Block products for algebras over countable words and applications to logic

Bharat Adsul  
IIT Bombay, India  
Email: adsl@cse.iitb.ac.in

Saptarshi Sarkar  
IIT Bombay, India  
Email: sapta@cse.iitb.ac.in

A. V. Sreejith  
IIT Goa, India  
Email: sreejithav@iitgoa.ac.in

Abstract—We propose a seamless integration of the block product operation to the recently developed algebraic framework for regular languages of countable words. A simple but subtle accompanying block product principle has been established. Building on this, we generalize the well-known algebraic characterizations of first-order logic (resp. first-order logic with two variables) in terms of strongly (resp. weakly) iterated block products. We use this to arrive at a complete analogue of Schützenberger-McNaughton-Papert theorem for countable words. We also explicate the role of block products for linear temporal logic by formulating a novel algebraic characterization of a natural fragment.

I. INTRODUCTION

The seminal works of Büchi and Elgot (see the survey article [23]) established fundamental connections between languages, automata and logics. This was achieved via effective back-and-forth translations between finite-state automata and monadic second-order (MSO) logic over finite words. These connections were enriched with the introduction of algebraic structures such as the syntactic monoids/semigroups (see [12], [13]). The celebrated theorems of Schützenberger [17] and McNaughton-Papert [11] showed the equivalence between star-free regular languages, first-order (FO) logic and recognizability by aperiodic monoids. Many such equivalences have been established for ‘small’ fragments of FO. See [7] for a survey.1

One of the central tools in the algebraic theory of monoids is the notion of a block product and its related cousin - a wreath product. The classical Krohn-Rhodes decomposition theorem (see [18]) asserts that every finite monoid divides (can be simulated by) block/wreath products of ‘simple’ monoids. As a special case, every finite aperiodic monoid divides block products of a simple canonical two-element aperiodic monoid called U1. In other words, the simple monoid U1 serves as a basis for generating all aperiodic monoids under the block product operation!

Linear temporal logics (LTL) [8] provide a very convenient formalism for specifying properties of finite (as well as infinite) words. They typically admit decision procedures of low complexity (unlike FO/MSO) and have high expressive power (LTL has the same expressive power as FO over finite and infinite words by Kamp’s theorem). These properties have contributed to the wide-spread use of temporal logics in formal verification. Underlying these applications are efficient translations from temporal logic specifications to automata. These translations use more intricate automata models developed in the famous works of Büchi and McNaughton which extended the automata-logic connections to infinite words.

The algebraic approach has also been very useful in understanding the expressive power of LTL operators such as since and until. For instance, [22] provides an effective algebraic characterization (in terms of block products) of the until-since hierarchy. See the survey [20] for many such results.

Going beyond finite and infinite words, Rabin [14] already showed that decidability of MSO over countable words (labelled linear orders with countable domains) can be obtained using his deep result on the decidability of MSO over infinite trees. However, an appropriate automaton model naturally working over countable words was missing. There have been some earlier works extending the automata-logic connection to labelled linear orderings beyond finite and infinite words. More recently, [4], [15], [2] showed such connections for countable scattered words that is, labelled countable linear orders without a dense subset. These works introduced appropriate automata, rational expressions and algebras and showed that they all equal MSO in expressive power when restricted to countable scattered words.

In a recent breakthrough, [5] established an algebraic approach to MSO-definability over countable linear orderings. It also showed very impressive applications to logic. One such is the first known (optimal) collapse of the (set)-quantifier alternation hierarchy of MSO to its second level. This study restores the beautiful automata-logic connections from the well-understood settings to the setting of countable words, albeit in the form of an algebra-logic connection. Further, it opens up the possibility of characterizing various sublogics (and their natural fragments) via algebraic means. An elaborate study over a variety of sublogics of MSO over countable words was carried out in [6] where FO, weak MSO etc. were characterized algebraically. It is important to note that all these characterizations are decidable. In a similar vein, a decidable algebraic characterization of the two variable fragment of FO was obtained in [10].

Our investigations in this work are directed towards enriching the algebraic framework of [5]. Motivated by the decisive role played by block products in the standard settings, we introduce block products in the countable setting. The block product construction in the standard setting associates to a pair of monoids \((M, N)\) a new monoid \(M □ N\). From a formal-language theoretic viewpoint, the importance of the block

1
product construction is best described by the accompanying block product principle. Roughly speaking the block product principle (see [18], [13]) says that evaluating a finite word $u$ in $M \Box N$ can be achieved by the following two-stage process: 1) evaluate the word $u$ in $M$ and label every position $x$ of $u$ with the additional information about evaluations of $u|_{<x}$ and $u|_{>x}$ in $M$ where $u|_{<x}$ and $u|_{>x}$ are such that $u = u|_{<x}u[x]u|_{>x}$ (that is, $u|_{<x}$ and $u|_{>x}$ are the left and right factors/contexts at position $x$); 2) now evaluate this extended word (with the additional information) in $N$. Said differently, $M$ ‘operates’ on $u$ as usual; while when $N$ ‘operates’ on $u$, it has access, at every position, to evaluations of $M$ on left-right contexts at that position.

Our generalization of the block product operation and the accompanying block product principle extend this intuitively appealing ‘operational’ description from finite words to countable words. More specifically, we work with the central algebraic objects in [5] such as finite $\oplus$-monoids and $\oplus$-algebras. A $\oplus$-monoid allows evaluations of countable words. So it seems natural to define the block product operation for $\oplus$-monoids. However, a priori it is not clear if this construction is effective. We overcome this difficulty by first defining the block product operation for $\oplus$-algebras which is effective. We then use the non-trivial technical machinery developed in [5] to lift it to $\oplus$-monoids.

Other sources of difficulties when working with countable words are dense subsets and the presence of gaps in the underlying linear ordering. A gap in a countable word $u$ is a factorization of $u$ as $u_1u_2$ such that the word $u_1$ (resp. $u_2$) has no last (resp. first) position. The above mentioned (point-based) operational description of the block product principle is oblivious to left-right contexts at gaps. In spite of this, we provide first algebraic characterizations of some natural logics in terms of the newly defined block products.

Our first application is in the context of first-order logic. We provide generalizations of the well-known algebraic characterizations of FO (resp. FO$^2$, first-order logic with two variables) in terms of strongly (resp. weakly) iterated block products of a natural two-element $\oplus$-algebra. We crucially use this result to show the equivalence between languages definable in FO and those which admit marked star-free regular expressions. This answers an open question posed in [3] which showed a similar result in the setting of countable scattered words. Next, we combine our result with an equationsalgebraic characterization of FO from [6], to give a complete analogue of Schützenberger-McNaughton-Papert theorem for countable words. An interesting consequence of these results is a suitable version of the aforementioned special case of the Krohn-Rhodes theorem.

Our next application concerns LTL. It is known [9] that over countable words, Kamp’s theorem does not hold! That is, LTL[$S,U$] (with natural strict since-until temporal operators) is not as expressive as FO. We provide a very natural characterization of LTL[$S,U$] in terms of weakly iterated block products of two canonical $\oplus$-algebras. This result is delicate as the weak iteration of the standard block product (of monoids) of the underlying monoids of these two $\oplus$-algebras is as expressive as FO over finite words! This supports our hypothesis that the block product construction proposed in this work will allow a fine-grained distinction of several natural fragments of LTL.

Structure of the paper: In the next section we define countable linear orderings and countable words, followed by the section on $\oplus$-monoids and the block product operation on them. We end the section with the block product principle. Section IV talks about the application of block product principle on first order logic and its two variable sub-class. Finally, in section V we characterize an interesting subclass of linear temporal logic.

II. COUNTABLE WORDS

A linear ordering $(Z, <)$ is a set $Z$ equipped with a total order $\cdot$. For example, the reals $(\mathbb{R}, <)$, integers $(\mathbb{I}, <)$, the natural numbers $(\mathbb{N}, <)$ are linear orderings. We say $(X, <)$ is a sub-ordering of a linear ordering $(Z, <)$ if $X \subseteq Z$ and the ordering in $Z$ is preserved in $X$. For $X, Y \subseteq Z$ we write $X < Y$ if $x < y$ for each $x$ in $X$ and $y$ in $Y$. Also if $X < Y$, $Y < Z$ and $Y$ is nonempty, then $X < Z$. A cut of the linear ordering $\alpha$ is a pair $(Z_1, Z_2)$ such that $Z = Z_1 \cup Z_2$ and $Z_1 < Z_2$. We also use the following notation $Z_1; Z_2$ to denote the ordering $Z = Z_1 \cup Z_2$ where $(Z_1, Z_2)$ is a cut in $Z$. A set $L$ is a prefix of $X$ if $X = L \cup K$ and $L < K$ for some $K \subseteq X$. Similarly if $X = L \cup K$ and $L < K$, then $K$ is a suffix of $X$. The set $X$ is dense if between any two elements in the set there is another element. Reals are a dense linear ordering. Set $X$ is scattered if it has no dense subsets. An ordering is a countable linear ordering if the set $Z$ is countable. For example, the reals are not a countable linear ordering. On the other hand the rationals $(\mathbb{Q}, \cdot)$ form a countable linear ordering. Apart from the rationals, the integers, naturals, set of negative numbers $(\mathbb{N}^-, <)$ are all countable linear orderings. Let us see a few more examples. Consider the ordering of two sets of natural numbers one followed by the other, $(\mathbb{N}; \mathbb{N}, <)$. In this linear ordering, there are two positions which do not have a predecessor. Consider another ordering of natural numbers followed by negative numbers $(\mathbb{N}; \mathbb{N}^-, <)$. Here all positions except for the first position has a predecessor. Finally let us see an example which has a dense subordering. Consider the integers followed by rationals followed by integers $(\mathbb{I}; \mathbb{Q}; \mathbb{I}, <)$. We can thus build more and more complex countable linear orderings. A set $X$ is right-open (resp. left-open) if it has no maximum element (resp. minimum element). Nonempty suffixes of right-open sets are right-open and nonempty prefixes of left-open sets are left-open. It is known that all countable linear orderings are isomorphic to some sub-ordering of rational numbers. See [16] for an indepth study of linear orderings. For $x, y$ in a linear ordering $Z$, we denote by $(x, y)$ the set of all points greater than $x$ and less than or equal to $y$.

A subset $X$ of $(Z)$ is convex if for all $x, y \in X$ and $z \in Z$, $x < z < y$ implies that $z \in X$. A condensation $\sim$ is an equivalence relation on $Z$ whose equivalences classes are convex. The order on $Z$ naturally induces a corresponding
order on the quotient $Z/\sim$. We refer to this ordering as the condensed ordering.

For a finite alphabet $\Sigma$ and a linear ordering $\alpha = (Z, <)$, we define a word $w: \alpha \to \Sigma$ to be a mapping from the set $Z$ to $\Sigma$. We call $\alpha$ the domain of $w$, $\text{dom}(w)$. For a word $w$, we say a point/position $x$ to denote an element $x \in \text{dom}(w)$. The notation $w[x]$ denote the letter at the $x^{th}$ position in $w$. A word has a minimal (respectively maximal) element if its domain has a minimal (maximal) element. The word $u$ is a suffix (prefix) of $w$ if $\text{dom}(u) \subseteq \text{dom}(w)$. If $u$ and $v$ are words, then $uv$ denotes the unique word $w$ where $(\text{dom}(u); \text{dom}(v))$ is equal to $\text{dom}(w)$. This operation is naturally extended to a set of words $\{w_i\}_\alpha$ indexed by a linear ordering $\alpha$ as $\prod_{i \in \alpha} w_i$ (see [6] for more details). For a set $X \subseteq \text{dom}(w)$, we denote the restriction of $w$ to the positions in $X$ by $w|_X$.

A word is a countable word if the domain of the word is countable. In this paper, we will be concerned with countable words only. Therefore, from henceforth a word means a countable word. The following countable words are of special interest. $\epsilon$ stands for the empty word (the word over an empty domain). The omega word, $a^\omega$ denotes the word over the domain $(\mathbb{N}, <)$ such that every position is mapped to the letter $a$. Similarly, the omega word $a^{-\omega}$ denotes the word over the domain $(\mathbb{N}^-, <)$ where every position is mapped to letter $a$. The word $a^\omega a^{-\omega}$ is the word over the domain $(\mathbb{N}; \mathbb{N}, <)$ where all positions are mapped to the letter $a$. A perfect shuffle over a nonempty set $S \subseteq \Sigma$ of letters, denoted by $S^0$, is the word over domain $(\mathbb{Q}, <)$ such that any nonempty open interval contains each of the letters in $S$. This is a unique word (up to isomorphism) (see [5]) and is an example of a dense word, i.e. a word whose domain is dense. The word $v \in \Sigma^\oplus$ is a factor of word $w$ if $w = u_0 v u_1$ where $u_0, u_1 \in \Sigma^\oplus$.

For an alphabet $\Sigma$, the set of all countable words is denoted by $\Sigma^\oplus$ and the set of all countable words over non-empty domain is denoted by $\Sigma^\oplus$. A language over the alphabet $\Sigma$ is a subset of $\Sigma^\oplus$.

III. BLOCK PRODUCT

A. Countable Products, Algebras and Evaluation Trees

We recall the algebraic framework from [5] along with some technical definitions/notions which are useful for this work. Most of these are lifted almost verbatim from there.

A $\oplus$-semigroup $(S, \pi)$ consists of a set $S$ with an operation $\pi: S^{\oplus} \to S$ such that, $\pi(a) = a$ for all $a \in S$ and $\pi$ satisfies the generalized associativity property – that is $\pi(\prod_{i \in \alpha} u_i) = \pi(\prod_{i \in \alpha} \pi(u_i))$ for every countable linear ordering $\alpha$. If the generalized associativity holds with $\pi: S^{\oplus} \to S$, then $(S, \pi)$ is called a $\oplus$-monoid.

For a set $\Sigma$, $(\Sigma^\oplus, \Pi)$ (resp. $(\Sigma^\oplus, \Pi)$) is the free $\oplus$-semigroup (resp. free $\oplus$-monoid) generated by $\Sigma$.

A neutral element of a $\oplus$-semigroup $(S, \pi)$ is an element $1 \in S$ such that for every $w \in S^{\oplus}$, $\pi(w) = \pi(w|_{\neq 1})$ if $w|_{\neq 1}$ is non-empty. Here $w|_{\neq 1}$ is $w$ restricted to positions $i$ where $w[i] \neq 1$. If a neutral element exists, it is unique. A $\oplus$-monoid $(S, \pi)$ admits $\pi(\epsilon)$ as the unique neutral element.

Clearly, a $\oplus$-monoid can be naturally viewed as a $\oplus$-semigroup. Further, a $\oplus$-semigroup $S$ can be easily extended to a $\oplus$-monoid (denoted $S^1$) by introducing an additional neutral element if necessary. Thanks to this, any result for $\oplus$-semigroups has a suitable analogue for $\oplus$-monoids. In view of this, most of the technical results in this work are simply stated and proved for $\oplus$-semigroups.

Example 1. $U_1 = (\{1, 0\}, \pi)$ is a $\oplus$-monoid where $\pi$ is defined for all $u \in \{1, 0\}^*$ as:

$$\pi(u) = \begin{cases} 1 & \text{if } u \in \{1\}^* \\ 0 & \text{otherwise} \end{cases}$$

Let $(S, \pi)$ be a $\oplus$-semigroup. Even if $S$ is finite, $\pi$ need not be finitely presentable and hence not suitable for algorithmic purposes. Fortunately, it is possible to capture $\pi$ through finitely presentable operators. This is precisely the reason for introducing $\oplus$-algebras.

A $\oplus$-algebra $(S, \cdot, \tau, \pi, \kappa)$ consists of a set $S$ with $\cdot: S^2 \to S$, $\tau: S \to S, \tau^*: S \to S, \kappa: \mathcal{P}(S) \setminus \{\emptyset\} \to S$ such that (with infix notation for $\cdot$ and superscript notation for $\tau, \tau^*$, $\kappa$)

- A-1 $(S, \cdot)$ is a semigroup.
- A-2 $(a \cdot b)^\tau = a \cdot (b \cdot a)^\tau$ and $(a^n)^\tau = a^\tau \cdot a$ for $a, b \in S$ and $n > 0$.
- A-3 $(b \cdot a)^\tau = (a \cdot b)^\tau \cdot a$ and $(a^n)^\tau = a^\tau \cdot a$ for $a, b \in S$ and $n > 0$.
- A-4 For every non-empty subset $P$ of $S$, every element $c \in P$, every subset $P'$ of $P$, and every non-empty subset $P''$ of $\{P^*, a, P^\oplus, b | a, b \in P\}$, we have $P^\oplus = P^\oplus \cdot a = P^\oplus \cdot b = (P^\oplus)^\tau = (P^\oplus)^\tau = (c, P^\oplus)^\tau = (P' \cup P'' \oplus \kappa}$.

A $\oplus$-algebra is a $\oplus$-algebra where $(S, \cdot, 1)$ is a monoid, $1^\tau = \{1\}^\kappa = 1$ and for all non-empty subsets $P \subseteq \mathcal{P}(S)$, $P^\oplus = (P \cup \{1\})^\kappa$.

It is shown in [5] that a $\oplus$-semigroup $(\oplus$-monoid) naturally induces a $\oplus$-algebra $(\oplus$-algebra respectively). We simply set $a \cdot b = \pi(ab)$, $a^\tau = \pi(a^\omega)$, $a^\tau = \pi(a^\omega)$ and $P^\oplus = \pi(P^\oplus)$ (recall that $P^\oplus$ is the unique perfect shuffle word over $P$).

Example 2. The $\oplus$-algebra induced by $U_1$ is given below:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>\tau</th>
<th>\tau^*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 1. The $\oplus$-algebra $U_1$. We also have $0^\kappa = \{1, 0\}^\kappa = 0$ and $1^\kappa = 1$.

In [5], it is also shown that a finite $\oplus$-algebra has a unique extension to a finite $\oplus$-semigroup. In short, the algebra operators allow to ‘evaluate’ $\pi$ on ‘simple’ countable words. In order to extend this evaluation to arbitrary countable words, one needs the important notion of an evaluation tree which in turn uses the notion of a condensation tree. We first recall these notions and note that they will be used crucially later in the proof of a key technical lemma.

A condensation tree over a linear ordering $\alpha$ is a set $T$ of non-empty convex subsets of $\alpha$ such that
• \( \alpha \in T \),
• for all \( I, J \in T \), either \( I \subseteq J \) or \( J \subseteq I \) or \( I \cap J = \emptyset \).
• for all \( i \in T \), the union of all \( J \in T \) such that \( J \subseteq i \) is either \( i \) or \( \emptyset \).
• every subset of \( T \) totally ordered by inclusion is finite.

Elements of \( T \) are called nodes. The node \( \alpha \) is called the root of the tree, nodes minimal for \( \subseteq \) are called leaves and non-leaf nodes (including the root) are called internal nodes. A node \( I \) is a child of a node \( J \) if \( I \subseteq J \) and, for all \( K \in T \), \( I \subseteq K \) implies \( J \subseteq K \). If \( I \) is an internal node, then it has a set of children that forms a partition of \( I \) into convex subsets. We denote this partition by \( \text{children}_T(I) \) which naturally corresponds to a condensation of \( \alpha \mid J \). By a slight abuse of notation, \( \text{children}_T(I) \) denotes this condensation as well as the corresponding condensed ordering. Finally, for a node \( I \), the subtree of \( T \) rooted at \( I \) is simply the condensation tree obtained by restricting \( T \) to \( T \) with \( J \subseteq I \).

The following lemma from [5] plays an important part later.

**Lemma 1.** It is possible to associate with each condensation tree a countable ordinal \( \text{rank}(T) \) in such a way that \( \text{rank}(T') < \text{rank}(T) \) for all subtrees \( T' \) rooted at non-root nodes of \( T \).

Now we are ready to introduce the important notion of an evaluation tree. Towards this, fix a finite \( \oplus \)-algebra \((S, \cdot, \tau, \tau^*, \kappa)\).

Let \( \pi_0 \) be the partial function from \( S^\oplus \) to \( S \) such that
• \( \pi_0(ab) = a \cdot b \) for all \( a, b \in S \),
• \( \pi_0(e^{\omega}) = e^\tau \) for all idempotents \( e \in S \),
• \( \pi_0(e^{\omega^*}) = e^{\tau^*} \) for all idempotents \( e \in S \),
• \( \pi_0(P^n) = P^\kappa \) for all non-empty sets \( P \subseteq S \),
• in all remaining cases \( \pi_0 \) is undefined.

An evaluation tree over a word \( u \in S^\oplus \) is a pair \( T = (T, \gamma) \) where \( T \) is a condensation tree over the domain of \( u \) and \( \gamma \) is a function from \( T \) to \( S \) such that:
• every leaf of \( T \) is a singleton of the form \( \{ x \} \) (with \( x \in \alpha \)) and \( \gamma(x) = u[x] \),
• for every internal node \( I \) of \( T \), the partial function \( \pi_0 \) is defined on the word \( \gamma(\text{children}_T(I)) \) that has domain \( \text{children}_T(I) \) and labels each position \( J \in \text{children}_T(I) \) with \( \gamma(J) \); in addition, we have \( \gamma(I) = \pi_0(\gamma(\text{children}_T(I))) \).

The value of \( T \) is defined to be \( \gamma(\alpha) \).

Now we are ready to state the fundamental result of [5].

**Theorem 1.** For every word \( u \in S^\oplus \), there exists an evaluation tree \( T = (T, \gamma) \) over \( u \). Evaluation trees over the same word have the same value. The (well-defined) function \( \pi : S^\oplus \to S \) defined as: for a word \( u, \pi(u) \) is the value of some evaluation tree over \( u \), satisfies the generalized associativity property and provides a unique extension of the \( \oplus \)-algebra structure \((S, \cdot, \tau, \tau^*, \kappa)\) to a \( \oplus \)-semigroup structure \((S, \pi)\). Further, for an evaluation tree \( T = (T, \gamma) \) over a word \( u \) and for every node \( I \in T \), \( \gamma(I) = \pi(u|_I) \).

Now we briefly discuss some natural algebraic notions. Let \((S, \pi)\) and \((S', \pi')\) be \( \oplus \)-semigroups. A morphism from \((S, \pi)\) to \((S', \pi')\) is a map \( h : S \to S' \) such that, for every \( w \in S^\oplus \), \( h(\pi(w)) = \pi'(h(w)) \) where \( h \) is the pointwise extension of \( h \) to words. A \( \oplus \)-subsemigroup of \((S, \pi)\) is a subset \( S \subseteq S \) such that \( \pi \) restricts from \( S^\oplus \) to \( S \). We simply denote by \((S, \pi) \subseteq (S, \pi)\). We say \((S, \pi)\) divides \((S', \pi')\) if there exists \((S'', \pi'') \subseteq (S', \pi')\) and a surjective morphism from \((S'', \pi'')\) to \((S, \pi)\). The notions of morphism, subsemigroup (subalgebra), division are also defined for algebras along expected lines. Further, they are naturally compatible with those of \( \oplus \)-semigroups.

**B. Semidirect Product Construction**

The direct product is an important standard construction in algebra. It is straightforward to adapt this notion to \( \oplus \)-semigroups and \( \oplus \)-algebras. In this section, we propose a generalization of semidirect product from semigroups (see [18], [13]) to \( \oplus \)-semigroups. Keeping in mind the 'finite presentation issue', we first define this construction for \( \oplus \)-algebras.

We begin by introducing the setup of two commuting actions of a \( \oplus \)-algebra on another.

Consider two \( \oplus \)-algebra \((M, \cdot, \tau, \tau^*, \kappa)\) and \((N, \cdot, \hat{\tau}, \hat{\tau}^*, \hat{\kappa})\). A function \( \delta_1 : M^1 \times N \to N \) is said to be a left action of \( M \) on \( N \) if it satisfies the following conditions. \( \delta_1(m, n) \) is denoted by \( m \cdot n \) for convenience.

- **C-1** \( 1 \cdot n = n \)
- **C-2** \((m_1 \cdot m_2) \cdot n = m_1 \cdot (m_2 \cdot n) \)
- **C-3** \( m \cdot (n_1 \cdot n_2) = (m \cdot n_1) \cdot n_2 \)
- **C-4** \( m \cdot n^{\tau} = (m \cdot n)^{\tau} \)
- **C-5** \( m \cdot n^{\tau^*} = (m \cdot n)^{\tau^*} \)
- **C-6** \( m \cdot \{n_1, \ldots, n_j \}^{\kappa} = \{m \cdot n_1, \ldots, m \cdot n_j \}^{\kappa} \)

Similarly, a function \( \delta_2 : N^1 \times M \to N \) is said to be a right action of \( M \) on \( N \) if it satisfies the following conditions. \( \delta_2(n, m) \) is denoted by \( n \cdot m \) for convenience.

- **C-7** \( n \cdot 1 = n \)
- **C-8** \( n \cdot (m_1 \cdot m_2) = (n \cdot m_1) \cdot m_2 \)
- **C-9** \( (n_1 \cdot n_2) \cdot m = n_1 \cdot (n_2 \cdot m) \)
- **C-10** \( n^{\tau} \cdot m = (n \cdot m)^{\tau} \)
- **C-11** \( n^{\tau^*} \cdot m = (n \cdot m)^{\tau^*} \)
- **C-12** \( \{n_1, \ldots, n_j \}^{\kappa} \cdot m = \{n_1 \cdot m, \ldots, n_j \cdot m \}^{\kappa} \)

\( \delta_1 \) and \( \delta_2 \) are compatible with each other if they satisfy the following condition.

- **C-13** \((m_1 \cdot n) \cdot m_2 = m_1 \cdot (n \cdot m_2) \)

For \( m \in M \), define as \( \delta_1^{m} : N \to N \) as \( \delta_1^m(n) = m \cdot n \). By abuse of notation, the natural pointwise extension of \( \delta_1^m \) from \( N^\oplus \) to itself will also be denoted by \( \delta_1^m \). The above conditions for the left action may be stated simply by saying that, for each \( m \in M \), \( \delta_1^m \) is a morphism of \( \oplus \)-algebras (equivalently, \( \oplus \)-semigroups) and for \( m_1, m_2 \in M \), the morphism \( \delta_1^{m_1 \cdot m_2} \) is the composition of \( \delta_1^{m_1} \) and \( \delta_1^{m_2} \). A similar remark applies to the right action and these two actions commute.

Suppose \( M \) and \( N \) are \( \oplus \)-algebras with neutral elements 1 and 1. We say that \( \delta_1 \) is monoidal if \( 1 \cdot n = n \) \( \forall n \in N \) and \( m \cdot 1 = 1 \) \( \forall m \in M \). Right action is monoidal for symmetric conditions.
Definition 1. We define the semidirect product of the two $\oplus$-algebras as $M \ltimes N = (M \times N, \cdot, \cdot, \cdot, \kappa)$ where

1) $(m_1, n_1) \cdot (m_2, n_2) = (m_1 \cdot m_2, n_1 \cdot n_2)$
2) $(m, n) \cdot (m', n') = (m \cdot m', n \cdot n')$
3) $(m, n) \cdot (m', n') = (m \cdot m', n \cdot n')$
4) $(m, n) \cdot (m', n') = (m \cdot m', n \cdot n')$

In the above definition if $e$ is an idempotent of $M$, then the equations for $\tau$ and $\pi$ simplify to the following:

- $(e, n)^\tau = (e^n, n^\pi)$
- $(e, n)^\tau = (e^n, n^\pi)$

The following lemma verifies that $M \ltimes N$ is a $\oplus$-algebra by checking the axioms A-1 to A-4. Please refer to [1] for the tedious calculations regarding this.

Lemma 2. The structure $(M \ltimes N, \cdot, \cdot, \cdot, \kappa)$ is a $\oplus$-algebra

We remark here that if $M$ and $N$ are $\oplus$-algebras (with neutral elements 1 and 1 respectively) and the actions are monoidal, then $M \ltimes N$ is a $\oplus$-algebra with $(1, 1)$ as the neutral element.

Now we state the main result of this subsection which is an immediate consequence of the previous lemmas and theorems.

Proposition 1. The $\oplus$-algebra $(M \ltimes N, \cdot, \cdot, \cdot, \kappa)$ admits a unique extension, denoted by $(M \ltimes N, \hat{\pi})$, as a $\oplus$-semigroup. Further, in the monoidal setting, $(M \ltimes N, \cdot, \cdot, \cdot, \kappa)$ is a $\oplus$-algebra and $(M \ltimes N, \hat{\pi})$ is a $\oplus$-monoid.

We now present an example of a semidirect product construction and highlight some important elements in the $\oplus$-algebra. Recall that an element $e$ of $M$ is called an idempotent if $e \cdot e = e$. We call it an $\omega$-idempotent (resp. $\omega^\kappa$-idempotent and $\eta$-idempotent) if $e^\omega = e$ (resp. $e^\omega = e$ and $e^\kappa = e$).

An element $r$ of $M$ is called a right zero if $m \cdot r = r$ for every element $m$ of $M$. The notion of a left zero is defined similarly. An element $z$ is called a zero, if it is both a left zero and a right zero. If a zero exists, it is unique.

Example 3. Consider $M = U_1$ acting on $N = U_1$ with a trivial left action and a non-trivial monoidal right action where everything in $N$ maps to 1. The $\oplus$-algebra generated $M^r = M \ltimes N = U_1 \ltimes U_1$ is given below. Write the element $(i, j)$ as $ij$ in this example.

<table>
<thead>
<tr>
<th></th>
<th>11</th>
<th>10</th>
<th>00</th>
<th>01</th>
<th>\tau</th>
<th>\tau^*</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>11</td>
<td>10</td>
<td>00</td>
<td>01</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>00</td>
<td>00</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
</tbody>
</table>

Fig. 2. The monoid $M^r$: Note that all elements are $\omega^\tau$ idempotents, and the elements 00, 01 are right zero. We have $(11)^{\kappa} = 10^{\kappa} = (11,10)^{\kappa} = 10$ and $X^\kappa = 01$ if $X$ contains any right zero.

C. A Key Lemma

Let $(M, \pi)$ and $(N, \hat{\pi})$ be two finite $\oplus$-semigroups with induced $\oplus$-algebras $(M, \cdot, \cdot, \cdot, \kappa)$ and $(N, \cdot, \cdot, \cdot, \hat{\kappa})$ respectively.

As seen before, given a compatible pair of left and right actions of $M$ on $N$, we can construct the semidirect product $\oplus$-algebra $(M \ltimes N, \hat{\cdot}, \hat{\cdot}, \hat{\cdot}, \hat{\kappa})$ which is induced by a unique $\oplus$-semigroup $(M \ltimes N, \hat{\pi})$.

In this section, we relate the evaluation $\hat{\pi}$ of $M \ltimes N$ to the evaluations $\pi$ and $\hat{\pi}$ of $M$ and $N$ respectively. This will be later used to derive appropriate semidirect-product/block-product principles.

Towards this, consider a word $u \in (M \ltimes N)^\oplus$ with domain $\alpha$ being the underlying linear ordering. To the word $u$, we associate two words $v \in M^\oplus$ and $\omega \in N^\oplus$ with domain $\omega$. We begin by defining $u_1 \in M^\oplus$ and $u_2 \in N^\oplus$ (with domain $\alpha$) as simply the ‘projections’ of $u$ on $M$ and $N$ respectively. In other words, for $x \in \alpha$, $u_1[x] = m$ and $u_2[x] = n$ if $u[x] = (m, n)$. The word $v$ is simply $u_1$. On the other hand, the word $w$ is derived from $u_2$ by making use of the evaluations of $\pi$ of $M$ on appropriate ‘point-based-factors’ of $u_1$ and their actions on $N$. Precisely, for $x \in \alpha$, $u_2[x] = \pi(u_1[x]) \ast u_2[x] \ast \pi(u_1[x])$.

Definition 2. The map $\theta : (M \ltimes N)^\oplus \to M^\oplus \times N^\oplus$ is defined as: $\theta(u) = (v, w)$.

Lemma 3. Suppose $u = \Pi_{i \in \beta}u_i$ with $\theta(u) = (v, w)$ and, for $i \in \beta$, $\theta(u_i) = (v_i, w_i)$. Then $v = \Pi_{i \in \beta}v_i$ and $w = \Pi_{i \in \beta}w_i$ where, for $i \in \beta$.

Proof: The proof of this proposition crucially uses the technical machinery of evaluation trees introduced in subsection III-A to deal with arbitrary countable words by a suitable induction.

Let $u$ be a word in $(M \ltimes N)^\oplus$ with domain $\alpha$ and $T = (T, \gamma)$ be an evaluation tree over $u$. If the root $\alpha$ of $T$ is a leaf, then the result is immediate.

Consider the case when the root $\alpha$ is the only internal node of the tree. By property of the evaluation tree, $u$ is one of four special kinds of words over $(M \times N)$ for which $\hat{\pi}_0$ is defined. Using the equations from 1, we can verify the lemma holds for each of these cases. Note that for any $c \in M \ltimes N$, $\theta(c) = c$.

- Case 1: $u = (m_1, n_1)(m_2, n_2)$ is a two letter word. Then $\hat{\pi}(u) = (m_1, n_1) \ast (m_2, n_2)$ which from definition 1 equals $(m_1 \cdot m_2, n_1 \cdot n_2)$.

- Case 2: $u$ is an $\omega$-sequence of idempotent $(m, n)$. Then $\hat{\pi}(u) = \hat{\pi}((m, n)^{\omega}) = (m, n)^\omega$. By lemma 3, $v = \Pi_{i \in \omega}m \ast m^\omega$.

- Case 3: $u$ is a $\omega^\kappa$-sequence of idempotent $(m, n)$. Then $\hat{\pi}(u) = \hat{\pi}((m, n)^{\kappa}) = (m, n)^\kappa$. By lemma 3, $v = \Pi_{i \in \omega}m \ast m^{\kappa}$. So $\pi(v) = v \omega^\kappa$. Using lemma 3 for $w$, $\omega[i] = n \ast \pi((\Pi_{i \in (\omega \cup \omega^\kappa)}m) = n \ast \pi(m^{\omega^\kappa}) = n \ast m^\kappa$ and $w[i] = n \ast n \ast m^\kappa$ for any $i \in \alpha \setminus \{0\}$.
will also be denoted by $M$. Thus the induction step reduces to the situation where the forces $m$ to be an idempotent.

- Case 3: $u$ is a $w^*$-sequence. Similar to case 2.

- Case 4: $u$ is a perfect shuffle of $P = \{ (m_1, k_1), \ldots, (m_k, n_k) \}$. Then $\bar{\pi}(u) = \bar{\pi}(P^n) = P^n$.
This reveals that for each position \( h \) recognized by \( \Pi \) (block product principle) related wreath product principle).

For the sake of simplicity, the corollary below paper. It is also possible to characterize recognition via \( M \)

We proceed further with some simple calculations.

\[
\begin{align*}
\Pi_{m_1,m_2}(w[x]) &= (h_1(u_{<x}) \downarrow f_a \uparrow h_1(u_{>x})) (m_1,m_2) \\
&= f_a(m_1 h_1(u_{<x}), h_1(u_{>x}) m_2) \\
h_2(m_1 u'[x]m_2) &= h_2((m_1 h_1(u_{<x}), a, h_1(u_{>x}) m_2)) \\
&= f_a(m_1 h_1(u_{<x}), h_1(u_{>x}) m_2)
\end{align*}
\]

This reveals that for each position \( x \), \( \Pi_{m_1,m_2}(w[x]) = h_2(m_1 u'[x]m_2) \). Thanks to the fact that both \( \Pi_{m_1,m_2}(w) \) and \( m_1 u'[x]m_2 \) are defined pointwise, we have \( \Pi_{m_1,m_2}(w) = h_2(m_1 u'[x]m_2) \). We let \( f \) denote the evaluation of \( w \) in \( K \) and exploit the fact that both \( \Pi_{m_1,m_2}(w) \) and \( h_2 \) are morphisms to conclude that, for \( m_1,m_2 \in M \), \( f(m_1,m_2) = h_2(m_1 u'[x]m_2) \in N \).

With \( h_1(u) = m \), the proof of the proposition is now immediate by Lemma 4 which asserts that \( h(u) = (m,f) \).

An important corollary of the above proposition is recorded below. In most applications of the block products, a morphism into \( \mathbb{M} \times \mathbb{N} \) uses 'only' a subset of \( M \times N \) for recognition (unlike an arbitrary subset of the underlying set \( M \times K \)). This 'special' case is all that we need for the purpose of this paper. It is also possible to characterize recognition via arbitrary subsets similarly. For the sake of simplicity, the corollary below addresses recognition via a subset of \( M \times N \). We will often refer to it as the block product principle (see [13], [18] for the related wreath product principle).

**Corollary 1** (block product principle). Let \( L \subseteq \Sigma^\oplus \) be recognized by \( h : \Sigma \rightarrow \mathbb{M} \times \mathbb{N} \) via a subset \( F \subseteq M \times N \).

Assume that \( M \) is a \( \oplus \)-monoid. Then \( L \) can be expressed as a finite union of languages of the form \( L_1 \cap \sigma^{-1}(L_2) \) where \( L_1 \) and \( L_2 \) are recognized by \( M \) and \( N \) respectively.

**Proof:** For \( (m,n) \in F \), let \( F_{m,n} = \{(m,f) \in M \times K| f(1,1) = n\} \). The above proposition implies that, for \( u \in \Sigma^\oplus \), \( h(u) \in F_{m,n} \) iff \( h_1(u) = m \) and \( h_2(\sigma(u)) = n \). As a consequence, it is easy to check that

\[
L = \bigcup_{(m,n) \in F} h_1^{-1}(m) \cap \sigma^{-1}(h_2^{-1}(n))
\]

This completes the proof.

**IV. FIRST ORDER LOGIC**

**A. Definitions**

First-order logic (FO[<]) over a finite alphabet \( \Sigma \) is a logic which can be inductively built using the following operations.

\[
a(x) \ | \ x < y \ | \ x = y \ | \ \phi \lor \phi \ | \neg \phi \ | \exists x \phi
\]

Here \( a \in \Sigma \) and \( \phi \) is any FO[<] formula. We use the letters \( \phi, \psi, \varphi \) (with or without subscripts) to represent FO[<] formulas, and the letters \( x, y, z \) (with or without subscripts) to represent FO[<] variables.

A variable is free if it is not quantified in the formula. The set of free variables in a formula \( \varphi \) is denoted by free(\( \varphi \)). A formula with no free variables is called a sentence. The language of a sentence \( \varphi \) (denoted by \( L(\varphi) \)) is the set of all words \( u \in \Sigma^\oplus \) that satisfies \( \varphi \).

Let us look at some examples of countable languages definable in FO[<] and its two variable sub-class. Over finite words, FO[<] can talk about occurrence of letters and also about the order in which they appear [24]. Over countable linear orderings, it can also say that there is no maximal point for a letter. For example, the following formula states that every position is labelled by \( a \) and there is no maximum position.

\[
(\forall x \ \exists y > x) \land (\forall x \ a(x))
\]

Analogously, FO[2]<] can also talk about left open words. However, the two variable fragment is not as expressive as full first order. FO[2]<] satisfies a downward property (similar to Löwenheim-Skolem downward theorem for first order logic):
a satisfiable \( \text{FO}^2[<] \) formula has a scattered satisfying model (see [10]). Therefore, the following language, which says the linear ordering is dense, is not definable in \( \text{FO}^2[<] \).

\[
\forall x, y \ (x < y) \Rightarrow (\exists z \ x < z < y)
\]

**Example 4.** Consider the language \( L \subseteq \{a, b\}^\omega \) of all words \( w \) which satisfy the property: There is a gap in \( w \) towards which the letter \( b \) approaches from the left and on the right of the gap there is an interval with only \( a \)'s. This is definable in \( \text{FO}[<] \) by first guessing two points \( x_1 \) and \( x_2 \) on both sides of the gap. Let \( \psi(x_1, x_2, y) \) be a formula which is true if and only if \( y \) is between \( x_1 \) and \( x_2 \) and is after all occurrences of \( b \)'s between \( x_1 \) and \( x_2 \). The three formulas below say that (1) the \( b \)s form an omega sequence, (2) the \( a \)'s after \( b \) form an omega* sequence, and (3) there is factor after the gap which contains only \( a \)'s.

1. \( \phi(x_1, x_2) := \forall y \in (x_1, x_2) \ (b(y) \Rightarrow (\exists z > y \ b(z))) \)
2. \( \psi(x_1, x_2) := \forall y \psi(x_1, y, x_2) \Rightarrow (\exists y < x \ \psi(x_1, x_2, y)) \)
3. \( \varphi(x_1, x_2) := (\forall x < y) (\psi(x_1, x, y) \wedge \psi(x_1, y, x_2)) \Rightarrow (\forall z \in (x_1, x_2) \ \psi(x_1, x, z)) \)

The formula \( \exists x_1, x_2 \ (\varphi \wedge \psi \wedge \varphi) \) defines the language \( L \).

In the following subsections, we provide block product characterizations of \( \oplus \)-algebra recognizing \( \text{FO}[<] \) and \( \text{FO}^*[<] \) languages over linear countable orderings. We also give an equivalent regular expression.

**B. Iterated block product**

Block product of \( \oplus \)-algebra is not associative and so the order of product (equivalently nesting of brackets) varies the resulting structure for a given list of \( \oplus \)-algebra. For example, for three \( \oplus \)-algebras, there are only two distinct nesting possible and the following lemma shows that one of them is at least as powerful as the other.

**Lemma 6.** For any three \( \oplus \) algebra \( M, N, \) and \( P \),

\[
M \square (N \square P) \preceq (M \square N) \square P
\]

For multiple monoids, there can be several ways of bracketing a list of monoids. For any set \( S \) of \( \oplus \)-algebra, it is defined recursively as follows:

1. \( K \square K \) is an iterated block product for any \( K \in S \)
2. If \( M \) is an iterated product, then \( M \square K \) is an iterated product for any \( K \in S \)

For example, in lemma 6 the RHS is an iterated block product over \( \{M, N, P\} \). Basically the iterated block product is where the feedback comes from a complicated structure to a simple structure. The set of all iterated block products of a set \( S \) is denoted by \( \mathcal{B} S \). For a singleton set, we drop the set notation.

The iterated block product is an important way of nesting as the following lemma shows: given \( k \) many monoids \( M_1, M_2, \ldots, M_k \) in a fixed left to right order, the iterated block product is the most powerful way of bracketing them.

**Lemma 7.** The iterated block product is the strongest way of bracketing a list of monoids.

**Theorem 2.** \( L(\text{FO}[<]) = L(\square \text{U}_1) \)

**Proof:** First we show left to right inclusion. Our proof goes via structural induction on first order logic. We know that \( \text{FO}[<] \) has letter and order predicates, is closed under boolean operations and existential quantification. Inductively we prove that for any FO formula \( \varphi = \varphi(x_1, x_2, \ldots, x_n) \), the language \( L(\varphi) \subseteq (\Sigma \times \{0, 1\})^\omega \) is recognized by an iterated block product of \( \text{U}_1 \).

It is easy to show that both the languages definable by \( \varphi = a(x) \) and \( \varphi = x < y \) can be recognized by \( \text{U}_1 \square \text{U}_1 \). Similarly, boolean combinations of first order formulas can also be recognized by cartesian product of the corresponding monoids (from lemma 8 the cartesian product divides block product). The interesting case is when \( \varphi = \exists x \phi \). Let \( L(\phi) \subseteq (\Sigma \times \{0, 1\})^\omega \) be recognized by a monoid \( M \in \square \text{U}_1 \) via the morphism \( h' : (\Sigma \times \{0, 1\})^\omega \to M \). That is, \( \exists F' \subseteq M, h'^{-1}(F') = L(\phi) \). Consider the morphism which extends \( h : \Sigma \to M \square \text{U}_1 \), where \( h(a) = (h'(a, 0), f_a) \) and

\[
f_a(m_1, m_2) = \begin{cases} 0, & \text{if } m_1, h'(a, 1), m_2 \in F' \\ 1, & \text{otherwise} \end{cases}
\]

We claim that \( u \in L(\varphi) \) if and only if \( h_2(\sigma(u)) = 0 \). First we prove the forward direction. Let \( u \in L(\phi) \). Then \( u = u_1 a u_2 \) such that \( u_1(1, 0, 1) u_2 \models \varphi \) (for a word \( v \in \Sigma \), we denote by \( v^0 \) the word over the same domain as \( v \) such that \( v^0[i] = (v[i], 0) \)). Then \( h'(u_1(1, 0, 1) u_2) \in F' \) and hence \( h_2(\sigma(u)) = 0 \). For the other direction, let us assume \( h_2(\sigma(u)) = 0 \). Therefore, there exists \( m_1, m_2 \in M \) such that \( m_1, h'(a, 1), m_2 \in F' \). But this means \( u \) can be factored as \( u_1 a u_2 \) such that \( m_1 = h_1(u_1) \) and \( m_2 = h_1(u_2) \). Therefore \( u_1(1, 0, 1) u_2 \in L(\phi) \) and hence \( u \in L(\varphi) \).

The right to left inclusion is via induction on the number of iterated blocks of \( \text{U}_1 \). It is easy to observe the base case. That is, all languages recognized by exactly one \( \text{U}_1 \) can be defined in first order logic. Let the hypothesis hold for monoid \( M \). We show that, the language \( L \) recognized by the morphism \( h : \Sigma^\omega \to M \square \text{U}_1 \) can be defined in \( \text{FO}[<] \). Let \( \sigma : \Sigma^\omega \to (M \times \Sigma \times M)^\omega \) be a transducer. From the block product principle, \( L \) can be expressed as a finite union of languages of the form \( L_1 \cap \sigma^{-1}(L_2) \) where \( L_1 \) and \( L_2 \) are recognized by \( M \) and \( \text{U}_1 \) respectively. By the induction hypothesis both \( L_1 \) and \( L_2 \) are \( \text{FO}[<] \) definable. So, it suffices to show that for an \( \text{FO}[<] \) language \( L_2 \) over the alphabet \( M \times \Sigma \times M \) the language \( \sigma^{-1}(L_2) \) is also \( \text{FO}[<] \) definable. We prove by structural induction that for every \( \text{FO}[<] \) formula \( \varphi \) over alphabet \( M \times \Sigma \times M \), there exists a \( \text{FO}[<] \) formula \( \psi \) over alphabet \( \Sigma \) such that free(\( \varphi \)) = free(\( \psi \)) and for all words \( w \in \Sigma^\omega \), \( w \models \psi(t_1, \ldots, t_k) \) if and only if \( \sigma(w) \models \varphi(t_1, \ldots, t_k) \). The base case when \( \varphi \) is a letter is the most interesting. Let \( \varphi = (m, a, n) \). By induction hypothesis (and relativisation) there is a formulas for all \( m, n \in M \), \( \varphi_m \) (similarly \( \varphi_n \)) such that \( h_1(w|_{<i}) = m \)
(similarly \( h_1(w_{>|i|}) = n \)) if and only if \( w_{|<i|} \models \varphi^I_m \) (similarly \( w_{|>i|} \models \varphi^I_n \)). We define \( \psi = a(x) \land \varphi^I_m |_{<x} \land \varphi^I_n |_{>x} \). Note that \( \sigma(w) \models \varphi(i) \) if and only if \( w \models \psi(i) \). Inductively applying this translation for other formulas gives the required formula \( \psi \).

C. Weakly iterated block product

The weakly iterated block product is where the feedback always comes from one of the simplest structures. For any set \( S \) of \( \oplus \)-algebra, it is defined recursively as follows:

1) \( K \square K \) is a weakly iterated block product for any \( K \in S \)
2) If \( M \) is an iterated product, then \( K \square M \) is a weakly iterated product for any \( K \in S \)

For example, in lemma 6 the LHS is a weakly iterated product over \( \{ M, N, P \} \). The set of all weakly iterated block products of a set \( S \) is denoted by \( \square^*_S \). For a singleton set, we drop the set notation for convenience.

The following lemma or rather its generalization will be used later to prove theorem 3.

Lemma 8. For any \( \oplus \)-monoids \( M, N, \) and \( P \),

\[
(M \times N) \square P \leq M \square (N \square P)
\]

This can be generalized to say \( (M_1 \times \ldots \times M_k) \square N \) divides the weakly iterated product of the same.

Theorem 3. \( L(\text{FO}^2[\subset]) = L(\square_w U_1) \)

Proof: The right to left inclusion is via induction on the number of blocks of \( U_1 \)'s. First, observe that languages recognized by a single \( U_1 \) can be defined in \( \text{FO}^2[\subset] \). For the induction step, we follow the proof in Theorem 2 closely. Let the hypothesis hold for monoid \( M \in \square^*_U \). We show that, the language \( L \) recognized by the morphism \( h : \Sigma^\oplus \to U_1 \square M \) can be defined in \( \text{FO}^2[\subset] \). Let \( \sigma : \Sigma^\oplus \to (U_1 \times \Sigma \times U_1)^\oplus \) be a transducer. From the block product principle, \( L \) can be expressed as a finite union of languages of the form \( L_1 \cap \sigma^{-1}(L_2) \) where \( L_1 \) and \( L_2 \) are recognized by \( U_1 \) and \( M \) respectively. By the induction hypothesis both \( L_1 \) and \( L_2 \) are \( \text{FO}^2[\subset] \) definable. So, it suffices to show that for an \( \text{FO}^2[\subset] \) language \( L_2 \) over the alphabet \( U_1 \times \Sigma \times U_1 \) the language \( \sigma^{-1}(L_2) \) is also \( \text{FO}^2[\subset] \) definable. As observed in Theorem 2 the base case is the non-trivial case. The following formula accepts \( \sigma^{-1}(L_2) \) if \( L_2 \) is defined by the formula \( (0, a, 1)(x) \).

\[
a(x) \land (\exists y < x \bigvee_{h_1(b)=0} b(y)) \land (\forall y > x \bigvee_{h_1(c)=1} c(y))
\]

Note that we used only two variables for the above translation. The other base cases are similar. We apply this translation inductively for other formulas.

Now we show the other direction of the proof. \( \text{FO}^2[\subset] \) has a “normal form” [19] where the quantifier at the maximum depth along with its scope is of the form \( \exists x(a(x) \land x < y) \) or \( \exists x \left( a(x) \land x > y \right) \). Let us call them base formulas. Note that these formulas have a free variable \( y \). \( \text{FO}^2[\subset] \) sentences can be inductively built by replacing letters \( c(y) \) in a formula by one of the base formulas. Let us assume that languages definable by \( \text{FO}^2[\subset] \) formulas of quantifier depth \( k \) can be recognized by weak block product of \( U_1 \)'s. Consider a formula \( \alpha \) over the alphabet \( \Sigma \) of quantifier depth \( k + 1 \). From our observation, there exists a set \( \Gamma = \{ \gamma_1, \ldots, \gamma_l \} \) and a formula \( \phi \) over \( \mathcal{P}(\Gamma) \) such that replacing every occurrence of \( \gamma_i(y) \) by a base formula \( \psi_i(y) \) over the alphabet \( \Sigma \) gives you the formula \( \alpha \). Here \( \phi \) is a formula of quantifier depth \( k \). From the inductive hypothesis, we know there is a monoid \( M \) and a morphism \( h : \mathcal{P}(\Gamma) \to M \) which recognizes \( L(\phi) \). We now show how to get, for any word \( w \in \Sigma^\oplus \), the corresponding word over \( \mathcal{P}(\Gamma) \).

Consider \( \mathcal{P}(\Sigma) \) as a \( \oplus \)-monoid where the finite product and shuffle operation are union of sets, and the omega and omega-star operations are identity operations. So every element is a shuffle idempotent. Notice that \( \mathcal{P}(\Sigma) \) is essentially cartesian product of \( |\Sigma|-\)many \( U_1 \)'s. Now there exists a morphism \( g : \Sigma^\oplus \to \mathcal{P}(\Sigma) \) such that \( g(w) = \{ a \mid \text{the letter } a \text{ occurs in } w \} \). The transducer associated with \( g \) is \( \sigma : \Sigma^\oplus \to (\mathcal{P}(\Sigma) \times \mathcal{P}(\Sigma)) \) where \( \sigma(w)[i] = (g(w)[<i], g(w)[>i]) \). It is easy to construct a map \( \tau : (\mathcal{P}(\Sigma) \times \mathcal{P}(\Sigma)) \to \mathcal{P}(\Gamma) \) because \( \sigma(w)[i] \in (\mathcal{P}(\Sigma) \times \mathcal{P}(\Sigma)) \) is enough to decide if \( w \models \psi_i(y) \). Thus \( \mathcal{P}(\Sigma) \square M \) can recognize the language of the \( \text{FO}^2[\subset] \) sentence, and by lemma 8 this is in \( \square^*_w U_1 \).

D. Schützenberger-McNaughton-Papert Theorem

The marked star-free regular expressions over \( \Sigma \) are defined by the following grammar

\[
r = 0 \mid a \mid \neg r \mid r_1 \lor r_2 \mid r_1 \land r_2 \mid r_1 a r_2
\]

Each such expression \( r \) naturally corresponds to a language \( L(r) \subseteq \Sigma^\oplus \) in a straightforward manner: \( L(\emptyset) = \emptyset, L(a) = \{ a \}, L(\neg r) = \Sigma^\oplus \setminus L(r), L(r_1 \lor r_2) = L(r_1) \cup L(r_2) \) and \( L(r_1 a r_2) = L(r_1) \cdot a \cdot L(r_2) \) etc. We are using the marked concatenation operation defined in [3] for scattered words. A language \( L \) is said to be marked star-free iff there exists a marked star-free regular expression \( r \) such that \( L = L(r) \). Before we show that marked star free languages are exactly the first order definable languages, let us see an example.

Example 5. The following expression defines the language, there is no last b: \( \{ w \in \{ a, b \}^\oplus \mid \forall x (b(x) \Rightarrow \exists y > x b(y)) \} \)

\[
r ::= \Sigma^\oplus ba^\oplus
\]

Theorem 4. A language \( L \subseteq \Sigma^\oplus \) is marked star-free iff \( L \) is \( \text{FO}[\subset] \)-definable.

Proof: The left to right direction is by structural induction: an expression \( r \) is translated into a \( \text{FO}[\subset] \)-sentence \( \phi_r \) such that \( L(r) = \{ w \in \Sigma^\oplus \mid w \models \phi_r \} \). The only non-trivial case being \( \phi_{r_1 a r_2} = \exists x \phi_{r_1}[<x] \land (a(x) \land \phi_{r_2}>x \land \phi_{r_1}[<x] ) \) is obtained by relativizing \( \phi_{r_1} \) (resp. \( \phi_{r_2} \)) to positions strictly less (resp. greater) than \( x \). We skip the details.

Towards the right to left direction, in view of Theorem 2, it suffices to show that languages recognized by iterated block-products of \( U_1 \) admit marked star-free regular expressions. We proceed by induction on the number of iterated blocks of \( U_1 \) in the recognizing \( \oplus \)-algebra. The base case is easy.
Let \( h : \Sigma^* \to M \sqcup U_1 \) recognize a language \( L \) via a subset of \( M \times U_1 \). As marked star-free languages are closed under union, it suffices to work in the case where \( L \) is recognized by a fixed \((m,u) \in M \times U_1\). By the block product principle (see Corollary 1 or Figure 3), for \( w \in \Sigma^o \),

\[
   w \in L \iff h_1(w) = m \text{ and } h_2(\sigma(w)) = u
\]

With \( L' = h_1^{-1}(m) \) and \( L'' = \sigma^{-1}(h_2^{-1}(u)) \), this simply means that \( w \in L \iff w \in L' \cap L'' \). Therefore, it suffices if we show \( L' \) and \( L'' \) have marked star-free expressions.

For \( m' \in M \), we denote by \( L_{m'} \), the language \( h_1^{-1}(m') \), recognized by \( M \) and hence, by induction, has a marked star-free expression. As \( L' = L_{m'} \), \( L_m \) has a marked star-free expression over \( \Sigma \). Consider \( u = 0 \), then it is easy to see that \( \sigma^{-1}(h_2^{-1}(u)) = \bigcup (m_1,a,m_2) \in h_2^{-1}(0) L_{m_1} \cdot a \cdot L_{m_2} \). The above ‘decomposition’ of \( \sigma^{-1}(h_2^{-1}(u = 0)) \) makes it evident that it too admits a marked star-free expression over \( \Sigma \). The expression for the case with \( u = 1 \) is simply the complement expression. Thus \( L'' \) too has a marked star-free expression. \( \blacksquare \)

Now we are ready to state the analogue of Schützenberger-McNaughton-Papert theorem for countable words.

**Theorem 5.** [Schützenberger-McNaughton-Papert Theorem]

Let \( L \subseteq \Sigma^+ \). Then the following are equivalent.

1. \( L \) is FO\([<]\)-definable.
2. \( L \) admits a marked star-free regular expression.
3. \( L \) is recognized by block products of \( U_1 \).
4. The syntactic \( \oplus \)-monoid of \( L \) satisfies the equations\(^3\):
   - \( e^2 = e \Rightarrow e^* \cdot e^* = e \).
   - \( e^* = e^* = e \Rightarrow \{e\}^* = e \).
   - \( \{e\}^k = e \land eae = e \Rightarrow \{e,a\}^k = e \).

**Proof:** The equivalence of the first three conditions follows from theorems 2 and 4. The equivalence of the first and the last condition was shown in [6]. \( \blacksquare \)

Over finite words, the above theorem holds with the last condition replaced by: ‘the syntactic monoid of \( L \) is aperiodic’ and with some caution: the negation operation in the regular expressions is interpreted wrt \( \Sigma^+ \) and the block products are applied to monoids.

Over countable scattered words, it was shown in [3] that FO and marked star-free regular expressions have the same expressive power. It also gives an algebraic characterization (similar in spirit to the last condition in Theorem 5 invoked for \( \diamond \)-algebras). They also asked if the ‘scattered’ hypothesis can be removed. Theorem 5 resolves this question satisfactorily.

Now we point out an interesting consequence of Theorem 5. Observe that both conditions 3 and 4 are algebraic in nature. The implication\(^4\) 3 \( \Rightarrow \) 4 is rather easy to prove. The implication\(^4\) 4 \( \Rightarrow \) 3 may be viewed as a suitable language-theoretic version of a special case of the Krohn-Rhodes theorem. As mentioned in the introduction, a special case of Krohn-Rhodes theorem asserts that an aperiodic monoids divides a block product of \( U_1 \). Roughly speaking, 4 \( \Rightarrow \) 3 asserts that a \( \oplus \)-monoid satisfying the equations in 4 can be simulated by a block products of \( U_1 \).

**V. LINEAR TEMPORAL LOGIC**

**A. Definitions**

For the rest of the section \( \Sigma \) will denote the alphabet unless mentioned otherwise. A marked word \( w \) is a word in \( \Sigma^+ \cdot (\Sigma, \#) \cdot \Sigma^+ \). We denote by \( \Sigma^\# \) the set of all marked words over \( \Sigma \). We will alternatively denote a marked word \( w \in \Sigma^\# \) by \((w,i) \in \Sigma^\# \), \( i \in \text{dom}(w) \) and \( w = w_{[i)[w[i, \#)]w_{>i}} \).

In this section we look at the subclass \( \text{LTL}[S,U] \) which is closed under boolean operations but only the strict until and since operators are allowed. It is known that \( \text{LTL}[S,U] \) is expressively less powerful than FO\([<]\). The language in Example 4 is not definable in \( \text{LTL}[S,U] \). See [9] for more details. \( \text{LTL}[S,U] \) over the alphabet \( \Sigma \) is closed under boolean operations and the following operators

\[
   p \in \Sigma \mid X \alpha \mid Y \alpha \mid \alpha S \beta \mid \alpha U \beta \mid \alpha S' \beta \mid \alpha U' \beta
\]

\( \text{LTL}[S,U] \) formulas are interpreted on marked words over \( \Sigma \). Given a formula \( \alpha \) and a marked word \((w,i) \), we say that \((w,i) \) satisfies \( \alpha \) (denoted by \((w,i) \models \alpha \)) if \( \alpha \) is true at position \( i \) in the word \( w \). We denote by \((w,i) \not\models \alpha \) if \((w,i) \) does not satisfy \( \alpha \). Inductively the semantics for proposition and boolean combinations are defined as follows

\[
\begin{align*}
   (w,i) &\models p \text{ if } w[i] = p \\
   (w,i) &\not\models \neg \alpha \text{ if } w[i] \not\models \alpha \\
   (w,i) &\models \alpha \lor \beta \text{ if } w[i] \models \alpha \text{ or } w[i] \models \beta
\end{align*}
\]

We say that \((w,i) \models \alpha S \beta \) if

\[
   \exists j < i \ (w,j) \models \beta \text{ and } \forall k \in (j,i) \ (w,k) \models \alpha
\]

Similarly \((w,i) \models \alpha U \beta \) if

\[
   \exists j > i \ (w,j) \models \beta \text{ and } \forall k \in (i,j) \ (w,k) \models \alpha
\]

\( X \alpha \) and \( Y \alpha \) are defined as \( \perp U \alpha \) and \( \perp S \alpha \) respectively. Similarly, \( P \alpha \), \( G \alpha \) and \( F \alpha \) stands for \( T \alpha \), \( \neg (P \neg \alpha) \) and \( T \ U \alpha \) respectively. We introduce two new operators: \( \alpha S \beta \) and \( \alpha U \beta \). The semantics follows:

\[
\begin{align*}
   (w,i) &\models \alpha S \beta \text{ if } \\
   \exists j > i \ (w,j) \models \beta \text{ and } \forall k > j \ (w,k) \models \alpha
\end{align*}
\]

Similarly, we say that \((w,i) \models \alpha U \beta \) if

\[
   \exists j < i \ (w,j) \models \beta \text{ and } \forall k < j \ (w,k) \models \alpha
\]

Note that both the above operators are definable using the until and since operators. Henceforth we will denote by \( \text{LTL}[S,U] \) to mean the logic closed under boolean operations and the four temporal operators we defined above: \( \{S, U, F, U \} \). Note that both the logics are expressively equivalent but this new logic helps characterizing \( \text{LTL}[S,U] \) algebraically. Let \( A, B \subseteq \Sigma \). Then, we use the following shorthand notation: \( A \) stands for the formula \( V_{a \in A} \). For example, \( A S B \) denotes the formula \( \bigvee_{a \in A} S \bigvee_{b \in B} \).

We also define the operator depth of a formula to be the maximum number of operators in a path.
of the parse tree of the formula. It is inductively defined as follows - the depth of a formula with no temporal operator is zero, and the operator depth of any formula of type \( \alpha \times \beta \), where \( X \) is one of the four operators, is one plus the maximum of the operator depth of \( \alpha \) and \( \beta \).

For a formula \( \alpha \), the marked language of \( \alpha \) (denoted by \( \mathcal{L}_\#(\alpha) \)) is the set of all marked words \( (w, i) \) which satisfy \( \alpha \).

For a monoid \( M \), we denote by \( \mathcal{L}_\#(\Phi) \) the set of all marked languages, marked languages defined by formulas in \( \Phi \).

For a monoid morphism \( h : \Sigma^\circ \rightarrow M \), we define the function \( h : \Sigma^\circ \rightarrow (M \times \Sigma \times M) \) by \( h(u, a, #)v) = (h(u), a, h(v)). \) We say that a marked language \( L \) can be recognized by the morphism \( h \), if there exists a set \( S \subseteq (M \times \Sigma \times M) \) such that \( L = h^{-1}(S) \).

Consider the monoids \( M' \) and \( M \) shown in Figs. 5 and 6. Our aim is to show that languages recognized by weak block products of \( M' \) and \( M \) are exactly those definable in Fig. 1.LTL[S,U]. First the base case. The following lemma shows that the marked language of \( A S B \) is recognizable by \( M' \).

**Lemma 9.** Let \( A, B \subseteq \Sigma \). The language \( \mathcal{L}_\#(A S B) \) can be recognized by a morphism, \( h : \Sigma^\circ \rightarrow M' \). Moreover \( \mathcal{L}_\#(A S B) \) can be recognized by a morphism, \( h : \Sigma^\circ \rightarrow M' \). Similarly \( \mathcal{L}_\#(A B) \) and \( \mathcal{L}_\#(A U B) \) can be recognized by morphisms \( h : \Sigma^\circ \rightarrow M' \).

**Proof:** We show that \( L = L_\#(A S B) \) can be recognized by \( M' \) and leave the rest of the proofs since it is of similar flavour. Note that if \( (w, i) \in A S B \) then \( w_{<i} \in \Sigma^\circ BA^\circ \).

We claim that the morphism, \( h : \Sigma^\circ \rightarrow M' \) which extends

\[
h(s) = \begin{cases} a, & \text{if } s \in A \cap B \\ 1, & \text{if } s \in A \setminus B \\ ba, & \text{if } s \in B \setminus A \\ (ba)^\kappa, & \text{o.w.} \end{cases}
\]

can recognize the language \( L \). It is sufficient if we show: for all words \( u \in \Sigma^\circ \), \( u \in \Sigma^\circ BA^\circ \) if and only if \( h(u) \in \{a, ba\} \).

First we show the forward direction. Let \( u \in \Sigma^\circ BA^\circ \), then there exists a position \( i \) such that \( u[i] \in B \) and for all \( j > i \), we have \( u[j] \in A \). In other words \( h(u[i]) \in \{a, ba\} \) and if \( u[i] \) is not empty \( h(u[i]) \in \{1, a\} \). In all cases \( h(u[i]) \in \{a, ba\} \) and \( h(v_2) = a \). In both the case \( u \in L \).

Below we show that the other direction is also true. That is, the marked languages definable by \( M' \) can be recognized by boolean combinations of depth one formulas.

**Lemma 10.** Consider the morphism \( h : \Sigma^\circ \rightarrow M' \). All marked languages definable using \( h \) can be recognized by a boolean combination of formulas of the form \( A S B \) and \( A S B \).

Similarly marked languages definable using \( h : \Sigma^\circ \rightarrow M' \) can be recognized by a boolean combination of formulas of the form \( A U B \) and \( A U B \).

**Proof:** We show that marked languages definable using \( h : \Sigma^\circ \rightarrow M' \) can be recognized by a boolean combination of formulas of the form \( A S B \) and \( A S B \). The case of \( M' \) can be similarly proved. Let \( L \) be a marked language recognized by \( h \). Let \( h \) the associated function, \( h : \Sigma^\circ \rightarrow (M', \Sigma, M') \) and \( S \subseteq (M', \Sigma, M') \) such that \( L = h^{-1}(S) \). We will show that for all elements \( (m, s, n) \in S \), the marked language \( h^{-1}(m, s, n) \) is recognized by a boolean combination of since and until.

Let \( h(w, i) = (m, s, n) \). It is easy to check that the marked position (i.e. \( w[i] \)) contains letter \( s \in \Sigma \).

First, for each \( m \in M' \), we give a formula which is true if and only if \( h(w[i]) = m \). For a set \( S \subseteq M' \), we define the alphabet \( \Sigma_S = \{s \in \Sigma \mid h(s) \in S \} \). Then \( h^{-1}(m) \) for each \( m \in M' \) can be recognized as follows:

\[
\begin{align*}
    h^{-1}(1) &= GG_1 \\
    h^{-1}(a) &= G\Sigma_1(a) \land P\Sigma_a \\
    h^{-1}(ba) &= G\Sigma_1 a \land P\Sigma_{ba} \lor (G\Sigma_{1,a} \land P\Sigma_{ba,ba}^{-1}) \\
    h^{-1}(0) &= (ba)^\kappa
\end{align*}
\]

\( h^{-1}(0) \) is the set of all words not in \( h^{-1}(x) \) for an \( x \neq (ba)^\kappa \) and hence it is also definable: complement of the union of all the above languages.

Now, for each \( n \in M' \), we give a formula which is true if and only if \( h(w[i]) = n \). Again, consider the sets \( \Sigma_S \) which were defined above. Note that if \( n = 1 \), then for all \( k > i \)
we have \( w[k] \in \Sigma_1 \). The following formula recognize this: 
\( \neg(\Sigma S \bigwedge \{a,ba,(ba)^r\}) \). Similarly if \( n = a, \) then for all \( k > i, \) we have \( w[k] \in \Sigma_{ba} \). This is also definable as shown before. Now consider the other two elements which are right zeros. If \( n = ba, \) then it could be because of two cases.

1. There is a position \( k > i \) such that \( w[k] \in \Sigma_{ba} \) and all future positions contains only letters from \( \Sigma_1 \).
2. There are two positions \( k_1 > k_2 > i \) such that \( w[k_2] \in \Sigma_{ba}, w[k_1] \in \Sigma_a \) and \( w[i] \in \Sigma_{(ba)^r} \).

The following formula recognize the above two conditions.

\[ \Sigma_1 \hat{S} \Sigma_{ba} \bigvee \{\Sigma_{\{a,ba,(ba)^r\}\}} \]

Note that the future operator (\( F \)) can defined using \( \hat{S} \). Finally if \( n = (ba)^n, \) the formula can be defined by a boolean combinations of the other formulas.

The three formulas in conjunction gives a formula \( \phi \) such that \( L_{\hat{S}}(\hat{\phi}) = h^{-1}(m,s,n) \).

Until now we saw that languages definable by simple formulas (formulas of operator depth 1) are expressively equivalent to those definable by cartesian products of \( M' \) and \( M^t \). _Substitution_ is a standard way to build temporal logics of greater operator depths [22], [21]. Let \( \phi \) be a formula over the alphabet \( \Gamma \). Let \( \{\psi_a\}_{a \in \Gamma} \) be a set of formulas over the alphabet \( \Sigma \). Then \( \phi[a \mapsto \psi_a] \) stands for the formula over \( \Sigma \) got from \( \phi \) by replacing all occurrence of letters \( a \in \Sigma \) by \( \psi_a \).

Substitution is inductively defined as follows.

1. \( \alpha[a \mapsto \psi_a] = \psi_a \)
2. \( \alpha \lor \beta[a \mapsto \psi_a] = \alpha[a \mapsto \psi_a] \lor \beta[a \mapsto \psi_a] \)
3. \( \neg \alpha[a \mapsto \psi_a] = \neg(\alpha[a \mapsto \psi_a]) \)
4. \( \alpha X \beta[a \mapsto \psi_a] = (\alpha[a \mapsto \psi_a])X(\beta[a \mapsto \psi_a]) \)

where \( X \) is one of the operators \( \{S, U, \hat{S}, \hat{U}\} \). In the rest of the section we use substitution to show that \( \text{LTL}_{[S, U]} \) is expressively equivalent to the weak block product of \( M' \) and \( M^t \). Let \( \sigma \) be a state-based transducer. The next lemma shows that under special circumstance \( \text{LTL}_{[S, U]} \) is closed under \( \sigma^{-1} \).

That is, for any formula \( \alpha \) we can find another formula \( \beta \) also in \( \text{LTL}_{[S, U]} \) such that the marked word \( (\sigma(w), i) \) is accepted by \( \alpha \) if and only if \( (w, i) \) is accepted by \( \beta \).

**Lemma 11.** Let \( \Gamma = (M^t \times \Sigma \times M') \) and \( \sigma : \Sigma^* \rightarrow \Gamma^* \) be a state-based transducer. Then, there is a function \( f : \text{LTL}_{[S, U]} \rightarrow \text{LTL}_{[S, U]} \) such that for all \( w \in \Sigma^* \) and \( \alpha \in \text{LTL}_{[S, U]} \) over \( \Gamma \)

\[ (\sigma(w), i) \models \alpha \iff (w, i) \models f(\alpha) \]

**Proof:** The proof is by structural induction. Lemma 10 gives \( f(\alpha) \) for a letter \( \alpha = (m, a, n) \in \{(M^t \times \Sigma \times M'), \} \). Other formulas are inductively given as follows: \( f(\alpha_1 \lor \alpha_2) = f(\alpha_1) \lor f(\alpha_2), \) \( f(\neg \alpha) = \neg f(\alpha), \) \( f(\alpha X \alpha_2) = f(\alpha_1)X f(\alpha_2) \) where \( X \in \{S, U, \hat{S}, \hat{U}\} \). It is easy to check that \( (\sigma(w), i) \models \alpha \iff (w, i) \models f(\alpha) \) holds for the substitution \( f \).

We have built all the tools necessary to show that weak block products of \( M' \) and \( M^t \) can be “simulated” by \( \text{LTL}_{[S, U]} \).

**Lemma 12.** Let \( N \) be a monoid such that \( \mathcal{L}(N) \subseteq \text{LTL}_{[S, U]} \). Then, \( \mathcal{L}(M^t \square N) \subseteq \text{LTL}_{[S, U]} \)

**Proof:** Let \( \hat{N} = (M' \square N) \) and consider the morphism \( h : \Sigma^* \rightarrow \hat{N} \). Our aim is to show that the languages recognized by \( h \) can be defined in \( \text{LTL}_{[S, U]} \). It is enough to show that for an arbitrary \( \tau = (s, f) \in \hat{N}, \) the language \( h^{-1}(\tau) \) can be defined. Let \( w \in \Sigma^* \). By the block product principle, \( w \in h^{-1}(\tau) \) if and only if \( w \in L_1 \) and \( \sigma(w) \in L_2 \) where \( L_1 \) and \( L_2 \) are languages definable in \( M' \) and \( N \) respectively.

From Lemma 10 there is a formula \( \psi \) which defines \( L_1 \). From the assumption of this lemma, there is a formula \( \psi \) which defines the language \( L_2 \). Lemma 11 gives a formula \( f(\psi) \) such that \( \sigma(w) \in L_2 \) if and only if \( w \in \mathcal{L}(f(\psi)) \). Thus the formula \( \phi \land f(\psi) \) recognize the language \( h^{-1}(\tau) \).

Finally, we show the other direction of the above lemma. We observe that any \( \text{LTL}_{[S, U]} \) formula can be built by iteratively substituting formulas of operator depth one [22].

**Lemma 13.** Let \( \Gamma \) and \( \Sigma \) be two distinct alphabets and let \( \alpha \in \text{LTL}_{[S, U]} \) be a formula over the alphabet \( \Gamma \) and \( N \) be a monoid such that \( \mathcal{L}(\alpha) \in \mathcal{L}(N) \). Then \( \mathcal{L}(\alpha[a \mapsto A_1 X_1 B_1]) \in \mathcal{L}(M \square N) \) where for all \( i \leq |\Gamma|, X_i \in \{S, U, \hat{S}, \hat{U}\}, A_i, B_i \subseteq \Sigma \) and \( M \) is a cartesian product of the monoids \( M' \) and \( M^t \).

The above arguments show \( \text{LTL}_{[S, U]} \) can be recognized by weak block products of \( M' \) and \( M^t \).

**Theorem 6.**

\[ \mathcal{L}(\text{LTL}_{[S, U]}) = \mathcal{L}(\square_{[S, U]}(M', M^t)) \]

The above theorem follows from the fact that boolean combinations of formulas can be simulated by cartesian product and vice versa. Lemma 8 shows that cartesian product of two monoids divide block product of them. Also note that, to simulate every substitution we used one block product operation and vice versa. We believe that, with little more work, one should be able to algebraically characterize each class of k-operator depth formulas.

**VI. CONCLUSION**

We have incorporated block products into the recently developed rich algebraic framework for regular languages of countable words and provided a suitable block product principle. Our applications to logic demonstrate the uses of these constructs. More specifically, we have crucially used them to arrive at a Schützenberger-McNaughton-Papert theorem for countable words. We have also obtained a block-product based algebraic characterization of \( \text{LTL} \) with until-since temporal modalities.

We strongly believe that the block product construction presented in this work is well-suited for classifying several natural logics and the fragments thereof. Our work also exposes the possibility of Krohn-Rhodes theorem for finite monoids satisfying generalized associativity.

**REFERENCES**

w = Πi∈αw[i] = Πj∈β(Πj∈dom(w)[j])  
For w the crucial thing to note is that for any j ∈ dom(w[i]), w[i][j] ≠ w[j] in general because w[j] = π(Πk∈b<k<υ1k1) * w[i] * π(Πk∈b<b1)k1). So Πj∈dom(w)[j] = π(Πk∈b<k<υ1k1) * w[i] * π(Πk∈b<b1)k1) = w[i]. This concludes the proof.

B. First order logic

Lemma 6. For any three ⊕-algebra M, N, and P,

M ≤ (M ⊕ N) ≤ P

Proof (sketch): By block product principle we can see that over any alphabet Σ, in LHS P works on the alphabet N × M × Σ × M × N whereas in RHS, it works on the alphabet M × N × M × Σ × M × M. So the feedback in RHS can contain more information. But if we don’t use the full power of the feedback and only check the entry (1, 1) ∈ M × M, for example, to decide the morphism to P, then it effectively becomes similar to LHS.

Lemma 7. The iterated block product is the strongest way of bracketing a list of monoids.

Proof: We prove this by induction on number of monoids in the list and applying lemma 5. If M1, M2, . . ., Mk are not in iterated product form, then it is of the form (M1 ⊗ M2)⊙(M3 ⊗ . . . ⊗ Mk) where i + 1 < k or (M1 ⊗ . . . ⊗ Mi) is not in iterated form. By induction hypothesis, (M1 ⊗ . . . ⊗ Mi) divides its iterated product version and so, if i + 1 = k, then we are done. Else, (Mi ⊗ . . . ⊗ Mk) is of the form (N ⊕ P) and considering (M1 ⊗ . . . ⊗ Mi) = M we have (M1 ⊗ . . . ⊗ Mk) ⊕ (M ⊕ N) ≤ (M ⊕ N) ≤ P. We continue this till P = M and at that point we are done.

C. Linear temporal logic

Lemma 13. Let Γ and Σ be two distinct alphabets and let α ∈ LTL[S,U] be a formula over the alphabet Γ and Σ be a monoid such that L(α) ⊆ L(N). Then L(α[a1 ↦ A1,X,B1]) ⊆ L(M ⊗ N) where for all i ≤ |Γ|, X1 ∈ {S, U, S[ U]}, A1, B1 ⊆ Σ and M and N is a cartesian product of the monoids M1 and M′.

Proof: Since we are substituting letters we assume that the pairwise intersection of the marked languages of A1,X,B1 is empty. From the assumptions of the lemma, there exists a morphism g : Γ0 → N and T ⊆ N such that g−1(T) = L(α). From lemma 10 we know that there exists a monoid M ∈ [M1] × [M′] and a morphism h : Σ0 → M and S1 ⊆ M × Σ × M for i ≤ |Γ| such that h−1(Si) = L(α[A1,X,B1]). Consider the morphism which extends f : M × Σ × M → Γ where f(m, b, n) = a1, if a1 ↦ A1,X,B1 is a substitution and (m, b, n) ∈ Si. Let the state-based transducer be σ : Σ0 → (M × Σ × M)0. Now for a word w ∈ Σ0 we claim the following
This can be proved by induction on the formula $\beta$. It is clear that the claim holds when $\beta$ is a letter. The induction hypothesis goes through easily. So, let us now consider the composition of functions $g \circ f : M \times \Sigma \times M \to N$ where $g(f(m, b, n)) \in N$.

From the above claim we get that

$$w \in L(\alpha[a_i \mapsto A_iX_iB_i]) \iff g(f(\sigma(w))) \in T$$

This proves the lemma.