Muller–Schupp の定理: 群論と形式言語理論

Muller–Schupp theorem: groups and languages

y. (http://iso.2022.jp)

2021 年 M 月 D 日

最終更新日: 2021 年 5 月 1 日

はじめに

概要

Muller–Schupp の定理は数学における群論と形式言語理論という 2 つの分野に関係する定理であり，有限生成群 G に対して，次の 2 条件が同値であることを主張するものである．

(1) G が**実質的自由群** (virtually free group) である．つまり G の部分群で，自由群であってしかも群指数が有限であるようなものが存在する．
(2) G の語の問題が**文脈自由言語** (context-free language) である．つまり，G の生成系 Σ をひとつ固定したとき，Σ の元の有限列であって G の中で単位元を表すようなもの全体の集合が，ある**文脈自由文法** (context-free grammar) によって生成される．

ここで条件 (1) は純粋に群論的な条件であり，条件 (2) は形式言語理論的な条件である．両者に含まれる「自由 (free)」という単語は互いにまったく異なる由来を持つので，これらが同値となることは決して当たり前のことではない．本稿では Muller–Schupp の定理の主張に現れる「実質的自由群」「語の問題」「文脈自由言語」などの用語の定義を例とともに解説し，Muller–Schupp の定理の厳密かつ自己完結的な証明を与える．本稿の内容は Diekert と Weiß によるノート [DW17] を大いに参考にしている．

Keywords: 実質的自由群 (virtually free groups)，群の語の問題 (word problem for groups)，文脈自由言語 (context-free languages)，オートマトン (automata)，書き換え系 (rewriting systems)，擬等長同型 (quai-isometry)，木幅 (treewidth)，最適入れ子状カット (optimally nested cut)，群付きグラフの基本群 (fundamental group of a graph of groups)，Bass–Serre 木 (Bass–Serre tree)

本稿の内容

本稿の目標は，以下の定理に完全な証明を与えることである．定理の主張に含まれる用語については必要に応じてその都度解説することにする．

定理 0.1. 有限生成群 G に対し，以下の条件はすべて同値である．

(A) G は実質的自由群である．
(B) G の語の問題は決定性文脈自由言語である．
(C) G の語の問題は文脈自由言語である．
(D) G の Cayley グラフはある木に擬等長同型である．
(E) G の Cayley グラフの木幅が有限である．
(F) G は頂点群がどれも有限群であるような，ある有限な群付きグラフの基本群である．

(G) G はある有限群 Q と自由群 F の半直積 $F \rtimes Q$ の有限指数部分群である．
(H) G はある木に作用し，その軌道は有限個であり，かつ各頂点の固定群が有限群である．

本稿の構成

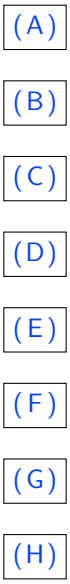


図 1　本稿で証明する導出関係

本稿の性格

本稿は「入門書を兼ねた筆者の備忘録」である．入門書であるから，理解を助けるための具体例を随所に挿入しつつ，自己完結的で行間の少ない証明を心掛けた．しかしその厳密さと備忘録という性質上，本稿には必要以上にくどい証明が多く，ページ数も長大になってしまった．細かい証明などは書いておかないと筆者自身が忘れてしまうので，仕方のないことではあるのだが，そのせいで却って読み難くなっているところもあるかもしれない．そんなときには証明のアイデアだけ把握して適度に読み飛ばしてほしい．証明に際してはアイデアや意図ができるだけ伝わるように書いたつもりなので，多少読み飛ばしても問題は生じないと思う．

目次

記号一覧

記号	意味	初出
$A := B$	記号 A を B で定義する	
$P :\Longleftrightarrow Q$	P であることを Q が成り立つこととして定義する	
$\forall x\,[\cdots]$	すべての x に対して…が成り立つ	
$\exists x\,[\cdots]$	…が成り立つような x が存在する	
$\exists! x\,[\cdots]$	…が成り立つような x がただ一つ存在する	
$\wedge$	かつ	
$\vee$	または (少なくとも一方)	
$\neg$	…でない	
$\mathbb{N}$	自然数 (0 を含む) 全体の集合	
$\mathbb{Z}$	整数全体の集合	
$X - Y$	X と Y の差集合	
$X \hookrightarrow Y$	X から Y への単射	
$X \twoheadrightarrow Y$	X から Y への全射	
Σ^*	アルファベット Σ 上の文字列全体の集合	
$\lvert w\rvert$	文字列 w の長さ	
$\lvert w\rvert_a$	文字列 w に含まれる文字 a の個数	
L^*	L が生成する部分モノイド	
$\mathrm{L}(G)$	文法 G が生成する言語	
$\mathrm{L}(\mathcal{A})$	オートマトン $\mathcal{A}$ が認識する言語	
REG	正則言語 (正規言語) 全体のクラス	
CFL	文脈自由言語全体のクラス	

第 I 部

文字列の数学

第 1 章

書き換え系入門

本章では**書き換え系** (rewriting system) と呼ばれる概念を整備する．書き換え系は本稿全体にわたって用いられる基本的な道具であり，特に**文字列書き換え系** (string rewriting system) は，ともすれば組合せ群論において省かれがちな証明の細部を補完にするのに役立つ強力なツールである．書き換え系についてより詳しく知りたい読者は，例えば Baader と Nipkow による教科書 [BN98] などを参照のこと．

書き換え系の感じをつかんでもらうために，文字列書き換え系の代わりに**項書換え系** (term rewriting system) を例にとって説明する (一般的には文字列書き換え系よりも項書き換え系の方が有名であろう)．算数のテストで，「$1+2$ を計算しなさい」という問題が出されたとしよう．当然，答えは「3」であるが，ひねくれた回答者が答案用紙に「$7-4$」と書いて提出したら，採点者は丸をつけないだろう．値としては 3 と等しいにも関わらず，なぜ不正解なのだろうか？もちろん，こんな答案を許してしまっては「$1+2=?$」に対して「$1+2$」と答えても正答となってしまい，テストをする意味がなくなってしまうからであるが，「3」だけが正解として扱われるのは，数値としての 3 を表す表記の中で「3」が “最も簡単な形” だからだと考えられる．

ここで，算数の計算問題とは「与えられた式 (= 項) を，その値を変えずに，最も簡単な形に “書き換え” よ」という問題である，という立場をとることにする．ここで「最も簡単」という表現は，「これ以上計算を進めることができない」とも言い換えられる．「計算を進める」という表現からもわかる通り，「$1+2$」という表記から「3」という表記を得るプロセスは一方通行であり，この意味で計算という行為はある種の非対称性を内包している．同じ値を表す表記が複数あるとき，それらを一緒くたに同一視せず，優劣を付けて “非対称化” するのが書き換え系のアイデアである，ということもできる．もちろん，優劣の定め方は一通りではなく，例えば $(x+1)/x$ と $1+1/x$ のどちらがより “簡単” な表記かを一概に決めることはできないだろう．

「これ以上計算を進めることができない状態」のことを，書き換え系の言葉では**正規形** (normal form) あるいは**既約である** (irreducible) という．自然数の足し算については結合法則が成り立つので，例えば $5+3+7$ を $5+3$ から計算 (= 書き換え) しても $3+7$ から計算しても，最終的な結果はどちらも 15 になる．このように，表記を書き換える順序によらずに同じ計算結果が得られるという性質を，書き換え系の言葉では**合流性** (confluence) という．また，有限個の自然数の足し算は有限回の書き換えによって計算することができ，無限ループに陥ることがない．このように，どんな計算も (どんな順序で計算しても) いつか必ず終わるということは，書き換え系の言葉では**停止性を持つ** (terminating) という条件に対応する．

書き換え系は表記が表す値が異なることを証明するのに役立つことがある．例えば，$3+5+8$ と $1+2+9$ は明らかに異なる値を表すが，ではその理由は，と尋ねられれば「$16 \neq 12$ だから」と答えるだろう．これは書き換え系の観点からは，両者を正規形になるまで書き換え，正規形が異なる表記だから違う値であると結論していることになる．これと似たような議論を有限表示群に対して行うことで，群論における数多くの有用な結果を得ることができる．

書き換え系に対する感覚をつかんだところで，早速数学的な内容に入っていこう．

1.1　二項関係に対する演算

集合 X 上の**二項関係** (binary relation) とは，直積集合 $X \times X$ の部分集合のことであった．まずは矢印的な形状の二項関係に関する記法をいくつか用意しておく．

定義 1.1.　X を集合とし，$\longrightarrow$ を X 上の二項関係とする．つまり，$\longrightarrow \subseteq X \times X$ である．2 つの元 $x, y \in X$ に対し，$(x, y) \in \longrightarrow$ が成り立つことを $x \longrightarrow y$ と表す．このとき，以下のように記号を定義する．

$$
\begin{aligned}
\longrightarrow \circ \longrightarrow &:= \{\, (x,y) \in X \times X \mid \exists z \in X\ [x \longrightarrow z \wedge z \longrightarrow y] \,\}, && \text{(合成)} \\
\xrightarrow{0} &:= \{\, (x,y) \in X \times X \mid x = y \,\}, && \text{(恒等関係，対角集合)} \\
\xrightarrow{n+1} &:= \xrightarrow{n} \circ \longrightarrow \quad (n \in \mathbb{N}), && \text{(反復合成)} \\
\xrightarrow{\leq k} &:= \bigcup_{n \leq k} \xrightarrow{n} \quad (k \in \mathbb{N}), && \text{(k 回以下の合成)} \\
\xrightarrow{+} &:= \bigcup_{n \geq 1} \xrightarrow{n}, && \text{(推移閉包)} \\
\xrightarrow{*} &:= \bigcup_{n \geq 0} \xrightarrow{n} = \xrightarrow{0} \cup \xrightarrow{+}, && \text{(反射推移閉包，Kleene 閉包)} \\
\longleftarrow &:= \{\, (x,y) \in X \times X \mid y \longrightarrow x \,\}, && \text{(逆関係)} \\
\longleftrightarrow &:= \longrightarrow \cup \longleftarrow. && \text{(対称閉包)}
\end{aligned}
$$

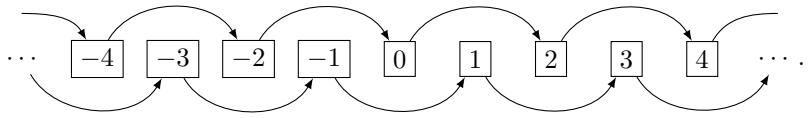
$\xleftrightarrow{*}$ を $\longrightarrow$ の反射推移対称閉包と呼ぶ．

一般に，集合 X 上の二項関係であって，反射律と推移律をみたすもの (つまり，(半) 順序の定義から反対称律を除いたもの) を**前順序** (preorder, quasi-ordering) と呼ぶのであった．定義 1.1 から明らかに次が成り立つ (証明は簡単なので省略する)．

命題 1.2.　集合 X 上の同値関係 $\longrightarrow$ に対し，以下が成り立つ．

(1) $\xrightarrow{*}$ は $\longrightarrow$ を含む最小の前順序関係である．
(2) $\xleftrightarrow{*}$ は $\longrightarrow$ を含む最小の同値関係である．

例 1.3.　整数全体の集合 $\mathbb{Z}$ 上の二項関係 $\longrightarrow \subseteq \mathbb{Z} \times \mathbb{Z}$ を，$m, n \in \mathbb{Z}$ に対し $m \longrightarrow n \ :\Longleftrightarrow\ n = m + 2$ と定義する．この状況を図で表すと以下のようになる:



このとき，$m, n \in \mathbb{Z}$ に対し

$$
\begin{aligned}
m \longleftrightarrow n &\iff |n - m| = 2, \\
m \xrightarrow{*} n &\iff \exists k \geq 0\ [n = m + 2k], \\
m \xleftrightarrow{*} n &\iff m \equiv n \pmod 2
\end{aligned}
$$

であり，同値関係 $\xleftrightarrow{*}$ による $\mathbb{Z}$ の商は $\mathbb{Z}/\xleftrightarrow{*} = \mathbb{Z}/2\mathbb{Z}$ となる．

1.2　抽象書き換え系

ここでは抽象書き換え系と呼ばれる，最も基本的な書き換え系の性質を扱う．抽象書き換え系という大層な名前が付けられてはいるが，その実体はただの二項関係である．

定義 1.4 (抽象書き換え系). 集合 X 上の二項関係 $\longrightarrow \subseteq X \times X$ のことを，X 上の**抽象書き換え系** (abstract rewriting system) という．$x, y \in X$ が $x \xrightarrow{*} y$ をみたすとき，$\longrightarrow$ によって x から y を**導出する** (derive) ことができるといい，定義から存在する X の元の有限列 $x = x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_n = y$ $(n \geq 0)$ のことを $x \xrightarrow{*} y$ の**導出** (derivation) という．$\xleftrightarrow{*}$ に対する導出も同様に定義される．

定義 1.5 (書き換え系の性質). $\longrightarrow$ を X 上の抽象書き換え系とする．

(1) $\longrightarrow$ が**強合流的である** (strongly confluent) とは，$x, y, z \in X$ について $y \longleftarrow x \longrightarrow z$ ならば，ある $w \in X$ が存在して $y \xrightarrow{\leq 1} w \xleftarrow{\leq 1} z$ となることである．強合流的であるという性質を**強合流性** (strong confluence) という．

(2) $\longrightarrow$ が**局所合流的である** (locally confluent) とは，$x, y, z \in X$ について $y \longleftarrow x \longrightarrow z$ ならば，ある $w \in X$ が存在して $y \xrightarrow{*} w \xleftarrow{*} z$ となることである．局所合流的であるという性質を**局所合流性** (local confluence) という．

(3) $\longrightarrow$ が**合流的である** (confluent) とは，$x, y, z \in X$ について $y \xleftarrow{*} x \xrightarrow{*} z$ ならば，ある $w \in X$ が存在して $y \xrightarrow{*} w \xleftarrow{*} z$ となることである．合流的であるという性質を**合流性** (confluence) という．

(4) $\longrightarrow$ が **Church–Rosser 性** (Church–Rosser property) を持つとは，$x, y \in X$ について $x \xleftrightarrow{*} y$ ならば，ある $z \in X$ が存在して $x \xrightarrow{*} z \xleftarrow{*} y$ となることである．

(5) $\longrightarrow$ が**停止性を持つ** (terminating, Noetherian) とは，導出の無限列

$$x_0 \longrightarrow x_1 \longrightarrow x_2 \longrightarrow \cdots$$

が存在しないことをいう．言い換えると，二項関係 $\longrightarrow$ が**逆整礎である** (converse well-founded) ということである．

(6) $\longrightarrow$ が**収束的である** (convergent) であるとは，$\longrightarrow$ が局所合流的かつ停止性を持つことをいう．

まず，整礎関係に関する帰納法[*1]の正当性を確かめておく．

補題 1.6 (整礎関係に関する帰納法). 集合 X 上の二項関係 $\longleftarrow \subseteq X \times X$ が整礎であるとする．すなわち，$\longleftarrow$ に関する X の元の無限下降列

$$\cdots \longleftarrow x_2 \longleftarrow x_1 \longleftarrow x_0$$

が存在しないとする[*2]．このとき，X の元に関する性質 $P(x)$ ついて

$$\forall x \in X\, [\forall y \in X\, [y \longleftarrow x \implies P(y)] \implies P(x)] \implies \forall x \in X\, [P(x)]$$

が成り立つ．

[*1] 俗に整礎帰納法と呼ばれることがある (かもしれない).

[*2] 整礎性は通常「空でないどんな部分集合も極小元を持つ」ことと定義されるが，これは従属選択公理のもとでは無限下降列の非存在と同値である．

証明. 背理法で示す．結論が成り立たないと仮定すると，

$$\forall x \in X\ [\neg P(x) \implies \exists y \in X\ [y \longleftarrow x \wedge \neg P(y)]], \tag{1.1}$$
$$\exists x \in X\ [\neg P(x)] \tag{1.2}$$

が成り立つ．式 (1.2) より，$\neg P(x_0)$ をみたす元 $x_0 \in X$ が存在する．式 (1.1) の x に x_0 を代入すると，$x_1 \longleftarrow x_0$ かつ $\neg P(x_1)$ をみたす元 $x_1 \in X$ を得る．再び式 (1.1) の x に x_1 を代入すると，$x_2 \longleftarrow x_1$ かつ $\neg P(x_2)$ をみたす元 $x_2 \in X$ を得る．同様にして，帰納的に $P(x)$ をみたさない元の無限下降列 $\cdots \longleftarrow x_2 \longleftarrow x_1 \longleftarrow x_0$ が得られるが，これは $\longleftarrow$ が整礎であるという仮定に反する． □

整礎関係に関する帰納法を用いると，以下の Newman の補題を比較的容易に示すことができる．

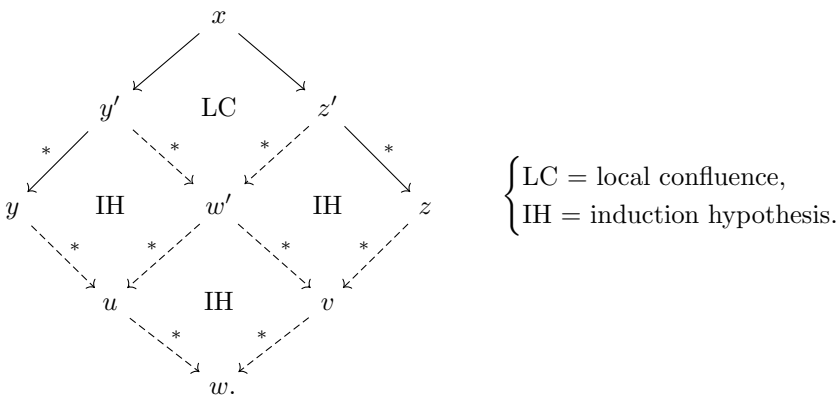
定理 1.7. 集合 X 上の抽象書き換え系 $\longrightarrow$ に対し，以下が成り立つ．

(1) $\longrightarrow$ が強合流的ならば局所合流的である．
(2) $\longrightarrow$ が収束的ならば合流的である．(**Newman の補題**)
(3) $\longrightarrow$ が合流的であることと Church–Rosser 性を持つことは同値である．

証明.

(1) 明らか．

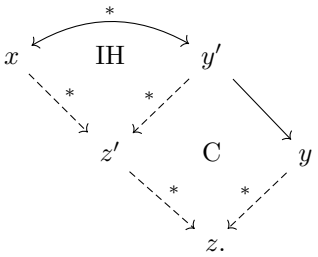
(2) $\longrightarrow$ が逆整礎であることより，その推移閉包 $\overset{+}{\longrightarrow}$ も逆整礎である．逆整礎関係 $\overset{+}{\longrightarrow}$ に関する帰納法 1.6 で示す．任意に $x \in X$ をとる．$x \overset{+}{\longrightarrow} x'$ なる任意の $x' \in X$ に対し合流性が成り立つと仮定する，すなわち $y', z' \in X$ について $y' \overset{*}{\longleftarrow} x' \overset{*}{\longrightarrow} z'$ ならば $y' \overset{*}{\longrightarrow} w' \overset{*}{\longleftarrow} z'$ となる $w' \in X$ が存在すると仮定する (帰納法の仮定)．$y \overset{*}{\longleftarrow} x \overset{*}{\longrightarrow} z$ であるとする．$y = x$ または $z = x$ のときは明らかなので $y \neq x \neq z$ であるとしてよい．よって推移閉包の定義から $y \overset{*}{\longleftarrow} y' \longleftarrow x \longrightarrow z' \overset{*}{\longrightarrow} z$ となる $y', z' \in X$ が存在する．$\longrightarrow$ の局所合流性から $y' \overset{*}{\longrightarrow} w' \overset{*}{\longleftarrow} z'$ となる $w' \in X$ が存在する．$x \longrightarrow y'$ かつ $y \overset{*}{\longleftarrow} y' \overset{*}{\longrightarrow} w'$ だから帰納法の仮定より $y \overset{*}{\longrightarrow} u \overset{*}{\longleftarrow} w'$ となる $u \in X$ が存在する．同様に $x \longrightarrow z'$ かつ $w' \overset{*}{\longleftarrow} z' \overset{*}{\longrightarrow} z$ だから帰納法の仮定より $w' \overset{*}{\longrightarrow} v \overset{*}{\longleftarrow} z$ となる $v \in X$ が存在する．最後に，$x \overset{+}{\longrightarrow} w'$ かつ $u \overset{*}{\longleftarrow} w' \overset{*}{\longrightarrow} v$ だから帰納法の仮定より $u \overset{*}{\longrightarrow} w \overset{*}{\longleftarrow} v$ となる $w \in X$ が存在する．



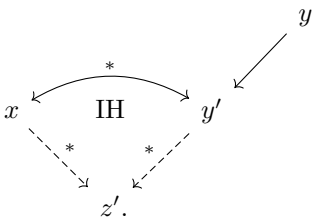
(3) Church–Rosser 性を持つならば合流的であることは明らか．$\longrightarrow$ が合流的であるとする．$x, y \in X$ について $x \overset{*}{\longleftrightarrow} y$ であると仮定し，$x \overset{*}{\longrightarrow} z \overset{*}{\longleftarrow} y$ となる $z \in X$ が存在することを，$x \overset{*}{\longleftrightarrow} y$ の導出の長さ，すなわち $x \overset{n}{\longleftrightarrow} y$ となる n に関する帰納法で示す．長さが 0 のとき，すなわち $x = y$ のときは明らか．$x \neq y$ とし，$x \overset{*}{\longleftrightarrow} y' \longleftrightarrow y$ であるとする．帰納法の仮定より，$x \overset{*}{\longrightarrow} z' \overset{*}{\longleftarrow} y'$ となる $z' \in X$ が存在する．

- $y' \longrightarrow y$ のとき，$z' \overset{*}{\longleftarrow} y' \longrightarrow y$ だから合流性より $z' \overset{*}{\longrightarrow} z \overset{*}{\longleftarrow} y$ となる $z \in X$ が存在し，

$x \xrightarrow{*} z' \xrightarrow{*} z \xleftarrow{*} y$ が成り立つ．

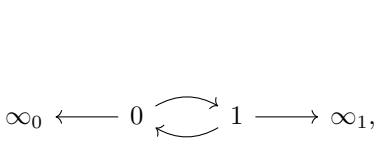


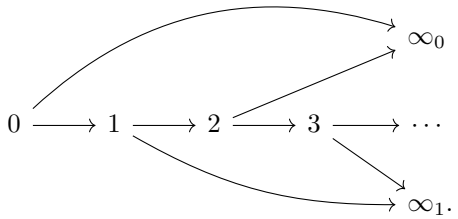
- $y' \longleftarrow y$ のとき，$x \xrightarrow{*} z' \xleftarrow{*} y' \longleftarrow y$ が成り立つ．



□

注意 1.8.　抽象書き換え系は局所合流的であっても合流的でないことがありえる．実際，以下の 2 つの図は局所合流的だが合流的でない書き換え系の例になっている．





実際，これらの書き換え系は停止性を持たない．

定義 1.9 (正規形).　$\longrightarrow$ を集合 X 上の抽象書き換え系とする．$x \in X$ が $\longrightarrow$ に関する**正規形** (normal form) あるいは**既約である** (irreducible) とは，$x \longrightarrow y$ をみたす $y \in X$ が存在しないことである．$\longrightarrow$ に関する正規形全体の集合を $\mathrm{IRR}(\longrightarrow)$ で表す．

以下の定理は正規形を考える動機の一つである．

定理 1.10.　$\longrightarrow$ を集合 X 上の収束的な抽象書き換え系とするとき，正規形の全体 $\mathrm{IRR}(\longrightarrow) \subseteq X$ は商集合 $X/\xleftrightarrow{*}$ の完全代表系を与える．

証明. 元 $x \in X$ が代表する同値類 $[x] \in X/\xleftrightarrow{*}$ をとる．$[x] \cap \mathrm{IRR}(\longrightarrow)$ が一点集合であることを示せばよい．仮に $[x] \cap \mathrm{IRR}(\longrightarrow) = \emptyset$ であるとすると，$\mathrm{IRR}(\longrightarrow)$ に属さない元の無限下降列 $x \longrightarrow x_1 \longrightarrow x_2 \longrightarrow \cdots$ ができてしまい，$\longrightarrow$ が停止性を持つことに反する．よって $[x] \cap \mathrm{IRR}(\longrightarrow) \neq \emptyset$ である．任意に 2 元 $y, z \in [x] \cap \mathrm{IRR}(\longrightarrow)$ をとると，$y \xleftrightarrow{*} z$ である．$\longrightarrow$ は収束的なので定理 1.7 により特に Church–Rosser 性を持ち，よって $y \xrightarrow{*} w \xleftarrow{*} z$ となる $w \in X$ が存在する．ここで $y, z \in \mathrm{IRR}(\longrightarrow)$ は正規形だから $y = w = z$ でなければならない．□

1.3 文字列書き換え系

文字列書き換え系とは，文字列集合上の抽象書き換え系のことである．まず，文字列を定義することから始めよう．

定義 1.11 (文字列). Σ を集合とする (大抵は有限集合を考える)．Σ 上の**文字列** (string) とは，集合 $\Sigma^* := \bigcup_{n\geq 0}\Sigma^n$ の元のことをいう．すなわち，文字列とは Σ の元の有限列のことである．文字列 $w \in \Sigma^n$ の長さ n を記号 $|w|$ で表す．長さ 0 の文字列が唯一つ存在するので，それを特別に**空文字列** (empty string) と呼び ε で表す．集合 Σ を**アルファベット** (alphabet) と呼ぶ[*3]．2 つの文字列 $u \in \Sigma^m, v \in \Sigma^n$ について，u の直後に v を並べてできる長さ $m+n$ の文字列を uv で表し，これを u と v の**連接** (concatenation) という．Σ^* は連接により ε を単位元とするモノイドをなす．文字列 $u, v \in \Sigma^*$ について，ある文字列 $w \in \Sigma^*$ が存在して $uw = v$ となるとき，u は v の**接頭辞** (prefix) であるといい，記号 $u \preccurlyeq v$ で表す．$u \preccurlyeq v$ かつ $u \neq v$ であることを $u \prec v$ で表す．

定義 1.12 (文字列書き換え系). Σ を有限アルファベットとする．Σ^* 上の抽象書き換え系を $R \subseteq \Sigma^* \times \Sigma^*$ を**文字列書き換え系** (string rewriting system) または**半 Thue 系** (semi-Thue system) と呼ぶ．文字列書き換え系 $R \subseteq \Sigma^* \times \Sigma^*$ に対し，新たな抽象書き換え系 $\underset{R}{\Longrightarrow} \subseteq \Sigma^* \times \Sigma^*$ を

$$\underset{R}{\Longrightarrow} := \{\, (xuy, xvy) \in \Sigma^* \times \Sigma^* \mid x, y, u, v \in \Sigma^*, (u, v) \in R \,\}$$

で定義する．抽象書き換え系の性質 P に対し，「$\underset{R}{\Longrightarrow}$ が P をみたす」と言う代わりに，「R が P をみたす」と言うことにする．

感覚的には，各元 $(u, v) \in R$ は「u を v に書き換えてもよい」という規則を表しており，$u \underset{R}{\Longrightarrow} v$ は「u の部分文字列を R のある規則に従って書き換えることで v が得られる」ことを表す．

停止性を導く十分条件として，次の性質がある．

定義 1.13. Σ 上の文字列書き換え系 R が**短縮的である**[*4](length-reducing, length-decreasing) とは，$x, y \in \Sigma^*$ について $x \underset{R}{\Longrightarrow} y$ ならば $|x| > |y|$ となることをいう．

次の命題は文字列の長さが自然数であることから直ちにわかる．

命題 1.14. 文字列書き換え系 R が短縮的ならば停止性を持つ．

[*3] Σ の元ではなく，あくまで集合 Σ のことをアルファベットと呼ぶことに注意する．
[*4] 短縮的，という訳語は本稿だけのものである (あまり良い訳とも思えないが)．

第 2 章

形式言語理論入門

本章では，Muller–Schupp の定理 0.1 の主張を理解するために必要な，形式言語理論の基礎・基本について解説する．形式言語理論はその名の通り言語学とも関係が深い分野であるが，こんにちではむしろ計算機科学や数学の一分野であるとみなされることが多い (と思う)．形式言語理論は計算の理論と切っても切れない関係にあり，なによりプログラミング言語は形式言語の最たる例である．形式言語理論について，より詳しく知りたい読者は例えば計算理論の基本的な教科書 [Sip08, HMU03, Koz97] などを参照のこと．

形式言語理論 (formal language theory) とは「形式的な言語理論」というよりはむしろ「形式言語の理論」である (少なくとも数学においては)．数学的には，形式言語というのは単に文字列の集合のことである．

定義 2.1 (言語). Σ を有限アルファベットとする．Σ 上の文字列全体の集合 Σ^* の部分集合 $L \subseteq \Sigma^*$ のことを Σ 上の**形式言語** (formal language) または単に**言語** (language) と呼ぶ．

とはいえ，すべての部分集合 $L \subseteq \Sigma^*$ に興味があるわけではない．そもそも $\Sigma \neq \emptyset$ なら Σ 上の言語は非可算無限個存在するが，その中で有限の長さの定義を持つ言語は可算個しかない．そこで，言語の中でも有限の長さの “よい” 定義を持つ言語を扱いたい．これは例えるならば，すべての複素数を考える代わりに，有理数や代数的数といった対象に注目することに似ている[*1]．

形式言語を有限の情報で定義する方法の中で，代表的なものが二つある．一つは**文法** (grammar) による定義であり，開始記号から文法規則に沿って**生成** (generate) される文字列を集めることにより言語を定める．もう一つは**オートマトン** (automaton) による定義であり，これは与えられた文字列が言語に属すか否かを判定する有限状態機械を構成するものである．

本稿では数ある言語クラスの中でも，特に重要な**正則言語**[*2](regular language, 正規言語とも) と**文脈自由言語** (context-free langauge) を取り上げる．正則言語は**正則文法** (regular grammar, 正規文法とも) によって定義される言語であり，**有限オートマトン** (finite automaton) という機械によって認識される言語としても特徴付けられる．正則言語はあまり高い表現力を持つわけではないものの，シンプルで扱いやすく，性質も非常によいので入門に最適である．文脈自由言語は**文脈自由文法** (context-free grammar) または**プッシュダウンオートマトン** (pushdown automaton) によって定義される言語であり，正則言語よりも高い表現力を持ち，ある程度簡単な自然言語や，プログラミング言語の文法を記述するとができる．本稿の主題である Muller–Schupp の定理 0.1 は，群の語の問題が文脈自由言語であるための必要十分条件を与える定理である．

[*1] これは単なる比喩ではなく，実際に正則言語や無曖昧文脈自由言語の母関数が有理関数や代数関数になることが知られている．詳しくは新屋 [新屋良 17] などを参照されたい．

[*2] regular language はふつう正規言語と訳されることが多いが，本稿では数学の慣習に合わせて regular に正則という訳を当てることにする．実際，[HMU03] では regular に正則という訳があてられている．

2.1 文法による言語の定義

まずは文法を用いた言語の定義から始めよう．形式文法は言語学者の Noam Chomsky が自然言語の構造を研究するために提案した**生成文法** (generative grammar) に由来する，自然言語の数学的モデルである [風上松町 04]．とはいえ，以降の話は完全に数学的なものなので，自然言語のことは忘れてしまって構わない．

定義 2.2 (形式文法). **形式文法** (formal grammar)，あるいは単に**文法** (grammar) とは以下のデータからなる 5 つ組 $G = (V, \Sigma, P, S)$ である．

- **変数** (variable)[*3]の有限集合 V，
- **終端記号** (terminal symbol) の有限集合 Σ，ただし $\Sigma \cap V = \emptyset$，
- **生成規則** (production rule) の有限集合 $P \subseteq (V \cup \Sigma)^* \times (V \cup \Sigma)^*$，すなわち P は $(V \cup \Sigma)^*$ 上の文字列書き換え系，
- **開始記号** (start symbol) $S \in V$．

生成規則 $(\alpha, \beta) \in P$ のことを $\alpha \to \beta$ と書くことがある．さらに，第 1 成分が同じ複数の生成規則 $(\alpha, \beta_1), (\alpha, \beta_2), \ldots, (\alpha, \beta_n) \in P$ をまとめて $\alpha \to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$ と書くことがある．Σ 上の言語

$$\mathrm{L}(G) := \{\, w \in \Sigma^* \mid S \overset{*}{\underset{P}{\Longrightarrow}} w \,\}$$

のことを文法 G が**生成する** (generate) 言語という．

形式文法は非常に高い表現力を持つ反面，非常に複雑になりうるので分析が容易ではない．そこで，扱いやすいように文法の定義に制限を加えることを考える．

定義 2.3 (i 型文法, Chomsky 階層). $G = (V, \Sigma, P, S)$ を文法とする．

- G が **0 型文法** (type-0 grammar) または**句構造文法** (phrase structure grammar) であるとは，G が文法であること (つまり無条件) をいう．0 型文法 G の生成する言語 $\mathrm{L}(G)$ のことを**計算可枚挙言語** (computably enumerable language) または**再帰的可枚挙言語** (recursively enumerable language) と呼ぶ．
- G が **1 型文法** (type-1 grammar) または**文脈依存文法** (context-sensitive grammar)，あるいは**単調文法** (monotone grammar) であるとは，各生成規則 $\alpha \to \beta \in P$ が $|\alpha| \leq |\beta|$ をみたすことをいう．1 型文法 G の生成する言語 $\mathrm{L}(G)$ のことを**文脈依存言語** (context-sensitive language) と呼ぶ．
- G が **2 型文法** (type-2 grammar) または**文脈自由文法** (context-free grammar) であるとは，各生成規則 $\alpha \to \beta \in P$ が $\alpha \in V$ をみたすことをいう．2 型文法 G の生成する言語 $\mathrm{L}(G)$ のことを**文脈自由言語** (context-free language) と呼ぶ．
- G が **3 型文法** (type-3 grammar) または**正則文法** (regular grammar, 正規文法とも) であるとは，各生成規則 $\alpha \to \beta \in P$ が $\alpha \in V$ かつ $\beta = \varepsilon \vee \exists w \in \Sigma^*\ \exists A \in V\ [\beta = wA]$ をみたすことをいう．3 型文法 G の生成する言語 $\mathrm{L}(G)$ のことを**正則言語** (regular language, 正規言語とも) と呼ぶ．

明らかに，G が $(i+1)$ 型文法 $(i = 0, 1, 2)$ であれば G は i 型文法でもある．

先程は自然言語のことは忘れてしまって構わないと言ったが，最初くらいは自然言語 (風) の例を見ておくこととしよう．

[*3] V の元は**非終端記号** (non-terminal symbol) とも呼ばれる．

例 2.4.　文脈自由文法 $G = (V, \Sigma, P, S)$ を以下のように定義する．

$$
\begin{aligned}
V &:= \{[\text{文}], [\text{名詞}], [\text{格助詞}], [\text{形容詞}], [\text{副詞}]\},\\
\Sigma &:= \{\ \text{箱}, \text{は}, \text{赤い}, \text{青い}, \text{大きい}, \text{小さい}, \text{とても}, \text{の隣にある}, \text{の中にある}\ \},\\
P &:= \begin{cases}
[\text{文}] \to [\text{名詞}][\text{格助詞}][\text{形容詞}],\\
[\text{名詞}] \to [\text{形容詞}][\text{名詞}] \mid [\text{名詞}]\ \text{の隣にある}\ [\text{名詞}] \mid [\text{名詞}]\ \text{の中にある}\ [\text{名詞}] \mid \text{箱},\\
[\text{格助詞}] \to \text{は},\\
[\text{形容詞}] \to [\text{副詞}][\text{形容詞}] \mid \text{赤い} \mid \text{青い} \mid \text{大きい} \mid \text{小さい},\\
[\text{副詞}] \to \text{とても},
\end{cases}\\
S &:= [\text{文}].
\end{aligned}
$$

このとき，例えば次のような導出がある．

$$
\begin{aligned}
[\text{文}] &\underset{P}{\Longrightarrow} [\text{名詞}][\text{格助詞}][\text{形容詞}] \underset{P}{\Longrightarrow} [\text{名詞}]\ \text{は}\ [\text{形容詞}] \underset{P}{\Longrightarrow} [\text{名詞}]\ \text{の中にある}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} [\text{名詞}]\ \text{の隣にある}\ [\text{名詞}]\ \text{の中にある}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} [\text{形容詞}][\text{名詞}]\ \text{の隣にある}\ [\text{名詞}]\ \text{の中にある}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} [\text{副詞}][\text{形容詞}][\text{名詞}]\ \text{の隣にある}\ [\text{名詞}]\ \text{の中にある}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても}\ [\text{形容詞}][\text{名詞}]\ \text{の隣にある}\ [\text{名詞}]\ \text{の中にある}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい}\ [\text{名詞}]\ \text{の隣にある}\ [\text{名詞}]\ \text{の中にある}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい}\ [\text{形容詞}][\text{名詞}]\ \text{の隣にある}\ [\text{名詞}]\ \text{の中にある}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い}\ [\text{名詞}]\ \text{の隣にある}\ [\text{名詞}]\ \text{の中にある}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い箱の隣にある}\ [\text{名詞}]\ \text{の中にある}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い箱の隣にある}\ [\text{形容詞}][\text{名詞}]\ \text{の中にある}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い箱の隣にある青い}\ [\text{名詞}]\ \text{の中にある}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い箱の隣にある青い箱の中にある}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い箱の隣にある青い箱の中にある}\ [\text{形容詞}][\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い箱の隣にある青い箱の中にある小さい}\ [\text{名詞}]\ \text{は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い箱の隣にある青い箱の中にある小さい箱は}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い箱の隣にある青い箱の中にある小さい箱は}\ [\text{副詞}][\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い箱の隣にある青い箱の中にある小さい箱はとても}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い箱の隣にある青い箱の中にある小さい箱はとても}\ [\text{副詞}][\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い箱の隣にある青い箱の中にある小さい箱はとてもとても}\ [\text{形容詞}]\\
&\underset{P}{\Longrightarrow} \text{とても大きい赤い箱の隣にある青い箱の中にある小さい箱はとてもとても赤い} \in \mathrm{L}(G).
\end{aligned}
\tag{2.1}
$$

文脈自由文法の場合には，導出に対応する**構文木** (syntax tree) (正確な定義は 2.3.1 節で行う) を次の図 2.1 のように描くことができる．

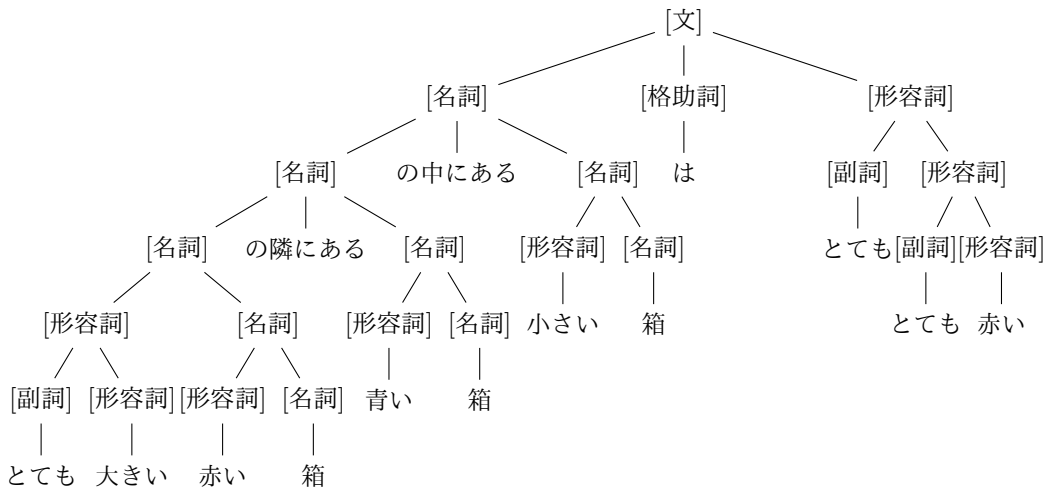


図 2.1　導出 (2.1) に対応する構文木

2.2　正則言語 (正規言語)

形式言語に慣れ親しむために，まずは扱いが比較的簡単な正則言語から始めよう．正則文法の表現力はそれほど高いわけではないものの，正則言語には特出した数学的性質のよさがある．その証拠に，正則言語は多種多様な特徴付けを持っている．数学において，よい概念は多くの特徴付けを持つ．本稿ではそのすべてを紹介することはできないが，特に重要な有限オートマトン，正則表現，統語モノイドによる特徴付けについては証明を行う (定理 2.22)．定義 2.3 では正則言語は正則文法により生成される言語としたが，他の文献では同値な他の定義が採用されていることが多い．正則言語についてさらに詳しく知りたい読者は，新屋によるサーベイ論文 [新屋良 17] や Sakarovitch [Sak09]，Lawson [Law04] などの教科書を参照されたい．

2.2.1　有限オートマトン

有限オートマトンとは，固定された有限の記憶領域のみを持つ，非常に単純な計算モデルのひとつである．有限オートマトンは与えられた文字列が言語に属すかどうかを，次のように判定する．文字列を左から 1 文字ずつ順番に読んでいき，現在の内部状態と読み込んだ文字に応じて内部状態を変化させることができる．文字列を読み終わった段階で内部状態があらかじめ指定された状態のいずれかになっていれば，文字列は受理され，言語に属していると判定される．

定義 2.5 (有限オートマトン)． **非決定性有限オートマトン** (non-deterministic finite automaton; NFA) とは，以下のデータからなる 5 つ組 $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ のことである．

- **状態** (state) の有限集合 Q,
- 有限アルファベット Σ,
- **遷移関係** (transition relation) $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$,
- **開始状態** (start state) $q_0 \in Q$,
- **受理状態** (accept state, final state) の集合 $F \subseteq Q$.

NFA $\mathcal{A}$ が**決定性有限オートマトン** (deterministic finite automaton; DFA) であるとは，遷移関係 δ が $Q \times \Sigma$ から Q への関数であること，すなわち

- $\forall (q, a) \in Q \times \Sigma\ \exists! r \in Q\ [(q, a, r) \in \delta]$,

- $\neg\exists q, r \in Q\ [(q, \varepsilon, r) \in \delta]$

をみたすことをいう．NFA $\mathcal{A}$ が文字列 $w \in \Sigma^*$ を**受理する** (accept) とは，以下の条件をみたす状態の有限列 $r_0, r_1, \ldots, r_m \in Q$ $(m \geq 0)$ と文字または ε の有限列 $a_1, a_2, \ldots, a_m \in \Sigma \cup \{\varepsilon\}$ が存在することをいう．

- $a_1 a_2 \cdots a_m = w$,
- $r_0 = q_0$,
- $i = 0, \ldots, m-1$ に対し $(r_i, a_{i+1}, r_{i+1}) \in \delta$,
- $r_m \in F$.

$\mathcal{A}$ が DFA のときは，$\mathcal{A}$ が文字列 $w = a_1 a_2 \cdots a_n \in \Sigma^*$ $(a_i \in \Sigma)$ を受理することは，帰納的に $r_0 := q_0, r_{i+1} := \delta(r_i, a_{i+1})$ $(0 \leq i \leq n-1)$ と定義したときに $r_n \in F$ となることと言い換えられる．Σ 上の言語

$$\mathrm{L}(\mathcal{A}) := \{\, w \in \Sigma^* \mid \mathcal{A} \text{ は } w \text{ を受理する} \,\}$$

のことを $\mathcal{A}$ が**認識する** (recognize) 言語という．Σ 上の言語 $L \subseteq \Sigma^*$ が $L = \mathrm{L}(\mathcal{A})$ をみたすとき，L は $\mathcal{A}$ によって**認識される** (recognized) という．

NFA の遷移関係 δ の各元 $(q, a, r) \in \delta$ のことを**遷移** (transition) という．遷移 $(q, a, r) \in \delta$ は「現在の状態が q であるとき，文字 (または ε) a を読み込んで状態を r に変化させてよい」ということを意味する．文字を読まずに状態を変化させる，$(q, \varepsilon, r) \in \delta$ の形の遷移を ε **遷移** (ε-transition) という．同じ文字を読み込んだ際に (あるいは読み込まずに) 遷移する先の状態に複数の可能性があることが，**非決定性** (non-deterministic) という形容詞が付いている理由である．

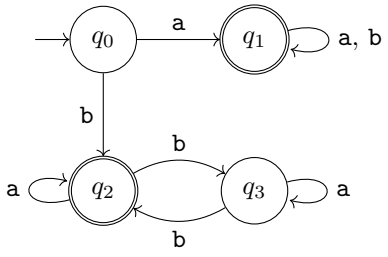
一方，DFA の場合には，遷移 $(q, a, r) \in \delta$ は関数として $\delta(q, a) = r$ という関係が成り立つことを意味し，すなわち「現在の状態が q であるとき，文字 a を読み込んだら次の状態が r に変化する」ということを意味する．**決定性** (deterministic) という形容詞はもちろん，文字を読み込んだ際に遷移する先が一意に決まることを表している．

有限オートマトンを図で表す便利な方法に**状態遷移図** (state diagram) というものがある．状態遷移図の形式的な定義をだらだらと述べるよりも，以下の例を見て納得する方が手っ取り早いだろう．

例 2.6.　NFA $\mathcal{A}_1 = (Q, \Sigma, \delta, q_0, F)$ を

$$\begin{aligned} Q &:= \{q_0, q_1, q_2, q_3\}, \\ \Sigma &:= \{\mathtt{a}, \mathtt{b}\}, \\ \delta &:= \left\{ \begin{matrix} (q_0, \mathtt{a}, q_1), (q_0, \mathtt{b}, q_2), (q_1, \mathtt{a}, q_1), (q_1, \mathtt{b}, q_1), \\ (q_2, \mathtt{a}, q_2), (q_2, \mathtt{b}, q_3), (q_3, \mathtt{a}, q_3), (q_3, \mathtt{b}, q_2) \end{matrix} \right\}, \\ F &:= \{q_1, q_2\} \end{aligned}$$

と定義する．$\mathcal{A}_1$ は DFA になっている．$\mathcal{A}_1$ の状態遷移図は次のようになる．



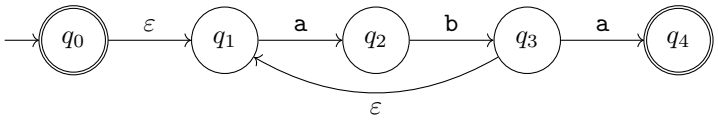
すなわち，NFA の状態遷移図は次のような規則で描かれるラベル付きグラフである．

- 各々の状態をグラフの頂点とし，丸で囲む．
- 開始状態 q_0 を示す矢印を書く．
- 受理状態は二重丸にする．
- 各遷移 $(q, a, r) \in \delta$ に対し，q から r へ向かって a でラベル付けされた辺を引く．

$\mathcal{A}_1$ が認識する言語は $\mathrm{L}(\mathcal{A}_1) = \{\, w \in \Sigma^* \mid w \text{ は } \mathtt{a} \text{ で始まるか，または } \mathtt{b} \text{ を奇数個含む} \,\}$ である．
次に，NFA $\mathcal{A}_2 = (Q, \Sigma, \delta, q_0, F)$ を

$$
\begin{aligned}
Q &:= \{q_0, q_1, q_2, q_3, q_4\}, \\
\Sigma &:= \{\mathtt{a}, \mathtt{b}\}, \\
\delta &:= \{(q_0, \varepsilon, q_1), (q_1, \mathtt{a}, q_2), (q_2, \mathtt{b}, q_3), (q_3, \varepsilon, q_1), (q_3, \mathtt{a}, q_4)\}, \\
F &:= \{q_0, q_4\}
\end{aligned}
$$

と定義する．$\mathcal{A}_2$ は DFA ではない NFA である．$\mathcal{A}_2$ の状態遷移図は次のようになる．



$\mathcal{A}_2$ が認識する言語は $\mathrm{L}(\mathcal{A}_2) = \{\varepsilon\} \cup \{\, w\mathtt{a} \in \Sigma^* \mid w \text{ は } \mathtt{ab} \text{ の 1 回以上の繰り返し} \,\}$ である．

次の定理は，NFA が正則文法と同等の表現力を持つことを主張するものである．すなわち，正則言語とはある NFA で認識される言語のことである．

定理 2.7. Σ 上の言語 $L \subseteq \Sigma^*$ に対し，以下の条件は同値である．

(1) L はある正則文法によって生成される．
(2) L はある NFA によって認識される．

証明. (1) $\Longrightarrow$ (2). L を生成する正則文法 $G = (V, \Sigma, P, S)$ をとる．このとき，NFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ を

$$
\begin{aligned}
Q &:= V \cup \bigcup \{\, \{\, a_j a_{j+1} \cdots a_n B \mid 2 \le j \le n \,\} \mid A, B \in V, a_i \in \Sigma, A \to a_1 a_2 \cdots a_n B \in P \,\}, \\
\delta &:= \{\, (A, \varepsilon, B) \mid A \to B \in P \,\} \cup \{\, (A, a, wB) \mid A, B \in V, a \in \Sigma, w \in \Sigma^*, A \to awB \in P \,\} \\
&\quad \cup \{\, (awB, a, wB) \mid a \in \Sigma, w \in \Sigma^*, B \in V, awB, wB \in Q \,\}, \\
q_0 &:= S, \\
F &:= \{\, A \in V \mid A \to \varepsilon \in P \,\}.
\end{aligned}
$$

と定義すれば $\mathrm{L}(\mathcal{A}) = \mathrm{L}(G) = L$ となる (下の例 2.8 も参照のこと).

(2) $\Longrightarrow$ (1). L を認識する NFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ をとる．このとき，正則文法 $G = (V, \Sigma, P, S)$ を

$$
\begin{aligned}
V &:= Q, \\
P &:= \{\, q \to ar \mid (q, a, r) \in \delta \,\} \cup \{\, q \to \varepsilon \mid q \in F \,\}, \\
S &:= q_0
\end{aligned}
$$

と定義すれば $\mathrm{L}(G) = \mathrm{L}(\mathcal{A}) = L$ となる． □

例 2.8. 正則文法 $G = (V, \Sigma, P, S)$ を

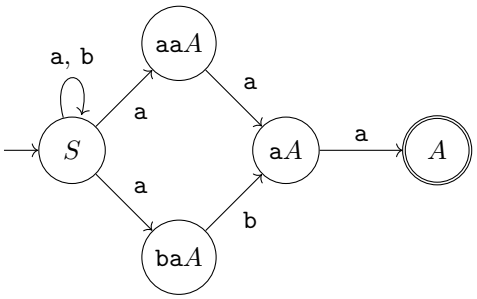
$$V := \{S, A\},$$

$$\Sigma := \{\mathtt{a}, \mathtt{b}\},$$
$$P := \begin{cases} S \to \mathtt{a}S \mid \mathtt{b}S \mid \mathtt{aaa}A \mid \mathtt{aba}A, \\ A \to \varepsilon \end{cases}$$

と定義する．このとき

$$\begin{aligned} \mathrm{L}(G) &= \{\, w\mathtt{aaa} \mid w \in \Sigma^* \,\} \cup \{\, w\mathtt{aba} \mid w \in \Sigma^* \,\} \\ &= \{\, w \in \Sigma^* \mid w \text{ の一番最後の文字と後ろから 3 番目の文字が } \mathtt{a} \,\} \end{aligned}$$

である．この G に対応する NFA $\mathcal{A}$ を定理 2.7 の証明に従って構成すると以下のようになる．



DFA は定義より明らかに NFA であるが，逆は一般には成り立たない．では，NFA は DFA よりも真に高い表現力を持っているだろうか？ 言い換えると，ある NFA によって認識することができるが，いかなる DFA によっても認識されないような，そんな言語が存在するだろうか？ 実はそのような言語は存在しない．すなわち，NFA を DFA に制限することは表現力をまったく損なわないのである．

定理 2.9 (Rabin–Scott の冪集合構成 (powerset construction)). Σ 上の言語 $L \subseteq \Sigma^*$ に対し，以下の 2 条件は同値である．

(1) L はある DFA により認識される．
(2) L はある NFA により認識される．

証明のアイデアは定理のインパクトに反してとても単純である: 文字列を手前から読んでいったとき，ある時点で「いる」ことができる状態の全体をそのときの状態とすればよい．

証明. (1) $\Longrightarrow$ (2) は定義より明らか．(2) $\Longrightarrow$ (1) を示す．$L = \mathrm{L}(\mathcal{A})$ となる NFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ をとる．$\mathcal{A}$ の状態の集合 $R \subseteq Q$ に対し，次のように記号を定義する．

$$\begin{aligned} E_0(R) &:= R, \\ E_{n+1}(R) &:= \{\, r \in Q \mid q \in E_n(R), (q, \varepsilon, r) \in \delta \,\} \\ &= \{\, q \in Q \mid q \text{ は } R \text{ のある元から } n+1 \text{ 回以下の } \varepsilon \text{ 遷移で到達できる} \,\}, \\ E(R) &:= \bigcup_{n=0}^{\infty} E_n(R) \\ &= \{\, q \in Q \mid q \text{ は } R \text{ のある元から } 0 \text{ 回以上の } \varepsilon \text{ 遷移で到達できる} \,\}. \end{aligned}$$

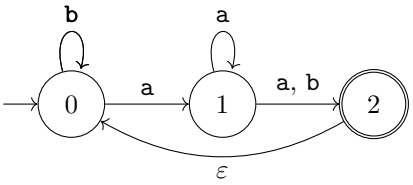
実際には $E(R)$ は有限集合なので，無限個の $E_n(R)$ を計算する必要はないことに注意する．このとき，DFA $\mathcal{B} = (Q', \Sigma, \delta', q_0', F')$ を

$$Q' := \mathcal{P}(Q) \quad (Q \text{ の冪集合}),$$

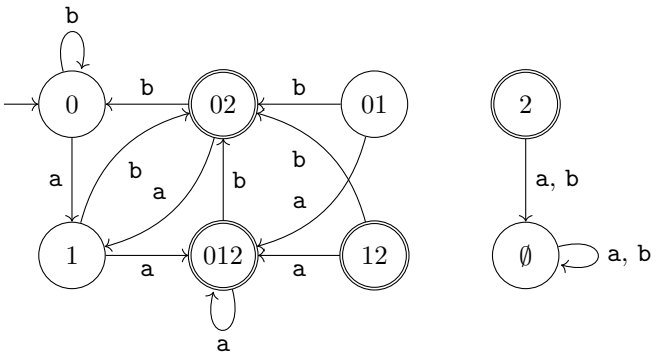
$$\begin{aligned}\delta'(R,a) &:= \bigcup_{r\in R} E(\{\, s\in Q \mid (r,a,s)\in\delta \,\}),\\ q_0' &:= E(\{q_0\}),\\ F' &:= \{\, R\subseteq Q \mid R\cap F\neq\emptyset \,\}\end{aligned}$$

と定義すると $\mathrm{L}(\mathcal{B}) = \mathrm{L}(\mathcal{A}) = L$ が成り立つ (下の例 2.10 も参照のこと). □

例 2.10. アルファベット $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ を固定する．NFA $\mathcal{A}$ を以下の状態遷移図で定義されるものとする．



このとき，$\mathrm{L}(\mathcal{A}) = \{\, w\in\Sigma^* \mid w$ の後ろから 2 文字目が $\mathtt{a}\,\}$ であり，対応する DFA $\mathcal{B}$ は次のようになる．



冪集合構成 2.9 から特に次の決定可能性が得られる．

系 2.11 (正則言語の所属判定問題の決定可能性). 与えられた正則言語 $L\subseteq\Sigma^*$ と文字列 $w\in\Sigma^*$ が $w\in L$ をみたすかどうかを判定するアルゴリズムを構成できる．ただし，正則言語は正則文法や NFA などの形で与えられるものとする．

証明. 定理 2.9 に基づいて L を認識する DFA $\mathcal{A}$ を構成し，与えられた文字列 $w\in\Sigma^*$ が $\mathcal{A}$ に受理されるかどうかを確かめればよい. □

2.2.2 正則表現 (正規表現) と有理的部分集合

有限オートマトンを用いた正則言語の定義は直観的でわかりやすいが，込み入った定義や状態遷移図を毎回描くのは面倒である．正則文法を使えば多少楽になるものの，定義が一行に収まることは少ない．正則言語をよりコンパクトに表現するために，正則表現と呼ばれる表記法を導入する．

定義 2.12 (正則表現). アルファベット Σ 上の**正則表現** (regular expression, 正規表現とも) は，以下のように帰納的に定義される項のことである．

(1) 空集合 $\emptyset$ は正則表現である．
(2) 各文字 $a\in\Sigma$ に対し，a は正則表現である．

(3) 空文字列 ε は正則表現である．
(4) E_1, E_2 が正則表現なら，$(E_1 \cup E_2)$ も正則表現である．(**和**)
(5) E_1, E_2 が正則表現なら，$(E_1 \cdot E_2)$ も正則表現である．(**連接**)
(6) E が正則表現なら，(E^*) も正則表現である．(**Kleene 閉包**)
(7) 以上により定義されるものだけが正則表現である．

正則表現の括弧は曖昧さを生じない範囲でなるべく省略することにする．また，連接 $E_1 \cdot E_2$ の $\cdot$ を省略して E_1E_2 とも書く．よって例えば $(\mathtt{a} \cdot (\mathtt{b} \cdot (\mathtt{b}^*)))$ と書く代わりに $\mathtt{a}(\mathtt{bb}^*)$ のように書く．正則表現 E が定める言語 $\mathrm{L}(E)$ を以下のように帰納的に定義する．

(1) $\mathrm{L}(\emptyset) := \emptyset$.
(2) $a \in \Sigma$ に対し $\mathrm{L}(a) := \{a\}$.
(3) $\mathrm{L}(\varepsilon) := \{\varepsilon\}$.
(4) E_1, E_2 が正則表現のとき，$\mathrm{L}(E_1 \cup E_2) := \mathrm{L}(E_1) \cup \mathrm{L}(E_2)$.
(5) E_1, E_2 が正則表現のとき，$\mathrm{L}(E_1 \cdot E_2) := \mathrm{L}(E_1) \cdot \mathrm{L}(E_2) = \{\, uv \mid u \in \mathrm{L}(E_1), v \in \mathrm{L}(E_2) \,\}$.
(6) E が正則表現のとき，$\mathrm{L}(E^*) := \bigcup_{n=0}^{\infty} \mathrm{L}(E)^n = \{\, w_1 w_2 \cdots w_n \mid n \geq 0, \forall i \in \{1, \ldots, n\}\ [w_i \in \mathrm{L}(E)] \,\}$.

言語 $L \subseteq \Sigma^*$ がある正則表現 E によって $L = \mathrm{L}(E)$ と表されるとき，L は Σ^* の**有理的部分集合** (rational subset) であるという[*4].

例 2.13. $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ とする．このとき，

- $\mathrm{L}(\emptyset^*) = \{\varepsilon\}$,
- $\mathrm{L}(\mathtt{aaa}^*) = \{\, \mathtt{a}^n \in \Sigma^* \mid n \geq 2 \,\}$,
- $\mathrm{L}((\mathtt{a} \cup \mathtt{b})^*\mathtt{ab}(\mathtt{a} \cup \mathtt{b})^*) = \{\, w \in \Sigma^* \mid w \text{ は } \mathtt{ab} \text{ を含む} \,\}$,
- $\mathrm{L}(((\mathtt{a} \cup \mathtt{b})(\mathtt{a} \cup \mathtt{b}))^*) = \{\, w \in \Sigma^* \mid |w| \equiv 0 \pmod 2 \,\}$.

命題 2.14. Σ 上の任意の正則表現 E に対し，$\mathrm{L}(E) = \mathrm{L}(G)$ となる正則文法 $G = (V, \Sigma, P, S)$ が存在する．

証明. 正則表現 E の構成に関する帰納法で示す．

(1) $E = \emptyset$ のとき，$V := \{S\}, P := \emptyset$ とおけば $\mathrm{L}(G) = \emptyset = \mathrm{L}(E)$ となる．
(2) $E = a \in \Sigma$ のとき，$V := \{S\}, P := \{S \to a\}$ とおけば $\mathrm{L}(G) = \{a\} = \mathrm{L}(E)$ となる．
(3) $E = \varepsilon$ のとき，$V := \{S\}, P := \{S \to \varepsilon\}$ とおけば $\mathrm{L}(G) = \{\varepsilon\} = \mathrm{L}(E)$ となる．
(4) $E = E_1 \cup E_2$ のとき，帰納法の仮定から $\mathrm{L}(G_1) = \mathrm{L}(E_1)$，$\mathrm{L}(G_2) = \mathrm{L}(E_2)$ となる文法 $G_1 = (V_1, \Sigma, P_1, S_1)$，$G_2 = (V_2, \Sigma, P_2, S_2)$ が存在する．このとき，文法 $G = (V, \Sigma, P, S)$ を $V = V_1 \cup V_2 \cup \{S\}, P := P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}$ とおけば $\mathrm{L}(G) = \mathrm{L}(G_1) \cup \mathrm{L}(G_2) = \mathrm{L}(E_1) \cup \mathrm{L}(E_2) = \mathrm{L}(E_1 \cup E_2)$ となる．
(5) $E = E_1 \cdot E_2$ のとき，帰納法の仮定から $\mathrm{L}(G_1) = \mathrm{L}(E_1)$，$\mathrm{L}(G_2) = \mathrm{L}(E_2)$ となる文法 $G_1 = (V_1, \Sigma, P_1, S_1)$，$G_2 = (V_2, \Sigma, P_2, S_2)$ が存在する．このとき，文法 $G = (V, \Sigma, P, S)$ を $V = V_1 \cup V_2 \cup \{S\}, P := P_1 \cup P_2 \cup \{S \to S_1S_2\}$ とおけば $\mathrm{L}(G) = \mathrm{L}(G_1) \cup \mathrm{L}(G_2) = \mathrm{L}(E_1) \cup \mathrm{L}(E_2) = \mathrm{L}(E_1 \cup E_2)$ となる．
(6) $E = E_0^*$ のとき，帰納法の仮定から $\mathrm{L}(G_0) = \mathrm{L}(E_0)$ となる文法 $G_0 = (V_0, \Sigma, P_0, S_0)$ が存在する．このとき，文法 $G = (V, \Sigma, P, S)$ を $V = V_0 \cup \{S\}, P := P_0 \cup \{S \to \varepsilon \mid S_0S\}$ とおけば $\mathrm{L}(G) = \mathrm{L}(E_0^*)$ となる． □

[*4] 一般には自由モノイド Σ^* に限らないモノイドの有理的部分集合も考えることができる．(付録で扱うかもしれない．)

次の定理は正則表現がすべての正則言語を表すのに十分なだけの表現力を持っていることを主張するものである．2.4.1 節で見るように，正則言語のクラスは補集合をとる操作や共通部分をとる操作で閉じているので，この定理から特に正則表現で表される言語の補集合や共通部分を表す正則表現が存在することになるが，このことは正則表現の定義からはまったく明らかではない．

定理 2.15 (Kleene). どんな NFA $\mathcal{A}$ に対しても，$\mathrm{L}(E) = \mathrm{L}(\mathcal{A})$ となる正則表現 E が存在する．

証明. **動的計画法** (dynamic programming) の考え方を利用して証明する[*5]．NFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ をとる．状態に番号を付けて $Q = \{q_0, q_1, \ldots, q_n\}$ とする．以下，$i, j \in \{0, 1, \ldots, n\}, k \in \{0, 1, \ldots, n+1\}$ とする．$Q_k := \{\, q_l \in Q \mid l < k \,\}$ とおく．このとき $Q_0 = \emptyset$，$Q_{n+1} = Q$ である．以下，正則表現 $E^k_{i,j}$ を，文字列 $w \in \Sigma^*$ に対し以下の 2 条件が同値になるように帰納的に構成していく．

- $w \in \mathrm{L}(E^k_{i,j})$,
- ある状態の有限列 $r_0, r_1, \ldots, r_m \in Q$ $(m \geq 0)$ と $a_1, a_2, \ldots, a_m \in \Sigma \cup \{\varepsilon\}$ が存在して以下の条件をみたす．
  - $a_1 a_2 \cdots a_m = w$,
  - $r_0 = q_i$,
  - $l = 0, \ldots, m-1$ に対し $(r_l, a_{l+1}, r_{l+1}) \in \delta$,
  - $r_m = q_j$,
  - $r_1, \ldots, r_{m-1} \in Q_k$.

すなわち，$\mathrm{L}(E^k_{i,j})$ は「読み込むことで，Q_k の状態だけを通って q_i から q_j へ到達できる文字列」の全体である．$k = 0$ のときは

$$E^0_{i,j} = \begin{cases} \bigcup\{\, a \in \Sigma \mid (q_i, a, q_j) \in \delta \,\} \cup \varepsilon & \text{if } i = j, \\ \bigcup\{\, a \in \Sigma \mid (q_i, a, q_j) \in \delta \,\} & \text{if } i \neq j \end{cases}$$

と書ける．すべての i, j に対し $E^k_{i,j}$ が定義されたとする．$w \in \mathrm{L}(E^{k+1}_{i,j})$ となるとき，対応する経路が途中で q_k を 1 回も通らないのであれば $w \in \mathrm{L}(E^k_{i,j})$ であるし，1 回以上通るのであれば，その回数 m に応じて $w \in \mathrm{L}(E^k_{i,k}(E^k_{k,k})^{m-1}E^k_{k,j})$ となる．よって対応する正則表現は

$$E^{k+1}_{i,j} = E^k_{i,j} \cup E^k_{i,k}(E^k_{k,k})^* E^k_{k,j}$$

となる．最後に，$E := \bigcup_{q_j \in F} E^{n+1}_{0,j}$ とおけば $\mathrm{L}(E) = \mathrm{L}(\mathcal{A})$ を得る． □

2.2.3 認識可能部分集合，統語モノイド，Myhill–Nerode の定理

これまで，正則言語を正則文法，NFA，正則表現といった具体的なデータにより定義する方法を見てきた．実はもう少し抽象的に，正則言語を有限モノイドの言葉を使って定義することができる．

定義 2.16 (認識可能部分集合). Σ 上の言語 $L \subseteq \Sigma^*$ が Σ^* の**認識可能部分集合** (recognizable subset) であるとは，ある有限モノイド N と全射なモノイド準同型写像 $\pi\colon \Sigma^* \twoheadrightarrow N$ が存在して，$L = \pi^{-1}(\pi(L))$ をみたすことをいう[*6]．言い換えると，Σ^* 上の同値関係 $u \sim v \ :\Longleftrightarrow\ \pi(u) = \pi(v)$ について，L が $\sim$ に関するいくつかの同値類の和集合になっているということである．

[*5] 動的計画法の代わりに**一般化 NFA**(generalized NFA; GNFA) を用いて**状態消去法** (state elimination method) により証明することもできるが，実際にやっていることはほとんど同じである．

[*6] 一般にモノイド準同型写像の像は終域の部分モノイドになるので，全射性は本質的な仮定ではない．

定義 2.17 (統語モノイド).　$L \subseteq \Sigma^*$ を Σ 上の言語とする．L の**統語的合同関係** (syntactic congruence) $\equiv_L$ とは，以下のように定義される Σ^* 上の同値関係である．

$$u \equiv_L v \ :\Longleftrightarrow\ \forall x, y \in \Sigma^*\ [xuy \in L \iff xvy \in L].$$

$\equiv_L$ が実際に同値関係の公理をみたすことは簡単に確認できる．L の統語的合同関係による商モノイド

$$\mathrm{Syn}(L) := \Sigma^*/\equiv_L$$

のことを L の**統語モノイド** (syntactic monoid) と呼ぶ[*7]．商集合 $\Sigma^*/\equiv_L$ 上の積が実際に well-defined になることも簡単に確認できる．

例 2.18.　$\Sigma = \{\mathtt{a}, \mathtt{b}\}$ とする．

- $L_1 := \{\, w \in \Sigma^* \mid |w| \equiv 0 \pmod 2 \,\}$ とする．任意の $u, v, x, y \in \Sigma^*$ に対し，

$$\begin{aligned}[xuy \in L_1 \iff xvy \in L_1] &\iff [|xuy| \equiv 0 \pmod 2 \iff |xvy| \equiv 0 \pmod 2] \\ &\iff |xuy| \equiv |xvy| \pmod 2 \\ &\iff |u| \equiv |v| \pmod 2\end{aligned}$$

  となるので，統語的合同関係 $\equiv_{L_1}$ の同値類は文字列の長さの偶奇で決まる．よって $\mathrm{Syn}(L_1) = \{[\varepsilon], [\mathtt{a}]\} \cong \mathbb{Z}/2\mathbb{Z}$ となる．
- 文字列 $w \in \Sigma^*$ に含まれる文字 $a \in \Sigma$ の個数を $|w|_a$ と書くことにする．$L_2 := \{\, w \in \Sigma^* \mid |w|_{\mathtt{a}} = |w|_{\mathtt{b}} \,\}$ とする．任意の $u, v, x, y \in \Sigma^*$ に対し，

$$\begin{aligned}&[xuy \in L_2 \iff xvy \in L_2] \\ \iff\ &[|xuy|_{\mathtt{a}} = |xuy|_{\mathtt{b}} \iff |xvy|_{\mathtt{a}} = |xvy|_{\mathtt{b}}] \\ \iff\ &[|x|_{\mathtt{a}} + |u|_{\mathtt{a}} + |y|_{\mathtt{a}} = |x|_{\mathtt{b}} + |u|_{\mathtt{b}} + |y|_{\mathtt{b}} \iff |x|_{\mathtt{a}} + |v|_{\mathtt{a}} + |y|_{\mathtt{a}} = |x|_{\mathtt{b}} + |v|_{\mathtt{b}} + |y|_{\mathtt{b}}] \\ \iff\ &[|u|_{\mathtt{a}} - |u|_{\mathtt{b}} = |x|_{\mathtt{b}} - |x|_{\mathtt{a}} + |y|_{\mathtt{b}} - |y|_{\mathtt{a}} \iff |v|_{\mathtt{a}} - |v|_{\mathtt{b}} = |x|_{\mathtt{b}} - |x|_{\mathtt{a}} + |y|_{\mathtt{b}} - |y|_{\mathtt{a}}] \\ \iff\ &[|u|_{\mathtt{a}} - |u|_{\mathtt{b}} = |v|_{\mathtt{a}} - |v|_{\mathtt{b}}]\end{aligned}$$

  となるので，統語的合同関係 $\equiv_{L_2}$ の同値類は $\mathtt{a}$ の個数と $\mathtt{b}$ の個数の差で決まる．よって $\mathrm{Syn}(L_2) = \{\ldots, [\mathtt{bb}], [\mathtt{b}], [\varepsilon], [\mathtt{a}], [\mathtt{aa}], \ldots\} \cong \mathbb{Z}$ となる．

命題 2.19.　Σ 上の言語 $L \subseteq \Sigma^*$ に対し，以下の 2 条件は同値である．

(1) L は Σ^* の認識可能部分集合である．
(2) L の統語モノイド $\mathrm{Syn}(L)$ が有限モノイドである．

証明. (1) $\Longrightarrow$ (2). 仮定より存在する有限モノイド N と全射モノイド準同型写像 $\pi\colon \Sigma^* \twoheadrightarrow N$ をとる．$L = \pi^{-1}(\pi(L))$ だから，

$$\begin{aligned}u \equiv_L v &\iff \forall x, y \in \Sigma^*\ [xuy \in L \iff xvy \in L] \\ &\iff \forall x, y \in \Sigma^*\ [xuy \in \pi^{-1}(\pi(L)) \iff xvy \in \pi^{-1}(\pi(L))] \\ &\iff \forall x, y \in \Sigma^*\ [\pi(xuy) \in \pi(L) \iff \pi(xvy) \in \pi(L)] \\ &\iff \forall x, y \in \Sigma^*\ [\pi(x)\pi(u)\pi(y) \in \pi(L) \iff \pi(x)\pi(v)\pi(y) \in \pi(L)]\end{aligned}$$

[*7] L の統語モノイドを $M(L)$ と書く文献も多い．

$$\Longleftarrow\ \pi(u) = \pi(v)$$

となるので，$|\mathrm{Syn}(L)| \leq |N| < \infty$ を得る．

(2) $\Longrightarrow$ (1). 商写像 $\pi\colon \Sigma^* \twoheadrightarrow \mathrm{Syn}(L)$ をとる．$L = \pi^{-1}(\pi(L))$ を示せばよい．$L \subseteq \pi^{-1}(\pi(L))$ は無条件に成り立つ．任意に $u \in \pi^{-1}(\pi(L))$ をとると，$\pi(u) = \pi(v)$ なる $v \in L$ が存在する．π が商写像であることより $u \equiv_L v$ つまり $\forall x, y \in \Sigma^*\ [xuy \in L \iff xvy \in L]$ であるので，特に $x = y = \varepsilon$ の場合を考えれば $u \in L \iff v \in L$ だから $u \in L$ となる． □

定理 2.20 (Myhill–Nerode の定理). 次の 2 条件が成り立つ．

(1) 任意の DFA $\mathcal{A}$ に対し，統語モノイド $\mathrm{Syn}(\mathrm{L}(\mathcal{A}))$ は有限モノイドである．
(2) Σ 上の言語 L が Σ^* の認識可能部分集合ならば，ある DFA $\mathcal{A}$ が存在して $\mathrm{L}(\mathcal{A}) = L$ となる．

証明. (1) DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ をとる．各文字列 $w \in \Sigma^*$ に対し，Q から Q への写像 $f_w \in Q^Q$ を次のように定める．状態 $q \in Q$ からスタートして，w を左から順番に読み込みつつ，遷移関数 δ に従って遷移する．w を読み終えた段階で状態 r にいるならば，$f_w(q) := r$ と定義する．定義から任意の $u, v \in \Sigma^*$ に対し $f_{uv}(q) = f_v(f_u(q))$ が成り立つことがわかるので，

$$\begin{aligned} u \equiv_{\mathrm{L}(\mathcal{A})} v &\iff \forall x, y \in \Sigma^*\ [xuy \in \mathrm{L}(\mathcal{A}) \iff xvy \in \mathrm{L}(\mathcal{A})] \\ &\iff \forall x, y \in \Sigma^*\ [f_{xuy}(q_0) \in F \iff f_{xvy}(q_0) \in F] \\ &\iff \forall x, y \in \Sigma^*\ [f_y(f_u(f_x(q_0))) \in F \iff f_y(f_v(f_x(q_0))) \in F] \\ &\Longleftarrow f_u = f_v \end{aligned}$$

となる．よって $|\mathrm{Syn}(\mathrm{L}(\mathcal{A}))| \leq |Q^Q| = |Q|^{|Q|} < \infty$ を得る．

(2) 仮定より存在する有限モノイド N と全射モノイド準同型写像 $\pi\colon \Sigma^* \twoheadrightarrow N$ をとる．DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ を

$$\begin{aligned} Q &:= N, \\ \delta(n, a) &:= n \cdot \pi(a), \\ q_0 &:= \pi(\varepsilon), \\ F &:= \pi(L) \end{aligned}$$

とおく．$\mathcal{A}$ において，開始状態 $q_0 = \pi(\varepsilon)$ からスタートして，文字列 $w \in \Sigma^*$ を読み終えた時点で状態 $\pi(w)$ にいることが，長さ $|w|$ に関する帰納法で簡単にわかる．よって

$$w \in \mathrm{L}(\mathcal{A}) \iff \pi(w) \in F = \pi(L) \iff w \in \pi^{-1}(\pi(L)) = L$$

となるので $\mathrm{L}(\mathcal{A}) = L$ を得る． □

例 2.21. 定理 2.20 より，例 2.18 の L_1 は正則言語だが L_2 は正則言語ではない．

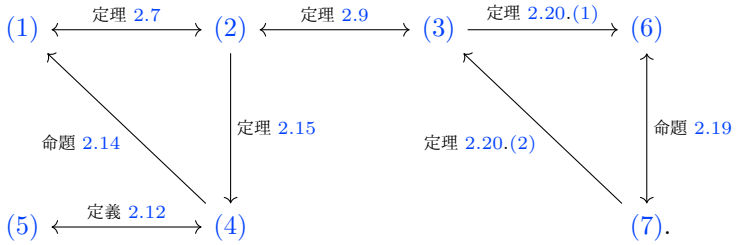
2.2.4 まとめ

ここまでの議論をまとめると，正則言語とは以下のいずれか，したがってすべての条件をみたす言語である．よって正則言語の定義としてどれを採用しても同じである．

定理 2.22.　Σ を有限アルファベットとする．言語 $L \subseteq \Sigma^*$ に対し，以下はすべて同値である．

(1) L はある正則文法で生成される．
(2) L はある NFA により認識される．
(3) L はある DFA により認識される．
(4) L はある正則表現で表される．
(5) L は Σ^* の有理的部分集合である．
(6) L の統語モノイド $\mathrm{Syn}(L)$ が有限モノイドである．
(7) L は Σ^* の認識可能部分集合である．

証明. 以下の図からわかる．



□

2.3　文脈自由言語

文脈自由言語は文脈自由文法によって定義される言語であり，簡単な自然言語の文やプログラミング言語のソースコードを表現することができる．文脈自由言語は Muller–Schupp の定理 0.1 の重要な構成要素でもある．正則言語について，正則文法に対応する NFA という機械があったように，文脈自由文法にも対応する機械があり，**プッシュダウンオートマトン** (pushdown automaton; PDA) と呼ばれている．プッシュダウンオートマトンは NFA に加えて**スタック** (stack) と呼ばれる追加の記憶領域を持っており，そのため正則言語よりも多くの言語を認識することが可能になっている．本章では文脈自由言語の 2 つの定義，文脈自由文法とプッシュダウンオートマトンが同等の表現力を持つことを見ることにする．

例 2.4 では自然言語の例を見たが，他にもいくつか例を見てみよう．

例 2.23.

- $G_1 = (V = \{S\}, \Sigma = \{\mathtt{a}, \mathtt{b}\}, P = \{S \to \mathtt{a}S\mathtt{b} \mid \varepsilon\}, S)$ とすると $\mathrm{L}(G_1) := \{\, \mathtt{a}^n\mathtt{b}^n \mid n \geq 0 \,\}$ となる．
- $G_2 = (V = \{S\}, \Sigma = \{(,)\}, P = \{S \to (S)S \mid \varepsilon\}, S)$ とすると，$\mathrm{L}(G_2) = \{\varepsilon, (), (()), ()(), \dots\}$ は左右の対応がとれた括弧の列の全体である．$\mathrm{L}(G_2)$ を (括弧が 1 種類の) **半 Dyck 言語** (semi-Dyck language) という．

2.3.1　構文木と最左導出

ここでは例 2.4 で紹介した構文木の正確な定義を行う．動機付けのための例として，文脈自由文法 $G = (V, \Sigma, P, S)$ を

$$
\begin{aligned}
V &:= \{S\}, \\
\Sigma &:= \{\mathtt{a}, \mathtt{b}\}, \\
P &:= \{S \to \varepsilon \mid SS \mid \mathtt{a}S\mathtt{b} \mid \mathtt{b}S\mathtt{a}\}
\end{aligned}
$$

と定義する．文字列 $\mathtt{abab} \in \Sigma^*$ に対し，文法 G のもとでの 2 つの導出

$$\begin{aligned} &S \underset{P}{\Longrightarrow} SS \underset{P}{\Longrightarrow} \mathtt{a}S\mathtt{b}S \underset{P}{\Longrightarrow} \mathtt{ab}S \underset{P}{\Longrightarrow} \mathtt{aba}S\mathtt{b} \underset{P}{\Longrightarrow} \mathtt{abab}, \\ &S \underset{P}{\Longrightarrow} SS \underset{P}{\Longrightarrow} S\mathtt{a}S\mathtt{b} \underset{P}{\Longrightarrow} S\mathtt{ab} \underset{P}{\Longrightarrow} \mathtt{a}S\mathtt{bab} \underset{P}{\Longrightarrow} \mathtt{abab} \end{aligned} \tag{2.2}$$

は，生成規則の適用の順序が違うだけで，本質的には同じことをやっていると考えられる．一方で，$\mathtt{abab}$ の別の導出

$$S \underset{P}{\Longrightarrow} \mathtt{a}S\mathtt{b} \underset{P}{\Longrightarrow} \mathtt{ab}S\mathtt{ab} \underset{P}{\Longrightarrow} \mathtt{abab} \tag{2.3}$$

は導出 (2.2) とは本質的に異なるように思われる．これらの感覚をきちんと定式化するために，文脈自由文法の導出に対応する構文木を定義する．

定義 2.24 (文脈自由文法の導出の構文木). $G = (V, \Sigma, P, S)$ を文脈自由文法とする．導出 $S = \alpha_0 \underset{P}{\Longrightarrow} \alpha_1 \underset{P}{\Longrightarrow} \alpha_2 \underset{P}{\Longrightarrow} \cdots \underset{P}{\Longrightarrow} \alpha_n = w \in \Sigma^*$ $(\alpha_i \in (V \cup \Sigma)^*)$ をとる．G が文脈自由文法であることより，各 i について $\alpha_i = \alpha_i' A_i \alpha_i''$，$\alpha_{i+1} = \alpha_i' \beta_i \alpha_i''$ $(\alpha_i', \alpha_i'', \beta_i \in (V \cup \Sigma)^*$，$A_i \in V$，$A_i \to \beta_i \in P)$ という形をしているとしてよい．$\mathtt{[}, \mathtt{]}$ を $V \cup \Sigma$ に含まれない文字とする．この導出に対応する**構文木** (syntax tree) または**導出木** (derivation tree) $T \in (V \cup \Sigma \cup \{\mathtt{[}, \mathtt{]}\})^*$ を次のように帰納的に定義する．

- $\alpha_n = w = w_1 w_2 \cdots w_m$ $(w_i \in \Sigma)$ に対しては $T_n := w_1 \mathtt{[]} w_2 \mathtt{[]} \cdots w_n \mathtt{[]}$ と定義する．
- $\alpha_{i+1} = \alpha_i' \beta_i \alpha_i'' = \alpha_{i,1}' \cdots \alpha_{i,m'}' \beta_{i,1} \cdots \beta_{i,m} \alpha_{i,1}'' \cdots \alpha_{i,m''}''$ $(\alpha_{i,j}', \beta_{i,j}, \alpha_{i,j}'' \in V \cup \Sigma)$ に対して

  $$T_{i+1} = \alpha_{i,1}' \mathtt{[}\cdots\mathtt{]} \cdots \alpha_{i,m'}' \mathtt{[}\cdots\mathtt{]} \beta_{i,1} \mathtt{[}\cdots\mathtt{]} \cdots \beta_{i,m} \mathtt{[}\cdots\mathtt{]} \alpha_{i,1}'' \mathtt{[}\cdots\mathtt{]} \cdots \alpha_{i,m''}'' \mathtt{[}\cdots\mathtt{]}$$

  が定義されているとき，

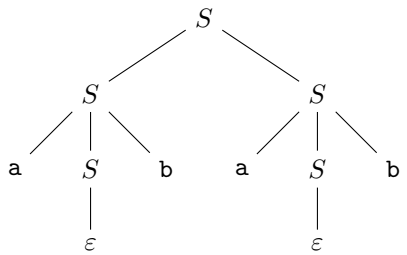
  $$T_i := \alpha_{i,1}' \mathtt{[}\cdots\mathtt{]} \cdots \alpha_{i,m'}' \mathtt{[}\cdots\mathtt{]} A_i \mathtt{[}\beta_{i,1} \mathtt{[}\cdots\mathtt{]} \cdots \beta_{i,m} \mathtt{[}\cdots\mathtt{]}\mathtt{]} \alpha_{i,1}'' \mathtt{[}\cdots\mathtt{]} \cdots \alpha_{i,m''}'' \mathtt{[}\cdots\mathtt{]}$$

  と定義する ($\beta_i = \varepsilon$ のときは T_i は T_{i+1} に $A_i \mathtt{[]}$ を挿入したものになることに注意)．
- $T := T_0$ とおく．

例 2.25. 導出 (2.2) に対応する構文木はどちらも

$$T = S\mathtt{[}S\mathtt{[a[]}S\mathtt{[]b[]]}S\mathtt{[a[]}S\mathtt{[]b[]]]}$$

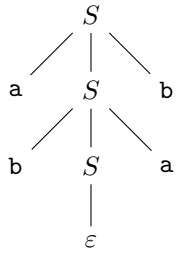
であり，実際に木の形に描くと



となる．一方，導出 (2.3) に対応する構文木は

$$T = S\mathtt{[a[]}S\mathtt{[b[]}S\mathtt{[]a[]]b[]]}$$

であり，実際に木の形に描くと



となる．

上で見たように，同じ構文木を与えるような導出が 2 つ以上存在することがある．それらの中で最も「標準的な」導出が次の最左導出である．

定義 2.26 (最左導出). $G = (V, \Sigma, P, S)$ を文脈自由文法とする．文字列 $w \in \Sigma^*$ に対し，導出 $S \overset{*}{\underset{P}{\Longrightarrow}} w$ が**最左導出** (leftmost derivation) であるとは，導出の各段階において，最も左側にある変数が書き換えられていることをいう．形式的に書けば，導出 $S = \alpha_0 \underset{P}{\Longrightarrow} \alpha_1 \underset{P}{\Longrightarrow} \cdots \underset{P}{\Longrightarrow} \alpha_n = w$ が最左導出であるとは，$0 \leq i < n$ なる i について $\alpha_i = wA\beta$ $(w \in \Sigma^*, A \in V, \beta \in (V \cup \Sigma)^*)$ と表したとき，ある規則 $A \to \gamma \in P$ が存在して $\alpha_{i+1} = w\gamma\beta$ となっていることをいう．

補題 2.27. 文脈自由文法 $G = (V, \Sigma, P, S)$ のもとで文字列 $w \in \Sigma^*$ の導出 $S \overset{*}{\underset{P}{\Longrightarrow}} w$ があるとき，この導出に対応する構文木を与えるような，S から w への最左導出が唯一つ存在する．

証明. まず，導出 $S \overset{*}{\underset{P}{\Longrightarrow}} w$ に対応する構文木 T を構成する．T の文字を左から順に読んでいき，変数に出会ったら対応する変数に生成規則を適用していくことで最左導出が得られる (構文木を図に描いたとき，根からスタートして木の左下方向へ向かって**深さ優先探索** (depth-first search) を行い，訪れた変数記号に生成規則を適用していくということもできる)．また，対応する構文木が T になるような最左導出はこれしかない．下の例 2.28 も参照のこと． □

例 2.28. 文脈自由文法 $G = (V, \Sigma, P, S)$ を

$$V := \{S, A\},$$
$$\Sigma := \{\mathtt{a}, \mathtt{b}\},$$
$$P := \begin{cases} S \to \mathtt{b} \mid SS \mid SA, \\ A \to \varepsilon \mid \mathtt{a}A\mathtt{bb} \end{cases}$$

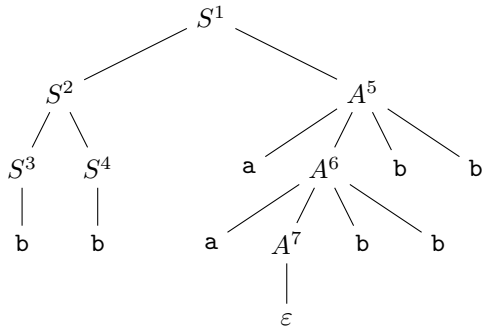
と定義する．このとき，導出

$$S \underset{P}{\Longrightarrow} SA \underset{P}{\Longrightarrow} SSA \underset{P}{\Longrightarrow} SS\mathtt{a}A\mathtt{bb} \underset{P}{\Longrightarrow} SS\mathtt{aa}A\mathtt{bbbb} \underset{P}{\Longrightarrow} SS\mathtt{aabbbb} \underset{P}{\Longrightarrow} S\mathtt{baabbbb} \underset{P}{\Longrightarrow} \mathtt{bbaabbbb}$$

に対応する構文木は

$$S\mathtt{[}S\mathtt{[}S\mathtt{[b[]]}S\mathtt{[b[]]]}A\mathtt{[a[]}A\mathtt{[a[]}A\mathtt{[]b[]b[]]b[]b[]]}$$

であり，図に描けば



であり (変数の右上の番号は深さ優先探索をしたときに頂点を訪れる順番)，これより最左導出

$$S \underset{P}{\Longrightarrow} SA \underset{P}{\Longrightarrow} SSA \underset{P}{\Longrightarrow} \mathtt{b}SA \underset{P}{\Longrightarrow} \mathtt{bb}A \underset{P}{\Longrightarrow} \mathtt{bba}A\mathtt{bb} \underset{P}{\Longrightarrow} \mathtt{bbaa}A\mathtt{bbbb} \underset{P}{\Longrightarrow} \mathtt{bbaabbbb}$$

が得られる．

補題 2.27 の証明より各導出に対し，構文木と最左導出の間には一対一の対応があることがわかる．

2.3.2 Chomsky 標準形

定義 2.3 の文脈自由文法の定義では，生成規則 $\alpha \to \beta$ の右辺 $\beta \in (V \cup \Sigma)^*$ には制限がなく，どんな文字列でもよいことになっている．このような無制限な文法は取り回しが悪く，扱いが難しいことが多い．そこで，より制限された扱いやすい文法として，次の Chomsky 標準形がある．

定義 2.29 (Chomsky 標準形). 文脈自由文法 $G = (V, \Sigma, P, S)$ が **Chomsky 標準形** (Chomsky normal form) であるとは，G のどの生成規則も次のいずれかの形をしていることをいう．

(1) 変数 $A, B, C \in V$ で $B \neq S \neq C$ なるものについて $A \to BC$.
(2) 変数 $A \in V$ と終端記号 $a \in \Sigma$ について $A \to a$.
(3) 開始記号 S について $S \to \varepsilon$.

Chomsky 標準形の文法はもともとの文脈自由文法に比べて制限が強く，一見すると表現力が落ちているように思えるかもしれない．しかし，実はどんな文脈自由文法も等価な Chomsky 標準形に書き換えることができるのである．

定理 2.30. どんな文脈自由文法 $G = (V, \Sigma, P, S)$ に対しても，Chomsky 標準形の文脈自由文法 $G' = (V', \Sigma, P', S')$ であって $\mathrm{L}(G') = \mathrm{L}(G)$ となるものを構成できる．

証明. 以下のように段階的に文法を変形していく．

(1) まず，新しい開始記号 $S' \notin V$ を用意して，$G' = (V \cup \{S'\}, \Sigma, P \cup \{S' \to S\}, S')$ とおく．このとき明らかに $\mathrm{L}(G') = \mathrm{L}(G)$ だから，これ以降 G の生成規則の右辺には開始記号 S は現れないと仮定してよい．
(2) 次に，$A \to \varepsilon$ の形の規則を除去することを考える．G の変数を $V = \{S = A_0, A_1, \ldots, A_n\}$ $(n \geq 0)$ のように番号付けして，$A_i \to \varepsilon$ $(1 \leq i \leq n)$ という規則を帰納的に取り除いていく (Chomsky 標準形の定義から，$A_0 \to \varepsilon$ という規則はあってもよい)．$1 \leq i \leq n$ なる i を固定する．P には $A_j \to \varepsilon$ $(1 \leq j < i)$ の形の規則はないと仮定する (帰納法の仮定)．$A_i \to \varepsilon \notin P$ のとき，するべきことは何もない．$A_i \to \varepsilon \in P$ とする．まず，P から $A_i \to \varepsilon$ を削除したものを $P' := P - \{A_i \to \varepsilon\}$ とおく．A_i を右辺に含む規則 $A_j \to \beta \in P$ ごとに，次の操作を行う．

($\beta = A_i^n$ ($n \geq 1$) かつ $j \leq i$ のとき) $A_j \to A_i \mid A_i^2 \mid \cdots \mid A_i^n$ を P' に加える ($A_j \to \varepsilon$ は帰納法の仮定から既に削除してあるので，新たに加える必要はない).

(それ以外のとき) $\beta = \beta_0 A_i \beta_1 A_i \cdots A_i \beta_n$ ($\beta_i \in ((V - \{A_i\}) \cup \Sigma)^*$) とする．このとき，各 $(e_1, e_2, \ldots, e_n) \in \{0,1\}^n$ に対し，規則 $A_j \to \beta_0 A_i^{e_1} \beta_1 A_i^{e_2} \cdots A_i^{e_n} \beta_n$ を P' に加える．例えば，$A_j \to \mathtt{a}A_i\mathtt{b}A_i\mathtt{a}$ という規則があったら $A_j \to \mathtt{aba} \mid \mathtt{a}A_i\mathtt{ba} \mid \mathtt{ab}A_i\mathtt{a} \mid \mathtt{a}A_i\mathtt{b}A_i\mathtt{a}$ をすべて P' に加える．$j \leq i$ なら β が A_i 以外の記号を含むことより，右辺が ε になることはないことに注意する ($j > i$ のときは右辺が ε になっても問題ない).

このとき $\mathrm{L}(G') = \mathrm{L}(G)$ が成り立つことを示す．$\mathrm{L}(G') \subseteq \mathrm{L}(G)$ であること，すなわち各文字列 $w \in \Sigma^*$ に対し $S \overset{*}{\underset{P'}{\Longrightarrow}} w$ ならば $S \overset{*}{\underset{P}{\Longrightarrow}} w$ であることは簡単にわかる．逆を示す．導出 $S \overset{*}{\underset{P}{\Longrightarrow}} w$ をひとつ固定する．この導出が $A_i \to \varepsilon$ という規則を用いていないならば，示すことはない．そこで，この導出が規則 $A_i \to \varepsilon$ を 1 回以上用いているとする．そのうちの 1 回に着目すると，$A_i \neq A_0 = S$ より，ある規則 $A_j \to \alpha' A_i \gamma' \in P$ ($\alpha', \gamma' \in (V \cup \Sigma)^*$) と $\alpha, \gamma, \alpha'', \gamma'' \in (V \cup \Sigma)^*$ が存在して

$$S \overset{*}{\underset{P}{\Longrightarrow}} \alpha A_j \gamma \underset{P}{\Longrightarrow} \alpha\alpha' A_i \gamma' \gamma \overset{*}{\underset{P}{\Longrightarrow}} \alpha'' A_i \gamma'' \underset{P}{\Longrightarrow} \alpha'' \gamma'' \overset{*}{\underset{P}{\Longrightarrow}} w$$

となる．このとき P' の作り方から $A_j \to \alpha'\gamma' \in P'$ なので，新たな導出

$$S \overset{*}{\underset{P}{\Longrightarrow}} \alpha A_j \gamma \underset{P'}{\Longrightarrow} \alpha\alpha'\gamma'\gamma \overset{*}{\underset{P}{\Longrightarrow}} \alpha''\gamma'' \overset{*}{\underset{P}{\Longrightarrow}} w$$

が得られ，この導出 $S \overset{*}{\underset{P \cup P'}{\Longrightarrow}} w$ は規則 $A_i \to \varepsilon$ の使用回数がもとの導出より 1 つ少なくなっている．よってこの導出の変形を繰り返せば，最終的に導出 $S \overset{*}{\underset{P'}{\Longrightarrow}} w$ が得られる．以上より，これ以降 G は $A \to \varepsilon$ の形の規則を含まないとしてよい.

(3) 次に，$A \to B$ ($A, B \in V$) の形の規則を除去することを考える．V 上の二項関係 $\longrightarrow \subseteq V \times V$ を，$A, B \in V$ に対して $A \longrightarrow B :\Longleftrightarrow A \to B \in P$ と定義し，さらに前順序 $\leq \subseteq V \times V$ を $\leq := \overset{*}{\longrightarrow}$ と定義する (cf. 命題 1.2)．最初に $P' := P$ とおく．各変数 $A \in V$ に対し，次の操作を行う.

> $A \leq B$ となる変数 $B \in V$ のそれぞれに対し，次の操作を行う.
>
> > $B \to \beta \in P$ ($\beta \in (V \cup \Sigma)^*$) という形の規則のそれぞれに対し，$A \to \beta$ を P' に加える.

最後に，P' から $A \to B$ という形の規則をすべて削除する．このとき $\mathrm{L}(G') = \mathrm{L}(G)$ が成り立つことを示す．$\mathrm{L}(G') \subseteq \mathrm{L}(G)$ であること，すなわち各文字列 $w \in \Sigma^*$ に対し $S \overset{*}{\underset{P'}{\Longrightarrow}} w$ ならば $S \overset{*}{\underset{P}{\Longrightarrow}} w$ であることは簡単にわかる．逆を示す．導出 $S \overset{*}{\underset{P}{\Longrightarrow}} w$ をひとつ固定する．この導出が $A \to B$ という形の規則を用いていないならば，示すことはない．そこで，この導出が $A \to B$ という形の規則を 1 回以上用いているとする．そのうちの 1 回に着目すると，$B \notin \Sigma$ よりある規則 $B \to \beta$ ($\beta \in (V \cup \Sigma)^*$) と $\alpha, \gamma, \alpha', \gamma' \in (V \cup \Sigma)^*$ が存在して

$$S \overset{*}{\underset{P}{\Longrightarrow}} \alpha A \gamma \underset{P}{\Longrightarrow} \alpha B \gamma \overset{*}{\underset{P}{\Longrightarrow}} \alpha' B \gamma' \underset{P}{\Longrightarrow} \alpha' \beta \gamma' \overset{*}{\underset{P}{\Longrightarrow}} w$$

となる．このとき P' の作り方から $A \to \beta \in P'$ なので，新たな導出

$$S \overset{*}{\underset{P}{\Longrightarrow}} \alpha A \gamma \underset{P'}{\Longrightarrow} \alpha\beta\gamma \overset{*}{\underset{P}{\Longrightarrow}} \alpha'\beta\gamma' \overset{*}{\underset{P}{\Longrightarrow}} w$$

が得られ，この導出 $S \overset{*}{\underset{P \cup P'}{\Longrightarrow}} w$ は規則 $A \to B$ の使用回数がもとの導出より 1 つ少なくなっている．よってこの導出の変形を繰り返せば，最終的に導出 $S \overset{*}{\underset{P'}{\Longrightarrow}} w$ が得られる．以上より，これ以降 G は $A \to B$ の形の規則を含まないとしてよい.

(4) ここまでの変形により，G の各生成規則 $A \to \beta \in P$ は $|\beta| \geq 1$ をみたし，さらに $|\beta| = 1 \iff \beta \in \Sigma$ が成り立っている．$P' := P, V' := V$ から始めて，各規則 $A \to \beta = v_1 v_2 \cdots v_k \in P$ $(v_i \in V \cup \Sigma)$ で $k \geq 3$ なるものに対し，次の操作を行う．新たな変数 $A_1^\beta, A_2^\beta, \ldots, A_{k-2}^\beta$ をすべて V' に加え，新たな規則 $A \to v_1 A_1^\beta, A_1^\beta \to v_2 A_2^\beta, \ldots, A_{k-2}^\beta \to v_{k-1} v_k$ をすべて P' に加える．このとき，$\mathrm{L}(G') = \mathrm{L}(G)$ となることは明らかであろう．よってこれ以降，各規則 $A \to \beta \in P$ は $1 \leq |\beta| \leq 2$ をみたすとしてよい．

(5) 最後に，$A \to v_1 v_2 \in P$ の形の規則で，$v_1 \in \Sigma$ または $v_2 \in \Sigma$ となるものを除去する．$P' := P, V' := V$ から始めて，各規則 $A \to v_1 v_2 \in P$ について，$v_i \in \Sigma$ であれば U_{v_i} という新たな変数を V' に，$U_{v_i} \to v_i$ を P' に加え，さらに $A \to v_1 v_2 \in P'$ の右辺の v_i を U_{v_i} で置き換える (もとの規則は削除する)．明らかに $\mathrm{L}(G') = \mathrm{L}(G)$ であり，この G' は Chomsky 標準形である． □

例 2.31. 文脈自由文法 $G = (V, \Sigma, P, S)$ を

$$
\begin{aligned}
V &:= \{S, A, B\}, \\
\Sigma &:= \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, \\
P &:= \begin{cases} S \to \varepsilon \mid AAS \mid BS, \\ A \to \mathtt{a} \mid \mathtt{b}A\mathtt{b} \mid \mathtt{c}A\mathtt{c}, \\ B \to \varepsilon \mid \mathtt{bc} \mid \mathtt{cb} \end{cases}
\end{aligned}
$$

と定義する．この G を定理 2.30 の証明に沿って変形していく．

(1) まず，新しい開始変数 S' と規則 $S' \to S$ を追加して

$$
\begin{aligned}
&S' \to S, \\
&S \to \varepsilon \mid AAS \mid BS, \\
&A \to \mathtt{a} \mid \mathtt{b}A\mathtt{b} \mid \mathtt{c}A\mathtt{c}, \\
&B \to \varepsilon \mid \mathtt{bc} \mid \mathtt{cb}.
\end{aligned}
$$

(2) 次に，$S \to \varepsilon$ を除去すると

$$
\begin{aligned}
&S' \to S \mid \varepsilon, \\
&S \to AAS \mid AA \mid BS \mid B, \\
&A \to \mathtt{a} \mid \mathtt{b}A\mathtt{b} \mid \mathtt{c}A\mathtt{c}, \\
&B \to \varepsilon \mid \mathtt{bc} \mid \mathtt{cb}
\end{aligned}
$$

であり，さらに $B \to \varepsilon$ を除去すると

$$
\begin{aligned}
&S' \to S \mid \varepsilon, \\
&S \to AAS \mid AA \mid BS \mid S \mid B, \\
&A \to \mathtt{a} \mid \mathtt{b}A\mathtt{b} \mid \mathtt{c}A\mathtt{c}, \\
&B \to \mathtt{bc} \mid \mathtt{cb}.
\end{aligned}
$$

(3) 次に，$S \to B$ を除去すると

$$
\begin{aligned}
&S' \to S \mid \varepsilon, \\
&S \to AAS \mid AA \mid BS \mid S \mid \mathtt{bc} \mid \mathtt{cb}, \\
&A \to \mathtt{a} \mid \mathtt{b}A\mathtt{b} \mid \mathtt{c}A\mathtt{c}, \\
&B \to \mathtt{bc} \mid \mathtt{cb}
\end{aligned}
$$

であり，さらに $S \to S$ を除去して

$$\begin{aligned} S' &\to S \mid \varepsilon, \\ S &\to AAS \mid AA \mid BS \mid \mathtt{bc} \mid \mathtt{cb}, \\ A &\to \mathtt{a} \mid \mathtt{b}A\mathtt{b} \mid \mathtt{c}A\mathtt{c}, \\ B &\to \mathtt{bc} \mid \mathtt{cb}, \end{aligned}$$

さらに $S' \to S$ を除去すると

$$\begin{aligned} S' &\to AAS \mid AA \mid BS \mid \mathtt{bc} \mid \mathtt{cb} \mid \varepsilon, \\ S &\to AAS \mid AA \mid BS \mid \mathtt{bc} \mid \mathtt{cb}, \\ A &\to \mathtt{a} \mid \mathtt{b}A\mathtt{b} \mid \mathtt{c}A\mathtt{c}, \\ B &\to \mathtt{bc} \mid \mathtt{cb}. \end{aligned}$$

(4) 新しい変数 $A_1^{AAS}, A_1^{\mathtt{b}A\mathtt{b}}, A_1^{\mathtt{c}A\mathtt{c}}$ を追加して

$$\begin{aligned} S' &\to AA_1^{AAS} \mid AA \mid BS \mid \mathtt{bc} \mid \mathtt{cb} \mid \varepsilon, \\ S &\to AA_1^{AAS} \mid AA \mid BS \mid \mathtt{bc} \mid \mathtt{cb}, \\ A &\to \mathtt{a} \mid \mathtt{b}A_1^{\mathtt{b}A\mathtt{b}} \mid \mathtt{c}A_1^{\mathtt{c}A\mathtt{c}}, \\ B &\to \mathtt{bc} \mid \mathtt{cb}, \\ A_1^{AAS} &\to AS, \\ A_1^{\mathtt{b}A\mathtt{b}} &\to A\mathtt{b}, \\ A_1^{\mathtt{c}A\mathtt{c}} &\to A\mathtt{c}. \end{aligned}$$

(5) 最後に，$U_{\mathtt{b}} \to \mathtt{b}, U_{\mathtt{c}} \to \mathtt{c}$ を追加して，Chomsky 標準形

$$\begin{aligned} S' &\to AA_1^{AAS} \mid AA \mid BS \mid U_{\mathtt{b}}U_{\mathtt{c}} \mid U_{\mathtt{c}}U_{\mathtt{b}} \mid \varepsilon, \\ S &\to AA_1^{AAS} \mid AA \mid BS \mid U_{\mathtt{b}}U_{\mathtt{c}} \mid U_{\mathtt{c}}U_{\mathtt{b}}, \\ A &\to \mathtt{a} \mid U_{\mathtt{b}}A_1^{\mathtt{b}A\mathtt{b}} \mid U_{\mathtt{c}}A_1^{\mathtt{c}A\mathtt{c}}, \\ B &\to U_{\mathtt{b}}U_{\mathtt{c}} \mid U_{\mathtt{c}}U_{\mathtt{b}}, \\ A_1^{AAS} &\to AS, \\ A_1^{\mathtt{b}A\mathtt{b}} &\to AU_{\mathtt{b}}, \\ A_1^{\mathtt{c}A\mathtt{c}} &\to AU_{\mathtt{c}}, \\ U_{\mathtt{b}} &\to \mathtt{b}, \\ U_{\mathtt{c}} &\to \mathtt{c} \end{aligned}$$

を得る．

Chomsky 標準形の文法を用いると，次の決定可能性が得られる．

系 2.32 (文脈自由言語の所属判定問題の決定可能性). 与えられた文脈自由文法 G と文字列 $w \in \Sigma^*$ が $w \in \mathrm{L}(G)$ をみたすかどうかを判定するアルゴリズムを構成できる．

証明. まず，与えられた文脈自由文法 G を定理 2.30 に従って Chomsky 標準形に変形することで，$G = (V, \Sigma, P, S)$ は最初から Chomsky 標準形であると仮定してよい．$w = \varepsilon$ であれば $S \to \varepsilon \in P$ かどうかを確かめればよい．$w \neq \varepsilon$ と仮定する．仮に導出 $S \overset{*}{\underset{P}{\Longrightarrow}} w$ が存在したとすると，G が Chomsky 標準形であることより，$A \to BC$ の形

の規則を適用すると長さが 1 だけ増え，$A \to a$ の形の規則では長さは変化しないので，$S \overset{2|w|-1}{\underset{P}{\Longrightarrow}} w$ でなければならない．ここで，S から始まる長さ $2|w|-1$ の導出は有限個しか存在しないので，それらが w を生成するかどうかをチェックすればよい． □

2.3.3 プッシュダウンオートマトン

正則言語の場合には正則文法に対応する NFA という概念があったように，文脈自由文法に対応する機械の概念がプッシュダウンオートマトンである．プッシュダウンオートマトンは NFA に加えて**スタック** (stack) と呼ばれる記憶領域を持つ．スタックは**最後に入れたものが最初に出てくる** (last in, first out) データ構造であり，言うなればスーパーマーケットやコンビニエンスストアにおける買い物カゴの山のようなものである．プッシュダウンオートマトンは計算の過程でスタックの一番上 (トップ) に文字をプッシュ (積む) したりポップ (降ろす) したりすることができる．スタックは無制限の記憶容量を持つが，遷移の際には

- 入力文字列中の，次に読み込まれる文字，
- スタックの一番上に積まれている文字

しか参照することができない．本稿では紙面の都合上，スタックを上ではなく右に伸びる文字列として表すことにする．

定義 2.33 (プッシュダウンオートマトン). **プッシュウダウンオートマトン** (pushdown automaton; PDA) とは，以下のデータからなる 6 つ組 $\mathcal{A} = (Q, \Sigma, Z, \delta, q_0, F)$ のことである．

- 状態の有限集合 Q，
- 有限アルファベット Σ，
- 有限の**スタックアルファベット** (stack alphabet) Z[*8]，
- 文字列書き換え系 $\delta \subseteq Z^*Q\Sigma^* \times Z^*Q$，
- 開始状態 $q_0 \in Q$，
- 受理状態の集合 $F \subseteq Q$.

ただしここで $Z^*Q\Sigma^* := \{\, \sigma q w \mid \sigma \in Z^*, q \in Q, w \in \Sigma^* \,\}, Z^*Q := \{\, \sigma q \mid \sigma \in Z^*, q \in Q \,\}$ である．集合 $Z^*Q\Sigma^*$ の元 $\sigma q w$ を**計算状況** (configuration) または**時点表示** (instantaneous description) と呼び，PDA の動作のある時点においてスタックの内容が σ (右端がスタックトップ)，内部状態が q，入力文字列のうち未だ読み込んでいない部分が w であることを表している．文字列書き換え系 δ の各元 $(\sigma q w, \sigma' q') \in \delta$ は「入力文字列のまだ読み込んでいない部分から w を読み込み (消費し)，PDA の内部状態を q から q' に変更し，さらにスタック上方の σ をポップし，σ' をプッシュしてよい」ことを表している．
PDA $\mathcal{A}$ が**決定性プッシュダウンオートマトン** (deterministic PDA; DPDA) であるとは，関係 $\underset{\delta}{\Longrightarrow}$ が $Z^*Q\Sigma^*$ から Z^*Q への**部分関数** (partial function) であること，すなわちどんな計算状況 $\sigma q w \in Z^*Q\Sigma^*$ に対しても，$\sigma q w \underset{\delta}{\Longrightarrow} \sigma' q' w'$ となる計算状況 $\sigma' q' w' \in Z^*Q\Sigma^*$ が高々 1 つであることをいう．PDA $\mathcal{A}$ が文字列 $w \in \Sigma^*$ を**受理する** (accept) とは，ある受理状態 $q \in F$ とスタックの内容 $\sigma \in Z^*$ について

$$q_0 w \overset{*}{\underset{\delta}{\Longrightarrow}} \sigma q$$

となることをいう．Σ 上の言語

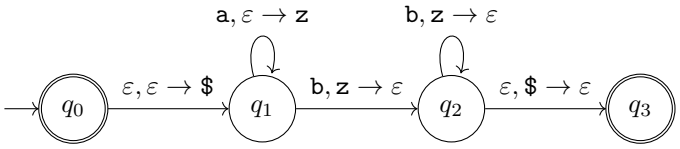
$$\mathrm{L}(\mathcal{A}) := \{\, w \in \Sigma^* \mid \mathcal{A} \text{ は } w \text{ を受理する} \,\}$$

のことを $\mathcal{A}$ が認識する言語という．Σ 上の言語 $L \subseteq \Sigma^*$ が $L = \mathrm{L}(\mathcal{A})$ をみたすとき，L は $\mathcal{A}$ によって認識されるという．

例 2.34.　(1) PDA $\mathcal{A}_1 = (Q, \Sigma, Z, \delta, q_0, F)$ を

$$
\begin{aligned}
Q &:= \{q_0, q_1, q_2, q_3\},\\
\Sigma &:= \{\mathtt{a}, \mathtt{b}\},\\
Z &:= \{\mathtt{z}, \mathtt{\$}\},\\
\delta &:= \{(q_0, \mathtt{\$}q_1), (q_1\mathtt{a}, \mathtt{z}q_1), (\mathtt{z}q_1\mathtt{b}, q_2), (\mathtt{z}q_2\mathtt{b}, q_2), (\mathtt{\$}q_2, q_3)\},\\
F &:= \{q_0, q_3\}
\end{aligned}
$$

と定義する．この $\mathcal{A}_1$ に対応する状態遷移図は次のようになる．



ここで，各元 $(\sigma q w, \sigma' q') \in \delta$ に対し，頂点 q から頂点 q' へ $w, \sigma \to \sigma'$ でラベル付けされた辺を引いている．この PDA $\mathcal{A}_1$ が認識する言語は $\mathrm{L}(\mathcal{A}_1) = \{\, \mathtt{a}^n\mathtt{b}^n \in \Sigma^* \mid n \geq 0 \,\}$ である．ここで，$\mathtt{aabb} \in \Sigma^*$ に対して導出

$$
\begin{aligned}
q_0\mathtt{aabb} &\underset{\delta}{\Longrightarrow} \mathtt{\$}q_1\mathtt{aabb}\\
&\underset{\delta}{\Longrightarrow} \mathtt{\$z}q_1\mathtt{abb}\\
&\underset{\delta}{\Longrightarrow} \mathtt{\$zz}q_1\mathtt{bb}\\
&\underset{\delta}{\Longrightarrow} \mathtt{\$z}q_2\mathtt{b}\\
&\underset{\delta}{\Longrightarrow} \mathtt{\$}q_2\\
&\underset{\delta}{\Longrightarrow} q_3
\end{aligned}
$$

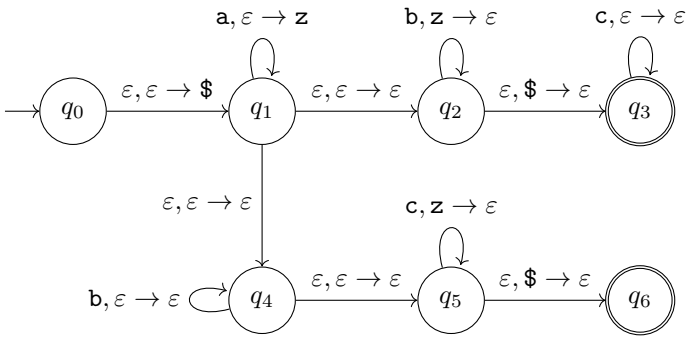
があることから，$\mathcal{A}_1$ が $\mathtt{aabb}$ を受理することがわかる．$\mathcal{A}_1$ は $\mathtt{a}$ が来たらスタックに $\mathtt{z}$ をプッシュして，$\mathtt{b}$ が来たらスタックから $\mathtt{z}$ をポップする．$\mathtt{\$}$はスタックの “底” を表す記号であり，これにより $\mathtt{a}$ と $\mathtt{b}$ が同数であるかどうかを判定している．

(2) PDA $\mathcal{A}_2 = (Q, \Sigma, Z, \delta, q_0, F)$ を

$$
\begin{aligned}
Q &:= \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\},\\
\Sigma &:= \{\mathtt{a}, \mathtt{b}, \mathtt{c}\},\\
Z &:= \{\mathtt{z}, \mathtt{\$}\},\\
\delta &:= \left\{ \begin{matrix} (q_0, \mathtt{\$}q_1), (q_1\mathtt{a}, \mathtt{z}q_1), (q_1, q_2), (\mathtt{z}q_2\mathtt{b}, q_2), (\mathtt{\$}q_2, q_3), (q_3\mathtt{c}, q_3),\\ (q_1, q_4), (q_4\mathtt{b}, q_4), (q_4, q_5), (\mathtt{z}q_5\mathtt{c}, q_5), (\mathtt{\$}q_5, q_6) \end{matrix} \right\},\\
F &:= \{q_3, q_6\}
\end{aligned}
$$

[*8] スタックアルファベットは Γ で表されることが多いが，本稿では Γ をグラフを表す文字として使いたいので，[DW17] に倣って Z を用いている．

と定義する．この $\mathcal{A}_2$ に対応する状態遷移図は次のようになる．



この PDA $\mathcal{A}_2$ が認識する言語は $\mathrm{L}(\mathcal{A}_2) = \{\, \mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \in \Sigma^* \mid i = j \vee i = k \,\}$ である．$\mathcal{A}_2$ は a と同数の z をスタックにプッシュした後，$i = j$ であるか $i = k$ であるかを非決定性を利用して “推測” し，$i = j$ だと思ったら q_2 へ遷移し，$i = k$ だと思ったら q_4 へと遷移する．ここで，$\mathtt{aabcc} \in \Sigma^*$ に対して導出

$$
\begin{aligned}
q_0\mathtt{aabcc} &\underset{\delta}{\Longrightarrow} \mathtt{\$}q_1\mathtt{aabcc} \\
&\underset{\delta}{\Longrightarrow} \mathtt{\$z}q_1\mathtt{abcc} \\
&\underset{\delta}{\Longrightarrow} \mathtt{\$zz}q_1\mathtt{bcc} \\
&\underset{\delta}{\Longrightarrow} \mathtt{\$zz}q_4\mathtt{bcc} \\
&\underset{\delta}{\Longrightarrow} \mathtt{\$zz}q_4\mathtt{cc} \\
&\underset{\delta}{\Longrightarrow} \mathtt{\$zz}q_5\mathtt{cc} \\
&\underset{\delta}{\Longrightarrow} \mathtt{\$z}q_5\mathtt{c} \\
&\underset{\delta}{\Longrightarrow} \mathtt{\$}q_5 \\
&\underset{\delta}{\Longrightarrow} q_6
\end{aligned}
$$

があることから，$\mathcal{A}_2$ が aabcc を受理することがわかる．

注意 2.35. NFA と DFA の場合とは異なり，PDA は DPDA よりも真に強い表現力を持っている．このことは付録??で確認する．

2.3.4 文脈自由文法とプッシュダウンオートマトンの等価性

ここでは文脈自由文法と PDA が同等の表現能力を持つことを証明する．まず，PDA が文脈自由文法による文字列の生成を模倣するのに十分な能力を持っていることから確認する．

定理 2.36. どんな文脈自由文法 G に対しても，$\mathrm{L}(\mathcal{A}) = \mathrm{L}(G)$ となる PDA $\mathcal{A}$ を構成することができる．

証明. 文脈自由文法 $G = (V, \Sigma, P, S)$ に対し，PDA $\mathcal{A} = (Q, \Sigma, Z, \delta, q_0, F)$ を

$$
\begin{aligned}
Q &:= \{q_0, q_1, q_2\}, \\
Z &:= V \cup \Sigma \cup \{\mathtt{\$}\}, \\
\delta &:= \{(q_0, \mathtt{\$}q_1)\} \cup \{\, (q_1 a, a q_1) \mid a \in \Sigma \,\} \cup \{\, (\alpha q_1, A q_1) \mid A \to \alpha \in P \,\} \cup \{(\mathtt{\$}S q_1, q_2)\}, \\
F &:= \{q_2\}
\end{aligned}
$$

と定義する．

($\mathrm{L}(G) \subseteq \mathrm{L}(\mathcal{A})$ であること)．　$w \in \mathrm{L}(G)$ をとると，補題 2.27 と同様にして**最右導出** (rightmost derivation) $S = \beta_0 \underset{P}{\Longrightarrow} \beta_1 \underset{P}{\Longrightarrow} \cdots \underset{P}{\Longrightarrow} \beta_n = w$ がとれる．このとき各 $i = 0, 1, \ldots, n-1$ に対し，$\beta_i = \gamma_i A_i w_i$ ($\gamma_i \in (V \cup \Sigma)^*, A_i \in V, w_i \in \Sigma^*$) と書ける．導出の各段階において $A_i \to \alpha_i \in P$ という規則が適用されるとすると，$\beta_{i+1} = \gamma_{i+1} A_{i+1} w_{i+1} = \gamma_i \alpha_i w_i$ である．各 $i = n-1, \ldots, 1, 0$ に対して帰納的に $q_0 w \overset{*}{\underset{\delta}{\Longrightarrow}} \$\gamma_i \alpha_i q_1 w_i$ となることを示す．$i = n-1$ のときは $w = \gamma_{n-1}\alpha_{n-1}w_{n-1}$ だから

$$q_0 w \underset{\delta}{\Longrightarrow} \$q_1 w \overset{*}{\underset{\delta}{\Longrightarrow}} \$\gamma_{n-1}\alpha_{n-1}q_1 w_{n-1}$$

となるのでよい．$q_0 w \overset{*}{\underset{\delta}{\Longrightarrow}} \$\gamma_{i+1}\alpha_{i+1}q_1 w_{i+1}$ のとき，$(\alpha_i q_1, A_i q_1) \in \delta$ より

$$\$\gamma_{i+1}\alpha_{i+1}q_1 w_{i+1} \underset{\delta}{\Longrightarrow} \$\gamma_{i+1}A_{i+1}q_1 w_{i+1}$$

ここで $\gamma_{i+1}A_{i+1}w_{i+1} = \beta_i = \gamma_i\alpha_i w_i$ かつ，最右導出であることより A_{i+1} は $\gamma_i\alpha_i$ 中に出現するから

$$\overset{*}{\underset{\delta}{\Longrightarrow}} \$\gamma_i \alpha_i q_1 w_i$$

となる．よって $i = 0$ について

$$q_0 w \overset{*}{\underset{\delta}{\Longrightarrow}} \$\gamma_0\alpha_0 q_1 w_0 = \$\alpha_0 q_1 \underset{\delta}{\Longrightarrow} \$S q_1 \underset{\delta}{\Longrightarrow} \$q_2 \in F$$

となり，$w \in \mathrm{L}(\mathcal{A})$ を得る．

($\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(G)$ であること)．　$\mathcal{A}$ が文字列 $w \in \Sigma^*$ を受理する際の導出は $\mathcal{A}$ の作り方より

$$q_0 w \underset{\delta}{\Longrightarrow} \$q_1 w = \alpha_1 \underset{\delta}{\Longrightarrow} \alpha_2 \underset{\delta}{\Longrightarrow} \cdots \underset{\delta}{\Longrightarrow} \alpha_{n-1} \underset{\delta}{\Longrightarrow} \alpha_n = \$S q_1 \underset{\delta}{\Longrightarrow} q_2$$

の形しかありえない．ここでモノイド準同型写像 $h\colon (Z \cup Q)^* \to Z^*$ を，Q の元と $\$$ を削除する写像，すなわち

$$\begin{array}{llll} h(A) := A & (A \in V \text{ のとき}), & h(a) := a & (a \in \Sigma \text{ のとき}), \\ h(q) := \varepsilon & (q \in Q \text{ のとき}), & h(\$) := \varepsilon & \end{array}$$

と定める．このとき

$$S = h(\alpha_n) \overset{\leq 1}{\underset{P}{\Longrightarrow}} h(\alpha_{n-1}) \overset{\leq 1}{\underset{P}{\Longrightarrow}} \cdots \overset{\leq 1}{\underset{P}{\Longrightarrow}} h(\alpha_2) \overset{\leq 1}{\underset{P}{\Longrightarrow}} h(\alpha_1) = w$$

は G による導出 $S \overset{*}{\underset{P}{\Longrightarrow}} w$ を与えている．　□

例 2.37.　文脈自由文法 $G = (V, \Sigma, P, S)$ を

$$\begin{aligned} V &:= \{S, A\}, \\ \Sigma &:= \{\mathtt{a}, \mathtt{b}\}, \\ P &:= \begin{cases} S \to \mathtt{a} \mid A S, \\ A \to \mathtt{a} \mid \mathtt{b} A \end{cases} \end{aligned}$$

とするとき，定理 2.36 の証明に沿って G に対応する PDA $\mathcal{A} = (Q, \Sigma, Z, \delta, q_0, F)$ を構成すると

$$\begin{aligned} Q &= \{q_0, q_1, q_2\}, \\ Z &= \{\mathtt{a}, \mathtt{b}, S, A, \$\}, \\ \delta &= \{(q_0, \$q_1), (q_1\mathtt{a}, \mathtt{a}q_1), (q_1\mathtt{b}, \mathtt{b}q_1), (\mathtt{a}q_1, Sq_1), (ASq_1, Sq_1), (\mathtt{a}q_1, Aq_1), (\mathtt{b}Aq_1, Aq_1), (\$Sq_1, q_2)\}, \\ F &= \{q_2\} \end{aligned}$$

となる．このとき，G の最右導出

$$S \underset{P}{\Longrightarrow} AS \underset{P}{\Longrightarrow} AAS \underset{P}{\Longrightarrow} AA\mathtt{a} \underset{P}{\Longrightarrow} A\mathtt{b}A\mathtt{a} \underset{P}{\Longrightarrow} A\mathtt{baa} \underset{P}{\Longrightarrow} \mathtt{abaa}$$

に対応する $\mathcal{A}$ の導出は

$$\begin{aligned} q_0\mathtt{abaa} &\underset{\delta}{\Longrightarrow} \mathtt{\$}q_1\mathtt{abaa} \underset{\delta}{\Longrightarrow} \mathtt{\$a}q_1\mathtt{baa} \underset{\delta}{\Longrightarrow} \mathtt{\$}Aq_1\mathtt{baa} \underset{\delta}{\Longrightarrow} \mathtt{\$}A\mathtt{b}q_1\mathtt{aa} \underset{\delta}{\Longrightarrow} \mathtt{\$}A\mathtt{ba}q_1\mathtt{a} \underset{\delta}{\Longrightarrow} \mathtt{\$}A\mathtt{b}Aq_1\mathtt{a} \\ &\underset{\delta}{\Longrightarrow} \mathtt{\$}AAq_1\mathtt{a} \underset{\delta}{\Longrightarrow} \mathtt{\$}AA\mathtt{a}q_1 \underset{\delta}{\Longrightarrow} \mathtt{\$}AASq_1 \underset{\delta}{\Longrightarrow} \mathtt{\$}ASq_1 \underset{\delta}{\Longrightarrow} \mathtt{\$}Sq_1 \underset{\delta}{\Longrightarrow} q_2 \end{aligned}$$

となる．

次に，PDA による文字列の受理を模倣する文脈自由文法を構成する．こちらの構成は先程よりやや複雑である．話を簡単にするために，PDA を一度に 1 文字ずつしか入力を読み込んだりスタックを操作したりできないようなものに変形できることを示しておく．

補題 2.38. どんな PDA $\mathcal{A} = (Q, \Sigma, Z, \delta, q_0, F)$ に対しても，PDA $\mathcal{B} = (Q', \Sigma, Z, \delta', q_0, F)$ で，2 条件

- $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})$,
- $\delta' \subseteq (Q' \times Q') \cup (ZQ' \times Q') \cup (Q' \times ZQ') \cup (Q'\Sigma \times Q')$[*9]

をみたすものを構成することができる．

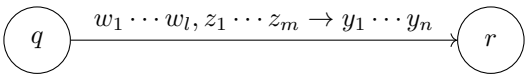
ここで $Q' \times Q', ZQ' \times Q', Q' \times ZQ', Q'\Sigma \times Q'$ の元はそれぞれ ε 遷移，1 文字ポップする遷移，1 文字プッシュする遷移，1 文字読み込む遷移であることに注意する．

証明. PDA $\mathcal{A} = (Q, \Sigma, Z, \delta, q_0, F)$ をとる．δ 中の各遷移を，1 文字ずつ操作する小さな遷移に分解することを考える．新たな PDA $\mathcal{B} = (Q', \Sigma, Z, \delta', q_0, F)$ を

$$Q' := Q \cup \left\{ \begin{array}{c} [q^t], [z_l q^t], \dots, [z_1 \cdots z_m q^t], \\ [z_1 \cdots z_m q^t w_1], \dots, [z_1 \cdots z_m q w_1 \cdots w_l], \\ [y_1 \cdots y_n \bar{r}^t], \dots, [y_n \bar{r}^t], [\bar{r}^t] \end{array} \middle| \, t = (z_1 \cdots z_m q w_1 \cdots w_l, y_1 \cdots y_n r) \in \delta \right\},$$

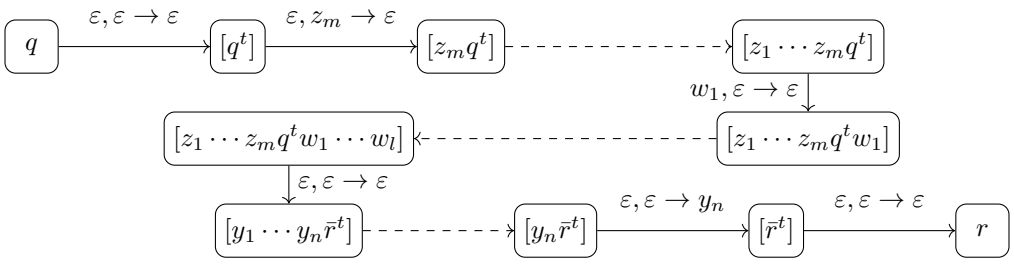
$$\delta' := \left\{ \begin{array}{c} (q, [q^t]), (z_j[z_{j+1} \cdots z_m q^t], [z_j z_{j+1} \cdots z_m q^t]), \\ ([z_1 \cdots z_m q^t w_1 \cdots w_i] w_{i+1}, [z_1 \cdots z_m q^t w_1 \cdots w_i w_{i+1}]), \\ ([z_1 \cdots z_m q^t w_1 \cdots w_l], [y_1 \cdots y_n \bar{r}^t]), \\ ([y_k y_{k+1} \cdots y_n \bar{r}^t], y_k[y_{k+1} \cdots y_n \bar{r}^t]), ([\bar{r}^t], r) \end{array} \middle| \begin{array}{c} t = (z_1 \cdots z_m q w_1 \cdots w_l, y_1 \cdots y_n r) \in \delta, \\ 1 \le j \le m,\ 0 \le i < l,\ 1 \le k \le n \end{array} \right\}$$

とおく．すなわち，$\mathcal{B}$ は $\mathcal{A}$ における



[*9] むしろ PDA の定義にこれに類似した条件を採用している文献の方が多いかもしれない．

という遷移を



という遷移の連なりに変換したものである．作り方から条件 $\delta' \subseteq (Q' \times Q') \cup (ZQ' \times Q') \cup (Q' \times ZQ') \cup (Q'\Sigma \times Q')$ は明らかに成り立っている．

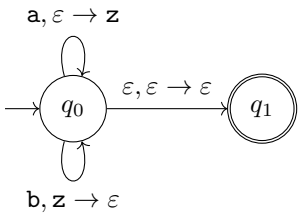
($\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{B})$ であること) 任意に $w \in \mathrm{L}(\mathcal{A})$ をとると，ある $\sigma \in Z^*$ と $q \in F$ に対し導出 $q_0 w = c_0 \underset{\delta}{\Longrightarrow} c_1 \underset{\delta}{\Longrightarrow} \cdots \underset{\delta}{\Longrightarrow} c_n = \sigma q$ $(c_i \in Z^* Q \Sigma^*)$ がとれる．このとき各 i に対し，構成から $c_i \underset{\delta}{\Longrightarrow} c_{i+1}$ ならば $c_i \overset{*}{\underset{\delta'}{\Longrightarrow}} c_{i+1}$ であるので，導出 $q_0 w \overset{*}{\underset{\delta'}{\Longrightarrow}} \sigma q$ が得られ，$w \in \mathrm{L}(\mathcal{B})$ がわかる．

($\mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{A})$ であること) 任意に $w \in \mathrm{L}(\mathcal{B})$ をとると，ある $\sigma \in Z^*$ と $q \in F$ に対し導出 $q_0 w = c_0 \underset{\delta'}{\Longrightarrow} c_1 \underset{\delta'}{\Longrightarrow} \cdots \underset{\delta'}{\Longrightarrow} c_n = \sigma q$ $(c_i \in Z^* Q' \Sigma^*)$ がとれる．ここで $i < j$ について $c_i, c_j \in Z^* Q \Sigma^*$ かつ，どの $k \in \{i, i+1, \ldots, j\}$ についても $c_k \in Z^*(Q' - Q)\Sigma^*$ となっているとする．このとき δ' の定め方より，ある $t \in \delta$ が存在し，各 k について $c_k = [\sigma_k q^t w_k]$ $(\sigma_k \in Z^*, w_k \in \Sigma^*)$ という形をしている．よって $c_i \underset{\delta}{\Longrightarrow} c_j$ となるから導出 $q_0 w \overset{*}{\underset{\delta}{\Longrightarrow}} \sigma q$ が得られ，$w \in \mathrm{L}(\mathcal{A})$ がわかる． □

例 2.39. PDA $\mathcal{A} = (Q, \Sigma, Z, \delta, q_0, F)$ を

$$
\begin{aligned}
Q &:= \{q_0, q_1\}, \\
\Sigma &:= \{\mathtt{a}, \mathtt{b}\}, \\
Z &:= \{\mathtt{z}\}, \\
\delta &:= \{t_1 = (q_0\mathtt{a}, \mathtt{z}q_0), t_2 = (\mathtt{z}q_0\mathtt{b}, q_0), t_3 = (q_0, q_1)\}, \\
F &:= \{q_1\}
\end{aligned}
$$

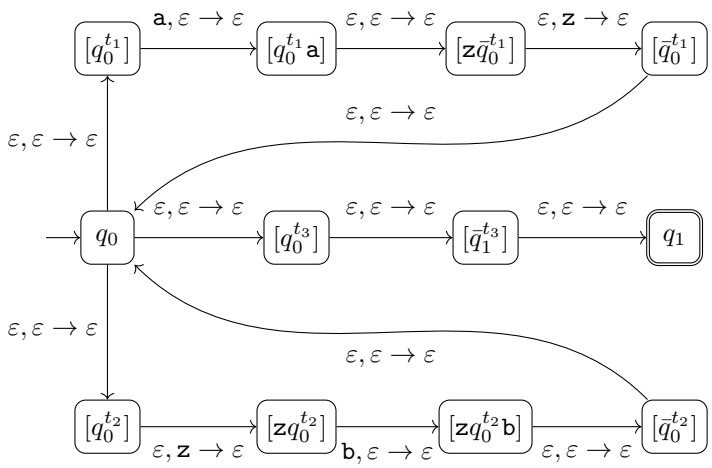
とおく．この $\mathcal{A}$ に対応する状態遷移図は



である．文字列 $w \in \Sigma^*$ に含まれる文字 $a \in \Sigma$ の個数を $|w|_a$ と書くと，この $\mathcal{A}$ が認識する言語は $\mathrm{L}(\mathcal{A}) = \{\, w \in \Sigma^* \mid \forall v \preccurlyeq w \; [|v|_{\mathtt{a}} \geq |v|_{\mathtt{b}}] \,\}$ である．補題 2.38 に従って PDA $\mathcal{B} = (Q', \Sigma, Z, \delta', q_0, F)$ を構成すると

$$
\begin{aligned}
Q' &= \{q_0, q_1, [q_0^{t_1}], [q_0^{t_1}\mathtt{a}], [\mathtt{z}\bar{q}_0^{t_1}], [\bar{q}_0^{t_1}], [q_0^{t_2}], [\mathtt{z}q_0^{t_2}], [\mathtt{z}q_0^{t_2}\mathtt{b}], [\bar{q}_0^{t_2}], [q_0^{t_3}], [\bar{q}_1^{t_3}]\}, \\
\delta' &= \left\{ \begin{array}{c}
(q_0, [q_0^{t_1}]), ([q_0^{t_1}]\mathtt{a}, [q_0^{t_1}\mathtt{a}]), ([q_0^{t_1}\mathtt{a}], [\mathtt{z}\bar{q}_0^{t_1}]), ([\mathtt{z}\bar{q}_0^{t_1}], \mathtt{z}[\bar{q}_0^{t_1}]), ([\bar{q}_0^{t_1}], q_0), \\
(q_0, [q_0^{t_2}]), (\mathtt{z}[q_0^{t_2}], [\mathtt{z}q_0^{t_2}]), ([\mathtt{z}q_0^{t_2}]\mathtt{b}, [\mathtt{z}q_0^{t_2}\mathtt{b}]), ([\mathtt{z}q_0^{t_2}\mathtt{b}], [\bar{q}_0^{t_2}]), ([\bar{q}_0^{t_2}], q_0]), \\
(q_0, [q_0^{t_3}]), ([q_0^{t_3}], [\bar{q}_1^{t_3}]), ([\bar{q}_1^{t_3}], q_1)
\end{array} \right\}
\end{aligned}
$$

となり，対応する状態遷移図は



となる．

証明を簡潔にするために，もう一つだけ簡単な変形を施しておく．

補題 2.40. どんな PDA $\mathcal{A} = (Q, \Sigma, Z, \delta, q_0, F)$ に対しても，PDA $\mathcal{B} = (Q', \Sigma, Z, \delta', q_0, F')$ で，4 条件

- $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})$,
- $\delta' \subseteq (Q' \times Q') \cup (ZQ' \times Q') \cup (Q' \times ZQ') \cup (Q'\Sigma \times Q')$
- 受理状態がただ一つ，すなわちある状態 $q_{\mathrm{accept}} \in Q$ について $F' = \{q_{\mathrm{accept}}\}$,
- 入力を受理するならばスタックが空の状況で受理できる，つまり $\mathrm{L}(\mathcal{B}) = \{\, w \in \Sigma^* \mid q_0 w \overset{*}{\underset{\delta}{\Longrightarrow}} q_{\mathrm{accept}} \,\}$

をみたすものを構成することができる．

証明. 補題 2.38 より $\mathcal{A}$ は前半の 2 条件をみたすとしてよい．このとき PDA $\mathcal{B} = (Q', \Sigma, Z, \delta', q_0, F')$ を

$$
\begin{aligned}
Q' &:= Q \cup \{q_{\mathrm{accept}}\} \quad (\text{ただし } q_{\mathrm{accept}} \notin Q),\\
\delta' &:= \delta \cup \{\, (q, q_{\mathrm{accept}}) \mid q \in F \,\} \cup \{\, (zq_{\mathrm{accept}}, q_{\mathrm{accept}}) \mid z \in Z \,\},\\
F' &:= \{q_{\mathrm{accept}}\}
\end{aligned}
$$

とおけば，$\mathcal{B}$ は所望の条件をみたす． □

それでは定理の証明に入ろう．

定理 2.41. どんな PDA $\mathcal{A}$ に対しても，$\mathrm{L}(G) = \mathrm{L}(\mathcal{A})$ となる文脈自由文法 G を構成することができる．

証明. 補題 2.40 より，PDA $\mathcal{A} = (Q, \Sigma, Z, \delta, q_0, F)$ は

- $\delta \subseteq (Q \times Q) \cup (ZQ \times Q) \cup (Q \times ZQ) \cup (Q\Sigma \times Q)$,
- 受理状態がただ一つ，すなわちある状態 $q_{\mathrm{accept}} \in Q$ について $F = \{q_{\mathrm{accept}}\}$,
- 入力を受理するならばスタックが空の状況で受理できる，つまり $\mathrm{L}(\mathcal{A}) = \{\, w \in \Sigma^* \mid q_0 w \overset{*}{\underset{\delta}{\Longrightarrow}} q_{\mathrm{accept}} \,\}$

という条件をみたすとしてよい．文脈自由文法 $G = (V, \Sigma, P, S)$ を

$$V := \{\, A_{p,q} \mid p, q \in Q \,\},$$

$$
\begin{aligned}
P := {} & \{\, A_{p,p} \to \varepsilon \mid p \in Q \,\} \cup \{\, A_{p,q} \to A_{p,r}A_{r,q} \mid p,q,r \in Q \,\} \\
& \cup \{\, A_{p,q} \to \varepsilon \mid p,q \in Q, (p,q) \in \delta \,\} \\
& \cup \{\, A_{p,q} \to A_{r,s} \mid p,q,r,s \in Q, z \in Z, (p,zr), (zs,q) \in \delta \,\} \\
& \cup \{\, A_{p,q} \to aA_{r,q} \mid p,q,r \in Q, a \in \Sigma, (pa,r) \in \delta \,\}, \\
S := {} & A_{q_0,q_{\text{accept}}}
\end{aligned}
$$

と定義する．P は明らかに有限集合であることに注意する．この文法は次の条件をみたすように定義されている．

主張 2.42. 各変数 $A_{p,q} \in V$ と文字列 $w \in \Sigma^*$ に対し，

$$\left[A_{p,q} \overset{*}{\underset{P}{\Longrightarrow}} w\right] \iff \left[pw \overset{*}{\underset{\delta}{\Longrightarrow}} q\right].$$

証明. ($\Longrightarrow$) 導出 $A_{p,q} \overset{*}{\underset{P}{\Longrightarrow}} w$ の長さに関する帰納法で示す．$A_{p,q} \underset{P}{\Longrightarrow} w$ のとき，P の定め方から $p = q$ かつ $w = \varepsilon$，または $(p,q) \in \delta$ かつ $w = \varepsilon$ しかありえないので，前者ならば $pw = p \overset{0}{\underset{\delta}{\Longrightarrow}} p = q$，後者ならば $pw = p \overset{1}{\underset{\delta}{\Longrightarrow}} q$ である．$A_{p,q} \underset{P}{\Longrightarrow} A_{p,r}A_{r,q} \overset{*}{\underset{P}{\Longrightarrow}} w$ のとき，$A_{p,r} \overset{*}{\underset{P}{\Longrightarrow}} w_1, A_{r,q} \overset{*}{\underset{P}{\Longrightarrow}} w_2$ となるような分割 $w = w_1w_2$ がある．よって帰納法の仮定より $pw_1 \overset{*}{\underset{\delta}{\Longrightarrow}} r, rw_2 \overset{*}{\underset{\delta}{\Longrightarrow}} q$ となるので $pw = pw_1w_2 \overset{*}{\underset{\delta}{\Longrightarrow}} rw_2 \overset{*}{\underset{\delta}{\Longrightarrow}} q$ を得る．次に $A_{p,q} \underset{P}{\Longrightarrow} A_{r,s} \overset{*}{\underset{P}{\Longrightarrow}} w$ のとき，P の定義から $(p,zr), (zs,q) \in \delta$ となるような $z \in Z$ がとれる．よって帰納法の仮定から $rw \overset{*}{\underset{\delta}{\Longrightarrow}} s$ であるので $pw \underset{\delta}{\Longrightarrow} zrw \overset{*}{\underset{\delta}{\Longrightarrow}} zs \underset{\delta}{\Longrightarrow} q$ を得る．最後に $A_{p,q} \underset{P}{\Longrightarrow} aA_{r,q} \overset{*}{\underset{P}{\Longrightarrow}} w$ $(a \in \Sigma, (pa,r) \in \delta)$ のとき，$w = aw'$ とすると帰納法の仮定より $rw' \overset{*}{\underset{\delta}{\Longrightarrow}} q$ であるので $pw = paw' \underset{\delta}{\Longrightarrow} rw' \overset{*}{\underset{\delta}{\Longrightarrow}} q$ を得る．

($\Longleftarrow$) 導出 $pw \overset{*}{\underset{\delta}{\Longrightarrow}} q$ の長さに関する帰納法で示す．長さが 0 のときは $pw = q$ より $p = q, w = \varepsilon$ しかありえないから $A_{p,q} = A_{p,p} \underset{P}{\Longrightarrow} \varepsilon = w$ よりよい．長さが 1 以上のとき，導出 $pw \overset{*}{\underset{\delta}{\Longrightarrow}} q$ において pw に最初に適用される δ の規則によって場合分けする．pw はスタックが空の計算状況であるから，δ に関する仮定と合わせて，以下の 3 通りのいずれかになる．

- ($(p,r) \in \delta$ の形のとき) $pw \underset{\delta}{\Longrightarrow} rw \overset{*}{\underset{\delta}{\Longrightarrow}} q$ であるから，帰納法の仮定より $A_{r,q} \overset{*}{\underset{P}{\Longrightarrow}} w$ である．$(p,r) \in \delta$ から $A_{p,r} \to \varepsilon \in P$ であることと合わせて $A_{p,q} \underset{P}{\Longrightarrow} A_{p,r}A_{r,q} \underset{P}{\Longrightarrow} A_{r,q} \overset{*}{\underset{P}{\Longrightarrow}} w$ を得る．
- ($(p,zr) \in \delta$ の形のとき) $pw \underset{\delta}{\Longrightarrow} zrw \overset{*}{\underset{\delta}{\Longrightarrow}} q$ となっているが，ここで q はスタックが空の計算状況であるから，最初にスタックの底にプッシュされた z が初めてポップされる瞬間がなければならない．よってその瞬間に適用される規則を $(zs,q') \in \delta$ とすると，ある分解 $w = uv$ について $pw \underset{\delta}{\Longrightarrow} zrw = zruv \overset{*}{\underset{\delta}{\Longrightarrow}} zsv \underset{\delta}{\Longrightarrow} q'v \overset{*}{\underset{\delta}{\Longrightarrow}} q$ となる．$zsv \underset{\delta}{\Longrightarrow} q'v$ が z が初めてポップされる瞬間であることより，$ru \overset{*}{\underset{\delta}{\Longrightarrow}} s$ が成り立つ (一般には $zruv \overset{*}{\underset{\delta}{\Longrightarrow}} zsv$ であるからといって $ruv \overset{*}{\underset{\delta}{\Longrightarrow}} sv$ とは限らないことに注意)．よって帰納法の仮定より $A_{r,s} \overset{*}{\underset{P}{\Longrightarrow}} u$ かつ $A_{q',q} \overset{*}{\underset{P}{\Longrightarrow}} v$ であるので $A_{p,q} \underset{P}{\Longrightarrow} A_{p,q'}A_{q',q} \underset{P}{\Longrightarrow} A_{r,s}A_{q',q} \overset{*}{\underset{P}{\Longrightarrow}} uA_{q',q} \overset{*}{\underset{P}{\Longrightarrow}} uv = w$ を得る．
- ($(pa,r) \in \delta$ の形のとき) $w = aw'$ $(a \in \Sigma, w' \in \Sigma^*)$ の形に分解でき，$pw = paw' \underset{\delta}{\Longrightarrow} rw' \overset{*}{\underset{\delta}{\Longrightarrow}} q$ となっている．よって帰納法の仮定より $A_{r,q} \overset{*}{\underset{P}{\Longrightarrow}} w'$ であるので $A_{p,q} \underset{P}{\Longrightarrow} aA_{r,q} \overset{*}{\underset{P}{\Longrightarrow}} aw' = w$ を得る． □

よって

$$
\begin{aligned}
\mathrm{L}(G) &= \{\, w \in \Sigma^* \mid A_{q_0,q_{\text{accept}}} \overset{*}{\underset{P}{\Longrightarrow}} w \,\} && (S = A_{q_0,q_{\text{accept}}} \text{ より}) \\
&= \{\, w \in \Sigma^* \mid q_0w \overset{*}{\underset{\delta}{\Longrightarrow}} q_{\text{accept}} \,\} && (\text{主張 2.42 より})
\end{aligned}
$$

$= \mathrm{L}(\mathcal{A})$　　($\mathcal{A}$ に関する最初の仮定より)

となる．□

以上をまとめると，次の系が得られる．

系 2.43.　Σ 上の言語 $L \subseteq \Sigma^*$ に対し，以下の 2 条件は同値である．

(1) L はある文脈自由文法によって生成される．
(2) L はある PDA によって認識される．

証明. 定理 2.36 と定理 2.41 より．□

2.4　言語クラスの閉包性

これまで，個別の言語 $L \subseteq \Sigma^*$ が正則言語であるとか文脈自由言語であるとかいった条件について調べてきた．ここでは個々の言語ではなく，「正則言語全体のクラス」や「文脈自由言語全体のクラス」などが持つ性質を調べる．特に「言語クラス $\mathcal{L}$ に属す言語 L にある特定の操作を施して得られる言語が再び $\mathcal{L}$ に属す」という種類の性質 (閉包性) について考察する．

定義 2.44.　正則言語全体のクラス，文脈自由言語全体のクラスをそれぞれ記号 $\mathsf{REG}, \mathsf{CFL}$ で表す．すなわち

$$\mathsf{REG} = \{\, L \mid \exists \Sigma\, [\Sigma \text{ は有限アルファベット} \wedge L \subseteq \Sigma^* \wedge L \text{ は正則言語}] \,\},$$
$$\mathsf{CFL} = \{\, L \mid \exists \Sigma\, [\Sigma \text{ は有限アルファベット} \wedge L \subseteq \Sigma^* \wedge L \text{ は文脈自由言語}] \,\}$$

である[*10]．定義より $\mathsf{REG} \subseteq \mathsf{CFL}$ が成り立っている．

2.4.1　Boole 演算と正則演算

言語に対する操作として，まず Boole 演算と連接，Kleene 閉包について調べよう．Boole 演算は和集合・共通部分・補集合という 3 つの操作からなる．**正則演算** (regular operations) とは，正則表現の定義 2.12 に現れる和集合・連接・Kleene 閉包の 3 つを指す．すなわち，次のように定義される操作である．

定義 2.45 (連接，Kleene 閉包).　$L_1, L_2, L \subseteq \Sigma^*$ を言語とするとき，

- L_1 と L_2 の**連接** (concatenation) を $L_1 \cdot L_2 := \{\, uv \mid u \in L_1, v \in L_2 \,\}$,
- L の **Kleene 閉包** (Kleene closure, Kleene star) を $L^* := \{\, w_1 w_2 \cdots w_n \mid n \geq 0, \forall i \in \{1, \ldots, n\}\, [w_i \in L] \,\}$

と定義する．

まず正則言語のクラス REG について述べる．REG は非常に性質がよいクラスであり，実際，上で挙げたすべての操作について閉じている．

定理 2.46.　正則言語のクラス REG について，以下が成り立つ．

(1) REG は和集合について閉じている．すなわち，$L_1, L_2 \in \mathsf{REG}$ かつ $L_1, L_2 \subseteq \Sigma^*$ ならば $L_1 \cup L_2 \in \mathsf{REG}$ である．

[*10] これら $\mathsf{REG}, \mathsf{CFL}$ は Σ のとり方に制限がないため真のクラスになるが，必要ならアルファベットを $\Sigma \subseteq \mathbb{N}$ に制限することにより，$\mathsf{REG}, \mathsf{CFL}$ を集合であると考えてもよい．

(2) REG は共通部分について閉じている．すなわち，$L_1, L_2 \in \mathsf{REG}$ かつ $L_1, L_2 \subseteq \Sigma^*$ ならば $L_1 \cap L_2 \in \mathsf{REG}$ である．
(3) REG は補集合について閉じている．すなわち，$L \in \mathsf{REG}$ かつ $L \subseteq \Sigma^*$ ならば $\Sigma^* - L \in \mathsf{REG}$ である．
(4) REG は連接について閉じている，すなわち $L_1, L_2 \in \mathsf{REG}$ かつ $L_1, L_2 \subseteq \Sigma^*$ ならば $L_1 \cdot L_2 \in \mathsf{REG}$ である．
(5) REG は Kleene 閉包について閉じている，すなわち $L \in \mathsf{REG}$ かつ $L \subseteq \Sigma^*$ ならば $L^* \in \mathsf{REG}$ である．

証明. (1)，(4)，(5) について，$L_1, L_2, L \subseteq \Sigma^*$ を正則言語とすると，ある正則表現 E_1, E_2, E によって $L_1 = \mathrm{L}(E_1), L_2 = \mathrm{L}(E_2), L = \mathrm{L}(E)$ と表せる．よって $L_1 \cup L_2 = \mathrm{L}(E_1) \cup \mathrm{L}(E_2) = \mathrm{L}(E_1 \cup E_2) \in \mathsf{REG}$，$L_1 \cdot L_2 = \mathrm{L}(E_1) \cdot \mathrm{L}(E_2) = \mathrm{L}(E_1 \cdot E_2) \in \mathsf{REG}$，$L^* = \mathrm{L}(E)^* = \mathrm{L}(E^*) \in \mathsf{REG}$ を得る．
(3) を示す．$L \subseteq \Sigma^*$ を正則言語とすると，$L = \mathrm{L}(\mathcal{A})$ となる DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ がとれる．このとき DFA $\mathcal{B} = (Q, \Sigma, \delta, q_0, F')$ を $F' := Q - F$ とおけば $\Sigma^* - L = \Sigma^* - \mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B}) \in \mathsf{REG}$ を得る．
(2) は (1)，(3) と de Mogan の法則からわかる． □

文脈自由言語のクラスは正則演算 (和集合・連接・Kleene 閉包) についてのみ閉じている．CFL が共通部分と補集合について閉じていないことは付録**??**で見る．

定理 2.47. 文脈自由言語のクラス CFL について，以下が成り立つ．

(1) CFL は和集合について閉じている．すなわち，$L_1, L_2 \in \mathsf{CFL}$ かつ $L_1, L_2 \subseteq \Sigma^*$ ならば $L_1 \cup L_2 \in \mathsf{CFL}$ である．
(2) CFL は連接について閉じている，すなわち $L_1, L_2 \in \mathsf{CFL}$ かつ $L_1, L_2 \subseteq \Sigma^*$ ならば $L_1 \cdot L_2 \in \mathsf{CFL}$ である．
(3) CFL は Kleene 閉包について閉じている，すなわち $L \in \mathsf{CFL}$ かつ $L \subseteq \Sigma^*$ ならば $L^* \in \mathsf{CFL}$ である．

証明. $L_1, L_2, L \subseteq \Sigma^*$ を文脈自由言語とすると，$L_1 = \mathrm{L}(G_1), L_2 = \mathrm{L}(G_2), L = \mathrm{L}(G)$ となる文脈自由文法 $G_1 = (V_1, \Sigma, P_1, S_1), G_2 = (V_2, \Sigma, P_2, S_2), G = (V, \Sigma, P, S)$ がとれる．$V_1 \cap V_2 = \emptyset$ と仮定してよい．

(1) 文脈自由文法 $G_\cup = (V_\cup, \Sigma, P_\cup, S_\cup)$ を

$$\begin{aligned} V_\cup &:= V_1 \cup V_2 \cup \{S_\cup\} \quad (\text{ただし } S_\cup \notin V_1 \cup V_2), \\ P_\cup &:= P_1 \cup P_2 \cup \{S_\cup \to S_1 \mid S_2\} \end{aligned}$$

とおけば $L_1 \cup L_2 = \mathrm{L}(G_1) \cup \mathrm{L}(G_2) = \mathrm{L}(G_\cup) \in \mathsf{CFL}$ を得る．
(2) 文脈自由文法 $G_\bullet = (V_\bullet, \Sigma, P_\bullet, S_\bullet)$ を

$$\begin{aligned} V_\bullet &:= V_1 \cup V_2 \cup \{S_\bullet\} \quad (\text{ただし } S_\bullet \notin V_1 \cup V_2), \\ P_\bullet &:= P_1 \cup P_2 \cup \{S_\bullet \to S_1 S_2\} \end{aligned}$$

とおけば $L_1 \cdot L_2 = \mathrm{L}(G_1) \cdot \mathrm{L}(G_2) = \mathrm{L}(G_\bullet) \in \mathsf{CFL}$ を得る．
(3) 文脈自由文法 $G_* = (V_*, \Sigma, P_*, S_*)$ を

$$\begin{aligned} V_* &:= V \cup \{S_*\} \quad (\text{ただし } S_* \notin V_1 \cup V_2), \\ P_* &:= P \cup \{S_* \to \varepsilon \mid S S_*\} \end{aligned}$$

とおけば $L^* = \mathrm{L}(G)^* = \mathrm{L}(G_*) \in \mathsf{CFL}$ を得る． □

2.4.2　モノイド準同型による像と逆像

2 つの言語 $L_1 \subseteq \Sigma^*, L_2 \subseteq \Delta^*$ に対し，モノイド準同型 $h\colon \Sigma^* \to \Delta^*$ は L_1 の像 $h(L_1) = \{\, h(w) \mid w \in L_1 \,\} \subseteq \Delta^*$ と L_2 の逆像 $h^{-1}(L_2) = \{\, w \in \Sigma^* \mid h(w) \in L_2 \,\} \subseteq \Sigma^*$ という 2 つの言語を定める．ここでは，正則言語のクラス REG と文脈自由言語のクラス CFL がこれら 2 つの操作について閉じていることを確かめる．4 章以降で群の語の問題を考える際に，逆像をとる操作で閉じた言語クラスを考えることが重要になる．

例 2.48.　$\Sigma := \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, \Delta := \{(,)\}$ とし，$h\colon \Sigma^* \to \Delta^*$ を $h(\mathtt{a}) := (), h(\mathtt{b}) :=)(, h(\mathtt{c}) := \varepsilon$ と定義する．$L_1 := \mathrm{L}((\mathtt{acbc})^*) \subseteq \Sigma^*$ とし $L_2 \subseteq \Delta^*$ を例 2.23 の semi-Dyck 言語とする．このとき，$h(L_1) = \mathrm{L}(((\,)\,)(\,)^*), h^{-1}(L_2) = \mathrm{L}((a \cup c)^*)$ となる[*11]．

モノイド準同型の像をとる操作で閉じていることは，文法を用いて簡単に証明することができる．

定理 2.49.　正則言語のクラス REG と文脈自由言語のクラス CFL はモノイド準同型の像をとる操作で閉じている (closed under homomorphism)．すなわち言語クラス $\mathcal{L} \in \{\mathsf{REG}, \mathsf{CFL}\}$ について，$L \in \mathcal{L}$ かつ $L \subseteq \Sigma^*$ でさらに $h\colon \Sigma^* \to \Delta^*$ がモノイド準同型ならば $h(L) \in \mathcal{L}$ である．

証明.　$i \in \{2, 3\}$ をとり，言語 $L \subseteq \Sigma^*$ を生成する i 型文法 $G = (V, \Sigma, P, S)$ をとる (i 型文法については定義 2.3 を見よ)．モノイド準同型写像 $\tilde{h}\colon (V \cup \Sigma)^* \to (V \cup \Delta)^*$ を

$$\begin{aligned} \tilde{h}(A) &:= A && (A \in V \text{ のとき}), \\ \tilde{h}(a) &:= h(a) && (a \in \Sigma \text{ のとき}) \end{aligned}$$

と定義する．このとき，文法 $G' = (V, \Delta, P', S)$ を $P' := \{\, A \to \tilde{h}(\alpha) \mid A \to \alpha \in P \,\}$ とおくと G' も i 型文法であり，$h(L) = h(\mathrm{L}(G)) = \mathrm{L}(G')$ であることが容易にわかる．□

次に，逆像をとる操作で閉じていることを示す．まず，簡単な正則言語の場合から証明する．

定理 2.50.　正則言語のクラス REG はモノイド準同型の逆像をとる操作で閉じている (closed under inverse homomorphism)．すなわち，$L \in \mathsf{REG}$ かつ $L \subseteq \Delta^*$ でさらに $h\colon \Sigma^* \to \Delta^*$ がモノイド準同型ならば $h^{-1}(L) \in \mathsf{REG}$ である．

証明.　$L = \mathrm{L}(\mathcal{A})$ となる DFA $\mathcal{A} = (Q, \Delta, \delta, q_0, F)$ をとる．遷移関数 $\delta\colon Q \times \Delta \to Q$ を以下のように拡張して $\hat{\delta}\colon Q \times \Delta^* \to Q$ を定める[*12]．

$$\begin{aligned} \hat{\delta}(q, \varepsilon) &:= q && (q \in Q), \\ \hat{\delta}(q, bv) &:= \hat{\delta}(\delta(q, b), v) && (q \in Q, b \in \Delta, v \in \Delta^*). \end{aligned}$$

このとき，DFA $\mathcal{B} = (Q, \Sigma, \delta', q_0, F)$ を $\delta'(q, a) := \hat{\delta}(q, h(a))$ と定めると $h^{-1}(L) = h^{-1}(\mathrm{L}(\mathcal{A})) = \mathrm{L}(\mathcal{B}) \in \mathsf{REG}$ を得る．□

文脈自由言語の場合には状態だけでなくスタックの内容も考慮する必要があるため，証明はもう少し複雑である．

[*11] $h^{-1}(L_2)$ の元が $\mathtt{b}$ を含みえないことは，例えば L_2 の元は括弧列を左側から 1 文字ずつ読んでいったときに常に (の個数) $\geq$) の個数) をみたすことからわかる．

[*12] この $\hat{\delta}(q, v)$ は Myhill–Nerode の定理 2.20 の (1) の証明で定義した $f_v(q)$ と同じものである．

定理 2.51.　文脈自由言語のクラス CFL はモノイド準同型の逆像をとる操作で閉じている．すなわち，$L \in \mathsf{CFL}$ かつ $L \subseteq \Delta^*$ でさらに $h\colon \Sigma^* \to \Delta^*$ がモノイド準同型ならば $h^{-1}(L) \in \mathsf{CFL}$ である．

証明. $L = \mathrm{L}(\mathcal{A})$ となる PDA $\mathcal{A} = (Q, \Delta, Z, \delta, q_0, F)$ をとる．補題 2.38 より $\delta \subseteq (Q \times Q) \cup (ZQ \times Q) \cup (Q \times ZQ) \cup (Q\Delta \times Q)$ が成り立つと仮定してよい．自然数 $c := \max\{\, h(a) \mid a \in \Sigma \,\}$ をとり，$\Delta^{\leq c} := \bigcup_{k \leq c} \Delta^k$ とおく．ここで PDA $\mathcal{B} = (Q', \Sigma, Z, \delta', q_0', F')$ を

$$
\begin{aligned}
Q' &:= Q \times \Delta^{\leq c}, \\
\delta' &:= \{\, ((q, \varepsilon)a, (q, h(a))) \mid q \in Q, a \in \Sigma \,\} \\
&\qquad \cup \{\, (\sigma(q, vv'), \sigma'(q', v')) \mid \sigma \in Z \cup \{\varepsilon\}, q, q' \in Q, v \in \Delta \cup \{\varepsilon\}, vv' \in \Delta^{\leq c}, (\sigma q v, \sigma' q') \in \delta \,\}, \\
q_0' &:= (q_0, \varepsilon), \\
F' &:= Q \times \{\varepsilon\}
\end{aligned}
$$

とおく．$\mathcal{B}$ は入力文字列 $w \in \Sigma^*$ に対し，$\mathcal{A}$ の入力 $h(w) \in \Delta^*$ に対する動作を模倣する．ただし，一般に文字 $a \in \Sigma$ に対し $h(a) \in \Delta^*$ は 2 文字以上になりうるので，一旦 $h(a)$ を状態 $(q, h(a)) \in Q'$ として記憶しておき，$h(a)$ から 1 文字ずつ読み込むことにより $\mathcal{A}$ の動作を模倣する．つまり，$\Delta^{\leq c}$ は $h(a)$ を一時保存しておくための**バッファー** (buffer) である．

($h^{-1}(\mathrm{L}(\mathcal{A})) \subseteq \mathrm{L}(\mathcal{B})$ であること) 入力文字列を $w = a_1 \cdots a_n a_{n+1} \in \Sigma^*$ $(n \geq 0, a_i \in \Sigma, a_{n+1} = \varepsilon)$，それらの像を $h(a_i) = b_{i,1} \cdots b_{i,l(i)}$ $(l(i) \geq 0, b_{i,j} \in \Delta)$ とする．このとき $\sigma \in Z^*, q \in Q, 1 \leq i \leq n+1, 1 \leq j \leq l(i)$ であれば

$$
\left[q_0 h(w) \overset{*}{\underset{\delta}{\Longrightarrow}} \sigma q b_{i,j} \cdots b_{i,l(i)} h(a_{i+1}) \cdots h(a_{n+1}) \right] \implies \left[(q_0, \varepsilon) w \overset{*}{\underset{\delta'}{\Longrightarrow}} \sigma (q, b_{i,j} \cdots b_{i,l(i)}) a_{i+1} \cdots a_{n+1} \right]
$$

となることを，導出 $q_0 h(w) \overset{*}{\underset{\delta}{\Longrightarrow}} \sigma q b_{i,j} \cdots b_{i,l(i)} h(a_{i+1}) \cdots h(a_{n+1})$ の長さに関する帰納法で示す．長さ 0 のときは $(q_0, \varepsilon) w \underset{\delta'}{\Longrightarrow} (q_0, b_{1,1} \cdots b_{1,l(1)}) a_2 \cdots a_{n+1}$ よりよい．長さ 1 以上のとき，最後に用いられた書き換え規則が入力文字列を消費したかどうかで場合分けする．

($\delta \cap ((Q \times Q) \cup (Q \times ZQ) \cup (ZQ \times Q))$ の元とき) このとき導出は

$$
q_0 h(w) \overset{*}{\underset{\delta}{\Longrightarrow}} \sigma q b_{i,j} \cdots b_{i,l(i)} h(a_{i+1}) \cdots h(a_{n+1}) \underset{\delta}{\Longrightarrow} \sigma' q' b_{i,j} \cdots b_{i,l(i)} h(a_{i+1}) \cdots h(a_{n+1})
$$

という形をしているので，帰納法の仮定と合わせて

$$
q_0 w \overset{*}{\underset{\delta'}{\Longrightarrow}} \sigma (q, b_{i,j} \cdots b_{i,l(i)}) a_{i+1} \cdots a_{n+1} \underset{\delta'}{\Longrightarrow} \sigma' (q', b_{i,j} \cdots b_{i,l(i)}) a_{i+1} \cdots a_{n+1}
$$

を得る．

($\delta \cap (Q\Delta \times Q)$ の元とき) このとき，導出が $j < l(i)$ について

$$
q_0 h(w) \overset{*}{\underset{\delta}{\Longrightarrow}} \sigma q b_{i,j} b_{i,j+1} \cdots b_{i,l(i)} h(a_{i+1}) \cdots h(a_{n+1}) \underset{\delta}{\Longrightarrow} \sigma q' b_{i,j+1} \cdots b_{i,l(i)} h(a_{i+1}) \cdots h(a_{n+1})
$$

という形をしているならば，帰納法の仮定と合わせて

$$
q_0 w \overset{*}{\underset{\delta'}{\Longrightarrow}} \sigma (q, b_{i,j} b_{i,j+1} \cdots b_{i,l(i)}) a_{i+1} \cdots a_{n+1} \underset{\delta'}{\Longrightarrow} \sigma (q', b_{i,j+1} \cdots b_{i,l(i)}) a_{i+1} \cdots a_{n+1}
$$

となり，$j = l(i)$ のときは

$$
\begin{aligned}
q_0 h(w) &\overset{*}{\underset{\delta}{\Longrightarrow}} \sigma q b_{i,l(i)} b_{i+1,1} \cdots b_{i+1,l(i+1)} h(a_{i+2}) \cdots h(a_{n+1}) \\
&\underset{\delta}{\Longrightarrow} \sigma q' b_{i+1,1} \cdots b_{i+1,l(i+1)} h(a_{i+2}) \cdots h(a_{n+1})
\end{aligned}
$$

という形だから，帰納法の仮定と合わせて

$$q_0 w \overset{*}{\underset{\delta'}{\Longrightarrow}} \sigma(q, b_{i,l(i)})a_{i+1}a_{i+2}\cdots a_{n+1} \underset{\delta'}{\Longrightarrow} \sigma(q', \varepsilon)a_{i+1}a_{i+2}\cdots a_{n+1}$$
$$\underset{\delta'}{\Longrightarrow} \sigma(q', b_{i+1,1}\cdots b_{i+1,l(i+1)})a_{i+2}\cdots a_{n+1}$$

を得る．

$(\mathrm{L}(\mathcal{B}) \subseteq h^{-1}(\mathrm{L}(\mathcal{A}))$ であること$)$ δ' の定め方から，どんな $\sigma \in Z^*, q \in Q, v \in \Delta^{\leq c}, u \in \Sigma^*$ に対しても

$$\left[(q_0, \varepsilon)w \overset{*}{\underset{\delta'}{\Longrightarrow}} \sigma(q, v)u\right] \implies \left[q_0 h(w) \overset{*}{\underset{\delta}{\Longrightarrow}} \sigma q v h(u)\right]$$

が成り立つことが (導出の長さに関する帰納法で) 容易にわかるのでよい．

以上より $h^{-1}(L) = h^{-1}(\mathrm{L}(\mathcal{A})) = \mathrm{L}(\mathcal{B}) \in \mathsf{CFL}$ を得る． □

第 II 部

群から定まる言語とグラフ

第 3 章

群論再入門

第 4 章

群の語の問題

第 5 章

群の Cayley グラフ

第 III 部

遠くから見たグラフと木

第 IV 部

Bass–Serre 理論入門

第Ⅴ部

付録

付録 A

形式言語理論続論

A.1　決定性プッシュダウンオートマトンと決定性文脈自由言語

A.2　単カウンターオートマトンと単カウンター言語

A.3　続・言語クラスの閉包性

A.4　言語クラスの分離

付録 B

実質的巡回群と単カウンター言語

付録 C

Muller–Schupp の原論文における証明法

付録 D

Stallings の前群と測地的書き換え系

参考文献

[風上松町 04] 風間喜代三, 上野善道, 松村一登, and 町田健, 言語学 [第 2 版], 東京大学出版会, 2004.

[新屋良 17] 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** (2017), no. 3, 3–35.

[BN98] Franz Baader and Tobias Nipkow, *Term rewriting and all that*, Cambridge University Press, Cambridge, 1998. MR1629216

[DW17] Volker Diekert and Armin Weiß, *Context-free groups and Bass-Serre theory*, Algorithmic and geometric topics around free groups and automorphisms, 2017, pp. 43–110. MR3729161

[HMU03] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman, オートマトン 言語理論 計算論 I [第 2 版], サイエンス社, 2003. 野崎昭弘・高橋正子・町田元・山崎秀記 共訳.

[Koz97] Dexter C. Kozen, *Automata and computability*, Undergraduate Texts in Computer Science, Springer-Verlag, New York, 1997. MR1633052

[Law04] Mark V. Lawson, *Finite automata*, Chapman & Hall/CRC, Boca Raton, FL, 2004. MR2002708

[Sak09] Jacques Sakarovitch, *Elements of automata theory*, Cambridge University Press, Cambridge, 2009. Translated from the 2003 French original by Reuben Thomas. MR2567276

[Sip08] Michael Sipser, 計算理論の基礎 [原著第 2 版] 1. オートマトンと言語, 共立出版, 2008. 太田和夫・田中圭介 監訳, 阿部正幸・植田広樹・藤岡淳・渡辺治 訳.

変更履歴

2021/05/01　第 I 部のみ公開