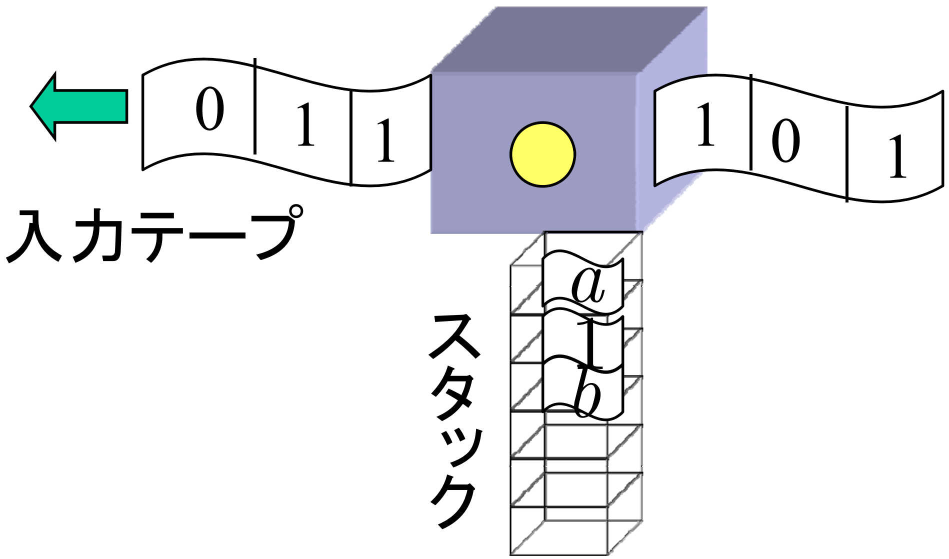
３. プッシュダウンオートマトンと
文脈自由文法

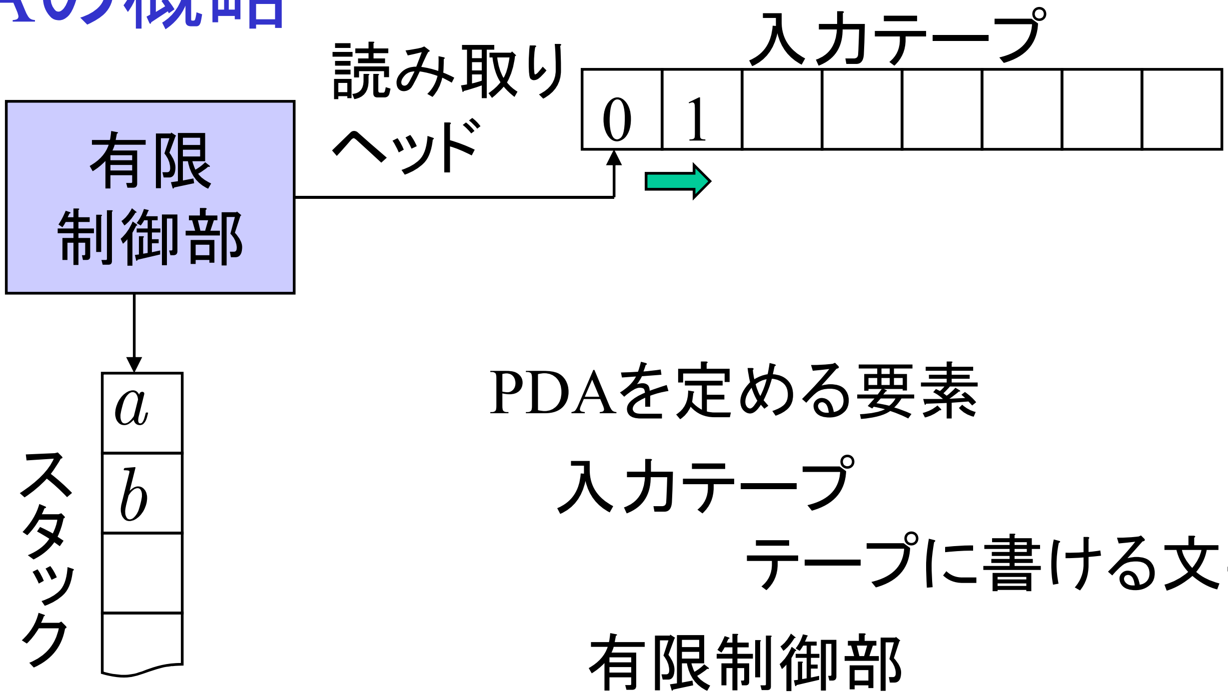
3－1. プッシュダウンオートマトン

オートマトンはメモリがほとんど無かった。
この制限を除いた機械を考える。
理想的なスタックを利用できるようなオートマトンを
プッシュダウンオートマトン(Push Down Automaton,PDA)
という。



入力テープを一度走査したあと、
「はい」ならランプ点灯
「いいえ」ならランプ消灯。

PDAの概略



PDAを定める要素

- 入力テープ
  - テープに書ける文字
- 有限制御部
  - 内部状態
  - 初期状態
  - 状態変化
  - 受理かどうかの判断
- スタック（無限長）
  - スタックに書ける文字

PDAの数学的定義

定義：（プッシュダウンオートマトン）

PDAは、$P=(Q,\Sigma,\Gamma,\delta,q_0,F)$ の6項組で与えられる。
ここで、

1. Q は有限集合で、状態を表す。
2. Σ は有限集合で、入力アルファベットを表す。
3. Γ は有限集合で、スタックアルファベットを表す。
3. δは $Q\times\Sigma_\varepsilon\times\Gamma_\varepsilon$ から $\mathcal{P}(Q\times\Gamma_\varepsilon)$ への写像
（ $\delta:Q\times\Sigma_\varepsilon\times\Gamma_\varepsilon\rightarrow\mathcal{P}\left(Q\times\Gamma_\varepsilon\right)$ ）で、
状態遷移を表す。δ を状態遷移関数という。
4. $q_0\in Q$ は、初期状態を表す。
5. $F\subseteq Q$ は受理状態の集合を表す。

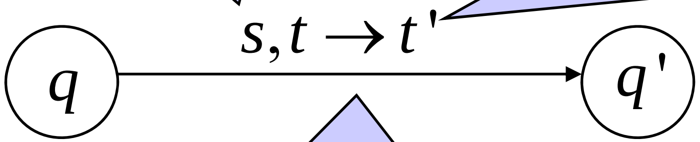
ここで、$\Sigma_\varepsilon=\Sigma\cup\{\varepsilon\}$ $\Gamma_\varepsilon=\Gamma\cup\{\varepsilon\}$ である。

PDAの図式表現(状態遷移図)

PDAは、状態遷移図で表現できる。

$(q', t') \in \delta(q, s, t)$ のとき、

入力記号



状態の変化

スタックの変化

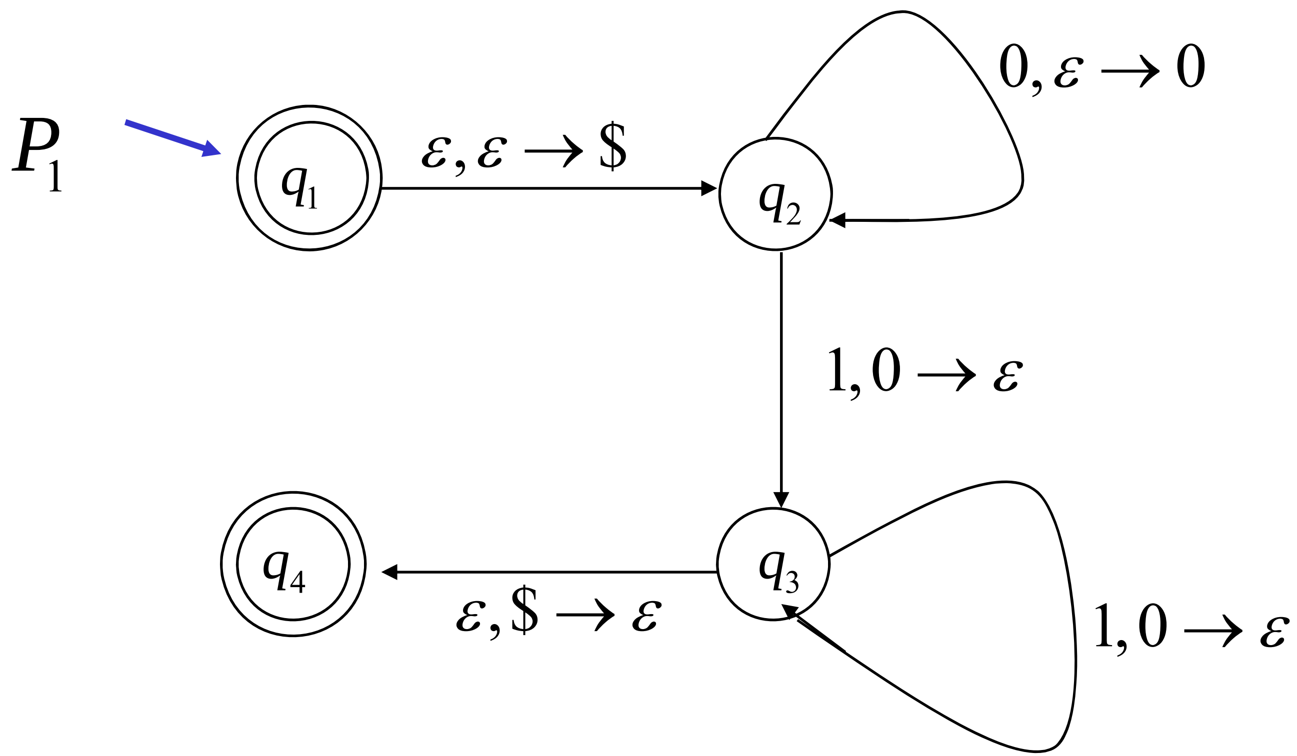
スタック先頭の記号を
t から t' へ変化させる。

$push(t') : \varepsilon \longrightarrow t'$

$t = pop() : t \longrightarrow \varepsilon$

PDAの例

PDA例　$B = \{0^n 1^n \mid n \geq 0\}$ を認識するPDA P_1



形式的定義

$$P_1 = (Q, \Sigma, \Gamma, \delta, q_1, F)$$

ただし、

$Q = \{q_1, q_2, q_3, q_4\}$ （状態集合）

$\Sigma = \{0, 1\}$ （入力アルファベット）

$\Gamma = \{0, \$\}$ （スタックアルファベット）

スタックの“底”を表す
特別な記号。

q_1 （初期状態）

$F = \{q_1, q_4\}$ （受理状態）

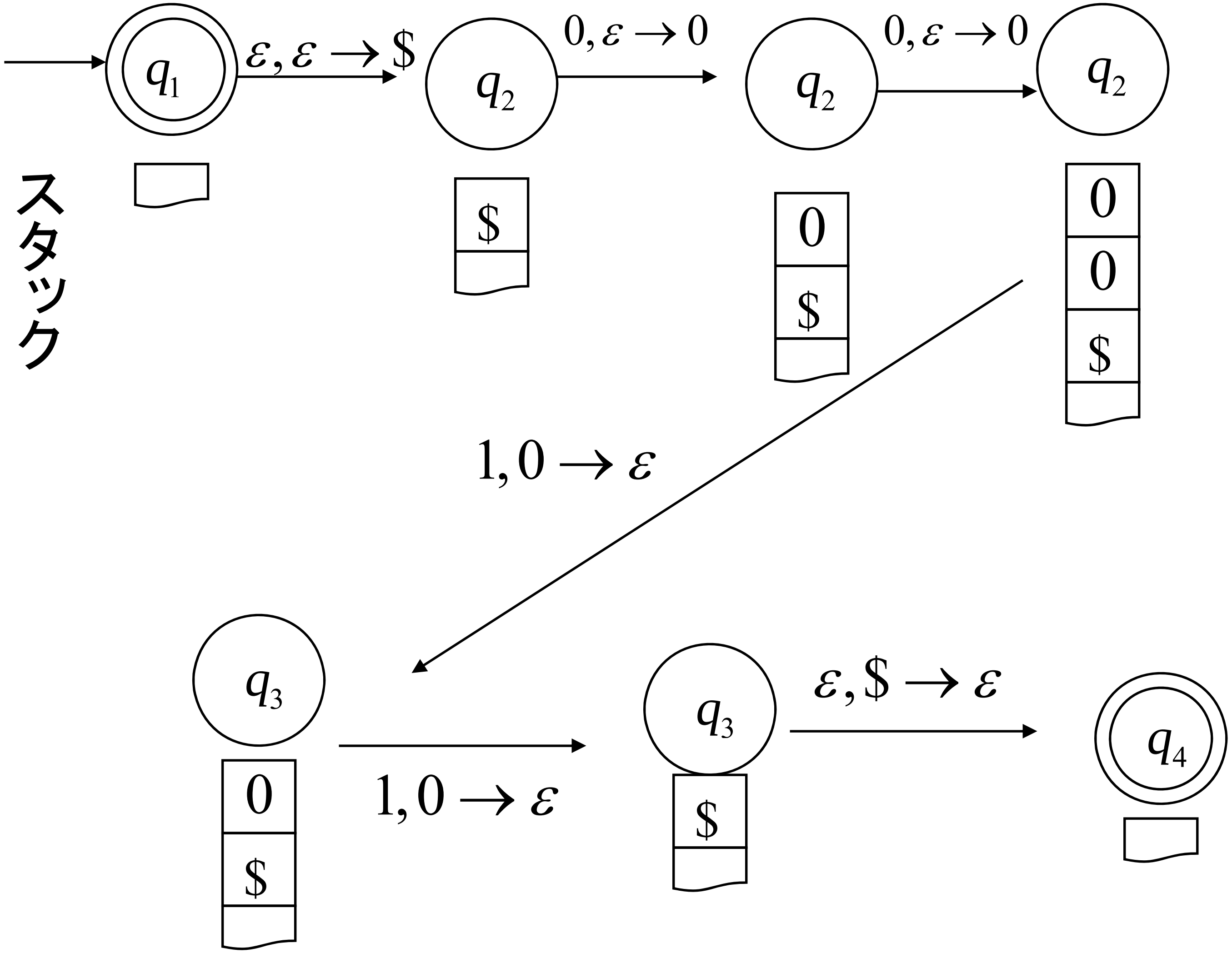
δ 状態遷移関数

入力	0			1			ε		
スタック	0	\$	ε	0	\$	ε	0	\$	ε
q_1									$\{(q_2,\$)\}$
q_2			$\{(q_2,0)\}$	$\{(q_3,\varepsilon)\}$					
q_3				$\{(q_3,\varepsilon)\}$				$\{(q_4,\varepsilon)\}$	
q_4									

この表において、空白は空集合 ϕ を表している。

PDAの状態遷移

$w = 0011$ による状態遷移

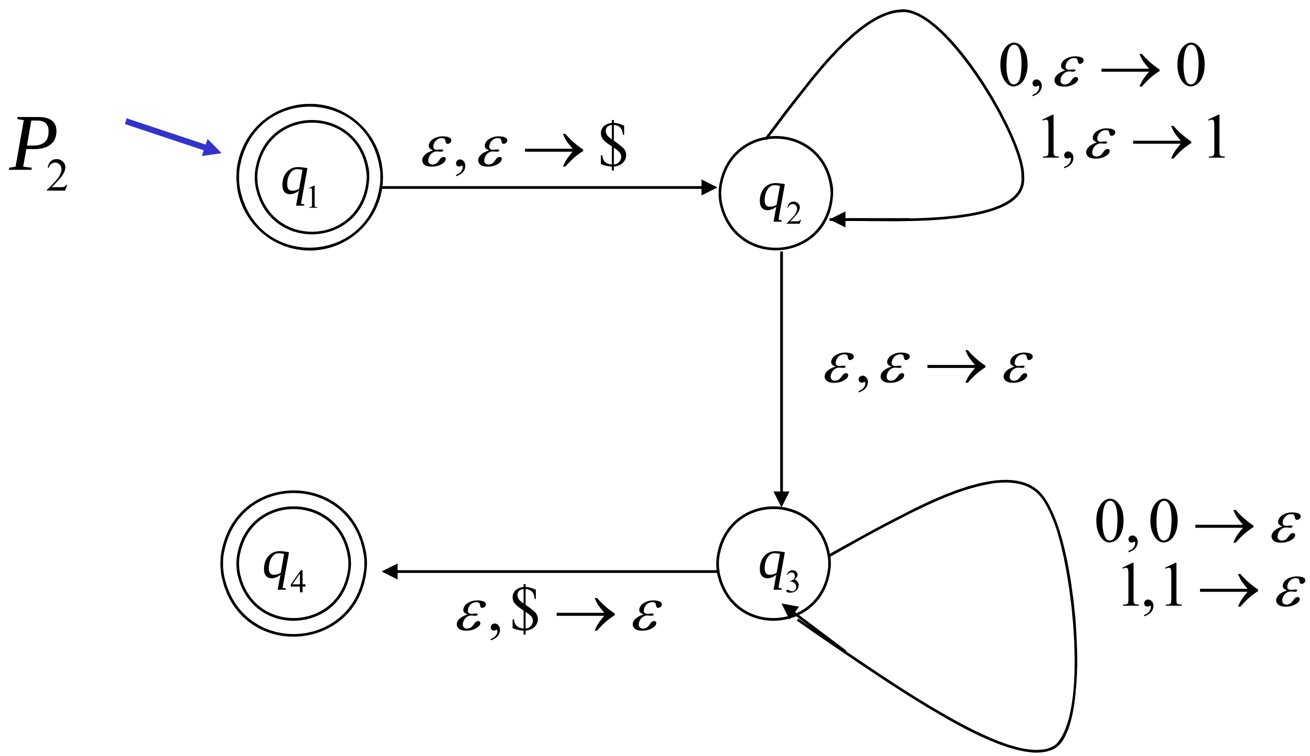


例2

次の言語を認識するPDAを与える。

$$\{ww^{\mathcal{R}} \mid w \in \{0,1\}^*\}$$

ここで、$w^{\mathcal{R}}$は w を逆に書いた文字列。



練習

P_2 に対する形式的な定義を求めよ。
また、$s = 10111101$
に対する P_2 の遷移をスタックの内容と共に示せ。

３－2.文脈自由文法

以前、
DFAが認識できる言語のクラス（正規言語）に対して、
異なる表現法(正規表現）を与えた。
ここでは、
PDAが認識できる言語のクラス（文脈自由言語）に対して、
もう一つの表現法（文脈自由文法）を与える。

文脈自由文法とは

文法例　G_1

$$
\begin{aligned}
A &\to 0A1 \\
A &\to B \\
B &\to \varepsilon
\end{aligned}
$$

導出　$A \to 0A1 \to 00A11 \to 00B11 \to 00\varepsilon 11 \to 0011$

定義：（文脈自由文法関連）

文脈自由文法は生成規則あるいは書き換え規則と呼ばれる式の集合で定められる。生成規則の左辺は、一つの変数（非終端記号）であり、右辺は変数とアルファベット(終端記号）の列である。文脈自由文法では、開始記号から生成規則を基に書き換えられる。すべて記号が終端記号になった時点で終了する。（上の例　G_1 では、開始記号はAとしている。）文脈自由文法において、終端記号列に変換する過程（生成記号系列）を導出という。

CFGのの形式的定義

定義：（文脈自由文法）

CFGは、$C=(V,\Sigma,R,S)$ の4項組で与えられる。
ここで、

1. V は変数（非終端記号）と呼ばれる有限集合。
2. Σ はアルファベット（終端記号）と呼ばれ有限集合。
   Vとは共通部分を持たない。つまり、$V\cap\Sigma=\phi$ 。
3. Rは、生成規則の有限集合である。ただし、
   生成規則の左辺は一つの非終端記号であり、
   右辺は変数と終端記号の文字列からなる。
   すなわち、各生成規則は $A\in V,\alpha\in(V+\Sigma)^*$ として、
   $$A\to\alpha$$
   と表される。
4. $S\in V$ は開始記号。

導出可能性を表す表現

ある系列 $\alpha \in (V + \sum)^*$ に任意回($k(\geq 1)$ 回)の規則の適用で系列 $\beta \in (V + \sum)^*$ が得れることを $\alpha \xrightarrow{*} \beta$ とも書く。
すなわち、 $\alpha \xrightarrow{*} \beta$ は、

$$\alpha = \alpha_1 \rightarrow \alpha_2 \rightarrow \cdots \rightarrow \alpha_k = \beta$$

のことである。

文脈自由言語(CFL)

定義: (文脈自由言語)

文脈自由文法(Context-Free Grammar,CFG)で
記述できる言語を
文脈自由言語(Context-Free Language,CFL)と呼ぶ。
ある文脈自由文法 G に対して、 G から導出できる言語
を

$$L(G)$$

と書く。

導出列

G_1 が 000111 を導出できることを示す。

$$A \longrightarrow 0A1 \longrightarrow 00A11 \longrightarrow 000A111 \longrightarrow$$

$$000B111 \longrightarrow 000\varepsilon 111 \longrightarrow 000111$$

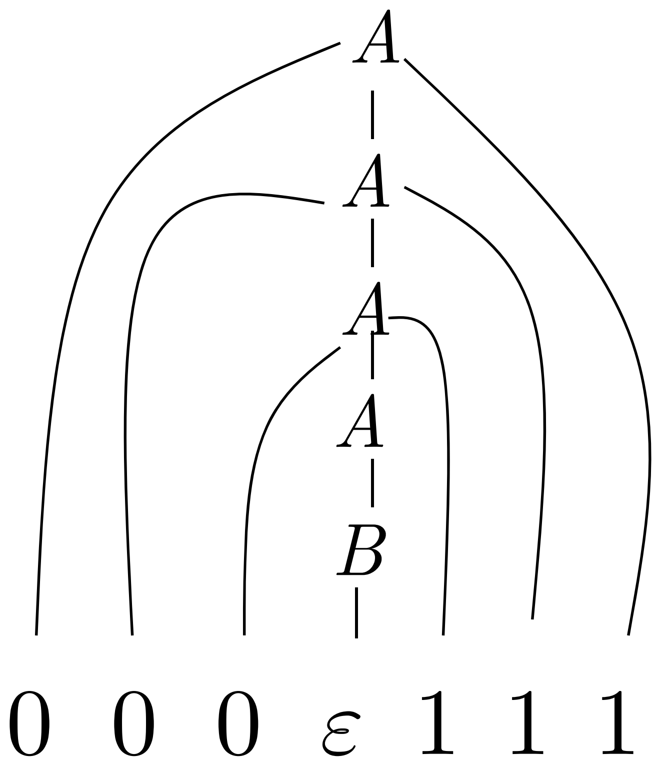
定義：（導出列）

このような、生成規則の適用される順序を示したものを**導出列**とよぶ。

構文解析木

定義：（構文解析木）

文字列に対して、導出における生成規則の適用を
図式的に表現できる。このような導出過程を表す木状の図形を
構文解析木と呼ぶ。



CFGの例２

G_2

$$< Sentence > \rightarrow < Noun - Phrase > < Verb - Phrase >$$

$$< Noun - Phrase > \rightarrow < Cmplx - Noun > | < Cmplx - Noun > < Prep - Phrase >$$

$$< Verb - Phrase > \rightarrow < Cmplx - Verb > | < Cmplx - Verb > < Prep - Phrase >$$

$$< Prep - Phrase > \rightarrow < Prep > < Cmplx - Noun >$$

$$< Cmplx - Noun > \rightarrow < Article > < Noun >$$

$$< Cmplx - Verb > \rightarrow < Verb > | < Verb > < Noun - Phrase >$$

$$< Article > \rightarrow \mathrm{a} \mid \mathrm{the}$$

$$< Noun > \rightarrow \mathrm{boy} | \mathrm{girl} | \mathrm{flower}$$

$$< Verb > \rightarrow \mathrm{touches} | \mathrm{likes} | \mathrm{sees}$$

$$< Prep > \rightarrow \mathrm{with}$$

開始記号　$< Sentence >$

導出列2

G_2 から “a boy sees” が導出できることを示す。

<Sentence> →<Norn-Phrase><Verb-Phrase>
→<Cmplx-Noun><Verb-Phrase>
→ <Article><Noun><Verb-Phrase>
→a <Noun><Verb-Phrase>
→a boy <Verb-Phrase>
→a boy <Cmplx-Verb>
→a boy <Verb>
→a boy sees

練習

G_2によって、次の文字列が導出できることを、
導出列および構文解析木によって示せ。

(1) the girl touches the boy
(2) a girl with a flower likes the boy

CFGの形式的定義例

G_1

$$G_1 = (\{A, B\}, \{0, 1, \varepsilon\}, \{A \to 0A1, A \to B, B \to \varepsilon\}, A)$$

G_2

$$G_2 = (V, \Sigma, R, < Sentence >)$$

ただし、

$$V = \{< Sentence >, < Noun - Phraes >, < Verb - Phrase >,$$

$$< P\mathrm{r}ep - Phrase >, < Cmplx - Noun >, < Cmplx - Verb >,$$

$$< Article >, < Noun >, < Verb >, < P\mathrm{r}ep >\}$$

$$\Sigma = \{a, b, c, \cdots, z, (\text{スペース})\}$$

R は前述の規則の集合

曖昧性

定義：（曖昧性）

CFGにおいて、異なった構文解析木を持つにもかかわらず、同じ文字列を生成することがある。
このように、2つ以上の構文解析木を持つような文字列を生成できるとき、そのCFGは**曖昧**であるといわれる。

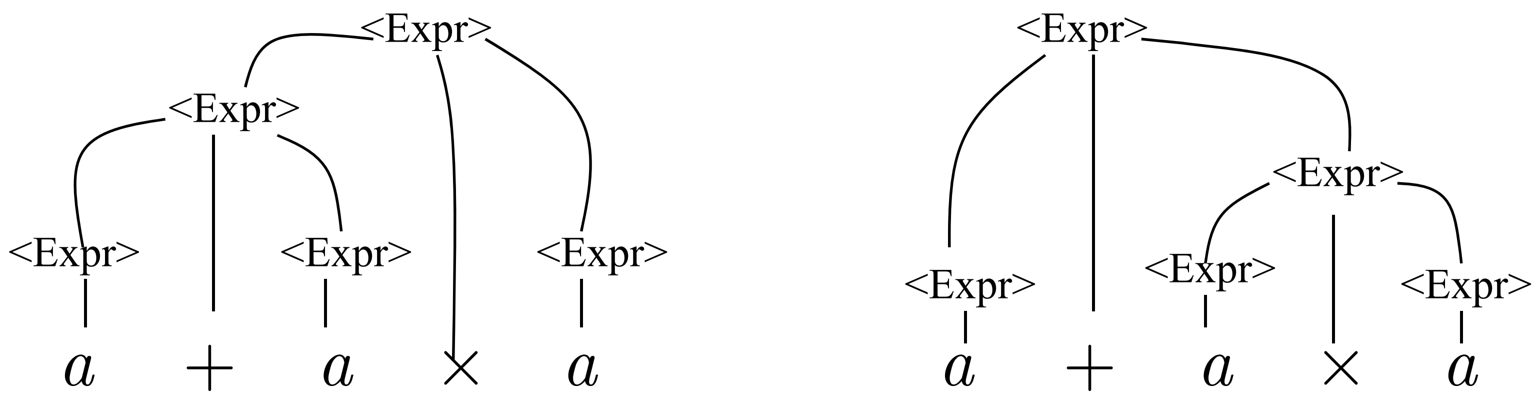
曖昧なCLG例

$$G_3 = (V, \Sigma, R, < Exp >)$$

$$V = \{< Expr >\}$$

$$\Sigma = \{a, +, \times, (,)\}$$

$$\begin{aligned} R = \{< Expr > \rightarrow\ & < Expr > + < Expr > \,| \\ & < Expr > \times < Expr > \,| \\ & (< Expr >) \,|\, a\} \end{aligned}$$



練習

G_2 によって、次の文字列が生成できる。

the girl touches the boy with the flower

この文字列の構文解析木を2つ示すことによって、
G_2が曖昧であることを示せ。

曖昧性の除去

簡単な数式を生成するCLG G_3 は曖昧であった。

ここでは、簡単な数式を生成する
曖昧でないCLG G_4 を示す。

$$G_4 = (V, \Sigma, R, < Exp >)$$

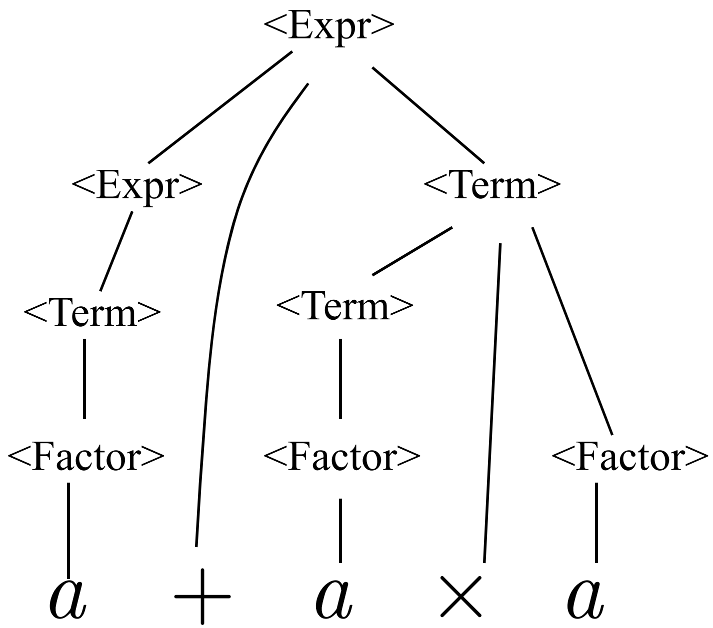
$$V = \{< Expr >, < Term >, < Factor >\}$$

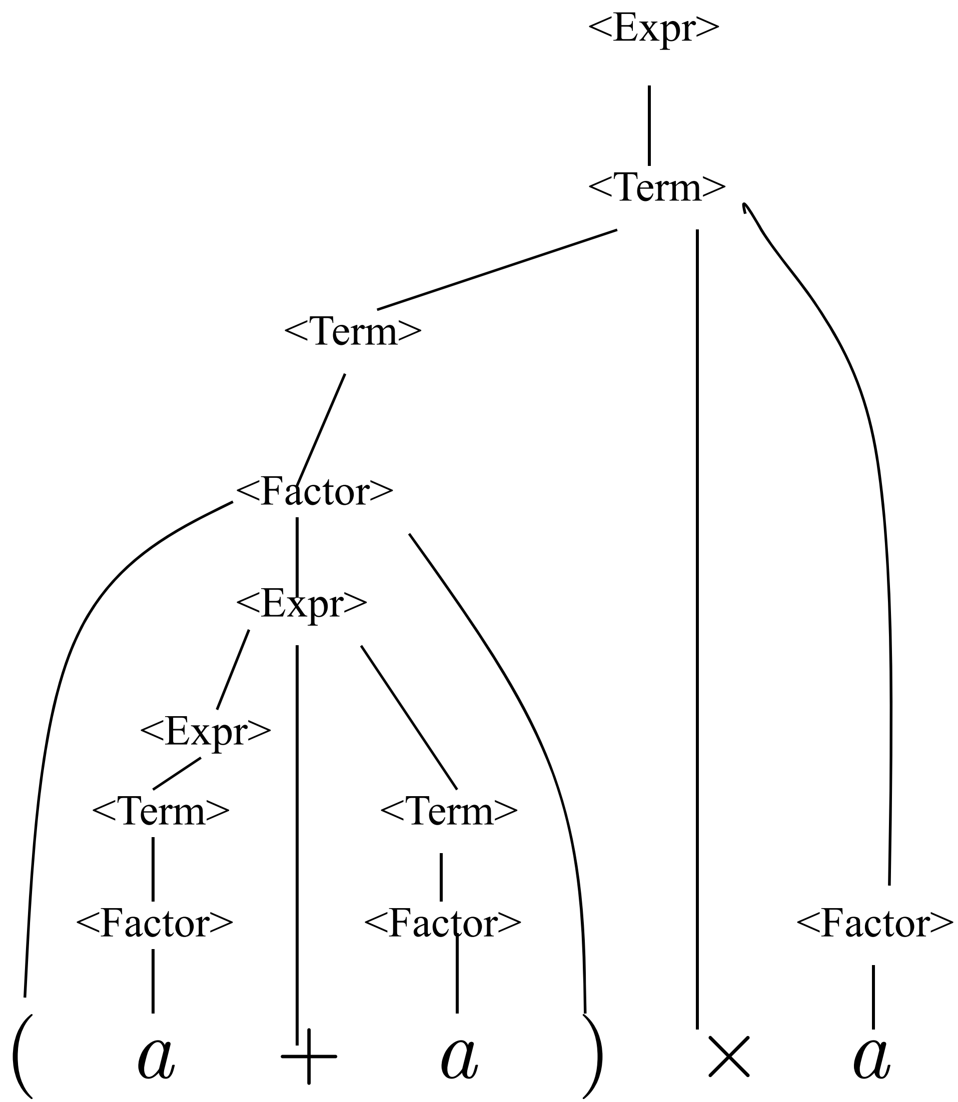
$$\Sigma = \{a, +, \times, (,)\}$$

$$R = \{< Expr > \to < Expr > + < Term > | < Term >,$$

$$< Term > \to < Term > \times < Factor > | < Factor >,$$

$$< Factor > \to (< Expr >) \mid a\}$$


<Expr>
<Expr>
<Term>
<Term>
<Term>
<Factor>
<Factor>
<Factor>
a + a × a


<Expr>
<Term>
<Term>
<Factor>
<Expr>
<Expr>
<Term>
<Term>
<Factor>
<Factor>
<Factor>
(a + a) × a

本質的に曖昧なCFL

曖昧な文法に対して、同じ言語を生成する曖昧でない
文法を構成できることがある。
（例えば、G_3 と G_4 ）
しかし、
曖昧な文法によってのみ生成可能な言語が存在する。
次の言語は、CFLであるが、曖昧な文法だけからしか
生成できない。
（このような言語は本質的に曖昧と呼ばれることがある。）

$$\{0^i 1^j 2^k \mid i = j \text{または} j = k\}$$

CFGの応用

プログラミング言語の文法定義

C言語の文法定義の一部

statement:

lableled-statement
expression-statement
compound-statement
selection-statement
iteration-statement
jump-statement

selection-statement:

if(*expression*) *statement*
if(*expression*) *statement* else *statement*
switch (*expression*) *statement*

斜体:非終端記号、立体:終端記号