivalent:*

   (i) S *is regular.*

   (ii) $S = [\![\varphi]\!]$ *for a* Val*-restricted and commutatively-*$\otimes$*-restricted sentence* φ *of* $\mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$.

Proof. We prove the direction '$(ii) \Rightarrow (i)$' for all three statements as follows. By Theorem 4.17, we may assume that φ is syntactically restricted. By induction on the structure of φ, we show that $[\![\varphi]\!]$ is regular. If φ is almost boolean, regularity of $[\![\varphi]\!]$ follows from Propositions 4.15 (a) and 4.10. To deal with $\oplus$, $\otimes$, $\bigoplus$, and Val_x, we apply Lemmata 4.19, 4.20, 4.21, and 4.22, respectively.

The converse '$(i) \Rightarrow (ii)$' for all three statements is proven by the following construction.

Let $\mathcal{A} = (Q, I, \delta, \mathfrak{F})$ be the wsMNWA over $\mathbb{PD}$ with $[\![\mathcal{A}]\!] = S$. We construct a Val-restricted and strongly-$\otimes$-restricted $\mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$-sentence φ with $[\![\varphi]\!] = [\![\mathcal{A}]\!]$. We follow the ideas of [DG07, DP14b, DR06] and simulate the behavior of the automaton on an arbitrary nested ω-word nw step by step with a formula to obtain $[\![\varphi]\!](nw) = [\![\mathcal{A}]\!](nw)$.

In the formula, we use the following second order variables $X^{\mathrm{call}}_{p,a,q}$, $X^{\mathrm{int}}_{p,a,q}$, and $X^{\mathrm{ret}}_{p,r,a,q}$, where $a \in \Sigma$ and $p, q, r \in Q$. Let

$$\mathcal{V} = \{X^{\mathrm{call}}_{p,a,q},\ X^{\mathrm{int}}_{p,a,q},\ X^{\mathrm{ret}}_{p,r,a,q} \mid a \in \Sigma,\ p, q, r \in Q\} \ .$$

Since Σ and Q are finite, there is an enumeration $\bar{X} = (X_1, .., X_m)$ of all variables of $\mathcal{V}$. We use the following classical abbreviations for unweighted MSO-formulas:

$$\begin{aligned}
(\beta \wedge \varphi) &= \neg(\neg\beta \vee \neg\varphi) \ , \\
(\beta \to \varphi) &= (\neg\beta \vee \varphi) \ , \\
(\forall x.\varphi) &= \neg(\exists x.\neg\varphi) \ , \\
(y = x) &= (x \leq y) \wedge (y \leq x) \ , \\
(y = x + 1) &= (x \leq y) \wedge \neg(y \leq x) \wedge \forall z.(z \leq x \vee y \leq z) \ .
\end{aligned}$$

Also, we use the following shorthands for unweighted formulas of $\mathrm{MSO}(\mathrm{NW}(\Sigma))$:

$$\begin{aligned}
\min(x) &= \forall y.(x \leq y) \ , \\
\mathrm{int}(x) &= \neg\mathrm{call}(x) \wedge \neg\mathrm{ret}(x) \ , \\
\mathrm{pcall}(x) &= \mathrm{call}(x) \wedge \neg\exists y.\nu(x, y) \ , \\
\mathrm{pret}(x) &= \mathrm{ret}(x) \wedge \neg\exists y.\nu(y, x) \ .
\end{aligned}$$

Furthermore, we use the following almost boolean formula of $\mathrm{MSO}(\mathbb{PD}, \mathrm{NW}(\Sigma))$

$$((x \in X) \to d) = (x \notin X) \oplus ((x \in X) \otimes d) \ ,$$

with the semantics

$$\begin{aligned}
[\![(x \in X) \to d]\!](nw, \gamma) &= [\![x \notin X \oplus ((x \in X) \otimes d)]\!](nw, \gamma) \\
&= [\![x \notin X]\!](nw, \gamma) + ([\![x \in X]\!](nw, \gamma) \diamond d) \\
&= \begin{cases} d & , \text{ if } \gamma(x) \in \gamma(X) \\ 1 & , \text{ otherwise } . \end{cases}
\end{aligned} \tag{4.4}$$

We define the unweighted formula ψ to characterize all accepted runs

$$\psi = \mathit{Partition}(\bar{X}) \wedge \mathit{Label} \wedge \mathit{Init} \wedge \mathit{Trans}_1 \wedge \mathit{Trans}_2 \wedge \mathit{Final} \ .$$

Here, the subformula *Partition* will enforce the sets X_1 to X_m to be a partition of all positions. *Label* controls the symbols of the run, *Init* the initial condition, Trans_1 and Trans_2 the transitions, and *Final* the acceptance conditions of the run. These formulas are similar to those of [DP14b] and included for the convenience of the reader:

$$\mathit{Partition}((X_1, ..., X_m)) = \forall x. \bigvee_{i=1}^{m} \big[(x \in X_i) \wedge \bigwedge_{i \neq j} \neg(x \in X_j)\big] \ ,$$

$$\begin{aligned}
\mathit{Label} = \bigwedge_{p,q,r \in Q,\, a \in \Sigma} \forall x. \big[& (x \in X^{\mathrm{call}}_{p,a,q} \to (\mathrm{Lab}_a(x) \wedge \mathrm{call}(x))) \\
& \wedge (x \in X^{\mathrm{int}}_{p,a,q} \to (\mathrm{Lab}_a(x) \wedge \mathrm{int}(x))) \\
& \wedge (x \in X^{\mathrm{ret}}_{p,r,a,q} \to (\mathrm{Lab}_a(x) \wedge \mathrm{ret}(x)))\big] \ ,
\end{aligned}$$

$$\mathit{Init} = \exists y. \big[\min(y) \wedge \bigvee_{\substack{p \in I,\, q \in Q \\ a \in \Sigma}} (y \in X^{\mathrm{call}}_{p,a,q} \vee y \in X^{\mathrm{int}}_{p,a,q} \vee y \in X^{\mathrm{ret}}_{p,p,a,q})\big] \ ,$$

$$\begin{aligned}
\mathit{Trans}_1 = \forall x \forall y. \big[& y = x + 1 \to \bigvee_{\substack{p,q,r,s,t \in Q \\ a,b \in \Sigma}} (x \in X^{\mathrm{call}}_{p,a,q} \vee x \in X^{\mathrm{int}}_{p,a,q} \vee x \in X^{\mathrm{ret}}_{p,r,a,q}) \\
& \wedge (y \in X^{\mathrm{call}}_{q,b,t} \vee y \in X^{\mathrm{int}}_{q,b,t} \vee y \in X^{\mathrm{ret}}_{q,s,b,t})\big] \ ,
\end{aligned}$$

$$\begin{aligned}
\mathit{Trans}_2 = \forall x \forall y. \big[& (\nu(x,y) \to \bigvee_{\substack{p,q,r,s \in Q \\ a,b \in \Sigma}} (x \in X^{\mathrm{call}}_{p,a,q} \wedge y \in X^{\mathrm{ret}}_{r,p,b,s})) \\
& \wedge (\mathrm{pcall}(x) \to \bigvee_{\substack{p,q \in Q \\ a \in \Sigma}} (x \in X^{\mathrm{call}}_{p,a,q})) \\
& \wedge \big((\mathrm{pret}(x) \wedge \min(y)) \to \bigvee_{\substack{p,q,r,s,t \in Q \\ a,b \in \Sigma}} (x \in X^{\mathrm{ret}}_{p,r,a,q} \\
& \qquad \wedge (y \in X^{\mathrm{call}}_{r,b,t} \vee y \in X^{\mathrm{int}}_{r,b,t} \vee y \in X^{\mathrm{ret}}_{r,s,b,t}))\big)\big] \ .
\end{aligned}$$

A difference is found in *Final*, where we have to check the stair Muller acceptance. For this, we need to characterize top-level positions. We define

$$\text{toplevel}(y) = \forall u \forall w [\nu(u,w) \to (w \leq y \vee y \leq u)] \ .$$

Using toplevel, we characterize whether a state is accepting by the formula

$$q \in Q^t_\infty = \forall x \exists y. \big((y > x) \wedge \bigvee_{p,r \in Q, a \in \Sigma} (y \in X^{\text{call}}_{p,a,q} \vee y \in X^{\text{int}}_{p,a,q} \vee y \in X^{\text{ret}}_{p,r,a,q}) \\ \wedge \text{toplevel}(y)\big) \ .$$

This formula is satisfied if for every position, there exists a greater top-level position at which the state q is visited. Then we define

$$\mathit{Final} = \bigvee_{F \in \mathcal{F}} \big(\bigwedge_{q \in F} (q \in Q^t_\infty) \wedge \bigwedge_{q \notin F} \neg (q \in Q^t_\infty)\big) \ .$$

This formula describes the stair Muller acceptance condition. For a run $\rho = (q_0, q_1, ...)$ of $\mathcal{A}$ on nw, we define the $(\mathcal{V}, nw)$-assignment γ_ρ by

$$\begin{aligned} \gamma_\rho(X^{\text{call}}_{p,a,q}) &= \{i \mid i \text{ is a call and } (q_{i-1}, a_i, q_i) = (p,a,q)\} \ , \\ \gamma_\rho(X^{\text{int}}_{p,a,q}) &= \{i \mid i \text{ is an internal and } (q_{i-1}, a_i, q_i) = (p,a,q)\} \ , \\ \gamma_\rho(X^{\text{ret}}_{p,r,a,q}) &= \{i \mid \nu(j,i) \text{ for some } 1 \leq j < i \text{ and } (q_{i-1}, q_{j-1}, a_i, q_i) = (p,r,a,q) \\ &\qquad \text{or } \nu(-\infty, i) \text{ and } (q_{i-1}, q_0, a_i, q_i) = (p,r,a,q)\} \ . \end{aligned} \tag{4.5}$$

Analogously to [DP14b] and [DR06], this implies that (nw, γ_ρ) satisfies ψ iff ρ is an accepted run of $\mathcal{A}$. Furthermore, for every $(nw, \gamma) \in \text{NW}^\omega(\Sigma)$ satisfying ψ, there exists an accepted run ρ of $\mathcal{A}$ with $\gamma = \gamma_\rho$.

We interpret the unweighted formula ψ as a boolean weighted formula. Hence, $[\![\psi]\!](nw, \gamma) = 1$ iff γ has an accepted run of $\mathcal{A}$ on nw.

Now, we add weights. We define

$$\theta = \psi \otimes \text{Val}_x \bigotimes_{p,q \in Q,\, a \in \Sigma} \big((x \in X^{\text{call}}_{p,a,q} \to \delta_{\text{call}}(p,a,q)) \\ \otimes (x \in X^{\text{int}}_{p,a,q} \to \delta_{\text{int}}(p,a,q)) \\ \otimes \bigotimes_{r \in Q} (x \in X^{\text{ret}}_{p,r,a,q} \to \delta_{\text{ret}}(p,r,a,q))\big) \ .$$

Since the subformulas $x \in X^{()}_{()} \to \delta()$ of θ are almost boolean, the subformula $\text{Val}_x .(...)$ of θ is Val-restricted. Furthermore, ψ is boolean and so θ is strongly-$\otimes$-restricted. Thus, θ is a syntactically restricted formula.

Let (nw, γ_ρ) be as mentioned before. Since $\bar{X}$ is a partition, every position i belongs to exactly one set $\gamma_\rho(X_k)$. Then, by formula (4.4), we obtain for the

weight δ_i of the corresponding transition

$$\begin{aligned}
&\Big[\!\Big[\bigotimes_{p,q\in Q, a\in\Sigma} \big((x \in X^{\mathrm{call}}_{p,a,q} \to \delta_{\mathrm{call}}(p,a,q)) \\
&\qquad\qquad \otimes (x \in X^{\mathrm{int}}_{p,a,q} \to \delta_{\mathrm{int}}(p,a,q)) \\
&\qquad\qquad \otimes \bigotimes_{r\in Q} (x \in X^{\mathrm{ret}}_{p,r,a,q} \to \delta_{\mathrm{ret}}(p,r,a,q))\big)\Big]\!\Big](nw, \gamma_\rho[x \to i]) \\
&= 1 \diamond 1 \diamond \ldots \diamond 1 \diamond \delta_i \diamond 1 \diamond \ldots \diamond 1 \\
&= \delta_i \ .
\end{aligned}$$

By the definition of the weight functions for wsMNWA (4.1) and the assignment γ_ρ (4.5), we obtain that δ_i equals $\mathrm{wt}_{\mathcal{A}}(\rho, nw, i)$ and the following holds

$$\begin{aligned}
[\![\theta]\!](nw, \gamma_\rho) &= \begin{cases} 1 \diamond \mathrm{Val}^\omega((\delta_i)_{i\in\mathbb{N}_+}) & , \text{ if } \rho \text{ is an accepted run} \\ 0 & , \text{ otherwise} \end{cases} \\
&= \begin{cases} \mathrm{Val}^\omega((\mathrm{wt}_{\mathcal{A}}(\rho, nw, i))_{i\in\mathbb{N}_+}) & , \text{ if } \rho \text{ is an accepted run} \\ 0 & , \text{ otherwise } . \end{cases}
\end{aligned}$$

Finally, we define

$$\varphi = \bigoplus_{X_1} \bigoplus_{X_2} \cdots \bigoplus_{X_m} \theta \ .$$

This implies for all $nw \in \mathrm{NW}^\omega(\Sigma)$:

$$\begin{aligned}
[\![\varphi]\!](nw) &= [\![\textstyle\bigoplus_{X_1} \bigoplus_{X_2} \cdots \bigoplus_{X_m} \theta]\!](nw) \\
&= \sum_{\gamma\colon (\mathcal{V},nw)\text{-assignment}} [\![\theta]\!](nw, \gamma) \\
&= \sum_{\rho\colon \text{run of } \mathcal{A}} [\![\theta]\!](nw, \gamma_\rho) \\
&= \sum_{\rho\in\mathrm{acc}(\mathcal{A})} \mathrm{Val}^\omega((\mathrm{wt}_{\mathcal{A}}(r, nw, i))_{i\in\mathbb{N}_+}) \\
&= [\![\mathcal{A}]\!](nw) \ .
\end{aligned}$$

Therefore, φ is our required sentence with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$. □

Chapter 5

Weighted Operator Precedence Languages

In this chapter, we study operator precedence languages and weighted extensions of them. The language class of *operator precedence languages (OPL)*, introduced by Floyd as a proper subclass of context-free languages in his seminal work [Flo63], was inspired by the precedence of multiplicative operations over additive ones in the execution of arithmetic expressions. Floyd extended this notion to a relation on the whole input alphabet in such a way that it could drive a deterministic parsing algorithm that builds the syntax tree of any word that reflects the word's semantics.

Due to the advent of LR grammars, which unlike OPL grammars generate all deterministic CFL, the theoretical investigation of OPL has undergone a pause until quite recently, when it was discovered that they significantly generalize the well-known visibly pushdown languages (VPL). In the following renewed studies of OPL, they were shown to be closed with respect to all major operations like Boolean operations, concatenation, Kleene*, and renaming [CM12]. Furthermore, OPL are also characterizable, besides Floyd's original grammar family, in terms of an appropriate class of pushdown automata and in terms of a MSO logic which is a fairly natural but not trivial extension of the previous MSO logics defined to characterize regular languages and VPL [LMPP15].

Both language classes, OPL and VPL, can be intuitively called *input-driven*, i.e., when and where an automaton uses pushes and pops depends exclusively on the input alphabet and on the additional relation defined on the alphabet. However, one major difference of OPL in comparison to VPL is that the structure of a word of an OPL is not visible, e.g, OPL can include unparenthesized arithmetic expressions where the precedence of multiplicative operators over additive ones is explicit in the syntax trees but hidden in the word itself (see Figure 5.1 for an example). Furthermore, OPL include deterministic CFL that are not real-time [LMPP15], in particular, a run of an

OPA is potentially longer than the input word.

All in all, OPL enjoy many of the valuable properties of regular languages but considerably extend their applicability by breaking the barrier of visibility and real-time push-down recognition.

As remarked earlier, in another area of research, quantitative models of systems are also greatly in demand. The result of this is a rich collection of *weighted logics*, first studied by Droste and Gastin [DG07], associated with *weighted tree automata* [DV06] and *weighted VPAs* the automata recognizing VPLs, also called weighted NWAs [Mat10b].

In this chapter, fitting the other approaches of this thesis, we build on concepts from both research fields. We introduce OPL as language class accepted by *operator precedence automata (OPA)*. Afterwards, we extend these automata with semiring weights to define *weighted OPA* and their behaviors, *weighted OPL*. We show that weighted OPA are able to model system behaviors that cannot be specified by means of less powerful weighted formalisms such as weighted VPL. For instance, one might be interested in the behavior of a system which handles calls and returns but is subject to some emergency interrupts. Then it is important to evaluate how critically the occurrences of interrupts affect the normal system behavior, e.g., by counting the number of pending calls that have been preempted by an interrupt. As another example, we consider a system logging all hierarchical calls and returns over words where this structural information is hidden. Depending on changing exterior factors like energy level, such a system could decide to log the above information in a selective way.

Further contributions of this chapter are the following.

We prove that weighted OPL strictly include weighted VPL while enjoying the same closure properties as them. We show that for arbitrary semirings, there is a relevant difference in the expressive power of the model depending on whether it permits assigning weights to pop transitions or not. For commutative semirings, however, weights on pop transitions do not increase the expressive power of the automata. This difference in descriptive power between weighted OPA with arbitrary weights and without weights at pop transitions is due to the fact that OPL may be non-real-time and therefore OPA may execute several pop moves without advancing their reading heads.

We prove an extension of the classical result of Nivat [Niv68] to weighted OPL. This robustness result shows that the behaviors of weighted OPA without weights at pop transitions are exactly those that can be constructed from weighted OPA with only one state, intersected with OPL, and applying projections which preserve the structural information.

We adapt our weighted MSO logic to OPL and show that it is expressively equivalent to weighted OPA without weights at pop transitions. As a corollary, for commutative semirings this weighted logic is equivalent to weighted OPA including weights at pop transitions.

$$S \to S + T \mid T$$
$$T \to T \times A \mid A$$
$$A \to a \mid (S)$$

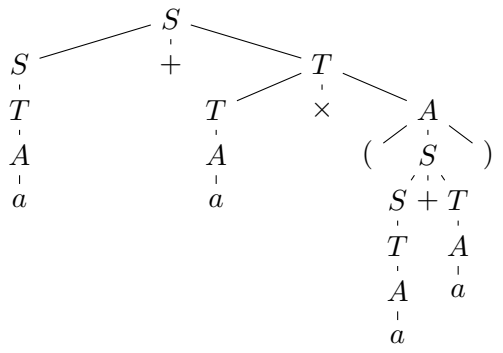


Figure 5.1: A grammar generating arithmetic expressions (left) and an example derivation tree (right) for $a + a \times (a + a)$

The results of this chapter are were published in [DDMP17].

5.1 Operator Precedence Languages

In the following, we consider in more detail the notion of a *precedence word* over (Σ, M) as introduced in Section 2.2. We are taking a look at the origins of this structure and show how they correspond to the class of *operator precedence languages*.

Operator precedence languages find their origin in Floyd's precedence grammars, which made the hidden precedences between symbols (e.g. multiplication takes precedence over addition) occurring in a grammar explicit in parse trees [Flo63]. From these precedences, we get an additional hierarchical structure, which has some additional features compared to the matching relation of nested words. In particular, we can model multiple layers of interrupts, and we can start and end multiple hierarchical levels at the same position. Also, the hierarchical structure is not immediately visible in the word.

For example, consider classical arithmetic expressions with two operators; addition and multiplication. We want to model that the multiplication takes precedence over the addition, in the sense that, during the interpretation of the expression, multiplications must be executed before sums. Additionally, parentheses are used to force different precedence hierarchies. Figure 5.1 presents a grammar and the derivation tree of the expression $a + a \times (a + a)$.

Notice that the precedence of multiplication over addition is not immediately visible in the expression. For this reason, we say that such grammars "hide" the structure associated with a sentence, whereas parenthesis grammars and other input-driven ones make the structure explicit in the sentences they generate.

In this context, it is possible to derive from the grammar a precedence relation for every pair of symbols, see e.g. [CM12] for details and Figure 5.2

	a	$+$	$\times$	$($	$)$
a		$\gtrdot$	$\gtrdot$		$\gtrdot$
$+$	$\lessdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\gtrdot$
$\times$	$\lessdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\gtrdot$
$($	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$
$)$		$\gtrdot$	$\gtrdot$		$\gtrdot$

Figure 5.2: A suitable precedence matrix M for the grammar of Figure 5.1

for an example of the precedences displayed as a matrix.

In contrast to this grammar based approach, in the following, we introduce and work with general *precedence matrices*. These matrices can restrict the set of all expressions and, more importantly, as previously, model precedences between alphabet symbols. For example, the matrix M of Figure 5.2 has no entry $M_{a,a}$, thus it forbids that we encounter 'aa' in a word. Additionally, the entry $+ \lessdot x$ gives the product precedence over the addition. In the same manner, every operation yields precedence to the opening parentheses and whatever follows after it.

It follows that every word over Σ has an additional hierarchical structure defined by the given precedence matrix M, as defined by the following definitions from Section 2.2 (page 18) which we recall here for convenience.

An *OP alphabet* as a pair (Σ, M), where Σ is an alphabet and M, the *operator precedence matrix (OPM)* is a $|\Sigma \cup \{\#\}|^2$ array, where every entry is either $\lessdot$ (*yields precedence*), $\doteq$ (*equal in precedence*, $\gtrdot$ (*takes precedence*), or empty (no relation).

We use the symbol # to mark the beginning and the end of a word and let always be $\# \lessdot a$ *and* $a \gtrdot \#$ *for all* $a \in \Sigma$.

Let $w = (a_1 ... a_n) \in \Sigma^+$ be a word and (Σ, M) an OP alphabet. We say $a_0 = a_{n+1} = \#$ and define the *chain relation* $\curvearrowright$ on the set of all positions of $\#w\#$, inductively, as follows. Let $i, j \in \{0, 1, ..., n+1\}$, $i < j$. Then, $i \curvearrowright j$ holds if and only if there exists a sequence of positions $k_1 ... k_m$, $m \geq 3$, such that $i = k_1 < ... < k_m = j$, $a_{k_1} \lessdot a_{k_2} \doteq ... \doteq a_{k_{m-1}} \gtrdot a_{k_m}$, and either $k_s + 1 = k_{s+1}$ or $k_s \curvearrowright k_{s+1}$ for each $s \in \{1, ..., m-1\}$.

We say w is *compatible* with M if for $\#w\#$ we have $0 \curvearrowright n+1$. In particular, this forces $M_{a_i a_j} \neq \emptyset$ for all $i+1 = j$ and for all $i \curvearrowright j$. We say w is a *precedence word* over M if w is not the empty word and w is compatible with M. We denote by $(\Sigma, M)^+$ the set of all precedence words over (Σ, M). For an OPM M without empty entries, we have $(\Sigma, M)^+ = \Sigma^+$, due to $\# \lessdot a$ and $a \gtrdot \#$ for all $a \in \Sigma$. For an example of a precedence word, see the previous Figure 2.6.

We recall the definition of operator precedence automata from [LMPP15].

Definition 5.1. A *(nondeterministic) operator precedence automaton (OPA)* $\mathcal{A}$ over an OP alphabet (Σ, M) is a quadruple $\mathcal{A} = (Q, I, F, \delta)$ with $\delta = (\delta_{\text{push}}, \delta_{\text{shift}}, \delta_{\text{pop}})$, consisting of

- Q, a finite set of states,
- $I \subseteq Q$, the set of initial states,
- $F \subseteq Q$, a set of final states, and
- the transition relations $\delta_{\text{shift}}, \delta_{\text{push}} \subseteq Q \times \Sigma \times Q$, and $\delta_{\text{pop}} \subseteq Q \times Q \times Q$.

Let $\Gamma = \Sigma \times Q$. A *configuration* of $\mathcal{A}$ is a triple $C = \langle \Pi, q, w\# \rangle$, where $\Pi \in \bot\Gamma^*$ represents a stack, $q \in Q$ the current state, and w the remaining input to read.

A *run* of $\mathcal{A}$ on $w = a_1 ... a_n$ is a finite sequence of configurations $C_0 \vdash ... \vdash C_m$ such that every transition $C_i \vdash C_{i+1}$ has one of the following forms, where a is the first component of the topmost symbol of the stack Π, or # if the stack is $\bot$, and b is the next symbol of the input to read:

$$\begin{array}{rrll} \textit{push move}: & \langle \Pi, q, bx \rangle & \vdash \langle \Pi[b, q], r, x \rangle & \text{if } a \lessdot b \text{ and } (q, b, r) \in \delta_{\text{push}}, \\ \textit{shift move}: & \langle \Pi[a, p], q, bx \rangle & \vdash \langle \Pi[b, p], r, x \rangle & \text{if } a \doteq b \text{ and } (q, b, r) \in \delta_{\text{shift}}, \\ \textit{pop move}: & \langle \Pi[a, p], q, bx \rangle & \vdash \langle \Pi, r, bx \rangle & \text{if } a \gtrdot b \text{ and } (q, p, r) \in \delta_{\text{pop}}. \end{array}$$

An *accepting run* of $\mathcal{A}$ on w is a run from $\langle \bot, q_I, w\# \rangle$ to $\langle \bot, q_F, \# \rangle$, where $q_I \in I$ and $q_F \in F$. The *language accepted by* $\mathcal{A}$, denoted $L(\mathcal{A})$, consists of all precedence words over $(\Sigma, M)^+$ which have an accepting run on $\mathcal{A}$. We say that $L \subseteq (\Sigma, M)^+$ is an *OPL* if L is accepted by an OPA over (Σ, M). As proven by [LMPP15], the deterministic variant of an OPA, using a single initial state instead of I and transition functions instead of relations, is equally expressive to nondeterministic OPA.

An example of an OPA is depicted in Figure 5.3. It uses the OPM of Figure 5.2 and accepts the same language as the grammar of Figure 5.1.

Definition 5.2 ([LMPP15])**.** The logic MSO(Σ, M), short MSO, is defined as

$$\beta ::= \text{Lab}_a(x) \mid x \leq y \mid x \curvearrowright y \mid x \in X \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta \mid \exists X.\beta$$

where $a \in \Sigma \cup \{\#\}$, x, y are first-order variables; and X is a second order variable.

The relation $\curvearrowright$ is defined by the chain relation introduced above. Then, the semantics for this (unweighted) logic are defined as in Section 2.3.

Theorem 5.3 ([LMPP15])**.** *A language L over (Σ, M) is an OPL iff it is MSO-definable.*

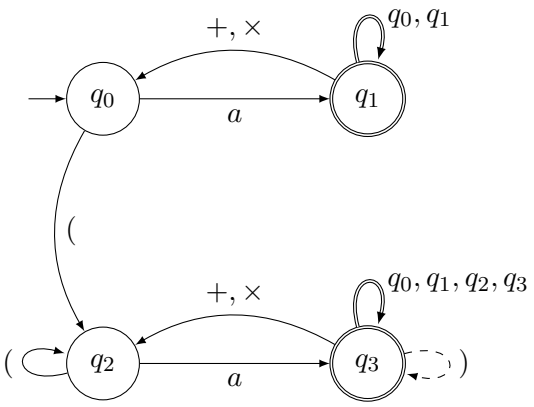


Figure 5.3: An automaton over the OPM of Figure 5.2 for the language of the grammar of Figure 5.1. Shift, push, and pop transitions are denoted by dashed, normal, and double arrows, respectively. Note that pop transitions are not consuming a symbol, but instead check the state where the symbol on top of the stack was pushed.

5.2 Weighted OPL and Their Relation to Weighted VPL

In this section, we introduce a weighted extension of operator precedence automata. We show that weighted OPL include weighted VPL and give examples showing how these weighted automata can express behaviors which were not expressible before. In the following, let $\mathbb{K} = (K, +, \cdot, 0, 1)$ be a *semiring* (cf. Section 3.1).

Definition 5.4. A *weighted OPA (wOPA)* $\mathcal{A}$ over an OP alphabet (Σ, M) and a semiring $\mathbb{K}$ is a tuple $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$, where $\mathrm{wt} = (\mathrm{wt}_{\mathrm{push}}, \mathrm{wt}_{\mathrm{shift}}, \mathrm{wt}_{\mathrm{pop}})$, consisting of

- an OPA $\mathcal{A}' = (Q, I, F, \delta)$ over (Σ, M) and
- the weight functions $\mathrm{wt}_{op} : \delta_{op} \to K$, $op \in \{\mathrm{push}, \mathrm{shift}, \mathrm{pop}\}$.

We call a wOPA *restricted*, denoted by *rwOPA*, if $\mathrm{wt}_{\mathrm{pop}}(q, p, r) = 1$ for each $(q, p, r) \in \delta_{\mathrm{pop}}$.

A *configuration* of a wOPA is a tuple $C = \langle \Pi, q, w\#, k \rangle$, where $(\Pi, q, w\#)$ is a configuration of the OPA $\mathcal{A}'$ and $k \in \mathbb{K}$. A *run* of $\mathcal{A}$ is a sequence of configurations $C_0 \vdash C_1 \ldots \vdash C_m$ satisfying the previous conditions and, additionally, the weight of a configuration is updated by multiplying with the weight of the encountered transition, as follows. As before, we let a be the first component of the topmost symbol of the stack Π, or $\#$ if the stack is $\bot$, and let b be the next symbol of the input to read:

$$\begin{aligned}
\langle \Pi, q, bx, k\rangle &\vdash \langle \Pi[b,q], r, x, k \cdot \mathrm{wt}_{\mathrm{push}}(q,b,r)\rangle && \text{if } a \lessdot b \text{ and } (q,b,r) \in \delta_{\mathrm{push}},\\
\langle \Pi[a,p], q, bx, k\rangle &\vdash \langle \Pi[b,p], r, x, k \cdot \mathrm{wt}_{\mathrm{shift}}(q,b,r)\rangle && \text{if } a \doteq b \text{ and } (q,b,r) \in \delta_{\mathrm{shift}},\\
\langle \Pi[a,p], q, bx, k\rangle &\vdash \langle \Pi, r, bx, k \cdot \mathrm{wt}_{\mathrm{pop}}(q,p,r)\rangle && \text{if } a \gtrdot b \text{ and } (q,p,r) \in \delta_{\mathrm{pop}}.
\end{aligned}$$

We call a run ρ *accepting* if it goes from $\langle \bot, q_I, w\#, 1\rangle$ to $\langle \bot, q_F, \#, k\rangle$, where $q_I \in I$ and $q_F \in F$. For such an accepting run, the *weight of* ρ is defined as $\mathrm{wt}(\rho) = k$. We denote by $\mathrm{acc}(\mathcal{A}, w)$ the set of all accepting runs of $\mathcal{A}$ on w.

Finally, the *behavior of* $\mathcal{A}$ is a function $[\![\mathcal{A}]\!] : (\Sigma, M)^+ \to K$, defined as

$$[\![\mathcal{A}]\!](w) = \sum_{\rho \in \mathrm{acc}(\mathcal{A},w)} \mathrm{wt}(\rho) \ .$$

Every function $S : (\Sigma, M)^+ \to K$ is called an *OP-series* (short: *series*, also *weighted language*). A wOPA $\mathcal{A}$ *recognizes* or *accepts* a series S if $[\![\mathcal{A}]\!] = S$. A series S is called *regular* or a *wOPL* if there exists an wOPA $\mathcal{A}$ accepting it. S is *strictly regular* or an *rwOPL* if there exists an rwOPA $\mathcal{A}$ accepting it.

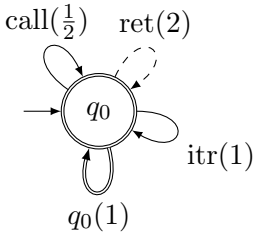
Example 5.1. Let us resume, in a simplified version, an example presented in [LMPP15] (Example 8) which exploits the ability of OPA to pop many items from the stack without advancing the input head: in this way we can model a system that manages calls and returns in a traditional LIFO policy but discards all pending calls if an interrupt occurs[1]. The weighted automaton of Figure 5.4 attaches weights to the OPA's transitions in such a way that the final weight of a string is 1 only if no pending call is discarded by any interrupt. Otherwise, the more calls are discarded, the lower the "quality" of the input as measured by its weight.

More precisely, we define $\Sigma = \{\mathrm{call}, \mathrm{ret}, \mathrm{itr}\}$ and the precedence matrix M as a subset of the matrix of Example 8 of [LMPP15], i.e., $\mathrm{call} \lessdot \mathrm{call}$, $\mathrm{call} \doteq \mathrm{ret}$, $\mathrm{call} \gtrdot \mathrm{itr}$, $\mathrm{itr} \lessdot \mathrm{itr}$, $\mathrm{itr} \gtrdot \mathrm{call}$, and $\mathrm{ret} \gtrdot a$ for all $a \in \Sigma$. Then, we define $\mathcal{A}_{\mathrm{penalty}}$ over (Σ, M) and the semiring $(\mathbb{Q}, +, \cdot, 0, 1)$ as in Figure 5.4. We use the same graphical notation for transitions as before and give the weights in brackets at transitions. Then, $\mathcal{A}_{\mathrm{penalty}}$ operates as follows.

Whenever, the automaton reads a call, it pushes 'call' and multiplies the accumulated weight by $\frac{1}{2}$. Afterwards, if we read a return, the precedence relation $\mathrm{call} \doteq \mathrm{ret}$ ensures that the automaton executes a shift with weight 2 that replaces the 'call' on top of the stack with a 'ret'. Then, M forces that the next transition is a pop.

If, on the other hand, after reading a call, we encounter an interrupt instead of a return, then the precedence relation $\mathrm{call} \gtrdot \mathrm{itr}$ forces an instant pop of this 'call'. In this case, the automaton continues to pop everything on the stack, until the stack is empty. Then, $\mathcal{A}_{\mathrm{penalty}}$ can proceed with another interrupt (which, however, would have no more significant effect) or a call as before.

[1] A similar motivation inspired the recent extension of VPL as colored nested words by [AF16].



	call	ret	itr
call	$\lessdot$	$\doteq$	$\gtrdot$
ret	$\gtrdot$	$\gtrdot$	$\gtrdot$
itr	$\gtrdot$		$\lessdot$

Figure 5.4: The weighted OPA $\mathcal{A}_{\text{penalty}}$ penalizing unmatched calls together with the given precedence matrix.

Let $\#\text{pcall}(w)$ be the number of pending calls of w, i.e., calls which are never answered by a return. Then, it follows that the behavior of $\mathcal{A}_{\text{penalty}}$ is $[\![\mathcal{A}_{\text{penalty}}]\!](w) = (\frac{1}{2})^{\#\text{pcall}(w)}$.

The example can be easily enriched by following the path outlined in [LMPP15]: we could add symbols specifying the serving of an interrupt, add different types of calls and interrupts with different priorities and more sophisticated policies (e.g., lower level interrupts disable new calls but do not discard them, whereas higher level interrupts reset the whole system, etc.). $\diamond$

Example 5.2. The wOPA of Figure 5.4 is "rooted" in a deterministic OPA; thus the semiring of weights is exploited in a fairly trivial way since only the $\cdot$ operation is used. The automaton $\mathcal{A}_{\text{policy}}$ given in Figure 5.5, instead, formalizes a more complex system where the penalties for unmatched calls may change nondeterministically within intervals delimited by the special symbol \$. More precisely, the occurrences of \$ mark intervals during which sequences of calls, returns, and interrupts occur. Then, usually unmatched calls are not penalized, but there is a special, nondeterministically chosen interval during which they are penalized. Finally, the global weight assigned to an input sequence is the maximum over all nondeterministic runs that are possible when recognizing the sequence.

Here, the alphabet is $\Sigma = \{\text{call}, \text{ret}, \text{itr}, \$\}$, and the OPM M, with $a \lessdot \$$ and $\$ \gtrdot a$, for all $a \in \Sigma$ is a natural extension of the OPM of Example 5.1. As semiring, we take $\mathbb{R}_{\max} = (\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)$. Then, $[\![\mathcal{A}_{\text{policy}}]\!](w)$ equals the maximal number of pending calls between two consecutive \$. Again, the automaton $\mathcal{A}_{\text{policy}}$ can be easily modified and enriched to formalize several variations of its policy: e.g., different policies could be associated with different intervals, different weights could be assigned to different types of calls or interrupts, and different policies could also be defined by choosing other semirings. $\diamond$

Note that both automata, $\mathcal{A}_{\text{penalty}}$ and $\mathcal{A}_{\text{policy}}$, do not use the weight assignment for pops.

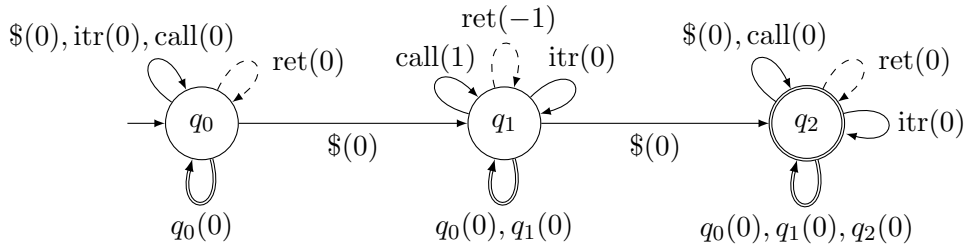


Figure 5.5: The weighted OPA $\mathcal{A}_{\text{policy}}$ penalizing unmatched calls nondeterministically

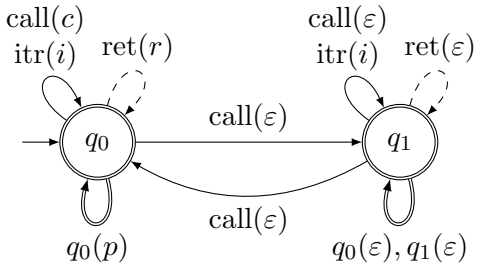


Figure 5.6: The wOPA $\mathcal{A}_{\text{log}}$ nondeterministically writing logs at different levels of detail

Example 5.3. The next automaton $\mathcal{A}_{\text{log}}$, depicted in Figure 5.6, chooses non-deterministically between logging everything and logging only 'important' information, e.g., only interrupts (this could be a system dependent on energy, WiFi, ...). Notice that, unlike the previous examples, in this case assigning nontrivial weights to pop transitions is crucial.

Let $\Sigma = \{\text{call}, \text{ret}, \text{itr}\}$, and define M as for $\mathcal{A}_{\text{penalty}}$. We employ the semiring $(\text{Fin}_{\Sigma'}, \cup, \circ, \emptyset, \{\varepsilon\})$ of all finite languages over $\Sigma' = \{c, r, p, i\}$. Then, $[\![\mathcal{A}_{\text{log}}]\!](w)$ yields all possible logs on w. ◇

As hinted at by our last example, the following proposition shows that in general, wOPA are more expressive than rwOPA.

Proposition 5.5. *There exists an OP alphabet (Σ, M) and a semiring $\mathbb{K}$ such that there exists a weighted language S which is regular but not strictly regular.*

Proof. Let $\Sigma = \{c, r\}$, $c \lessdot c$, and $c \doteq r$. Consider the semiring $\text{Fin}_{\{a,b\}}$ of all finite languages over $\{a, b\}$ together with union and concatenation. Let $n \in \mathbb{N}$ and $S : (\Sigma, M)^+ \to \text{Fin}_{\{a,b\}}$ be the following series

$$S(w) = \begin{cases} \{a^n b a^n\} & , \text{ if } w = c^n r \\ \emptyset & , \text{ otherwise} \end{cases} .$$

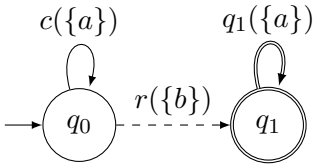


Figure 5.7: The wOPA recognizing $S(c^n r) = \{a^n b a^n\}$ and $S(w) = 0$, otherwise

Then, we can define a wOPA which only reads $c^n r$, assigns the weight $\{a\}$ to every push and pop, and the weight $\{b\}$ to the one shift, and therefore accepts S, as in Figure 5.7.

Now, we show with a pumping argument that there exists no rwOPA which recognizes S. Assume there is an rwOPA $\mathcal{A}$ with $[\![\mathcal{A}]\!] = S$. Note that for all $n \in \mathbb{N}$, the structure of $c^n r$ is $c \lessdot c \lessdot ... \lessdot c \doteq r$. Let ρ be an accepting run of $\mathcal{A}$ on $c^n r$ with $\mathrm{wt}(\rho) = \{a^n b a^n\}$. Then, the transitions of ρ consist of n pushes, followed by a shift, followed by n pops and can be written as follows.

$$q_0 \xrightarrow{c} q_1 \xrightarrow{c} ... \xrightarrow{c} q_{n-1} \xrightarrow{c} q_n \xrightarrow{r} q_{n+1} \overset{q_{n-1}}{\Longrightarrow} q_{n+2} \overset{q_{n-2}}{\Longrightarrow} ... \overset{q_1}{\Longrightarrow} q_{2n} \overset{q_0}{\Longrightarrow} q_{2n+1}$$

Both the amount of states and the amount of pairs of states are bounded. If n is sufficiently large, there exists two pop transitions $\mathrm{pop}(q, p, r)$ and $\mathrm{pop}(q', p', r')$ in this sequence such that $q = q'$ and $p = p'$. This means that we have a loop in the pop transitions going from state q to $q' = q$. Furthermore, the corresponding push to the first transition of this loop was invoked when the automaton was in state p', while the corresponding push to the last pop was invoked in state p. Since $p = p'$, we also have a loop at the corresponding pushes. Then, the run where we skip both loops in the pops and in the pushes is an accepting run for $c^{n-k} r$, for some $k \in \mathbb{N} \setminus \{0\}$.

Since the weight of all pops is trivial, the weight of the loop of pop transitions is ε. If the weight of the loop of push transitions is also ε, then we have an accepting run for $c^{n-k} r$ of weight $\{a^n b a^n\}$, a contradiction. If the weight of the push-loop is not trivial, then by a simple case distinction it has to be either $\{a^i\}$ for some $i \in \mathbb{N} \setminus \{0\}$ or it has to contain the b. In the first case, the run without both loops has weight $\{a^{n-i} b a^n\}$ or $\{a^n b a^{n-i}\}$, in the second case it has weight $\{a^j\}$, for some $j \in \mathbb{N}$. All these runs are not of the form $a^{n-k} b a^{n-k}$, a contradiction. □

We notice that using the same arguments, we can show that also no weighted nested word automata as defined in [Mat10b, DP14b] can recognize this series. Even stronger, we can prove that restricted weighted OPLs are a generalization of weighted VPLs in the following sense. We shortly recall the important definitions. Let $\Sigma = \Sigma_{\mathrm{call}} \uplus \Sigma_{\mathrm{int}} \uplus \Sigma_{\mathrm{ret}}$ be a *visibly pushdown alphabet*. A *VPA* is a pushdown automaton which uses a push transition if and only if it reads a call symbol and a pop transition if and only if it reads a return symbol.

	Σ_{call}	Σ_{ret}	Σ_{int}
Σ_{call}	$\lessdot$	$\doteq$	$\lessdot$
Σ_{ret}	$\gtrdot$	$\gtrdot$	$\gtrdot$
Σ_{int}	$\gtrdot$	$\gtrdot$	$\gtrdot$

Figure 5.8: OPM for VPL

In [CM12], it was shown that using the complete OPM of Figure 5.8, for every VPA, there exists an equivalent operator precedence grammar which in turn can be transformed into an equivalent OPA.

In [Mat10b] and [DP14b], *weighted nested word automata* (wNWA) were introduced. These add semiring weights at every transition again depending on the information which symbols are calls, internals, or returns. Since every nested word has a unique representation over a visibly pushdown alphabet Σ, it can be interpreted as a precedence word of $(\Sigma, M)^+$, where M is the OPM of Figure 5.8. In particular, we can interpret a wVPL, i.e., the language of a wNWA, as an OP-series $(\Sigma, M)^+ \to \mathbb{K}$.

Theorem 5.6. *Let $\mathbb{K}$ be a semiring, Σ be a visibly pushdown alphabet, and M be the OPM of Figure 5.8. Then for every wNWA $\mathcal{A}$ defined as in [DP14b], there exists an rwOPA $\mathcal{B}$ with $[\![\mathcal{A}]\!](w) = [\![\mathcal{B}]\!](w)$ for all $w \in (\Sigma, M)^+$.*

We give an intuition for this result as follows. Note that although sharing some similarities, pushes, shifts, and pops are not the same thing as calls, internals, and returns. Indeed, a return of a (w)NWA reads and 'consumes' a symbol, while a pop of an (rw)OPA just pops the stack and leaves the next symbol untouched.

After studying Figure 5.8, this leads to the important observation that every symbol of Σ_{ret} and therefore every return transition of an NWA is simulated not by a pop, but by a shift transition of an OPA followed by a pop transition (in, both, the unweighted and weighted case).

We give a short demonstrating example: Let $\Sigma_{\text{int}} = \{a\}$, $\Sigma_{\text{call}} = \{\langle c\}$, $\Sigma_{\text{ret}} = \{r\rangle\}$, $w = a\langle car\rangle$. Then every run of an NWA for this word looks like

$$q_0 \xrightarrow{\quad a \quad} q_1 \xrightarrow{\langle c} q_2 \xrightarrow{\quad a \quad} q_3 \xrightarrow{\quad r\rangle \quad} q_4 \ .$$

Every run of an OPA on w (using the OPM of Figure 5.8) looks as follows

$$q_0 \xrightarrow{a} q_1' \Rightarrow q_1 \xrightarrow{\langle c} q_2 \xrightarrow{a} q_3' \Rightarrow q_3 \overset{r\rangle}{\dashrightarrow} q_4' \Rightarrow q_4 \ ,$$

where the return was substituted (forced by the OPM) by a shift followed by a pop.

It follows that we can simulate a weighted call by a weighted push, a weighted internal by a weighted push together with a pop and a weighted return by a weighted shift together with a pop. Therefore, we may indeed omit weights at pop transitions.

Proof of Theorem 5.6. Given a weighted NWA $\mathcal{A} = (Q, I, F, (\delta_{\text{call}}, \delta_{\text{int}}, \delta_{\text{ret}}), (\text{wt}_{\text{call}}, \text{wt}_{\text{int}}, \text{wt}_{\text{ret}}))$ over Σ and $\mathbb{K}$, we construct an rwOPA $\mathcal{B} = (Q', I', F', (\delta_{\text{push}}, \delta_{\text{shift}}, \delta_{\text{pop}}), (\text{wt}'_{\text{push}}, \text{wt}'_{\text{shift}}, \text{wt}'_{\text{pop}}))$ over (Σ, M) and $\mathbb{K}$. We set $Q' = Q \cup (Q \times Q)$, $I' = I$, and $F' = F$. We define the relations δ_{push}, δ_{shift}, δ_{pop}, and the functions wt'_{push}, $\text{wt}'_{\text{shift}}$, and wt'_{pop} as follows.

We let δ_{push} contain all triples (q, a, r) with $(q, a, r) \in \delta_{\text{call}}$, and all triples $(q, a, (q, r))$ with $(q, a, r) \in \delta_{\text{int}}$. We set $\text{wt}'_{\text{push}}(q, a, r) = \text{wt}_{\text{call}}(q, a, r)$ and $\text{wt}'_{\text{push}}(q, a, (q, r)) = \text{wt}_{\text{int}}(q, a, r)$. Moreover, we let δ_{shift} contain all triples $(q, a, (p, r))$ with $(q, p, a, r) \in \delta_{\text{ret}}$ and set $\text{wt}'_{\text{shift}}(q, a, (p, r)) = \text{wt}_{\text{ret}}(q, p, a, r)$. Furthermore, we let δ_{pop} contain all triples $((q, r), q, r)$ with $(q, a, r) \in \delta_{\text{int}}$, and all triples $((p, r), p, r)$ with $(q, p, a, r) \in \delta_{\text{ret}}$, and set $\text{wt}'_{\text{pop}}((q, r), q, r) = \text{wt}'_{\text{pop}}((p, r), p, r) = 1$.

Then, a run analysis of $\mathcal{A}$ and $\mathcal{B}$, following the observations above, shows that $[\![\mathcal{B}]\!] = [\![\mathcal{A}]\!]$. □

Together with the result that OPA are strictly more expressive than VPAs [CM12], this yields a strict hierarchy of these three classes of weighted languages:

$$\text{wVPL} \subsetneq \text{rwOPL} \subsetneq \text{wOPL} \ .$$

Note that in the context of formal power series, wVPL strictly contain recognizable power series and wOPL form a proper subset of the class of algebraic power series, i.e., series recognized by weighted pushdown automata (see e.g. [KS86]). The following result shows that for commutative semirings the second part of this hierarchy collapses, i.e., rwOPA are equally expressive as wOPA (and therefore can be seen as a kind of normal form in this case).

Theorem 5.7. *Let $\mathbb{K}$ be a commutative semiring and (Σ, M) an OP alphabet. Let $\mathcal{A}$ be a wOPA. Then, there exists an rwOPA $\mathcal{B}$ with $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$.*

Proof. Let $\mathcal{A} = (Q, I, F, \delta, \text{wt})$ be a wOPA over (Σ, M) and $\mathbb{K}$. Note that for every pop transition of a wOPA, there exists exactly one push transition. We construct an rwOPA $\mathcal{B}$ over the state set $Q' = Q \times Q \times Q$ and with the same behavior as $\mathcal{A}$ with the following idea in mind. In the first state component $\mathcal{B}$ simulates $\mathcal{A}$. In the second and third state component of Q' the automaton $\mathcal{B}$ preemptively guesses the states q and r of the pop transition (q, p, r) of $\mathcal{A}$ which corresponds to the next push transition following after this configuration. This enables us to transfer the weight from the pop transition to the correct push transition.

The detailed construction of $\mathcal{B} = (Q', I', F', \delta', \text{wt}')$ over (Σ, M) and $\mathbb{K}$ is the following. If $Q = \emptyset$, then $[\![\mathcal{A}]\!] \equiv 0$ is trivially strictly regular. If Q

is nonempty, let $q \in Q$ be a fixed state. Then, we set $Q' = Q \times Q \times Q$, $I' = \{(q_1, q_2, q_3) \mid q_1 \in I, q_2, q_3 \in Q\}$, $F' = \{(q_1, q, q) \mid q_1 \in F\}$, and

$$\begin{aligned}
\delta'_{\text{push}} &= \{((q_1, q_2, q_3), a, (r_1, r_2, r_3)) \mid (q_1, a, r_1) \in \delta_{\text{push}} \text{ and } (q_2, q_1, q_3) \in \delta_{\text{pop}}\} \\
\delta'_{\text{shift}} &= \{((q_1, q_2, q_3), a, (r_1, q_2, q_3)) \mid (q_1, a, r_1) \in \delta_{\text{shift}}\} \\
\delta'_{\text{pop}} &= \{((q_1, q_2, q_3), (p_1, q_1, r_1), (r_1, q_2, q_3)) \mid (q_1, p_1, r_1) \in \delta_{\text{pop}}\} \ .
\end{aligned}$$

Here, every push of $\mathcal{B}$ controls that the previously guessed q_2 and q_3 can be used by a pop transition of $\mathcal{A}$ going from q_2 to q_3 with q_1 on top of the stack. Every pop controls that the symbols on top of the stack are exactly the ones used at this pop. Since the second and third state component are guessed for the next push, they are passed on whenever we read a shift or pop. The second and third component pushed at the first position of a word are guessed by an initial state. At the last push, which therefore has no following push and will propagate the second and third component to the end of the run, the automaton $\mathcal{B}$ has to guess the distinguished state used in the final states.

Therefore, $\mathcal{B}$ has exactly one accepting run (of the same length) for every accepting run of $\mathcal{A}$, and vice versa. Finally, we define the transition weights as follows.

$$\begin{aligned}
\text{wt}'_{\text{push}}((q_1, q_2, q_3), a, (r_1, r_2, r_3)) &= \text{wt}_{\text{push}}(q_1, a, r_1) \cdot \text{wt}_{\text{pop}}(q_2, q_1, q_3) \\
\text{wt}'_{\text{shift}}((q_1, q_2, q_3), a, (r_1, r_2, r_3)) &= \text{wt}_{\text{shift}}(q_1, a, r_1) \\
\text{wt}'_{\text{pop}} &\equiv 1 \ .
\end{aligned}$$

Then, the runs of $\mathcal{A}$ simulated by $\mathcal{B}$ have exactly the same weights but in a different ordering. Since $\mathbb{K}$ is commutative, it follows that $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$. □

In the following, we study closure properties of weighted OPA and restricted weighted OPA. As usual, we extend the operation $+$ and $\cdot$ to series $S, T : (\Sigma, M)^+ \to K$ by means of pointwise definitions as follows:

$$\begin{aligned}
(S + T)(w) &= S(w) + T(w) \text{ for each } w \in (\Sigma, M)^+ \\
(S \odot T)(w) &= S(w) \cdot T(w) \text{ for each } w \in (\Sigma, M)^+ \ .
\end{aligned}$$

Proposition 5.8. *The sum of two regular (resp. strictly regular) series over $(\Sigma, M)^+$ is again regular (resp. strictly regular).*

Proof. We use a standard disjoint union of two (r)wOPA accepting the given series to obtain a (r)wOPA for the sum as follows.

Let $\mathcal{A} = (Q, I, F, \delta, \text{wt})$ and $\mathcal{B} = (Q', I', F', \delta', \text{wt}')$ be two wOPA over (Σ, M) and $\mathbb{K}$. We construct a wOPA $\mathcal{C} = (Q'', I'', F'', \delta'', \text{wt}'')$ over (Σ, M) and $\mathbb{K}$ by defining $Q'' = Q \uplus Q'$, $I'' = I \cup I$, $F'' = F \cup F'$, $\delta'' = \delta \cup \delta'$. The weight function is defined by

$$\text{wt}''(t) = \begin{cases} \text{wt}(t) & , \text{ if } t \in \delta \\ \text{wt}'(t) & , \text{ if } t \in \delta' \end{cases} \ .$$

Then, $[\![\mathcal{C}]\!] = [\![\mathcal{A}]\!] + [\![\mathcal{B}]\!]$. Furthermore, if $\mathcal{A}$ and $\mathcal{B}$ are restricted, i.e. $\mathrm{wt} \equiv 1$ and $\mathrm{wt}' \equiv 1$, it follow that $\mathrm{wt}'' \equiv 1$, and therefore $\mathcal{C}$ is also restricted. □

Given a series $S : (\Sigma, M)^+ \to K$ and a language $L \subseteq (\Sigma, M)^+$, we define the series $S \cap L$ as follows

$$(S \cap L)(w) = \begin{cases} S(w) & , \text{ if } w \in L \\ 0 & , \text{ otherwise} \end{cases} .$$

Proposition 5.9. *Let $S : (\Sigma, M)^+ \to K$ be a regular (resp. strictly regular) series and $L \subseteq (\Sigma, M)^+$ an OPL. Then, the series $(S \cap L)$ is regular (resp. strictly regular). Furthermore, if $\mathbb{K}$ is commutative, then the product of two regular (resp. strictly regular) series over $(\Sigma, M)^+$ is again regular (resp. strictly regular).*

Proof. We use a product construction of automata.

Let $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$ be a wOPA over (Σ, M) and $\mathbb{K}$ with $[\![\mathcal{A}]\!] = S$ and let $\mathcal{B} = (Q', q_0', F', \delta')$ be a deterministic OPA over (Σ, M) with $L(\mathcal{B}) = L$. We construct a wOPA $\mathcal{C} = (Q'', I'', F'', \delta'', \mathrm{wt}'')$ over (Σ, M) and $\mathbb{K}$, with $[\![\mathcal{C}]\!] = (S \cap L)$, as follows. We define $Q'' = Q \times Q'$, $I'' = I \times \{q_0'\}$, $F'' = F \times F'$, and

$$\begin{aligned}
\delta''_{\mathrm{push}} &= \{((q,q'),a,(r,r')) \mid (q,a,r) \in \delta_{\mathrm{push}} \text{ and } \delta'_{\mathrm{push}}(q',a) = r'\} \ , \\
\delta''_{\mathrm{shift}} &= \{((q,q'),a,(r,r')) \mid (q,a,r) \in \delta_{\mathrm{shift}} \text{ and } \delta'_{\mathrm{shift}}(q',a) = r'\} \ , \\
\delta''_{\mathrm{pop}} &= \{((q,q'),(p,p'),(r,r')) \mid (q,p,r) \in \delta_{\mathrm{pop}} \text{ and } \delta'_{\mathrm{pop}}(q',p') = r'\} \ .
\end{aligned}$$

Then the weights of $\mathcal{C}$ are defined as

$$\begin{aligned}
\mathrm{wt}''_{\mathrm{push}}((q,q'),a,(r,r')) &= \mathrm{wt}_{\mathrm{push}}(q,a,r) \ , \\
\mathrm{wt}''_{\mathrm{shift}}((q,q'),a,(r,r')) &= \mathrm{wt}_{\mathrm{shift}}(q,a,r) \ , \\
\mathrm{wt}''_{\mathrm{pop}}((q,q'),(p,p'),(r,r')) &= \mathrm{wt}_{\mathrm{pop}}(q,p,r) \ .
\end{aligned}$$

Note that given a word w, the automata $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ have to use pushes, shifts, and pops at the same positions. Hence, every accepting run of $\mathcal{C}$ on w defines exactly one accepting run of $\mathcal{B}$ and exactly one accepting run of $\mathcal{A}$ on w with matching weights, and vice versa. We obtain

$$\begin{aligned}
[\![\mathcal{C}]\!](w) &= \sum_{\rho \in \mathrm{acc}(\mathcal{C},w)} \mathrm{wt}(\rho) \\
&= \sum_{\substack{\rho, \text{ such that} \\ \rho_{\restriction Q} \in \mathrm{acc}(\mathcal{A},w) \\ \rho_{\restriction Q'} \in \mathrm{acc}(\mathcal{B},w)}} \mathrm{wt}(\rho) \\
&= \begin{cases} \sum_{\rho \in \mathrm{acc}(\mathcal{A},w)} \mathrm{wt}(\rho) & , \text{ if the run of } \mathcal{B} \text{ on } w \text{ is accepting} \\ 0 & , \text{ otherwise} \end{cases} \\
&= (S \cap L)(w) \ .
\end{aligned}$$

It follows that, $[\![\mathcal{C}]\!] = S \cap L$.

For the second part of the proposition, let $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$ and $\mathcal{B} = (Q', I', F', \delta', \mathrm{wt}')$ be two wOPA. We construct a wOPA $\mathcal{P}$ as $\mathcal{P} = (Q \times Q', I \times I', F \times F', \delta^{\mathcal{P}}, \mathrm{wt}^{\mathcal{P}})$ where $\delta^{\mathcal{P}} = (\delta^{\mathcal{P}}_{\mathrm{push}}, \delta^{\mathcal{P}}_{\mathrm{shift}}, \delta^{\mathcal{P}}_{\mathrm{pop}})$ and set

$$\begin{aligned}
\delta^{\mathcal{P}}_{\mathrm{push}} &= \{((q,q'),a,(r,r')) \mid (q,a,r) \in \delta_{\mathrm{push}} \text{ and } (q',a,r') \in \delta'_{\mathrm{push}}\} \ , \\
\delta^{\mathcal{P}}_{\mathrm{shift}} &= \{((q,q'),a,(r,r')) \mid (q,a,r) \in \delta_{\mathrm{shift}} \text{ and } (q',a,r') \in \delta'_{\mathrm{shift}}\} \ , \\
\delta^{\mathcal{P}}_{\mathrm{pop}} &= \{((q,q'),(p,p'),(r,r')) \mid (q,p,r) \in \delta_{\mathrm{pop}} \text{ and } (q',p',r') \in \delta'_{\mathrm{pop}}\} \ ,
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{wt}^{\mathcal{P}}_{\mathrm{push}}((q,q'),a,(r,r')) &= \mathrm{wt}'_{\mathrm{push}}(q,a,r) \cdot \mathrm{wt}''_{\mathrm{push}}(q',a,r') \ , \\
\mathrm{wt}^{\mathcal{P}}_{\mathrm{shift}}((q,q'),a,(r,r')) &= \mathrm{wt}'_{\mathrm{shift}}(q,a,r) \cdot \mathrm{wt}''_{\mathrm{shift}}(q',a,r') \ , \\
\mathrm{wt}^{\mathcal{P}}_{\mathrm{pop}}((q,q'),(p,p'),(r,r')) &= \mathrm{wt}'_{\mathrm{pop}}(q,p,r) \cdot \mathrm{wt}''_{\mathrm{pop}}(q',p',r') \ .
\end{aligned}$$

It follows that $[\![\mathcal{P}]\!] = [\![\mathcal{A}]\!] \odot [\![\mathcal{B}]\!]$. Furthermore, if $\mathcal{A}$ and $\mathcal{B}$ are restricted, then so is $\mathcal{P}$. □

Next, we show that regular series are closed under projections which preserve the OPM. For two OP alphabets (Σ, M), (Γ, M') and a mapping $h : \Sigma \to \Gamma$, we write $h : (\Sigma, M) \to (\Gamma, M')$ and say h is *OPM-preserving* if for all $\boxdot \in \{\lessdot, \doteq, \gtrdot\}$, we have $a \boxdot b$ if and only if $h(a) \boxdot h(b)$. We can extend such an h to a function $h : (\Sigma, M)^+ \to (\Gamma, M')^+$ as follows. Given a word $w = (a_1 a_2 ... a_n) \in (\Sigma, M)^+$, we define $h(w) = h(a_1 a_2 ... a_n) = h(a_1)h(a_2)...h(a_n)$. Let $S : (\Sigma, M)^+ \to K$ be a series. Then, we define $h(S) : (\Gamma, M')^+ \to K$ for each $v \in (\Gamma, M')^+$ by

$$h(S)(v) = \sum_{\substack{w \in (\Sigma,M)^+ \\ h(w)=v}} S(w) \ . \tag{5.1}$$

Proposition 5.10. *Let $\mathbb{K}$ be a semiring, $S : (\Sigma, M)^+ \to K$ regular (resp. strictly regular), and $h : \Sigma \to \Gamma$ an OPM-preserving projection. Then, $h(S) : (\Gamma, M')^+ \to K$ is regular (resp. strictly regular).*

Proof. We follow an idea of [DV12] and its application in [DP14b] and [DD17]. Let $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$ be a wOPA over (Σ, M) and $\mathbb{K}$ with $[\![\mathcal{A}]\!] = S$. The main idea is to remember the last symbol read in the state to distinguish different runs of $\mathcal{A}$ which would otherwise coincide in $\mathcal{B}$. We construct the wOPA $\mathcal{B} = (Q', I', F', \delta', \mathrm{wt}')$ over (Σ, M) and $\mathbb{K}$ as follows. We set $Q' = Q \times \Sigma$, $I' = I \times \{a_0\}$ for some fixed $a_0 \in \Sigma$, and $F' = F \times \Sigma$. We define the transition relations $\delta' = (\delta'_{\mathrm{push}}, \delta'_{\mathrm{shift}}, \delta'_{\mathrm{pop}})$ for every $b \in \Gamma$ and $(q,a), (q',a'), (q'',a'') \in Q'$,

as

$$\begin{aligned}\delta'_{\text{push}} &= \{((q,a),b,(q',a')) \mid (q,a',q') \in \delta_{\text{push}} \text{ and } b = h(a')\} \ , \\ \delta'_{\text{shift}} &= \{((q,a),b,(q',a')) \mid (q,a',q') \in \delta_{\text{shift}} \text{ and } b = h(a')\} \ , \\ \delta'_{\text{pop}} &= \{((q,a),(q',a'),(q'',a)) \mid (q,q',q'') \in \delta_{\text{pop}}\} \ .\end{aligned}$$

Then, the weight functions are defined by

$$\begin{aligned}\text{wt}'_{\text{push}}((q,a),h(a'),(q',a')) &= \text{wt}_{\text{push}}(q,a',q') \ , \\ \text{wt}'_{\text{shift}}((q,a),h(a'),(q',a')) &= \text{wt}_{\text{shift}}(q,a',q') \ , \\ \text{wt}'_{\text{pop}}((q,a),(q',a'),(q'',a'')) &= \text{wt}_{\text{pop}}(q,q',q'') \ .\end{aligned}$$

Analogously to [DP14b] and [DD17], this implies that for every run ρ of $\mathcal{A}$ on w, there exists exactly one run ρ' of $\mathcal{B}$ on v with $h(w) = v$ and $\text{wt}(\rho) = \text{wt}(\rho')$. One difference to previous works is that a pop of a wOPA is not consuming the symbol. Therefore, while processing a pop, we choose not to change the symbol that we are currently remembering.

It follows that $[\![\mathcal{A}']\!] = h([\![\mathcal{A}]\!])$, so $h(S) = [\![\mathcal{A}']\!]$ is regular. Furthermore, if $\mathcal{A}$ is restricted, then so is $\mathcal{B}$. □

5.3 A Nivat Theorem for Weighted OPL

In this section, we establish a connection between weighted OPLs and strictly regular series. We show that strictly regular series are exactly those series which can be derived from a restricted weighted OPA with only one state, intersected with an unweighted OPL, and using an OPM-preserving projection of the alphabet.

Let $h : \Sigma' \to \Sigma$ be a map between two alphabets. Given an OP alphabet (Σ, M), we define $h^{-1}(M)$ by setting $h^{-1}(M)_{a'b'} = M_{h(a')h(b')}$ for all $a', b' \in \Sigma'$. As h is OPM-preserving, for every series $S : (\Sigma, M)^+ \to K$, we get a series $h(S) : (\Sigma', h^{-1}(M))^+ \to K$, using the sum over all pre-images as in formula (5.1).

Let $\mathcal{N}(\Sigma, M, \mathbb{K})$ comprise all series $S : (\Sigma, M)^+ \to K$ for which there exist an alphabet Σ', a map $h : \Sigma' \to \Sigma$, and a one-state rwOPA $\mathcal{B}$ over $(\Sigma', h^{-1}(M))$ and $\mathbb{K}$ and an OPL L over $(\Sigma', h^{-1}(M))$ such that $S = h([\![\mathcal{B}]\!] \cap L)$.

Now, we show that every strictly regular series can be decomposed into the above introduced fragments.

Proposition 5.11. *Let $S : (\Sigma, M)^+ \to K$ be a series. If S is strictly regular, then S is in $\mathcal{N}(\Sigma, M, \mathbb{K})$.*

Proof. We follow some ideas of [DKar] and [DP14a].

Let $\mathcal{A} = (Q, I, F, \delta, \text{wt})$ be a rwOPA over (Σ, M) and $\mathbb{K}$ with $[\![\mathcal{A}]\!] = S$. We set $\Sigma' = Q \times \Sigma \times Q$ as the extended alphabet. The intuition is that Σ' consists

of the push and the shift transitions of $\mathcal{A}$. Let h be the projection of Σ' to Σ and let $M' = h^{-1}(M)$.

Let $L \subseteq (\Sigma', M')^+$ be the language consisting of all words w' over the extended alphabet such that $h(w')$ has an accepting run on $\mathcal{A}$ which uses at every position the push, resp. the shift transition defined by the symbol of Σ' at this position.

We construct the unweighted OPA $\mathcal{A}' = (Q', I', F', \delta')$ over (Σ', M'), accepting L, as follows. We set $Q' = Q$, $I' = I$, $F' = F$, and define δ' as follows

$$\begin{aligned}
\delta'_{\text{push}} &= \{\, (q, (q, a, p), p) \mid (q, a, p) \in \delta_{\text{push}} \,\} \ , \\
\delta'_{\text{shift}} &= \{\, (q, (q, a, p), p) \mid (q, a, p) \in \delta_{\text{shift}} \,\} \ , \\
\delta'_{\text{pop}} &= \delta_{\text{pop}} \ .
\end{aligned}$$

Hence, $\mathcal{A}'$ has an accepting run on a word $w' \in (\Sigma', M')^+$ if and only if $\mathcal{A}$ has an accepting run on $h(w')$, using the push and shift transitions defined by w'.

We construct the one-state rwOPA $\mathcal{B} = (Q'', I'', F'', \delta'', \mathrm{wt}'')$ over (Σ', M') and $\mathbb{K}$ as follows. Set $Q'' = I'' = F'' = \{q\}$, $\delta''_{\text{push}} = \delta''_{\text{shift}} = \{(q, a', q) \mid a' \in \Sigma'\}$, $\delta''_{\text{pop}} = \{(q, q, q)\}$, $\mathrm{wt}''_{\text{push}}(q, a', q) = \mathrm{wt}_{\text{push}}(a')$, $\mathrm{wt}''_{\text{shift}}(q, a', q) = \mathrm{wt}_{\text{shift}}(a')$, for all $a' \in \Sigma'$, and $\mathrm{wt}''_{\text{pop}}(q, q, q) = 1$.

Let ρ be a run of $w = a_1 ... a_n \in (\Sigma, M)^+$ on $\mathcal{A}$ and ρ' a run of $w' = a'_1 ... a'_n \in (\Sigma', M')^+$ on $\mathcal{B}$. We denote by $\mathrm{wt}_{\mathcal{A}}(\rho, w, i)$, resp. $\mathrm{wt}_{\mathcal{B}}(\rho', w', i)$, the weight of the push or shift transition used by the run ρ, resp. ρ', at position i. Since $\mathcal{A}$ and $\mathcal{B}$ are restricted, for all their runs ρ, ρ', we have $\mathrm{wt}(\rho) = \prod_{i=1}^{|w|} \mathrm{wt}_{\mathcal{A}}(\rho, w, i)$, resp. $\mathrm{wt}(\rho') = \prod_{i=1}^{|w'|} \mathrm{wt}_{\mathcal{B}}(\rho', w', i)$. Furthermore, following its definition, the rwOPA $\mathcal{B}$ has exactly one run ρ for every word $w' \in (\Sigma', M')$ and for all $h(w') = w$ and for all $i \in \{1...n\}$, we have $\mathrm{wt}_{\mathcal{B}}(\rho', w', i) = \mathrm{wt}_{\mathcal{A}}(\rho, w, i)$. It follows that

$$\begin{aligned}
h([\![\mathcal{B}]\!] \cap L)(w) &= \sum_{\substack{w' \in (\Sigma', M')^+ \\ h(w') = w}} ([\![\mathcal{B}]\!] \cap L)(w') \\
&= \sum_{\substack{w' \in L(\mathcal{A}') \\ h(w') = w}} [\![\mathcal{B}]\!](w') \\
&= \sum_{\rho \in \mathrm{acc}(\mathcal{A}, w)} \prod_{i=1}^{|w|} \mathrm{wt}_{\mathcal{A}}(\rho, w, i) \\
&= \sum_{\rho \in \mathrm{acc}(\mathcal{A}, w)} \mathrm{wt}(\rho) \\
&= [\![\mathcal{A}]\!](w) = S(w) \ .
\end{aligned}$$

Hence, $S = h([\![\mathcal{B}]\!] \cap L)$, thus $S \in \mathcal{N}(\Sigma, M, \mathbb{K})$. □

Using this proposition and closure properties of series, we get the following Nivat theorem for weighted operator precedence automata.

Theorem 5.12. *Let $\mathbb{K}$ be a semiring and $S : (\Sigma, M)^+ \to K$ be a series. Then S is strictly regular if and only if $S \in \mathcal{N}(\Sigma, M, \mathbb{K})$.*

Proof. The "only if"-part of is immediate by Proposition 5.11.

For the converse, let Σ' be an alphabet, $h : \Sigma' \to \Sigma$, $L \subseteq (\Sigma', h^{-1}(M))^+$ be an OPL, $\mathcal{B}$ a one-state rwOPA, and $S = h([\![\mathcal{B}]\!] \cap L)$. Then Proposition 5.9 shows that $[\![\mathcal{B}]\!] \cap L$ is strictly regular. Now, Proposition 5.10 yields the result. □

5.4 Weighted MSO-Logic for OPL

In the following, we study the instance of the weighted MSO logic of Section 3.3 for precedence words and operator precedence languages. Since we are only studying these structures in the context of semirings, we only consider the logic also featuring a product.

Definition 5.13. We define the weighted logic $\mathrm{MSO}(\mathbb{K}, (\Sigma, M))$, for short $\mathrm{MSO}(\mathbb{K})$, as

$$\beta ::= \mathrm{Lab}_a(x) \mid x \leq y \mid x \curvearrowright y \mid x \in X \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta \mid \exists X.\beta$$
$$\varphi ::= \beta \mid k \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi \mid \bigoplus_x \varphi \mid \bigoplus_X \varphi \mid \prod_x \varphi$$

where $k \in \mathbb{K}$; x, y are first-order variables; and X is a second order variable.

Let $w \in (\Sigma, M)^+$ and $\varphi \in \mathrm{MSO}(\mathbb{K})$. We denote by $[w] = \{1, ..., |w|\}$ the set of all positions of w and define an *assignment* γ of variables $\mathcal{V}$ containing $\mathrm{free}(\varphi)$, as in Section 2.3. Given the extended alphabet $\Sigma_\mathcal{V} = \Sigma \times \{0,1\}^\mathcal{V}$, we define its natural OPM $M_\mathcal{V}$ such that for all $(a, s), (b, t) \in \Sigma_\mathcal{V}$ and all $\boxdot \in \{\lessdot, \doteq, \gtrdot\}$, we have $(a, s) \boxdot (b, t)$ if and only if $a \boxdot b$.

The *semantics* of φ (cf. Section 3.3) is a function $[\![\varphi]\!]_\mathcal{V} : (\Sigma_\mathcal{V}, M_\mathcal{V})^+ \to K$ for all valid (w, γ) inductively as seen in Figure 5.9. For not valid words, we set $[\![\varphi]\!]_\mathcal{V}$ to 0. For a sentence φ, we get $[\![\varphi]\!] : (\Sigma, M)^+ \to K$.

Example 5.4. Let us go back to the automaton $\mathcal{A}_{\text{policy}}$ over the semiring $\mathbb{K} = (\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)$ (as depicted in Figure 5.5). The following boolean formula β defines three subsets of positions, X_0, X_1, and X_2, representing, respectively, the positions where $\mathcal{A}_{\text{policy}}$ is in the states q_0, q_1, or q_2. Then, X_0 and X_2 contain the positions where unmatched calls are not penalized, and X_1 the positions where they are.

$$\begin{aligned}\beta = \quad & x \in X_0 \leftrightarrow \exists y \exists z (y > x \wedge z > x \wedge y \neq z \wedge \mathrm{Lab}_\$(y) \wedge \mathrm{Lab}_\$(z)) \\ & \wedge x \in X_1 \leftrightarrow \exists y \exists z \begin{pmatrix} y \leq x \leq z \wedge y \neq z \wedge \mathrm{Lab}_\$(y) \wedge \mathrm{Lab}_\$(z) \\ \wedge((x \neq y \wedge x \neq z) \rightarrow \neg\, \mathrm{Lab}_\$(x)) \end{pmatrix} \\ & \wedge x \in X_2 \leftrightarrow \exists y \exists z (y < x \wedge z < x \wedge y \neq z \wedge \mathrm{Lab}_\$(y) \wedge \mathrm{Lab}_\$(z))\end{aligned}$$

$$
\begin{array}{lcl}
[\![\beta]\!]_{\mathcal{V}}(w,\gamma) & = & \begin{cases} 1 & , \text{ if } (w,\gamma) \models \beta \\ 0 & , \text{ otherwise} \end{cases} \\
[\![k]\!]_{\mathcal{V}}(w,\gamma) & = & k \quad \text{for all } k \in \mathbb{K} \\
[\![\varphi \oplus \psi]\!]_{\mathcal{V}}(w,\gamma) & = & [\![\varphi]\!]_{\mathcal{V}}(w,\gamma) + [\![\psi]\!]_{\mathcal{V}}(w,\gamma) \\
[\![\varphi \otimes \psi]\!]_{\mathcal{V}}(w,\gamma) & = & [\![\varphi]\!]_{\mathcal{V}}(w,\gamma) \cdot [\![\psi]\!]_{\mathcal{V}}(w,\gamma) \\
[\![\bigoplus_x \varphi]\!]_{\mathcal{V}}(w,\gamma) & = & \sum\limits_{i \in |w|} [\![\varphi]\!]_{\mathcal{V} \cup \{x\}}(w, \gamma[x \to i]) \\
[\![\bigoplus_X \varphi]\!]_{\mathcal{V}}(w,\gamma) & = & \sum\limits_{I \subseteq |w|} [\![\varphi]\!]_{\mathcal{V} \cup \{X\}}(w, \gamma[X \to I]) \\
[\![\prod_x \varphi]\!]_{\mathcal{V}}(w,\gamma) & = & \prod\limits_{i \in |w|} [\![\varphi]\!]_{\mathcal{V} \cup \{x\}}(w, \gamma[x \to i])
\end{array}
$$

Figure 5.9: Semantics of MSO($\mathbb{K}$) for precedence words.

To simplify the weight assignment, we define the following weighted formula for any given boolean formula β' and any weight k (cf. Formula 4.4 on page 61)

$$(\beta' \to k) = \neg\beta' \oplus (\beta' \otimes k) \ .$$

Then, this formula yields the weight k if β holds, and the neutral element of the product of the semiring otherwise. That is

$$[\![\beta' \to k]\!](w,\gamma) = \begin{cases} k & , \text{ if } (w,\gamma) \models \beta' \\ 1_{\mathbb{K}} & , \text{ otherwise } . \end{cases}$$

Note that in this example, $1_{\mathbb{K}}$ equals the real weight 0.

Then, the weight assignment is formalized by $\varphi_{0,2}$ and φ_1 as follows. The formula $\varphi_{0,2}$ assigns the real weight 0 to calls, returns, and interrupts whenever the position is in X_0 or X_2 as follows

$$\varphi_{0,2} = \big((x \in X_0 \vee x \in X_2) \wedge (\mathrm{Lab}_{\mathrm{call}}(x) \vee \mathrm{Lab}_{\mathrm{ret}}(x) \vee \mathrm{Lab}_{\mathrm{itr}}(x))\big) \to 0 \ .$$

The formula φ_1 assigns the real weights $1, -1, 0$ to calls, returns, and interrupts, respectively, whenever the position is in X_1 as follows

$$
\begin{array}{rl}
\varphi_1 = & ((x \in X_1 \wedge \mathrm{Lab}_{\mathrm{call}}(x)) \to 1) \\
& \otimes ((x \in X_1 \wedge \mathrm{Lab}_{\mathrm{ret}}(x)) \to -1) \\
& \otimes ((x \in X_1 \wedge \mathrm{Lab}_{\mathrm{itr}}(x)) \to 0) \\
& \otimes (\mathrm{Lab}_{\$}(x) \to 0) \ .
\end{array}
$$

Note that here $\otimes$ equals the addition which ignores all assigned weights 0. Then, the formula $\psi = \prod_x(\beta \otimes \varphi_{0,2} \otimes \varphi_1)$ ensures that β holds at every

position and defines the weight assigned by $\mathcal{A}_{\text{policy}}$ to an input string through a single nondeterministic run. Finally, the formula $\chi = \bigoplus_{X_0} \bigoplus_{X_1} \bigoplus_{X_2} \psi$ defines the global weight of every string in an equivalent way as the one defined by $\mathcal{A}_{\text{policy}}$. $\diamond$

As before, by Lemma 3.5, we know that $[\![\varphi]\!]_{\mathcal{V}}(w,\gamma) = [\![\varphi]\!](w, \gamma\restriction_{\text{free}(\varphi)})$ for each valid $(w,\gamma) \in (\Sigma_{\mathcal{V}}, M)^+$ and by a standard induction on the structure of φ we get following.

Lemma 5.14. *Let $\varphi \in \text{MSO}(\mathbb{K})$ and let $\mathcal{V}$ be a finite set of variables with $\text{free}(\varphi) \subseteq \mathcal{V}$. Then, $[\![\varphi]\!]$ is regular (resp. strictly regular) iff $[\![\varphi]\!]_{\mathcal{V}}$ is regular (resp. strictly regular).*

As shown by [DG07] in the case of words, the full weighted logic is strictly more powerful than weighted automata. A similar example also applies here. Therefore, in the following, we restrict our logic in an appropriate way as in Section 4.4. The main idea for this is to allow only functions with finitely many different values (step functions) after a product quantification. Furthermore, in the non-commutative case, we either also restrict the application of $\otimes$ to step functions or we enforce all occurring weights (constants) of $\varphi \otimes \theta$ to commute.

As in Definition 4.14, we call a weighted formula *almost boolean* if it contains no weighted quantification. Adapting ideas and definitions from Section 4.4, the following propositions show that almost boolean formulas precisely describe a certain form of rwOPA's behaviors, which we call *OPL step functions*.

Definition 5.15. For $k \in \mathbb{K}$ and a language $L \subseteq (\Sigma, M)^+$, we define $\mathbb{1}_L : (\Sigma, M)^+ \to \mathbb{K}$, the *characteristic series* of L, i.e. $\mathbb{1}_L(w) = 1$ if $w \in L$, and $\mathbb{1}_L(w) = 0$ otherwise. We denote by $k\mathbb{1}_L : (\Sigma, M)^+ \to \mathbb{K}$ the characteristic series of L multiplied by k, i.e. $k\mathbb{1}_L(w) = k$ if $w \in L$, and $k\mathbb{1}_L(w) = 0$ otherwise.

A series S is called an *OPL step function*, if it has a representation

$$S = \sum_{i=1}^{n} k_i \mathbb{1}_{L_i} \ ,$$

where L_i are OPL forming a partition of $(\Sigma, M)^+$ and $k_i \in \mathbb{K}$ for each $i \in \{1, ..., n\}$; so $[\![\varphi]\!](w) = k_i$ iff $w \in L_i$, for each $i \in \{1, ..., n\}$.

Lemma 5.16. *The set of all OPL step functions is closed under $+$ and $\odot$.*

Proof. Let $S = \sum_{i=1}^{k} k_i \mathbb{1}_{L_i}$ and $S' = \sum_{j=1}^{\ell} k'_j \mathbb{1}_{L'_j}$ be OPL step functions. Then the following holds

$$S + S' = \sum_{i=1}^{k} \sum_{j=1}^{\ell} (d_i + d'_j) \mathbb{1}_{L_i \cap L'_j} \ ,$$

$$S \odot S' = \sum_{i=1}^{k} \sum_{j=1}^{\ell} (d_i \cdot d'_j) \mathbb{1}_{L_i \cap L'_j} \ .$$

Since $(L_i \cap L'_j)$ are also OPL and form a partition of $(\Sigma, M)^+$, it follows that $S + S'$ and $S \odot S'$ are also OPL step functions. □

Proposition 5.17. *(a) For every almost boolean formula φ, $[\![\varphi]\!]$ is an OPL step function.*

(b) If S is an OPL step function, then there exists an almost boolean formula φ such that $S = [\![\varphi]\!]$.

Proof. (a) We show the first statement by structural induction on φ. If φ is boolean, then $[\![\varphi]\!] = \mathbb{1}_{L(\varphi)}$, were $L(\varphi)$ and $L(\neg\varphi)$ are OPL due to Theorem 5.3. Therefore, $[\![\varphi]\!] = 1_K \mathbb{1}_{L(\varphi)} + 0_K \mathbb{1}_{L(\neg\varphi)}$ is an OPL step function. If $\varphi = k$, $k \in \mathbb{K}$, then $[\![k]\!] = k\mathbb{1}_{(\Sigma,M)^+}$ is an OPL step function. Let $\mathcal{V} = \mathrm{free}(\varphi_1) \cup \mathrm{free}(\varphi_2)$. By lifting Lemma 5.14 to OPL step functions as in [DP14a] and by Lemma 5.16, we see that $[\![\varphi_1 \oplus \varphi_2]\!] = [\![\varphi_1]\!]_\mathcal{V} + [\![\varphi_2]\!]_\mathcal{V}$ and $[\![\varphi_1 \otimes \varphi_2]\!] = [\![\varphi_1]\!]_\mathcal{V} \odot [\![\varphi_2]\!]_\mathcal{V}$ are also OPL step functions.

(b) Given an OPL step function $[\![\varphi]\!] = \sum_{i=1}^n k_i \mathbb{1}_{L_i}$, we use Theorem 5.3 to get φ_i with $[\![\varphi_i]\!] = \mathbb{1}_{L_i}$. Then, the second statement follows from setting $\varphi = \bigoplus_{i=1}^n (k_i \otimes \varphi_i)$ and the fact that the OPL $(L_i)_{1\leq i\leq n}$ form a partition of $(\Sigma, M)^+$. □

Proposition 5.18. *Let S be an OPL step function. Then S is strictly regular.*

Proof. Let $n \in \mathbb{N}$, $(L_i)_{1\leq i\leq n}$ be OPL forming a partition of $(\Sigma, M)^+$ and $k_i \in \mathbb{K}$ for each $i \in \{1, ..., n\}$ such that

$$S = \sum_{i=1}^n k_i \mathbb{1}_{L_i} \ .$$

It is easy to construct a rwOPA with two states recognizing the constant series $[\![k_i]\!]$ which assigns the weight k_i to every word. Hence, $k_i \mathbb{1}_{L_i} = [\![k_i]\!] \cap L_i$ is strictly regular by Proposition 5.9. Therefore, by Proposition 5.8, S is strictly regular. □

Definition 5.19. Let $\varphi \in \mathrm{MSO}(\mathbb{K})$. We denote by $\mathrm{const}(\varphi)$ all weights of $\mathbb{K}$ occurring in φ and we call φ *$\otimes$-restricted* if for all subformulas $\psi \otimes \theta$ of φ, the formula ψ is almost boolean or $\mathrm{const}(\psi)$ and $\mathrm{const}(\theta)$ commute elementwise. We call φ *$\prod$-restricted* if for all subformulas $\prod_x \psi$ of φ, ψ is almost boolean. We call φ *restricted* if it is both $\otimes$- and $\prod$-restricted.

In Example 5.4, the formula β is boolean, the formulas ϕ are almost boolean, and ψ and χ are restricted. Notice that ψ and χ would be restricted even if $\mathbb{K}$ were not commutative.

The following proposition will find usage in Section 5.5.

Proposition 5.20. *Let $S : (\Sigma, M)^+ \to K$ be a regular (resp. strictly regular) series and $k \in \mathbb{K}$. Then $[\![k]\!] \odot S$ is regular (resp. strictly regular).*

Proof. Let $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$ be a wOPA such that $[\![\mathcal{A}]\!] = S$. Then we construct a wOPA $\mathcal{B} = (Q', I', F, \delta', \mathrm{wt}')$ as follows.

We set $Q \uplus I'$ and $I' = \{q'_I \mid q_I \in I\}$. The new transition relations δ' and weight functions wt' consist of all transitions of $\mathcal{A}$ with their respective weights and the following additional transitions: For every push transition (q_I, a, q) of δ_{push}, we add a push transition (q'_I, a, q) to δ'_{push} with $\mathrm{wt}'_{\mathrm{push}}(q'_I, a, q) = k \cdot \mathrm{wt}_{\mathrm{push}}(q_I, a, q)$.

Note that, as for OPA, every run of a wOPA has to start with a push transition. Therefore, $\mathcal{B}$ starts in a state $q'_I \in I$, uses one of the added transitions, and afterwards exactly simulates the automaton $\mathcal{A}$. Together with the weight assignment, this ensures that $\mathcal{B}$ uses the same weights as $\mathcal{A}$ except at the very first transition of every run which is multiplied by k from the left. It follows that $[\![\mathcal{B}]\!] = [\![k]\!] \odot S$. In particular, we did not change the weight of any pop transition, thus if $\mathcal{A}$ is restricted, so is $\mathcal{B}$. □

5.5 Characterization of Weighted OPL

In this section, we show the expressive equivalence of weighted operator precedence automata and the introduced weighted MSO logic. We follow the approach of Section 4.5. Note, however, that here we have to distinguish between regular and strictly regular series and the translation from automata to formula (Proposition 5.25) is more sophisticated.

Lemma 5.21 (Closure under weighted disjunction)**.** *Let φ and ψ be two formulas of* MSO($\mathbb{K}$) *such that $[\![\varphi]\!]$ and $[\![\psi]\!]$ are regular (resp. strictly regular). Then, $[\![\varphi \oplus \psi]\!]$ is regular (resp. strictly regular).*

Proof. We put $\mathcal{V} = \mathrm{free}(\varphi) \cup \mathrm{free}(\psi)$. Then, $[\![\varphi \oplus \psi]\!] = [\![\varphi]\!]_{\mathcal{V}} + [\![\psi]\!]_{\mathcal{V}}$ is regular (resp. strictly regular) by Lemma 5.14 and Proposition 5.8. □

Proposition 5.22 (Closure under restricted weighted conjunction)**.** *Let $\psi \otimes \theta$ be a subformula of a $\otimes$-restricted formula φ of* MSO($\mathbb{K}$) *such that $[\![\psi]\!]$ and $[\![\theta]\!]$ are regular (resp. strictly regular). Then, $[\![\psi \otimes \theta]\!]$ is regular (resp. strictly regular).*

Proof. Since φ is $\otimes$-restricted, the formula ψ is almost boolean or the constants of ψ and θ commute.

Case 1: Let us assume ψ is almost boolean. Then, we can write $[\![\psi]\!]$ as OPL step function, i.e., $[\![\psi]\!] = \sum_{i=1}^{n} k_i \mathbb{1}_{L_i}$, where L_i are OPL. So, the series $[\![\psi \otimes \theta]\!]$ equals a sum of series of the form $([\![k_i \otimes \theta]\!] \cap L_i)$. Then, by Proposition 5.20, $[\![k_i \otimes \theta]\!]$ is a regular (resp. strictly regular) series. Therefore, $([\![k_i \otimes \theta]\!] \cap L_i)$ is regular (resp. strictly regular) by Proposition 5.9. Hence, $[\![\psi \otimes \theta]\!]$ is (strictly) regular by Proposition 5.8.

Case 2: Let us assume that the constants of ψ and θ commute. Then, the second part of Proposition 5.9 yields the claim. □

Lemma 5.23 (Closure under $\sum_x$, $\sum_X$)**.** *Let φ be a formula of* MSO($\mathbb{K}$) *such that $[\![\varphi]\!]$ is regular (resp. strictly regular). Then, $[\![\sum_x \varphi]\!]$ and $[\![\sum_X \varphi]\!]$ are regular (resp. strictly regular).*

Proof. This is a proven by Lemma 5.14 and Proposition 5.10 analogously to Lemma 4.21 (cf. also [DG07]). □

Proposition 5.24 (Closure under restricted $\prod_x$)**.** *Let φ be an almost boolean formula of* MSO($\mathbb{K}$)*. Then, $[\![\prod_x \varphi]\!]$ is strictly regular.*

Proof. We use ideas of [DG07] and the extensions in [DP14b]. The proof follows the structure of Lemma 4.22 with the following intuition.

In the first part, we write $[\![\varphi]\!]$ as OPL step function and encode the information to which language $(w, \gamma[x \to i])$ belongs in a specially extended language $\tilde{L}$. Then we construct an MSO-formula for this language. Therefore, by Theorem 5.3, we get a deterministic OPA recognizing $\tilde{L}$. In the second part, we add the weights k_i to this automaton and return to our original alphabet. Note that in the second part, we only have to add weights to pushes and shifts, thus making sure that $[\![\prod_x \varphi]\!]$ is not only regular, but strictly regular.

More detailed, let $\varphi \in \text{MSO}(\mathbb{K}, (\Sigma, M))$. We define $\mathcal{V} = \text{free}(\prod x.\varphi)$ and $\mathcal{W} = \text{free}(\varphi) \cup \{x\}$. We consider the extended alphabets $\Sigma_\mathcal{V}$ and $\Sigma_\mathcal{W}$ together with their natural OPMs $M_\mathcal{V}$ and $M_\mathcal{W}$. By Proposition 5.17 and lifting Lemma 5.14 to OPL step functions, $[\![\varphi]\!]$ is an OPL step function. Let $[\![\varphi]\!] = \sum_{j=1}^{m} k_j \mathbb{1}_{L_j}$ where L_j is an OPL over $(\Sigma_\mathcal{W}, M_\mathcal{W})$ for all $j \in \{1, ..., m\}$ and (L_j) is a partition of $(\Sigma_\mathcal{W}, M_\mathcal{W})^+$. By the semantics of the product quantifier, we get for $(w, \gamma) \in (\Sigma_\mathcal{V}, M_\mathcal{V})^+$

$$
\begin{aligned}
[\![\textstyle\prod_x \varphi]\!](w, \gamma) &= \prod_{i \in [w]} ([\![\varphi]\!]_\mathcal{W}(w, \gamma[x \to i])) \\
&= \prod_{i \in [w]} (k_{g(i)}), \\
\text{where} \qquad g(i) &= \begin{cases} 1 & \text{, if } (w, \gamma[x \to i]) \in L_1 \\ \dots & \\ m & \text{, if } (w, \gamma[x \to i]) \in L_m \end{cases} \text{, for all } i \in [w] \ . \qquad (5.2)
\end{aligned}
$$

Now, in the first part, we encode the information to which language $(w, \gamma[x \to i])$ belongs in a specially extended language $\tilde{L}$ and construct an MSO-formula for this language. We define the extended alphabet $\tilde{\Sigma} = \Sigma \times \{1, ..., n\}$, together with its natural OPM $\tilde{M}$ which only refers to Σ, so:

$$
(\tilde{\Sigma}_\mathcal{V}, \tilde{M}_\mathcal{V})^+ = \{(w, g, \gamma) \mid (w, \gamma) \in (\Sigma_\mathcal{V}, M_V) \text{ and } g \in \{1, ..., m\}^{[w]}\} \ .
$$

We define the languages $\tilde{L}, \tilde{L}_j, \tilde{L}'_j \subseteq (\tilde{\Sigma}_\mathcal{V}, \tilde{M}_\mathcal{V})^+$ as follows:

$$\tilde{L} = \left\{ (w,g,\gamma) \;\middle|\; \begin{array}{l} (w,\gamma) \in (\tilde{\Sigma}_\mathcal{V}, \tilde{M}_\mathcal{V})^+ \text{ is valid and} \\ \text{for all } i \in [w],\ j \in \{1,...,m\} : \\ \quad g(i) = j \Rightarrow (w, \gamma[x \to i]) \in L_j \end{array} \right\} ,$$

$$\tilde{L}_j = \left\{ (w,g,\gamma) \;\middle|\; \begin{array}{l} (w,\gamma) \in (\tilde{\Sigma}_\mathcal{V}, \tilde{M}_\mathcal{V})^+ \text{ is valid and} \\ \text{for all } i \in [w] :\ g(i) = j \Rightarrow (w, \gamma[x \to i]) \in L_j \end{array} \right\} ,$$

$$\tilde{L}'_j = \{\, (w,g,\gamma) \mid \text{for all } i \in [w] :\ g(i) = j \Rightarrow (w, \gamma[x \to i]) \in L_j \,\} \ .$$

Then, $\tilde{L} = \bigcap_{j=1}^m \tilde{L}_j$. Hence, in order to show that $\tilde{L}$ is an OPL, it suffices to show that each $\tilde{L}_j$ is an OPL. By a standard procedure, compare [DG07], we obtain a formula $\tilde{\varphi}_j \in \mathrm{MSO}(\tilde{\Sigma}_\mathcal{V}, \tilde{M}_\mathcal{V})$ with $L(\tilde{\varphi}_j) = \tilde{L}'_j$. Therefore, by Theorem 5.3, $\tilde{L}'_j$ is an OPL. It is straightforward to define an OPA accepting $\tilde{N}_\mathcal{V}$, the language of all valid words. By closure under intersection, $\tilde{L}_j = \tilde{L}'_j \cap \tilde{N}_\mathcal{V}$ is also an OPL and so is $\tilde{L}$. Hence, there exists a deterministic OPA $\tilde{\mathcal{A}} = (Q, q_0, F, \tilde{\delta})$ recognizing $\tilde{L}$.

In the second part, we add weights to $\tilde{\mathcal{A}}$ as follows. We construct the rwOPA $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$ over $(\Sigma_\mathcal{V}, M_\mathcal{V})$ and $\mathbb{K}$ by adding to every push and shift transition of $\tilde{\mathcal{A}}$ with $g(i) = j$ the weight k_j.

That is, we keep the states, the initial state, and the accepting states, and for $\delta = (\delta_{\text{push}}, \delta_{\text{shift}}, \delta_{\text{pop}})$ and all $q, q', p \in Q$ and $(a,j,s) \in \tilde{\Sigma}_\mathcal{V}$, we set

$$\begin{aligned} \delta_{\text{push}} &= \{(q,(a,s),q') \mid (q,(a,j,s),q') \in \tilde{\delta}_{\text{push}}\} \ , \\ \delta_{\text{shift}} &= \{(q,(a,s),q') \mid (q,(a,j,s),q') \in \tilde{\delta}_{\text{shift}}\} \ , \\ \delta_{\text{pop}} &= \tilde{\delta}_{\text{pop}} \ , \\ \mathrm{wt}_{\text{push}}(q,(a,s),q') &= \begin{cases} k_j & , \text{ if } (q,(a,j,s),q') \in \tilde{\delta}_{\text{push}} \\ 0 & , \text{ otherwise} \end{cases} , \\ \mathrm{wt}_{\text{shift}}(q,(a,s),q') &= \begin{cases} k_j & , \text{ if } (q,(a,j,s),q') \in \tilde{\delta}_{\text{shift}} \\ 0 & , \text{ otherwise} \end{cases} , \\ \mathrm{wt}_{\text{pop}}(q,p,q') &= 1 \ . \end{aligned}$$

Since $\tilde{\mathcal{A}}$ is deterministic, for every $(w,g,\gamma) \in \tilde{L}$, there exists exactly one accepted run $\tilde{r}$ of $\tilde{\mathcal{A}}$. On the other hand, for every $(w,g,\gamma) \notin \tilde{L}$, there is no accepted run of $\tilde{\mathcal{A}}$. Since (L_j) is a partition of $(\Sigma_\mathcal{W}, M_\mathcal{W})^+$, for every $(w,\gamma) \in (\Sigma_\mathcal{V}, M_\mathcal{V})$, there exists exactly one g with $(w,g,\gamma) \in \tilde{L}$. Thus, every $(w,\gamma) \in (\Sigma_\mathcal{V}, M_\mathcal{V})$ has exactly one run r of $\mathcal{A}$ determined by the run $\tilde{r}$ of (w,g,γ) of $\tilde{\mathcal{A}}$. We denote by $\mathrm{wt}_\mathcal{A}(r,(w,\gamma),i)$ the weight used by the run r on (w,γ) over $\mathcal{A}$ at position i, which is always the weight of the push or shift transition used at this position. Then by definition of $\mathcal{A}$ and $\tilde{L}$, the following holds for all $i \in [w]$

$$g(i) = j \Rightarrow \mathrm{wt}_\mathcal{A}(r,(w,\gamma),i) = k_j \wedge (w, \gamma[x \to i]) \in L_j \ .$$

By formula (5.2), we obtain

$$[\![\varphi]\!]_{\mathcal{W}}(w, \gamma[x \to i]) = k_j = \mathrm{wt}_{\mathcal{A}}(r, (w, \gamma), i) \ .$$

Hence, for the behavior of the automaton $\mathcal{A}$ the following holds

$$\begin{aligned}
[\![\mathcal{A}]\!](w, \gamma) &= \sum_{r' \in \mathrm{acc}(\mathcal{A}, w)} \mathrm{wt}(r') \\
&= \prod_{i=1}^{|w|} \mathrm{wt}_{\mathcal{A}}(r, (w, \gamma), i) \\
&= \prod_{i=1}^{|w|} [\![\varphi]\!]_{\mathcal{W}}(w, \gamma[x \to i]) \\
&= [\![\textstyle\prod_x \varphi]\!](w, \gamma) \ .
\end{aligned}$$

Thus, $\mathcal{A}$ recognizes $[\![\prod_x \varphi]\!]$. □

The following proposition is a summary of the previous results.

Proposition 5.25. *For every restricted* MSO($\mathbb{K}$)*-sentence φ, there exists an rwOPA $\mathcal{A}$ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$.*

Proof. We use structural induction on φ. If φ is an almost boolean formula, then by Proposition 5.17, $[\![\varphi]\!]$ is an OPL step function. By Proposition 5.18, every OPL step function is strictly regular.

Closure under $\oplus$ is dealt with by Lemma 5.21, closure under $\otimes$ by Proposition 5.22. The sum quantifications $\sum_x$ and $\sum_X$ are dealt with by Lemma 5.23. Since φ is restricted, we know that for every subformula $\prod_x \psi$, the formula ψ is an almost boolean formula. Therefore, we can apply Proposition 5.24 to maintain recognizability of our formula in this case. □

The next proposition shows that the converse also holds.

Proposition 5.26. *For every rwOPA $\mathcal{A}$, there exists a restricted* MSO($\mathbb{K}$)*-sentence φ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$. If $\mathbb{K}$ is commutative, then for every wOPA $\mathcal{A}$, there exists a restricted* MSO($\mathbb{K}$)*-sentence φ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$.*

Proof. The rationale adopted to build formula φ from $\mathcal{A}$ integrates the approach followed in [DG07, DP14b] with the one of [LMPP15]. On the one hand we need second order variables suitable to "carry" weights; on the other hand, unlike previous non-OP cases which are managed through real-time automata, an OPA can perform several transitions while remaining in the same position. Thus, we introduce the following second order variables: $X^{\mathrm{push}}_{p,a,q}$ represents the set of positions where $\mathcal{A}$ performs a push move from state p, reading symbol a and reaching state q; $X^{\mathrm{shift}}_{p,a,q}$ has the same meaning as $X^{\mathrm{push}}_{p,a,q}$ for a shift operation;

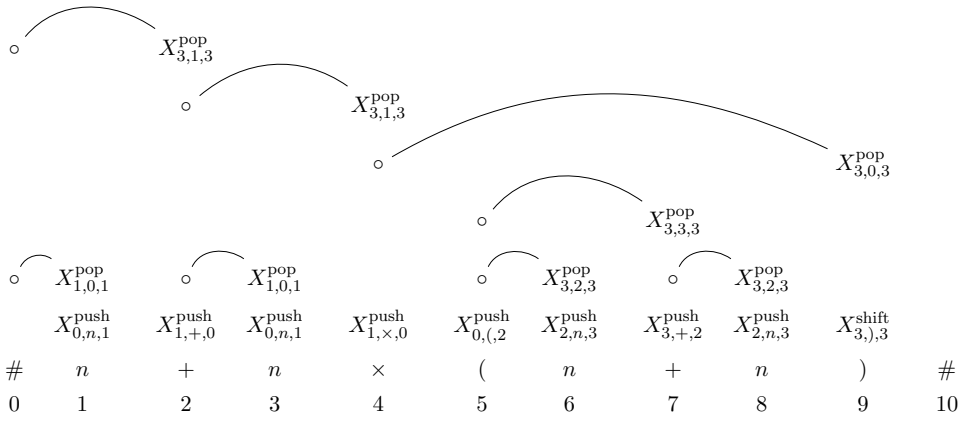


Figure 5.10: The string $n+n\times(n+n)$ with the second order variables evidenced for the automaton of Figure 5.3. The symbol $\circ$ marks the positions of the symbols that precede the push corresponding to the connected pop transition.

$X^{\text{pop}}_{p,q,r}$ represents the set of positions of the symbol that is on top of the stack when $\mathcal{A}$ performs a pop transition from state p, with q on top of the stack, reaching r.

Let $\mathcal{V}$ consist of all $X^{\text{push}}_{p,a,q}$, $X^{\text{shift}}_{p,a,q}$, and $X^{\text{pop}}_{p,q,r}$ such that $a \in \Sigma$, $p,q,r \in Q$ and $(p,a,q) \in \delta_{\text{push}}$ resp. δ_{shift}, resp. $(p,q,r) \in \delta_{\text{pop}}$. Since Σ and Q are finite, there is an enumeration $\bar{X} = (X_1, .., X_m)$ of all variables of $\mathcal{V}$. We denote by $\bar{X}^{\text{push}}$, $\bar{X}^{\text{shift}}$, and $\bar{X}^{\text{pop}}$ enumerations over only the respective set of second order variables.

We use the following usual abbreviations for unweighted formulas of MSO:

$$
\begin{aligned}
(\beta \wedge \varphi) &= \neg(\neg\beta \vee \neg\varphi) \ , \\
(\beta \to \varphi) &= (\neg\beta \vee \varphi) \ , \\
(\beta \leftrightarrow \varphi) &= (\beta \to \varphi) \wedge (\varphi \to \beta) \ , \\
(\forall x.\varphi) &= \neg(\exists x.\neg\varphi) \ , \\
(y = x) &= (x \le y) \wedge (y \le x) \ , \\
(y = x+1) &= (x \le y) \wedge \neg(y \le x) \wedge \forall z.(z \le x \vee y \le z) \ , \\
\min(x) &= \forall y.(x \le y) \ , \\
\max(x) &= \forall y.(y \le x) \ .
\end{aligned}
$$

Also, we use the shortcuts $\text{Tree}(x,z,v,y)$, $\text{Next}_i(x,y)$, $\text{Succ}_q(x,y)$, $Q_i(x,y)$, and $\text{Tree}_{p,q}(x,z,v,y)$, which were originally defined in [LMPP15] and are reported and adapted here for the reader's convenience.

The main idea behind the formula $\text{Tree}(x,z,v,y)$ is the following. Given $x \curvearrowright y$, we have a sequence of positions $x = k_1 < ... < k_m = y$, as defined in Section 2.2 (page 18). Then, we encode in z and v two other important

positions for this sequence: The position z should be the successor of x in this sequence, that is, the position where we execute the push resulting from $x \lessdot z$. The position v should be the predecessor of y in this sequence, that is, the position we mark with the respective X^{pop} resulting from $v \gtrdot y$.

For instance, with reference to Figure 5.10, $\text{Tree}(5,7,7,9)$ and $\text{Tree}(4,5,9,10)$ hold.

$$x \circ y := \bigvee_{a,b \in \Sigma, M_{a,b} = \circ} \text{Lab}_a(x) \wedge \text{Lab}_b(y), \text{ for } \circ \in \{\lessdot, \doteq, \gtrdot\} \ ,$$

$$\text{Tree}(x,z,v,y) := x \curvearrowright y \wedge \left(\begin{array}{c} (x+1 = z \vee x \curvearrowright z) \wedge \neg \exists t (z < t < y \wedge x \curvearrowright t) \\ \wedge \\ (v+1 = y \vee v \curvearrowright y) \wedge \neg \exists t (x < t < v \wedge t \curvearrowright y) \end{array} \right)$$

Furthermore, $\text{Succ}_q(x,y)$ holds for two successive positions where the OPA reaches state q through a push or shift at position y, while $\text{Next}_q(x,y)$ holds when a pop move reaches state q while completing a chain $x \curvearrowright y$.

$$\text{Succ}_q(x,y) := (x+1 = y) \wedge \bigvee_{p \in Q, a \in \Sigma} (x \in X^{\text{push}}_{p,a,q} \vee x \in X^{\text{shift}}_{p,a,q} \vee \min(x)) \ ,$$

$$\text{Next}_r(x,y) := \exists z \exists v. \left(\text{Tree}(x,z,v,y) \wedge \bigvee_{p,q \in Q} v \in X^{\text{pop}}_{p,q,r} \right) \ ,$$

$$Q_i(x,y) := \text{Succ}_i(x,y) \vee \text{Next}_i(x,y) \ .$$

Finally,

$$\text{Tree}_{i,j}(x,z,v,y) := \text{Tree}(x,z,v,y) \wedge Q_i(v,y) \wedge Q_j(x,z)$$

refines the predicate Tree by making explicit that i and j are, respectively, the current state and the state on top of the stack when the pop move is executed.

We now define the unweighted formula ψ to characterize all accepted runs

$$\psi = \mathit{Partition}(\bar{X}^{\text{push}}, \bar{X}^{\text{shift}}) \wedge \mathit{Unique}(\bar{X}^{\text{pop}}) \wedge \mathit{InitFinal} \\ \wedge \mathit{Trans}_{\text{push}} \wedge \mathit{Trans}_{\text{shift}} \wedge \mathit{Trans}_{\text{pop}} \ .$$

Here, the subformula *Partition* will enforce the push and shift sets to be (together) a partition of all positions, while the formula *Unique* will make sure that we mark every position with at most one X^{pop}. *InitFinal* controls the

initial and the acceptance condition.

$$\mathit{Partition}(X_1, ..., X_n) = \forall x. \bigvee_{i=1}^{n} \big[(x \in X_i) \wedge \bigwedge_{i \neq j} \neg(x \in X_j)\big] \ ,$$

$$\mathit{Unique}(X_1^{\text{pop}}, .., X_n^{\text{pop}}) = \forall x. \bigwedge_{i \neq j} \neg(x \in X_i^{\text{pop}} \wedge x \in X_j^{\text{pop}}) \ ,$$

$$\begin{aligned}\mathit{InitFinal} = \exists x \exists y \exists x' \exists y'. \big[& \min(x) \wedge \max(y) \wedge x + 1 = x' \wedge y' + 1 = y \\ & \wedge \bigvee_{\substack{i \in I,\, q \in Q \\ a \in \Sigma}} x' \in X_{i,a,q}^{\text{push}} \\ & \wedge \bigvee_{\substack{f \in F,\, q \in Q \\ a \in \Sigma}} (y' \in X_{q,a,f}^{\text{push}} \vee y' \in X_{q,a,f}^{\text{shift}}) \\ & \wedge \bigvee_{f \in F} (\text{Next}_f(x, y) \wedge \bigwedge_{j \neq f} \neg \text{Next}_j(x, y))\big] \ .\end{aligned}$$

The formulas $\mathit{Trans}_{\text{push}}$ $\mathit{Trans}_{\text{shift}}$ $\mathit{Trans}_{\text{pop}}$ control the respective transitions of the run according to their labels as follows.

$$\mathit{Trans}_{\text{push}} = \forall x. \bigwedge_{p,q \in Q, a \in \Sigma} \big(x \in X_{p,a,q}^{\text{push}} \rightarrow \big[\text{Lab}_a(x) \wedge \exists z.(z \lessdot x \wedge Q_p(z, x))\big]\big) \ ,$$

$$\mathit{Trans}_{\text{shift}} = \forall x. \bigwedge_{p,q \in Q, a \in \Sigma} \big(x \in X_{p,a,q}^{\text{shift}} \rightarrow \big[\text{Lab}_a(x) \wedge \exists z.(z \doteq x \wedge Q_p(z, x))\big]\big) \ ,$$

$$\mathit{Trans}_{\text{pop}} = \forall v. \bigwedge_{p,q \in Q} \big(\big[\bigvee_{r \in Q} v \in X_{p,q,r}^{\text{pop}}\big] \leftrightarrow \big[\exists x \exists y \exists z.(\text{Tree}_{p,q}(x, z, v, y))\big]\big) \ .$$

Note that these formulas, compared to [LMPP15], had to be adapted to fit our new second order variables. These new variables are crucial for the following assignment of weights. Also notice that in the transition formulas, the partition (resp. uniqueness) axioms guarantee that in every run the left side of the implication (resp. equivalence) is satisfied for only one triple (p, a, q), resp. (p, q, r).

Thus, with arguments similar to [LMPP15] it can be shown that the sentences satisfying ψ are exactly those recognized by the unweighted OPA subjacent to $\mathcal{A}$.

For an unweighted formula β and two weights k_1 and k_2, we define the following shortcut for an almost boolean weighted formula:

$$\beta \,?\, k_1 : k_2 = (\beta \otimes k_1) \oplus (\neg\beta \otimes k_2) \ .$$

Now, we add weights to ψ by defining the following restricted weighted formula

$$\theta = \psi \otimes \prod_x \bigotimes_{p,q\in Q} \Big(\bigotimes_{a\in\Sigma} (\, x \in X^{\mathrm{push}}_{p,a,q} \,?\, \mathrm{wt}_{\mathrm{push}}(p,a,q) : 1) \\ \otimes \bigotimes_{a\in\Sigma} (\, x \in X^{\mathrm{shift}}_{p,a,q} \,?\, \mathrm{wt}_{\mathrm{shift}}(p,a,q) : 1) \\ \otimes \bigotimes_{r\in Q} (\, x \in X^{\mathrm{pop}}_{p,q,r} \,?\, \mathrm{wt}_{\mathrm{pop}}(p,q,r) : 1) \Big) \ .$$

Here, the second part of θ multiplies up all weights of the encountered transitions. This is the crucial part where we either need that $\mathbb{K}$ is commutative or all pop weights are trivial because the product quantifier of θ assigns the pop weight at a different position than the occurrence of the respective pop transition in the automaton. Using only one product quantifier (weighted universal quantifier) this is unavoidable, since the number of pops at a given position is only bounded by the word length.

Since the subformulas, $x \in X^{()}_{()} ?\, \mathrm{wt}() : 1$, of θ are almost boolean, the subformula $\prod_x(...)$ of θ is $\prod$-restricted. Furthermore, ψ is boolean and so θ is $\otimes$-restricted. Thus, θ is a restricted formula.

Finally, we define

$$\varphi = \bigoplus_{X_1} \bigoplus_{X_2} \cdots \bigoplus_{X_m} \theta \ .$$

This implies $[\![\varphi]\!](w) = [\![\mathcal{A}]\!](w)$, for all $w \in (\Sigma, M)^+$. Therefore, φ is our required sentence with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$. □

The following theorem summarizes the main results of this section.

Theorem 5.27. *Let $\mathbb{K}$ be a semiring and $S : (\Sigma, M)^+ \to K$ a series.*

1. *The following are equivalent:*
   - *(i) $S = [\![\mathcal{A}]\!]$ for some rwOPA.*
   - *(ii) $S = [\![\varphi]\!]$ for some restricted sentence φ of* $\mathrm{MSO}(\mathbb{K})$.
2. *Let $\mathbb{K}$ be commutative. Then, the following are equivalent:*
   - *(i) $S = [\![\mathcal{A}]\!]$ for some wOPA.*
   - *(ii) $S = [\![\varphi]\!]$ for some restricted sentence φ of* $\mathrm{MSO}(\mathbb{K})$.

This theorem is a further step in the path of generalizing a series of results beyond the barrier of regular and structured CFLs of which VPL certainly obtained much attention. OPLs further generalize this language class not only in terms of strict inclusion, but mainly because they are not visible, in the sense explained in the introduction, nor are they necessarily real-time: this allows them to cover important examples that could not be adequately modeled through more restricted classes.

Theorem 4.23 also shows that the typical logical characterization of weighted languages does not generalize in the same way to the whole class wOPL since we either need the extra hypothesis that $\mathbb{K}$ is commutative or we have to restrict the usage of weights at pop-transitions. This is due to the fact that pop transitions are applied in the reverse order than that of positions to which they refer (position v in formula $\mathit{Trans}_{\mathrm{pop}}$). Notice, however, that also rwOPL do not forbid unbounded pop sequences and they too include languages that are neither real-time nor visible.

Chapter 6

Graph Automata

The theorem of Büchi, Elgot, and Trakhtenbrot [Büc60, Elg61, Tra61] states the equivalence of languages of words recognizable by a finite state machine and languages definable in monadic second order theory. As noted previously, this result had a huge positive impact in formal language theory and found many extensions to different structures; a general result for finite graphs was given by Thomas [Tho91]. Similarly, the weighted extension of Büchi's theorem by Droste and Gastin found many extensions (see e.g. [DV06, Mei06, Fic11, Mat10b]).

However, a general result for weighted automata and weighted logics covering graphs and therefore linking the previous results remained, up to now, open. Furthermore, it has remained an open question whether it is possible to get such a result even in the unweighted case for infinite graphs. In particular, different research groups tried unsuccessfully to prove this equivalence in the case of infinite pictures.

In this chapter, we solve the first question and give a promising approach to the second one. The detailed contributions of this chapter are the following.

- We establish a model of weighted automata on graphs, which extends both Thomas' graph acceptors [Tho96] and the previous weighted automata models for words, trees, pictures, and others. We show that this enables us to model new quantitative properties of graphs which could not be expressed by the previous models.
- To show the robustness of our model, we extend a classical result of Nivat [Niv68] to weighted automata on graphs, showing that their behaviors are exactly those which can be constructed from very particular weighted graph automata and recognizable graph languages, together with operations like morphisms and intersections.
- We derive a Büchi-type equivalence result for the expressive power of weighted automata and a suitable weighted logic on graphs. We obtain corresponding equivalence results for structures like words, trees, pictures,

and nested words as a consequence.

- We introduce graph acceptors operating on infinite graphs which are able to distinguish between finitely and infinitely many occurrences of patterns. We show a Büchi-like equivalence between these graph acceptors and an extended EMSO-logic for infinite graphs.
- We extend the Büchi characterization of weighted automata to infinite graphs.
- We study both semirings and valuation monoids as weight structure and apply the Transformation Theorem of Section 3.4 to graphs. This gives us the means to sharpen both Büchi-type equivalences according to the strength of assumptions on the weight structure.

We note that for graph walking automata, an interesting approach connecting pebble navigating weighted automata and weighted first-order logic was given in [BGMZ14, Mon13]. The present automata model is different, using tiles of graphs.

We consider directed graphs with bounded vertex degree as defined in Section 2.2. In contrast to special classes of graphs like words, trees, or pictures, these graphs are very general structures. For example, they do not have to admit an order on their vertices, can be disconnected, and can contain cycles. Thus, in comparison to previous cases, the following difficulties arise in our proofs.

The crucial difference to previous approaches is that a graph automaton has a global acceptance condition in form of a check of occurrence numbers of the tiles that appear in a run of the automaton on a graph. We need this condition to connect logic and automata. Furthermore, since we are dealing with graphs, the underlying unweighted automata model cannot be determinized. Accordingly, the unweighted logic subfragment covers only existential MSO. Also, the closure under weighted universal quantifications requires new methods; here we employ a theorem of Thomas [Tho96] whose proof in turn relied on Hanf's theorem [Han65].

Notably, these two results cannot be applied to infinite graph acceptors which contain an acceptance condition that is able to distinguish between finitely and infinitely many occurrences of states. Instead, we apply proof ideas from Thomas [Tho96] and Bollig and Kuske [BK07] and our result from Section 2.4 to show that graph acceptors and an EMSO-logic with an additional infinity operator for infinite graphs are equally expressive. To enhance readability, we first develop our weighted results in the finite case, designed in an adaptable way, thereby facilitating the later extensions to infinite graphs.

The majority of the results of this chapter was published in [DD15] and [Düc16]. Additionally, in Section 6.2.4, we show the following new result. We prove that weighted graph automata cannot recognize the constant series of rational non-integer numbers. On the other hand, if given a natural number or

an integer n, we can construct a weighted graph automaton which yields n for every input graph. This shows that while the natural numbers $\mathbb{N}$ are what we call a *regular weight structure*, the rational numbers $\mathbb{Q}$ are not. The distinction between regular and non-regular weight structures influences the applicability of our characterization result.

6.1 Thomas' Graph Acceptors

In this section, we restate the definitions of a *graph acceptor* as introduced by Thomas in [Tho91] and [Tho96]. We recapitulate the main result of [Tho96] which connects these graph acceptors to the existential fragment of classical MSO logic on graphs.

Definition 6.1. A *graph acceptor (GA)* $\mathcal{A}$ over the alphabets A and B is defined as a quadruple $\mathcal{A} = (Q, \Delta, \mathrm{Occ}, r)$ where

- Q is a finite set of states,
- $r \in \mathbb{N}$, the *tile-size*,
- Δ is a finite set of pairwise non-isomorphic r-tiles[1] over $A \times Q$ and B,
- Occ, the *occurrence constraint*, is a boolean combination of formulas "$\mathrm{occ}(\tau) \geq n$", where $n \in \mathbb{N}$ and $\tau \in \Delta$.

Given a graph G of $\mathrm{DG}_t(A, B)$ with a vertex set V, we call a mapping $\rho : V \to Q$ a *run (or tiling) of* $\mathcal{A}$ *on* G if for every $v \in V$, $\mathrm{sph}^r(G_\rho, v)$ is isomorphic to a tile in Δ. For every graph G and every run ρ, we consider the graph $G_\rho \in \mathrm{DG}_t(A \times Q, B)$, which consists of the same vertices and edges as G and is additionally labeled with $\rho(v)$ at every vertex v.

As before, we say G_ρ *satisfies* $\mathrm{occ}(\tau) \geq n$ if there exist at least n distinct vertices $v \in V$ such that $\mathrm{sph}^r(G_\rho, v)$ is isomorphic to τ. The semantics of "G_ρ *satisfies* Occ" are then defined in the usual way.

We call a run ρ *accepting* if G_ρ satisfies the constraint Occ. We say that $\mathcal{A}$ *accepts* the graph $G \in \mathrm{DG}_t(A, B)$ if there exists an accepting run ρ of $\mathcal{A}$ on G. We define $L(\mathcal{A}) = \{G \in \mathrm{DG}_t(A, B) \mid \mathcal{A} \text{ accepts } G\}$, the *language accepted by* $\mathcal{A}$. We call a language $L \subseteq \mathrm{DG}_t(A, B)$ *recognizable* if $L = L(\mathcal{A})$ for some GA $\mathcal{A}$. Since every accepting run applies to every vertex of the graph G a unique tile, we may also say that $\mathcal{A}$ *tiles* the graph G using specific tiles of Δ. An example of a graph G together with a run ρ is given in Figure 6.1.

Similarly, we also define acceptance of graph acceptors for languages of pointed graphs $L \subseteq \mathrm{pDG}_t(A, B)$. In this case the definitions above are exactly the same but we use pointed tiles instead of tiles, as defined in Section 2.2. In addition to the *center* which every tile has, a pointed tile has a second pointing

[1] For the definition of r-tiles, refer to page 14.

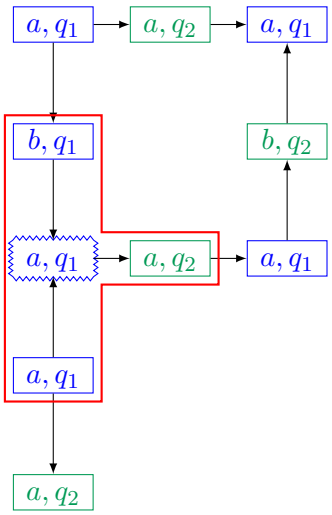


Figure 6.1: A graph G_ρ of $\mathrm{DG}_3(\{a, b\} \times \{q_1, q_2\}, \{1\})$ together with its 1-sphere around the serrated vertex. The edge labels were omitted. Given an automaton with tile-size 1, the marked sphere has to be part of the tile-set Δ of the automaton recognizing this graph. Note that the tiles necessarily 'overlap', because the sphere around *every* vertex of G has to be part of Δ.

which is either *empty* or marks a vertex in the pointed tile that is the root u of the whole pointed graph (G, u).

Note that Thomas (cf. [Tho91, Tho96]) uses non-pointed graphs. As the definition of a graph acceptor can be applied to both contexts, the pointing of a graph and therefore of its tiles can be seen as optional additional information to distinguish tiles from each other.

The following main result of Thomas [Tho96] states that graph acceptors exactly describe all graph languages definable by an existential MSO sentence as introduced in Section 2.3.

Theorem 6.2 ([Tho96])**.** *Let $L \subseteq \mathrm{DG}_t(A, B)$ be a set of graphs. Then:*

1. *L is recognizable by a one-state GA iff L is definable by an FO-sentence.*
2. *L is recognizable iff L is definable by an EMSO-sentence.*

This result is easily adaptable to languages of pointed graphs. In this case, the MSO logic additionally contains the formula $\mathrm{root}(x)$ denoting that x is the root of a pointed graph (G, u).

6.2 Weighted Graph Automata (WGA)

In this section, we introduce and investigate a quantitative version of graph automata. We follow the approach of [DD15] for finite graphs, but use the more general structure of *(graph-) valuation monoids* instead of commutative semirings (see [Düc16] and Section 3.2). We give multiple examples of this new model, incorporating both semirings and valuation monoids.

Furthermore, we give a Nivat-like and a Büchi-like characterization of weighted graph automata. The latter is the first result characterizing general weighted languages of graphs by a weighted MSO logic and linking previous results for special classes of graphs. Additionally, we study the regularity of weight structures for graphs in Section 6.2.4.

To avoid repetition, we concentrate on graphs of $\mathrm{DG}_t(A, B)$, which are not necessarily pointed. However, we can always go from non-pointed graphs to pointed graphs of $\mathrm{pDG}_t(A, B)$ by going from tiles to pointed tiles (cf. Section 2.2). The root of a graph is helpful for some examples (especially for infinite graphs) and for the application of the Transformation Theorem for weighted logics over left-multiplicative valuation monoids.

6.2.1 Introduction and Properties of WGA

In the following, we give the definition of a weighted graph automaton, we illustrate this new model with several examples, and we study first properties of weighted graph automata. By abuse of notation, we also consider graphs $\mathrm{DG}_t(M, B)$ over an infinite set M. Note that we use this notation only in our weight assignments of the weighted automaton and never as part of the input or within a tile. Note that as a special case of valuation monoids, the following theory is also applicable to commutative semirings.

Definition 6.3. A *weighted graph automaton (wGA)* over the alphabets A and B, and the valuation monoid $\mathbb{D}$ is a tuple $\mathcal{A} = (Q, \Delta, \mathrm{wt}, \mathrm{Occ}, r)$ where

- $\mathcal{A}' = (Q, \Delta, \mathrm{Occ}, r)$ is a graph acceptor over the alphabets A and B,
- $\mathrm{wt} : \Delta \to D$ is the weight function assigning to every tile of Δ a value of D.

An *run* $\rho : V \to Q$ *of* $\mathcal{A}$ *on* G and an *accepting run* are defined as an (accepting) run of $\mathcal{A}'$ on G. As in the unweighted case, G can be either unpointed or pointed.

For every vertex v of G, a run ρ uniquely defines $\mathrm{sph}^r(G_\rho, v)$, which is an r-tile of Δ around v. Let G_ρ^D be the unique graph over $\mathrm{DG}_t(D, B)$ resulting from the graph G where for all vertices v, $\mathrm{Lab}_{G_\rho^D}(v) = \mathrm{wt}(\mathrm{sph}^r(G_\rho, v))$. In other words, G_ρ^D is the graph G, where every vertex is labeled with the weight of the tile the run ρ defines around this vertex.

We denote by $\mathrm{acc}_{\mathcal{A}}(G)$ the set of all accepting runs of $\mathcal{A}$ on G. Then the behavior $[\![\mathcal{A}]\!] : \mathrm{DG}_t(A,B) \to D$ of a wGA $\mathcal{A}$ is defined, for each $G \in \mathrm{DG}_t(A,B)$, as

$$[\![\mathcal{A}]\!](G) = \sum_{\rho \in \mathrm{acc}_{\mathcal{A}}(G)} \mathrm{Val}(G_\rho^D) \ .$$

We call any function $S : \mathrm{DG}_t(A,B) \to D$ a *series*. Then S is *recognizable* if $S = [\![\mathcal{A}]\!]$ for some wGA $\mathcal{A}$. By the usual identification of languages with their characteristic functions, we see that graph acceptors are expressively equivalent to wGA over the Boolean semiring $\mathbb{B}$. In the special case that $\mathbb{D}$ is a semiring, we get

$$[\![\mathcal{A}]\!](G) = \sum_{\rho \in \mathrm{acc}_{\mathcal{A}}(G)} \prod_{v \in V} \mathrm{wt}(\mathrm{sph}^r(G_\rho, v)) \ .$$

Compared to classical weighted automata over words, $\mathrm{wt}(\mathrm{sph}^r(G_\rho, v))$ takes the place of a transition weight.

The following property of a weight structure is directly connected to this automata model and will prove very important in the later parts of this section.

Definition 6.4. A valuation monoid $\mathbb{D}$ is called *wGA-regular*, short *regular*, if all constant series of $\mathbb{D}$ are recognizable. An pv-monoid $\mathbb{PD}$ is called *regular* if its underlying valuation monoid is regular.

For example, the valuation monoid $\mathbb{D}_1 = (\mathbb{R} \cup \{-\infty\}, \sup, \mathrm{avg}, -\infty)$ is regular. This can be seen by building a one-state wGA $\mathcal{A}_d$ which assigns to every tile the value d. Then $[\![\mathcal{A}_d]\!](G) = d$ for every graph G. The valuation monoid $\mathbb{D}_2 = (\mathbb{R} \cup \{-\infty\}, \sup, \mathrm{disc}_\lambda, -\infty)$ is regular over pointed graphs. In this case, we need to assign the weight d to exactly the tile at the root of the graph and the weight 0 elsewhere.

We will discuss sufficient conditions for regularity of valuation monoids in Section 6.2.4. In the following, we give several examples of weighted graph automata.

Example 6.1. Weighted graph automata operate over the general class of t-bounded graphs, in particular, disconnected graphs. Therefore, the number of connected components of a graph is an interesting property. The following wGA describes this property by counting the number of connected components as exponent of 2.

We define the wGA $\mathcal{A} = (Q, \Delta, \mathrm{wt}, \mathrm{Occ}, r)$ over arbitrary alphabets A and B, and the semiring $\mathbb{K} = (\mathbb{N}, +, \cdot, 0, 1)$. We set $r = 1$, $\mathrm{Occ} = \mathit{true}$, $\mathrm{wt} \equiv 1$, and $Q = \{q_1, q_2\}$. The set of tiles is defined as $\Delta = \Delta_1 \cup \Delta_2$, where

$$\Delta_1 = \{\tau \mid \text{every vertex of } \tau \text{ is labeled with some } (a, q_1), a \in A\} \ ,$$
$$\Delta_2 = \{\tau \mid \text{every vertex of } \tau \text{ is labeled with some } (a, q_2), a \in A\} \ .$$

Note that we can use different alphabet symbols for vertices of a tile, but our tiles allow no mix of q_1 and q_2. Furthermore, the tile size is 1, thus, whenever the automaton labels one vertex with q_1 (resp. q_2), it has to label all directly connected vertices also with q_1 (resp. q_2). Since every 1-sphere of the labeled graph has to be part of Δ, it follows that every connected component of a given graph G is tiled either completely with q_1 or completely with q_2.

Consequently, for every connected component, we have two independent choices in an accepting run. Since $\mathrm{wt} \equiv 1$, $[\![\mathcal{A}]\!](G)$ equals the number of runs, thus

$$[\![\mathcal{A}]\!](G) = 2^{m(G)} \ ,$$

where $m(G)$ is the number of connected components of G. ◇

Example 6.2. The following weighted graph automaton uses the weights of a semiring $\mathbb{K}$ to attach to every vertex its degree. To simplify the example, the wGA does not distinguish between in- and out-going edges or edges with different labels (this would also be possible). Then we can use different semirings to assign to every graph $G \in \mathrm{DG}_t(A, B)$ its maximal or minimal degree or the sum of all degrees. Since the latter would always yield two times the number of edges, we give another wGA which works likewise, but takes only vertices labeled with $a \in A$ into account.

We define the wGAs $\mathcal{A}_i = (Q, \Delta, \mathrm{wt}_i, \mathrm{Occ}, r)$, $i \in \{1, 2\}$ over arbitrary alphabets A and B, and the semiring $\mathbb{K}$. We set $r = 1$, $\mathrm{Occ} = \mathit{true}$, $Q = \{q_0\}$, and $\Delta = \{\tau \mid \tau \text{ is a 1-tile}\}$. Let $\tau = (H, v)$ be a 1-tile around the center v. The weight functions wt_1 and wt_2 are defined as

$$\mathrm{wt}_1(\tau) = \mathrm{wt}_1(H, v) = |\mathrm{vertices}(H)| - 1 = \mathrm{degree}(v) \ ,$$
$$\mathrm{wt}_2(\tau) = \begin{cases} \mathrm{wt}_1(\tau) & , \text{ if } v \text{ is labeled with } (a, q_0) \\ 1_{\mathbb{K}} & , \text{ otherwise} \end{cases} .$$

Now, $[\![\mathcal{A}_i]\!](G) = \sum_{\rho \text{ acc. run}} \prod_{v \in V} \mathrm{wt}_i(\mathrm{sph}^r_{\mathcal{A}_i}(G_\rho, v))$, $i \in \{1, 2\}$, and since we have only one state, every graph has exactly one accepting run, so

$$[\![\mathcal{A}_1]\!](G) = \prod_{v \in V} \mathrm{degree}(v) \ ,$$
$$[\![\mathcal{A}_2]\!](G) = \prod_{\substack{v \in V, \\ v \text{ labeled with } (a, q_0)}} \mathrm{degree}(v) \ ,$$

where the product depends on the semiring. If, for example, we set $\mathbb{K} = (\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$, we get the sum of all degrees, resp. the sum of all degrees at a vertex labeled with $a \in A$. If, on the other hand, we use the max-min semiring $\mathbb{R}_{\max,\min} = (\mathbb{R}_+ \cup \{\infty\}, \max, \min, 0, \infty)$, we get the minimal degree of all vertices (of all vertices labeled with a, respectively). ◇

Example 6.3. Let $A = \{a, b\}$ and $B = \{x\}$. We are interested in the proportion of nodes labeled with a and without outgoing edges. For instance, in a tree this would refer to all leafs labeled with a. For a given graph G, we can compute this value with the following wGA over the valuation monoid $\mathbb{D}_1 = (\mathbb{R} \cup \{-\infty\}, \sup, \text{avg}, -\infty)$.

Set $\mathcal{A} = (Q, \Delta, \text{wt}, \text{Occ}, r)$, with $Q = \{q\}$, $r = 1$, $\Delta = \{\tau \mid \tau \text{ is a 1-tile}\}$, and Occ = *true*. Furthermore, we define $\text{wt}(\tau) = 1$ if the center v of τ is labeled with (a, q) and there is no vertex w of τ s.t. $E_x(v, w)$, and $\text{wt}(\tau) = 0$, otherwise. Then $[\![\mathcal{A}]\!](G)$ is the desired proportion. Note that using more states and bigger tile radii, we can easily modify this example to compute the ratio of more complex patterns. It is also possible to compute graph wide values like the average degree of a graph or the average of the number of neighbors labeled with b of all vertices. ◇

Example 6.4. Let us assume our graph represents a social network. Now, we are interested in the affinity of a person to a certain characteristic (a hobby, a political tendency, an attribute, etc.) be it to use this information in a matching process or for personalized advertising.

We assume that this affinity is not only related to the person itself, but also takes the social environment of the person into account. Here, in the first step, we assume a radius of 1 (which are all direct friends) and we assume that for every friend the affinity increases linearly. E.g., I am more inclined to watch soccer if I play soccer myself or I have close friends who play soccer. In the second step, to take a bigger social environment into account, we use a discount function to sum up over all the assigned values discounted by the distance to the person we are interested in.

Now, let $\mathbb{D}_2 = (\mathbb{R} \cup \{-\infty\}, \sup, \text{disc}_\lambda, -\infty)$. We define the wGA $\mathcal{A} = (q, \{\tau \mid \tau \text{ is a 1-tile}\}, \text{wt}, \textit{true}, 1)$ over $A = \{a, b\}$, $B = \{x\}$, and $\mathbb{D}_2$, with $\text{wt}(\tau) = \#_a(\tau)$, where $\#_a(\tau)$ is the number of vertices of τ labeled with a. Then depending on λ, $\mathcal{A}$ computes for a given pointed graph (G, u) the affinity of u to the characteristic a. Note that again $\mathcal{A}$ could make use of more sophisticated weight assignments. ◇

In the following, we show that it is possible to increase the size of the tiles without losing definability by Occ or recognizability in our automaton. To count occurrences of smaller tiles within bigger tiles or, more generally, about occurrences of tiles with a certain pattern, we introduce the following notation. Let τ^* be a finite set of tiles enumerated by $(\tau_1, ..., \tau_m)$. For $N \in \mathbb{N}$, we define the formula

$$\Big(\sum_{\tau \in \tau^*} \text{occ}(\tau)\Big) \geq N \quad \text{as} \quad \bigvee_{\substack{\sum_{i=1}^m n_i = N \\ n_i \in \{0,...,N\}}} \bigwedge_{i=1,...,m} \text{occ}(\tau_i) \geq n_i \ . \tag{6.1}$$

We can interpret τ^* as a set of tiles matching a certain pattern. Then this

formula is true if the occurrence number of all tiles matching this pattern is at least N.

Lemma 6.5. *Let A and B be two alphabets and $s \in \mathbb{N}$. Let* Occ *be a boolean combination of formulas "$\mathrm{occ}(\tau) \geq n$" where τ are s-tiles over A and B and $n \in \mathbb{N}$.*

Then for all $r \geq s$, there exists a boolean combination Occ$'$ *of formulas "$\mathrm{occ}(\tau') \geq m$" where τ' are r-tiles over A and B and $m \in \mathbb{N}$, such that for all $G \in \mathrm{DG}_t(A,B)$, G satisfies* Occ *if and only if G satisfies* Occ$'$.

Proof. Let $r \geq s$. Let $\varphi = \mathrm{occ}(\tau) \geq N$ occur in of Occ. We denote by τ^* the set of all r-tiles (H, v) where $\mathrm{sph}^s(H, v)$ is isomorphic to τ. Since A and B are finite and the tiles are of bounded degree, τ^* is finite and can be enumerated by $(\tau_1, ..., \tau_m)$. We rewrite φ using these r-tiles and the formula (6.1):

$$\varphi = \big(\sum_{\tau \in \tau^*} \mathrm{occ}(\tau) \big) \geq N \ .$$

Following the arguments above, this formula is true if the occurrence number of all tiles matching τ^* (i.e. all r-tiles that match τ if restricted to their s-sphere) is at least N, which is precisely φ. We apply this method to all (atomic) parts of Occ, and our claim follows. □

Lemma 6.6. *Let $S : \mathrm{DG}_t(A,B) \to D$ be a series recognizable by a wGA $\mathcal{A}$ with tile-size s. Then for all $r \geq s$, S is recognizable by a wGA $\mathcal{B}$ with tile-size r.*

Proof. Let $\mathcal{A} = (Q, \Delta_\mathcal{A}, \mathrm{wt}_\mathcal{A}, \mathrm{Occ}_\mathcal{A}, s)$ be a wGA with $[\![\mathcal{A}]\!] = \mathrm{S}$. We construct the wGA $\mathcal{B} = (Q, \Delta_\mathcal{B}, \mathrm{wt}_\mathcal{B}, \mathrm{Occ}_\mathcal{B}, r)$. We use Lemma 6.5 to rewrite $\mathrm{Occ}_\mathcal{A}$ into $\mathrm{Occ}_\mathcal{B}$. We set $\Delta_\mathcal{B} = \{(H, v) \mid \mathrm{sph}^s(H, v) \in \Delta_\mathcal{A}\}$. For $(H, v) \in \Delta_\mathcal{B}$, we define $\mathrm{wt}_\mathcal{B}(H, v) = \mathrm{wt}_\mathcal{A}(\mathrm{sph}^s(H, v))$.

Let $G \in \mathrm{DG}_t(A,B)$. Hence, $\mathcal{B}$ mimics $\mathcal{A}$ exactly regarding the s-spheres, and has no additional restrictions on the vertices with distance greater than s within an r-sphere. Therefore, $\rho : V \to Q$ is an accepting run of $\mathcal{A}$ on G if and only if ρ is an accepting run of $\mathcal{B}$ on G. Every run uses the same weights on both automata. Thus, $[\![\mathcal{A}]\!](G) = [\![\mathcal{B}]\!](G)$. □

Example 6.5. We continue Example 6.2, and using formula (6.1), we construct an alternative wGA $\mathcal{A}_3 = (Q_3, \Delta_3, \mathrm{wt}_3, \mathrm{Occ}_3, r)$, as follows. We set $r = 1$, $Q_3 = \{q_0, q_1\}$, and $\Delta_3 = \{\tau \mid \tau \text{ is a 1-tile}\}$. We denote by τ^* all tiles τ which are labeled at the center with (a, q_1) for some $a \in A$, and define

$$\mathrm{wt}_3(\tau) = \begin{cases} \mathrm{wt}_1(\tau) & , \text{ if } \tau \in \tau^* \\ 1_\mathbb{K} & , \text{ otherwise} \end{cases} ,$$

$$\mathrm{Occ}_3 = \big(\sum_{\tau \in \tau^*} \mathrm{occ}(\tau) \big) \geq 1 \wedge \neg \big(\big(\sum_{\tau \in \tau^*} \mathrm{occ}(\tau) \big) \geq 2 \big) \ .$$

This ensures that we use the state q_1 at exactly one position. Therefore, for every vertex, we get exactly one accepting run. Moreover, every accepting run has exactly the weight $1_{\mathbb{K}} \cdot ... 1_{\mathbb{K}} \cdot \mathrm{wt}_1(\tau) \cdot 1_{\mathbb{K}} \cdot ... 1_{\mathbb{K}} = \mathrm{wt}_1(\tau)$ where $\mathrm{wt}_1(\tau)$ is the degree of the vertex marked with q_1. It follows that

$$[\![\mathcal{A}_3]\!](G) = \sum_{v \in V} \mathrm{degree}(v),$$

and we are again free to choose a semiring with the desired summation. For example, the tropical semirings now yield the maximal (resp. minimal) degree of all vertices, and the semiring of the natural numbers yields the sum. Again, this approach is modifiable by taking only special vertices or special edges into account. ◇

Example 6.6. The following weighted graph automaton computes the *weighted diameter* (i.e. the maximal distance between two vertices) of edge-weighted graphs up to a threshold $N \in \mathbb{N}$. Let $B = \{1, ..., N\}$ be the edge labels, which describe the length of an edge. Note that, here, we sum over the length of the edge when computing shortest paths between vertices. It is also possible to set $B = \{1\}$ to get the classical notion of the *diameter* of a graph.

Here, for the sake of convenience, we restrict ourselves to undirected graphs, i.e., $(v, w) \in E_i \Leftrightarrow (w, v) \in E_i$. Similarly, we ignore the vertex labels and set $A = \{a\}$, although, it would also be possible to take directions and vertex labels into account.

We define the wGA $\mathcal{A} = (Q, \Delta, \mathrm{wt}, \mathrm{Occ}, r)$ over A, B, and the tropical semiring $\mathbb{R}_{\max} = (\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)$ as follows. We set $Q = \{0, ..., N\} \times \{0, 1\}$, and $r = 1$.

Let $\tau = (H, v)$ be a 1-tile. We refer with $\mathrm{Lab}^{\mathrm{c}}_Q(\tau)$ to the first component of the state used at the center of the tile τ. We refer with $\mathrm{Lab}^{E_i}_Q(\tau, w)$ to the first component of the state at a vertex w which is connected to the center of the tile τ by an edge labeled with i, i.e. $(v, w) \in E_i$. For $x, y \in \mathbb{Z}$ with $x \leq y$, $y \geq 0$, and $x \leq N$, we define $[x, y] = \{x, ..., y\}$ and $[x, y]^0_N = \{\max(0, x), ..., \min(y, N)\}$.

Then we define the set of tiles as

$$\Delta = \Delta_0 \cup \Delta_N \cup \bigcup_{k=1}^{n-1} \Delta_k \ ,$$

$$\Delta_0 = \left\{ \tau = (H, v) \;\middle|\; \begin{array}{l} \mathrm{Lab}^{\mathrm{c}}_Q(\tau) = 0 \\ \wedge\ \mathrm{Lab}^{E_i}_Q(\tau, w) \in [1, i], \text{ for all } (v, w) \in E_i \end{array} \right\} \ ,$$

$$\Delta_N = \left\{ \tau = (H, v) \;\middle|\; \begin{array}{l} \mathrm{Lab}^{\mathrm{c}}_Q(\tau) = N \\ \wedge\ \mathrm{Lab}^{E_i}_Q(\tau, w) \in [N - i, N], \text{ for all } (v, w) \in E_i \end{array} \right\} \ ,$$

$$\Delta_k = \left\{ \tau = (H, v) \;\middle|\; \begin{array}{l} \mathrm{Lab}^{\mathrm{c}}_Q(\tau) = k \\ \wedge\ \mathrm{Lab}^{E_i}_Q(\tau, w) \in [k - i, k + i]^0_N, \text{ for all } (v, w) \in E_i \\ \wedge\ \mathrm{Lab}^{E_i}_Q(\tau, w) = k - i, \text{ for at least one } (v, w) \in E_i \end{array} \right\} \ .$$

We refer with $\mathrm{Lab}^{\mathrm{c2}}_Q(\tau)$ to the second component of the state used at the center of the tile τ, and we define the weight function as

$$\mathrm{wt}(\tau) = \begin{cases} \mathrm{Lab}^{\mathrm{c}}_Q(\tau) & , \text{ if } \mathrm{Lab}^{\mathrm{c2}}_Q(\tau) = 1 \\ 0 & , \text{ otherwise} \end{cases} \ .$$

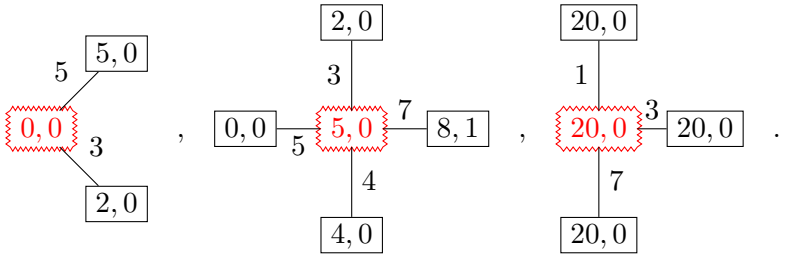
Finally, we define the occurrence constraint as

$$\begin{aligned} \mathrm{Occ} = \Big(\sum_{\mathrm{Lab}^{\mathrm{c}}_Q(\tau)=0} \mathrm{occ}(\tau) \Big) \geq 1 \wedge \neg \Big(\Big(\sum_{\mathrm{Lab}^{\mathrm{c}}_Q(\tau)=0} \mathrm{occ}(\tau) \Big) \geq 2 \Big) \\ \wedge \Big(\sum_{\mathrm{Lab}^{\mathrm{c2}}_Q(\tau)=1} \mathrm{occ}(\tau) \Big) \geq 1 \wedge \neg \Big(\Big(\sum_{\mathrm{Lab}^{\mathrm{c2}}_Q(\tau)=1} \mathrm{occ}(\tau) \Big) \geq 2 \Big) \ . \end{aligned}$$

This ensures that we have exactly one vertex y labeled with the state 0 in the first component and exactly one vertex z labeled with the state 1 in the second component.

Furthermore, every first component of a state we assign to a vertex has to be equal to the distance of this vertex to y up to the threshold N, which can be proved by induction.

For example, the following tiles would be part of the wGA for $N = 20$ and $t \geq 4$ (we are omitting the a here):



Then every accepting run has the weight of the vertex z, i.e., the distance of this vertex to the state y up to the threshold N. Thus,

$$\begin{aligned}[\![\mathcal{A}]\!](G) &= \max_{\rho \text{ acc. run}} \mathrm{wt}_{\mathcal{A},G}(\rho) \\ &= \max_{y,z \in V} \mathrm{dist}_N(y,z) \ ,\end{aligned}$$

which is our desired property. Note again that, here, the distance sums over the edge-labels, whereas setting $B = \{1\}$ yields the unweighted setting and the *geodesic distance* to y up to the threshold N. $\diamond$

We extend the operation $+$ of our valuation monoid to series by means of point-wise definition, i.e.,

$$(S+T)(G) = S(G) + T(G) \text{ for each } G \in \mathrm{DG}_t(A,B) \ .$$

In the case of a semiring, we apply the same to the product, i.e.,

$$(S \odot T)(G) = S(G) \cdot T(G) \text{ for each } G \in \mathrm{DG}_t(A,B) \ .$$

Proposition 6.7. *The class of recognizable series is closed under* $+$.

Proof. Let $\mathcal{A} = (Q_\mathcal{A}, \Delta_\mathcal{A}, \mathrm{wt}_\mathcal{A}, \mathrm{Occ}_\mathcal{A}, r_\mathcal{A})$ and $\mathcal{B} = (Q_\mathcal{B}, \Delta_\mathcal{B}, \mathrm{wt}_\mathcal{B}, \mathrm{Occ}_\mathcal{B}, r_\mathcal{B})$ be weighted graph automata over the valuation monoid $\mathbb{D}$ and the alphabets A and B with $[\![\mathcal{A}]\!] = S_1$, $[\![\mathcal{B}]\!] = S_2$, and $Q_\mathcal{A} \cap Q_\mathcal{B} = \emptyset$. Using Lemma 6.6, we can assume that both automata have the same tile-radius r.

We construct the wGA $\mathcal{C} = (Q_\mathcal{C}, \Delta_\mathcal{C}, \mathrm{wt}_\mathcal{C}, \mathrm{Occ}_\mathcal{C}, r)$ with $[\![\mathcal{C}]\!] = S_1 + S_2$, as follows. We define $Q_\mathcal{C} = Q_\mathcal{A} \uplus Q_\mathcal{B}$ and $\Delta_\mathcal{C} = \Delta_\mathcal{A} \uplus \Delta_\mathcal{B}$. Now, the idea is to enforce $\mathcal{C}$ to tile a graph G either completely with tiles from $\mathcal{A}$ or completely with tiles from $\mathcal{B}$. If restricted to $r > 1$ and connected graphs, this follows directly from the choice of $\Delta_\mathcal{C}$ (which allows no "mix-tiles"). In the general case, we define $\mathrm{Occ}_\mathcal{C}$ in the following way to ensure this property:

$$\mathrm{Occ}_\mathcal{C} = (\mathrm{Occ}_\mathcal{A} \wedge \bigwedge_{\tau \in \Delta_\mathcal{B}} \mathrm{occ}(\tau) = 0) \vee (\mathrm{Occ}_\mathcal{B} \wedge \bigwedge_{\tau \in \Delta_\mathcal{A}} \mathrm{occ}(\tau) = 0) \ .$$

For $\tau \in \Delta_\mathcal{C}$, we set $\mathrm{wt}_\mathcal{C}(\tau) = \begin{cases} \mathrm{wt}_\mathcal{A}(\tau) & \text{, if } \tau \in \Delta_\mathcal{A} \\ \mathrm{wt}_\mathcal{B}(\tau) & \text{, if } \tau \in \Delta_\mathcal{B} \end{cases}$. Then every accepting run of $\mathcal{C}$ is exactly one accepting run in either $\mathcal{A}$ or $\mathcal{B}$, and vice versa. This run uses the respective weights, thus $[\![\mathcal{C}]\!](G) = [\![\mathcal{A}]\!](G) + [\![\mathcal{B}]\!](G) = (S_1 + S_2)(G)$. $\square$

Proposition 6.8. *Let* $\mathbb{D} = \mathbb{K}$ *be a commutative semiring. Then, the class of recognizable series is closed under* $\odot$.

Proof. Let $\mathcal{A} = (Q_\mathcal{A}, \Delta_\mathcal{A}, \mathrm{wt}_\mathcal{A}, \mathrm{Occ}_\mathcal{A}, r)$ be a wGA recognizing a series $S_1 : \mathrm{DG}_t(A,B) \to D$ and let $\mathcal{B} = (Q_\mathcal{B}, \Delta_\mathcal{B}, \mathrm{wt}_\mathcal{B}, \mathrm{Occ}_\mathcal{B}, r)$ be a wGA recognizing

$S_2 : \mathrm{DG}_t(A,B) \to D$. Using Lemma 6.6, we can assume that both automata have the same tile-radius r.

With a product construction, we construct a wGA $\mathcal{C} = (Q_\mathcal{C}, \Delta_\mathcal{C}, \mathrm{wt}_\mathcal{C}, \mathrm{Occ}_\mathcal{C}, r)$ with $[\![\mathcal{C}]\!] = S_1 \odot S_2$, as follows. We define $Q_\mathcal{C} = Q_\mathcal{A} \times Q_\mathcal{B}$. If τ is a tile over $A \times Q_\mathcal{C}$, we denote by $\tau_{|A\times Q_\mathcal{A}}$ the same tile over $A \times Q_\mathcal{A}$ where we forget the labels $Q_\mathcal{B}$. Similarly, we define $\tau_{|A\times Q_\mathcal{B}}$. We let $\Delta_\mathcal{C}$ consist of all tiles τ such that $\tau_{|A\times Q_\mathcal{A}} \in \Delta_\mathcal{A}$ and $\tau_{|A\times Q_\mathcal{B}} \in \Delta_\mathcal{B}$. For $\tau \in \Delta_\mathcal{C}$, we put $\mathrm{wt}_\mathcal{C}(\tau) = \mathrm{wt}_\mathcal{A}(\tau_{|A\times Q_\mathcal{A}}) \cdot \mathrm{wt}_\mathcal{B}(\tau_{|A\times Q_\mathcal{B}})$. Let $\tau_\mathcal{A} \in \Delta_\mathcal{A}$. We define $\tau_\mathcal{A}^{\mathcal{B}*}$ as the set of all tiles τ over $A \times Q_\mathcal{C}$ with $\tau_{|A\times Q_\mathcal{A}} = \tau_\mathcal{A}$. We use formula (6.1) to define $\mathrm{Occ}'_\mathcal{A}$ as the formula $\mathrm{Occ}_\mathcal{A}$ where we substitute every incidence of $\mathrm{occ}(\tau_\mathcal{A}) \geq n$ with $(\sum_{\tau' \in \tau_\mathcal{A}^{\mathcal{B}*}} \mathrm{occ}(\tau')) \geq n$. Analogously, we define $\mathrm{Occ}'_\mathcal{B}$. We set $\mathrm{Occ}_\mathcal{C} = \mathrm{Occ}'_\mathcal{A} \wedge \mathrm{Occ}'_\mathcal{B}$.

Hence, for every graph G, every accepting run ρ of $\mathcal{C}$ on G defines exactly one accepting run $\rho_{|Q_\mathcal{A}}$ of $\mathcal{A}$ and exactly one accepting run $\rho_{|Q_\mathcal{B}}$ of $\mathcal{B}$ on G (restricted to the respective state-labels), matching the respective occurrence constraints and weights. Using distributivity of our semiring, we get

$$\begin{aligned}
[\![\mathcal{C}]\!](G) &= \sum_{\rho \text{ acc. run of } \mathcal{C} \text{ on } G} \prod_{v \in V} \mathrm{wt}_{\mathcal{C},G,\rho}(v) \\
&= \sum_{\substack{\rho: V \to Q_\mathcal{C}, \text{ such that} \\ \rho_{|Q_\mathcal{A}} \text{ acc. run of } \mathcal{A} \text{ on } G \\ \rho_{|Q_\mathcal{B}} \text{ acc. run of } \mathcal{B} \text{ on } G}} \prod_{v \in V} \mathrm{wt}_{\mathcal{A},G,\rho_{|Q_\mathcal{A}}}(v) \cdot \mathrm{wt}_{\mathcal{B},G,\rho_{|Q_\mathcal{B}}}(v) \\
&= \sum_{\rho_{|Q_\mathcal{A}}} \prod_{v \in V} \mathrm{wt}_{\mathcal{A},G,\rho_{|Q_\mathcal{A}}}(v) \cdot \sum_{\rho_{|Q_\mathcal{B}}} \prod_{v \in V} \mathrm{wt}_{\mathcal{B},G,\rho_{|Q_\mathcal{B}}}(v) \\
&= [\![\mathcal{A}]\!](G) \cdot [\![\mathcal{B}]\!](G) \ .
\end{aligned}$$

Hence, $[\![\mathcal{C}]\!] = [\![\mathcal{A}]\!] \odot [\![\mathcal{B}]\!] = S_1 \odot S_2$. □

Note that in the proof above, the distributivity is crucial. For valuation monoids, which in general are not distributive, we later will need the following. Let $S : \mathrm{DG}_t(A,B) \to D$ and $L \subseteq \mathrm{DG}_t(A,B)$. We define the *restriction* $S \cap L : \mathrm{DG}_t(A,B) \to D$ by

$$(S \cap L)(G) = \begin{cases} S(G) & , \text{ if } G \in L \\ 0 & , \text{ otherwise} \end{cases} \ .$$

Proposition 6.9. *Let $S : \mathrm{DG}_t(A,B) \to D$ be a recognizable series and $L \subseteq DG_t(A,B)$ be a recognizable language. Then the following holds*

1. *If L is recognizable by a one-state GA, then $S \cap L$ is recognizable.*
2. *If $\mathbb{D}$ is idempotent, then $S \cap L$ is recognizable.*

Proof. We use product constructions of automata.

Let $\mathcal{A} = (Q_\mathcal{A}, \Delta_\mathcal{A}, \mathrm{wt}_\mathcal{A}, \mathrm{Occ}_\mathcal{A}, r)$ be the wGA recognizing $S : \mathrm{DG}_t(A, B) \to D$ and let $\mathcal{B} = (Q_\mathcal{B}, \Delta_\mathcal{B}, \mathrm{Occ}_\mathcal{B}, r)$ be the GA recognizing $L \subseteq \mathrm{DG}_t(A, B)$. We construct a wGA $\mathcal{C} = (Q_\mathcal{C}, \Delta_\mathcal{C}, \mathrm{wt}_\mathcal{C}, \mathrm{Occ}_\mathcal{C}, r)$ with $[\![\mathcal{C}]\!] = S \cap L$ as follows. We set $Q_\mathcal{C} = Q_\mathcal{A} \times Q_\mathcal{B}$. We define $\Delta_\mathcal{C}$ and $\mathrm{Occ}_\mathcal{C}$ as in the proof of Proposition 6.8.

As previously, for a tile τ over $A \times Q_\mathcal{C}$, we denote by $\tau_{|A\times Q_\mathcal{A}}$ the tile over $A \times Q_\mathcal{A}$ where we forget the labels $Q_\mathcal{B}$. Then, for $\tau \in \Delta_\mathcal{C}$, we set $\mathrm{wt}_\mathcal{C}(\tau) = \mathrm{wt}_\mathcal{A}(\tau_{|A\times Q_\mathcal{A}})$.

Hence, as before, for every graph G, every accepting run ρ of $\mathcal{C}$ on G defines exactly one accepting run $\rho_{|Q_\mathcal{A}}$ of $\mathcal{A}$ and exactly one accepting run $\rho_{|Q_\mathcal{B}}$ of $\mathcal{B}$ on G. Conversely, if $\mathcal{B}$ has no accepting run on G, $[\![\mathcal{C}]\!](G)$ will yield 0.

Furthermore, in the first case, we can construct $\mathcal{B}$ with one state, thus $\mathcal{B}$ has at most one accepting run on every graph. In the second case, $\mathcal{B}$ could have multiple accepting runs on a graph, but using the idempotence of $\mathbb{D}$, we only need to regard one accepting run. In both cases, we obtain

$$\begin{aligned}
[\![\mathcal{C}]\!](G) &= \sum_{\rho \in \mathrm{acc}_\mathcal{C}(G)} \mathrm{Val}(G_{\mathcal{C},\rho}) \\
&= \sum_{\substack{\rho : V \to Q_\mathcal{C}, \text{ such that} \\ \rho_{|Q_\mathcal{A}} \text{ acc. run of } \mathcal{A} \text{ on } G \\ \rho_{|Q_\mathcal{B}} \text{ acc. run of } \mathcal{B} \text{ on } G}} \mathrm{Val}(G_{\mathcal{C},\rho}) \\
&= \begin{cases} \sum_{\rho \in \mathrm{acc}_\mathcal{A}(G)} \mathrm{Val}(G_{\mathcal{A},\rho}) & \text{, if } \mathcal{B} \text{ has an accepting run on } G \\ 0 & \text{, otherwise} \end{cases} \\
&= (S \cap L)(G) \ .
\end{aligned}$$

Hence, $[\![\mathcal{C}]\!] = S \cap L$. □

In the following, we show that recognizable series are closed under projection. Let $h : A' \to A$ be a mapping between two alphabets. Then h defines naturally a relabeling of graphs from $\mathrm{DG}_t(A', B)$ into graphs from $\mathrm{DG}_t(A, B)$, also denoted by h. Let $S : \mathrm{DG}_t(A', B) \to D$ be a series. We define $h(S) : \mathrm{DG}_t(A, B) \to D$ by

$$h(S)(G) = \sum_{\substack{G' \in \mathrm{DG}_t(A',B) \\ h(G') = G}} S(G') \ . \tag{6.2}$$

Proposition 6.10. *Let $S : \mathrm{DG}_t(A', B) \to D$ be a recognizable series and $h : A' \to A$. Then $h(S) : \mathrm{DG}_t(A, B) \to D$ is recognizable.*

Proof. We use ideas of [DV12] in a modified form.

Let $\mathcal{A} = (Q_\mathcal{A}, \Delta_\mathcal{A}, \mathrm{wt}_\mathcal{A}, \mathrm{Occ}_\mathcal{A}, r)$ be a weighted graph automaton over A' and B with $[\![\mathcal{A}]\!] = S$. We construct the wGA $\mathcal{B} = (Q_\mathcal{B}, \Delta_\mathcal{B}, \mathrm{wt}_\mathcal{B}, \mathrm{Occ}_\mathcal{B}, r)$ over A and B with $[\![\mathcal{B}]\!] = h(S)$, as follows. Since h could map two symbols of A' to the same symbol of A, we have to keep track of the original labeling of our

graphs. Therefore, we set $Q_{\mathcal{B}} = A' \times Q_{\mathcal{A}}$. We define $\Delta_{\mathcal{B}}$ as the set of all tiles τ over $A \times A' \times Q_{\mathcal{A}}$ and B such that every vertex of τ has the form $(h(a'), a', q)$ with $a' \in A'$, $q \in Q$, and the respective tile where we forget the first label A, i.e., $\tau_{|A' \times Q_{\mathcal{A}}}$, is in $\Delta_{\mathcal{A}}$. For $\tau \in \Delta_{\mathcal{B}}$, we define $\mathrm{wt}_{\mathcal{B}}(\tau) = \mathrm{wt}_{\mathcal{A}}(\tau_{|A' \times Q_{\mathcal{A}}})$.

Let $\tau_{\mathcal{A}} \in \Delta_{\mathcal{A}}$. We define $\tau'_{\mathcal{A}} \in \Delta_{\mathcal{B}}$ as the respective tile where we substitute every vertex (a', q) with the vertex $(h(a'), a', q)$. We define $\mathrm{Occ}_{\mathcal{B}}$ as $\mathrm{Occ}_{\mathcal{A}}$ where we substitute every incidence of $\mathrm{occ}(\tau_{\mathcal{A}}) \geq n$ with $\mathrm{occ}(\tau'_{\mathcal{A}}) \geq n$. We note that we could also (analogously to the proof of Proposition 6.8) use a sum over the occurrences of all tiles matching the pattern described by $\tau_{\mathcal{A}}$, but following the definition of $\Delta_{\mathcal{B}}$, these are exactly the tiles with vertices of the form $(h(a'), a', q)$.

Now, every run of the wGA $\mathcal{B}$ on a graph $G \in \mathrm{DG}_t(A, B)$ tiles the graph such that every vertex of G is labeled with (a, a', q) such that a' is an element of the preimage of a under h. Such a run is accepting if and only if the respective run of $\mathcal{A}$ on G' (where G' is the preimage of G under h, defined by the elements $a' \in A$ of the tiles) is accepting. It follows that every accepting run of $\mathcal{B}$ on G defines exactly an accepting run of $\mathcal{A}$ on a preimage of G under h, and vice versa. Together with the weights of the tiles matching each other, we obtain

$$\begin{aligned}
[\![\mathcal{B}]\!](G) &= \sum_{\rho \in \mathrm{acc}_{\mathcal{B}}(G)} \mathrm{Val}(G_{\mathcal{B},\rho}) \\
&= \sum_{\substack{G' \in \mathrm{DG}_t(A',B) \\ h(G')=G}} \sum_{\rho \in \mathrm{acc}_{\mathcal{A}}(G')} \mathrm{Val}(G'_{\mathcal{A},\rho}) \\
&= \sum_{\substack{G' \in \mathrm{DG}_t(A',B) \\ h(G')=G}} [\![\mathcal{A}]\!](G') \\
&= h([\![\mathcal{A}]\!])(G) = h(S)(G) \ .
\end{aligned}$$

Hence, $[\![\mathcal{B}]\!] = h(S)$. □

6.2.2 Robustness of WGA: A Nivat Theorem

In this section, we establish a connection between unweighted recognizable languages and recognizable graph series over semirings. Note that a corresponding result for weighted automata on words (cf. [DKar]) makes crucial use of the possible determinization of every unweighted word automaton. Unfortunately, graph acceptors are not determinizable [Tho91]. To deal with this problem, we require either the underlying semiring to be idempotent or the considered languages to be recognizable by a one-state graph acceptor. For a similar distinction, see [DP14a].

In this section, let $\mathbb{D} = \mathbb{K} = (K, +, \cdot, 0, 1)$ be a semiring. As noted before, for every map $h : A' \to A$, we get a homomorphism $h : \mathrm{DG}_t(A', B) \to \mathrm{DG}_t(A, B)$.

For a series $S : \mathrm{DG}_t(A', B) \to K$, we get $h(S) : \mathrm{DG}_t(A, B) \to K$, using the sum over all pre-images as in formula (6.2).

Let $S : \mathrm{DG}_t(A', B) \to K$ and $L \subseteq \mathrm{DG}_t(A', B)$. As before, we consider $S \cap L : \mathrm{DG}_t(A, B) \to K$ by letting $(S \cap L)(G) = S(G)$ if $G \in L$ and $(S \cap L)(G) = 0$, otherwise.

Let $g : A' \to K$ be a map. Let $G \in \mathrm{DG}_t(A', B)$ and let $\mathrm{Lab}_G(v) \in A'$ be the label of a vertex v of G. We define the map $\mathrm{prod} \circ g : \mathrm{DG}_t(A', B) \to K$ by $(\mathrm{prod} \circ g)(G) = \prod_{v \in V} g(\mathrm{Lab}_G(v))$. So, $\mathrm{prod} \circ g : \mathrm{DG}_t(A', B) \to K$ is a very particular series obtained by assigning, for a graph $G \in \mathrm{DG}_t(A', B)$, to each vertex a weight (depending only on its label) and then multiplying all these weights.

Let $\mathcal{N}_t(A, B, \mathbb{K})$ comprise all series $S : \mathrm{DG}_t(A, B) \to K$ for which there exist an alphabet A', a map $g : A' \to K$, a map $h : A' \to A$, and a recognizable language $L \subseteq \mathrm{DG}_t(A', B)$ such that $S = h((\mathrm{prod} \circ g) \cap L)$. We denote by $\mathcal{N}_t^{\mathrm{one}}(A, B, \mathbb{K})$ the set of series defined similarly but with a language L which is recognizable by a one-state GA. Trivially, $\mathcal{N}_t^{\mathrm{one}}(A, B, \mathbb{K}) \subseteq \mathcal{N}_t(A, B, \mathbb{K})$.

Proposition 6.11. *Let $S : \mathrm{DG}_t(A, B) \to K$ be a series. If S is recognizable, then S is in $\mathcal{N}_t^{one}(A, B, \mathbb{K})$.*

Proof. We follow some ideas of [DKar] and [DP14a]. However, our proof has two major differences to these approaches. Firstly, we reduce the blowup of the alphabet A' as follows. In the previous works, the extended alphabet was set to all possible edges (which would translate in our context to all possible tiles) together with all possible weights. Here, we only add one state and one weight at every vertex of our graphs. In the case of an idempotent semiring, we could further reduce this blowup to only adding possible weights. Secondly, in contrast to the previous approaches, we have to deal with the occurrence constraint of our weighted graph automata, using formula (6.1).

Let $\mathcal{A} = (Q, \Delta, \mathrm{wt}, \mathrm{Occ}, r)$ be a wGA over A, B, and $\mathbb{K}$ with $[\![\mathcal{A}]\!] = S$. We denote by WT the set of all weights used by $\mathcal{A}$, i.e. $WT = \mathrm{wt}(\Delta) \subseteq K$.

We set $A' = A \times Q \times WT$ as the extended alphabet. Let h be the projection of A' to A, let g be the projection of A' to WT, and let f be the projection of A' to $A \times Q$.

Let $L \subseteq \mathrm{DG}_t(A', B)$ be the language consisting of all graphs G' over the extended alphabet such that assigning to every vertex v' of G' the second component of $f(v')$ defines an accepting run of $\mathcal{A}$ on $h(G')$ and the added weights are consistent with the weight function wt of $\mathcal{A}$. We construct the (unweighted) graph acceptor $\mathcal{A}' = (Q', \Delta', \mathrm{Occ}', r)$ over A' and B, accepting L, as follows. We set $Q' = \{q_0\}$. Let τ' be a tile over $A' \times Q'$ and B. Since Q' consists of only one state, we omit Q' in the following and consider every τ' as a tile over A' and B. We define $\mathrm{Lab}^{\mathrm{c}}_{WT}(\tau') = \mathrm{Lab}^{\mathrm{c}}(g(\tau'))$ as the WT-label at the center of τ', and define the set of tiles of $\mathcal{A}'$ as

$$\Delta' = \{\tau' \mid f(\tau') \in \Delta \wedge \mathrm{Lab}^{\mathrm{c}}_{WT}(\tau') = \mathrm{wt}(f(\tau'))\} \ .$$

We build up the occurrence constraint Occ' inductively in the same way as Occ, where for every tile τ of Δ every atomic incidence of $\mathrm{occ}(\tau) \geq n$ is replaced by $\big(\sum_{f(\tau')=\tau} \mathrm{occ}(\tau')\big) \geq n$, see formula (6.1).

Hence, $\mathcal{A}'$ has an accepting run on a graph $G' \in \mathrm{DG}_t(A,B)$ if and only if $f(G')$ defines an accepting run of $\mathcal{A}$ on $h(G')$, using the corresponding weights $g(G')$ at the center of the tiles. Therefore,

$$\begin{aligned}
h((\mathrm{prod} \circ g) \cap L)(G) &= \sum_{\substack{G' \in \mathrm{DG}_t(A',B) \\ h(G')=G}} ((\mathrm{prod} \circ g) \cap L)(G') \\
&= \sum_{\substack{G' \in L(\mathcal{A}') \\ h(G')=G}} \mathrm{prod}(g(G')) \\
&= \sum_{\substack{f(G') \text{ acc. run of } \mathcal{A} \text{ on } h(G') \\ h(G')=G}} \prod_{v \in V} \mathrm{Lab}^{\mathrm{c}}_{WT}(\mathrm{sph}^r(f(G'),v)) \\
&= \sum_{\rho \text{ acc. run of } \mathcal{A} \text{ on } G} \prod_{v \in V} \mathrm{wt}_{\mathcal{A},G,\rho}(v) \\
&= [\![\mathcal{A}]\!](G) = S(G) \ .
\end{aligned}$$

Hence, $S = h((\mathrm{prod} \circ g) \cap L)$, thus $S \in \mathcal{N}_t^{\mathrm{one}}(A,B,\mathbb{K})$. □

Using this proposition and closure properties of series, we get the following Nivat theorem for weighted graph automata.

Theorem 6.12. *Let $\mathbb{K}$ be a commutative semiring and $S : \mathrm{DG}_t(A,B) \to K$ be a series. Then S is recognizable if and only if $S \in \mathcal{N}_t^{one}(A,B,\mathbb{K})$. If $\mathbb{K}$ is idempotent, then S is recognizable if and only if $S \in \mathcal{N}_t(A,B,\mathbb{K})$.*

Proof. The "only if"-part of both statements is immediate by Proposition 6.11.

For the converse, let A' be an alphabet, $g : A' \to K$, $h : A' \to A$, $L \subseteq \mathrm{DG}_t(A',B)$ be a recognizable language, and $S = h((\mathrm{prod} \circ g) \cap L)$.

In the first case, $S \in \mathcal{N}_t^{\mathrm{one}}(A,B,\mathbb{K})$, we can assume that $L \subseteq \mathrm{DG}_t(A',B)$ is recognizable by a one-state GA $\mathcal{A}$. Then we construct a wGA $\mathcal{B}$ which uses the same states and tiles as $\mathcal{A}$ and assigns to every tile of $\mathcal{A}$ weight 1. Since every one-state GA has for every graph G at most one accepting run, the behavior of $\mathcal{B}$ is $[\![\mathcal{B}]\!] = \mathbb{1}_{L(\mathcal{A})} = \mathbb{1}_L$.

In the second case, $\mathbb{K}$ is idempotent and L is recognizable by a general GA. Then we construct the wGA $\mathcal{B}$ in a similar way, but now, $\mathcal{B}$ could have multiple runs for an input graph G. However, the idempotence of $\mathbb{K}$ again ensures that $[\![\mathcal{B}]\!] = \mathbb{1}_L$.

It is easy to construct a wGA $\mathcal{C}$ which simulates $\mathrm{prod} \circ g$. Then Proposition 6.9 shows that $(\mathrm{prod} \circ g) \cap L$ is recognizable. Now, Proposition 6.10 yields the result. □

The following examples show that in the general setting there exist series in $\mathcal{N}_t(A, B, \mathbb{K})$ which are not recognizable. For this purpose, we say that a GA $\mathcal{A}$ is *unambiguous* if for every graph G, $\mathcal{A}$ has at most one accepting run on G. We call a graph language L *unambiguously recognizable* if there exists an unambiguous GA accepting L.

Example 6.7. Let L be the class of all graphs of $\mathrm{DG}_t(A, B)$ which are not connected. Then we claim that L is recognizable but not unambiguously recognizable. For the first claim, it suffices to construct a GA $\mathcal{A}$ using the idea of Example 6.1. That is, we define $\mathcal{A}$ with tile size 1, consisting of two states q_1 and q_2, consisting of all tiles which are completely labeled with q_1 or completely labeled with q_2, and with an occurrence constraint checking that both q_1 and q_2 are occurring.

For the second claim, we assume there exists an unambiguous GA $\mathcal{A}$ accepting L. Let $a \in A$. Let r be the tile size of $\mathcal{A}$. Consider a graph G consisting of two unconnected and isomorphic circles of length $2r + 2$ labeled with a at every vertex. We refer with $v_1, ..., v_{2r+2}$ to the vertices of the first circle and with $w_1, ..., w_{2r+2}$ to the vertices of the second circle. Then, by assumption, $\mathcal{A}$ has exactly one accepting run ρ on G. If both circles are not labeled identical by ρ, i.e. if not $\rho(v_i) = \rho(w_i)$ for all $i = 1, ..., 2r + 2$, then we would get a different accepting run of G by interchanging the complete labeling of the two circles (this is possible because both circles are identical, therefore both runs would use the same tiles overall), which would be a contradiction to our assumption. Therefore, let $q_i = \rho(v_i) = \rho(w_i)$. Hence, G_ρ consists of two circles labeled with $(a, q_1), (a, q_2), ..., (a, q_{2r+2})$. Then, we claim that the connected circle consisting of $v_1, ..., v_{2r+2}, w_1, ..., w_{2r+2}$ is also accepted by A. Indeed, setting again $q_i = \rho'(v_i) = \rho'(w_i)$, we get an accepting run ρ' which uses the exact same tiles as the two unconnected circles. Therefore, our assumption that L is unambiguously recognizable was wrong. ◇

Example 6.8. We follow ideas of [DP14a] to construct a series in $\mathcal{N}_t(A, B, \mathbb{K})$ that is not recognizable, as follows. We set $\mathbb{K} = (\mathbb{N}, +, \cdot, 0, 1)$ and $A' = A$. Letting h be the identity-function and $g \equiv 1$ be the constant function to 1, we get $h((\mathrm{prod} \circ g) \cap L) = \mathbb{1}_L$. Therefore, it suffices to show that there exists a recognizable language L such that $\mathbb{1}_L$ is not recognizable by a wGA over $\mathbb{N}$.

Let L be a recognizable but not unambiguously recognizable language. We claim $\mathbb{1}_L$ is not recognizable by a wGA over A, B, and $\mathbb{N}$. Indeed, assume $\mathbb{1}_L$ is recognizable by a wGA $\mathcal{A}$. Hence, $[\![\mathcal{A}]\!](G) = \sum_{\text{acc. run } \rho} \prod_{v \in V} \mathrm{wt}_{\mathcal{A},G,\rho}(v) = \mathbb{1}_L$. Since we can only use weights of $\mathbb{N}$, this cannot be true if there exists either an accepting run with weight greater than 1 or more than one accepting run with weight 1. Essentially, we only need the weights 0 and 1, and $\mathcal{A}$ can be transformed into an unweighted GA $\mathcal{A}$ which either has no accepting runs or exactly one accepting run (with weight 1). But then $\mathcal{A}'$ is an unambiguous automaton accepting L – which is a contradiction.

The existence of a recognizable but not unambiguously recognizable language is proven in Example 6.7. There also exist examples of recognizable but not unambiguously recognizable languages of connected graphs. For example, over the class of pictures the existence of such a language was shown in [AGMR06], however, there the proof is more complex. ◇

In the Examples 6.7 and 6.8, we proved the following.

Proposition 6.13. *1. The class of unambiguously recognizable languages is a proper subclass of all recognizable languages.*

2. There exists a non idempotent semiring $\mathbb{K}$ and a recognizable language L such that $\mathbb{1}_L$ is not recognizable by a wGA over $\mathbb{K}$.

6.2.3 Characterization of WGA: A Büchi-Theorem

In the following, we introduce a weighted MSO-logic for graphs as a special case of the weighted logic on relational structures in Section 3.3. Utilizing the previous closure results, we will then show that every sentence of this logic can be simulated by a weighted graph automaton. Finally, also showing the converse, we get our main result of this chapter, a Büchi-like characterization of weighted graph automata. As starting point, we give the syntax of MSO($\mathbb{D}$) for graphs utilizing the 'if..then..else'-operator $\beta?\varphi_1 : \varphi_2$.

Definition 6.14. We define the logic MSO($\mathbb{D}, \mathrm{DG}_t(A, B)$), short MSO($\mathbb{D}$), as

$$\beta ::= \mathrm{Lab}_a(x) \mid E_b(x,y) \mid x = y \mid x \in X \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta \mid \exists X.\beta$$
$$\varphi ::= d \mid \varphi \oplus \varphi \mid \beta?\varphi : \varphi \mid \bigoplus_x \varphi \mid \bigoplus_X \varphi \mid \mathrm{Val}_x\, \varphi$$

where $d \in D$; x, y are first-order variables; and X is a second order variable.

In the special case of a semiring or more generally, a product valuation monoid, we define MSO($\mathbb{PD}$) for graphs as follows, cf. Section 3.3.

Definition 6.15. Given a product valuation monoid $\mathbb{PD}$, we define the weighted logic MSO($\mathbb{PD}, \mathrm{DG}_t(A, B)$), short MSO($\mathbb{PD}$), as

$$\beta ::= \mathrm{Lab}_a(x) \mid E_b(x,y) \mid x = y \mid x \in X \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta \mid \exists X.\beta$$
$$\varphi ::= \beta \mid d \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi \mid \bigoplus_x \varphi \mid \bigoplus_X \varphi \mid \mathrm{Val}_x\, \varphi$$

where $d \in D$; x, y are first-order variables; and X is a second order variable. If $\mathbb{PD}$ is a semiring, Val can be written as $\prod_x \varphi$.

As noted in Section 3.3, the semantics following below will yield that $\beta?\varphi : \psi$ is expressible by $(\beta \otimes \varphi) \oplus (\neg\beta \otimes \psi)$. Thus, given a product valuation monoid $\mathbb{PD}$ with the underlying valuation monoid $\mathbb{D}$, we can consider every MSO($\mathbb{D}$)-sentence also as an MSO($\mathbb{PD}$)-sentence. Later, we will see that MSO($\mathbb{D}$) is

sufficient to characterize runs of weighted graph automata. However, the second approach enriches the structure by the product and gives a direct connection to previous works [DG07, DM12, DD15].

We define *valid* graphs $(G,\gamma) \in \mathrm{DG}_t(A_\mathcal{V}, B)$, with an *assignment* γ of variables $\mathcal{V}$ containing $\mathrm{free}(\varphi)$ as in Section 2.3. Whether a graph is valid can be checked in Occ by a formula checking for every FO-variable x that the sum of all occurrences of tiles labeled with $x = 1$ at their center is 1. Therefore, being valid is a graph property which can be checked by a graph acceptor.

We define the *semantics* of $\varphi \in \mathrm{MSO}(\mathbb{D})$, resp. $\varphi \in \mathrm{MSO}(\mathbb{PD})$ (cf. Section 3.3), as a function $[\![\varphi]\!]_\mathcal{V} : \mathrm{DG}_t(A_\mathcal{V}, B) \to D$ for all valid (G,γ) inductively as seen in Figure 6.2. For not valid graphs G', we set $[\![\varphi]\!]_\mathcal{V}(G') = 0$. Note that

$$
\begin{array}{lcl}
[\![d]\!]_\mathcal{V}(G,\gamma) & = & d \quad \text{for all } d \in D \\
[\![\varphi \oplus \psi]\!]_\mathcal{V}(G,\gamma) & = & [\![\varphi]\!]_\mathcal{V}(G,\gamma) + [\![\psi]\!]_\mathcal{V}(G,\gamma) \\
[\![\beta ? \varphi : \psi]\!]_\mathcal{V}(G,\gamma) & = & \begin{cases} [\![\varphi]\!]_\mathcal{V}(G,\gamma) & \text{, if } (G,\gamma) \models \beta \\ [\![\psi]\!]_\mathcal{V}(G,\gamma) & \text{, otherwise} \end{cases} \\
[\![\beta]\!]_\mathcal{V}(G,\gamma) & = & \begin{cases} 1 & \text{, if } (G,\gamma) \models \beta \\ 0 & \text{, otherwise} \end{cases} \\
[\![\varphi \otimes \psi]\!]_\mathcal{V}(G,\gamma) & = & [\![\varphi]\!]_\mathcal{V}(G,\gamma) \diamond [\![\psi]\!]_\mathcal{V}(G,\gamma) \\
[\![\bigoplus_x \varphi]\!]_\mathcal{V}(G,\gamma) & = & \sum\limits_{v \in V} [\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(G, \gamma[x \to v]) \\
[\![\bigoplus_X \varphi]\!]_\mathcal{V}(G,\gamma) & = & \sum\limits_{I \subseteq V} [\![\varphi]\!]_{\mathcal{V}\cup\{X\}}(G, \gamma[X \to I]) \\
[\![\mathrm{Val}_x \varphi]\!]_\mathcal{V}(G,\gamma) & = & \mathrm{Val}((G,\gamma)_\varphi), \text{ where } (G,\gamma)_\varphi \text{ is the graph } (G,\gamma) \text{ s.t.} \\
 & & \text{every vertex } v \text{ is labeled with } [\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(G, \gamma[x \to v])
\end{array}
$$

Figure 6.2: Semantics of $\mathrm{MSO}(\mathbb{D})$ and $\mathrm{MSO}(\mathbb{PD})$ for graphs.

for the Boolean semiring $\mathbb{B}$, the unweighted MSO is expressively equivalent to $\mathrm{MSO}(\mathbb{B})$.

By Lemma 3.5, we have that $[\![\varphi]\!]_\mathcal{V}(G,\gamma) = [\![\varphi]\!](G, \gamma|_{\mathrm{free}(\varphi)})$ for each $(G,\gamma) \in \mathrm{DG}_t(A_\mathcal{V}, B)$. Then, the following lemma shows that a series remains recognizable if we add (superfluous) variables.

Lemma 6.16. *Let $\varphi \in \mathrm{MSO}(\mathbb{D})$ or $\varphi \in \mathrm{MSO}(\mathbb{PD})$ and $\mathcal{V}$ be a finite set of variables with $\mathcal{V} \supseteq \mathrm{free}(\varphi)$. If $[\![\varphi]\!]$ is recognizable, then $[\![\varphi]\!]_\mathcal{V}$ is recognizable.*

Proof. This is proved analogously to Proposition 3.3 of [DG07] as follows. Let $[\![\varphi]\!]$ be recognizable by a wGA $\mathcal{A}$ over $A_{\mathrm{free}(\varphi)}$. Then we can construct a wGA $\mathcal{A}'$ over $A_\mathcal{V}$, which ignores the superfluous variables using formula (6.1). Furthermore, in the occurrence constraint $\mathcal{A}'$ can check that we only read valid graphs. Then $[\![\mathcal{A}']\!] = [\![\varphi]\!]_\mathcal{V}$. □

We note that, in contrast to all previous papers of the literature on weighted logic, in general, the converse of Lemma 6.16 is not true as shown by the subsequent example together with results from Section 6.2.4. If we restrict ourselves to pointed graphs or to an idempotent valuation monoid, we can show that $[\![\varphi]\!]$ is recognizable if and only if $[\![\varphi]\!]_{\mathcal{V}}$ is recognizable. However, in the following proofs of this chapter, we do not need the converse of the implication of Lemma 6.16.

Example 6.9. We provide a counterexample for the converse of Lemma 6.16, as follows.

Let A and B be two arbitrary alphabets and let $\mathbb{D}$ be the rational numbers $\mathbb{D} = (\mathbb{Q}, +, \cdot, 0, 1)$. We set $\varphi = \frac{1}{2}$ and $\mathcal{V} = \{x\}$. Then $\mathrm{free}(\varphi) = \emptyset$ and $[\![\varphi]\!](G) = [\![\varphi]\!]_{\mathcal{V}}(G, \gamma) = \frac{1}{2}$ for each $G \in \mathrm{DG}_t(A, B)$ and for each valid $(G, \gamma) \in \mathrm{DG}_t(A_{\mathcal{V}}, B)$.

We construct the wGA $\mathcal{A} = (\{q_0\}, \Delta, \mathrm{wt}, \mathrm{Occ}, 0)$ over $A_{\mathcal{V}}$, B, and $\mathbb{D}$ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]_{\mathcal{V}}$, as follows. We set $\Delta = \{\tau \mid \tau \text{ is a 0-tile}\}$. We denote by τ^* all tiles τ which are labeled at the center with $(a, 1, q_0)$ for some $a \in A$, and define

$$\mathrm{Occ} = \big(\sum_{\tau \in \tau^*} \mathrm{occ}(\tau) \big) \geq 1 \wedge \neg \big(\big(\sum_{\tau \in \tau^*} \mathrm{occ}(\tau) \big) \geq 2 \big) \ .$$

We define $\mathrm{wt}(\tau) = \frac{1}{2}$ if $\tau \in \tau^*$, and $\mathrm{wt}(\tau) = 1$, otherwise. Hence, $[\![\mathcal{A}]\!](G, \gamma) = \frac{1}{2}$ for each valid $(G, \gamma) \in \mathrm{DG}_t(A_{\mathcal{V}}, B)$, and $[\![\mathcal{A}]\!](G, \gamma) = 0$, otherwise. Thus, $[\![\mathcal{A}]\!] = [\![\varphi]\!]_{\mathcal{V}}$.

It remains to show that $[\![\varphi]\!]$ is not recognizable. This follows from the fact that $\mathbb{Q}$ is not regular, which is proven in the following Section 6.2.4[2]. ◇

Now, we show that recognizable series are closed under $\bigoplus_x$- and $\bigoplus_X$-quantification (in previous papers called the weighted existential quantification).

Lemma 6.17. *Let $[\![\varphi]\!]$ be recognizable. Then $[\![\bigoplus_x \varphi]\!]$ and $[\![\bigoplus_X \varphi]\!]$ are recognizable by a wGA.*

Proof. This is a direct application of Proposition 6.10, cf. [DG07] and proven analogously to Lemma 4.21 as follows.

Let $\mathcal{V} = \mathrm{free}(\bigoplus_X \varphi)$. We define $\pi : \mathrm{DG}_t(A_{\mathcal{V} \cup \{X\}}, B) \to \mathrm{DG}_t(A_{\mathcal{V}}, B)$ by $\pi(G, \gamma) = (G, \gamma|_{\mathcal{V}})$ for any $(G, \gamma) \in \mathrm{DG}_t(A_{\mathcal{V} \cup \{X\}}, B)$. Then for $(G, \gamma) \in \mathrm{DG}_t(A_{\mathcal{V}}, B)$, the following holds

$$\begin{aligned}
[\![\bigoplus_X \varphi]\!](G, \gamma) &= \sum_{I \subseteq V} [\![\varphi]\!]_{\mathcal{V} \cup \{X\}}(G, \gamma[X \to I]) \\
&= \sum_{\substack{(G, \gamma') \in \mathrm{DG}_t(A_{\mathcal{V} \cup \{X\}}, B) \\ \pi(G, \gamma') = (G, \gamma)}} [\![\varphi]\!]_{\mathcal{V} \cup \{X\}}(G, \gamma') \\
&= \pi([\![\varphi]\!]_{\mathcal{V} \cup \{X\}})(G, \gamma) \ .
\end{aligned}$$

[2] as this combinatorial proof stands a bit apart from the other results of this section, or in other words, the section "is too narrow to contain" the full proof.

By Lemma 6.16, $[\![\varphi]\!]_{\mathcal{V}\cup\{X\}}$ is recognizable because $\mathrm{free}(\varphi) \subseteq \mathcal{V} \cup \{X\}$. Then $[\![\bigoplus_X \varphi]\!]$ is recognizable by Proposition 6.10. The operator $\bigoplus_x$ is dealt with analogously. □

The interesting case is the Val_x-quantification (which was in previous works called the weighted universal quantification [DG07]). Similarly to [DG07], our unrestricted logic is strictly more powerful than our automata model. Therefore, inspired by Gastin and Monmege [GM15], we introduce the following fragment.

We call a formula $\varphi \in \mathrm{MSO}(\mathbb{D})$ *almost FO-boolean* if φ is built up inductively from the grammar

$$\varphi ::= d \mid \beta ? d : \varphi$$

where $d \in D$ and β is an unweighted FO-formula. For product valuation monoids, we define all *almost FO-boolean* formulas $\varphi \in \mathrm{MSO}(\mathbb{PD})$, as in Section 3.3 (cf. also [DG07] and [DM12]), to be built up inductively from the following grammar

$$\varphi ::= d \mid \beta \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi$$

where $d \in D$ and β is an unweighted FO-formula.

In the following, using techniques inspired by [DM12], [GM15], and Section 4.4, we show that both fragments of almost FO-boolean formulas are equivalent to all formulas φ such that $[\![\varphi]\!]$ is an *FO-step function*, i.e., it takes only finitely many values and for each value its preimage is FO-definable. This also explains why we overload the notion of 'almost FO-boolean formulas'.

More precisely, denoting the constant series $d(G) = d$ for all $G \in DG_t(A, B)$ also with d, we call a series S an *FO-step function* if

$$S = \sum_{i=1}^{k} d_i \cap L_i \ ,$$

where $k \in \mathbb{N}$, $d_i \in D$, L_i are languages definable by an unweighted FO-formula, and $(L_i)_{i=1\ldots k}$ form a partition of $\mathrm{DG}_t(A, B)$; so $S(G) = d_i$ iff $G \in L_i$, for each $i \in \{1, ..., k\}$. In the following, we write $S = \sum_{i=1}^{k} d_i \mathbb{1}_{L_i}$ for such a given FO-step function.

Since every FO-definable language is definable by a one-state GA (Theorem 6.2), which intersected with recognizable series yield recognizable series (Proposition 6.9), which are closed under sum (Lemma 5.21), we get the following intermediate result.

Corollary 6.18. *Let $\mathbb{D}$ be a regular valuation monoid. Then every FO-step function is recognizable by a wGA.*

Lemma 6.19. *1. Let $\varphi \in \mathrm{MSO}(\mathbb{D})$ be an almost FO-boolean formula, then $[\![\varphi]\!]$ is an FO-step function.*

2. *Let* $\varphi \in \mathrm{MSO}(\mathbb{PD})$ *be an almost FO-boolean formula, then* $[\![\varphi]\!]$ *is an FO-step function.*
3. *For every FO-step function* $S : \mathrm{DG}_t(A, B) \to D$*, there exist almost FO-boolean sentences* $\varphi_1 \in \mathrm{MSO}(\mathbb{D})$ *and* $\varphi_2 \in \mathrm{MSO}(\mathbb{PD})$ *with* $S = [\![\varphi_1]\!] = [\![\varphi_2]\!]$*.*

Proof. 1. We use induction on the structure of an almost FO-boolean formula. If $\varphi = d \in D$, then $[\![\varphi]\!] = d = d \cap \mathrm{DG}_t(A, B)$ is an FO-step function. If φ is FO-boolean, we can interpret φ as unweighted formula with $[\![\varphi]\!] = \mathbb{1}_{L(\varphi)}$. Then $L(\varphi)$ is a recognizable language by Theorem 6.2. Hence, $[\![\varphi]\!]$ is an FO-step function. If $\varphi = \beta?d : \theta$ and $[\![\theta]\!] = \sum_{i=1}^{k} d_i \mathbb{1}_{L_i}$ is an FO-step function, then $[\![\varphi]\!] = d \cap L(\beta) + \theta \cap L(\neg\beta) = d\mathbb{1}_{L(\beta)} + \sum_{i=1}^{k} d_i \mathbb{1}_{L_i \cap L(\neg\beta)}$ is an FO-step function.

For 2., we have to additionally show closure under $\otimes$ and $\oplus$ as follows. We assume that $[\![\varphi]\!] = \sum_{i=1}^{k} d_i \mathbb{1}_{L_i}$ and $[\![\psi]\!] = \sum_{j=1}^{\ell} d'_j \mathbb{1}_{L'_j}$ are FO-step functions. Then $[\![\varphi \oplus \psi]\!]$ and $[\![\psi \otimes \varphi]\!]$ are FO-step functions since we have

$$[\![\varphi]\!] + [\![\psi]\!] = \sum_{i=1}^{k} \sum_{j=1}^{\ell} (d_i + d'_j) \mathbb{1}_{L_i \cap L'_j} \ ,$$

$$[\![\varphi]\!] \diamond [\![\psi]\!] = \sum_{i=1}^{k} \sum_{j=1}^{\ell} (d_i \diamond d'_j) \mathbb{1}_{L_i \cap L'_j} \ ,$$

where $(L_i \cap L'_j)$ are also FO-definable and form a partition of $\mathrm{DG}_t(A, B)$.

3. Let $S : \mathrm{DG}_t(A, B) \to D$ be an FO-step function, so $S = \sum_{i=1}^{k} d_i \mathbb{1}_{L_i}$ with $d_i \in D$, (L_i) form a partition of $\mathrm{DG}_t(A, B)$, and for each $i \in \{1, ..., k\}$, let β_i be an FO-sentence such that $L(\beta_i) = L_i$. Then, the weighted semantics of β_i yield $[\![\beta_i]\!] = \mathbb{1}_{L_i}$. Hence, the almost FO-boolean sentence $\varphi \in \mathrm{MSO}(\mathbb{D})$ defined as

$$\varphi_1 = \beta_1?d_1 : (\beta_2?d_2 : ... : (\beta_k?d_k : 0)...)$$

satisfies $[\![\varphi_1]\!] = S$, since (L_i) form a partition.

For $\mathrm{MSO}(\mathbb{PD})$, we set

$$\varphi_2 = \bigoplus_{i=1}^{k} (d_i \otimes \beta_i)$$

which evaluates to $\sum_{i=1}^{k} (d_i \diamond \mathbb{1}_{L_i})$. Since for all i, $d_i \diamond \mathbb{1}_{L_i} = d_i \cap L_i$, we get $[\![\varphi_2]\!] = S$. □

Proposition 6.20. *Let* $\varphi \in \mathrm{MSO}(\mathbb{D})$ *or* $\varphi \in \mathrm{MSO}(\mathbb{PD})$ *be an almost FO-boolean formula. Then* $[\![\mathrm{Val}_x \varphi]\!]$ *is recognizable.*

Proof. We use ideas of [DG07] and the proofs of Lemma 4.22 and Proposition 5.24. However, we have to adapt to the fact that graph acceptors are not

determinizable, and we have to take special care of the occurrence constraint of the graph acceptors.

The main structure of the proof is the following: Since φ is almost FO-boolean, we can apply Lemma 6.19 to write it as FO-step function $[\![\varphi]\!] = \sum_{i=1}^{m} d_i \mathbb{1}_{L_i}$, where the L_i form a partition of all graphs. Then, we can encode the information in which language L_i a given graph falls into an FO-formula $\tilde{L}$. Using Theorem 6.2 yields a one-state GA $\tilde{\mathcal{A}}$ with $L(\tilde{\mathcal{A}}) = \tilde{L}$. We 'copy' this graph acceptor m-times and add weights to every tile depending on the state-label at its center, resulting in a wGA $\mathcal{A}$. Then, we can show that $[\![\mathcal{A}]\!] = [\![\mathrm{Val}_x\, \varphi]\!]$.

More detailed, let $\mathcal{V} = \mathrm{free}(\mathrm{Val}_x\, \varphi)$ and $\mathcal{W} = \mathcal{V} \cup \{x\}$. By Lemma 6.16 and Lemma 6.19, we have that $[\![\varphi]\!] = \sum_{i=1}^{m} d_i \mathbb{1}_{L_i}$, with FO-definable languages L_i forming a partition of $\mathrm{DG}_t(A_{\mathcal{W}}, B)$. Thus, there exist FO-formulas φ_j with $L(\varphi_j) = L_j$ for every $j \in \{1, ..., m\}$.

We use a modified form of the encoding of [DG07]: We set $\tilde{A}_{\mathcal{V}} = A_{\mathcal{V}} \times \{1, ..., m\}$. We write graphs over $\tilde{A}_{\mathcal{V}}$ and B as (G, g, γ) where γ is an assignment of free variables of $\mathrm{Val}_x\, \varphi$ and $g : V \to \{1, ..., m\}$, where V is set of vertices of (G, g, γ). We define $\tilde{L}$ as the language of all valid (G, g, γ) with the following property. For all $v \in V, j \in \{1, ...m\}$, and $g(v) = j$, we have $(G, \gamma[x \to v] \in L_j$.

Using the FO-formulas φ_j, it can be proved analogously to [DG07] and [Fic11] that $\tilde{L}$ is FO-definable. By Theorem 6.2, there exists a one-state graph acceptor $\tilde{\mathcal{A}} = (\{q_0\}, \tilde{\Delta}, \tilde{\mathrm{Occ}}, r)$ over $\tilde{A}_{\mathcal{V}}$ and B with $L(\tilde{\mathcal{A}}) = \tilde{L}$.

Now, we define the wGA $\mathcal{A} = (Q, \Delta, \mathrm{wt}, \mathrm{Occ}, r)$ over $A_{\mathcal{V}}$ and B with $[\![\mathcal{A}]\!] = [\![\mathrm{Val}_x\, \varphi]\!]$, as follows. We set $Q = \{q_1, ..., q_m\}$. Then every tile $\tilde{\tau}$ of $\tilde{\mathcal{A}}$ is in $\mathrm{DG}_t(A_{\mathcal{V}} \times \{1, ..., m\} \times \{q_0\}, B))$, and every tile τ of $\mathcal{A}$ will be in $\mathrm{DG}_t(A_{\mathcal{V}} \times \{q_1, ..., q_m\}, B))$. It is easy to see that there exists a natural bijection $\mathfrak{h}$ between these two sets. We define $\Delta = \mathfrak{h}(\tilde{\Delta})$ and $\mathrm{Occ} = \mathfrak{h}(\tilde{\mathrm{Occ}})$, where $\mathfrak{h}(\tilde{\mathrm{Occ}})$ is built up inductively in the same way as $\tilde{\mathrm{Occ}}$, and every atomic incidence of $\mathrm{occ}(\tilde{\tau}) \geq n$ is replaced by $\mathrm{occ}(\mathfrak{h}(\tilde{\tau})) \geq n$. If $\tau \in \Delta$ is a tile around v where v is labeled with q_j at the second component, we define the weight assignment of τ as $\mathrm{wt}(\tau) = d_j$.

Note that for every valid graph (G, γ), there exists exactly one graph $(G, g, \gamma) \in \tilde{L}$, due to (L_i) forming a partition. Therefore, $\mathcal{A}$ has at most one accepting run for every graph (G, γ). Furthermore, the run of $\tilde{\mathcal{A}}$ on (G, g, γ) is accepting if and only if it satisfies the occurrence constraints of $\tilde{\mathcal{A}}$ if and only if the respective run of $\mathcal{A}$ on (G, γ) satisfies the occurrence constraints of $\mathcal{A}$.

Let (G, γ) be a graph with an accepting run $\rho : V \to Q$ on $\mathcal{A}$. Recall that $(G, \gamma)^D_\rho$ denotes the graph (G, γ) labeled with the weights of the tiles defined by the run ρ. Following the definition of $\mathcal{A}$ and the arguments above, we get the following observation. For $v \in V$, the weight of $(G, \gamma)^D_\rho$ at v is d_j iff v is labeled with q_j by the run ρ of $\mathcal{A}$ iff v is labeled with j by the respective run

of $\tilde{\mathcal{A}}$ iff $g(v) = j$ iff $(G, \gamma[x \to v]) \in L_j$. Note that

$$[\![\varphi]\!]_{\mathcal{W}}(G, \gamma[x \to v]) = \begin{cases} d_1 & , \text{ if } (G, \gamma[x \to v]) \in L_1 \\ \dots & \\ d_m & , \text{ if } (G, \gamma[x \to v]) \in L_m \end{cases} .$$

Now, consider the graph $(G, \gamma)_\varphi$ where we label every vertex v of (G, γ) with $[\![\varphi]\!]_{\mathcal{W}}(G, \gamma[x \to v])$ (cf. semantics in Figure 6.2). Then the observation above and the representation of $[\![\varphi]\!]_{\mathcal{W}}$ give us $(G, \gamma)_\varphi = (G, \gamma)^D_\rho$. Finally, we obtain the following

$$\begin{aligned} [\![\mathcal{A}]\!](G, \gamma) &= \sum_{\rho \in \mathrm{acc}_A(G,\gamma)} \mathrm{Val}((G, \gamma)^D_\rho) \\ &= \mathrm{Val}((G, \gamma)^D_\rho) \\ &= \mathrm{Val}((G, \gamma)_\varphi) \\ &= [\![\mathrm{Val}_x\, \varphi]\!](G, \gamma) \ . \end{aligned}$$

Thus, $[\![\mathcal{A}]\!] = [\![\mathrm{Val}_x\, \varphi]\!]$. □

Let $\varphi \in \mathrm{MSO}(\mathbb{D})$. As in Section 3.4, we call φ *FO-restricted* if all unweighted subformulas β are FO-formulas and for all subformulas $\mathrm{Val}_x\, \psi$ of φ, ψ is almost FO-boolean.

These restrictions are motivated in [DG07] (restriction of $\mathrm{Val}_x\, \psi$) and [Fic11] (restriction to FO) where it is shown that the unrestricted versions of the logic are strictly more powerful than weighted automata on words, resp. pictures. For graphs this is also true, even for the Boolean semiring. The following example shows that we cannot relax our logic-restriction to include recognizable step functions instead of FO-step functions. For pictures such an example can be found in [Fic11] (Example 6.5). Both examples use that the underlying EMSO-logics are not closed under negation, respectively under the $\forall x$ operator.

Example 6.10. We consider the Boolean semiring $\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$ as a product valuation monoid. Then Val_x coincides with $\forall x$. Furthermore, a series S is a recognizable step function if and only if S is recognizable if and only if the support of S is a recognizable language (cf. [DG07]). It is known that, in general, the application of universal quantification to a recognizable language does not yield a recognizable language. Take for example the formula $\varphi = \mathrm{reach}(x, y)$ describing that y can be reached by x. This can be checked by an EMSO-formula [AF90]. Then the formula $\forall x \forall y.\mathrm{reach}(x, y)$ describes exactly whether a graph is connected, which is no EMSO-property. Therefore, $[\![\varphi]\!] = \mathbb{1}_{L(\varphi)}$ is recognizable, but $[\![\mathrm{Val}_x\, \mathrm{Val}_y\, \varphi]\!]$ is not recognizable. ◇

We summarize our results into the following proposition.

Proposition 6.21. *If $\mathbb{D}$ is regular, then for every FO-restricted* MSO($\mathbb{D}$)*-sentence φ, there exists a wGA $\mathcal{A}$ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$.*

Proof. We use structural induction on φ. For $\varphi = d$, we use regularity of $\mathbb{D}$.

Closure under $\oplus$ is dealt with by Proposition 6.7 and Lemma 6.16. A new case concerns the 'if..then..else'-operator as follows. Let $\beta?\varphi_1 : \varphi_2$ be a subformula of φ. Since β is an FO-formula, the languages $L(\beta)$ and $L(\neg\beta)$ are recognizable by a one-state GA. Furthermore, it holds that $[\![\beta?\varphi_1 : \varphi_2]\!] = [\![\varphi_1]\!] \cap L(\beta) + [\![\varphi_2]\!] \cap L(\neg\beta)$. Then Proposition 6.9, together with Proposition 6.7 and Lemma 6.16 yields that $[\![\beta?\varphi_1 : \varphi_2]\!]$ is recognizable. Existential quantifications are dealt with by Lemma 6.17. Since φ is restricted, we know that for every subformula $\mathrm{Val}_x \psi$, the formula ψ is an almost FO-boolean formula. Therefore, we can apply Lemma 6.19 and Proposition 6.20 to maintain recognizability of our formula. □

For a formula $\varphi \in \mathbb{PD}$ over a product valuation monoid, we additionally have to restrict the usage of the $\otimes$-operator. We introduce restricted (resp. weakly- and strongly-restricted) and FO-restricted (resp. weakly-FO- and strongly-FO-restricted) formulas as in Section 3.4. In contrast to FO-restricted formulas, a restricted formula also allows EMSO-definable formulas in the unweighted fragment. Then we can, similarly to Proposition 6.21, summarize our previous results.

Proposition 6.22. *1. If $\mathbb{PD}$ is regular, then for every strongly-FO-restricted* MSO($\mathbb{PD}$)*-sentence φ, there exists a wGA $\mathcal{A}$ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$.*

2. If $\mathbb{PD}$ is idempotent and regular, then for every strongly-restricted MSO($\mathbb{PD}$)*-sentence φ, there exists a wGA $\mathcal{A}$ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$.*

Proof. We show both statements simultaneously and use structural induction on φ. Then, additionally to the proof of Proposition 6.21, we have to show the following.

If $\varphi = \beta$, for an unweighted formula β, then in the first case, β is an FO-formula and we can apply Theorem 6.2 to get a one-state GA $\mathcal{A}$ with $L(\mathcal{A}) = L(\beta)$. In the second case, β is an EMSO-formula and we can apply Theorem 6.2 to get a GA $\mathcal{A}$ with $L(\mathcal{A}) = L(\beta)$. In both cases, we transform $\mathcal{A}$ into a wGA $\mathcal{A}'$, using the constant function $\mathrm{wt} \equiv 1$. Using that every one-state graph acceptor has at most on run in the first case and the idempotence of $\mathbb{D}$ in the second case, both cases result in $[\![\mathcal{A}']\!] = \mathbb{1}_{L(\mathcal{A})} = [\![\varphi]\!]$.

If $\varphi = \psi \otimes \theta$, then ψ and θ are almost FO-boolean or one of the two is definable by an unweighted FO-formula, because φ is strongly-$\otimes$ restricted. If ψ and θ are almost FO-boolean, φ is almost FO-boolean and therefore recognizable by Corollary 6.18. If ψ or θ is definable by an unweighted FO-formula, we apply Proposition 6.9 to get that φ is recognizable. □

Now, we show that every weighted graph automaton can be simulated by an FO-restricted MSO($\mathbb{D}$)-sentence.

Proposition 6.23. *Let $\mathbb{PD}$ be a product valuation monoid and $\mathbb{D}$ its underlying valuation monoid. Then, for every wGA $\mathcal{A}$ over $\mathbb{D}$, there exists an FO-restricted* MSO($\mathbb{D}$)*-sentence φ and an FO-restricted* MSO($\mathbb{PD}$)*-sentence φ' with $[\![\mathcal{A}]\!] = [\![\varphi]\!] = [\![\varphi']\!]$.*

Proof. Let $\mathcal{A} = (Q, \Delta, \mathrm{wt}, \mathrm{Occ}, r)$ be a weighted graph automaton over A, B, and $\mathbb{D}$. We define the set variables $\bar{X} = \{X_\tau \mid \tau \in \Delta\}$. This is a finite set and can be enumerated by $(X_{\tau_1}, ..., X_{\tau_m})$.

As in the classical unweighted case, it is possible to write down a formula $\varphi' = \exists X_{\tau_1} ... \exists X_{\tau_m} . \beta(\bar{X})$, where $\beta(\bar{X})$ is an FO-formula describing the accepting runs of $\mathcal{A}$ without regard to the weights. Then, for a run $\rho : V \to Q$, we define the $(\bar{X}, G)$-assignment γ_ρ by

$$\gamma_\rho(X_\tau) = \{v \mid \mathrm{sph}^r(G_\rho, v) \text{ is isomorphic to } \tau\} \ , \ \tau \in \Delta \ .$$

Hence, $(G, \gamma_\rho) \models \beta$ if and only if $\bar{X}$ defines an accepting run ρ of $\mathcal{A}$ on G. Furthermore, for every (G, γ) satisfying β, there exists an accepting run ρ of $\mathcal{A}$ on G with $\gamma = \gamma_\rho$.

We use nested 'if..then..else'-operators to define the weight used at a position described by the variable x as follows.

$$wt(x, \bar{X}) = x \in X_{\tau_1} ? \, \mathrm{wt}(\tau_1) : (x \in X_{\tau_2} ? \, \mathrm{wt}(\tau_2) : (... : (x \in X_{\tau_m} ? \, \mathrm{wt}(\tau_m) : 0)...)$$

Adding these weights to β, we define the weighted formula

$$\theta(\bar{X}) = \beta(\bar{X}) ? \, \mathrm{Val}_x \, wt(x, \bar{X}) : 0 \ .$$

Finally, we define

$$\varphi = \bigoplus_{X_1} ... \bigoplus_{X_m} \theta(X_1, ..., X_m) \ .$$

Now, $[\![\varphi]\!](G, \gamma)$ sums over all possible assignments of $\bar{X}$ and checks whether they encode an accepting run of $\mathcal{A}$ on G. If that is the case, it assigns to every tile τ the corresponding weight $\mathrm{wt}(\tau)$. Then, Val_x, is the valuation of all weights of this particular run. For assignments not encoding an accepting run, the weight 0 is applied. We get

$$\begin{aligned}
[\![\varphi]\!](G) &= \sum_{(\bar{X}, G) \text{ assignment } \gamma} [\![\theta]\!](G, \gamma) \\
&= \sum_{\rho \in \mathrm{acc}_{\mathcal{A}}(G)} [\![\theta]\!](G, \gamma_\rho) \\
&= \sum_{\rho \in \mathrm{acc}_{\mathcal{A}}(G)} \mathrm{Val}(G_{\mathcal{A}, \rho}) \\
&= [\![\mathcal{A}]\!](G) \ .
\end{aligned}$$

The formula $wt(x, \bar{X})$ is almost FO-boolean, and φ does not use the $\exists X$ operator. Therefore, θ and φ are FO-restricted. This concludes the proof for $\varphi \in \mathrm{MSO}(\mathbb{D})$.

The construction of $\varphi' \in \mathrm{MSO}(\mathbb{PD})$ is nearly analogously. Since it holds that $[\![\beta ? \varphi : \psi]\!] = [\![(\beta \otimes \varphi) \oplus (\neg\beta \otimes \psi)]\!]$, and this formula applied to wt and θ yields an FO-boolean formula, we could simply rewrite the 'if..then..else'-operator. However, we can also (shorter and in line with the usual approach of previous works, cf. [DG07] with symbolic changes) use the following weight assignment

$$wt(x) = \bigoplus_{X_\tau \in \bar{X}} (x \in X_\tau \otimes \mathrm{wt}(\tau)) \ .$$

Together with

$$\theta(\bar{X}) = \beta(\bar{X}) \otimes \mathrm{Val}_x \, wt(x)$$

and

$$\varphi' = \bigoplus_{X_1} ... \bigoplus_{X_m} \theta(X_1, ..., X_m) \ ,$$

this yields $[\![\varphi']\!] = [\![\mathcal{A}]\!]$. □

Joining Proposition 6.23 and Proposition 6.21, this gives the first main result of this section, a Büchi-like connection between the introduced weighted graph automata and the restricted weighted logic.

Theorem 6.24. *Let $\mathbb{D} = (D, +, \mathrm{Val}, 0)$ be a regular valuation monoid and let $S : \mathrm{DG}_t(A, B) \to D$ be a series. Then the following are equivalent:*

1. *S is recognizable by a wGA over $\mathbb{D}$.*
2. *S is definable by an FO-restricted $\mathrm{MSO}(\mathbb{D})$-sentence.*

Examples of regular valuation monoids are the introduced examples using average or discounting. In the following Section 6.2.4, we will show the following sufficient conditions for regularity. All idempotent semirings are regular. Furthermore, over pointed graphs all commutative semirings and all left-distributive pv-monoids are regular. In the case of unpointed graphs, the semiring of the natural numbers $\mathbb{N}$ is regular, while the rational numbers $\mathbb{Q}$ are not regular.

To take the product of a product valuation monoid, in particular a semiring, in account, we can apply the Transformation Theorem 3.6 to graphs and we get the following result taking special care of the interaction between assumptions on our product valuation monoid and necessary restrictions of the weighted logic. The detailed notions of a left-distributive pv-monoid and a cc-valuation semiring are the same as introduced in Section 3.2.

Theorem 6.25. *Let $\mathbb{PD} = (D, +, \mathrm{Val}, \diamond, 0, 1)$ be a product valuation monoid and let $S : \mathrm{DG}_t(A, B) \to D$ be a series.*

1. *Let $\mathbb{PD}$ be regular. Then S is recognizable by a wGA iff S is definable by a strongly-FO-restricted* MSO($\mathbb{PD}$)*-sentence.*
2. *Let $\mathbb{PD}$ be left-distributive. Then S is recognizable by a wGA iff S is definable by a FO-restricted* MSO($\mathbb{PD}$)*-sentence.*
3. *Let $\mathbb{PD}$ be a cc-valuation-semiring. Then S is recognizable by a wGA iff S is definable by a weakly-FO-restricted* MSO($\mathbb{PD}$)*-sentence.*
4. *If $\mathbb{PD}$ is also idempotent, then the statements of 1. (resp. 2., and 3.) are equivalent to*
   - *S is definable by a weakly-restricted (resp. restricted or strongly-restricted)* MSO($\mathbb{PD}$)*-sentence.*

Note that every cc-valuation semiring is left-distributive and therefore regular as seen in the following chapter. This covers for example $\mathbb{PD}_1 = (\mathbb{R} \cup \{-\infty\}, \sup, \text{avg}, +, -\infty, 0)$ and $\mathbb{PD}_2 = (\mathbb{R} \cup \{-\infty\}, \sup, \text{disc}_\lambda, +, -\infty, 0)$.

6.2.4 A Deeper Look at Regularity of Weight Structures

We defined weighted graph automata over valuation monoids, a very general weight structure. While we are motivated by semirings and examples like average and discounting, in general valuation monoids can also be monoids with less structural information. As usual, this power comes at a price. In Sections 3.3 and 3.4, we have already seen that the expressibility of a weighted MSO-formula is directly connected to how much we know about the valuation monoid, see Theorem 3.6.

Another crucial consideration we have seen in Example 6.9 and in the Büchi-characterization is the following. One of the basic formulas we want to be able to express with the weighted logic (and which is therefore part of its syntax) is just a weight d. This formula expresses that every considered relational structure is evaluated to the weight d. Therefore, for an equivalence between logic and automata, we have to ensure that for every weight d there exists a graph automaton which assigns the value d to every graph. As before, we call a valuation monoid $\mathbb{D}$ *regular*, if it satisfies this requirement, i.e., for every $d \in D$, the constant series assigning d to every graph is regular. In the following section, we study conditions which ensure regularity of general valuation monoids.

If we consider words, then every semiring is trivially regular, as we can give the first transition the desired weight d and all other transitions the weight 1. In fact, as we will see now, this is true for all pointed structures operating over commutative semirings. As before, the commutativity ensures that the behavior of an automaton is not dependent on any order on the vertices.

Lemma 6.26. *Let $\mathbb{K}$ be a commutative semiring. If $\mathbb{K}$ is idempotent, then $\mathbb{K}$ is regular. If $\mathcal{C}$ is a class of pointed graphs, then $\mathbb{K}$ is regular.*

Proof. First, let $\mathbb{K}$ be idempotent, i.e. $k + k = k$ for all $k \in K$. Given $k \in K$, we construct the wGA $\mathcal{A}_k = (Q, \Delta, \mathrm{wt}, \mathrm{Occ}, 0)$ as follows. We set $Q = \{q_0, q_1\}$. We let Δ consist of all 0-tiles, i.e., Δ consists of all tiles with one vertex. We define Δ_{q_1} as the set of all 0-tiles labeled with q_1 at their vertex. We define

$$\mathrm{wt}(\tau) = \begin{cases} k & \text{, if } \tau \in \Delta_{q_1} \\ 1 & \text{, otherwise} \end{cases} \text{ , for } \tau \in \Delta \text{ , and}$$
$$\mathrm{Occ} = \big(\sum_{\tau \in \Delta_{q_1}} \mathrm{occ}(\tau) \big) \geq 1 \wedge \neg \big(\big(\sum_{\tau \in \Delta_{q_1}} \mathrm{occ}(\tau) \big) \geq 2 \big) \ .$$

This ensures that for every graph, every accepting run uses the state q_1 exactly once, and assigns the weight k to the tile using q_1. Then every accepting run of $\mathcal{A}_k$ has the weight $1 \cdot ... \cdot 1 \cdot k \cdot 1 \cdot .... \cdot 1 = k$, and using the idempotence of $\mathbb{K}$, we get $[\![\mathcal{A}_k]\!](G) = k + ... + k = k$. Note that we have as many accepting runs as vertices because we can apply the tile with the state q_1 at every vertex, thus the use of idempotence is crucial.

For the second part, let $\mathcal{C}$ be a class of pointed graphs over $A' = A \times \{0, 1\}$ and B, where the second component refers to the pointing, i.e., for every graph there exists exactly one vertex labeled with 1 at the second component.

Given $k \in K$, we construct the wGA $\mathcal{A}_k = (\{q_0\}, \Delta, \mathrm{wt}, \mathit{true}, 0)$ as follows. As above, we allow every possible tile in Δ. All tiles consisting of a vertex labeled with $(a, 1, q_0)$ for some $a \in A$ are assigned the weight k and all other tiles weight 1. Then $[\![\mathcal{A}_k]\!](G) = k$ for all pointed graphs G. Note that the pointing ensures that we use the weight k exactly once. □

In the case of product valuation monoids, we get the following result, which shows that over pointed graphs every left-distributive pv-monoid is regular, generalizing the corresponding result for words [DM12] and nested words (cf. Section 4.3).

Lemma 6.27. *Every left-Val-distributive product valuation monoid $\mathbb{PD}$ is regular over graphs. Every left-multiplicative product valuation monoid $\mathbb{PD}$ is regular over pointed graphs.*

Proof. Let $\mathbb{PD}$ be left-Val-distributive and $d \in D$. We construct the wGA $\mathcal{A}_d = (\{q_0\}, \Delta, \mathrm{wt}, \mathit{true}, 0)$ as follows. We allow every possible tile of size 0 in Δ and assign the weight d to all tiles. Then, for any graph G, $[\![\mathcal{A}_d]\!](G) = \mathrm{Val}(G_d)$, were G_d is the same graph consisting of only vertices labeled with d. Let G_1 be the graph with the same structure as G but consisting of only vertices labeled with 1. Since $\mathbb{PD}$ is left-Val-distributive, we have $\mathrm{Val}(G_d) = d \diamond \mathrm{Val}(G_1) = d \diamond 1 = d$, where the last two equalities are axioms of the pv-monoid. Thus, $[\![\mathcal{A}_d]\!](G) = d$ for every graph G.

Now, let $\mathbb{PD}$ be left-multiplicative and $d \in D$. We construct the wGA $\mathcal{A}_d = (\{q_0\}, \Delta, \mathrm{wt}, \mathit{true}, 0)$ as follows. All pointed tiles consisting of a vertex which is marked as root of the graph are assigned the weight d and all other

tiles (i.e. all tiles where the root is empty) the weight 1. Then, for any given graph G, we have $[\![\mathcal{A}_d]\!](G) = \mathrm{Val}(G_{\mathrm{root}(d)})$ were $G_{\mathrm{root}(d)}$ is the same graph where the root is labeled with d and all other vertices are labeled with 1. Since $\mathbb{PD}$ is left-multiplicative, we get $\mathrm{Val}(G_{\mathrm{root}(d)}) = d \diamond \mathrm{Val}(G_1) = d \diamond 1 = d$. Thus, $[\![\mathcal{A}_d]\!](G) = d$ for every pointed graph G. □

In the case of non-pointed graphs the situation is more complex. In fact, even for well understood weight structures like the natural numbers $\mathbb{N}$ or the rational numbers $\mathbb{Q}$, the question, whether there is an automaton which assign to every graph the value 2 or respectively the value 0.5, is not trivial. In the following, we show that $\mathbb{N}$ is regular, thus the question for the value 2 can be answered with 'yes'. Furthermore, we prove that the same question for the value 0.5 can be answered with 'no', thus showing that $\mathbb{Q}$ is not regular.

In the following, given $a \in A$, we also write a for the 0-tile consisting of only the vertex a. Furthermore, consider a wGA $\mathcal{A}$ over a commutative semiring, such that $\mathcal{A}$ is not restricted by the alphabet, the tiles, or the occurrence constraint. Then, the following lemma gives a formula to calculate the behavior of $\mathcal{A}$.

Lemma 6.28. *Let $\mathcal{A} = (Q, \Delta, \mathrm{wt}, \mathrm{Occ}, r)$ be a wGA over a commutative semiring $\mathbb{K}$ with $|Q| = m$, $r = 0$, Δ consists of all 0-tiles, $\mathrm{Occ} = true$, and wt is only depending on the state, i.e. $\mathrm{wt}(a, q_i) = k_i \in \mathbb{K}$ for all $a \in A$ and $q_i \in Q$. Then for all graphs G with at least one vertex, we get*

$$[\![\mathcal{A}]\!](G) = \big(\sum_{i=1}^{m} k_i\big)^{|V_G|} \ ,$$

where $|V_G|$ is the number of vertices of G.

Proof. We prove this formula by induction over the number of vertices of G. If G has exactly one vertex, then $\mathcal{A}$ has exactly m accepting runs on G using the states q_1 to q_m at this vertex. These runs have the respective weights k_1 to k_m, thus $[\![\mathcal{A}]\!](G) = \sum_{i=1}^{m} k_i$.

Now consider a graph G with n vertices and let v be one of the vertices. Let G' be the graph G without the vertex v and its edges. By the induction hypothesis, $[\![\mathcal{A}]\!](G') = (\sum_{i=1}^{m} k_i)^{n-1}$. Since $\mathcal{A}$ has no restricting rules, we can write the behavior of $\mathcal{A}$ as

$$[\![\mathcal{A}]\!](G) = \sum_{i_1,..,i_n \in \{1,..,m\}} \prod_{j=1}^{n} k_{i_j} \ .$$

Using the distributivity of $\mathbb{K}$ at * and the induction hypothesis at **, it follows

that

$$\begin{aligned}
[\![\mathcal{A}]\!](G) &= \sum_{i_1,..,i_n \in \{1,..,m\}} \prod_{j=1}^{n} k_{i_j} \\
&\overset{*}{=} \big(\sum_{i=1}^{m} k_i\big)\big(\sum_{i_1,..,i_{n-1} \in \{1,..,m\}} \prod_{j=1}^{n-1} k_{i_j}\big) \\
&\overset{**}{=} \big(\sum_{i=1}^{m} k_i\big)\big(\sum_{i=1}^{m} k_i\big)^{n-1} \\
&= \big(\sum_{i=1}^{m} k_i\big)^{n} \ .
\end{aligned}$$

□

Theorem 6.29. $\mathbb{N}$ *is regular over graphs.*

Proof. We construct $\mathcal{A}$ with $[\![\mathcal{A}]\!](G) = 1$ for all graphs G as follows. Set $Q = \{q_1\}$, $r = 0$, wt $\equiv 1$, Occ $= true$, and Δ consists of all 0-tiles. Then every graph G has exactly one run over $\mathcal{A}$ with weight 1, so $[\![\mathcal{A}]\!] \equiv 1$.

Let $k \in \mathbb{N}$. Then, the constant series k equals the disjoint sum of k-many automata $\mathcal{A}$ recognizing 1. By Proposition 6.7, this sum is also recognizable. Additionally, we give a direct construction of $\mathcal{A}_k$ with $[\![\mathcal{A}_k]\!](G) = k$ for all graphs G, as follows. Set $Q = \{q_1, ..., q_k\}$, $r = 0$, wt $\equiv 1$, Δ consist of all 0-tiles, and

$$\text{Occ} = \bigvee_{i=1}^{k} \bigwedge_{\substack{\tau \in \Delta, \\ \tau \text{ contains no } q_i}} \neg\, \text{occ}(\tau) \geq 0 \ .$$

Then, every run of A_k uses exactly one state, thus every graph has exactly k different runs and every run has weight 1. Thus, $[\![\mathcal{A}_k]\!] \equiv k$. □

Theorem 6.30. $\mathbb{Z}$ *is regular over graphs.*

Proof. In Theorem 6.29, we have shown that the natural numbers are regular. Let $k \in \mathbb{N}$. By Proposition 6.7 and $-k = (-1) + (-1) + ... + (-1)$, it remains to show that there exists an automaton $\mathcal{A}$ with $[\![\mathcal{A}]\!](G) = -1$ for all graphs G. We construct $\mathcal{A}$ as follows.

Set $Q = \{q_1, q_2\}$, $r = 0$, $\text{wt}(a, q_1) = 1$ and $\text{wt}(a, q_2) = -1$ for all $a \in A$. Let Δ consist of all 0-tiles, and let Δ_{q_2} be the set of all tiles which are labeled with state q_1 at their only vertex. Then, using formula (6.1), we set

$$\text{Occ} = \big(\sum_{\tau \in \Delta_{q_2}} \text{occ}(\tau)\big) \geq 1 \ .$$

This enforces every run to use q_1 at least once. To calculate $[\![\mathcal{A}]\!]$, we consider $\mathcal{A}'$ which is the same automaton as $\mathcal{A}$ but without any occurrence constraint

($\mathrm{Occ}' = \mathit{true}$). By Lemma 6.28,

$$[\![\mathcal{A}']\!](G) = (1 + (-1))^{|V_G|} = 0 \ .$$

The occurrence constraint Occ disallows exactly the one run which applies q_2 to every vertex. This run has weight 1. Thus,

$$[\![\mathcal{A}]\!](G) = [\![\mathcal{A}']\!](G) - 1 = -1 \ . \qquad \square$$

Note that the monoid $(\mathbb{Z}, +)$ is not generated by 1. Therefore, Theorem 6.30 shows that there exists a regular semiring which is neither idempotent nor generated by only one element and addition.

In the following, we show that not all commutative semirings are regular. We prove that we cannot recognize the constant series of rational non-integer numbers, e.g. 0.5, in the ring of the rational numbers $\mathbb{Q}$. The proof consists of multiple steps and auxiliary results and uses analysis and combinatorics. If we disallow negative weights as in $\mathbb{Q}^+$, we can shorten it significantly. We use disconnected graphs to avoid extensive case distinctions. We strongly conjecture that the theorem also holds if we restrict the family of graphs to connected graphs. In this case, the main adaption to the proof would be considering big enough circles C_n as graphs instead of the graph family G_n used in the following.

Theorem 6.31. *(a) $\mathbb{Q}^+$ is not regular over graphs.*

(b) $\mathbb{Q}$ is not regular over graphs.

The simplified main idea of the proof is the following. Let $a \in A$. Let G_n be the graph consisting of n disconnected nodes labeled with a. We prove that for every automaton $\mathcal{A}$, $\lim_{n\to\infty}[\![\mathcal{A}]\!](G_n) \neq 0.5$. Every graph G_n is highly symmetric, so given an accepted run with a certain distribution of states, we can count how many distinct runs there are with this distribution of states. Multiplying with the respective weights of these runs, we show that for increasing n the limit of the number of runs times the weight of the run has to be 0, 1, or tends to infinity. Since the behavior of every wGA is the sum over the possible runs, and therefore the sum over all possible distributions of states, the value $[\![\mathcal{A}]\!](G_n)$ tends to a natural number or to infinity.

We give an example as follows. Assume we have an wGA $\mathcal{A}$ with $[\![\mathcal{A}]\!](G_n) = 0.5$ for all n. If $\mathcal{A}$ has only one state q with $\mathrm{wt}(a, q) = k$, then $[\![\mathcal{A}]\!](G_n) = k^n \not\to_n 0.5$. So, $\mathcal{A}$ has to have at least two different states. Also, for any G_n, $\mathcal{A}$ has at least one run with weight < 1. Furthermore, since n can be arbitrarily big, $\mathcal{A}$ has to have tiles which can be used arbitrarily often. If one of these tiles has weight greater than or equal to 1, we can construct n (symmetrical but distinct) runs of $\mathcal{A}$ with a fixed weight. Therefore, $[\![\mathcal{A}]\!]$ gets arbitrarily large, a contradiction. If the sum of tiles occurring potentially arbitrarily often

have weight less than 1, then $[\![\mathcal{A}]\!](G_n)$ converges to 0 since the weight of these runs decreases faster than the number of possible runs increases.

In the following, taking possible occurrence-constraints of $\mathcal{A}$ into account, we make this precise. To enhance readability of the proof, we transfer some auxiliary results to the back.

Proof of Theorem 6.31. We show that for all $p \in \mathbb{Q}$, $p \notin \mathbb{N}$, there exists no weighted graph automaton which evaluates every graph to p. Note that showing this for, e.g., $p = 0.5$ would suffice.

As before, let $a \in A$ and let $(G_n)_{n\in\mathbb{N}}$ be the family of graphs consisting of n nodes labeled with a and without any edges. Assume $[\![\mathcal{A}]\!](G_n) = p$ and let $\mathcal{A}$ have m states, $q_1, ..., q_m$. Since G_n has no edges, $\mathcal{A}$ can use only 0-tiles in Δ, which are tiles consisting of only one node labeled with a and q_i. Let $x_1, ..., x_m$ be the weights of the respective tiles. We may omit the label a of the tiles in the following.

Let Occ be the occurrence constraint of $\mathcal{A}$, a boolean formula over a finite set of atomic formulas $\mathrm{occ}(\tau) \geq k$, with $\tau \in \Delta$ and $k \in \mathbb{N}$. We define

$$\mathrm{occ}(\tau) = k \quad \text{as} \quad \mathrm{occ}(\tau) \geq k \wedge \neg\, \mathrm{occ}(\tau) \geq k+1.$$

We transform Occ into a formula Occ′ without negated '$\geq$' by replacing

$$\neg(\mathrm{occ}(\tau) \geq k) \quad \text{with} \quad \mathrm{occ}(\tau) = 0 \vee \mathrm{occ}(\tau) = 1 \vee ... \vee \mathrm{occ}(\tau) = k-1 \ .$$

Then Occ′ consists only of formulas $\mathrm{occ}(\tau) = k$ and $\mathrm{occ}(\tau) \geq k$ together with disjunctions and conjunctions. Hence, we can transform Occ′ into its *canonical disjunctive normal form* (CDNF), consisting of disjunctions of *product terms*, i.e., minimal and complete conjunctions. In this form, every graph satisfying Occ satisfies exactly one product term. Therefore, using fixed c_i, $i = 1, ..., m$, every product term can be written as

$$\Big(\bigwedge_{i=1,...,m'} (\mathrm{occ}(a, q_i) \geq c_i) \wedge \bigwedge_{i=m'+1,...,m} (\mathrm{occ}(a, q_i) = c_i)\Big) \ ,$$

where m' is between 1 and m. It follows that the behavior of $\mathcal{A}$ can be written as $[\![\mathcal{A}]\!] = [\![\mathcal{A}_1]\!] + ... + [\![\mathcal{A}_\ell]\!]$, where ℓ is the number of product terms and $\mathcal{A}_i$ is the automaton where we replaced Occ by the i-th product term of Occ′.

Now, we distinguish between $\mathbb{Q}^+$ and $\mathbb{Q}$.

(a) If we encounter no negative weights, it is possible to show that

$$\lim_{n\to\infty} [\![\mathcal{A}_i]\!](G_n) \in \{0, 1, \infty\} \ ,$$

as follows. Let $n' = n - (c_{m'+1} + ... + c_m)$. In Lemma 6.34, we show that by counting possible runs together with their respective weights, we get

$$[\![\mathcal{A}_i]\!](G_n) = \frac{n(n-1)...(n'+1)}{c_{m'+1}!...c_m!}\Big(\prod_{i=m'+1}^{m} x_i^{c_i}\Big) \sum_{\substack{k_1,...,k_{m'}\in\mathbb{N} \\ k_1+...+k_{m'}=n' \\ \forall i\leq m': k_i \geq c_i}} \frac{n'!}{k_1!...k_{m'}!} \prod_{i=1}^{m'} x_i^{k_i} \ .$$

While this formula looks rather nasty, we are only interested in its asymptotic behavior. Since the c_i are constant, the term

$$\frac{n(n-1)...(n'+1)}{c_{m'+1}!...c_m!}\Big(\prod_{i=m'+1}^{m} x_i^{c_i}\Big)$$

is polynomial in n. Furthermore, Lemma 6.35 shows that the second part of this formula

$$\sum_{\substack{k_1+...+k_{m'}=n' \\ \forall i\leq m': k_i \geq c_i}} \frac{n'!}{k_1!...k_{m'}!} \prod_{i=1}^{m'} x_i^{k_i}$$

is asymptotically equal to $(x_1 + ... + x_{m'})^{n'}$. Therefore, if $x_1 + ... + x_{m'} > 1$, then $[\![\mathcal{A}_i]\!](G_n)$ gets arbitrarily large, a contradiction. If $x_1 + ... + x_{m'} = 1$ and $m = m'$, then the polynomial term disappears and $[\![\mathcal{A}_i]\!](G_n)$ equals 1, c.f. also Lemma 6.28. If $x_1 + ... + x_{m'} = 1$ and $m > m'$, then again $[\![\mathcal{A}_i]\!](G_n)$ gets arbitrarily large due the polynomial part which is positive because of $x_i > 0$. If $x_1 + ... + x_{m'} < 1$ then

$$(x_1 + ... + x_{m'})^{n'} = \frac{(x_1 + ... + x_{m'})^n}{(x_1 + ... + x_{m'})^{c_{m'+1}+...+c_m}}$$

is exponential in n and therefore tends stronger to 0 than any polynomial. So, $\lim_{n\to\infty}[\![\mathcal{A}_i]\!](G_n) = 0$.

From this case distinction, it follows that $\lim_{n\to\infty}[\![\mathcal{A}_i]\!](G_n) \in \{0, 1, \infty\}$. Therefore, the behavior of $\mathcal{A}$ can only be constant if it is an integer-value, which concludes our proof.

(b) We show by Lemma 6.34 together with Proposition 6.36 that we can describe the behavior of every $\mathcal{A}_i$, and therefore also the behavior of $\mathcal{A}$, as a finite sum $d_1q_1^n + ... + d_tq_t^n$, where $q_i \in \mathbb{Q}$, $t \in \mathbb{N}$, and all d_i have a very special form. In the following, we show that we can force all these d_i to be integers.

We make n large enough such that it is a multiple of all combinations of possible appearing constants, as follows. Since every x_i is a rational number, it can be written as

$$x_i = \frac{x_i^{\mathrm{D}}}{x_i^{\mathrm{N}}}, \qquad x_i^{\mathrm{D}} \in \mathbb{Z}, x_i^{\mathrm{N}} \in \mathbb{N} \ .$$

Furthermore, since we have only finitely many x_i, we can build the least common denominator lcd of all $x_1, ..., x_m$ and write

$$x_i = \frac{x_i^{\mathrm{M}}}{lcd}, \qquad x_i^{\mathrm{M}} \in \mathbb{Z} \ .$$

We set $c = \sum_{i=1}^{m} c_i$ and define the sequence $(n_k)_{k\in\mathbb{N}}$ by

$$n_k = k \cdot \Big((\prod_{i=0}^{m} c_i!) \cdot (\prod_{i=0}^{m} x_i^{\mathrm{N}})^c \cdot ((\sum_{i=0}^{m} |x_i^{\mathrm{M}}|)!)^c\Big)^2 \ .$$

In the following, we only consider graphs G_{n_k}. Clearly, if $\mathcal{A}$ is constant on all graphs, it also has to be constant on all graphs of the sequence G_{n_k}. Then in Lemma 6.34 and Proposition 6.36, we (possibly in both cases) extract constants from the sum of the following form:

$$\pm\frac{n(n-1)...(n-c_{j_1}-...-c_{j_{m'}}+1)}{c_{j_1}!...c_{j_{m'}}!}\Big(\prod_{i\in J}x_i^{c_i}\Big)\Big(\sum_{i\notin J}x_i\Big)^{-(\sum_{i\in J}c_i)} \ ,$$

for some index set $J=\{j_1,...,j_{m'}\}\subseteq\{1,...,m\}$, $m'\leq m$. If in these cases $m'=m$, resp. $J=\emptyset$, then this factor equals 1. Otherwise, for $n=n_k$, the factors at least contain n_k. All n_k are by construction a multiple of the first denominator, the denominators possibly appearing in $x_i^{c_i}$, and the denominators possibly appearing in $\big(\sum_{i\notin J}x_i\big)^{-(\sum_{i\in J}c_i)}$. It follows that for all n_k, all appearing d_i are integers. Therefore, by Lemma 6.37, $\lim_{k\to\infty}[\![\mathcal{A}]\!](G_{n_k})$ can only be constant if it equals an integer, which yields the result. □

In the following, we prove the auxiliary results that are used in the proof of Theorem 6.31. We employ the *multinomial coefficient*, which describes how many possibilities there are to distribute n distinguishable balls into m boxes when there have to be k_1 balls in box 1, k_2 balls in box 2, and so on. In our context, the balls are the nodes of G_n and the boxes are the states.

Definition 6.32. The *multinomial coefficient* $\binom{n}{k_1,...,k_m}$ is defined as

$$\binom{n}{k_1,...,k_m}=\frac{n!}{k_1!k_2!...k_m!} \ .$$

Theorem 6.33 (Multinomial theorem)**.** *Let $m\in\mathbb{N}_+$, $n\in\mathbb{N}$, and $x_1,...,x_m\in\mathbb{R}$. Then*

$$(x_1+...+x_m)^n=\sum_{\substack{k_1,...,k_m\in\mathbb{N}\\ k_1+...+k_m=n}}\binom{n}{k_1,...,k_m}\prod_{i=1}^m x_i^{k_i} \ .$$

The following lemma describes the behavior of a very particular weighted graph automaton.

Lemma 6.34. *Let $a\in A$ and $m,m'\in\mathbb{N}$. Let $\mathcal{A}=(Q,\Delta,\mathrm{wt},\mathrm{Occ},r)$ be a wGA over $\mathbb{Q}$ or $\mathbb{Q}^+$ with $Q=\{q_1,...,q_m\}$, $r=0$, $\Delta=\{a\}\times Q$, $\mathrm{wt}(a,q_i)=x_i$, for $i=1,...,m$, and*

$$\mathrm{Occ}:=\Big(\bigwedge_{i=1,...,m'}(\mathrm{occ}(a,q_i)\geq c_i)\wedge\bigwedge_{i=m'+1,...,m}(\mathrm{occ}(a,q_i)=c_i)\Big) \ .$$

Let G_n be a graph with n vertices labeled with a and without edges. Let $n' = n - (c_{m'+1} + ... + c_m)$. Then

$$[\![\mathcal{A}]\!](G_n) = \frac{n(n-1)...(n'+1)}{c_{m'+1}!...c_m!}\Big(\prod_{i=m'+1}^{m} x_i^{c_i}\Big) \sum_{\substack{k_1,...,k_{m'}\in\mathbb{N} \\ k_1+...+k_{m'}=n' \\ \forall i\leq m': k_i\geq c_i}} \frac{n'!}{k_1!...k_{m'}!} \prod_{i=1}^{m'} x_i^{k_i} \quad .$$

Proof. Let i in the following always refer to a number between 1 and m. Let k_i be the number of states q_i used in a run of $\mathcal{A}$ over G_n. Then, the value $[\![\mathcal{A}_i]\!](G_n)$ is the sum over the weight of all runs satisfying Occ, which are exactly all runs such that $k_1 + ... + k_m = n$, $k_i = c_i$ for all $i \leq m'$, and $k_i \geq c_i$ for all $i > m'$. Using combinatorics, we know that for such given $k_1, ..., k_m$, the multinomial coefficient $\binom{n}{k_1,...,k_m}$ describes the number of possible distributions of n nodes into states and therefore the number of possible runs. Furthermore, the weight of every run using these states is $\prod_{i=1}^{m} x_i^{k_i}$. It follows that

$$[\![\mathcal{A}]\!](G_n) = \sum_{\substack{k_1,...,k_m\in\mathbb{N} \\ k_1+...+k_m=n \\ \forall i\leq m': k_i\geq c_i \\ \forall i>m': k_i=c_i}} \binom{n}{k_1, ..., k_m} \prod_{i=1}^{m} x_i^{k_i} \quad .$$

Using $n' = n - (c_{m'+1} + ... + c_m)$ and extracting the constants $k_i = c_i$ from the following sum, we get

$$\begin{aligned}
[\![\mathcal{A}]\!](G_n) &= \sum_{\substack{k_1,...,k_m\in\mathbb{N} \\ k_1+...+k_m=n \\ \forall i\leq m': k_i\geq c_i \\ \forall i>m': k_i=c_i}} \binom{n}{k_1, ..., k_m} \prod_{i=1}^{m} x_i^{k_i} \\
&= \sum_{\substack{k,...,k_{m'}\in\mathbb{N} \\ k_1+...+k_{m'}=n' \\ \forall i\leq m': k_i\geq c_i}} \frac{n!}{k_1!...k_{m'}!c_{m'+1}!...c_m!} \prod_{i=1}^{m'} x_i^{k_i} \prod_{i=m'+1}^{m} x_i^{c_i} \\
&= \frac{n(n-1)...(n'+1)}{c_{m'+1}!...c_m!}\Big(\prod_{i=m'+1}^{m} x_i^{c_i}\Big) \sum_{\substack{k_1,...,k_{m'}\in\mathbb{N} \\ k_1+...+k_{m'}=n' \\ \forall i\leq m': k_i\geq c_i}} \frac{n'!}{k_1!...k_{m'}!} \prod_{i=1}^{m'} x_i^{k_i}
\end{aligned}$$

□

Now, we show that the asymptotic behavior of the multinomial formula stays the same if we exclude combinations were some of the exponents have to be above certain (fixed) thresholds.

Lemma 6.35. *Let $m \in \mathbb{N}$, $x_1, ..., x_m \in \mathbb{Q}^+$, and $c_1, ..., c_m \in \mathbb{N}$. Then for $n \to \infty$, we have asymptotically*

$$\sum_{\substack{k_1,...,k_m \in \mathbb{N} \\ k_1+...+k_m=n \\ \forall i \leq m: k_i \geq c_i}} \binom{n}{k_1, ..., k_m} \prod_{i=1}^{m} x_i^{k_i} \sim (x_1 + ... + x_m)^n \ .$$

Proof. By Theorem 6.33, the left hand side is smaller than the right hand side.

For the converse, We make a case distinction regarding $x_1 + ... + x_m$ as follows.

If $x_1 + ... + x_m < 1$, then the right hand side converges to 0. Since we encounter only positive values, and the left hand side is smaller than the right hand side, both sides have to converge against 0.

If $x_1 + ... + x_m = 1$, then $x_i \leq 1$ for all i. Moreover, by Theorem 6.33, we have

$$\sum_{\substack{k_1,...,k_m \in \mathbb{N} \\ k_1+...+k_m=n \\ \forall i \leq m: k_i \geq c_i}} \binom{n}{k_1, ..., k_m} \prod_{i=1}^{m} x_i^{k_i} = 1 - \sum_{\substack{k_1,...,k_m \in \mathbb{N} \\ k_1+...+k_m=n \\ \exists i \leq m: k_i < c_i}} \binom{n}{k_1, ..., k_m} \prod_{i=1}^{m} x_i^{k_i} \ .$$

Let $c = \max_i c_i$ and $x = \min_i x_i$. Then we have

$$\begin{aligned}
& \sum_{\substack{k_1,...,k_m \in \mathbb{N} \\ k_1+...+k_m=n \\ \exists i \leq m: k_i < c_i}} \binom{n}{k_1, ..., k_m} \prod_{i=1}^{m} x_i^{k_i} \\
\leq \quad & \sum_{\ell=1}^{m} \sum_{\substack{k_1,...,k_m \in \mathbb{N} \\ k_1+...+k_m=n \\ k_\ell < c_\ell}} \binom{n}{k_1, ..., k_m} \prod_{i=1}^{m} x_i^{k_i} \\
= \quad & \sum_{\ell=1}^{m} \sum_{j=0}^{c_\ell - 1} \sum_{\substack{k_1,...,k_m \in \mathbb{N} \\ k_1+...+k_m=n \\ k_j = c_j}} \frac{n!}{k_1!...k_m!} \prod_{i=1}^{m} x_i^{k_i} \\
= \quad & \sum_{\ell=1}^{m} \sum_{j=0}^{c_\ell - 1} \frac{n! x_j}{(n-c_j)! c_j!} \sum_{\substack{k_i \in \mathbb{N} \\ \sum_{i \neq j} k_i = n \\ i \in \{1,...,m\} \setminus \{j\}}} \frac{n!}{k_1!...k_m!} \prod_{i \in \{1,...,m\} \setminus \{j\}} x_i^{k_i} \\
\overset{\text{Thm.6.33}}{=} \quad & \sum_{\ell=1}^{m} \sum_{j=0}^{c_\ell - 1} \frac{n! x_j}{(n-c_j)! c_j!} \Big(\sum_{i \in \{1,...,m\} \setminus \{j\}} x_i \Big)^n \\
\leq \quad & mcn^c (1-x)^n \ ,
\end{aligned}$$

this converges against 0 for $n \to \infty$. Therefore, both sides converge against 1, which shows the claim.

Finally, let $x_1 + ... + x_m > 1$. Then for all $k, c \in \mathbb{N}$ with $0 \leq (k + c) \leq n$, it holds that

$$\frac{n!}{(k+c)!} = (k+c) \cdot ... \cdot n \geq k \cdot ... \cdot (n-c) = \frac{(n-c)!}{k!} \quad . \tag{6.3}$$

Set $c = \sum_{i=1}^{m} c_i$ and let $n > c$. Using an index shift $l_i = k_i - c_i$, and using Formula 6.3 at equation * for every k_i and c_i once, and Theorem 6.33 at equation **, we get

$$\begin{aligned}
&\sum_{\substack{k_1,...,k_m \in \mathbb{N} \\ k_1+...+k_m=n \\ \forall i \leq m: k_i \geq c_i}} \binom{n}{k_1, ..., k_m} \prod_{i=1}^{m} x_i^{k_i} \\
&\qquad = \sum_{\substack{l_1,...,l_m \in \mathbb{N} \\ l_1+...+l_m=n-c}} \frac{n!}{(l_1+c_1)!, ..., (l_m+c_m)!} \prod_{i=1}^{m} x_i^{l_i+c_i} \\
&\qquad \overset{*}{\geq} \sum_{\substack{l_1,...,l_m \in \mathbb{N} \\ l_1+...+l_m=n-c}} \frac{(n-c)!}{l_1!, ..., l_m!} \prod_{i=1}^{m} x_i^{l_i+c_i} \\
&\qquad \overset{**}{=} \Big(\prod_{i=1}^{m} x_i^{c_i}\Big)(x_1 + ... + x_m)^{n-c} \quad ,
\end{aligned}$$

which goes exponentially fast to infinity, as does the right hand side of Lemma 6.35. This concludes the case distinction and the proof. □

Proposition 6.36. *Let $n, m, c_i \in \mathbb{N}$ and $x_i \in \mathbb{Q}$, $i = 1, ..., m$. Then,*

$$\sum_{\substack{k_1,...,k_m \in \mathbb{N} \\ k_1+...+k_m=n \\ \forall i \leq m: k_i \geq c_i}} \frac{n!}{k_1!...k_m!} \prod_{i=1}^{m} x_i^{k_i}$$

can be written as a finite sum $d_1 q_1^n + ... + d_t q_t^n$, $t \in \mathbb{N}$, such that every $d_j, q_j \in \mathbb{Q}$ and every d_j is of the form

$$\pm \frac{n(n-1)...(n - c_{j_1} - ... - c_{j_{m'}} + 1)}{c_{j_1}!...c_{j_{m'}}!} \Big(\prod_{i \in J} x_i^{c_i}\Big)\Big(\sum_{i \notin J} x_i\Big)^{-(\sum_{i \in J} c_i)}$$

for some index set $J = \{j_1, ..., j_{m'}\} \subseteq \{1, ..., m\}$, $m' \leq m$.

Proof. Using the shortcut X for $\frac{n!}{k_1!...k_m!}\prod_{i=1}^{m} x_i^{k_i}$, and omitting $k_1, ..., k_m \in \mathbb{N}$, we break down the sum as follows

$$\begin{aligned}\sum_{\substack{k_1+...+k_m=n\\ \forall i\leq m: k_i\geq c_i}} X &= \sum_{k_1+...+k_m=n} X - \sum_{\substack{k_1+...+k_m=n\\ \exists i\leq m: k_i<c_i}} X \\ &\overset{\text{Thm.6.33}}{=} (x_1+...x_m)^n - \sum_{\substack{k_1+...+k_m=n\\ \exists i\leq m: k_i<c_i}} X \ .\end{aligned}$$

By the inclusion-exclusion principle, we can rewrite the second part as

$$\begin{aligned}\sum_{\substack{k_1+...+k_m=n\\ \exists i\leq m: k_i<c_i}} X &= \sum_{\substack{k_1+...+k_m=n\\ k_1<c_1}} X + ... + \sum_{\substack{k_1+...+k_m=n\\ k_m<c_m}} X - \sum_{\substack{k_1+...+k_m=n\\ k_1<c_1\\ k_2<c_1}} X - ... + ... \\ &= \sum_{J\subseteq\{1,...,m\}} (-1)^{|J|+1} \sum_{\substack{k_1+...+k_m=n\\ \forall i\in J: k_i<c_i}} X \ .\end{aligned}$$

Then the first sum is finite and given $J \subseteq \{1, ..., m\}$, one summand of it can be written as

$$\pm \sum_{\substack{k_1+...+k_m=n\\ \forall i\in J: k_i<c_i}} X = \pm \sum_{i\in J} \sum_{c_i'\in\{0,...,c_i-1\}} \sum_{\substack{k_1+...+k_m=n\\ \forall i\in J: k_i=c_i'}} X \ .$$

Again, the first two sums are finite. Let $m' = |J|$, and write J as $J = \{j_1, ..., j_{m'}\}$. We set $n' = n - \sum_{i\in J} c_i$ and extract the constants as in Lemma 6.34. Then the remaining k_i, $i \notin J$, are not bound anymore and we can apply Theorem 6.33 to the index set $\{i \mid 1 \leq i \leq m, i \notin J\}$ together with n'. Therefore, every summand of the inner sum can be written as

$$\begin{aligned}&\pm \sum_{\substack{k_1+...+k_m=n\\ \forall i\in J: k_i=c_i}} X \\ &\pm \sum_{\substack{k_1+...+k_m=n\\ \forall i\in J: k_i=c_i}} \frac{n!}{k_1!...k_m!} \prod_{i=1}^{m} x_i^{k_i} \\ \overset{\text{Thm.6.33}}{=} &\pm \frac{n(n-1)...(n'+1)}{c_{j_1}!...c_{j_{m'}}!} \Big(\prod_{i\in J} x_i^{c_i}\Big)\Big(\sum_{i\notin J} x_i\Big)^{n'} \\ = \quad &\pm \frac{n(n-1)...(n'+1)}{c_{j_1}!...c_{j_{m'}}!} \Big(\prod_{i\in J} x_i^{c_i}\Big)\Big(\sum_{i\notin J} x_i\Big)^{-(\sum_{i\in J} c_i)}\Big(\sum_{i\notin J} x_i\Big)^{n} \ .\end{aligned}$$

Since $\sum_{i\notin J} x_i \in \mathbb{Q}$, this concludes our proof. □

Lemma 6.37. *Let $m \in \mathbb{N}$ and $d_1, ..., d_m \in \mathbb{Q} \setminus \{0\}$. Let q_1,...,q_m be pairwise distinct rational numbers. Then, $\lim_{n\to\infty} d_1 q_1^n + ... + d_m q_m^n$ either does not converge or converges towards 0 or d_1 or ... or d_m.*

Proof. If there exists a $q_i > 0$ such that $|q_i| \geq |q_j|$ for all j, then $\lim_n d_1 q_1^n + ... + d_m q_m^n = d_i q_i^n$, which either diverges for $q_i > 1$, converges to d_i for $q_i = 1$ or converges to 0 for $q_i < 1$. Otherwise, there exists a $q_i < 0$ such that $|q_i| \geq |q_j|$ for all j. In this case, the partial limits alternate and therefore the limit does not exist. □

With this, the proof of Theorem 6.31 is complete.

6.3 Words and other Special Cases

In this section, using ideas from Thomas [Tho96], we show that existing quantitative automata models over over finite words [DG07], trees [DV06], pictures [Fic11], and nested words [Mat10b] can be seen as special instances of weighted graph automata operating on finite graphs. Furthermore, we show that over these special structures, weighted graph automata can be reduced to the known automata models operating on the respective structure. Hence, we get previous equivalence results connecting weighted logic and weighted automata over these structures as a consequence of Theorem 6.24 (in a slightly modified version). In the following, to simplify the transition to the automata models from the literature, we only consider weighted automata and logics over a commutative semiring $\mathbb{K}$ instead of a valuation monoid.

We use the representation of words (trees, pictures, respectively) as relational structures and therefore as certain graphs as introduced in Section 2.2. As already stated in [Tho96], two significant differences to the previous models are the occurrence constraint and the possibly bigger tile-size. As a first step, following [Tho96], we give a sufficient condition to drop the occurrence constraint.

6.3.1 Reduction to WGA without Occurrence Constraint

In the following, we study a sufficient condition on graphs to reduce the occurrence constraint of a weighted graph automaton to the trivial constraint.

This gives us the following lemma which is a weighted version of Proposition 5.3 of [Tho96]. There, Thomas proved that we do not need the occurrence constraint in the case that our graphs are acyclic and have *indexed out-edges* and a *co-root*. In this context, a graph G has indexed out-edges if every vertex of G has one fixed outgoing edge which is uniquely determined by the edge-labeling. We say a graph G has a co-root if it has a vertex which can be reached from all other vertices of G.

The essential idea is that under these conditions, we can propagate which types of r-spheres we are encountering and collect and check this information at the co-root. The same holds for acyclic graphs with *indexed in-edges* and a *root*.

However, we can reformulate this idea in a slightly more general version as follows. For the idea above, we do not need our graphs to be strictly acyclic. It suffices that we encounter no cycles whenever we propagate the information along our designed out-edges. Furthermore, the out-indices can be modeled as a special edge-labeling E (which is given implicitly in the formulation above).

Making these out-indices explicit, we call a class of graphs $\mathcal{C} \subseteq \mathrm{DG}_t(A, B)$ *partial sortable* if there exists an element $b_0 \in B$ such that for every graph $G = (V, (\mathrm{Lab}_a)_{a\in A}, (E_b)_{b\in B})$ of $\mathcal{C}$, the subgraph $G' = (V, (\mathrm{Lab}_a)_{a\in A}, E_{b_0})$ is acyclic, connected, and every vertex of G' has at most one outgoing edge (and thus G' necessarily has a co-root).

Note that whenever G has a co-root, then we can find a selection of edges E such that $G' = (V, (\mathrm{Lab}_a)_{a\in A}, E)$ is acyclic, connected, and every vertex of G' has at most one outgoing edge. In this sense, the only essential requirement to drop the occurrence constraint is having a co-root or, applying the same arguments on incoming edges, having a root. However, finding such a selection of E may be harder than taking one special label b_0 and could be dependent on the representation of the graph itself.

Therefore, in the following quantitative version of Thomas' [Tho96] Proposition 5.3., we only consider the above defined partial sortable graphs. We say that a wGA $\mathcal{A} = (Q, \Delta, \mathrm{wt}, \mathrm{Occ}, r)$ is *without occurrence constraints*, if its occurrence constraint is always true, that is $\mathrm{Occ} = \mathit{true}$.

Lemma 6.38. *Let $\mathcal{C}$ be a partial sortable class of graphs. Let $S : \mathcal{C} \to K$ be a series and $\mathcal{A}$ a wGA with tile-size $r \geq 1$ with $[\![\mathcal{A}]\!] = S$. Then there exists a wGA $\mathcal{B}$ with the same tile-size as $\mathcal{A}$ and without occurrence constraints such that $[\![\mathcal{B}]\!] = S$.*

Note that if $\mathcal{A}$ has tile-size 0, then using Lemma 6.6 yields a wGA $\mathcal{A}'$ with tile size 1 and $[\![\mathcal{A}']\!] = [\![\mathcal{A}]\!]$. Hence, $\mathcal{B}$ can be constructed with tile-size 1 and without occurrence constraints.

Proof. We follow the construction of Thomas [Tho96] and add weights at the corresponding tiles. Let $E = E_{b_0}$ be the set of edges describing the partial order of a graph of $\mathcal{C}$. In the following, we only talk about edges of this set E. Let $\mathcal{A} = (Q, \Delta, \mathrm{wt}, \mathrm{Occ}, r)$ be a wGA with $\Delta = \{\tau_1, ..., \tau_k\}$. Let m be the biggest natural number that occurs in Occ as $\mathrm{occ}(\tau_i) \geq m$.

We construct $\mathcal{B} = (Q_\mathcal{B}, \Delta_\mathcal{B}, \mathrm{wt}_\mathcal{B}, \mathit{true}, r)$ with the following idea. We set $Q_\mathcal{B} = Q \times \{0, ..., m\}^k$. Then, for every tile τ in Δ, we put the same tile in $\Delta_\mathcal{B}$ and at every vertex we add the information of how often all tiles occurred so far. Locally, this information is checked at the center of every tile by summing

up the entries over all incoming edges and adding 1 for the type of the current tile in the specific entry. Additionally, if the center of a tile has no outgoing edges, we check that the accumulated number of tiles satisfies the occurrence constraint of $\mathcal{A}$.

Since every vertex has only one outgoing edge, we do not duplicate our counts. Furthermore, since G is acyclic, we eventually reach a vertex with no outgoing edge. Note that the co-root is not given explicitly, but it is the unique vertex without outgoing edges. Also, for the sake of convenience, we keep track of all tiles of Δ, instead of only keeping track of the tiles appearing in Occ.

More precisely, for a tile τ of $Q_{\mathcal{B}}$ around the center v, we refer with $\tau_{|Q}$ to the same tile where we forget about the additional labels. For a vertex w of τ, we refer with w_i to its i-th component of $\{0, ..., m\}^k$. We say $(w_1, ..., w_k)$ *satisfies* $\mathrm{occ}(\tau_i) \geq n$ if $\mathrm{occ}(w_i) \geq n$. Then, we set $\mathrm{wt}_{\mathcal{B}}(\tau) = \mathrm{wt}(\tau_{|Q})$ and define $\Delta_{\mathcal{B}}$ as

$$\Delta_{\mathcal{B}} = \left\{ \tau = (G, v) \;\middle|\; \begin{array}{l} \tau_{|Q} \in \Delta \text{ and} \\ v_i = \sum\limits_{\{w | E(w,v)\}} w_i \quad \text{, for all } i \text{ with } \tau_i \neq \tau, \text{ and} \\ v_i = \sum\limits_{\{w | E(w,v)\}} w_i + 1 \quad \text{, for } i \text{ with } \tau_i = \tau, \text{ and} \\ (v_1, ..., v_k) \text{ satisfies Occ if } \nexists \text{ vertex } z \text{ with } E(v, z) \end{array} \right\} .$$

Following the argumentation above, we can show that $[\![\mathcal{B}]\!] = [\![\mathcal{A}]\!]$. □

Note that words, trees, traces, pictures, and nested words can be seen as partial sortable graph classes[3].

6.3.2 Words, Trees, and Pictures

Definition 6.39. A *weighted finite automaton (wFA)* $\mathcal{A} = (Q, I, F, \delta, \mu)$ consists of a finite set of states Q, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, a set of transitions $\delta \subseteq Q \times A \times Q$, and a weight function $\mu : \delta \to K$. We define an *accepting run*, the *language of* $\mathcal{A}$, and *recognizable word series* as usual (cf. [DG07, DKV09, Sch61]).

To show the equivalence of graph acceptors and word automata over words in the unweighted case, Thomas [Tho91] proceeds using a translation of both models to logics. Here, we give a direct construction of the respective models, resulting in an alternative proof for the Büchi-theorem for words.

We show the following two results explicitly for words. This already will give the essential ideas to adapt the proofs to trees and pictures. Afterwards,

[3] As already noted in [Tho96], for pictures this would require a slight relabeling of the edges in the last column of a picture.

since nested words are less known and the respective proofs for them are more complex, we develop the results in detail also for nested words.

Lemma 6.40. *Let $S : A^* \to K$ be a word series and $\mathcal{A}$ a weighted finite automaton with $[\![\mathcal{A}]\!] = S$. Then there exists a weighted graph automaton $\mathcal{B}$ such that $[\![\mathcal{B}]\!](w) = S(w)$ for all words w over A.*

Proof. Given a weighted finite automaton $\mathcal{A} = (Q, I, F, \delta, \mu)$, we define the wGA $\mathcal{B} = (Q, \Delta, \mathrm{wt}, \mathit{true}, 1)$ as follows. In the following, for convenience, we denote the r-tiles over $A \times Q$ by $(a_{-r}q_{-r}, ..., a_{-1}q_{-1}, \boxed{a_0q_0}, a_1q_1, ..., a_rq_r)$ where a_0q_0 is the center and $a_i \in A$, $q_i \in Q$. Similarly, we denote smaller tiles.

For every transition $(q_1, a, q_2) \in \delta$ of $\mathcal{A}$, we add all tiles of the form $(a^*q_1, \boxed{aq_2}, a_*q_*)$ to the set of tiles Δ of $\mathcal{B}$, where $a^*, a_* \in A$ can be any symbols and $q_* \in Q$ any state. We set $\mathrm{wt}(a^*q_1, \boxed{aq_2}, a_*q_*) = \mu(q_1, a, q_2)$ for all $a^*, a_* \in A$ and $q_* \in Q$.

Note that a run of a wGA assigns states to every position of a word (seen as a graph), while a run $(q_0, ..., q_{|w|})$ of a classical wFA consists of one more state than the length of word. This is solved by 'shifting' our assignment half a position to the right and encoding the information of the initial state into the tiles with no predecessor[4]. Then, for each $(q_I, a, q_1) \in \delta$ with $q_I \in I$, we add all tiles $(\boxed{aq_1}, a_*q_*)$ to Δ. Similarly, for each $(q_1, a, q_F) \in \delta$ with $q_F \in F$, we add all tiles $(a_*q_1, \boxed{aq_F})$ to Δ. We set $\mathrm{wt}(\boxed{aq_1}, a_*q_*) = \sum_{q_I \in I} \mu(q_I, a, q_1)$ and $\mathrm{wt}(a_*q_1, \boxed{aq_F}) = \mu(q_1, a, q_F)$ for all $a_* \in A$, $q_* \in Q$. Then it is straightforward to show that $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$. □

We have thus shown that a word series recognizable by a wFA is also recognizable by a wGA. Using a similar construction, we can show that the same is true for trees, and pictures, and their respective automata models. Now, we turn to the converse, which is the more difficult, because we have to reduce the possible larger tile-size. Using Lemma 6.38, we can assume the wGA to have no occurrence constraint.

Lemma 6.41. *Let $S : A^* \to K$ be a word series and $\mathcal{A}$ be a weighted graph automaton without occurrence constraints and $[\![\mathcal{A}]\!](w) = S(w)$ for all words w over A. Then there exists a weighted finite automaton $\mathcal{B}$ such that $[\![\mathcal{B}]\!] = S$.*

Proof. Let $S : A^* \to K$ be a word series and $\mathcal{A} = (Q_\mathcal{A}, \Delta, \mathrm{wt}, \mathit{true}, r)$ a wGA with $[\![\mathcal{A}]\!] = S$. We construct the wFA $\mathcal{B} = (Q_\mathcal{B}, I, F, \delta, \mu)$, as follows.

Let I_0 be a new symbol. We set $Q_\mathcal{B} = \Delta \cup \{I_0\}$ and $I = \{I_0\}$. The main idea is to encode every accepting run of $\mathcal{A}$ on a word w as exactly one accepting run of $\mathcal{B}$, which uses the state τ if and only if the run of $\mathcal{A}$ applies τ at this position. Since $\mathcal{B}$ only 'sees' the symbol at the actual position, it guesses the

[4]Similarly, we could shift the assignment half a position to the left and encode the information of the final state into tiles with no successor.

next r labels and states (and if necessary the word length for small words) and checks this guess at the respective position later on.

We set $F = \{\tau \in \Delta \mid \tau = (a_{-k}q_{-k}, ..., a_{-1}q_{-1}, \boxed{a_0q_0}) \text{ for some } 0 \leq k \leq r, a_i \in A, q_i \in Q\}$.

Furthermore, we define $\delta = \delta_{I_0} \cup \delta_I \cup \delta_F \cup \delta_{\mathrm{s}} \cup \delta_{\mathrm{m}}$ as follows. We use the transitions δ_{I_0} to initially guess the first r labels and states. For words smaller than r, we have to guess the whole run. The transitions of δ_I, resp. δ_F, take care of the r-tiles which are used at the beginning (up to the first r positions), resp. at the end (up to the last r positions), of a word of length $\geq r$. At the beginning, resp. the end of word, the number of vertices in the tile increases, resp. decreases, by 1 with every transition. The next type of transitions δ_{s} takes care of r-tiles which are used at the middle of words of length $\leq 2r$. The last and main type of transitions δ_{m} uses tiles where the number of vertices left and right of the center is r. These transitions are used at the middle of words of length $> 2r + 1$. The detailed definitions of these transitions are the following.

$$\delta_{I_0} = \left\{ (I_0, a, \tau) \;\middle|\; \tau = (\boxed{aq_0}, a_1q_1, ..., a_kq_k) \text{ for some } 0 \leq k \leq r, \tau \in \Delta \right\} ,$$

$$\delta_I = \left\{ (\tau_1, a, \tau_2) \;\middle|\; \begin{array}{l} \tau_1 = (a_{-k}q_{-k}, ..., a_{-1}q_{-1}, \boxed{a_0q_0}, a_1q_1, ..., a_rq_r) \text{ and} \\ \tau_2 = (a_{-k}q_{-k}, ..., a_0q_0, \boxed{a_1q_1}, a_2q_2, ..., a_{r+1}q_{r+1}) \\ \quad \text{for some } 0 \leq k \leq r, a_1 = a, \tau_i \in \Delta \end{array} \right\} ,$$

$$\delta_F = \left\{ (\tau_1, a, \tau_2) \;\middle|\; \begin{array}{l} \tau_1 = (a_{-r}q_{-r}, ..., a_{-1}q_{-1}, \boxed{a_0q_0}, a_1q_1, ..., a_kq_k) \text{ and} \\ \tau_2 = (a_{-r+1}q_{-r+1}, ..., a_0q_0, \boxed{a_1q_1}, ..., a_kq_k) \\ \quad \text{for some } 0 \leq k \leq r, a_1 = a, \tau_i \in \Delta \end{array} \right\} ,$$

$$\delta_{\mathrm{s}} = \left\{ (\tau_1, a, \tau_2) \;\middle|\; \begin{array}{l} \tau_1 = (a_{-k}q_{-k}, ..., a_{-1}q_{-1}, \boxed{a_0q_0}, a_1q_1, ..., a_\ell q_\ell) \text{ and} \\ \tau_2 = (a_{-k}q_{-k}, ..., a_0q_0, \boxed{a_1q_1}, a_2q_2, ..., a_\ell q_\ell) \\ \quad \text{for some } 0 \leq \ell \leq r, 0 \leq k < r, a_1 = a, \tau_i \in \Delta \end{array} \right\} ,$$

$$\delta_{\mathrm{m}} = \left\{ (\tau_1, a, \tau_2) \;\middle|\; \begin{array}{l} \tau_1 = (a_{-r}q_{-r}, ..., a_{-1}q_{-1}, \boxed{a_0q_0}, a_1q_1, ..., a_rq_r) \text{ and} \\ \tau_2 = (a_{-r+1}q_{-r+1}, ..., a_0q_0, \boxed{a_1q_1}, ..., a_{r+1}q_{r+1}) \\ \quad \text{with } a_1 = a, \tau_i \in \Delta \end{array} \right\} .$$

Note that every tile τ_2 matches the previous tile τ_1 at every position except for positions with distance r to one of the centers. Furthermore, we force the symbol $a \in A$ that is read by a transition to be at the center of the next tile. Finally, we define the weight of all transitions by $\mu(\delta) = \mu(I_0, a, \tau_2) = \mu(\tau_1, a, \tau_2) = \mathrm{wt}(\tau_2)$.

We claim that $[\![\mathcal{A}]\!](w) = [\![\mathcal{B}]\!](w) = S(w)$ for all words over A. It is easy to see that every accepting run of $\mathcal{A}$ on a word w describes an accepting run of $\mathcal{B}$ on w if $\mathcal{B}$ correctly guesses the next r labels and states (or the end of the word, if the remainder of the word is shorter than r) used by the accepted run

or tiling of $\mathcal{A}$. Moreover, every run attained in this manner yields the correct weight since $\mathcal{B}$ uses exactly the same weights as $\mathcal{A}$ for every position of the word.

The nontrivial part is to see that this is the only possibility to get an accepting run of $\mathcal{B}$. Or equivalently that every accepting run of $\mathcal{B}$ yields an accepting run of $\mathcal{A}$. Let w be a word, let $|w|$ denote the length of w, and let $\rho = (I_0, \tau_1, ..., \tau_{|w|})$ be an accepting run of $\mathcal{B}$ on w. Then at every position of the run, $\mathcal{B}$ guesses how many positions ℓ of the word are remaining, up to the threshold $r \geq \ell$, and which symbols are at these positions. The number ℓ is encoded as number of vertices right of the center of the state τ_2.

The construction ensures the correctness of the symbols guessed (as long as we read the next ℓ transitions) because in every step a position with distance $k \leq \ell$ is brought one position nearer to the center, and therefore compared to the center after at most ℓ steps.

The correct guessing of the positions remaining is implicit: Assume that at some position $\mathcal{B}$ guesses the wrong number of positions remaining. If $\mathcal{B}$ guesses more positions than there are remaining, we cannot reduce the number of vertices right of the center of the next states to zero because at every position we can reduce that number only by one, and therefore we will not reach a final state.

If $\mathcal{B}$ guesses less positions than there are remaining, say $\ell < r$, we subsequently can only apply transitions of δ_F or δ_s. Therefore, at every transition, we reduce the number of vertices right of the center of the next state strictly by one. After ℓ steps, we cannot apply any more transitions because in $\mathcal{B}$ there exists no transition whose number of vertices right of the center of the next state is zero.

These guesses ensure that the state at every position $x \in \{1, ..., |w|\}$ mirrors the r-sphere around x, hence every accepting run $(I_0, \tau_1, ..., \tau_{|w|})$ of $\mathcal{B}$ yields an accepting run of $\mathcal{A}$. Again, the weights used by both automata in this case are the same and the claim follows. □

Together with Lemma 6.38, it follows that if a word series is recognizable by a weighted graph automaton, then it is also recognizable by a weighted finite automaton. Again, a similar construction can be applied to trees, and pictures, and their respective automata models defining recognizable series (cf. [DV06], [Fic11]). Note that this is possible because we know the 'structure' of these specific graph classes, which allows an automaton with tile-size 1 to guess all labels and states in an r-tile around a position, as done in Lemma 6.41 for words. Thus, we can reduce the tile-size of a weighted graph automaton over these specific graphs from r to 1.

In general, a reduction to 1-tiles is not possible: For example, as already noted by Thomas [Tho96] in the unweighted setting, the language of all grids in the class of all graphs is recognizable by a GA with tile-size 2, but is

not recognizable by a GA with tile-size 1. A short argument for this is the following. Suppose we have a GA $\mathcal{A}$ with tile-size 1 recognizing all pictures. Then there exists a picture big enough such that we have to use the same tile at two different positions. Hence, we can swap the outgoing edges of these two positions in our picture. We get a graph which is still accepted by $\mathcal{A}$ (using the exact same tiles as before) but no picture anymore, a contradiction. Already in the unweighted setting, it is stated by Thomas [Tho96] as an open question to precisely describe the class of graphs where the use of 1-tiles suffices.

Using the results above together with Theorem 6.24, we get the following corollary. For the detailed definition of weighted tree automata (wTA) and logic, see [DV06] (pp. 231, 234). For weighted picture automata (wPA), see [Fic11] (Sections 3 and 4).

Corollary 6.42. *Let $\mathbb{K}$ be a commutative semiring and $S : A^* \to K$ be a word series. Then the following are equivalent:*

1. *S is recognizable by a wGA.*
2. *S is recognizable by a wGA with tile-size one and without occurrence constraint.*
3. *S is recognizable by a wFA.*
4. *S is definable by an FO-restricted* $\mathrm{MSO}(\mathbb{K})$*-sentence.*

If we replace "word" by "tree" or "picture" and $S : A^* \to K$ by $S : T_A \to K$ or $S : A^{**} \to K$, respectively, a similar result holds for weighted tree automata (wTA) and weighted picture automata (wPA), and their respective logics.

Note that our implication (4) $\Rightarrow$ (3) is a slightly weaker version of the result in [DG07] since in our logic, we can apply Val_x-quantification only to almost FO-boolean formulas, whereas in [DG07, DG09], we can use almost boolean formulas. As shown before, this difference originates from the fact that for words, EMSO equals MSO, which is neither true for pictures nor for graphs. In this sense, our result captures the 'biggest logic possible', if we want to include pictures.

6.3.3 Nested Words

In the following, we apply Theorem 6.24 to the previously studied structure of nested words. In contrast to Chapter 4, we focus on finite nested words which are well-matched (i.e. they have no pending calls or pending returns). We give a direct transformation of weighted graph automata to weighted nested word automata, which shows that the characterization of weighted regular languages of nested words in the form of a weighted MSO-logic shown in [Mat10b, DP14b] is also a consequence of Theorem 6.24. Following Section 2.2 and our definition of graphs as relational structures, we introduce the class of *nested words* $\mathrm{NW}(A) \subseteq \mathrm{DG}_3(A, B)$ where $B = \{lin, hier\}$, as follows. We

refer with $s = E_{\text{lin}}$ to the linear edges and with $\nu = E_{\text{hier}}$ to the hierarchical edges. Then, a nested word is a graph $nw = (V, (\text{Lab}_a)_{a\in A}, s, \nu)$ such that $V = \{1, ..., \ell\}$, s is the known successor relation, and ν is the *nesting relation* satisfying for all $i, j \in V$:

1. $(i, j) \in \nu \Rightarrow i < j$,
2. $1 \leq i \leq \ell \Rightarrow |\{j \mid (i, j) \in \nu\}| \leq 1 \wedge |\{i \mid (i, j) \in \nu\}| \leq 1$, and
3. $(i, j) \in \nu \wedge (i', j') \in \nu \wedge i < i' \Rightarrow j < i' \vee j > j'$ (nesting edges do not cross).

In short, we refer to a nested word as $nw = (w, \nu)$, where $w \in A^+$ and ν is the nesting relation.

Note that $\text{NW}(A)$ as a subclass of $\text{DG}_t(A, B)$ is not recognizable by a GA. Similarly, it cannot be checked by an EMSO sentence whether a relation is a nesting relation. In contrast, the complement $\text{DG}_t(A, B) \setminus \text{NW}(A)$ is recognizable by a GA.

Definition 6.43 ([AM09], [DD14], [Mat10b])**.** A *weighted nested word automaton (wNWA)* over the alphabet A and the semiring $\mathbb{K}$ is a quadruple $\mathcal{A} = (Q, I, \delta, F)$, where $\delta = (\delta_{\text{call}}, \delta_{\text{int}}, \delta_{\text{ret}})$, consisting of:

- a finite set of states Q,
- a set of initial states $I \subseteq Q$,
- a set of final states $F \subseteq Q$,
- the weight functions $\delta_{\text{call}}, \delta_{\text{int}} : Q \times A \times Q \to K$,
- the weight function $\delta_{\text{ret}} : Q \times Q \times A \times Q \to K$.

An accepting *run* ρ of the wNWA $\mathcal{A}$ on the nested word $nw = (a_1 a_2 ... a_\ell, \nu)$ is a sequence of states $\rho = (q_0, q_1, ..., q_\ell)$ where $q_i \in Q$ for each $i \in \{0, ..., \ell\}$, q_0 is the initial state of $\mathcal{A}$, and $q_\ell \in F$. We denote by $\text{wt}_{\mathcal{A}}(\rho, nw, i)$ the weight of the transition of ρ used at position $i \in \mathbb{N}_+$, defined as follows

$$\text{wt}_{\mathcal{A}}(\rho, nw, i) = \begin{cases} \delta_{\text{call}}(q_{i-1}, a_i, q_i) & , \text{ if } \nu(i, j) \text{ for some } j > i \\ \delta_{\text{int}}(q_{i-1,}, a_i, q_i) & , \text{ if } i \text{ is an internal} \\ \delta_{\text{ret}}(q_{i-1}, q_{j-1}, a_i, q_i) & , \text{ if } \nu(j, i) \text{ for some } 1 \leq j < i \ . \end{cases}$$

We define the *behavior of the automaton* $\mathcal{A}$ as the function $[\![\mathcal{A}]\!] : \text{NW}(A) \to K$ given by (where as usual, empty sums are defined to be 0)

$$[\![\mathcal{A}]\!](nw) = \sum_{\rho \text{ acc. run of } \mathcal{A} \text{ on } nw} \prod_{1 \leq i \leq \ell} \text{wt}_{\mathcal{A}}(r, nw, i) \ .$$

We call every function $S : \text{NW}(A) \to K$ a *nested word series*. We say that a wNWA $\mathcal{A}$ *recognizes* a series S if $[\![\mathcal{A}]\!] = S$. We call a series S *recognizable* (by a wNWA) if there exists a wNWA $\mathcal{A}$ accepting it.

Now, we show that within the special class of nested words, the use of weighted graph automata with tiles of size 1 and without occurrence constraints is sufficient. Consequently, we can show the equivalence of wNWA and wGA on the class of nested words and get a Büchi-like result for wNWA and a respective weighted logic, cf. [Mat10b].

As noted before, nested words are a partial sortable class of graphs where the classical edges defined by the successor relation give an ordering and a root. Therefore, we can use Lemma 6.38 to omit the occurrence constraint. The next statement gives the reduction to 1-tiles (compare ideas with Lemma 6.41).

Lemma 6.44. *Let $S : \mathrm{NW}(A) \to K$ be a nested word series and $\mathcal{A}$ be a wGA without occurrence constraint and $[\![\mathcal{A}]\!] = S$. Then there exists a wGA $\mathcal{B}$ with tile size 1 and without occurrence constraint such that $[\![\mathcal{B}]\!] = S$.*

Proof. Let $\mathcal{A} = (Q, \Delta, \mathrm{wt}, \mathit{true}, r)$ be a wGA with $[\![\mathcal{A}]\!] = S$. We construct the wGA $\mathcal{B} = (Q_\mathcal{B}, \Delta_\mathcal{B}, \mathrm{wt}_\mathcal{B}, \mathit{true}, 1)$ as follows.

We set $Q_\mathcal{B} = \Delta$. We let $A' = Q_\mathcal{B} \times A = \Delta \times A$ be the extended tile-alphabet, and refer with $\mathrm{Lab}_\Delta(w) \in \Delta$ and $\mathrm{Lab}_A(w) \in A$ to the respective labels of a vertex w of a tile over A' and $B = \{\mathit{lin}, \mathit{hier}\}$. We set $\mathrm{wt}_\mathcal{B}(H, v) = \mathrm{wt}(\mathrm{Lab}_\Delta(v))$.

To define Δ_B, which is a set of 1-tiles over A' and B, the main idea is the following: In every smaller tile the automaton makes a 'guess' on how the bigger tiles look like. Then, the following consistency property makes sure that this guess cannot be changed except at the border of the r-spheres. Subsequently, this guess is checked stepwise at the next r positions.

To make this precise, we need the following notions. Let $\tau = (H, v) = (V, (\mathrm{Lab}_a)_{a \in A'}, s, \nu, v)$ be a 1-tile over A' and B around the center $v \in V$, and let $w \in V$. Let $(\tau_v, a_v), (\tau_w, a_w) \in \Delta \times A$ such that $(\tau_v, a_v) = (\mathrm{Lab}_\Delta(v), \mathrm{Lab}_A(v))$ and $(\tau_w, a_w) = (\mathrm{Lab}_\Delta(w), \mathrm{Lab}_A(w))$. Let $(G_v, c_v) = \tau_v$ and $(G_w, c_w) = \tau_w$ be these two tiles of Δ, together with their respective centers. Then, we denote by $\mathrm{Lab}^{\mathrm{c}}_A(\tau_v)$ the A-label at the center of τ_v. Further, for a vertex x of τ_v, we denote by $\mathrm{sph}^r_{c_v}(\tau_v, x)$ the r-sphere of τ_v around x that is pointed with c_v instead of x. Finally, we say τ is *consistent in itself* if for all v, it holds that

$$\mathrm{Lab}^{\mathrm{c}}_A(\tau_v) = a_v \ ,$$

and for all v, w with $(v, w) \in E$, $E \in \{s, \nu\}$, there exists a vertex x of τ_v and a vertex y of τ_w with $(c_v, x) \in E$ and $(y, c_w) \in E$ such that

$$\mathrm{sph}^r_{c_v}(\tau_v, x) \text{ is isomorphic to } \mathrm{sph}^r(\tau_w, y) \ .$$

Then we define Δ_B as the set of all 1-tiles over A' and B which are consistent in itself. Using similar arguments to the proof of Lemma 6.41, it follows that every accepting run of $\mathcal{A}$ on a graph G gives an accepting run of $\mathcal{B}$ on G with the same weights, and vice versa. It follows that $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$. □

Lemma 6.45. *Let $S : \mathrm{NW}(A) \to K$ be a nested word series and $\mathcal{B}$ a wGA with tile size 1, without occurrence constraint, and $[\![\mathcal{B}]\!](nw) = S(nw)$ for all $nw \in \mathrm{NW}(A)$. Then there exists a wNWA $\mathcal{C}$ with $[\![\mathcal{C}]\!] = S$.*

Proof. Using ideas of Lemma 6.41 and Lemma 6.44, this is a straightforward construction. Therefore, we only sketch the proof as follows. As before, we use the tiles of $\mathcal{B}$ as states of $\mathcal{C}$. The distinction between calls, internals, and returns is handled by checking whether the center of a tile has an outgoing hierarchical edge, no hierarchical edges, or an incoming hierarchical edge, respectively. Consider a position x of a nested word nw. Since the wGA has tile-size 1, the wNWA $\mathcal{C}$ only has to guess the label and the state at the position $x + 1$. If the current position is a call, then $\mathcal{C}$ also guesses the label and the state at the position y with $\nu(x, y)$. This hierarchical guess is later controlled by the tile of the transition used at the position y. As before, the initial (resp. final) states are characterized by tiles without incoming (resp. outgoing) linear edges at the center.

It follows that by guessing correctly, $\mathcal{C}$ can assure that for every accepting run of $\mathcal{B}$ on a nested word nw there exists an accepting run of $\mathcal{C}$ on nw with the same weights as $\mathcal{B}$. Conversely, as in Lemma 6.41, we see that $\mathcal{C}$ can never guess wrong, i.e., for every accepting run of $\mathcal{C}$ there exists an accepting run of $\mathcal{B}$ with the same weight. Note that this is only true because we know that nesting edges do not cross. Therefore, whenever we encounter a return, i.e, an incoming hierarchical edge, we know that this hierarchical edge comes from the latest open call. □

Applying the definition for a weighted MSO logic as in Section 3.3 to nested words and a semiring yields the weighted logic $\mathrm{MSO}(\mathbb{K}, \mathrm{NW}(A))$ (cf. also Section 4.4 and [Mat10b]). Then as a consequence of Lemma 6.44, Lemma 6.45, and our main Theorem 6.24, we get the following result.

Corollary 6.46. *Let $\mathbb{K}$ be a commutative semiring and $S : \mathrm{NW}(A) \to K$ be a nested word series. Then the following are equivalent:*

1. *S is recognizable by a wGA.*
2. *S is recognizable by a wGA with tile-size one and without occurrence constraint.*
3. *S is recognizable by a wNWA.*
4. *S is definable by an FO-restricted $\mathrm{MSO}(\mathbb{K}, \mathrm{NW}(A))$-sentence.*
5. *S is definable by an FO-restricted $\mathrm{MSO}(\mathbb{K})$-sentence.*

6.4 Automata and Logics on Infinite Graphs

In this chapter, we show how to extend our results in both the unweighted and the weighted setting to infinite graphs. We benefit from the flexible presentation

of our previous results, which allows us to adapt them by concentrating only on the crucial differences to the finite case. Note that both the *infinite graph acceptors* and the *weighted infinite graph automata* are finite models which operate on infinite graphs.

6.4.1 Graph Acceptors for Infinite Graphs

In the following, we extend Theorem 6.2 to the infinite setting, thus showing a Büchi-like result for infinite graphs. We introduce infinite graph acceptors with an extended acceptance condition and an EMSO^∞ logic as introduced and studied in Section 2.4. This logic extends classical MSO by a first-order quantification $\exists^\infty x.\varphi$ expressing that there exist infinitely many vertices satisfying φ.

Using the previous occurrence constraint as an acceptance condition, Thomas' graph acceptors of Section 6.1 could also be interpreted as a model for infinite graphs. However, every occurrence constraint only checks for occurrences up to a certain threshold, i.e., it cannot express that a tile occurs infinitely often. Therefore, graph acceptors cannot distinguish between finitely and infinitely many occurrences of a state, which is a very important asset for models operating on infinite structures. This motivates the following definition.

Definition 6.47. An *infinite graph acceptor (GA^∞)* $\mathcal{A}$ over the alphabets A and B is defined as a quadruple $\mathcal{A} = (Q, \Delta, \mathrm{Occ}, r)$ where

- Q, Δ, r are defined as before, and
- Occ, the *extended occurrence constraint*, is a boolean combination of formulas "$\mathrm{occ}(\tau) \geq n$" and "$\mathrm{occ}(\tau) = \infty$", where $n \in \mathbb{N}$ and $\tau \in \Delta$.

The notions of an *accepting run* ρ of $\mathcal{A}$ on $G \in \mathrm{DG}_t^\infty(A, B)$ and a *recognizable language* $L = L(\mathcal{A}) \subseteq \mathrm{DG}_t^\infty(A, B)$ are defined as before. The additional check "$\mathrm{occ}(\tau) = \infty$" in particular can ask whether some state occurs infinitely often. Therefore, together with boolean combinations of these statements, we can encode classical Büchi and Muller acceptance conditions into the extended occurrence constraint.

As in Section 2.4 and in [BK07], we introduce $\mathrm{MSO}^\infty(\mathrm{DG}_t^\infty(A, B))$, short MSO^∞, by the following grammar

$$\varphi ::= \mathrm{Lab}_a(x) \mid E_b(x, y) \mid x = y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x.\varphi \mid \exists^\infty x.\varphi \mid \exists X.\varphi$$

The semantics and the fragments FO^∞ and EMSO^∞ of this logic are defined as previously. In Section 2.4, by employing result from [BK07] and [Tho91], we have proven Corollary 2.6 stating the following: For every FO^∞-sentence φ, there exists an extended occurrence constraint Occ such that for all $G \in \mathrm{DG}_t^\infty(A, B)$ it holds that $G \models \varphi$ if and only if $G \models \mathrm{Occ}$. This result provides us with the means to prove our first main theorem of this section.

Theorem 6.48. *Let $L \subseteq \mathrm{DG}_t^\infty(A,B)$ be a set of infinite graphs. Then:*

1. *L is recognizable by a one-state GA^∞ iff L is definable by an FO^∞-sentence.*
2. *L is recognizable iff L is definable by an EMSO^∞-sentence.*

Proof. The first statement is proven as follows. Given a one-state GA^∞ $\mathcal{A} = (\{q\}, \Delta, \mathrm{Occ}, r)$, we get an equivalent GA^∞ $\mathcal{A}' = (\{q\}, \Delta', \mathrm{Occ}', r)$, with $\Delta' = \{\tau \mid \tau \text{ is an } r\text{-tile}\}$ and $\mathrm{Occ}' = \mathrm{Occ} \wedge \bigwedge_{\tau \notin \Delta} \mathrm{occ}(\tau) = 0$. Furthermore, Occ' can be expressed by an equivalent FO^∞-formula φ. Then a graph $G \in \mathrm{DG}_t^\infty(A,B)$ is in $L(\mathcal{A}')$ if and only if G satisfies Occ' if and only if G satisfies φ.

For the converse, given an FO^∞-sentence φ, we apply Corollary 2.6 (2.), to get an equivalent extended occurrence constraint Occ over tiles of radius r. Then $\mathcal{A} = (\{q\}, \Delta, \mathrm{Occ}, r)$, with $\Delta = \{\tau \mid \tau \text{ is an } r\text{-tile}\}$, satisfies $L(\mathcal{A}) = L(\varphi)$.

The second statement is proven as follows. Given a graph automaton $\mathcal{A}$ with m states and an FO-formula ψ that describes the occurrence constraint of $\mathcal{A}$, we can construct an EMSO-sentence $\exists X_1 ... \exists X_m . \psi'$ that guesses the state used at every vertex, and $\psi' = \psi \wedge \theta$, such that θ ensures that the sphere around every vertex is isomorphic to a tile of $\mathcal{A}$. For the converse, given an EMSO-sentence $\exists X_1 ... \exists X_m . \psi$, we define $\mathcal{A} = (\{0,1\}^{\{1,...,m\}}, \Delta, \mathrm{Occ}, r)$, with $\Delta = \{\tau \mid \tau \text{ is an } r\text{-tile}\}$, where the states of $\mathcal{A}$ encode all possible assignments of $X_1 ... X_m$, and Occ describes ψ. □

6.4.2 A Büchi Result for WGA on Infinite Graphs

In the following, we extend our results in the weighted setting to infinite graphs. We utilize the ∞-(graph)-valuation monoids of Section 3.2 to introduce a model of weighted graph automata which is operating on infinite graphs. We recall how to modify our valuation function to read infinite graphs as follows.

Definition 6.49. An *∞-(graph)-valuation monoid* $(D, +, \mathrm{Val}^\infty, 0)$ consists of a complete monoid $(D, +, 0)$ together with an *∞-valuation function* $\mathrm{Val}^\infty : \mathrm{DG}_t^\infty(D, B) \to D$ with $\mathrm{Val}^\infty(G) = 0$ if at least one vertex v of G is labeled with 0.

A *product ∞-valuation monoid (∞-pv-monoid)* $\mathbb{PD} = (D, +, \mathrm{Val}^\infty, \diamond, 0, 1)$ is an ∞-valuation monoid $(D, +, \mathrm{Val}^\infty, 0)$ with a constant $1 \in D$ and an operation $\diamond : D^2 \to D$ satisfying $0 \diamond d = d \diamond 0 = 0$ and $1 \diamond d = d \diamond 1 = d$ for all $d \in D$.

For ∞-pv-monoids, we define the respective notion of a *left-distributive pv-monoid* and a *cc-∞-valuation semiring* as in Chapter 3.2. Note that the distributivity of $\diamond$ over $+$ now also includes infinite sums.

Example 6.11. Let $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$ and $-\infty + \infty = -\infty$. Let (G, u) be a pointed graph labeled with values of $\bar{\mathbb{R}}$. We denote by $|\operatorname{sph}^r(G, u)|$ the

number of vertices in the r-sphere of G around u. Then we define $\mathbb{PD}_1 = (\bar{\mathbb{R}}, \sup, \lim\,\mathrm{avg}, +, -\infty, 0)$ with

$$\lim\,\mathrm{avg}(G,u) = \liminf_{r\to\infty} \frac{1}{|\,\mathrm{sph}^r(G,u)|} \sum_{\mathrm{dist}(v,u)\leq r} \mathrm{Lab}_G(v) \ .$$

Now, let $\bar{\mathbb{R}}_+ = \{x \in \mathbb{R} \mid x \geq 0\} \cup \{\infty, -\infty\}$ and $-\infty + \infty = -\infty$. Let (G,u) be a pointed graph labeled with values of $\bar{\mathbb{R}}_+$. Let $t > 1$ be the maximal degree of our graphs and $0 < \lambda < \frac{1}{t-1}$. We define $\mathbb{PD}_2 = (\bar{\mathbb{R}}_+, \sup, \mathrm{disc}^\infty_\lambda, +, -\infty, 0)$, with

$$\mathrm{disc}^\infty_\lambda(G,u) = \lim_{n\to\infty} \sum_{r=0}^{n} \sum_{\mathrm{dist}(v,u)=r} \lambda^r\,\mathrm{Lab}_G(v) \ .$$

Then $\mathbb{PD}_1$ is a left-$+$-distributive and left-Val^ω-distributive ω-valuation monoid. Furthermore, $\mathbb{PD}_2$ is a left-multiplicative cc-∞-valuation semiring. $\diamond$

Definition 6.50. A *weighted infinite graph automaton (wGA$^\infty$)* over A, B, and $\mathbb{D}$ is a tuple $\mathcal{A} = (Q, \Delta, \mathrm{wt}, \mathrm{Occ}, r)$ where

- $\mathcal{A}' = (Q, \Delta, \mathrm{Occ}, r)$ is an infinite graph acceptor over A and B,
- $\mathrm{wt} : \Delta \to D$ is the weight function assigning to every tile of Δ a value of D.

We transfer the previous notions of *accepting run* and *recognizable series*.

Furthermore, the weighted MSO-logic for infinite graphs and its fragments are defined as extensions of MSO^∞ as in the finite case in Section 6.2.3 (using Val^∞ instead of Val) and denoted by $\mathrm{MSO}^\infty(\mathbb{D})$, resp. $\mathrm{MSO}^\infty(\mathbb{PD})$. The significant difference is again that we have the additional operator $\exists^\infty x$ in our underlying unweighted fragment.

Using Theorem 3.6 and adapting our previous results to the infinite setting, we get our second main result of this section.

Theorem 6.51. *Let $\mathbb{D} = (D, +, \mathrm{Val}, 0)$ be a regular ∞-valuation monoid and let $S : \mathrm{DG}^\infty_t(A,B) \to D$ be a series. Then*

1. *S is recognizable iff S is definable by an FO-restricted $\mathrm{MSO}^\infty(\mathbb{D})$-sentence.*

Let $\mathbb{PD}$ be an ∞-pv-monoid and let $S : \mathrm{DG}^\infty_t(A,B) \to D$ be a series.

2. *Let $\mathbb{PD}$ be regular. Then S is recognizable iff S is definable by a strongly-FO-restricted $\mathrm{MSO}^\infty(\mathbb{PD})$-sentence.*
3. *Let $\mathbb{PD}$ be left-distributive. Then S is recognizable iff S is definable by a FO-restricted $\mathrm{MSO}^\infty(\mathbb{PD})$-sentence.*
4. *Let $\mathbb{PD}$ be a cc-valuation-semiring. Then S is recognizable iff S is definable by a weakly-FO-restricted $\mathrm{MSO}^\infty(\mathbb{PD})$-sentence.*

Proof. As we are mainly following the proof of Theorem 6.25, we only note the differences to before.

To increase the tile-radius within the occurrence constraint, we now also have to rewrite "$\operatorname{occ}(\tau) = \infty$". This is done by the following formula, where $\tau^* = (\tau_1, ..., \tau_m)$ (cf. Formula 6.1)

$$\big(\sum_{\tau \in \tau^*} \operatorname{occ}(\tau)\big) = \infty \quad \text{is defined as a shorthand for} \quad \bigvee_{i=1,...,m} \operatorname{occ}(\tau_i) = \infty \ .$$

Closure under $+$, projections, and intersection with FO-definable languages are proven as before. From these results the closure under $\oplus$, $\beta?\varphi_1 : \varphi_2$, resp. $\otimes$, and under the quantifications $\bigoplus_x$ and $\bigoplus_X$ of our logics follows.

A notable difference is found in the closure under $\operatorname{Val}_x \varphi$ (in previous papers, e.g. [DG07], the weighted universal quantification). As in the finite case, we set $\mathcal{W} = \operatorname{free}(\operatorname{Val}_x \varphi) \cup \{x\}$ and require φ to be an almost FO-boolean formula, thus $[\![\varphi]\!]$ is an FO^∞-step function, i.e., $[\![\varphi]\!] = \sum_{i=1}^m d_i \mathbb{1}_{L_i} = \sum_{i=1}^m d_i \cap L_i$, where L_i are FO^∞-definable languages forming a partition of $\mathrm{DG}_t^\infty(A_\mathcal{W}, B)$. Since we have to deal with the additional quantifier $\exists^\infty x$, we cannot apply Theorem 6.2. However, Theorem 6.48 gives us the one-state infinite graph acceptors $\mathcal{A}_i$ recognizing L_i. Then the encodings and automata constructions of Proposition 6.20 give us a wGA^∞ $\mathcal{A}$ with $[\![\mathcal{A}]\!] = [\![\operatorname{Val}_x \varphi]\!]$. Together with the other closure properties, this shows that a restricted formula yields a recognizable series.

For the converse, we construct a sentence as in the proof of Proposition 6.23, where we encode the extended occurrence constraint into the FO^∞-formula describing all accepting runs.

The statements 2 to 4 follow from the respective version of Theorem 3.6. □

Note that while FO^∞ differs from FO, the proof of Theorem 3.6 never takes the particular first order operators into account. So, this theorem stays true even if we extend the underlying unweighted logic of MSO($\mathbb{PD}$) from MSO to MSO^∞. Then, *FO-boolean* formulas are still defined as MSO^∞-formulas without second order existential quantification. Thus, they contain the new operator $\exists^\infty x.\varphi$.

Chapter 7

Conclusion

We introduced weighted automata models for infinite nested words, operator precedence languages and rank-bounded graphs. As weight structures, we employed semirings and valuation monoids, a general structure which can model non-semiring operations like average or discounting [DM12]. We showed various closure properties for the resulting weighted languages and showed the following two major connections to previously studied weighted languages: Firstly, weighted operator precedence languages strictly include weighted visibly pushdown languages, also called weighted languages of nested words. Secondly, our weighted graph automata are able to simulate several weighted automata models for finite structures like words, trees, pictures, and nested words.

Furthermore, we introduced suitable weighted MSO logics for all of these structures either using weights of a semiring or a valuation monoid (cf. [DG07] for a semiring-weighted MSO for words). We could show that under suitable assumptions on the valuation monoids, two, resp. three, fragments of the weighted logic have the same expressive power with efficient conversions into the smallest fragment, which gives a kind of normal form for weighted formulas, in particular in the case of semirings. We explicitly employed these results for nested words and graphs, but they can also be applied to other discrete structures like trees and pictures. In our main results, we showed that the weighted automata and their respective weighted MSO logic have the same expressive power.

As in [AM09], we considered nested words possibly containing pending edges. We remark that our results similarly can be developed for finite nested words, which would give us results of [Mat10b] and [DP14b] as special cases. There exist different techniques to transform finite nested words into hedges, unranked trees, or binary trees, and vice versa. This provides the possibility to regard nested words as a special form of trees and nested word automata as special form of tree automata, cf. [AAB$^+$08, AM09]. A generalization of Büchi's theorem to trees and infinite trees was done in [Don70, Rab69, TW68]. Recently, the connection between weighted tree automata and weighted logics

has also found much interest, cf. [DGMM11, DV06, DV11]. However, the results in the quantitative case using valuation monoids cover only finite trees and cannot be used to draw conclusions for infinite nested words. Moreover, it is not clear how to translate infinite trees into infinite nested words. Another interesting research path would be to investigate decision problems for weighted nested word automata. This was done, for example, for automata on words and with average or discounted computations of weights in [CDH08, CDH09].

For operator precedence languages, we considered commutative and general semirings. In a Nivat-like result, we showed that the behavior of a wOPA can be described as homomorphic image of the behavior of a particularly simple wOPA together with an unweighted OPA. We showed that weighted operator precedence automata (wOPA) without pop weights and a restricted weighted MSO logic have the same expressive power. If the semiring is commutative, this result also applies to wOPA with arbitrary pop weights. This raises the problem to find, for arbitrary semirings and for wOPA with pop weights, both an expressively equivalent weighted MSO logic and a Nivat-like result. In [DV06], very similar problems occurred for weighted automata on unranked trees and weighted MSO logic. Very recently, in [DHV15], an equivalence result for a restricted weighted MSO logic could be derived with another definition of the behavior of weighted unranked tree automata. Thus, the following question arises: Is there another definition of the behavior of wOPA (with pop weights) which makes them expressively equivalent to our restricted weighted MSO logic?

In [LMPP15], operator precedence languages of infinite words were investigated and shown to be of practically relevance. Hence, it would be interesting to develop wOPA operating on infinite words. Another interesting question is how to extend our results from semirings to valuation monoids. Since already in the case of semirings, we have seen that there is a significant difference regarding the logical characterization between commutative and non-commutative semirings, this promises to be an intriguing challenge. As valuation monoids are especially useful to model long-term behaviors, they may also prove suitable for investigating infinite precedence words.

In the central chapter of this thesis, we introduced a weighted generalization of Thomas' graph acceptors [Tho91, Tho96] and a suitable weighted logic on graphs. We showed that weighted word, tree, picture, or nested word automata are special instances of these general weighted graph automata. Additionally, we gave several examples that our weighted graph automata can recognize particular quantitative properties of graphs which were not covered by previous automata models. Using valuation monoids, we proved a Nivat-like and a Büchi-like characterization of the expressive power of weighted graph automata. This gives us results of [DG07], [DV06], [Fic11], and [Mat10b] as corollaries (under appropriate restrictions to the respective logic). Although not considered explicitly, we conjecture that similar equivalence results also hold for other

finite structures like traces [Mei06], texts [Mat10a], and distributed systems [BM07] and their respective automata models. Similarly, we conjecture that the respective models using valuation monoids are special instances of weighted graph automata, which would give us results of [DM12, DGMM11, BD15] as corollaries. Further research could investigate applications of weighted graph algorithms in order to develop decision procedures for weighted graph automata.

Finally, we showed how to extend these results to infinite graphs. Utilizing results by Hanf [Han65], Thomas [Tho91], and Bollig and Kuske [BK07], we proved a Büchi-like theorem for infinite graphs. We gave new examples for this model, employing average and discounting. Among others, infinite graphs cover infinite words [DR06], infinite trees [Rah07], and infinite traces [DG00]. It would be interesting how to incorporate previous models for these infinite structures with weighted graph automata operating on infinite graphs.

Chapter 8

Acknowledgments

I want to thank Manfred Droste for supervising this thesis, especially for his continuous supply of research ideas and his support in finding solutions to arising problems. Also, I want to thank Heiko Vogler for his support as the second supervisor of this thesis.

I thank Tobias Weihrauch, Erik Paul, and Andreas Maletti for many helpful scientific discussions and occasionally helping me to put my thoughts in order. Also, I would like to thank Wolfgang Thomas, Paul Gastin, Nicole Schweikhardt, Dietrich Kuske, and Benjamin Monmege for enlightening scientific discussions on specific topics of this thesis.

Furthermore, I am grateful for the financial support of the Deutsche Forschungsgemeinschaft (DFG) through the project DR 202/11-1 and even more importantly, through the Graduiertenkolleg QuantLA. Also, I would like to thank all my colleagues at the University of Leipzig, and all students and associates of QuantLA for providing a very fruitful research environment.

Last but not least, I am deeply indebted to my family for their moral support throughout my whole studies and the work on this thesis.

Bibliography

[AAB+08] Rajeev Alur, Marcelo Arenas, Pablo Barceló, Kousha Etessami, Neil Immerman, and Leonid Libkin. First-order and temporal logics for nested words. *Logical Methods in Computer Science*, 4(4):1–44, 2008.

[AF90] Miklós Ajtai and Ronald Fagin. Reachability is harder for directed than for undirected finite graphs. *J. Symb. Log.*, 55(1):113–150, 1990.

[AF16] Rajeev Alur and Dana Fisman. Colored nested words. In Dediu et al. [DJMT16], pages 143–155.

[AGMR06] Marcella Anselmo, Dora Giammarresi, Maria Madonia, and Antonio Restivo. Unambiguous recognizable two-dimensional languages. *ITA*, 40(2):277–293, 2006.

[AM09] Rajeev Alur and Parthasarathy Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3):16:1–16:43, 2009.

[BD15] Parvaneh Babari and Manfred Droste. A Nivat theorem for weighted picture automata and weighted MSO logic. In Adrian-Horia Dediu, Enrico Formenti, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications, LATA 2015*, volume 8977 of *LNCS*, pages 703–715. Springer, 2015.

[BG09] Benedikt Bollig and Paul Gastin. Weighted versus probabilistic logics. In Volker Diekert and Dirk Nowotka, editors, *Developments in Language Theory, DLT 2009*, volume 5583 of *LNCS*, pages 18–38. Springer, 2009.

[BGMZ14] Benedikt Bollig, Paul Gastin, Benjamin Monmege, and Marc Zeitoun. Pebble weighted automata and weighted logics. *ACM Trans. Comput. Log.*, 15(2):15, 2014.

[BK07] Benedikt Bollig and Dietrich Kuske. Muller message-passing automata and logics. In Remco Loos, Szilárd Zsolt Fazekas, and

Carlos Martín-Vide, editors, *Language and Automata Theory and Applications, LATA 2007*, volume Report 35/07, pages 163–174. Research Group on Mathematical Linguistics, Universitat Rovira i Virgili, Tarragona, 2007.

[BM07] Benedikt Bollig and Ingmar Meinecke. Weighted distributed systems and their logics. In *Logical Foundations of Computer Science*, volume 4514 of *LNCS*, pages 54–68. Springer, 2007.

[BR88] Jean Berstel and Christophe Reutenauer. *Rational Series and Their Languages*, volume 12 of *EATCS Monographs in Theoretical Computer Science*. Springer, 1988.

[Büc60] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik und Grundlagen Math.*, 6:66–92, 1960.

[CDH08] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. In Michael Kaminski and Simone Martini, editors, *Computer Science Logic*, volume 5213 of *LNCS*, pages 385–400. Springer, 2008.

[CDH09] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Expressiveness and closure properties for quantitative languages. In *Symposium on Logic in Computer Science*, pages 199–208. IEEE Computer Society, 2009.

[Cho56] Noam Chomsky. Three models for the description of language. *IRE Trans. Information Theory*, 2(3):113–124, 1956.

[CM12] Stefano Crespi Reghizzi and Dino Mandrioli. Operator precedence and the visibly pushdown property. *J. Comput. Syst. Sci.*, 78(6):1837–1867, 2012.

[CMM78] Stefano Crespi Reghizzi, Dino Mandrioli, and David F. Martin. Algebraic properties of operator precedence languages. *Information and Control*, 37(2):115–133, 1978.

[DD14] Manfred Droste and Stefan Dück. Weighted automata and logics for infinite nested words. In Adrian Horia Dediu, Carlos Martín-Vide, José Luis Sierra-Rodríguez, and Bianca Truthe, editors, *Language and Automata Theory and Applications, LATA 2014*, volume 8370 of *LNCS*, pages 323–334. Springer, 2014.

[DD15] Manfred Droste and Stefan Dück. Weighted automata and logics on graphs. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella, editors, *Mathematical Foundations of Computer Science, MFCS 2015, Part I*, volume 9234 of *LNCS*, pages 192–204. Springer, 2015.

[DD17] Manfred Droste and Stefan Dück. Weighted automata and logics for infinite nested words. *Inf. Comput.*, 253:448–466, 2017.

[DDMP17] Manfred Droste, Stefan Dück, Dino Mandrioli, and Matteo Pradella. Weighted operator precedence languages. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *Mathematical Foundations of Computer Science, MFCS 2015*, volume 83 of *LIPIcs*, pages 31:1–31:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[DG00] Volker Diekert and Paul Gastin. LTL is expressively complete for Mazurkiewicz traces. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *International Colloquium on Automata, Languages, and Programming, ICALP 2000*, volume 1853 of *LNCS*, pages 211–222. Springer, 2000.

[DG07] Manfred Droste and Paul Gastin. Weighted automata and weighted logics. *Theor. Comput. Sci.*, 380(1-2):69–86, 2007. extended abstract in ICALP 2005.

[DG09] Manfred Droste and Paul Gastin. Weighted automata and weighted logics. In Droste et al. [DKV09], chapter 5, pages 175–211.

[DGMM11] Manfred Droste, Doreen Götze, Steffen Märcker, and Ingmar Meinecke. Weighted tree automata over valuation monoids and their characterization by weighted logics. In Werner Kuich and George Rahonis, editors, *Algebraic Foundations in Computer Science*, volume 7020 of *LNCS*, pages 30–55. Springer, 2011.

[DHV15] Manfred Droste, Doreen Heusel, and Heiko Vogler. Weighted unranked tree automata over tree valuation monoids and their characterization by weighted logics. In Andreas Maletti, editor, *Conference Algebraic Informatics, CAI 2015*, volume 9270 of *LNCS*, pages 90–102. Springer, 2015.

[DJMT16] Adrian Horia Dediu, Jan Janousek, Carlos Martín-Vide, and Bianca Truthe, editors. *Language and Automata Theory and Applications, LATA 2016*, volume 9618 of *LNCS*. Springer, 2016.

[DKar] Manfred Droste and Dietrich Kuske. Weighted automata. In Jean-Eric Pin, editor, *Handbook: "Automata: from Mathematics to Applications"*. Europ. Mathematical Soc., to appear.

[DKV09] Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of Weighted Automata*. EATCS Monographs in Theoretical Computer Science. Springer, 2009.

[DM12] Manfred Droste and Ingmar Meinecke. Weighted automata and weighted MSO logics for average and long-time behaviors. *Inf. Comput.*, 220:44–59, 2012.

[Don70] John Doner. Tree acceptors and some of their applications. *J. Comput. Syst. Sci.*, 4(5):406–451, 1970.

[DP14a] Manfred Droste and Vitaly Perevoshchikov. A Nivat theorem for weighted timed automata and weighted relative distance logic. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *International Colloquium on Automata, Languages, and Programming, ICALP 2014, Part II*, volume 8573 of *LNCS*, pages 171–182. Springer, 2014.

[DP14b] Manfred Droste and Bundit Pibaljommee. Weighted nested word automata and logics over strong bimonoids. *Int. J. Found. Comput. Sci.*, 25(5):641–666, 2014.

[DR95] Volker Diekert and Grzegorz Rozenberg, editors. *The Book of Traces*. World Scientific, 1995.

[DR06] Manfred Droste and George Rahonis. Weighted automata and weighted logics on infinite words. In Oscar H. Ibarra and Zhe Dang, editors, *Developments in Language Theory, DLT 2006*, volume 4036 of *LNCS*, pages 49–58. Springer, 2006.

[Düc13] Stefan Dück. Gewichtete Automaten und Logik für unendliche geschachtelte Wörter. Diplomarbeit, Universität Leipzig, 2013.

[Düc16] Stefan Dück. Weighted automata and logics on infinite graphs. In Srecko Brlek and Christophe Reutenauer, editors, *Developments in Language Theory, DLT 2016*, volume 9840 of *LNCS*, pages 151–163. Springer, 2016.

[DV06] Manfred Droste and Heiko Vogler. Weighted tree automata and weighted logics. *Theor. Comput. Sci.*, 366(3):228–247, 2006.

[DV11] Manfred Droste and Heiko Vogler. Weighted logics for unranked tree automata. *Theory Comput. Syst.*, 48(1):23–47, 2011.

[DV12] Manfred Droste and Heiko Vogler. Weighted automata and multi-valued logics over arbitrary bounded lattices. *Theor. Comput. Sci.*, 418:14–36, 2012.

[Ehr61] Andrzej Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fund. Math.*, 49:129–141, 1961.

[Eil74] Samuel Eilenberg. *Automata, Languages, and Machines*, volume 59-A of *Pure and Applied Mathematics*. Academic Press, 1974.

[ÉK09] Zoltán Ésik and Werner Kuich. Finite automata. In Droste et al. [DKV09], pages 69–104.

[Elg61] Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Am. Math. Soc.*, 98(1):21–52, 1961.

[Fic11] Ina Fichtner. Weighted picture automata and weighted logics. *Theory Comput. Syst.*, 48(1):48–78, 2011.

[Flo63] Robert W. Floyd. Syntactic analysis and operator precedence. *J. ACM*, 10(3):316–333, 1963.

[Fra54] Roland Fraïssé. Sur quelques classifications des systèmes de relations. *Publications Scientifiques de l'Université d'Alger*, series A 1:35–182, 1954.

[FSV12] Zoltán Fülöp, Torsten Stüber, and Heiko Vogler. A Büchi-like theorem for weighted tree automata over multioperator monoids. *Theory Comput. Syst.*, 50(2):241–278, 2012.

[GM15] Paul Gastin and Benjamin Monmege. A unifying survey on weighted logics and weighted automata. *Soft Comput.*, 2015. To appear.

[Gol99] Jonathan S. Golan. *Semirings and their Applications*. Kluwer Academic Publishers, 1999.

[GRST96] Dora Giammarresi, Antonio Restivo, Sebastian Seibert, and Wolfgang Thomas. Monadic second-order logic over rectangular pictures and recognizability by tiling systems. *Inf. Comput.*, 125(1):32–45, 1996.

[Han65] William Hanf. Model-theoretic methods in the study of elementary logic. In Addison, Henkin, and Tarski, editors, *The Theory of Models*, pages 132–145. North-Holland, 1965.

[HtP97] Hendrik Jan Hoogeboom and Paulien ten Pas. Monadic second-order definable text languages. *Theory Comput. Syst.*, 30(4):335–354, 1997.

[KS86] Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*, volume 6 of *EATCS Monographs in Theoretical Computer Science*. Springer, 1986.

[Lib04] Leonid Libkin. *Elements of Finite Model Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[LMPP15] Violetta Lonati, Dino Mandrioli, Federica Panella, and Matteo Pradella. Operator precedence languages: Their automata-theoretic and logic characterization. *SIAM J. Comput.*, 44(4):1026–1088, 2015.

[LMS04] Christof Löding, Parthasarathy Madhusudan, and Olivier Serre. Visibly pushdown games. In Kamal Lodaya and Meena Mahajan, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 3328 of *LNCS*, pages 408–420. Springer, 2004.

[LST94] Clemens Lautemann, Thomas Schwentick, and Denis Thérien. Logics for context-free languages. In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic, Selected Papers*, volume 933 of *LNCS*, pages 205–216. Springer, 1994.

[Mat09] Christian Mathissen. *Weighted Automata and Weighted Logics over Tree-like Structures.* PhD thesis, Universität Leipzig, 2009.

[Mat10a] Christian Mathissen. Definable transductions and weighted logics for texts. *Theory Comput. Sci.*, 411(3):631–659, 2010.

[Mat10b] Christian Mathissen. Weighted logics for nested words and algebraic formal power series. *Logical Methods in Computer Science*, 6(1), 2010. Selected papers of ICALP 2008.

[Meh80] Kurt Mehlhorn. Pebbling mountain ranges and its application of DCFL-recognition. In *Automata, Languages and Programming, ICALP 1980*, volume 85 of *LNCS*, pages 422–435, 1980.

[Mei06] Ingmar Meinecke. Weighted logics for traces. In *International Computer Science Symposium in Russia*, volume 3967 of *LNCS*, pages 235–246. Springer, 2006.

[Moh03] Mehryar Mohri. Edit-distance of weighted automata: General definitions and algorithms. *Int. J. Found. Comput. Sci.*, 14(6):957–982, 2003.

[Mon13] Benjamin Monmege. *Specification and Verification of Quantitative Properties: Expressions, Logics, and Automata.* Thèse de doctorat, ENS Cachan, France, 2013.

[Niv68] Maurice Nivat. Transductions des langages de Chomsky. *Ann. de l'Inst. Fourier*, 18:339–455, 1968.

[Rab69] Michael O. Rabin. Decidability of second order theories and automata on infinite trees. *Trans. Am. Math. Soc.*, 141:1–35, 1969.

[Rah07] George Rahonis. Weighted Muller tree automata and weighted logics. *J. Autom. Lang. Comb.*, 12(4):455–483, 2007.

[Sch61] Marcel Paul Schützenberger. On the definition of a family of automata. *Inf. Control*, 4(2-3):245–270, 1961.

[SS78] Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science. Springer, 1978.

[Tho90] Wolfgang Thomas. On logical definability of trace languages. In Volker Diekert, editor, *Proc. workshop ASMICS 1989*, pages 172–182. Technical University of Munich, 1990.

[Tho91] Wolfgang Thomas. On logics, tilings, and automata. In *International Colloquium on Automata, Languages, and Programming, ICALP 1991*, volume 510 of *LNCS*, pages 441–454. Springer, 1991.

[Tho96] Wolfgang Thomas. Elements of an automata theory over partial orders. In *Proc. DIMACS Workshop POMIV '96*, pages 25–40, New York, NY, USA, 1996. AMS Press, Inc.

[Tho97] Wolfgang Thomas. Languages, automata, and logic. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Vol. 3*, pages 389–455. Springer, 1997.

[TO15] Markus Teichmann and Johannes Osterholzer. A link between multioperator and tree valuation automata and logics. *Theor. Comput. Sci.*, 594:106–119, 2015.

[Tra61] Boris A. Trakhtenbrot. Finite automata and logic of monadic predicates (in Russian). *Doklady Akademii Nauk SSR*, 140:326–329, 1961.

[TW68] James W. Thatcher and Jesse B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Math. Syst. Theory*, 2(1):57–81, 1968.

[vBV83] Burchard von Braunmühl and Rutger Verbeek. Input-driven languages are recognized in log n space. In *Proceedings of the Symposium on Fundamentals of Computation Theory*, volume 158 of *LNCS*, pages 40–51. Springer, 1983.

[VDH16] Heiko Vogler, Manfred Droste, and Luisa Herrmann. A weighted MSO logic with storage behaviour and its Büchi-Elgot-Trakhtenbrot theorem. In Dediu et al. [DJMT16], pages 127–139.

Dissertationsbezogene bibliographische Daten

Aktualisierung zum 12.06.2017

Begutachtete Veröffentlichungen

- M. Droste, S. Dück, D. Mandrioli, and M. Pradella: Weighted operator precedence languages, in: Mathematical Foundations of Computer Science (MFCS 2017), Leibniz International Proceedings in Informatics, to appear.
- M. Droste, S. Dück: Weighted automata and logics for infinite nested words, Information and Computation, Special Issue of LATA 2014, vol. 253, Part 3, pp. 448–466, 2017.
- S. Dück: Weighted automata and logics on infinite graphs, in: 20th International Conference on Developments in Language Theory (DLT), Lecture Notes in Computer Science, vol. 9840, pp. 151-163, Springer, 2016.
- M. Droste, S. Dück: Weighted automata and logics on graphs, in: Mathematical Foundations of Computer Science (MFCS 2015), Lecture Notes in Computer Science, vol. 9234, pp. 192-204, Springer, 2015.
- M. Droste, S. Dück: Weighted automata and logics for infinite nested words, in: Language and Automata Theory and Applications (LATA 2014), Lecture Notes in Computer Science, vol. 8370, pp. 323-334, Springer, 2014.

Wissenschaftlicher Werdegang

seit 04/2017	Wissenschaftlicher Mitarbeiter Abteilung: Automaten und Sprachen, Institut für Informatik, Universität Leipzig, DFG Graduiertenkolleg Quantla
seit 10/2014	Promotionsstudent des DFG Graduiertenkollegs *Quantitative Logics and Automata* (QuantLA)
seit 08/2013	Promotionsstudium der Informatik an der Universität Leipzig
10/2013 - 10/2014	Assoziierter Promotionsstudent des DFG Graduiertenkollegs *Quantitative Logics and Automata* (QuantLA)
08/2013 - 10/2014	Wissenschaftlicher Mitarbeiter Abteilung: Automaten und Sprachen, Institut für Informatik, Universität Leipzig, DFG-Projekt DR 202/11-1
10/2008 - 08/2013	Studium der Mathematik an der Universität Leipzig Abschluss: Diplom-Mathematiker Titel der Abschlussarbeit: *Gewichtete Automaten und Logik für unendliche geschachtelte Wörter* Betreuer: Prof. Dr. Manfred Droste Abschlussnote: mit Auszeichnung (1,0)
09/1999 - 07/2007	Wilhelm-Ostwald-Gymnasium, Leipzig Abschluss: Abitur (Abschlussnote 1,1)

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialen oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, den 14. Dezember 2017

Stefan Dück