正規表現研究の過去・現在・未来（2024年版）

新屋　良磨
秋田大学 数理科学コース
ryoma@math.akita-u.ac.jp

1　はじめに

こんにちは，秋田大学の新屋と申します．2024年もいよいよ終わろうとしていますね．いきなり個人的な話で恐縮ですが，私は学部3年生の頃に「詳説 正規表現」（通称「ふくろう本」）を購入して正規表現に興味を持って以来，かれこれ15年ほど正規表現・オートマトン理論の研究にハマっております．正規表現・オートマトンの面白いところは，なんといっても「正規表現マッチング」や「モデル検査」など超実用的・応用的な研究方向もある一方,「代数的言語理論」や「重み付きオートマトン」などの基礎理論的な研究方向も深みがある点です．もちろん，計算機科学におけるどのような研究トピックも理論と実用・応用である程度の深さがあると思いますが，正規表現・オートマトンは一線を画していると思っています．

本稿では，正規表現が誕生してから現在にいたるまでどのような研究が行われてきたのか，私見を交えつつ超駆け足で解説しようと思います．私の知識と紙面に限りがありますので，網羅的に解説することは不可能でして，個人的趣味による偏りの多い歴史解説になることをご容赦ください．特に，節では2023年2024年の国内の研究動向についてまとめてみました．日本の正規表現研究の今がいかに盛り上がってるかが伝わると幸いです．

2　正規表現の誕生とその公理化

1951年，アメリカの数学者 Stephen Cole Kleene は *Representation of events in nerve nets and finite automata* [7] という論文にて正規表現とオートマトンを提案し，(A) **形式ニューロン**で計算できる言語[1]，(B) オートマトンで認識できる言語，(C) 正規表現で表現できる言語，の (A)~(C) の概念がそれぞれ等価であることを示しました．形式ニューロンとは，1943年に McCulloch と Pitts が論文 [12] で提唱した，人間の脳を構成するニューロンを基にした計算モデルです．図1は Kleene の原論文 [7] の概要ページの抜粋です．現代ではディープラーニングの文脈で取り上げられることの多いニューラルネットですが，起源をたどると形式ニューロンに行き着くという意味では正規表現と親戚関係にあると言えます．

Kleene は [7] で正規表現の等式系についても考察を行っています．例えば `a*` と `(a*)*`，`a(b|c)` と `ab|ac` は同じパターンを表す正規表現ですが，Kleene は23個の等式を列挙し証明を与えたのです．図2にそのうちの8つを抜粋します：[7] では現代の正規表現では `E|F` と書くところを

[1]形式言語理論において「言語」とは単に「文字列の集合」を指します．ちなみに，Kleene の原論文では文字列は「イベント (event)」と表現されていました．

Summary: To what kinds of events can a McCulloch–Pitts nerve net respond by firing a certain neuron? More generally, to what kinds of events can any finite automaton respond by assuming one of certain states? This memorandum is devoted to an elementary exposition of the problems and of results obtained on it during investigations in August 1951.

REPRESENTATION OF EVENTS

IN NERVE NETS AND FINITE AUTOMATA

S. C. Kleene

図 1: [7] の概要抜粋.

7.2 An algebraic transformation: We list several equivalences:

(1)	$(E \vee F) \vee G$	$\equiv$	$E \vee (F \vee G)$.	Associative laws
(2)	$(EF)G$	$\equiv$	$E(FG)$.	
(3)	$(E^*F)G$	$\equiv$	$E^*(FG)$.	
(4)	$(E \vee F)G$	$\equiv$	$EG \vee FG$.	Distributive laws
(5)	$E(F \vee G)$	$\equiv$	$EF \vee EG$.	
(6)	$E^*(F \vee G)$	$\equiv$	$E^*F \vee E^*G$.	
(7)	E^*F	$\equiv$	$F \vee E^*(EF)$.	
(8)	E^*F	$\equiv$	$F \vee E(E^*F)$.	

図 2: Kleene による正規表現の等式の考察．[7] の p.52 より一部抜粋.

$E \vee F$ と記述しています．(1)~(3) の等式は結合則を，(4)~(6) の等式は分配則をそれぞれあらわしています．(7) と (8) の等式は繰り返しを表す*が常に「繰り返しを 1 つだけ展開」できることを意味しています.

Kleene による正規表現の等式系の考察をさらに推し進めたのがイギリスの数学者 John Horton Conway です．Conway はライフゲームや組み合わせゲーム理論の創始者として知られ，群論を始め多分野で業績を残す大数学者ですが，実は彼は 1971 年に正規表現のテキスト *Regular algebra and finite machines* [3] を執筆・出版しています[2]．Conway は特に正規表現の等式理論に興味を持ち，[3] にてさまざまな等式系を考察しています．この書籍の中で Conway は

同じ言語を表現する任意の 2 つの正規表現は『いくつかの有限個の等式』と
『全ての有限群から生成される無限個の等式』のみを用いてお互いに変形できる　(◇)

[2] [3] は Conway が 1966 年にケンブリッジ大学で開講した講義を元にしたもので，おそらく Conway が書いた初めての書籍です.

という重要な予想を立てました（**正規表現の等式理論の群による公理化！**）．この予想は正規表現やオートマトン理論の道具だけではなく，群論的な深い考察が必要な難問で，長らく未解決でしたが，20 年後の 1991 年にフランスの計算機科学者 Daniel Krob によって肯定的に解決 [9] されました（[9] は **137 ページ**もある大作です！）．正規表現の等式理論には，Conway が立てた重要な未解決問題（*Conway* **の最終予想**）がまだ残っており，現在でも理論的に興味深い分野です．等式理論のより詳細な話については Conway の原典 [3] か，私の解説スライド [32]（オンラインで無料公開されています）をご参照ください．

3　オートマトンと正規表現マッチング

Kleene の原論文 [7] は正規表現だけではなく（現代的な意味での）オートマトンも初めて導入されています．オートマトンは「記憶領域を一切使用しないチューリング機械」とも解釈することができ，計算機との相性もとても良いです．イギリスの数学者・計算機科学者 Alan Mathison Turing によってチューリング機械が提唱されたのは 1936 年 [24] ですから，実は，表現力の高いチューリング機械の方が，はるかに単純で表現力の弱いオートマトンよりもだいぶ先に発見されている点は興味深いです．オートマトンの発明 [7] から 17 年経った 1968 年，アメリカの計算機科学者・凄腕ハッカー Kenneth Lane Thompson によって正規表現のオートマトン方式の実装に関する最初の論文 *Regular Expression Search Algorithm* [23] が発表されました．この論文 [23] はたったの 4 ページほどの論文ですが (D) 正規表現から非決定性オートマトン（NFA）を構成する方法，(E) NFA を効率よくシミュレーションする方法，(F) そのシミュレーションを実行する IBM7094 コードを直接生成する方法，の 3 点を論じており，どれも実装の世界においては記念碑的な仕事です．特に，(D) の構成法は現在では *Thompson* **の構成法**と呼ばれています．さらに，Thompson はこの論文を公開する 1 年前に同様の内容を特許として申請しており，1971 年に特許が承認されています[3]．

Thompson が正規表現エンジンの実装法を論文で紹介して以降，正規表現を検索に使えるエディタの普及が進んでいきました．正規表現をサポートする最初のエディタは Thompson 自身が実装した *ed*（Unix に搭載されているラインエディタ）です．その後，1973 年になって ed から「正規表現検索機能」のみを切り取ったコマンド *grep* が Thompson によって Unix に搭載されました．現在では各種エディタや grep だけではなく，あらゆるプログラミング言語処理系においても正規表現が標準でサポートされているのが当たり前です．それぞれの処理系において正規表現の構文や拡張機能・セマンティクスが微妙に異なり，アメリカの計算機科学者 Donald Ervin Knuth は以下のようなユーモアに溢れる言葉を残しています：

> I define UNIX as 30 definitions of regular expressions living under one roof. — Donald Knuth ([8]).

正規表現による検索・文字列処理は多くのプログラマに愛されるツールとなっており，その高速化は重要な研究課題です．特に，実世界で使われている正規表現は Kleene が提案した純粋な正規表現とは少しかけ離れたもので，多数の拡張機能が搭載されています．また，実装・拡

[3] `http://www.google.com/patents/US3568156` ソフトウェアの特許としては**最初のものの 1 つ**となります．

張のしやすさから，オートマトン方式ではなくバックトラック方式を採用した正規表現エンジンも（特にプログラミング言語の標準サポートエンジンに）多く，バックトラック方式特有の問題を回避・検出する技法は最近でも良く研究されています（5節で後述します）.

4　代数的言語理論

2節で正規表現の等式理論に群論が深く関わってくることを述べました．実は等式理論以外にも，代数はより直接的に正規表現の理論に関わってきます．文字集合 A 上の全ての文字列の集合を A^* で表すことにします．言語 $L \subseteq A^*$ と文字列 $w \in A^*$ に対して，w の L における**文脈** $C_L(w)$ を $C_L(w) = \{(x, y) \in A^* \times A^* \mid xwy \in L\}$ と定義します．すなわち $C_L(w)$ は「言語 L 中に w が出現する前後の文字列のペア」の集合です．また，2 つの文字列 u, v が同じ文脈を持つ $C_L(u) = C_L(v)$ 場合，$u \equiv_L v$ と表すことにします．この関係 $\equiv_L$ は明らかに同値関係になっていますが，A^* 上の合同関係にもなっています：$u \equiv_L v \Rightarrow \forall x, y \in A^* (xuy \equiv_L xvy)$. A^* は「2 つの文字列をくっつける（連接）」という結合法則を満たす二項演算を持つ**モノイド**（結合法則を満たす二項演算と単位元を持つ代数構造：A^* の単位元は長さ 0 の「空文字列」）であるため，合同関係 $\equiv_L$ による商 $A^*/\equiv_L$ もまたモノイドとなります．代数的言語理論においてはこの $A^*/\equiv_L$ を言語 L の**統語モノイド**と呼びます．実は，言語 $L \subseteq A^*$ が正規表現で記述できることと，統語モノイド $A^*/\equiv_L$ が有限になることは同値となるのです (*Myhill-Nerode* **の定理** [13]).

Myhill-Nerode の定理は正規言語と有限モノイドの間の密接な関係を述べている定理ですが、統語モノイドの重要性はそれだけに留まりません．統語モノイドを通じて，言語と代数の種々の概念がお互いに対応することが歴史的に明らかにされてきました．例えば，1965 年に M. P. Schützenberger が**星無し言語**と呼ばれる正期言語の部分クラスと**非周期的**な有限モノイドとの対応関係を見出しました．ここで，星無し言語は以下のように帰納的に定義される言語です：(i) 有限集合は星無し言語．(ii) 2 つの星無し言語の和集合も星無し言語．(iii) 2 つの星無し言語の連接も星無し言語．(iv) 星無し言語の補集合も星無し言語．任意の有限集合は正規言語であり，正規表現が和集合・連接に閉じていることも定義から明らかで，また，正規言語は補集合について閉じているという事実から星無し言語は正規言語の部分クラスであることがわかります．普通の正規表現では繰り返しを表す「星」*演算が使えますが，星無し言語は*が使えず，代わりに補集合が取れるという意味で「星無し (star-free)」と呼ばれています．例えば，正規表現 `E` の**否定**を`!E`（`E` に完全マッチしない文字列に`!E` は完全マッチする）で表すことにすると，`(ab)*`という正規表現で記述される言語は実は `(a.*&.*b&!(.*aa.*|.*bb.*))?`（文字が `a,b` の 2 種類しかない場合）と表現することができる（`.` は「任意の一文字」を表すメタ文字，`E?`は `E` にマッチする文字列か空文字列にマッチする正規表現を表す単項演算子，`&`は積集合を表しており，これは和集合`|`と否定`!`を組合せれば実現可能）ので，`(ab)*`は実は星無し言語を記述している正規表現です[4]．その一方，良く似た正規表現 `(aa)*`で記述される言語は星無し言語では**ありません**．すなわち `(aa)*`と等価な星無し正規表現は実は存在しないのです！こうなってくると，どのような正規表現が星無し言語を記述し，どのような正規表現が星無し言語でないものを記述しているのか，一般的に判定するアルゴリズムがあるのか気になってきます．1965

[4]なぜこのように書けるのか，考えてみてください.

言語	代数	論理	
正規	有限モノイド	単項二階述語論理	Myhill-Nerode の定理 [13]
星無し	非周期的	一階述語論理	Schützenberger の定理 [16]
密度 0 か 1	零元付き	?	*cf.* 新屋の博論 [18]
無曖昧多項式	**DA**	2 変数一階述語論理	Schützenberger の定理 [15]
区分検査可能	$\mathcal{J}$-自明	全称量化子無しの一階述語論理	Simon の定理 [17]
文字検査可能	可換かつ冪等	1 変数一階述語論理	folklore

表 1: 言語・代数・論理の対応.

年にフランスの数学者[5]Marcel-Paul Schützenberger が統語モノイドを通じてそのようなアルゴリズムを与えてくれました：Schützenberger は「正規言語 L が星無し言語であること」と「L の統語モノイドが**非周期的**（非自明な部分群を含まない）こと」が等価であることを示したのです．L を記述する正規表現から L の統語モノイド M_L を構成することができ，Myhill-Nerode の定理から M_L は必ず有限で，M_L が非自明な部分群を含むかどうかは M_L の全ての部分集合（単元集合は除く）に対して群になるかどうかを判定すれば良いので,「与えられた正規表現が星無し言語を記述しているか？」という判定問題は Schützenberger の定理から決定可能となります.

Schützenberger の定理は星無し言語という「言語（正規表現）的に定義された概念」が非周期的という「代数的な概念」と対応することを述べていますが， Schützenberger の定理以降，多くの言語的・代数的さらには論理的な概念に対応関係があることが発見されてきました（表 1にごく一部を例示）．表 1のほとんどのテクニカル・タームは本稿で説明していないものですが，それぞれ言語・代数・論理的に「**自然に**」定義された概念です．表の一番上「正規」から一番下「文字検査可能」に向けてより制限された言語クラスになっていきます（「密度 0 か 1」だけは例外で，これは星無し言語全体と比較不能な言語クラスです）．言語 L が「文字検査可能」であるとは，L が「ある文字 a が出現するかどうか？」を検査する正規表現 `.*a.*` のブール演算（和集合・積集合・補集合）で記述できることを意味しています．このように，言語・代数・論理と（見かけ上）全く異なる世界の「自然」な種々の概念が対応関係を持つのは全くの「偶然」なのでしょうか？いいえ，このような対応関係はある種「必然」であるということをアメリカ（出身はポーランド）の数学者 Samuel Eilenberg が 1976 年の彼の書籍 [4] にて証明しています. Eilenberg は，言語クラス $\mathcal{C}$ が「ブール演算・商・準同型の逆像」という 3 つの演算について閉じている（この場合 $\mathcal{C}$ のことを言語の**多様体** (variety) と呼びます）ならば，$\mathcal{C}$ の統語モノイド全体のクラスがある種の「等式」概念で定義でき，またその逆も成り立つということを示しました（*Eilenberg* **の多様体定理**[6]）.「モノイドのクラスが等式で定義される」とはどういうことか，簡単な例で紹介します：モノイド M が等式 $xy = yx \wedge x^2 = x$ を満たすとは，各元 $a, b \in M$ について $ab = ba$ かつ $a^2 = a$ が成り立つことを言います（すなわち M は**可換かつ冪等**なモノイド）．表 1の一番下の欄の対応は，文字検査可能言語の統語モノイドは必ず可換かつ冪等であり，またその逆も成り立つということを意味しています．Eilenberg の多様体定理は非常に奥深く示唆に富む定理であり，最近になってもその拡張・再解釈が試みられています [5, 1, 26].

[5]医学博士も取得している多才ぶりです！

[6]個人的「形式言語理論で最も好きな定理」の候補の 1 つです．Schützenberger の定理も大好きですが.

5　国内の最近の研究動向

私が博士課程学生をやっていた頃は統語モノイドを知っている国内の研究者はあまりおらず（少なくとも若手研究者は），統語モノイドという概念を知っている学生は極めて少数だったと思います．実際，統語モノイドに言及している和書は田村孝行先生の「半群論」[36]（用語として触れてるぐらい）と「コンピュータ基礎理論ハンドブック II」 [11] の橋口攻三郎先生が翻訳を担当された第 1 章「有限オートマトン」ぐらいだったと記憶しています[7]．しかしながら，最近では「統語モノイド（あるいは統語的モノイド）」で検索すると日本語の文献もいくつかヒットするようになり，代数的言語理論に興味を持つ若手研究者も少しずつ増えてきている印象です[8]．例えば，2023 年に東工大で博士号を取られた湯山孝雄さんは**群言語** (統語モノイドが群になる言語) に関する論文 [27] で国際会議 *Developments in Language Theory* の論文賞を受賞されています．代数的言語理論とはちょっとスコープが外れますが，東工大の中村誠希さんは拡張された正規表現体系の等式理論の決定可能性・計算量に関する研究で 2019 年に博士号を取られ，その方向の研究で 2023 年に理論計算機科学のトップ国際会議の 1 つである *Logic in Computer Science* に論文 [14] が採択されています．

正規表現・オートマトンがスコープに入っている国内の研究集会も多いと思いますが，私が今年の春に参加した「プログラミングおよびプログラミング言語ワークショップ（PPL2024）」では早稲田大の野上大成さんらの**後方参照**付き正規表現に関する研究 [39] が PPL2024 論文賞を受賞[9]されていました．後方参照は現代の正規表現の便利な拡張機能[10]の一つで，後方参照を備えた正規表現は表現力が格段に強くなることが古くから知られていましたが，[39] では後方参照付き正規表現と多重文脈自由文法（文脈自由文法を拡張した体系）の表現力の新たな関係を見出していました．同じく PPL2024 では名古屋大の井上裕介さんらが統語モノイドの埋込み・分解理論に関する講演 [29] を行っており，こちらも大変興味深かったです．現代の正規表現において，後方参照の他にある重要な拡張機能として**先読み・後読み**があります．後方参照とは違い，先読みは正規表現の表現能力を超える拡張機能ではありませんが，複雑なパターンを簡潔に記述できる便利な機能です．先読み・後読みが正規表現の表現能力を超えないことは古くから知られていましたが，東大の森畑明昌先生の 2012 年の論文 [35] では重み付きオートマトンを用いた先読みの効率の良い実装法が議論されています．それ以降も，先読みに関する理論解析や実装の研究講演も国内（主に PPL 界隈？）で近年も良く見かける印象です．ところで先読みと後方参照を**両方**備えた正規表現体系はどのような言語を記述できるのでしょうか？理論計算機科学のトップ国際会議の 1 つである *International Colloquium on Automata, Languages and Programming* 2024 に採択された上里友弥さん（サイバーエージェント）の論文 [25] では，先読みと後方参照を備えた正規表現体系が非決定性対数領域チューリングマシンで認識できる言語クラス (**NLOG**) と対応することが示されています．後方参照と先読みという古くから知られている正規表現の拡張機能が，同じく古くから研究されている **NLOG** という計算量クラスと一致することが今年になって示されたのは大変面白いです．

[7] 私自身は統語モノイドの基礎知識や Eilenberg の多様体定理は Mark V. Lawson の代数的言語理論の教科書 [10] で勉強しました．超良書です．

[8] 勘違いじゃないといいのですが...

[9] 野上さんは PPL2023 でも後方参照付き正規表現の研究で論文賞を受賞されています．すごい．

[10] というより**魔改造**？

3節で軽く言及したように，バックトラック方式の正規表現エンジンはバックトラック特有の問題点があり，与えられた正規表現と文字列によってはマッチングに非常に時間がかかってしまう場合があります．例えば Python で標準の正規表現モジュールである re モジュールをインポートし，`r = re.compile("^(a?){50}a{50}$")` を実行してみてください．変数 `r` に正規表現`^(a?){50}a{50}$`をコンパイルした結果が代入されます．この正規表現は「a の 50 から 100 回までの繰り返し」に（完全）マッチする正規表現です．続いて `r.match("a"*50)` を実行してみてください．`"a"*50` は a がたかが 50 回連続して現れる短い文字列ですが，マッチングが終了しないはずです．これはバックトラックの組合せが指数的に増えうることから起こる問題で，このような問題を意図的に引き起こし正規表現マッチングを行っているサーバーに負荷をかける行為は *ReDoS* **攻撃**と呼ばれています．NII の藤浪大弥さんは ReDoS を受ける可能性のある正規表現を検出するプログラム recheck を開発し，その実装に関する 2022 年のプログラミング・シンポジウムの講演 [37] で第 37 回山内奨励賞を受賞されています．また，藤浪さんは「メモ化を用いて ReDoS を解消する手法」を Ruby の正規表現エンジンに実装し，その功績で 2023 年の Ruby Prize の最終ノミネーターに選出[11]されています．PPL2023 サマースクールのテーマは「正規表現研究の最先端」[12]ということで，東工大の南出靖彦先生は「ReDoS の起こり得る正規表現を判定する技術」(*cf.* [30]) をテーマに講義を行われていました．同じく PPL2023 サマースクールで講師を務めた NTT 社会情報研究所の千田忠賢さんは「ユーザーが正規表現にマッチしてほしい文字列（正例）・ほしくない文字列（負例）を入力して正規表現を自動修正する技術」をテーマに講義を行われていました．千田さんと早稲田大の寺内多智弘先生がこの技術をまとめた論文は 2023 年にプログラミング言語分野のトップ国際会議である *Programming Language Design and Implementation* に採択されています [2].

Eilenberg の多様体定理から，新たな言語の多様体を発見すると，必ずそれに対応する（場合によっては既知の）モノイドの等式クラスが存在することが保証されます．これまで多くの言語の多様体が発見され，それに対応するモノイドの等式クラスが明らかになっていますが，まだ未発見の言語の多様体も多く存在すると思われます．私は 2021 年頃から ***C* 可測性**と呼ばれる測度論から着想をえた概念を形式言語理論に導入し，種々の新しい言語の多様体を発見しその代数的特徴付けを与えることに成功しています [19, 20, 21, 22, 34, 6, 28]．紙面に限りがあるため *C* 可測性の詳細を説明をすることはできませんが，言語の多様体や可測性について興味のある読者は私の最近の共著日本語論文 [34] をぜひご参照ください（オンラインで無料で読めます）.

6 おわりに

本稿では正規表現の誕生から始め最近の発展まで超駆け足かつ偏った視点でまとめてみました．理論的に非常に面白い「重み付きオートマトン」については本稿で触れることはできませんでしたが，興味のある方は拙著サーベイ [33] をご参照ください（こちらもオンラインで無料で読めます）．重み付きオートマトン以外にも，特に応用・実装面について興味深いトピックも数多くあります.「モデル検査」はプログラミング言語理論界隈へのオートマトン理論の最も重要

[11] `https://rubyprize.jp/`
[12] `http://ppl.jssst.or.jp/index.php?ss2023`

な応用先の一つです．モデル検査について興味のある読者は，東大の小林直樹先生の PPL2013 サマースクールの講義資料 [31]（オンラインで無料で読めます）を参照するか，2022 年に出版された「理論計算機科学事典」 [38] の「モデル検査」の節を参照することをお勧めします．

正規表現・オートマトンは非常に素朴な定義を持つ対象ですが，その素朴さ故に無数の拡張が存在します．それら拡張体系に対する研究などなど今後もテーマが尽きることはないでしょう．本稿を通じて正規表現研究に興味を持たれた方が少しでもいらっしゃいましたら幸いです．ぜひ一緒に正規表現・オートマトンに対する人類の理解を深めていきましょう！

参考文献

[1] Adámek, J., Milius, S., Myers, R.S.R., Urbat, H.: Generalized Eilenberg Theorem I: Local varieties of languages. In: Foundations of Software Science and Computation Structures. pp. 366–380 (2014)

[2] Chida, N., Terauchi, T.: Repairing regular expressions for extraction. In: Proceedings of the ACM on Programming Languages (Issue PLDI). vol. 7, pp. 1633–1656 (2023)

[3] Conway, J.H.: Regular algebra and finite machines. Chapman and Hall (1971)

[4] Eilenberg, S., Tilson, B.: Automata, languages and machines. Volume B. Pure and applied mathematics, Academic Press (1976)

[5] Gehrke, M., Grigorieff, S., Pin, J.É.: A topological approach to recognition. In: Automata, Languages and Programming. pp. 151–162 (2010)

[6] Inaba, K., Sin'ya, R., Nakamura, Y., Yamaguchi, Y.: Computational complexity of at-, pt- and gd-measurability for regular languages. In: The 26-th Workshop on Programming and Programming Langugaes (2024), (written in Japanese)

[7] Kleene, S.C.: Representation of events in nerve nets and finite automata. RAND Corporation (1951), Project RAND Research Memorandum RM-704

[8] Knuth, D.: Digital typography. CSLI Publications (1999)

[9] Krob, D.: Complete systems ofb-rational identities. Theoretical Computer Science **89**(2), 207–343 (1991)

[10] Lawson, M.V.: Finite Automata. Chapman and Hall/CRC (2003)

[11] Leeuwen, J.V.: コンピュータ基礎理論ハンドブック 形式的モデルと意味論, vol. II. 丸善出版 (1994)

[12] Mcculloch, W.S., Pitts, W.H.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics **5**, 115–133 (1943)

[13] Myhill, J.R.: Finite Automata and the Representation of Events. Tech. Rep. WADC TR-57-624, Wright-Paterson Air Force Base (1957)

[14] Nakamura, Y.: Existential calculi of relations with transitive closure: Complexity and edge saturations. In: Logic in Computer Science. pp. 1–13. IEEE (2023)

[15] Schützenberger, M.: Sur le produit de concaténation non ambigu. Semigroup Forum **13**, 47–75 (1976)

[16] Schützenberger, M.P.: On finite monoids having only trivial subgroups. Information and Control (8), 190–194 (1965)

[17] Simon, I.: Piecewise testable events. In: Automata Theory and Formal Languages. pp. 214–222 (1975)

[18] Sin'ya, R.: Zero-One Law for Regular Languages. Ph.D. thesis, Tokyo Institute of Technology (2016)

[19] Sin'ya, R.: Asymptotic approximation by regular languages. In: Current Trends in Theory and Practice of Computer Science. pp. 74–88 (2021)

[20] Sin'ya, R.: Carathéodory extensions of subclasses of regular languages. In: Developments in Language Theory. pp. 355–367 (2021)

[21] Sin'ya, R.: Measuring power of locally testable languages. In: Diekert, V., Volkov, M. (eds.) Developments in Language Theory. pp. 274–285. Springer International Publishing, Cham (2022)

[22] Sin'ya, R.: Measuring power of generalised definite languages. In: Implementation and Application of Automata. pp. 278–289. Springer International Publishing (2023)

[23] Thompson, K.: Programming techniques: Regular expression search algorithm. Commun. ACM **11**(6), 419–422 (Jun 1968)

[24] Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London mathematical society **42**(2), 230–265 (1936)

[25] Uezato, Y.: Regular expressions with backreferences and lookaheads capture NLOG. In: Bringmann, K., Grohe, M., Puppis, G., Svensson, O. (eds.) 51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia. LIPIcs, vol. 297, pp. 155:1–155:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2024). https://doi.org/10.4230/LIPICS.ICALP.2024.155, https://doi.org/10.4230/LIPIcs.ICALP.2024.155

[26] Uramoto, T.: Semi-galois categories I: The classical Eilenberg variety theory. In: Logic in Computer Science. pp. 545–554 (2016)

[27] Yuyama, T.: Groups whose word problems are accepted by abelian *g*-automata. In: Developments in Language Theory. pp. 246–257. Cham (2023)

[28] Yuyama, T. and Sin'ya, R.: Measuring power of commutative group languages. In: Implementation and Application of Automata. pp. 347–362. Springer International Publishing (2024)

[29] 井上裕介, 橋本健二, 関浩之: 周期的な正則言語の巡回群による半直積分解. In: 第 26 回プログラミングおよびプログラミング言語ワークショップ (2024)

[30] 高橋和也, 南出靖彦: 拡張正規表現マッチングの計算量解析. コンピュータ・ソフトウェア **38**(2), 53–70 (2021)

[31] 小林直樹: 高階モデル検査とその応用 （PPL サマースクール 2013 の講義スライド）, http://www-kb.is.s.u-tokyo.ac.jp/ koba/ss2013/

[32] 新屋良磨: 正規表現に潜む対称性〜等式公理による等価性判定〜（研究集会 SLACS2016 での招待講演スライド） (2016), http://math.akita-u.ac.jp/~ryoma/misc/slacs2016.pdf

[33] 新屋良磨: オートマトン理論再考. コンピュータ・ソフトウェア **34**(3), 3–35 (2017)

[34] 新屋良磨, 山口勇太郎, 中村誠希: 部分語の出現情報の検査のみで近似できる正規言語について（PPL2022 推薦論文）. コンピュータ・ソフトウェア **40**(2), 49–60 (2023)

[35] 森畑明昌: 先読み付き正規表現の有限状態オートマトンへの変換（PPL2011 推薦論文）**29**(1), 147–158 (2012)

[36] 田村孝行: 復刊 半群論. 共立出版 (2001)

[37] 藤浪大弥: Redos 検出プログラム recheck の開発. In: 第 62 回 プログラミング・シンポジウム (2021)

[38] 徳山豪・小林直樹（総編集）: 理論計算機科学事典. 朝倉書店 (2022)

[39] 野上大成, 寺内多智弘: Regular expressions with backreferences on multiple context-free languages, and star-closedness. In: 第 26 回プログラミングおよびプログラミング言語ワークショップ (2024)