Acta Cybernetica **17** (2006) 1–33.

Automata on Infinite Biposets*

Zoltán L. Németh†

Abstract

Bisemigroups are algebras equipped with two independent associative operations. Labeled finite sp-biposets may serve as a possible representation of the elements of the free bisemigroups. For finite sp-biposets, an accepting device, called parenthesizing automaton, was introduced in [6], and it was proved that its expressive power is equivalent to both algebraic recognizability and monadic second order definability. In this paper, we show, how this concept of parenthesizing automaton can be generalized for infinite biposets in a way that the equivalence of regularity (defined by acceptance with automata), recognizability (defined by homomorphisms and finite ω-bisemigroups) and MSO-definability remains true.

1 Introduction

The importance of automata and Büchi-automata is unquestionable in theoretical computer science from both theoretical and practical point of view. Its widespread applicability is mainly due to the fact that finite and infinite words can serve as models of a wide range of sequential systems. But, of course, there are many other computational models using more complex structures than words, such as trees, traces, posets, message sequence charts, graphs, etc. These models were introduced to capture other computational aspects, as timing or concurrency.

Besides the varying concept of automata and regularity, there is the more general notion of algebraic recognizability (by homomorphisms into finite algebras) and the concept of (counting) monadic second order logical definability. In many important cases these three notions can be suitably defined and they are known to be equivalent. In particular, this holds for finite trees, traces, message sequence charts, series-parallel posets of bounded width. See [23] for a recent survey on this topic. But sometimes we are confronted with serious difficulties. It is not always clear how to choose an appropriate algebraic or logic framework, and for graphs,

*An extended abstract of this paper appeared in the proceedings of AFL 2005 [19].

†Institute of Informatics, University of Szeged, P.O.B. 652, 6701 Szeged, Hungary, E-mail: zlnemeth@inf.u-szeged.hu

for posets, and even for sp-posets in general, a concept of automaton that matches algebraic recognizability is not known.

However, one of the most obvious generalizations of the case of words is the situation when we consider more than one, say n, associative operations. This naturally leads to the concept of n-semigroups and n-ω-semigroups. Accordingly, n-semigroups are sets equipped with n independent associative operations, and n-ω-semigroups are generalizations of the ω-semigroups of Perrin and Pin [20], where the formation of infinite (more precisely ω-ary) products is also allowed.

A description of the free n-semigroups by labeled finite n-posets was given by Ésik [5]. A Σ-labeled n-poset is a set P equipped with n patrial orders and a labeling function $P \to \Sigma$. One of the main results of [7] is a similar description of the free n-ω-semigroups by, so called, constructible n-posets. We say that a (finite or infinite) n-poset is constructible if it can be constructed from the singleton n-posets by the binary and the ω-ary product operations.

For simplicity, we only deal with the case when $n = 2$, i.e., we study bisemigroups and biposets only, although all of our notions and results can be generalized to n-semigroups and n-posets for any integer n greater than 2, without any difficulty.

In [6], an accepting device, called parenthesizing automaton, was introduced, and it was proved that for finite sp-biposets the recognizable, regular and MSO-definable languages coincide. Here we generalize the result mentioned above for infinite biposets. First, we show, with the help of a suitably defined notion of parenthesizing Büchi-automaton, that the class of regular languages of infinite constructible biposets coincides with the class of recognizable languages. We also demonstrate that, contrary to the word case, automata for infinite biposets must differ from automata for finite ones.

The equivalence of regular and recognizable sets implies that all MSO-definable languages are regular. Finally, we prove the converse inclusion, namely that every regular constructible biposet language is MSO-definable. (This verifies a conjecture of the preliminary version [19] of the present article.)

There are several branches of research that are in close connection with our investigations. Here we only briefly enumerate them, and refer to [6] where a whole section is devoted to a more detailed comparison. First of all, automata on series-parallel posets were studied by Lodaya and Weil in [15, 16, 17]. Their work was extended into two directions by Kuske [14], to automata on infinite posets and to (first- and second-order) logical definability. On text languages see the papers of Hoogeboom and ten Pas [12, 13]. On picture languages we refer to Giammarresi and Restivo [8] in general, and to Dolinka [1] in connection with sp-biposets. Finally, automata and languages over free bisemigroups (more precisely, free bisemigroups with identity, called binoids) have also been studied by Hashiguchi et al. [10, 11].

2 Basic concepts

2.1 Biposets and bisemigroups

In this paper, n always denotes a positive integer and Σ a finite alphabet. The empty word is denoted by ε. Let us call an algebra equipped with n associative operations *n-semigroup*. A *bisemigroup* is an n-semigroup for $n = 2$. It is proved in [5] that the elements of the free n-semigroups freely generated by some set Σ can be represented by finite Σ-labeled series-parallel n-posets defined as follows.

A *Σ-labeled n-poset*, or n-poset, for short, is a (finite or countably infinite) nonempty set P of vertices equipped with n (irreflexive) partial orders $<_i$ for $i = 1, \ldots, n$, and a labeling function $\lambda : P \to \Sigma$. We denote an n-poset by $P{=}(P, <_1, <_2, \ldots, <_n, \lambda)$, so we do not distinguish between the name of the biposet and the name of its vertex set. A *Σ-labeled biposet*, or biposet, is a Σ-labeled n-poset for $n = 2$.

The two partial orders of a biposet $(P, <_1, <_2, \lambda)$ are called the *horizontal* and the *vertical order*. Accordingly, instead of $<_1$ and $<_2$, we write $<_h$ and $<_v$, or $<_h^P$ and $<_v^P$ if we want to emphasize that these orderings belong to biposet P.

A *morphism* between biposets P and Q is a function on the vertices that preserves the partial orders and the labeling. An *isomorphism* is a bijective morphism whose inverse is also a morphism. Below we will identify isomorphic biposets.

Suppose that $P = (P, <_h^P, <_v^P, \lambda_P)$ and $Q = (Q, <_h^Q, <_v^Q, \lambda_Q)$ are Σ-labeled biposets. Without loss of generality, assume that P and Q are disjoint. We define their *horizontal product* as $P \bullet Q := (P \cup Q, <_h^{P\bullet Q}, <_v^{P\bullet Q}, \lambda_{P\bullet Q})$, and their *vertical product* as $P \circ Q := (P \cup Q, <_h^{P\circ Q}, <_v^{P\circ Q}, \lambda_{P\circ Q})$, where

$$\begin{array}{lcllcl} <_h^{P\bullet Q} & := & <_h^P \cup <_h^Q \cup (P \times Q), & <_h^{P\circ Q} & := & <_h^P \cup <_h^Q, \\ <_v^{P\bullet Q} & := & <_v^P \cup <_v^Q, & <_v^{P\circ Q} & := & <_v^P \cup <_v^Q \cup (P \times Q), \end{array}$$

and the labelings are $\lambda_{P\bullet Q} = \lambda_{P\circ Q} := \lambda_P \cup \lambda_Q$.

We say that a finite or infinite biposet P is *horizontal* if there are biposets P_1 and P_2 such that $P = P_1 \bullet P_2$, otherwise P is called *$\bullet$-irreducible* or *horizontally irreducible*. Similarly, P is *vertical* if it can be written as $P = P_1 \circ P_2$, and P is said to be *$\circ$-irreducible* or *vertically irreducible* if no such decomposition exists. The fact that P is a horizontal (vertical) biposet will be abbreviated as $\mathrm{Type}(P) = \bullet$ ($\mathrm{Type}(P) = \circ$, resp.) If P is a horizontal (vertical) biposet, then any factorization $P = P_1 \bullet P_2 \bullet \ldots \bullet P_m$ ($P = P_1 \circ P_2 \circ \ldots \circ P_m$), where $m \geq 2$, is called a *horizontal (vertical, resp.) decomposition* of P. A horizontal (vertical) decomposition is said to be *maximal* if every factor is horizontally (vertically, resp.) irreducible.

It is obvious that both product operations are associative. Each letter $\sigma \in \Sigma$ may be identified with the singleton biposet labeled σ. Let $\mathrm{SPB}(\Sigma)$ denote the collection of biposets that can be generated from the singletons corresponding to the letters in Σ by the two product operations in a finite number of steps. Clearly, these biposets are finite. The biposets in $\mathrm{SPB}(\Sigma)$ are called *series-parallel biposets*, or *sp-biposets*, for short. It is known that series-parallel biposets have a graph-theoretic characterization, which is an appropriate generalization of the "N-free"

condition for posets, cf. [5, 9, 22]. We say that an arbitrary biposet P is *complete* if every two vertices of P are related either horizontally or vertically, but not by both order relations. It is obvious that every sp-biposet is complete.

Proposition 1 ([5])**.** *A finite biposet* $(P, <_h, <_v, \lambda)$ *is in* $\mathrm{SPB}(\Sigma)$ *if and only if* P *is complete and both posets* $(P, <_h)$ *and* $(P, <_v)$ *are* N*-free.*

Proposition 2 ([5])**.** $\mathrm{SPB}(\Sigma)$ *is freely generated by* Σ *in the variety of bisemigroups.*

2.2 Term and tree representation of sp-biposets

The most evident way of representing sp-biposets is describing them by terms. For this reason, we extend the alphabet with operation symbols and parentheses. Let $\widehat{\Sigma} := \Sigma \cup \{ \bullet, \circ, \langle, \rangle \}$. As usual, we should put parentheses around the horizontal biposets that appear as vertical factors, and symmetrically, around the vertical biposets that appear as horizontal factors. The precise definition is the following.

Definition 3. *If* $P \in \mathrm{SPB}(\Sigma)$*, let* P^{tm} *denote the* term representation *of* P*. It is a word over the alphabet* $\widehat{\Sigma}$*, defined inductively as follows.*

(i) If $P = \sigma$ *is a singleton biposet, then* $P^{tm} := \sigma$*.*

(ii) If $P = P_1 \bullet P_2$*, then* $P^{tm} := \mathrm{Hform}(P_1) \bullet \mathrm{Hform}(P_2)$*.*

(iii) If $P = P_1 \circ P_2$*, then* $P^{tm} := \mathrm{Vform}(P_1) \circ \mathrm{Vform}(P_2)$*.*

Here $\mathrm{Hform}(P)$ *denotes the* horizontal form *of the sp-biposet* P*, defined as:*

$$\mathrm{Hform}(P) \quad := \quad \begin{cases} P^{tm} & \text{if } P \text{ is a singleton or horizontal biposet,} \\ \langle\, P^{tm} \,\rangle & \text{if } P \text{ is a vertical biposet.} \end{cases}$$

In (iii), $\mathrm{Vform}(P)$*, the* vertical form *of* P*, is defined symmetrically.*

It should be noted that in cases *(ii)* and *(iii)* above, the definition of P^{tm} does not depend on the choice of the factorization, since $\bullet$, $\circ$ and the concatenation of words are all associative operations.

We will also use finite ordered trees to represent sp-biposets. In that case, leaves are labeled from Σ, and the inner nodes are labeled by $\bullet$ or $\circ$.

Definition 4. *If* P *is an sp-biposet, its* tree form P^{tr}*, is defined as follows.*

(i) If $P = \sigma$ *is a singleton, then* P^{tr} *is a tree consisting of a single vertex labeled by* σ*.*

(ii) If P *is horizontal, then consider the maximal horizontal decomposition* $P = P_1 \bullet P_2 \bullet \ldots \bullet P_m$*,* $(m \geq 2)$*. Now* P^{tr} *is the tree whose root is labeled* $\bullet$ *and this root connects the subtrees* P_1^{tr}*,* P_2^{tr}*,* $\ldots$*,* P_m^{tr} *(in that order).*

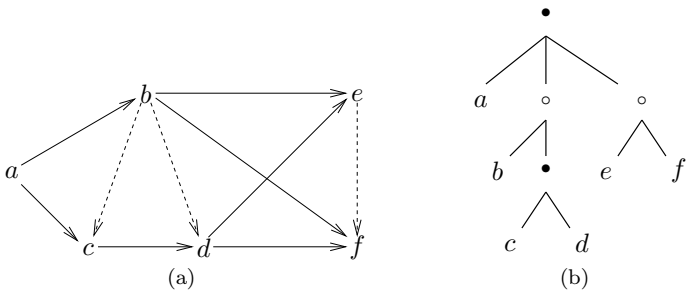

Figure 1: The biposet P of Example 5 (a), and its tree representation P^{tr} (b).

(iii) If P is vertical, then consider the maximal vertical decomposition $P = P_1 \circ P_2 \circ \ldots \circ P_m$, $(m \geq 2)$. Now P^{tr} is the tree whose root is labeled $\circ$ and this root connects the subtrees P_1^{tr}, P_2^{tr}, ..., P_m^{tr} (in that order).

Example 5. Consider the sp-biposet $P = (\{1, 2, \ldots, 6\}, <_h, <_v, \lambda)$, where $<_h$ and $<_v$ are the transitive closures of the relations $1 <_h 2$, $1 <_h 3$, $3 <_h 4$, $2 <_h 5$, $2 <_h 6$, $4 <_h 5$, $4 <_h 6$, and $2 <_v 3, 2 <_v 4, 5 <_v 6$, respectively. Moreover, $\lambda(1) = a$, $\lambda(2) = b$, $\lambda(3) = c$, $\lambda(4) = d$, $\lambda(5) = e$, $\lambda(6) = f$. Now $P^{tm} = a \bullet \langle b \circ \langle c \bullet d \rangle \rangle \bullet \langle e \circ f \rangle$, and the graphical representation of P and the tree representation P^{tr} are depicted in Figure 1. In the figure, horizontal and vertical relations are indicated by solid arrows and dashed arrows, respectively.

It is obvious that for any leaf node in P^{tr} there is a corresponding vertex in P. Hence, we may and will identify the leaves of P^{tr} with the corresponding vertices of P. This allows us to speak about elements and subsets of P as those of P^{tr}. Similarly, we can identify vertices of P with the corresponding letters in the term representation P^{tm}.

2.3 Infinite biposets and ω-bisemigroups

In this subsection, we briefly summarize the main results of [7] regarding infinite biposets. First, we introduce two types of operations that construct infinite biposets from finite ones: the ω-product and the ω-power.

Suppose that $P_1, P_2, \ldots$ are pairwise disjoint finite biposets. Their *horizontal ω-product* is defined as

$$\omega_\bullet(P_1, P_2, \ldots) := (P_1 \cup P_2 \cup \ldots, <_h, <_v, \lambda)$$

where

$$<_h := \bigcup_{i=1}^{\infty} <_h^{P_i} \ \cup \bigcup_{i<j} (P_i \times P_j), \qquad <_v := \bigcup_{i=1}^{\infty} <_v^{P_i}$$

and

$$\lambda := \lambda_{P_1} \cup \lambda_{P_2} \cup \ldots$$

The *vertical ω-product* $\omega_\circ(P_1, P_2, \ldots)$ is defined symmetrically. We will also refer to horizontal and vertical ω-products as $P_1 \bullet P_2 \bullet \ldots$ and $P_1 \circ P_2 \circ \ldots$, respectively. The two ω-product operations naturally induce a horizontal and a vertical power operation: $P^{\omega\bullet} := P \bullet P \bullet P \bullet \ldots$, and $P^{\omega\circ} := P \circ P \circ P \circ \ldots$

Note that the definition of the product operations applies to both finite and infinite operands. Nevertheless, in order to avoid constructing biposets which have chains not contained in ω, we will restrict the product operations $P \bullet Q$ and $P \circ Q$ to a finite biposet P only. The biposet Q may be finite or infinite. The ω-product and ω-power operations are applied only to finite biposets. These restrictions seem to be necessary for the proofs later.

All the restrictions just described imply that we should use two-sorted algebras as our algebraic framework making a difference between the finite and the infinite elements. Fortunately, this can be done in complete analogy to the case of finite and infinite words cf. [20]. But, as a minor difference from op. cit., we assume the binary product operations to be appropriately polymorphic, i.e., we use the same notation for the product of two finite biposets and for the product of a finite and an infinite biposet.

Accordingly, call an algebra $\mathcal{B} = (B_F, B_I, \bullet, \circ, \omega_\bullet, \omega_\circ)$ an *ω-bisemigroup* if it satisfies the following identities

$$\begin{aligned} x * (y * u) &= (x * y) * u, \\ x * \omega_*(x_1, x_2, \ldots) &= \omega_*(x, x_1, x_2, \ldots), \\ \omega_*(x_1 * \ldots * x_{k_1-1},\, x_{k_1} * \ldots * x_{k_2-1},\, \ldots) &= \omega_*(x_1, \ldots, x_{k_1-1},\, x_{k_1}, \\ &\quad x_{k_1+1}, \ldots, x_{k_2-1},\, \ldots), \end{aligned}$$

for all $x, y, x_1, x_2, \ldots \in B_F$, $u \in B_F \cup B_I$, $* \in \{\bullet, \circ\}$, and for all increasing sequences of positive integers $k_1 < k_2 < \ldots$

A *morphism* of ω-bisemigroups $\mathcal{C} = (C_F, C_I, \bullet, \circ, \omega_\bullet, \omega_\circ) \to \mathcal{D} = (D_F, D_I, \bullet', \circ', \omega'_\bullet, \omega'_\circ)$ is a pair of functions $h = (h_F : C_F \to D_F, h_I : C_I \to D_I)$ that jointly preserve the operations.

We call a Σ-labeled biposet *constructible* if it can be generated from the singleton Σ-labeled biposets by the (restricted) binary product operations $\bullet$ and $\circ$, and by the ω-ary product operations $\omega_\bullet$ and $\omega_\circ$.

Note that $\mathrm{SPB}(\Sigma)$ is exactly the set of those constructible biposets which are finite. Let $\mathrm{ISPB}(\Sigma)$ denote the set of infinite constructible biposets, and let

$$\omega\mathrm{SPB}(\Sigma) := (\mathrm{SPB}(\Sigma), \mathrm{ISPB}(\Sigma), \bullet, \circ, \omega_\bullet, \omega_\circ)$$

stand for the two-sorted algebra of all constructible biposets over Σ. It is clear that this is an ω-bisemigroup. Now, it is easily seen that the set of all finite and countably infinite biposets also form an ω-bisemigroup, and $\omega\mathrm{SPB}(\Sigma)$ is the smallest subalgebra of this ω-bisemigroup that contains Σ. The infinite counterpart of Proposition 2 is the following.

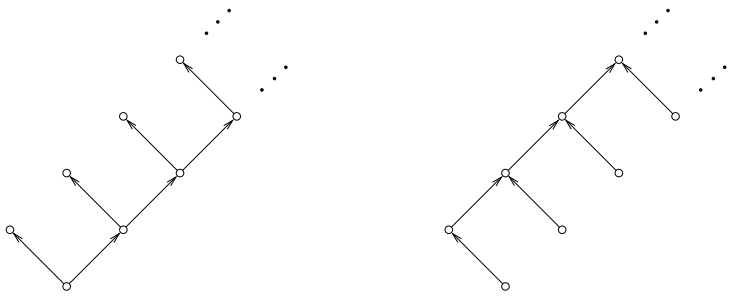

Figure 2: An upward comb (a) and a downward comb (b).

Proposition 6 ([7])**.** *The algebra* $\omega\mathrm{SPB}(\Sigma)$ *is freely generated by* Σ *in the variety of* ω*-bisemigroups.*

A graph-theoretic characterization of sp-biposets is also given in [7]. This, of course, is a suitable generalization of the "generalized N-free" condition of the finite case.

Proposition 7 ([7])**.** *An infinite biposet* $(P, <_h, <_v, \lambda)$ *is in* $\mathrm{ISPB}(\Sigma)$ *if and only if* P *is complete, and both posets* $(P, <_h)$ *and* $(P, <_v)$

(i) are N*-free,*

(ii) are free of "upward combs",

(iii) are free of "downward combs", and

(iv) have only finite principal ideals,

where the "upward comb" and "downward comb" posets are depicted in Figure 2.

See [7] for precise definitions.

In order to simplify the notations, in the sequel, we use $*$ to indicate any of the $\bullet$ and $\circ$ operations. Sometimes, we also give subscripts to the $*$-s, but in any formula all $*$ symbols, without subscript or with the same subscript, always denote the same operation.

A decomposition of P into an ω-product of infinitely many biposets $P = P_1 * P_2 * \ldots$ is said to be *maximal* if every P_i is $*$-irreducible. If P is an infinite constructible biposet, we say that P is *primitive* if it can be written as $P_1 * P_2 * \ldots$ for some finite sp-biposets $P_1, P_2, \ldots$ Now each infinite constructible biposet can be generated from the primitive biposets by multiplication with finite sp-biposets from the left. We define the *rank* of an infinite constructible biposet P as the least number of left multiplications with finite sp-biposets needed to construct P from the primitive infinite biposets. The rank of P is denoted by $\mathrm{Rank}(P)$.

It is easy to prove that if an infinite constructible biposet P is not primitive, than it can be uniquely written as $P = P' * P''$, where P'' is $*$-irreducible and $\mathrm{Rank}(P'') < \mathrm{Rank}(P)$. A direct consequence of this fact is that every infinite constructible biposet has the form

$$P_1 *_1 (P_2 *_2 (P_3 *_3 \ldots P_k *_k (Q_1 *_{k+1} Q_2 *_{k+1} Q_3 *_{k+1} \ldots))), \tag{1}$$

where all P_i and Q_i are finite biposets in $\mathrm{SPB}(\Sigma)$, and $*_1, *_2, *_3 \ldots$ is an alternating sequence of the $\bullet$ and $\circ$ operations. Moreover, this form is unique provided that every Q_i is $*_{k+1}$-irreducible. In this case, we call it the *normal form* of P. Note that if (1) is the normal form of P, then $\mathrm{Type}(P) = *_1$ and $\mathrm{Rank}(P) = k$.

2.4 Tree and term representations of infinite constructible biposets

Here we outline the changes to be made if one intends to represent infinite constructible biposets by terms and trees.

The only thing we need to describe is how to handle infinite products as $P = P_1 \bullet P_2 \bullet \ldots$ The definition of P^{tr} is straightforward if we allow ω-branching in trees. The tree P^{tr} has a root labeled by $\bullet$, and this root has ω branches connecting the tree representations of all the horizontally irreducible components of the P_i-s ($i \geq 1$).

There are only slight changes also in the term representation. The term representations of a biposet in $\mathrm{ISPB}(\Sigma)$ is an ω-word over the extended alphabet $\widehat{\Sigma}' := \Sigma \cup \{\langle, \rangle, [\}$. We should add two more cases to Definition 3:

(iv) *If* $P = P_1 \bullet P_2 \bullet \ldots$, *then* $P^{tm} := \mathrm{Hform}(P_1) \bullet \mathrm{Hform}(P_2) \bullet \ldots$

(v) *If* $P = P_1 \circ P_2 \circ \ldots$, *then* $P^{tm} := \mathrm{Vform}(P_1) \circ \mathrm{Vform}(P_2) \circ \ldots$

The definitions of the horizontal and vertical forms are also extended appropriately. In the representation of a product of a finite biposet with an infinite one, we use the [symbol if the type of the product differs from the type of the infinite factor. We only give the definition of the horizontal form:

$$\mathrm{Hform}(P) \quad := \quad \begin{cases} P^{tm} & \text{if } P \text{ is a singleton or a horizontal biposet,} \\ \langle\, P^{tm} \,\rangle & \text{if } P \text{ is a finite vertical biposet,} \\ [\, P^{tm} & \text{if } P \text{ is an infinite vertical biposet.} \end{cases}$$

2.5 Recognizability

A language consisting of finite sp-biposets is said to be *recognizable* if it is recognized by a homomorphism into a finite bisemigroup, i.e., $L \subseteq \mathrm{SPB}(\Sigma)$ is recognizable if and only if $L = \varphi^{-1}(F)$, for some bisemigroup homomorphism $\varphi : \mathrm{SPB}(\Sigma) \to B$, where B is a finite bisemigroup, and $F \subseteq B$.

Similarly, for a language that contains both finite and infinite biposets, $L = (L_F, L_I) \subseteq \omega\mathrm{SPB}(\Sigma)$, is recognizable if and only if there is a finite ω-bisemigroup $\mathcal{B} = (B_F, B_I)$, a subset of it, $T = (T_F, T_I) \subseteq (B_F, F_I)$, and a morphism $\varphi = (\varphi_F, \varphi_I) : \omega\mathrm{SPB}(\Sigma) \to \mathcal{B}$ such that $L = \varphi^{-1}(T)$. Here $(T_F, T_I) \subseteq (B_F, F_I)$ means $T_F \subseteq B_F$ and $T_I \subseteq F_I$, moreover, $L = \varphi^{-1}(T)$ stands for $L_F = \varphi_F^{-1}(T_F)$ and $L_I = \varphi_I^{-1}(T_I)$.

Example 8. Let $\Sigma = \{a, b, c\}$, and consider the following language $L \subseteq \mathrm{ISPB}(\Sigma)$ of infinite biposets

$$L = \{\, c^{\omega\bullet},\ a \bullet (b \circ (c^{\omega\bullet})),\ a \bullet (b \circ (a \bullet (b \circ (c^{\omega\bullet})))), \dots \}.$$

L is the least solution of the fixed point equation $a \bullet (b \circ X) + c^{\omega\bullet} = X$. It is not hard to show that L is recognizable. Indeed, consider the finite bisemigroup $B = (B_F, B_I)$, where $B_F = \{\, d_a, d_b, d_c, 0 \,\}$, and $B_I = \{\, t_1, t_2, t_3, \underline{0} \,\}$. The binary product operations are given by $d_c \bullet d_c = d_c$, and all other binary products of two finite elements are equal to 0, moreover, $d_c \bullet t_1 = t_1$, $d_b \circ t_1 = t_2$, $d_a \bullet t_2 = t_3$, $d_b \circ t_3 = t_2$, and all other products of a finite element with an infinite one are equal to $\underline{0}$. Finally, the ω-product operations are given by $d_c^{\omega\bullet} = t_1$, and all other ω-products are equal to $\underline{0}$. Now, if we take the homomorphism $\varphi : \omega\mathrm{SPB}(\Sigma) \to \mathcal{B}$ that is induced by the mapping $a \mapsto d_a$, $b \mapsto d_b$, $c \mapsto d_c$, then $L = \varphi_I^{-1}(\{\, t_1, t_3 \,\})$. This shows that L is recognizable.

2.6 Logical definability

By considering biposets as relational structures, there is a usual way of defining languages by logical formulas. Let $\mathcal{V} = \{x, y, \dots\}$ denote a fixed countable set of first-order variables, and $\mathcal{W} = \{X, Y, \dots\}$ a fixed countable set of monadic second-order variables.

Now we define monadic second order (MSO) formulas. An *atomic formula* (over Σ, $\mathcal{V}$ and $\mathcal{W}$) is of the form $Q_a(x)$, $X(x)$, $x <_h y$ or $x <_v y$, where $a \in \Sigma$, $x, y \in \mathcal{V}$, and $X \in \mathcal{W}$. MSO-*formulas* are composed from atomic formulas by the boolean connectives $\vee$ and $\neg$ and first- and second-order existential quantifiers $\exists x$ and $\exists X$, where $x \in \mathcal{V}$ and $X \in \mathcal{W}$.

We interpret formulas over both finite and infinite constructible biposets. Suppose that P is in $\mathrm{SPB}(\Sigma)$ or in $\mathrm{ISPB}(\Sigma)$. First order variables are interpreted to be vertices (also called positions) in P, whereas second order variables are interpreted to be sets of positions in P. Now, $Q_a(x)$ means that vertex x is labeled by a and $X(x)$ means that x belongs to X. The atomic formulas $x <_h y$ and $x <_v y$ have their expected meanings. The fact that a *closed formula (sentence) φ holds in, or is satisfied by P* is defined in the usual way, and it is denoted $P \models \varphi$. The *language defined by φ* is $L_\varphi := \{\, P \in (\mathrm{I})\mathrm{SPB}(\Sigma) \mid P \models \varphi \,\}$.

Definition 9. *We say that a language $L \subseteq (\mathrm{I})\mathrm{SPB}(\Sigma)$ is* MSO-definable *if there is sentence φ with $L = L_\varphi$.*

3 Automata and regularity

In this section, we will define parenthesizing automata operating on finite constructible biposets (i.e., on sp-biposets), and parenthesizing Büchi-automata operating on infinite constructible biposets.

3.1 Parenthesizing automata

An accepting device, called parenthesizing automaton, was introduced in [6] to define the class of regular languages of sp-biposets. Its definition below involves a finite set Ω of parentheses. *We assume that Ω is partitioned into opening and closing parentheses that are in a bijective correspondence.* We usually denote the corresponding pairs by $\langle_1, \rangle_1$ and $\langle_2, \rangle_2$, etc.

Definition 10. *A (nondeterministic)* parenthesizing automaton *is a 9-tuple $\mathcal{A} := (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$, where S is a nonempty, finite set of* states*, H and V are the sets of* horizontal *and* vertical states*, which give a disjoint partition of S, Σ is the* input alphabet*, Ω is a finite set of parentheses, moreover,*

- $\delta \subseteq (H \times \Sigma \times H) \cup (V \times \Sigma \times V)$ *is the* labeled transition relation*,*
- $\gamma \subseteq (H \times \Omega \times V) \cup (V \times \Omega \times H)$ *is the* parenthesizing transition relation*,*
- $I, F \subseteq S$ *are the sets of* initial *and* final states*, respectively.*

Let $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ be a parenthesizing automaton. If $t = (p, x, q)$ is a labeled or parenthesizing transition of $\mathcal{A}$, i.e., $t \in \delta \cup \gamma$, the *starting* and the *ending state* of t is denoted by $\mathrm{start}(t) := p$ and $\mathrm{end}(t) := q$, respectively. Moreover, if $\mathbf{r} = t_1 t_2 \dots t_n \in (\delta \cup \gamma)^*$ is a sequence of transitions, then let $\mathrm{start}(\mathbf{r}) := \mathrm{start}(t_1)$ and $\mathrm{end}(\mathbf{r}) := \mathrm{end}(t_n)$. We say that two parenthesizing transitions $t_1 = (p, \omega_1, q)$ and $t_2 = (s, \omega_2, t) \in \gamma$ form a *parenthesizing transition pair* if ω_1 is an opening parenthesis and ω_2 is its closing partner.

Definition 11. *Let $\mathcal{A}$ be a parenthesizing automaton. The set of its* runs*, $\mathrm{Runs}(\mathcal{A})$, is the least set of transition sequences that contains*

(i) (p, σ, q) *for every* $(p, \sigma, q) \in \delta$*;*

(ii) $\mathbf{r_1 r_2}$ *for every* $\mathbf{r_1}, \mathbf{r_2} \in \mathrm{Runs}(\mathcal{A})$ *provided that* $\mathrm{end}(\mathbf{r_1}) = \mathrm{start}(\mathbf{r_2})$*;*

(iii) $t_1 \mathbf{r}\, t_2$ *for every t_1 and t_2 parenthesizing transition pair such that* $\mathrm{end}(t_1) = \mathrm{start}(\mathbf{r})$*,* $\mathrm{end}(\mathbf{r}) = \mathrm{start}(t_2)$*, and for every* $\mathbf{r} \in \mathrm{Runs}(\mathcal{A})$ *such that* $\mathbf{r}$ *is of the form* $\mathbf{r} = \mathbf{r_1 r_2}$*, where* $\mathbf{r_1}, \mathbf{r_2} \in \mathrm{Runs}(\mathcal{A})$*.*

In case (i), the run is called singleton run*, in case (ii), it is called* direct run*, in case (iii) the run is called* indirect run*.*

Let $\mathcal{A}$ be a parenthesizing automaton, $\mathbf{r} = t_1 \dots t_n \in \mathrm{Runs}(\mathcal{A})$. A parenthesizing transition pair t_i, t_j, $(i < j)$ is said to be a *matching parenthesizing transition pair in* $\mathbf{r}$ if $t_i \dots t_j$ is an indirect run of $\mathcal{A}$. It is obvious that every run of $\mathcal{A}$ is of the form

$$\mathbf{r} = t_1 t_2 t_3 \dots t_n = (p_0, \omega_1, p_1)(p_1, \omega_2, p_2)(p_2, \omega_3, p_3) \dots (p_{n-1}, \omega_n, p_n),$$

where $p_i \in S$ and $\omega_i \in \Sigma \cup \Omega$ for all $i = 1, \dots, n$. If $\mathbf{r}$ is an indirect run, then t_1 and t_n is a matching parenthesizing transition pair, and $t_2 \dots t_{n-1}$ is a direct run of $\mathcal{A}$. Moreover, if $\mathbf{r}$ is a direct run, then it has a unique decomposition into subruns $\mathbf{r} = \mathbf{r}_1 \mathbf{r}_2 \dots \mathbf{r}_k$, where each $\mathbf{r}_i$ is either a singleton run or an indirect run for $i = 1, \dots, k$, and $k \geq 2$.

Definition 12. *Suppose that $\mathcal{A}$ is a parenthesizing automaton and $\mathbf{r} \in \mathrm{Runs}(\mathcal{A})$. The* biposet *of $\mathbf{r}$ is an element of* $\mathrm{SPB}(\Sigma)$ *defined inductively as follows:*

(i) If $\mathbf{r} = (p, \sigma, q)$, then $\mathrm{Biposet}(\mathbf{r}) := \sigma$.

(ii) If $\mathbf{r}$ is a direct run, and $\mathbf{r} = \mathbf{r_1} \mathbf{r_2}$ for some $\mathbf{r_1}, \mathbf{r_2} \in \mathrm{Runs}(\mathcal{A})$, then

- *if* $\mathrm{end}(\mathbf{r_1}) \in H$, *then* $\mathrm{Biposet}(\mathbf{r}) := \mathrm{Biposet}(\mathbf{r_1}) \bullet \mathrm{Biposet}(\mathbf{r_2})$;
- *if* $\mathrm{end}(\mathbf{r_1}) \in V$, *then* $\mathrm{Biposet}(\mathbf{r}) := \mathrm{Biposet}(\mathbf{r_1}) \circ \mathrm{Biposet}(\mathbf{r_2})$.

(iii) If $\mathbf{r}$ is an indirect run $\mathbf{r} = t_1 \mathbf{r}' t_2$, then $\mathrm{Biposet}(\mathbf{r}) := \mathrm{Biposet}(\mathbf{r}')$.

As in Definition 3, the definition of $\mathrm{Biposet}(\mathbf{r})$ is also independent of the choice of factorization in case *(ii)* above.

If $\mathbf{r} = (p_0, \omega_1, p_1)(p_1, \omega_2, p_2) \dots (p_{n-1}, \omega_n, p_n)$ is a run of $\mathcal{A}$, we define the *word* of $\mathbf{r}$ as

$$\mathrm{Word}(\mathbf{r}) := \omega_1' \omega_2' \dots \omega_n',$$

where

$$\omega_i' \quad := \quad \begin{cases} \omega_i & \text{if } \omega_i \in \Sigma, \\ \langle & \text{if } \omega_i \in \Omega \text{ is an opening parenthesis, and} \\ \rangle & \text{if } \omega_i \in \Omega \text{ is a closing parenthesis.} \end{cases}$$

The relationship between the term representation of a biposet and the word of a run on that biposet, is given by the following lemma. This is a straightforward consequence of Definition 3, Definition 11 and Definition 12. In the sequel, we write $\mathrm{Type}(q) = \bullet$ if q is a horizontal state, and $\mathrm{Type}(q) = \circ$ if q is a vertical state of an automaton $\mathcal{A}$.

Lemma 13. *Suppose that $\mathcal{A}$ is a parenthesizing automaton, $\mathbf{r} \in \mathrm{Runs}(\mathcal{A})$, and $P = \mathrm{Biposet}(\mathbf{r})$.*

(i) $\mathbf{r}$ is singleton or direct run $\Leftrightarrow \mathrm{Type}(\mathrm{start}(\mathbf{r})) = \mathrm{Type}(P)$
$\Leftrightarrow \mathrm{Word}(\mathbf{r}) = P^{tm}$.

(ii) $\mathbf{r}$ is indirect run $\Leftrightarrow \mathrm{Type}(\mathrm{start}(\mathbf{r})) \neq \mathrm{Type}(P) \Leftrightarrow \mathrm{Word}(\mathbf{r}) = \langle P^{tm} \rangle$.

Definition 14. *Suppose that* $P \in \mathrm{SPB}(\Sigma)$ *and* $p, q \in S$. *We say that* $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ has a run on P from p to q, *denoted* $[p, P, q]_{\mathcal{A}}$, *if there is a run* $\mathbf{r} \in \mathrm{Runs}(\mathcal{A})$ *with* $\mathrm{start}(\mathbf{r}) = p$, $\mathrm{end}(\mathbf{r}) = q$, *and* $\mathrm{Biposet}(\mathbf{r}) = P$.

Definition 15. *The sp-biposet* language $L(\mathcal{A})$ accepted by the automaton $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ *is defined as*

$$L(\mathcal{A}) \quad := \quad \{\, P \in \mathrm{SPB}(\Sigma) \mid [i, P, f]_{\mathcal{A}} \text{ for some } i \in I \text{ and } f \in F \,\}.$$

An sp-biposet language $L \subseteq \mathrm{SPB}(\Sigma)$ *is called* regular *if there exists a parenthesizing automaton* $\mathcal{A}$ *that accepts it, i.e.,* $L = L(\mathcal{A})$. *Two automata are said to be* equivalent *if they accept the same language.*

The following lemma is a straightforward consequence of Definition 11 and Definition 12.

Lemma 16. *Let* $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ *be a parenthesizing automaton, and let* P *be a horizontal sp-biposet, with maximal horizontal decomposition* $P = P_1 \bullet \ldots \bullet P_n$, $(n \geq 2)$.

If $p, q \in H$, *then*

$$[p, P, q]_{\mathcal{A}} \Leftrightarrow \exists\, r_1, \ldots, r_{n-1} \in H,\ r_0 = p,\ r_n = q \,:\, [r_{i-1}, P_i, r_i]_{\mathcal{A}}\ \forall i = 1, \ldots, n.$$

If $p, q \in V$, *then*

$$[p, P, q]_{\mathcal{A}} \Leftrightarrow \exists\, \langle_k, \rangle_k \in \Omega,\ \exists\, p', q' \in H \,:\, (p, \langle_k, p'),\ (q', \rangle_k, q) \in \gamma,\ [p', P, q']_{\mathcal{A}}.$$

Obviously, for vertical sp-biposets there are two analogous statements.

Corollary 17. *If* $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ *is a parenthesizing automaton, and* P *is a horizontal biposet, then*

$$\begin{array}{lll} P \in L(A) \Leftrightarrow & \textit{either} & \textit{i) } [i, P, f]_{\mathcal{A}},\ \textit{where } i \in I \cap H,\ \textit{and } f \in F \cap H; \\ & \textit{or} & \textit{ii) } [r, P, s]_{\mathcal{A}} \textit{ where } r, s \in H,\ \textit{and } (i, \langle, r), (s, \rangle, f) \in \gamma, \\ & & \quad i \in I \cap V,\ f \in F \cap V,\ \langle, \rangle \in \Omega. \end{array}$$

Again, an analogous statement holds for vertical sp-biposets.

We do not give examples of parenthesizing automata here, but several examples can be found in [6]. The main result concerning sp-biposet languages is the following.

Theorem 18 ([6])**.** *An sp-biposet language* $L \subseteq \mathrm{SPB}(\Sigma)$ *is recognizable if and only if it is regular if and only if it is* MSO*-definable.*

3.2 Parenthesizing Büchi-automata

Our next task is to define parenthesizing Büchi-automaton so that a language is recognizable if and only if it can be accepted by such an automaton. A straightforward approach would be to use the same accepting device and only extend the notion of run appropriately for the acceptance of languages of infinite biposets, but, as we shall see, this cannot be achieved. Thus, our definition is the following.

Definition 19. *A* parenthesizing Büchi-automaton *is a tuple* $\mathcal{A} := (S, H, V, \Sigma, \Omega, [, \delta, \beta, \gamma, I, F)$*, where* $\mathcal{A}' := (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ *is a parenthesizing automaton, called the* underlying parenthesizing automaton *of* $\mathcal{A}$*. And the new components are the following:*

- $[\notin (\Sigma \cup \Omega)$ *is the* separating parenthesis*, and*
- $\beta \subseteq (H \times \{ [\} \times V) \cup (V \times \{ [\} \times H)$ *is the* separating transition relation.

For the sake of simplicity, we will write $[p, P, q]_{\mathcal{A}}$ instead of $[p, P, q]_{\mathcal{A}'}$ if P is an sp-biposet, and $\mathcal{A}'$ is the underlying parenthesizing automaton of the parenthesizing Büchi-automaton $\mathcal{A}$.

Remark 20. It was proved in [18] that if we would like to accept all regular sp-biposet languages, we cannot give a universal upper bound for the number of parentheses used in parenthesizing automata. On the other hand, as we need not to close parentheses of the separating transitions, a single symbol in itself is enough for changing the type of the state at the borders of "finite-infinite" products.

Next, we define when a parenthesizing automaton $\mathcal{A}$ accepts an infinite biposet P from a given state p. For this, we choose Büchi's approach: for acceptance a run must contain a final state r (in certain "outer" positions) infinitely many times. Let $[p, P, r]^{\infty}_{\mathcal{A}}$ denote this fact. Its definition distinguishes two cases and uses induction on the rank of P. Recall that we write $\mathrm{Type}(p) = \bullet$ if p is a horizontal state, and $\mathrm{Type}(p) = \circ$ if p is a vertical state of $\mathcal{A}$. Similarly, $\mathrm{Type}(P) = \bullet$ ($\mathrm{Type}(P) = \circ$) indicates that P is a horizontal (vertical, resp.) biposet.

Definition 21. *Suppose that* $\mathcal{A} = (S, H, V, \Sigma, \Omega, [, \delta, \gamma, \beta, I, F)$ *is a parenthesizing Büchi-automaton,* p *and* r *are in* S*, and* P *is an infinite constructible biposet. We write* $[p, P, r]^{\infty}_{\mathcal{A}}$ *in the following cases.*

i) $\mathrm{Type}(p) = \mathrm{Type}(P)$*, and either*

- *α) P can be written as* $P = P_0 * P_1 * P_2 * \ldots$*, where each* P_i *is a finite (not necessarily* $*$*-irreducible) sp-biposet such that* $[p, P_0, r]_{\mathcal{A}}$ *and* $[r, P_i, r]_{\mathcal{A}}$ *for* $i > 0$*; or*
- *β)* $P = P' * P''$*, where* $\mathrm{Rank}(P'') < \mathrm{Rank}(P)$*, and there is a state* $q \in S$ *such that* $[p, P', q]_{\mathcal{A}}$ *and* $[q, P'', r]^{\infty}_{\mathcal{A}}$*, the latter is defined inductively.*

ii) $\mathrm{Type}(p) \neq \mathrm{Type}(P)$*, and there exists a state* $p' \in S$ *such that* $\mathcal{A}$ *has a separating transition* $(p, [, p') \in \beta$*, and* $[p', P, r]^{\infty}_{\mathcal{A}}$ *holds according to case i) above.*

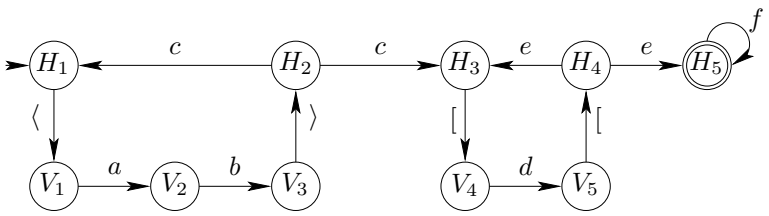

Figure 3: A parenthesizing Büchi-automaton

Definition 22. *A parenthesizing Büchi-automaton $\mathcal{A}$ accepts the following language*

$$L(\mathcal{A}) := \{\, P \in \mathrm{ISPB}(\Sigma) \mid [i, P, f]^{\infty}_{\mathcal{A}} \text{ for some } i \in I \text{ and } f \in F \,\}.$$

Again, a language $L \subseteq \mathrm{ISPB}(\Sigma)$ is regular *if there is a parenthesizing Büchi-automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.*

Similarly to Definition 11, one could also define infinite runs formally. In this case, runs are ω-words over the union of the sets of labeled, parenthesizing and separating transitions. Later we will use the same notations as in the finite case: $\mathrm{Runs}(\mathcal{A})$, $\mathrm{Biposet}(\mathbf{r})$, etc.

Example 23. Figure 3 shows a parenthesizing Büchi-automaton. The horizontal states are those labeled H_i and the vertical states are those labeled V_j for some i and j. There is a single initial state H_1 and a single final state H_5. The angle brackets indicate parenthesizing transitions, while the square brackets represent separating transitions. It is easy to check that this automaton accepts exactly the constructible biposets of the form

$$(a \circ b) \bullet c \bullet (a \circ b) \bullet c \bullet \ldots \bullet (a \circ b) \bullet c \bullet (d \circ (e \bullet (d \circ (e \bullet \ldots (d \circ (e \bullet f \bullet f \bullet f \ldots)) \ldots))))$$

Remark 24. As we mentioned above, we cannot use original parenthesizing automata for acceptance of infinite biposets. First, there is no meaningful definition of closing a parenthesis after the acceptance of an infinite subbiposet. E.g., suppose that $P = Q \bullet R$ is an infinite constructible biposet, where Q is a finite sp-biposet (either horizontal or vertical), and R is an infinite vertical biposet. Now, if there is a finite run $[p, Q, q]_{\mathcal{A}}$ for some horizontal states p and q, we must open a parenthesis by some transition $(q, \langle, r)$ in order to arrive at a vertical state r, from where the acceptance of the infinite vertical biposet R can be started. However, as R is infinite, we have no possibility to close this parenthesis. Second, it can be proved that we must distinguish between the normal and these "non-closable" parenthesizing transitions, otherwise there would be ω-recognizable languages that cannot be accepted by Büchi-automaton. To check this, consider the language $L := \{a \bullet (b \circ c) \bullet d^{\omega\bullet}, a \bullet (b \circ (a \bullet (b \circ c))) \bullet d^{\omega\bullet}, a \bullet (b \circ (a \bullet (b \circ (a \bullet (b \circ c))))) \bullet d^{\omega\bullet}, \ldots\}$.

4 From regularity to recognizability

The fact the regularity implies recognizability is easily follows from the finite-state property of automata.

Theorem 25. *Every regular language of infinite constructible biposets is recognizable.*

Proof. Let L denote a language containing only infinite constructible biposets, i.e., $L \subseteq \mathrm{ISPB}(\Sigma)$. We show how to transform a parenthesizing Büchi-automaton $\mathcal{A} = (S, H, V, \Sigma, \Omega, [, \delta, \gamma, \beta, I, F)$ accepting L into a finite ω-bisemigroup recognizing L analogously to [20].

Recall that $[p, P, q]_{\mathcal{A}}$ means that the automaton $\mathcal{A}$ has a run on the sp-biposet P from state p to state q. Moreover, we write $\mathrm{Type}(p) = \bullet$ if p is a horizontal state, and $\mathrm{Type}(p) = \circ$ if p is a vertical state of $\mathcal{A}$. Suppose that $\mathrm{Type}(p) = \mathrm{Type}(q) = *$, and consider an sp-biposet P. If P is $*$-irreducible, then take $P_1 = P$ and $m = 1$, otherwise let the maximal $*$-decomposition of P be $P = P_1 * P_2 * \ldots * P_m$ for some $m \geq 2$. According to Lemma 16, there are states $p_0 = p, p_1, \ldots, p_m = q$ of type $*$ such that $[p_0, P_1, p_1]_{\mathcal{A}}$, $[p_1, P_2, p_2]_{\mathcal{A}}, \ldots, [p_{m-1}, P_m, p_m]_{\mathcal{A}}$ hold. Let us write $[p, P, q]_{\mathcal{A}_+}$ to indicate the existence of such states for which $\{p_0, \ldots, p_m\} \cap F \neq \emptyset$. Thus, $[p, P, q]_{\mathcal{A}_+}$ if and only if there is at least one final state among the "outer" states of a possible run between p and q on P.

Next, we define for any $P, Q \in \mathrm{SPB}(\Sigma)$ and $P', Q' \in \mathrm{ISPB}(\Sigma)$

$$P \sim_F Q \text{ iff } \forall\, p, q \in S : \big([p, P, q]_{\mathcal{A}} \Leftrightarrow [p, Q, q]_{\mathcal{A}} \text{ and } [p, P, q]_{\mathcal{A}_+} \Leftrightarrow [p, Q, q]_{\mathcal{A}_+}\big),$$
$$P' \sim_I Q' \text{ iff } \forall\, p \in S : \big(\exists\, r \in F :\ [p, P', r]^{\infty}_{\mathcal{A}} \Leftrightarrow \exists\, r' \in F :\ [p, Q', r']^{\infty}_{\mathcal{A}}\big).$$

Now one can check that $\sim_F$ and $\sim_I$ are equivalence relations with finitely many equivalence classes. Furthermore, they satisfy the following equalities. If $P_i, Q_i \in \mathrm{SPB}(\Sigma)$, for $i = 1, 2, \ldots$, and $P', Q' \in \mathrm{ISPB}(\Sigma)$, $* \in \{\bullet, \circ\}$, then

$$\begin{aligned}
&P_1 \sim_F Q_1,\ P_2 \sim_F Q_2 && \Rightarrow && P_1 * P_2 \sim_F Q_1 * Q_2,\\
&P_i \sim_F Q_i \text{ for } i > 0 && \Rightarrow && P_1 * P_2 * \ldots \sim_I Q_1 * Q_2 * \ldots, \text{ and}\\
&P_1 \sim_F Q_1,\ P' \sim_I Q' && \Rightarrow && P_1 * P' \sim_I Q_1 * Q'.
\end{aligned}$$

Hence the quotient can be equipped with the structure of an ω-bisemigroup. Finally, the canonical epimorphism of $\omega\mathrm{SPB}(\Sigma)$ onto this quotient accepts $L(\mathcal{A})$. □

5 From recognizability to regularity

In this section, our aim is to prove that every recognizable infinite constructible biposet language is regular, i.e., can be accepted by a parenthesizing Büchi-automaton. First, we observe that every parenthesizing automaton is equivalent to one in normal form, i.e., with a single initial and a single final state.

In the sequel, we assume that no automaton has two opening or closing parenthesizing transitions with the same label. This can easily be achieved by replacing the multiple occurrences of the same parenthesizing transition pair with new transitions using different symbols. We start with the definition of the substitution product of parenthesizing automata.

Definition 26. *Suppose that* $\mathcal{A}_1 = (S_1, H_1, V_1, \Sigma, \Omega, \delta_1, \gamma_1, I_1, F_1)$ *and* $\mathcal{A}_2 = (S_2, H_2, V_2, \Sigma, \Omega, \delta_2, \gamma_2, I_2, F_2)$ *are parenthesizing automata, and either* $p, q \in H_1$ *and* $R, S \subseteq H_2$*; or* $p, q \in V_1$ *and* $R, S \subseteq V_2$*. We assume that* S_1 *and* S_2 *are disjoint. We define the* substitution product *of* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *with respect to* p*,* q*,* R *and* S*, as*

$$\mathcal{A}_1 *_{[p\to R, S\to q]} \mathcal{A}_2 := (S_3, H_3, V_3, \Sigma, \Omega_3, \delta_3, \gamma_3, I_1, F_1),$$

where

$$\begin{aligned}
S_3 &:= S_1 \cup S_2, \quad H_3 := H_1 \cup H_2, \quad V_3 := V_1 \cup V_2,\\
\Omega_3 &:= \Omega \cup \{\, \langle^{\text{first}}, \rangle^{\text{first}},\ \langle^{\text{last}}, \rangle^{\text{last}} \mid \langle, \rangle \in \Omega \,\},\\
\delta_3 &:= \delta_1 \cup \delta_2\\
&\quad \cup \{\, (p, a, x) \mid a \in \Sigma, x \in S_2, \exists r \in R : (r, a, x) \in \delta_2 \,\}\\
&\quad \cup \{\, (y, b, q) \mid y \in S_2, b \in \Sigma, \exists s \in S : (y, b, s) \in \delta_2 \,\},\\
\gamma_3 &:= \gamma_1 \cup \gamma_2\\
&\quad \cup \{\, (p, \langle^{\text{first}}, x), (y, \rangle^{\text{first}}, z) \mid x, y, z \in S_2, \exists r \in R : (r, \langle, x), (y, \rangle, z) \in \gamma_2 \,\}\\
&\quad \cup \{\, (x, \langle^{\text{last}}, y), (z, \rangle^{\text{last}}, q) \mid x, y, z \in S_2, \exists s \in S : (x, \langle, y), (z, \rangle, s) \in \gamma_2 \,\}.
\end{aligned}$$

The construction is illustrated in Figure 4. If both R and S are singletons, say $R = \{r\}$ and $S = \{s\}$, then we will write $\mathcal{A}_1 *_{[p\to r, s\to q]} \mathcal{A}_2$ instead of $\mathcal{A}_1 *_{[p\to\{r\},\{s\}\to q]} \mathcal{A}_2$. The next lemma formulates a key property of the substitution product.

Lemma 27. *Suppose that* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are parenthesizing automata as above,* $p, q \in H_1$*,* $R, S \subseteq H_2$ *and* $\mathcal{A}_3 = \mathcal{A}_1 *_{[p\to R, S\to q]} \mathcal{A}_2$*. Moreover,* $p \neq q$*, and no transition of* $\mathcal{A}_1$ *arrives at* p *or starts from* q*. Then, for every* $P \in \mathrm{SPB}(\Sigma)$

$$\begin{aligned}
[p, P, q]_{\mathcal{A}_3} \Leftrightarrow \quad & \textit{either} \quad \textit{i)}\ [p, P, q]_{\mathcal{A}_1},\\
& \textit{or} \quad \textit{ii)}\ \exists r \in R, \exists s \in S\ :\ [r, P, s]_{\mathcal{A}_2} \textit{ and } P \textit{ is horizontal.}
\end{aligned}$$

Proof. $[p, P, q]_{\mathcal{A}_3}$ implies that $P = \mathrm{Biposet}(\mathbf{r})$ for a run $\mathbf{r} = t_1 \dots t_m \in \mathrm{Runs}(\mathcal{A}_3)$ with $\mathrm{start}(t_1) = p$ and $\mathrm{end}(t_m) = q$.

If $\mathrm{end}(t_1) \in S_1$, then $\mathbf{r} \in \mathrm{Runs}(\mathcal{A}_1)$ also holds. This follows from the definition of $\mathcal{A}_3$ and from the fact that no transition arrives at p. Thus, case (i) is true.

If $\mathrm{end}(t_1) \in S_2$, then we can modify $\mathbf{r}$ to obtain a run $\mathbf{r}' := t'_1 \dots t'_m \in \mathrm{Runs}(\mathcal{A}_2)$ with $\mathrm{Biposet}(\mathbf{r}') = P$ and $\mathrm{start}(t'_1) \in R$, and $\mathrm{end}(t'_m) \in S$. Indeed, if t_1 is of the form $t_1 = (p, a, x)$, $a \in \Sigma$, then there is an $r \in R$ such that $t'_1 := (r, a, x) \in \delta_2$. Similarly, if $t_m = (y, b, q)$, $b \in \Sigma$, then there is an $s \in S$ such that $t'_m := (y, b, s) \in \delta_2$. On the other hand, if t_1 or t_m involves parenthesis, e.g., $t_1 = (p, \langle^{\text{first}}, x)$, then

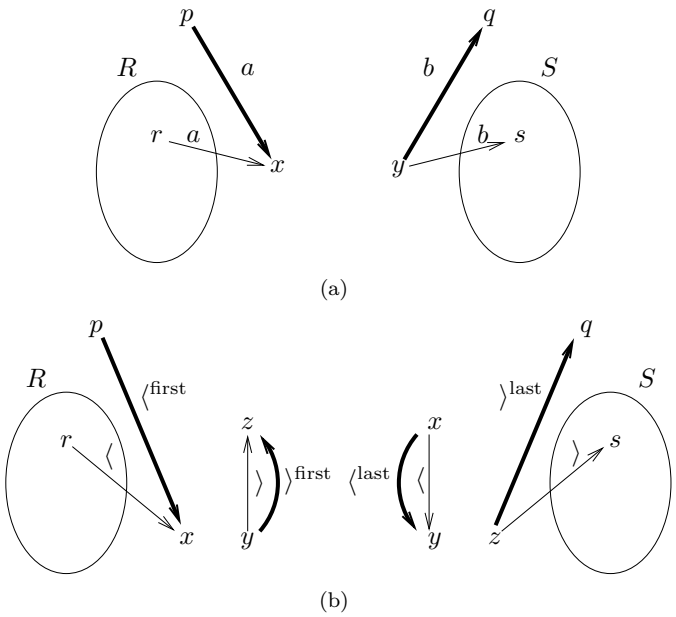

Figure 4: The labeling (a) and the parenthesizing (b) transitions used in Definition 26. The thin arrows represent the original transitions, and the thick arrows the new ones.

there is a closing transition partner of t_1, say $t_i = (y, \rangle^{\text{first}}, z)$, where $i < m$, and $x, y, z \in S_2$. Moreover, by definition, there is an $r \in R$ such that $t'_1 := (r, \langle, x)$, $t'_i := (y, \rangle, z) \in \gamma_2$. Similarly, if $t_m = (z, \rangle^{\text{last}}, q)$, then there is an index $j > 1$ such that $t_j = (x, \langle^{\text{last}}, y)$. So, we can set $t'_j := (x, \langle, y)$ and $t'_m := (z, \rangle, s) \in \gamma_2$ for a suitable $s \in S$. So far we have defined t'_k for at most four k-s. Let $t'_k := t_k$ for all other k-s (note that $t_k \in \delta_2 \cup \gamma_2$ in these cases). Now $\text{Biposet}(\mathbf{r}') = P$, $\text{start}(t'_1) = r \in R$, and $\text{end}(t'_m) = s \in S$ implies $[r, P, s]_{\mathcal{A}_2}$. Since $\langle^{\text{first}}$ and $\rangle^{\text{last}}$ do not match, t_1 and t_m cannot be a matching parenthesizing transition pair. Hence, $\mathbf{r}$ is a direct run, and so is $\mathbf{r}'$. Consequently, by Lemma 13, $r, s \in H$ implies that P is horizontal. Thus, (ii) holds.

For the converse direction, it is obvious that $[p, P, q]_{\mathcal{A}_1}$ implies $[p, P, q]_{\mathcal{A}_3}$. Assume that $ii)$ holds, so $P = \text{Biposet}(\mathbf{r})$ for $\mathbf{r} = t_1 \dots t_m \in \text{Runs}(\mathcal{A}_2)$ with $\text{start}(t_1) = r \in R$ and $\text{end}(t_m) = s \in S$. By Lemma 13, as P is horizontal and r and s are in H, $\mathbf{r}$ is a direct run. Hence, t_1 and t_m is not a matching parenthesizing transition pair. Thus, it is possible to replace both $\langle$ and $\rangle$ with $\langle^{\text{first}}$ and $\rangle^{\text{last}}$, in the first and in the last transitions, if necessary. We can also substitute their closing and opening partners by $\rangle^{\text{first}}$ and $\langle^{\text{last}}$, if needed. Therefore, the construction of $\mathcal{A}_3$ ensures that $[p, P, q]_{\mathcal{A}_3}$ holds. □

Definition 28. *We say that a parenthesizing automaton is in* horizontal normal form *if it has a single initial state i_h, and a single final state f_h, and both i_h and f_h are horizontal states, moreover, there is no transition into i_h or from f_h. Automata in* vertical normal form *can be defined accordingly.*

Lemma 29. *For every parenthesizing automaton $\mathcal{A}$, there exists an equivalent parenthesizing automaton $\mathcal{A}_h$ in horizontal normal form and an equivalent parenthesizing automaton $\mathcal{A}_v$ in vertical normal form.*

Proof. First we prove that for every parenthesizing automaton $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ there exists a parenthesizing automaton $\mathcal{A}^{\cap\mathcal{H}}$ in horizontal normal form that accepts exactly the horizontal biposets accepted by $\mathcal{A}$, i.e.,

$$L(\mathcal{A}^{\cap\mathcal{H}}) = L(\mathcal{A}) \cap \mathcal{H},$$

where $\mathcal{H}$ denotes the set of all horizontal biposets. Indeed, let

$$T := \{\, (s,t) \mid \exists i \in I \cap V, \exists f \in F \cap V, \exists \langle, \rangle \in \Omega \;:\; (i, \langle, s),\ (t, \rangle, f) \in \gamma \,\},$$

and assume that $T = \{\, (s_1,t_1), (s_2,t_2), \ldots, (s_n,t_n) \,\}$. Moreover, let $\mathcal{A}_0$ be the automaton without transitions, with two states only, an initial horizontal state i_h, and a final horizontal state f_h.

Now, with the help of the substitution product, we define

$$\begin{aligned} \mathcal{A}_1 &:= \mathcal{A}_0 *_{[i_h \to I\cap H, F\cap H \to f_h]} \mathcal{A}, \\ \mathcal{A}_{k+1} &:= \mathcal{A}_k *_{[i_h \to s_k, t_k \to f_h]} \mathcal{A} \quad \text{for } k = 1, \ldots, n, \\ \mathcal{A}^{\cap\mathcal{H}} &:= \mathcal{A}_{n+1}. \end{aligned}$$

Using Lemma 27 and Corollary 17, it is straightforward to check that $L(\mathcal{A}^{\cap\mathcal{H}}) = L(\mathcal{A}) \cap \mathcal{H}$, as claimed.

Similarly, there is an automaton $\mathcal{A}^{\cap\mathcal{V}}$ in vertical normal form which accepts exactly the vertical biposets accepted by $\mathcal{A}$. Let i_v and f_v denote the (single) initial and final vertical states of $\mathcal{A}^{\cap\mathcal{V}}$.

Now, we can construct $\mathcal{A}_h$ by taking the disjoint union of $\mathcal{A}^{\cap\mathcal{H}}$ and $\mathcal{A}^{\cap\mathcal{V}}$ and adding two new parenthesizing transitions, $(i_h, \{, i_v)$ and $(f_v, \}, f_h)$, where $\{$ and $\}$ is a new pair of parentheses. Of course, we do not regard i_v and f_v as initial and final states any longer. In order to accept the singleton biposets, we also define (i_h, σ, f_h) for each singleton biposet $\sigma \in L(\mathcal{A})$. As $\mathcal{A}^{\cap\mathcal{H}}$ accepts all horizontal, and $\mathcal{A}^{\cap\mathcal{V}}$ all vertical biposets of $L(\mathcal{A})$, the resulting automaton is equivalent to $\mathcal{A}$. Again, $\mathcal{A}_v$ can be defined symmetrically. □

Now, we are ready to prove the converse of Theorem 25.

Theorem 30. *Every recognizable language of infinite constructible biposets is regular.*

Proof. Suppose that a language $L \subseteq \mathrm{ISPB}(\Sigma)$ of infinite constructible biposets is recognized by a morphism $\varphi : \omega\mathrm{SPB}(\Sigma) \to \mathcal{B}$, where $\mathcal{B} = (B_F, B_I)$ is a finite ω-bisemigroup, and $L = \varphi^{-1}(F)$ for some $F \subseteq B_I$.

Let us call an element $e \in B_F$ *horizontally idempotent* if it is idempotent with respect to the horizontal product, i.e., $e \bullet e = e$. Similarly, e is said to be *vertically idempotent* if $e \circ e = e$. This notion is important for the fact that every primitive biposet $P_0 * P_1 * \ldots$ can be written in the form $P'_0 * P'_1 * \ldots$, where $\varphi(P'_0) = b$ and $\varphi(P'_i) = e$ for all $i > 0$, where e is a $*$-idempotent element of B_F. This follows from an application of the Ramsey-theorem, cf. [20]. We can even assume that $b = b * e$, but we do not need this now.

Thus, if we omit P'_0 from the above biposet, then the remaining primitive biposet is $P'_1 * P'_2 * \ldots$, where $\varphi(P'_i) = e$ for all $i > 0$. Let us call those biposets that can be written in such a form *e-$*$-primitive*.

For a given e and $*$, the set of all e-$*$-primitive biposets is easy to accept by a parenthesizing Büchi-automaton $\mathcal{A}_{e,*}$ constructed as follows. Since $\varphi^{-1}(e)$ is a recognizable set of finite sp-biposets, it is also regular by Theorem 18. So there is a parenthesizing (finite) automaton $\mathcal{A}$ accepting $\varphi^{-1}(e)$. Moreover, it can be assumed that $\mathcal{A}$ is in $*$-normal form (i.e. in horizontal normal form if $* = \bullet$, or in vertical normal form if $* = \circ$). Thus, $\mathcal{A}$ has a single initial state i and a single finite state f, both of them are of type $*$. We can transform $\mathcal{A}$ into $\mathcal{A}_{e,*}$ just by merging i and f. We will refer to this fused state as the *basic state* of $\mathcal{A}_{e,*}$. Now, it is obvious that if we regard $\mathcal{A}_{e,*}$ as a Büchi-automaton with its basic state as the only initial and final state, it accepts exactly the e-$*$-primitive biposets.

Recall that according to (1) the normal form of a constructible biposet is

$$P_1 *_1 (P_2 *_2 \ldots (P_k *_k (Q_1 *_{k+1} Q_2 *_{k+1} \ldots))).$$

We can assume that except for a finite factor Q', the biposet $Q_1 *_{k+1} Q_2 *_{k+1} \ldots$ is e-$*_{k+1}$-primitive for some $*_{k+1}$-idempotent e. Thus, we only need to build our automaton in a way that it can also process the finite "introductory slice" $P_1 *_1 (P_2 *_2 (\ldots P_k *_k (Q' *_{k+1}$ before the e-$*_{k+1}$-primitive tail.

Assume that $B_I = \{t_1, t_2, \ldots, t_m\}$. We start to construct a Büchi-automaton $\mathcal{A}$ from the horizontal states H_0, vertical states V_0, with separating transitions β, where

$$\begin{aligned} H_0 &:= \{ t_1^\bullet, t_2^\bullet, \ldots, t_m^\bullet \}, \\ V_0 &:= \{ t_1^\circ, t_2^\circ, \ldots, t_m^\circ \}, \text{ and} \\ \beta &:= \{ (t_i^\bullet, [, t_i^\circ), (t_i^\circ, [, t_i^\bullet) \mid i = 1, \ldots, m \}. \end{aligned}$$

For all $b \in B_F$, there is a parenthesizing (finite) automaton $\mathcal{A}_b$ recognizing $\varphi^{-1}(b)$. Similarly as before, we will incorporate these finite automata into $\mathcal{A}$.

More precisely, if p and q are states of $\mathcal{A}$ of the same type, say $*$, then one can take a copy of $\mathcal{A}_b$ in $*$-normal form and merge its initial state i and final state f with the states p and q of $\mathcal{A}$, respectively. In the sequel, we refer to this construction as *extension* of $\mathcal{A}$ (between p and q) by $\varphi^{-1}(b)$. Let us denote it by

$$p \overset{\varphi^{-1}(b)}{\Longrightarrow} q.$$

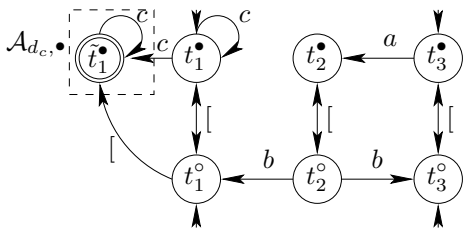

Figure 5: A Büchi-automaton accepting the recognizable language of Example 8

We need to add the following extensions to $\mathcal{A}$:

$$t_i^* \overset{\varphi^{-1}(b)}{\Longrightarrow} t_j^* \quad \text{for all } t_i = b * t_j, \quad b \in B_F,\, t_i, t_j \in B_I, * \in \{\bullet, \circ\}.$$

Now we obviously have

$$[\, t_i^*, P, t_j^* \,]_{\mathcal{A}} \Leftrightarrow t_i = \varphi(P) * t_j \text{ for any } P \in \mathrm{SPB}(\Sigma).$$

Furthermore,

$$[\, t_i^*, P_1 *_1 (P_2 *_2 (\dots P_{k-1} * (P_k *_k, t_j^* \,]_{\mathcal{A}} \Leftrightarrow t_i = \varphi(P_1) *_1 (\varphi(P_2) *_2 (\dots \varphi(P_k) *_k t_j)),$$

where the left hand side abbreviates the first part of an infinite run of $\mathcal{A}$ (as a Büchi-automaton) on an infinite biposet beginning with $P_1 *_1 (P_2 *_2 \dots (P_k *_k$.

Next, add an instance of $\mathcal{A}_{e,*}$ for each $*$-idempotent element e in B_F to $\mathcal{A}$, and assure the reachability of the new components by adding some new transitions. In more detail, for $\mathcal{A}_{e,*}$, consider $t := e^{\omega_*}$ and duplicate each transition arriving at t^* using the same source and label, but with the target of the basic state of $\mathcal{A}_{e,*}$ instead of t^*.

Our last task is to settle the initial and the final states. Let the initial states be the states $t^\bullet$ and t° for each $t \in F$, and set the basic states of the components $\mathcal{A}_{e,*}$-s as final states.

Finally, it can be argued by induction on the rank of the biposets that in fact this automaton accepts $L = \varphi^{-1}(F)$. We omit the formal proof. □

Example 31. Figure 5 shows a parenthesizing Büchi-automaton that was constructed according to the proof of Theorem 30 from the morphism φ, the ω-bisemigroup B, and the set $F \subseteq B$ of Example 8. We omitted the two shrink states $\underline{0}^\bullet, \underline{0}^\circ$ that correspond to the infinite zero element, and also the transitions pointing to them. We should admit that this example represents a somewhat special case, since, for every $x \in B_F$ the extension by $\varphi^{-1}(x)$ is a single transition, and there is only one idempotent in B_F. Of course, in the general case the constructed automaton can have a more complex structure.

6 From regularity to MSO-definability

In this section, we prove the equivalence of regularity and MSO-definability. First of all, it is not hard to demonstrate that MSO-definability implies recoginzability, and hence regularity. This can be shown by formula induction using the closure properties of the recognizable sets, more precisely, the closure under Boolean operations and direct letter-to-letter morphisms. See Chapter III.1 of Straubing [21] for a similar argument. Thus, we have the following theorem.

Theorem 32. *Every* MSO*-definable language of infinite constructible biposets is regular.*

The rest of the paper is devoted to the converse of the previous theorem:

Theorem 33. *Every regular language of infinite constructible biposets is* MSO*-definable.*

Before the proof, let us introduce a few definitions and lemmas.

The notion of clan is one of our key definition, that can be easily adapted from the theory of 2-structures [2] and texts [4, 13]. If $(P, <_h, <_v, \lambda)$ is a finite or infinite constructible sp-biposet, a subset X of P is said to be a *clan* of P if for all $x, y \in X$, $z \in P \setminus X$ and for each relation $\rho \in \{<_h, <_v, >_h, >_v\}$

$$x \rho z \Leftrightarrow y \rho z.$$

Two clans X and Y *overlap* if $X \cap Y \neq \emptyset$, $X \setminus Y \neq \emptyset$ and $Y \setminus X \neq \emptyset$. A clan is called *prime clan* if it does not overlap with any other clan. A clan of P is a *proper clan* if it is neither a singleton, nor equal to P.

Example 34. In the biposet of Example 5, the clans of P are the following: the singletons, P, $\{1, 2, 3, 4\}$, $\{2, 3, 4\}$, $\{3, 4\}$, $\{2, 3, 4, 5, 6\}$ and $\{5, 6\}$. Since, only $\{1, 2, 3, 4\}$ and $\{2, 3, 4, 5, 6\}$ overlap, the other clans are prime clans as well. Thus, the proper prime clans are $\{2, 3, 4\}$, $\{3, 4\}$ and $\{5, 6\}$. As we will see later in Lemma 38, these are the sets which are surrounded by parentheses in the term representation of P.

The proof of the following lemma is trivial and is left to the reader.

Lemma 35. *The property of being a clan, a prime clan or a proper prime clan can be expressed in* MSO *logic.*

Lemma 36. *If $P = (P, <_h, <_v, \lambda)$ is a (finite or infinite) constructible biposet, then $<_h \cup <_v$ is a linear order on P.*

Proof. By induction on the construction of P. □

In the sequel, let $<$ denote the $<_h \cup <_v$ relation, and we interpret the functions $+$ and $-$ also according to this relation.

By definition, clans form sections (or intervals) with respect to $<$. That is, if X is a clan then $x \in X$, $y \in X$ and $x < z < y$ imply $z \in X$. Thus, we can talk about *prefix* and *suffix* relations among the clans of P. Formally,

$$\begin{aligned} \mathrm{Prefix}(X,Y) &:= X \subsetneq Y \wedge \forall x \forall y \big(y < x \wedge X(x) \wedge Y(y) \rightarrow X(y)\big); \\ \mathrm{Suffix}(X,Y) &:= X \subsetneq Y \wedge \forall x \forall y \big(y > x \wedge X(x) \wedge Y(y) \rightarrow X(y)\big), \end{aligned}$$

where $X \subsetneq Y$ means that X is a proper subset of Y. Thus, under prefix and suffix relations we always mean proper prefix and suffix.

Recall that P^{tr} denotes the tree representation of P. Two (or more) subtrees of a tree are said to be *sibling subtrees* if their roots have the same parent.

Lemma 37. *Suppose that P is a (finite or infinite) constructible biposet and X is a subset of P, then*

(i) X is a clan of P if and only if there are consecutive sibling subtrees in P^{tr} such that X is exactly the union of the sets of leaves of these subtrees;

(ii) X is a prime clan of P if and only if X is the set of leaves of a single subtree of P^{tr}.

Proof. We start with the proof of case (i). The necessity of the condition is based on the following observation. Suppose that x and y are vertices of P. As we mentioned earlier, we can regard them as two leaves in the tree representation P^{tr}. The (horizontal or vertical) type of the order relation between x and y is solely determined by the label of their lowest common ancestor node. For this reason, let u denote the lowest common ancestor of x and y. If the label of u in P^{tr} is $\bullet$, then $x <_h y$ or $y <_h x$; if the label is $\circ$, then x and y are ordered vertically. We can also easily decide whether x is less or greater than y. Consider u_x and u_y, the children of u that are ancestors of x and y, respectively. Now, x is less than y if and only if u_x is less than u_y according to the order of the children nodes at u. It follows that if a subset X of P satisfies the condition of (i), then it also fulfills the requirements of being a clan. Indeed, the order relation between a vertex x from X and a vertex y outside X is independent of the choice of x from X.

For the converse direction, suppose, on the contrary, that X is a clan, but the condition does not hold. First let v denote the lowest common ancestor node of the vertices of X. Now consider those children of v that have leaves in X, and then take the subtrees generated by them. The condition can be violated in two ways. Either these subtrees are not consecutive or there is a subtree that has leaves both from X and $P \setminus X$. In the first case, it is straightforward to show that X is not a clan. In the second case, consider a child u of v that has a leaf x in X and has a leaf z in $P \setminus X$ as well. We can even assume that x and z are descendant of different children of u, so their lowest common ancestor is u. Moreover, there must also be a vertex y in X whose lowest common ancestor with x is v, otherwise the lowest common ancestor of the set X could not be v. But, x and z are related by an order of type determined by the label of u, while y and z are related by an order of type determined by the label of v. As u is a child of v, their labels in P^{tr} are different.

Consequently, the types of the order relations between x and z and between y and z are also different. This contradicts to our assumption that X is a clan.

Now case (ii) easily follows from case (i), since if X consists of the leaves of two or more (but not all) subtrees of a given node, then overlapping clans can be constructed by (i) showing that X is not prime. □

The following lemma is important for a later proof. It connects the various representations of biposets with the use of parentheses of automata. Recall that $\widehat{\Sigma} = \Sigma \cup \{ \bullet, \circ, \langle, \rangle \}$.

Lemma 38. *For any $P \in \mathrm{SPB}(\Sigma)$, $X \subseteq P$, parenthesizing automaton $\mathcal{A}$, and $\mathbf{r} \in \mathrm{Runs}(\mathcal{A})$ with $\mathrm{Biposet}(\mathbf{r}) = P$, the following statements are equivalent:*

(i) X is a proper prime clan of P.

(ii) X is the set of leaves of a proper subtree of P^{tr}.

(iii) P^{tm} can be written as $P^{tm} = u\langle X^{tm}\rangle v$, where $u, v \in \widehat{\Sigma}^*$, and the subword X^{tm} above corresponds to those vertices of P that are in X.[1]

(iv) $\mathbf{r}$ is of the form $\mathbf{r} = \mathbf{r_1} t_1 \mathbf{r_x} t_2 \mathbf{r_2}$, where $\mathbf{r_1 r_2} \neq \varepsilon$, t_1 and t_2 form a matching parenthesizing transition pair in $\mathbf{r}$, and $\mathbf{r_x}$ denotes the direct subrun of $\mathbf{r}$ on the vertices of X.

Proof. The equivalence of (i) and (ii) follows from Lemma 37. The equivalence of (ii) and (iii) is obvious, it expresses a usual correspondence between the term and the tree representations. Finally, the equivalence of (iii) and (iv) is a consequence of Lemma 13. □

Now we are ready to start the proof of the main theorem of this section.

Proof of Theorem 33. For sp-biposet languages, the equivalence of MSO-definability and recognizability (and hence regularity) directly follows from an analogous equivalence result on text languages shown by Hoogeboom and ten Pas [13]. See also [6] about the relationship between texts and biposets. Even though, here we outline an alternative proof of this fact, since it will serve as the base for the proof of the infinite case. Our argument does not rely on the equivalence of recognizability and MSO-definability of finite binary trees, but shows how the operations of parenthesizing automata can be described by logical formulas. We start with the finite case, i.e., with the case of sp-biposets, and explain the necessary changes for the infinite case later.

Let $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ be a parenthesizing automaton accepting an sp-biposet language L. Our aim is to construct an MSO-formula φ for which $L_\varphi = L$.

The proof of Lemma 29 implies that we may assume that $\mathcal{A}$ accepts via direct and singleton runs only. Therefore, it is sufficient to construct a formula $\varphi_{i,f}$ which expresses the fact that $\mathcal{A}$ has a direct run from an initial state i to a final state f.

[1] Note that the subword X^{tm} can also appear at other places in the word P^{tm}.

We use second order variables for storing information about the states of the runs. In more detail, two types of monadic second order variables are used. First, let X_s be a variable for each state s in S, and let $Z_{\langle_j\rangle_j}$ denote a variable for each pair of parentheses in Ω. Formally,

$$\mathrm{Var}_{\mathcal{A}} := \{\, X_s \mid s \in S \,\} \cup \{\, Z_{\langle_j\rangle_j} \mid \langle_j, \rangle_j \in \Omega \,\}.$$

The general form of $\varphi_{i,f}$ is the following

$$\varphi_{i,f} := \exists X_{s_1} \exists X_{s_2} \ldots \exists X_{s_m} \exists Z_{\langle_1\rangle_1} \exists Z_{\langle_2\rangle_2} \ldots \exists Z_{\langle_n\rangle_n} \psi_{i,f},$$

where, $\psi_{i,f}$ expresses the fact that the values of our variables in fact encode a direct run of $\mathcal{A}$ from i to f.

We need to check three conditions. First, the run must start from i. Second, it must end in f. Third, we need to verify that $\mathcal{A}$ has correct transitions everywhere between the states indicated by the variables. We handle the labeled and the parenthesizing transitions of the run separately.

For labeled transitions, the usual technique (cf. [21]) can be applied, that is, the intended meaning of $x \in X_s$ is that $\mathcal{A}$ reads position x in state s. The storage of parenthesizing transitions is more involved. Fortunately, by Lemma 38 the use of parentheses is always around proper prime clans. But we also need to arrange a unique position in the biposet for each matching parenthesizing transition pair used during the run.

For this purpose, the following rule can be applied. If a proper prime clan is a prefix of an other proper prime clan, then let the *designated position* be its last position, otherwise let the designated position be its first position. Thus, the statement that z is the designated position of a proper prime clan X, can be expressed as:

$$\begin{aligned} \mathrm{Dp}(z, X) \quad := \quad & \big[\neg\exists Y \big(\,\mathrm{PPC}(Y) \wedge \mathrm{Prefix}(X, Y)\,\big) \wedge \mathrm{First}(z, X)\,\big] \\ & \vee\big[\,\exists Y \big(\,\mathrm{PPC}(Y) \wedge \mathrm{Prefix}(X, Y)\,\big) \wedge \mathrm{Last}(z, X)\,\big], \end{aligned}$$

where $\mathrm{PPC}(Y)$ states that Y is a proper prime clan, and $\mathrm{First}(z, X)$ ($\mathrm{Last}(z, X)$) is true if and only is z is the first (last, resp.) position of the clan X.

Now, it can be verified that the prime property implies that the designated positions of any two proper prime clans do not coincide.

Lemma 39. *Different proper prime clans have different designated positions.*

Proof. For a contradiction, assume that X and Y are different proper prime clans, but z is their common designated position. If z is the first position of both X and Y, then either X is a prefix of Y, or Y is a prefix of X, in contradiction with the definition of designated position. If z is the first position of one clan and the last position of the other clan, then X and Y overlap, leading to a contradiction again. Finally, assume that z is the last position of both X and Y, and $X \subseteq Y$. By definition, there is a proper prime clan X' such that X is a prefix of X'. Therefore, in this case, Y and X' overlap, again a contradiction. □

Proof of Theorem 33, continued. If x is a position, then the intended meaning of $Z_{\langle_j,\rangle_j}(x)$ is that $\mathcal{A}$ uses parentheses $\langle_j,\rangle_j$ (more precisely the unique pair of transitions labeled $\langle_j$ and $\rangle_j$) before and after processing the proper prime clan whose designated position is x.

As usual, we require that every position belongs to exactly one X_s:

$$\psi_1 \quad := \quad \forall x\Big[\bigvee_{s\in S} X_s(x) \quad \wedge \bigwedge_{\substack{q_1,q_2\in S,\\ q_1\neq q_2}} \big(\neg X_{q_1}(x)\vee\neg X_{q_2}(x)\big)\Big].$$

Moreover, the designated positions of proper prime clans must also belong to a unique set $Z_{\langle_i\rangle_i}$:

$$\psi_2 := \forall x\Bigg[\exists X\big(\mathrm{PPC}(X)\wedge \mathrm{Dp}(x,X)\big) \\ \rightarrow \bigvee_{\langle_j,\rangle_j\in\Omega} Z_{\langle_j\rangle_j}(x) \quad \wedge \bigwedge_{\substack{\langle_j,\rangle_j\in\Omega,\\ \langle_k,\rangle_k\in\Omega,\\ j\neq k}} \big(\neg Z_{\langle_j\rangle_j}(x)\vee\neg Z_{\langle_k\rangle_k}(x)\big)\Bigg].$$

We know that for all positions x, the state of $\mathcal{A}$ before processing this position is indicated by a unique state p such that $x\in X_p$. Moreover, q, *the state after reading position* x, can be computed as follows. First we observe whether P has a proper prime clan that ends at x. If so, then we determine the smallest such clan, and q is the starting state of the closing parenthesizing transition of that clan. Of course, this state can be determined by observing the designated position of the clan. If there is no proper prime clan that ends at x, then q is indicated at the position $x+1$ or at the greatest prime clan that starts at $x+1$. Besides, if x is the last position, then q must be the final state of the run. Finally, we can check whether $\mathcal{A}$ in fact has a labeled transition with the label of x between p and q. We should perform this verification for every position x. The precise algorithm of this computation and the way of converting it to an MSO-formula are presented in the appendix.

We can also check the correctness of the parenthesizing transitions by a similar procedure. For all proper prime clans, we compute four states of the encoded run: the states before and after the opening, and before and after the closing parenthesizing transitions around the clan. In the pseudocode presented in the appendix, these states are denoted by *ob*, *oe*, *cb* and *ce*. Then, it is straightforward to check whether $\mathcal{A}$ has a parenthesizing transition pair between the computed states, and whether the labels of these transitions are indicated at the designated position of the clan. It is also a nontrivial computation, since we must take into consideration various inclusion relations of the clans. For more detail, see the appendix again.

Finally, note that the algorithm of verification can be transformed into an MSO-formula ψ_3. Hence we can write $\psi_{i,f}$ as $\psi_{i,f} := \psi_1\wedge\psi_2\wedge\psi_3$. This completes the proof for finite constructible biposets.

We now turn to a brief discussion of the infinite case. Here we only describe the necessary changes compared to the finite case. First, the adaptation of Lemma 38 for infinite constructible biposets is the following. Note that we must distinguish between finite and infinite clans, but this distinction is in parallel to the use of the parenthesizing and separating parentheses.

Lemma 40. *For any $P \in \mathrm{ISPB}(\Sigma)$, $X \subseteq P$, parenthesizing Büchi-automaton $\mathcal{A}$, infinite run $\mathbf{r} \in Runs(\mathcal{A})$ with $\mathrm{Biposet}(\mathbf{r}) = P$, the following statements are equivalent:*

(i) X is a finite proper prime clan of P.

(ii) X is the set of leaves of a finite proper subtree of P^{tr}.

(iii) P^{tm} can be written as $P^{tm} = u\langle X^{tm}\rangle v$, where $u, v \in \widehat{\Sigma}'^{*}$, and the subword X^{tm} above corresponds to those vertices of P that are in X.

(iv) $\mathbf{r}$ is of the form $\mathbf{r} = \mathbf{r_1} t_1 \mathbf{r_x} t_2 \mathbf{r_2}$, where t_1 and t_2 is a matching parenthesizing transition pair, and $\mathbf{r_x}$ denotes the direct subrun of $\mathbf{r}$ on the vertices of X.

Moreover, the following statements are also equivalent.

(i') X is an infinite proper prime clan of P.

(ii') X is the set of leaves of an infinite proper subtree of P^{tr}.

(iii') P^{tm} can be written as $P^{tm} = u[X^{tm}$, where $u \in \widehat{\Sigma}'^{*}$, and the subword X^{tm} above corresponds to those vertices of P that are in X.

(iv') $\mathbf{r}$ is of the form $\mathbf{r} = \mathbf{r_1} t \mathbf{r_x}$, where $\mathbf{r_1} \neq \varepsilon$, t is a separating transition of $\mathcal{A}$, and $\mathbf{r_x}$ denotes the direct subrun of $\mathbf{r}$ on the vertices of X.

Proof of Theorem 33, completed. It is trivial that we can express the finiteness of clans, as

$$\mathrm{Finite}(X) \quad := \neg\exists z\, \mathrm{Last}(z, X).$$

Hence, we can easily locate the separating transitions and check their correctness, as well. Furthermore, we have no trouble formulating the acceptance condition: a finite state has to appear infinitely often as outer state of the encoded run. We leave the reader to verify the correctness of the formulas bellow.

$$\begin{aligned}\psi_{\mathrm{acc}} := \bigvee_{f\in F} \forall z\, \exists X \Big[& \mathrm{MaxFiniteClan}(X) \wedge \mathrm{OuterState}_f(X) \\ & \wedge \forall x \big(\mathrm{Last}(x, X) \rightarrow (z < x) \big) \Big];\end{aligned}$$

$$\begin{aligned}\mathrm{MaxFiniteClan}(X) := {} & \mathrm{Finite}(X) \wedge \mathrm{Clan}(X) \\ & \wedge \neg\exists Y \big(\mathrm{Finite}(Y) \wedge \mathrm{Clan}(Y) \wedge X \subsetneq Y \big);\end{aligned}$$

$$\begin{aligned}\mathrm{OuterState}_f(X) := {} & \big[\,\mathrm{Singleton}(X) \wedge \exists x\,\big(X(x) \wedge X_f(x)\big)\,\big] \\ & \vee \big[\,\mathrm{PPC}(X) \wedge \bigvee_{\substack{(f,\langle_k,p)\in\gamma \\ \langle_k\in\Omega, p\in S}} \exists z\,\big(\mathrm{Dp}(z,X) \wedge Z_{\langle_k\rangle_k}(z)\big)\,\big].\end{aligned}$$

Of course, here the formulas $\mathrm{Finite}(X)$, $\mathrm{Clan}(X)$ and $\mathrm{Singleton}(X)$ have their expected meanings. □

Finally, we summarize the main results of the paper.

Theorem 41. *Let $L \subseteq \mathrm{ISPB}(\Sigma)$. Then L is recognizable if and only if L is regular if and only if L is* MSO*-definable.*

Acknowledgement. The author would like to thank the anonymous referees for their numerous valuable comments and suggestions.

Appendix

In this appendix, we give the detailed algorithm of verification of the correctness of encoded runs. And we briefly describe how to build formula ψ_3 that realizes the algorithm.

Suppose that $\mathcal{A} = (S, H, V, \Sigma, \Omega, \delta, \gamma, I, F)$ is a parenthesizing automaton, $i \in I$ and $f \in F$. Recall that $\mathrm{Var}_{\mathcal{A}} = \{\, X_{s_i} \mid s_i \in S \,\} \cup \{\, Z_{\langle_i\rangle_i} \mid \langle_i, \rangle_i \in \Omega \,\}$. Let $P = (P, <_h, <_v, \lambda) \in \mathrm{SPB}(\Sigma)$ denote an sp-biposet, and assume that η is an evaluation of the monadic second order variables, i.e.,

$$\eta : \mathrm{Var}_{\mathcal{A}} \to \mathcal{P}(P),$$

where $\mathcal{P}(P)$ denotes the power-set of P. Moreover, assume that P with η satisfies formulas ψ_1 and ψ_2 on page 25.

The following algorithm decides whether η encodes a direct run of $\mathcal{A}$ on P that starts from i and ends in f. For the sake of simplicity, we write X_j instead of $\eta(X_j)$. Moreover, in the names of the procedure calls below, "Clan" always means a proper prime clan of P.

Unfortunately, in the definition of function NextState a difficulty arises. As $\mathcal{A}$ is nondeterministic, for a given position x and $s \in S$, there can be more than one t such that $(s, \lambda(x), t) \in \delta$ holds. But when we convert our algorithm into an MSO-formula, we only need to test whether $\mathrm{NextState}(x) = t$ holds, which resolves the problem.

The pseudocode in Lines 1–10 verifies that the run starts from i and ends in f. The code in Lines 11–22 checks the correctness of the labeled transitions, while Lines 23–49 verify the parenthesizing transitions. The proof of correctness of the algorithm is omitted, but Figures 6–9 should help the reader to establish it.

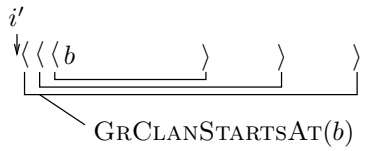

Figure 6: The computation of i' in Line 3.

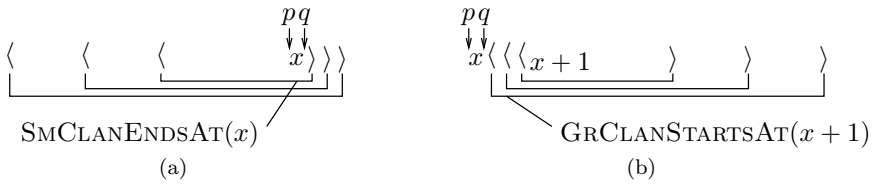

Figure 7: The computation of q in Line 14 (a) and in Line 18 (b).

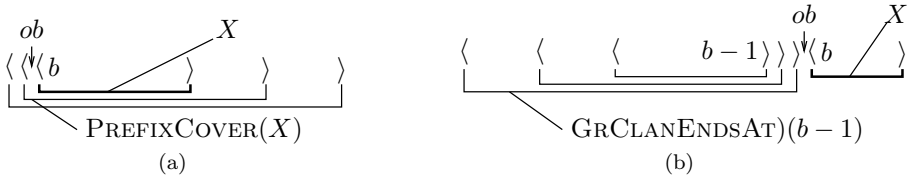

Figure 8: The computation of *ob* in Line 27 (a) and in Line 31 (b).

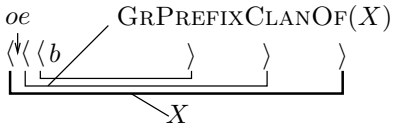

Figure 9: The computation of *oe* in Line 34.

Algorithm Correct-Run$(\mathcal{A}, i, f, P, \eta)$
1 $b \leftarrow$ FirstOf(P)
2 **if** IsClanStartsAt(b)
3 **then** $i' \leftarrow$ StartOfOpPar(GrClanStartsAt(b))
4 **else** $i' \leftarrow$ State(b)
5 $e \leftarrow$ LastOf(P)

```
6  if IsClanEndsAt(e)
7    then f' ← EndOfClPar(GrClanEndsAt(e))
8    else f' ← NextState(e)
9  if i ≠ i' or f ≠ f'
10   then return 'no'
11 for all x ∈ P
12     do p ← State(x)
13        if IsClanEndsAt(x)
14          then q ← StartOfClPar(SmClanEndsAt(x))
15          else if IsLastPosition(x)
16                 then q ← f
17                 else if IsClanStartsAt(x + 1)
18                        then q ← StartOfOpPar(GrClanStartsAt(x + 1))
19                        else q ← State(x + 1)
20        σ ← λ(x)
21        if not (p, σ, q) ∈ δ
22          then return 'no'
23 for all proper prime clans X ⊆ P
24     do b ← FirstOf(X)
25        e ← LastOf(X)
26        if IsPrefixOfClan(X)
27          then ob ← EndOfOpPar(PrefixCover(X))
28          else if IsFirstPosition(b)
29                 then ob ← i
30                 else if IsClanEndsAt(b − 1)
31                        then ob ← EndOfClPar(GrClanEndsAt(b − 1))
32                        else ob ← NextState(b − 1)
33        if IsPrefixClanIn(X)
34          then oe ← StartOfOpPar(GrPrefixClanOf(X))
35          else oe ← State(b)
36        if IsSuffixClanIn(X)
37          then cb ← EndOfClPar(GrSuffixClanOf(X))
38          else cb ← NextState(e)
39        if IsSuffixOfClan(X)
40          then ce ← StartOfClPar(SuffixCover(X))
41          else if IsLastPosition(e)
42                 then ce ← f
43                 else if IsClanStartsAt(e + 1)
44                        then ce ← StartOfOpPar(GrClanStartsAt(e + 1))
45                        else ce ← State(e + 1)
46        k ← IndexOfParUsedAround(X)
47        if not(ob, ⟨k, oe), (cb, ⟩k, ce) ∈ γ
48          then return 'no'
49 return 'yes'
```

The input-output specifications of the predicates and functions used in the algorithm are the following:

IsFirstPosition(x) / IsLastPosition(x)
input: a position $x \in P$;
output: 'yes' if x is the first/last position of P;
'no' otherwise.

IsClanStartsAt(x) / IsClanEndsAt(x)
input: a position $x \in P$;
output: 'yes' if there is a proper prime clan $X \subseteq P$ whose first/last position is x;
'no' otherwise.

IsPrefixOfClan(X) / IsSuffixOfClan(X)
input: a proper prime clan $X \subseteq P$;
output: 'yes' if there is a proper prime clan Y such that X is a prefix/suffix of Y;
'no' otherwise.

IsPrefixClanIn(X) / IsSuffixClanIn(X)
input: a proper prime clan $X \subseteq P$;
output: 'yes' if there is a proper prime clan Z such that Z is a prefix/suffix of X;
'no' otherwise.

State(x)
input: a position $x \in P$;
output: a state $s \in S$ in which $\mathcal{A}$ reads position x, i.e., $x \in X_s$.

NextState(x)
input: a position $x \in P$;
output: a state $t \in S$ at which $\mathcal{A}$ arrives after reading the position x, i.e., $x \in X_s$ and $(s, \lambda(x), t) \in \delta$.

FirstOf(X) / LastOf(X)
input: a proper prime clan $X \subseteq P$;
output: the first/last position of X.

StartOfOpPar(X) / EndOfOpPar(X)
input: a proper prime clan $X \subseteq P$;
output: a state $s \in S$ such that the s is the source/target of an opening parenthesizing transition $(s, \langle_j, t)$ / $(r, \langle_j, s) \in \gamma$, and this transition was used immediately before X, i.e., the designated position of X is in $Z_{\langle_j\rangle_j}$.

StartOfClPar(X) / EndOfClPar(X)
input: a proper prime clan $X \subseteq P$;
output: a state $s \in S$ such that the s is the source/target of a closing parenthesizing transition $(s, \rangle_j, t)$ / $(r, \rangle_j, s) \in \gamma$, and this transition was used immediately after X, i.e., the designated position of X is in $Z_{\langle_j\rangle_j}$.

SmClanEndsAt(x) / GrClanEndsAt(x)
input: a position $x \in P$;
output: the smallest/greatest proper prime clan of P that ends at position x.

GrClanStartsAt(x)
input: a position $x \in P$;
output: the greatest proper prime clan of P that starts at position x.

GrPrefixClanOf(X) / GrSuffixClanOf(X)
input: a proper prime clan $X \subseteq P$;
output: the greatest proper prime clan $Y \subseteq X$ that is a proper prefix/suffix of X.

PrefixCover(X) / SuffixCover(X)
input: a proper prime clan $X \subseteq P$;
output: the smallest proper prime clan Y such that is X is a proper prefix/suffix of Y.

IndexOfParUsedAround(X)
input: a proper prime clan X;
output: an index k such that the parentheses $\langle_k, \rangle_k$ were used before and after X in the encoded run, i.e., the designated position of X is in $Z_{\langle_k\rangle_k}$.

Finally, we outline the transformation of the algorithm into formula ψ_3. The following observations lead to this transformation.

1. All predicates of the algorithm can be expressed by MSO-formulas. For example, IsPrefixOfClan(X) can be formulated as
$$\exists Y \big(\mathrm{PPC}(Y) \wedge \mathrm{Prefix}(X, Y) \big)$$
2. For any function $f(x_1, \ldots, x_l)$ of the algorithm and for any element c in the range of f, the fact $f(x_1, \ldots, x_l) = c$ can also be expressed by an MSO-formula. For example, for any state s in S, StartOfOpPar(X) $= s$ can be written as
$$\bigvee_{j \in J} \exists z \big(\mathrm{Dp}(z, X) \wedge Z_{\langle_j\rangle_j}(z) \big),$$
where $J = \{\, j \mid \exists t \in S, (s, \langle_j, t) \in \gamma \,\}$ is a finite set.
3. The variables whose values are not positions or sets of positions of P, all take their values from a finite set. Namely, i, f, i', f', p, q, ob, oe, cb, ce take values from S, σ from Σ, and k is an index of a parenthesis in Ω.
4. The composition of functions can be handled with the help of auxiliary variables. For example, StartOfOpPar(GrClanStartsAt(b)) $= s$ can be expressed as
$$\exists Z \big(\textsc{GrClanStartsAt}(b) = Z \wedge \textsc{StartOfOpPar}(Z) = s \big)$$

5. Assignments like $y \leftarrow f(x_1, \ldots, x_l)$ can be treated as follows. We can consider all possible values c in the range of y in advance, and at the points of the assignments we can test whether $f(x_1, \ldots, x_l) = c$ holds. If the range of y is P or the power-set $\mathcal{P}(P)$, i.e., y is a 'standard' first or second order variables, then existential quantification can be used. On the other hand, if y is not 'standard', then it has a finite range by point 3. Hence we can use disjunction over this finite range. For example, we can start the formula realizing Lines 12–22 as
$$\bigvee_{p \in S} \bigvee_{q \in S} \bigvee_{\sigma \in \Sigma} \cdots$$

6. The control flow of the algorithm is easily expressible in the logic framework. For the sequential executions conjunctions, for "for all" loops universal quantifications, and for the conditional statements implications and negations can be used.

References

[1] I. Dolinka. A note on identities of two-dimensional languages. *Disc. Appl. Math.*, 146(2005), 43–50.

[2] A. Ehrenfeucht and G. Rozenberg. Theory of 2-structures, Part 1: clans, basic subclasses, and morphisms. Part 2: representation through labeled tree families. *Theoret. Comput. Sci.*, 70(1990), 277–303, 305–342.

[3] A. Ehrenfeucht and G. Rozenberg. Angular 2-structures. *Theoret. Comput. Sci.*, 92(1992), 227–248.

[4] A. Ehrenfeucht and G. Rozenberg. T-structures, T-functions and texts. *Theoret. Comput. Sci.*, 116(1993), 227–290.

[5] Z. Ésik. Free algebras for generalized automata and language theory. *RIMS Kokyuroku 1166*, Kyoto University, Kyoto, 2000, 52–58.

[6] Z. Ésik and Z. L. Németh. Higher dimensional automata, *J. of Autom. Lang. Comb.*, 9(2004), 3–29.

[7] Z. Ésik and Z. L. Németh. Algebraic and graph-theoretic properties of infinite n-poests. *Theoret. Informatics Appl.*, 39(2005), 305–322.

[8] D. Giammarresi and A. Restivo. Two-dimensional languages. In: *Handbook of Formal Languages*, Vol. 3, Springer-Verlag, Berlin, 1997, 215–267.

[9] J. Grabowski. On partial languages. *Fund. Inform.*, 4(1981), 427–498.

[10] K. Hashiguchi, S. Ichihara and S. Jimbo. Formal languages over free binoids. *J. Autom. Lang. Comb.*, 5(2000), 219–234.

[11] K. Hashiguchi, Y. Sakakibara and S. Jimbo. Equivalence of regular binoid expressions and regular expressions denoting binoid languages over free binoids. *Theoret. Comput. Sci.*, 312(2004), 251–266.

[12] H. J. Hoogeboom and P. ten Pas. Text languages in an algebraic framework. *Fund. Inform.*, 25(1996), 353–380.

[13] H. J. Hoogeboom and P. ten Pas. Monadic second-order definable text languages. *Theory Comput. Syst.*, 30(1997), 335–354.

[14] D. Kuske. Towards a language theory for infinite N-free pomsets. *Theoret. Comput. Sci.*, 299(2003), 347–386.

[15] K. Lodaya and P. Weil. Kleene iteration for parallelism. In: proc. *FST & TCS'98*, LNCS 1530, Springer-Verlag, 1998, 355–366.

[16] K. Lodaya and P. Weil. Series-parallel languages and the bounded-width property. *Theoret. Comput. Sci.*, 237(2000), 347–380.

[17] K. Lodaya and P. Weil. Rationality in algebras with series operation. *Inform. and Comput.*, 171(2001), 269–293.

[18] Z. L. Németh. A Hierachy Theorem for Regular Languages over Free Bisemigroups. *Acta Cybern.*, 16(2004), 567–577.

[19] Z. L. Németh. Automata on Infinite Biposets. In: *Automata and Formal Languages, 11th Int. Conf., AFL 2005, Dobogókő, Hungary, May 17–20, Proceedings*, Inst. of Informatics, Univ. of Szeged, Szeged, 2005, 213–226.

[20] D. Perrin and J.-E. Pin. *Infinite Words.* Pure and Applied Mathematics, Vol. 141, Elsevier, 2004.

[21] H. Straubing. *Automata, Formal Logic and Circuit Complexity.* Birkhuser, Boston, 1994.

[22] J. Valdes, R. E. Tarjan and E. L. Lawler. The recognition of series-parallel digraphs. *SIAM J. Comput.*, 11(1982), 298–313.

[23] P. Weil. Algebraic recognizability of languages. In: proc. *MFCS 2004*, LNCS 3153, Springer-Verlag, 2004, 149–175.

[24] Th. Wilke. An algebraic theory for regular languages of finite and infinite words. *Internat. J. Algebra Comput.*, 3(1993), 447–489.