コンパイラ

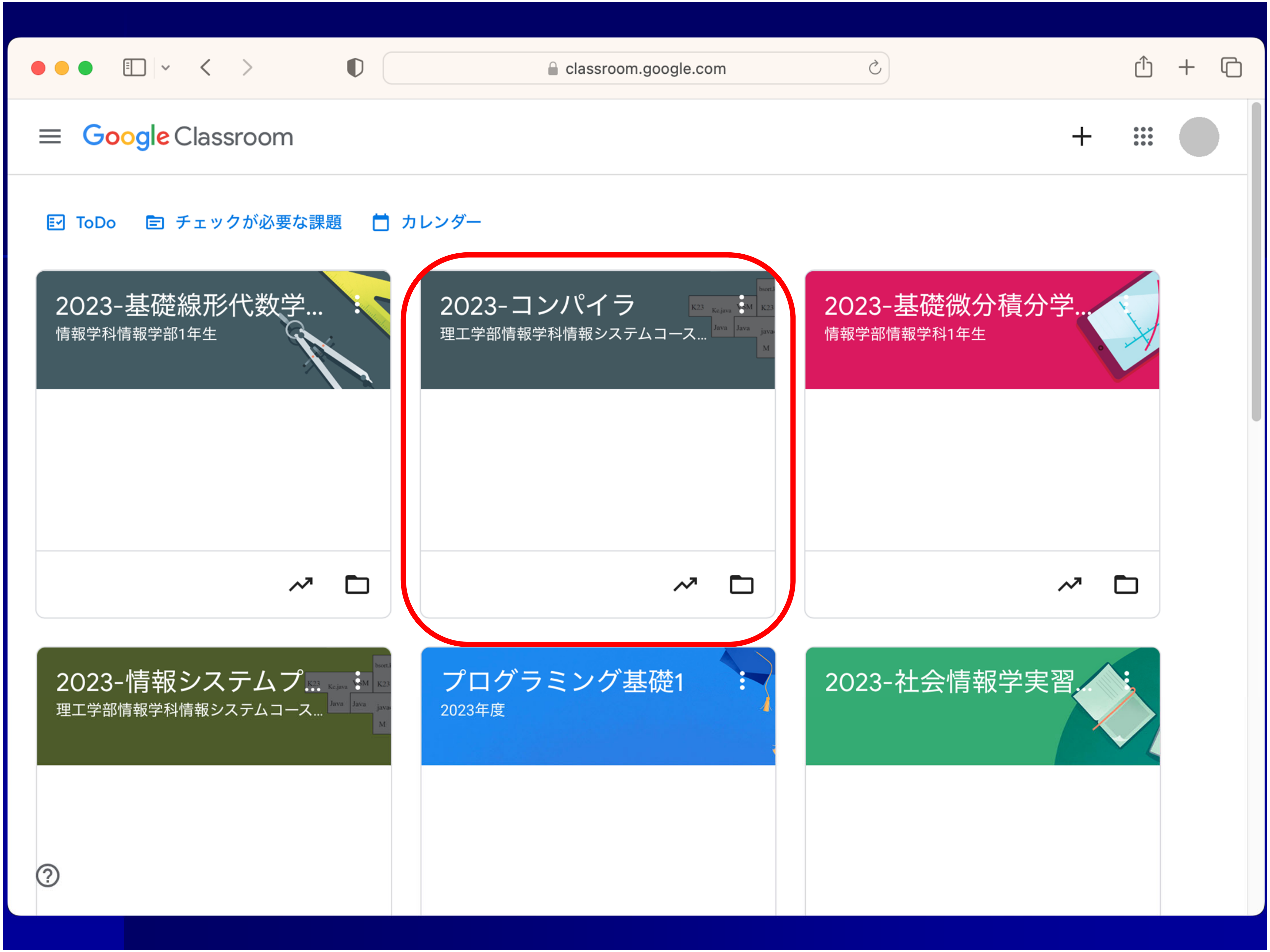
第2回 形式言語と形式文法

— 有限オートマトンの復習 —

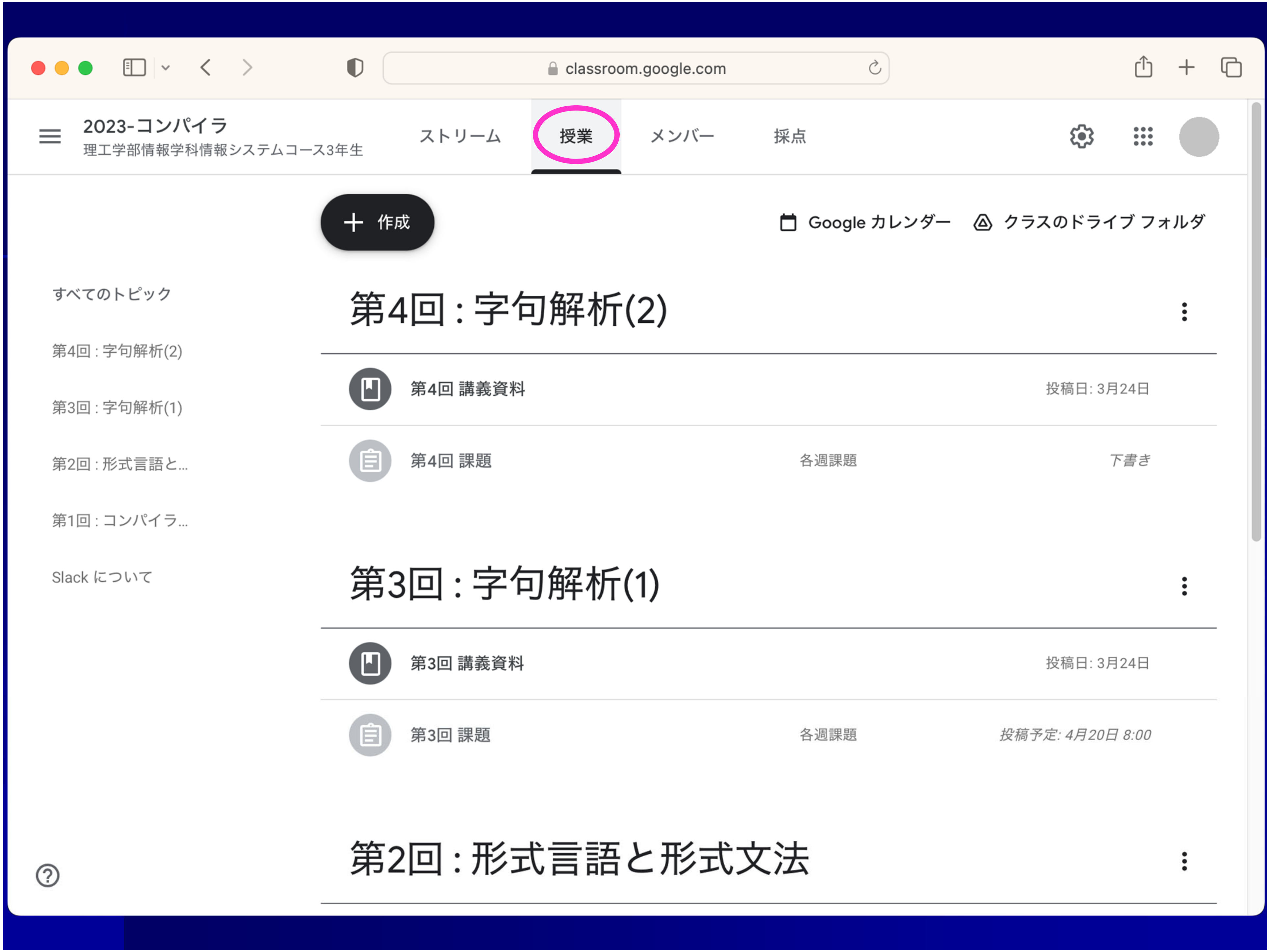
http://www.info.kindai.ac.jp/compiler

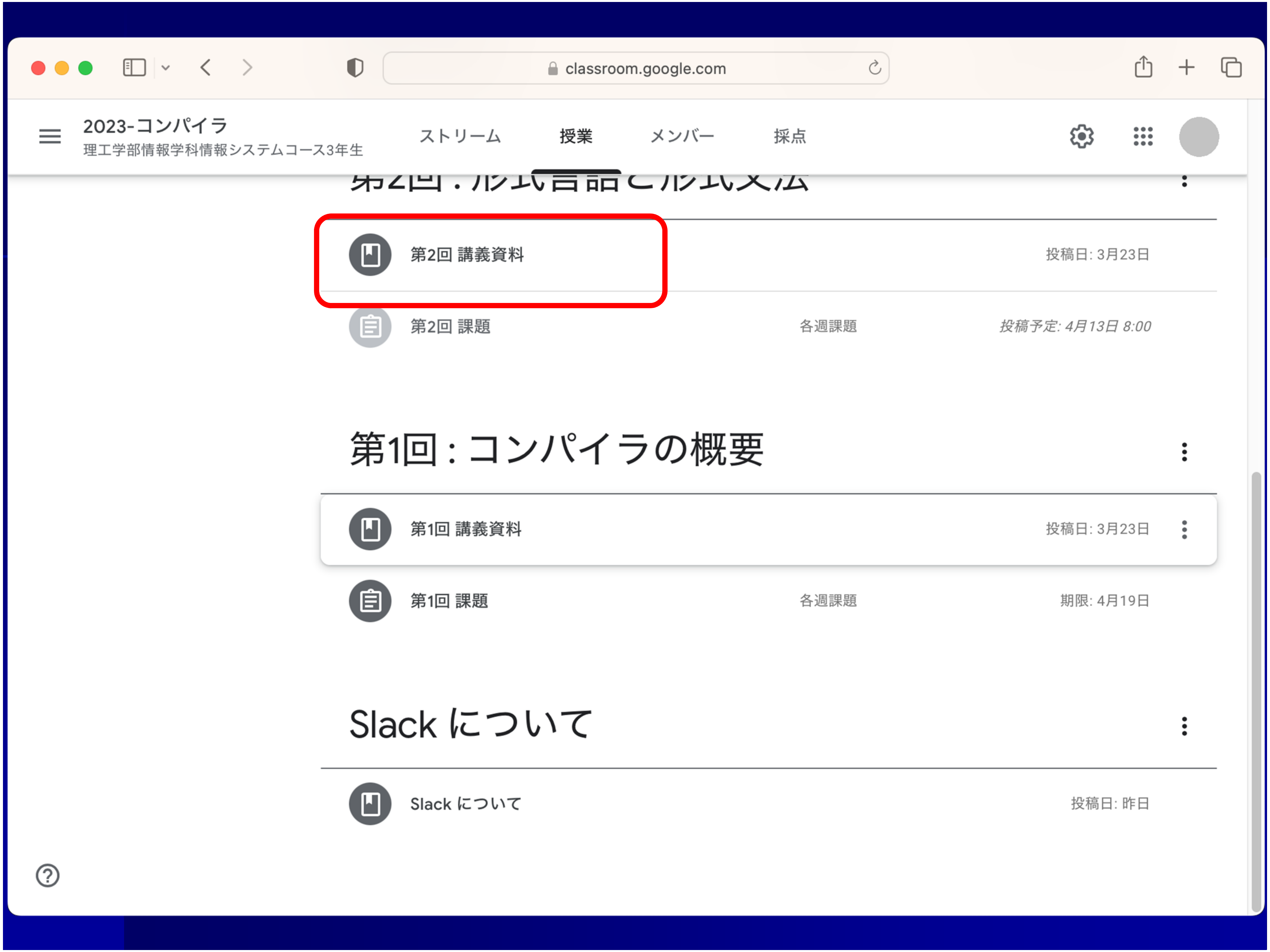
E館3階E-331 内線5459

takasi-i@info.kindai.ac.jp


classroom.google.com
Google Classroom
ToDo
チェックが必要な課題
カレンダー
2023-基礎線形代数学...
情報学科情報学部1年生
2023-コンパイラ
理工学部情報学科情報システムコース...
2023-基礎微分積分学...
情報学部情報学科1年生
2023-情報システムプ...
理工学部情報学科情報システムコース...
プログラミング基礎1
2023年度
2023-社会情報学実習...


classroom.google.com
2023-コンパイラ
理工学部情報学科情報システムコース3年生
ストリーム
授業
メンバー
採点
bsort.k
カスタマイズ
K23
Kc.java
VSM
K23
K23
Kc.class
VSM
VSM
Java
Java
javac
JBC
JBC
VSM
vsm
M
M
JBC
java
M
2023-コンパイラ
理工学部情報学科情報システムコース3年生
Meet
リンクを生成
クラスへの連絡事項を入力
石水隆 さんが新しい資料を投稿しました: Slack について
昨日
クラスコード
zokgxoo
石水隆 さんが新しい資料を投稿しました: 第4回 講義資料
3月24日
期限間近
提出期限の近い課題はありません
石水隆 さんが新しい資料を投稿しました: 第3回 講義資料


classroom.google.com
2023-コンパイラ
理工学部情報学科情報システムコース3年生
ストリーム
授業
メンバー
採点
作成
Google カレンダー
クラスのドライブ フォルダ
すべてのトピック
第4回 : 字句解析(2)
第3回 : 字句解析(1)
第2回 : 形式言語と...
第1回 : コンパイラ...
Slack について
第4回 : 字句解析(2)
第4回 講義資料
投稿日: 3月24日
第4回 課題
各週課題
下書き
第3回 : 字句解析(1)
第3回 講義資料
投稿日: 3月24日
第3回 課題
各週課題
投稿予定: 4月20日 8:00
第2回 : 形式言語と形式文法


classroom.google.com
2023-コンパイラ
理工学部情報学科情報システムコース3年生
ストリーム
授業
メンバー
採点
第2回 講義資料
投稿日: 3月23日
第2回 課題
各週課題
投稿予定: 4月13日 8:00
第1回：コンパイラの概要
第1回 講義資料
投稿日: 3月23日
第1回 課題
各週課題
期限: 4月19日
Slack について
Slack について
投稿日: 昨日

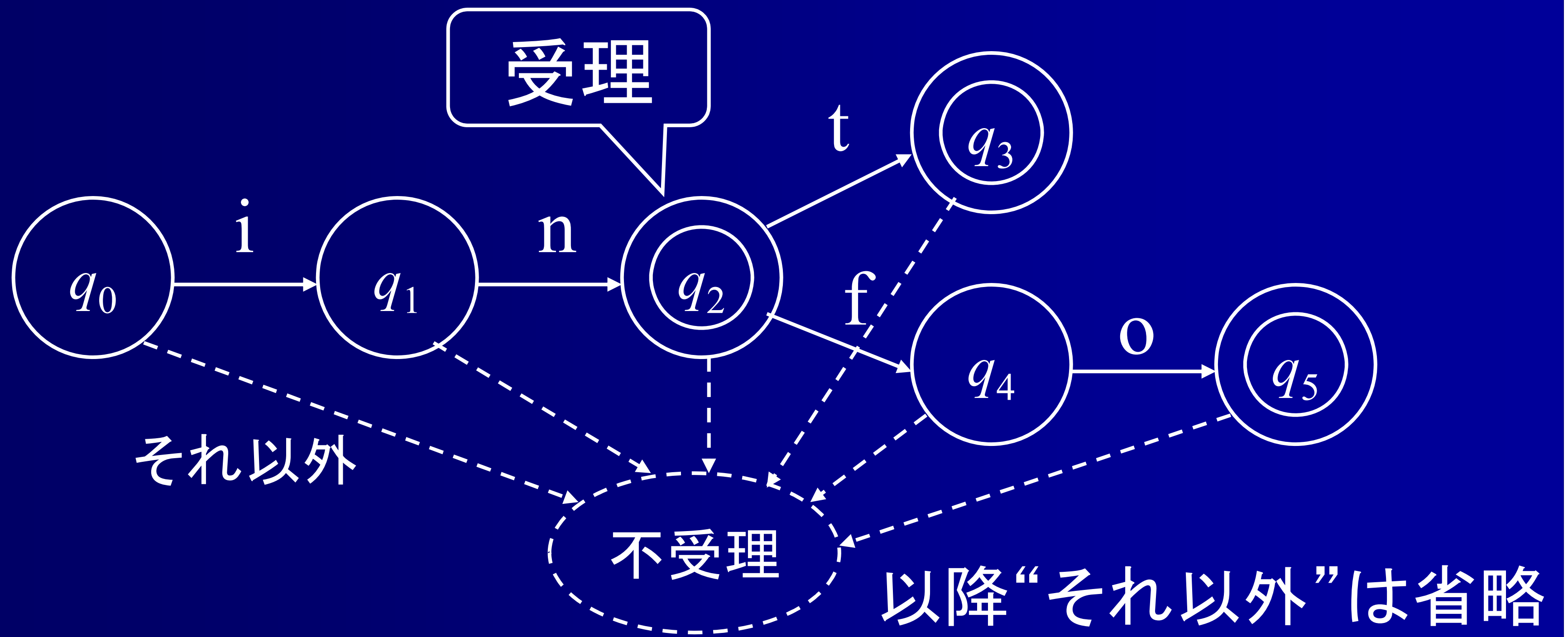
コンパイラの構造

- 字句解析系　⇒ 有限オートマトン
- 構文解析系
- 制約検査系
- 中間コード生成系
- 最適化系
- 目的コード生成系

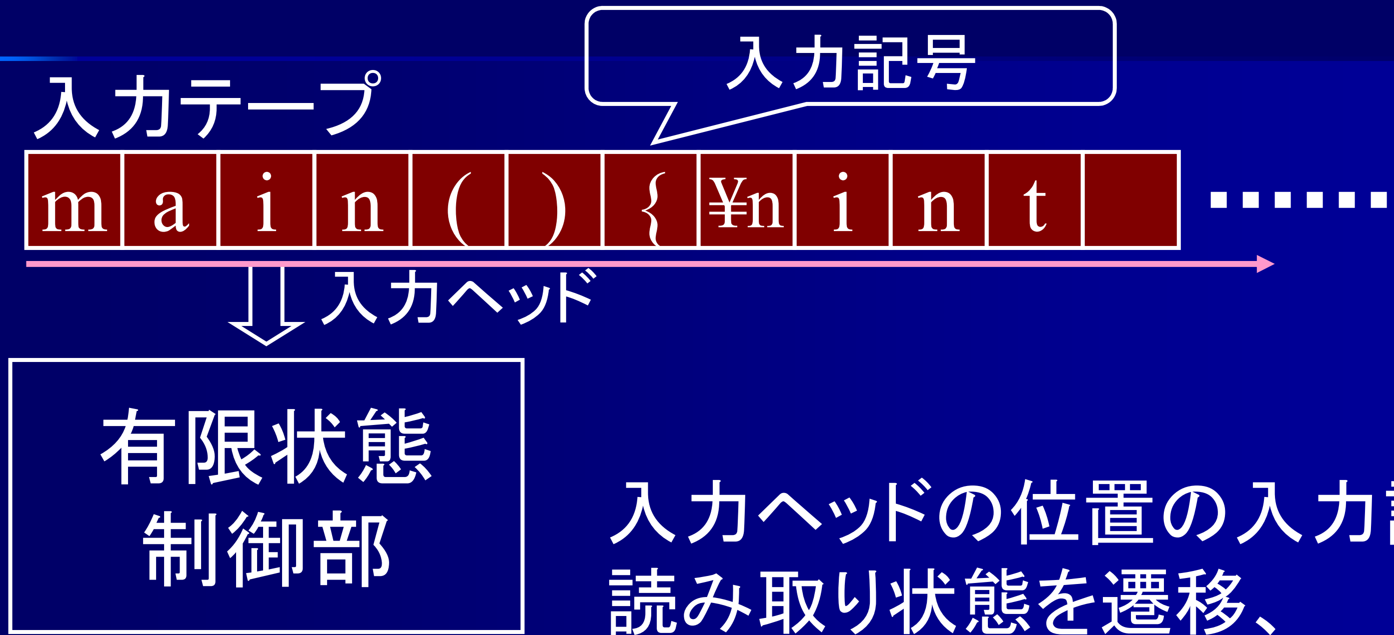
有限(状態)オートマトン
(finite (state) automaton)

- 特定の入力列を受理する状態遷移機械

例 : 文字列 “in”, “int”, “info” を受理するオートマトン



有限オートマトンの概念図



入力ヘッドの位置の入力記号を
読み取り状態を遷移、
入力ヘッドを1つ進める

有限オートマトンの5つ組

- $\mathrm{M} = (\mathbf{Q}, \mathbf{\Sigma}, \delta, q_0, \mathbf{F})$
  - $\mathbf{Q}$ ：状態の有限集合
  - $\mathbf{\Sigma}$: 入力記号の有限集合
  - δ : 状態遷移関数
    - $\delta(p, a) = q, \ (p, q \in \mathbf{Q}, a \in \mathbf{\Sigma})$
  - q_0 : 初期状態
    - $q_0 \in \mathbf{Q}$
  - $\mathbf{F}$ ：最終状態の有限集合
    - $\mathbf{F} \subseteq \mathbf{Q}$

有限オートマトンの例

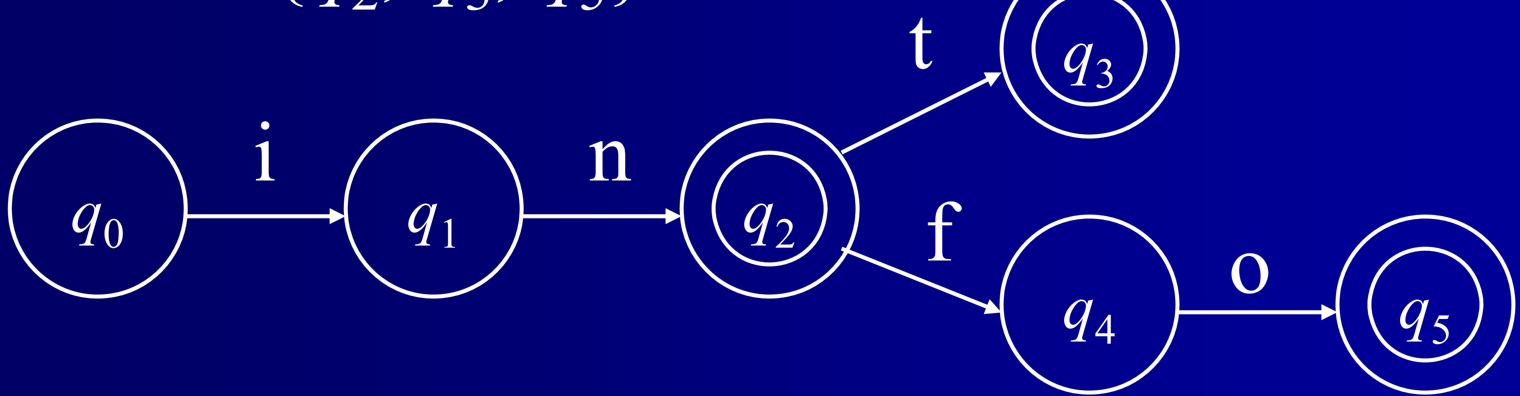
- $M = (\mathbf{Q}, \mathbf{\Sigma}, \delta, q_0, \mathbf{F})$

$\mathbf{Q} = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

$\mathbf{\Sigma} = \{a, b, c, \ldots, z\}$

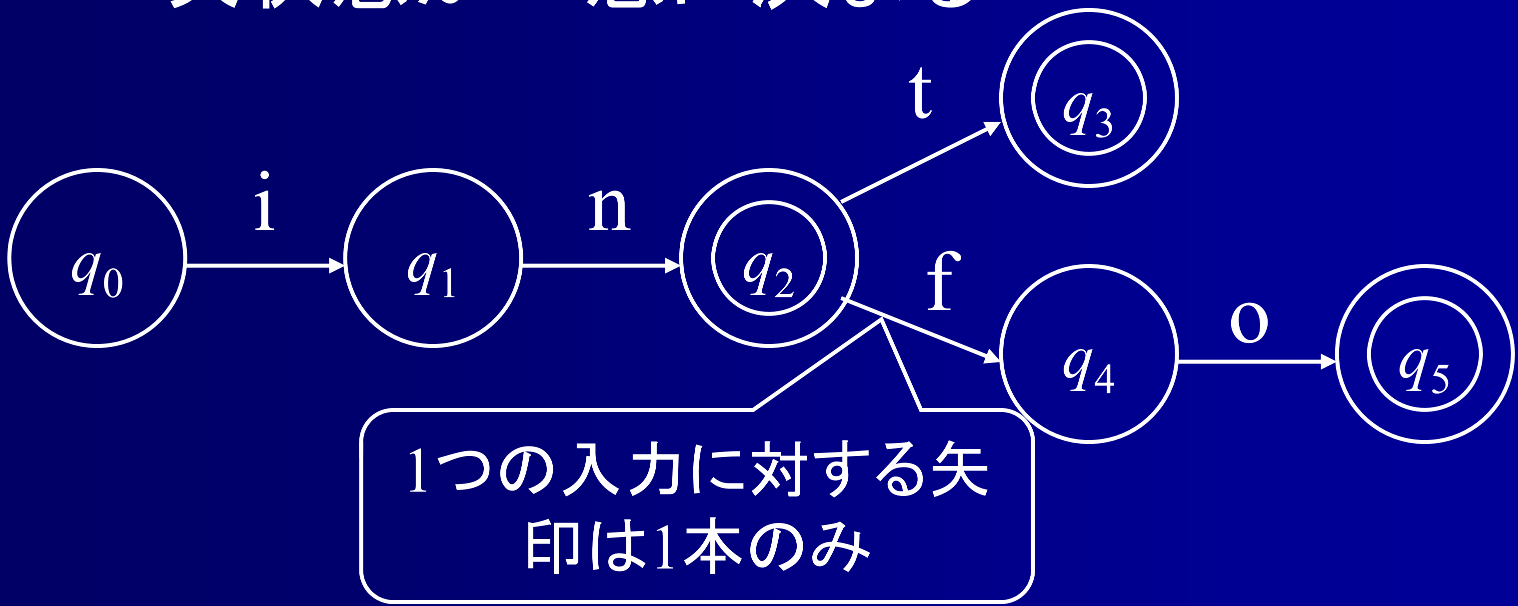
$\delta(q_0, \mathrm{i}) = q_1$, $\delta(q_1, \mathrm{n}) = q_2$, $\delta(q_2, \mathrm{t}) = q_3$,
$\delta(q_2, \mathrm{f}) = q_4$, $\delta(q_4, \mathrm{o}) = q_5$

$\mathbf{F} = \{q_2, q_3, q_5\}$



決定性有限オートマトン (deterministic finite automaton)

- 現在の状態と入力が決定
  ⇒次状態が一意に決まる

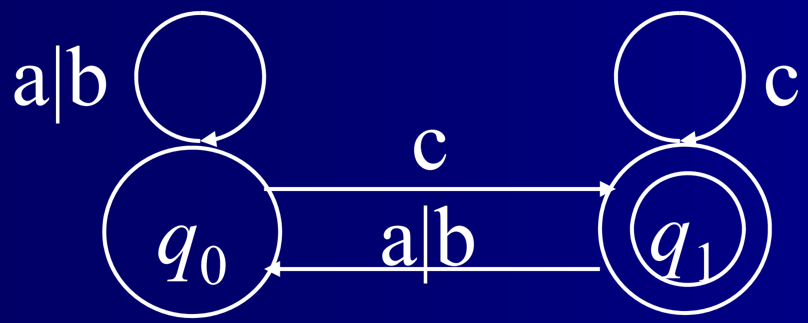


⇔非決定性オートマトン

決定性オートマトンの入力の受理

- 状態遷移で最終状態に到達すれば受理

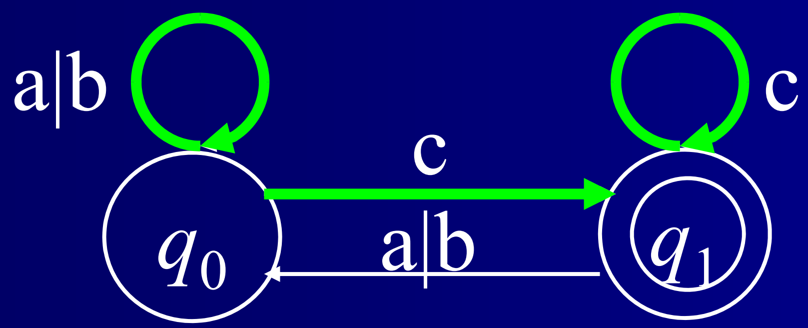
例：c で終わる文字列を受理 ($\Sigma=\{a,b,c\}$)



決定性オートマトンの入力の受理

- 状態遷移で最終状態に到達すれば受理

例：c で終わる文字列を受理 ($\Sigma=\{a,b,c\}$)

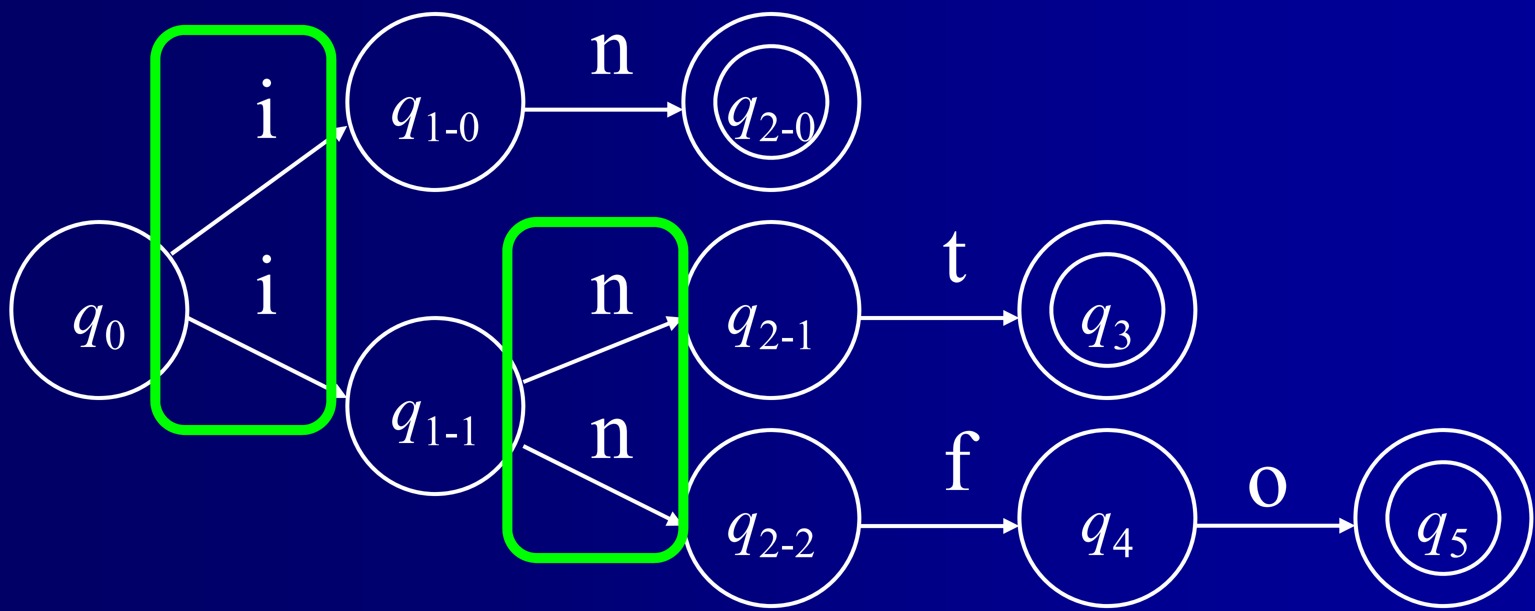


入力 acc

受理

非決定性有限オートマトン (non-deterministic finite automaton)

- 現在の状態と入力が決定
  ⇒次状態が複数あり得る

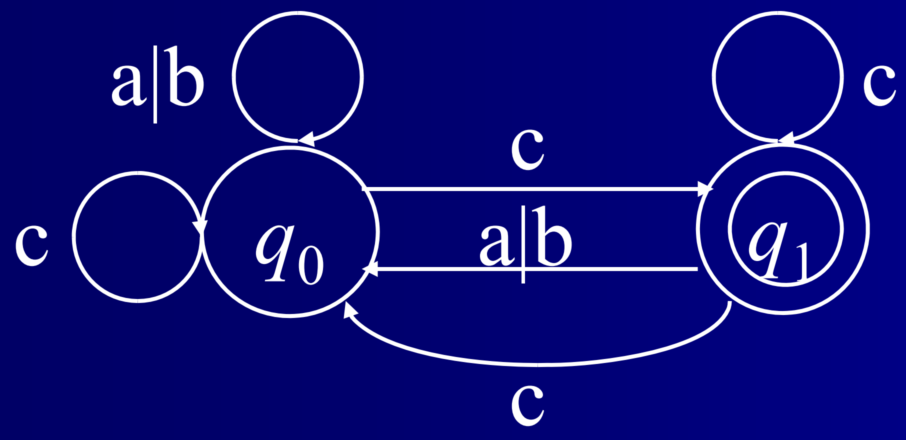


同じ入力に対する次状態が複数

非決定性オートマトンの入力の受理

- 状態遷移のうち**どれか一つ**が最終状態に到達すれば受理

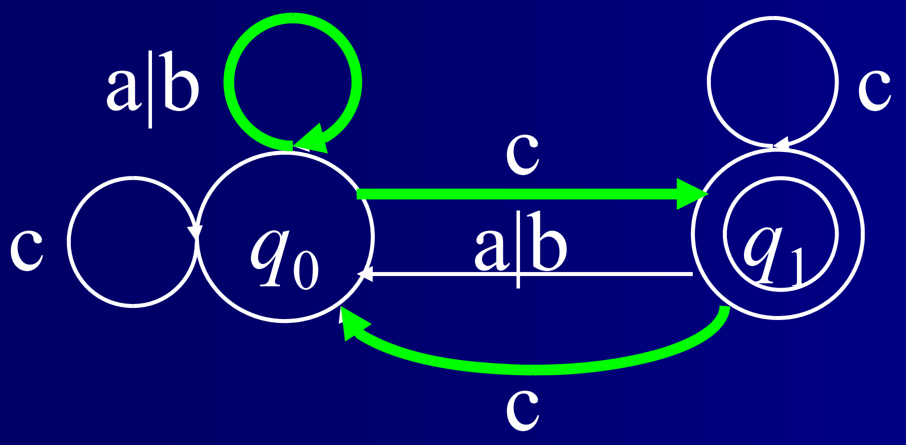
例 : c で終わる文字列を受理 (Σ={a,b,c})



非決定性オートマトンの入力の受理

- 状態遷移のうち**どれか一つ**が最終状態に到達すれば受理

例 : c で終わる文字列を受理 (Σ={a,b,c})

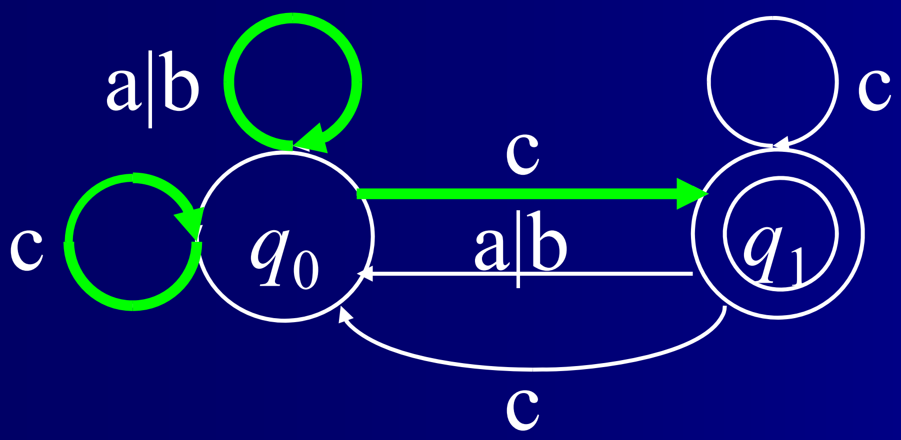


入力 acc

非決定性オートマトンの入力の受理

- 状態遷移のうちどれか一つが最終状態に到達すれば受理

例 : c で終わる文字列を受理 (Σ={a,b,c})

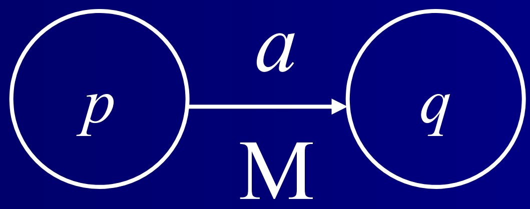


入力 acc

受理

オートマトンの状態遷移

- $M = (\mathbf{Q}, \mathbf{\Sigma}, \delta, q_0, \mathbf{F})$
  - $\delta(p, a) = q$



以下 M, ◯は省略

$$p \xrightarrow{a} q$$

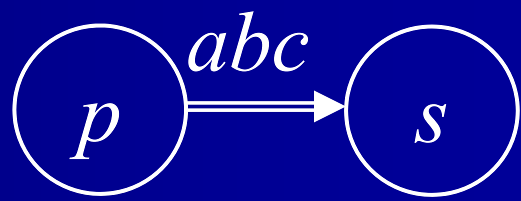
拡張状態遷移

- 入力列に対する状態遷移

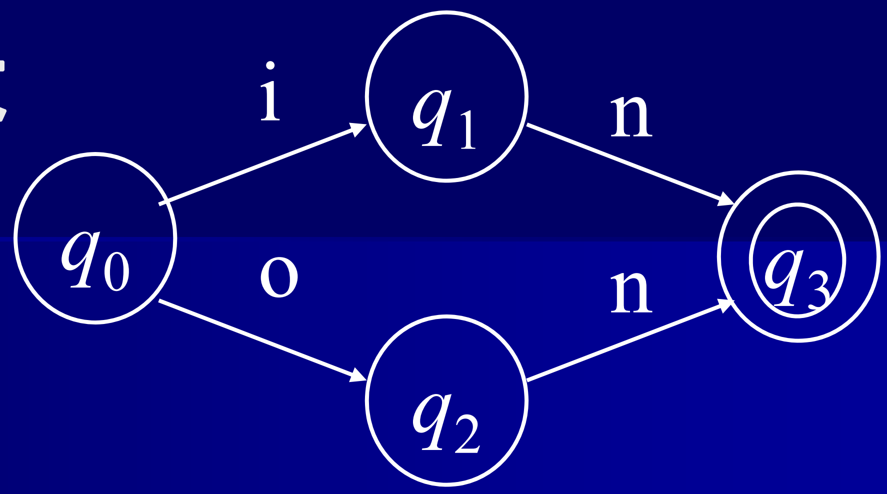
$$\delta(p, a) = q,\ \delta(q, b) = r,\ \delta(r, c) = s$$

$$p \xrightarrow{a} q \xrightarrow{b} r \xrightarrow{c} s$$

$$\delta(p, abc) = s \qquad p \overset{abc}{\Longrightarrow} s$$



状態遷移表



現状態	入力	次状態
q_0	i	q_1
q_0	o	q_2
q_1	n	q_3
q_2	n	q_3

本来なら |Q|*|Σ|行の表が必要

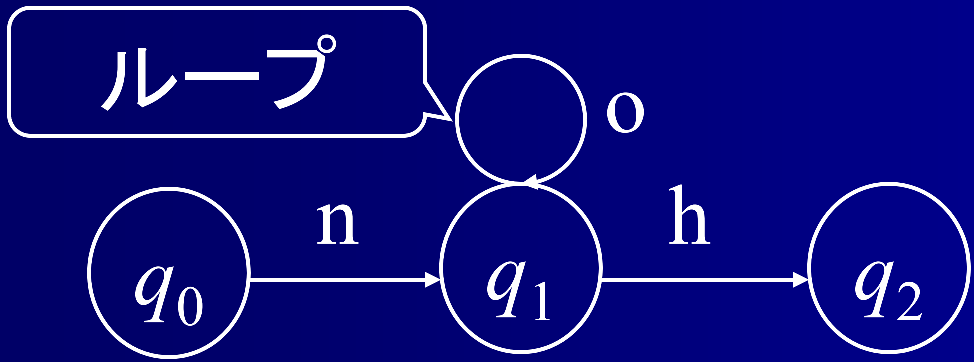
しかし大部分の行は次状態無し(不受理)

表に無い入力は不受理

状態遷移表

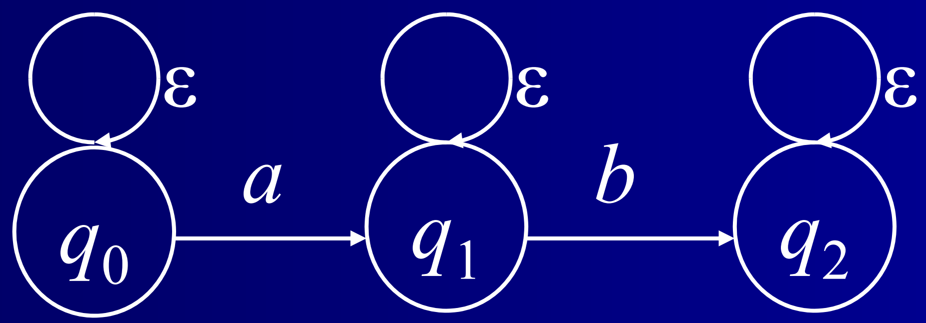
現状態	入力	次状態
q_0	n	q_1
q_1	o	q_1
q_1	h	q_2

現状態=次状態
⇔ループ



空記号列, 空系列

- ε : 空記号列, 空系列(長さ0の記号列)
  – $ab = \varepsilon ab = a\varepsilon b = ab\varepsilon$　　注意: $\varepsilon \notin \Sigma$



ループ以外のε-動作がある

＝ 入力が無くても状態遷移

⇒非決定性オートマトンになる

繰り返し

- $a \in \Sigma$に対して
  - $a^0 = \varepsilon,\ a^1 = a,\ a^2 = aa,\ a^3 = aaa,\ \ldots$
- $\Sigma = \{0, 1\}$ のときΣに対して
  - $\Sigma^0 = \varphi,$
  - $\Sigma^1 = \{0, 1\},$
  - $\Sigma^2 = \{00, 01, 10, 11\},$
  - $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\},$
  - …

閉包

$$a^* = \bigcup_{k=0}^{\infty} a^k = a^0 \cup a^1 \cup a^2 \cup a^3 \cup \ldots$$
$$= \{\varepsilon, a, aa, aaa, \ldots.\}$$
$$a^+ = \bigcup_{k=1}^{\infty} a^k = a^1 \cup a^2 \cup a^3 \cup \ldots$$
$$= \{a, aa, aaa, \ldots.\}$$
$$\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$$
$$= \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \ldots.\}$$
$$\Sigma^+ = \bigcup_{k=1}^{\infty} \Sigma^k = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$$
$$= \{0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \ldots.\}$$

($\mathbf{\Sigma}=\{0,1\}$のとき)

集合表現

- $\{a, aa, aaa, aaaa, \ldots\} = \{a^i \mid i \geqq 1\} = a^+$
- $\{\varepsilon, a, aa, aaa, aaaa, \ldots\} = \{a^i \mid i \geqq 0\} = a^*$
- $\{ab, aab, abb, aaab, aabb, abbb, \ldots\}$

  $= \{a^i b^j \mid i \geqq 1, j \geqq 1\} = a^+ b^+$
- $\{ab, aabb, aaabbb, aaaabbbb, \ldots.\}$

  $= \{a^i b^i \mid i \geqq 1\}$
- $\{ab, abab, ababab, abababab, \ldots\}$

  $= \{(ab)^i \mid i \geqq 1\} = (ab)^+$

コンパイラの構造

- 字句解析系　⇒ 有限オートマトン
- 構文解析系　⇒ 形式言語と形式文法
- 制約検査系
- 中間コード生成系
- 最適化系
- 目的コード生成系

文法

- 文法の例 日本語の場合
  - 「文」は「主語」「述語」から成る
  - 「主語」は「名詞」「助詞」から成る
  - 「述語」は「動詞」から成る
  - 「名詞」は「私」または「君」である
  - 「助詞」は「が」または「は」または「も」である
  - 「動詞」は「笑う」または「歌う」である

私が歌う　私は~~踊る~~　「踊る」は「動詞」では無い

君も笑う　~~君も私も~~歌う　「主語」「主語」「述語」は「文」ではない

~~歌う君が~~　「述語」「主語」は「文」ではない

形式文法(formal grammer)

- 形式文法 G = (**N**, **T**, S, **P**)
  - **N** : 非終端記号の有限集合
  - **T**: 終端記号の有限集合 (**N**∩**T**=φ)
  - S : 開始記号 (S∈**N**)
  - **P** : 生成規則の有限集合
    - α→β (α,β∈(**N**∪**T**)*)
      (非終端記号と終端記号とから成る文字列)

形式文法の例

- G = (**N**, **T**, 文, **P**)
  - **N** = {文, 主語, 述語, 名詞, 助詞, 動詞}
  - **T** = {私, 君, が, は, も, 笑う, 歌う}
  - **P** = {文→主語述語,

    主語→名詞助詞

    述語→動詞,

    名詞 →私, 名詞→君,

    助詞→が, 助詞→は, 助詞→も

    動詞→笑う, 動詞→歌う}

形式文法の例

- G = (**N**, **T**, S, **P**)
  - **N** = {S, B}
  - **T** = {a,b,c}
  - **P** = {S→abc, S→aBSc, Ba→aB, Bb→bb, S→ε}

生成規則α→β:文字列αを文字列βに置き換える

S

S→aBSc

aB**S**c→aB**abc**c

a**Ba**bcc→a**aB**bcc

aa**Bb**cc→aa**bb**cc

形式文法の例

- G = (N, T, 文, P)
  - N = {文, 主語, 述語, 名詞, 助詞, 動詞}
  - T = {私, 君, が, は, も, 笑う, 歌う}
  - P = {文→主語述語, 主語→名詞助詞, 述語→動詞,
    名詞 →私, 名詞→君,
    助詞→が, 助詞→は, 助詞→も
    動詞→笑う, 動詞→歌う}

文→主語述語 →名詞助詞動詞 →私が笑う

文→主語述語 →名詞助詞動詞 →君は歌う

導出(derivation)

- 導出
  - ある$\alpha \in (N \cup T)^*$から$\beta \in (N \cup T)^*$を生成規則を 有限回用いて導き出すこと
  - 書式：$\alpha \Rightarrow \beta$
    - $\alpha_1 \rightarrow \alpha_2 \rightarrow \alpha_3 \rightarrow \ldots \rightarrow \alpha_n$ のとき $\alpha_1 \Rightarrow \alpha_n$

$S \rightarrow aBSc \rightarrow aBabcc \rightarrow aaBbcc \rightarrow aabbcc$

$S \Rightarrow aabbcc$　　S から aabbcc を導出

形式言語(formal language)

- 形式言語L(G)
  - $\{\omega \mid \omega \in T^{*}, S \Rightarrow \omega\}$
  - 形式文法Gにより開始記号Sから導出される終端記号から成る文字列の集合

形式言語の例

- G = (N, T,文, P)
  - N = {文, 主語, 述語, 名詞, 助詞, 動詞}
  - T = {私, 君, が, は, も, 笑う, 歌う}
  - P = {文→主語述語, 主語→名詞助詞, 述語→動詞,
    名詞 →私, 名詞→君,
    助詞→が, 助詞→は, 助詞→も
    動詞→笑う, 動詞→歌う}

L(G) = {私が笑う, 私が歌う, 君が笑う, 君が歌う,
私は笑う, 私は歌う, 君は笑う, 君は歌う,
私も笑う, 私も歌う, 君も笑う, 君も歌う}

形式言語の例

- G = (**N**, **T**, S. **P**)
  - **N** = {S, B}
  - **T** = {a,b,c}
  - **P** = {S→abc, S→aBSc, Ba→aB, Bb→bb, S→ε}

S→ε

S→abc

S→aBSc→aBabcc→aaBbcc→aabbcc

S→aBSc→aBaBScc→aBaBabccc

→aBaaBbccc→aBaabbccc→aaBabbccc

→aaaBbbccc→aaabbbccc

$L(G)=\{a^n b^n c^n \mid n \geqq 0\}$

最左導出(left most derivation)

- 一番左にある非終端記号から置き換える

例：$\mathbf{N}=\{S,A,B,C,D\}$

$\mathbf{T}=\{a,b,c,d\}$

$\mathbf{P}=\{S\rightarrow ABC,\ A\rightarrow a,\ B\rightarrow bD,\ C\rightarrow c,\ D\rightarrow d\}$

$\mathbf{S}\rightarrow\mathbf{ABC}$

$\mathbf{A}BC\rightarrow\mathbf{a}BC$

$a\mathbf{B}C\rightarrow a\mathbf{bD}C$

$ab\mathbf{D}C\rightarrow ab\mathbf{d}C$

$abd\mathbf{C}\rightarrow abd\mathbf{c}$

⇔最右導出(right most derivation)

最左導出の利点

- 左から右に順に置き換えていけばいい
  - 左に戻る必要が無い

abcd **E**FGhIjK

↑これを変換

abcd**e** FGhIjK

← これより前は変換済

⇒変換場所を左に戻す必要が無い

チョムスキー階層 (Chomsky hierarchy)

- 形式文法のクラスの包含階層

階層	文法	生成規則
Type-0		制限無し
Type-1	文脈依存文法	$\alpha A\beta \rightarrow \alpha\gamma\beta$
Type-2	文脈自由文法	$A \rightarrow \gamma$
Type-3	正規文法	$A \rightarrow a$ or $A \rightarrow aB$

$A,B \in N$, $a \in T$,

$\alpha,\beta,\gamma \in (N \cup T)^*$ (終端記号と非終端記号で構成される文字列)

チョムスキー階層と受理可能な文法

文法	生成規則	受理可能な文法例
文脈依存文法	$\alpha A\beta \rightarrow \alpha\gamma\beta$	$a^n b^n c^n \ (n \geqq 0)$
文脈自由文法	$A \rightarrow \gamma$	$a^n b^n \quad (n \geqq 0)$
正規文法	$A \rightarrow a$ or $A \rightarrow aB$	$a^n, b^n \ (n \geqq 0)$

$A,B \in N$, $a,b,c \in T$,

$\alpha,\beta,\gamma \in (N \cup T)^*$ (終端記号と非終端記号で構成される文字列)

文脈自由文法 (context-free grammer)

$A \longrightarrow \gamma \ (A \in \mathbf{N}, \gamma \in ((\mathbf{N}\text{-S}) \cup \mathbf{T})^{*})$

左辺	非終端記号1つ
右辺	S以外の非終端記号と終端記号の列

例外: 開始記号 S のみ $S \rightarrow \varepsilon$ も可

文脈自由文法の例

- $G = (\mathbf{N}, \mathbf{T}, S, \mathbf{P})$
  - $\mathbf{N} = \{S, A\}$
  - $\mathbf{T} = \{a, b\}$
  - $\mathbf{P} = \{S \to A,\ A \to ab,\ A \to aAb,\ S \to \varepsilon\}$

左辺は非終端記号1個

右辺は終端記号と非終端記号の列

εはS→εのみ可

$S \to \varepsilon$

$S \to A \to ab$

$S \to A \to aAb \to aabb$

$S \to A \to aAb \to aaAbb \to aaabbb$

$L(G) = \{a^n b^n \mid n \geqq 0\}$

正規文法, 正則文法 (regular grammer)

$A \to a$, $A \to aB$ ($A \in \mathbf{N}$, $B \in (\mathbf{N}\text{-}S)$, $a \in \mathbf{T}$)

左辺	非終端記号1つ
右辺	終端記号 or 終端記号･S以外の非終端記号

例外: 開始記号 S のみ $S \to \varepsilon$ も可

正規文法で生成される言語は
有限オートマトンで受理可能

正規文法の例

- $G = (\mathbf{N}, \mathbf{T}, S, \mathbf{P})$
  - $\mathbf{N} = \{S, B\}$
  - $\mathbf{T} = \{a, b\}$
  - $\mathbf{P} = \{S \rightarrow a,\ S \rightarrow b,\ S \rightarrow aA,\ S \rightarrow bB,\ A \rightarrow a,$
    $A \rightarrow aA,\ B \rightarrow b,\ B \rightarrow bB,\ S \rightarrow \varepsilon\}$

左辺は非終端記号1個

右辺は終端記号1個 or
終端記号1個・非終端記号1個

$S \rightarrow \varepsilon$

$S \rightarrow a \quad S \rightarrow b$

$S \rightarrow aA \rightarrow aa \quad S \rightarrow bB \rightarrow bb$

$S \rightarrow aA \rightarrow aaA \rightarrow aaa \quad S \rightarrow bB \rightarrow bbB \rightarrow bbb$

$L(G) = \{a^n, b^n \mid n \geqq 0\}$

正規文法の例

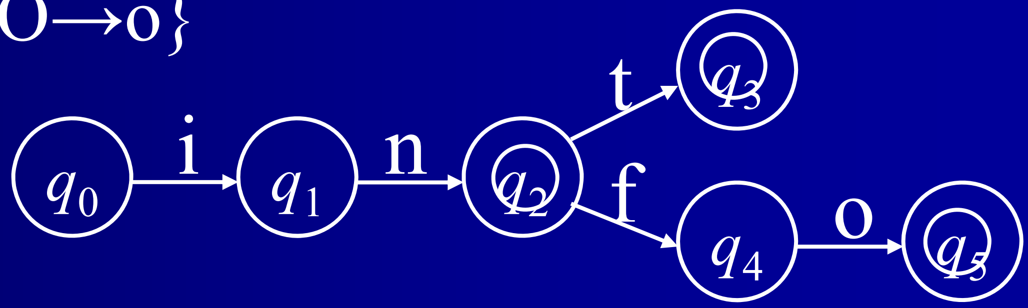
in, int, info を受理する正規文法

- G = (**N**, **T**, S, **P**)
  - **N** = {S,N,T,F,O}
  - **T**= {i,n,t,f,o}
  - **P** = {S→iN, N→n, N→nT, N→nF, T→t, F→fO, O→o}

S→iN→in

S→iN→inT→int

S→iN→inF→infO→info



L(G) = {in, int, info}

正規言語, 正則言語 (regular language)

- 有限オートマトンで受理される言語

$$L(M) = \{\ \alpha \in \boldsymbol{\Sigma}^* \mid q_0 \overset{\alpha}{\Rightarrow} r,\ r \in \mathbf{F}\ \}$$

$$= \{\ \alpha \in \boldsymbol{\Sigma}^* \mid \delta(q_0, \alpha) \in \mathbf{F}\ \}$$

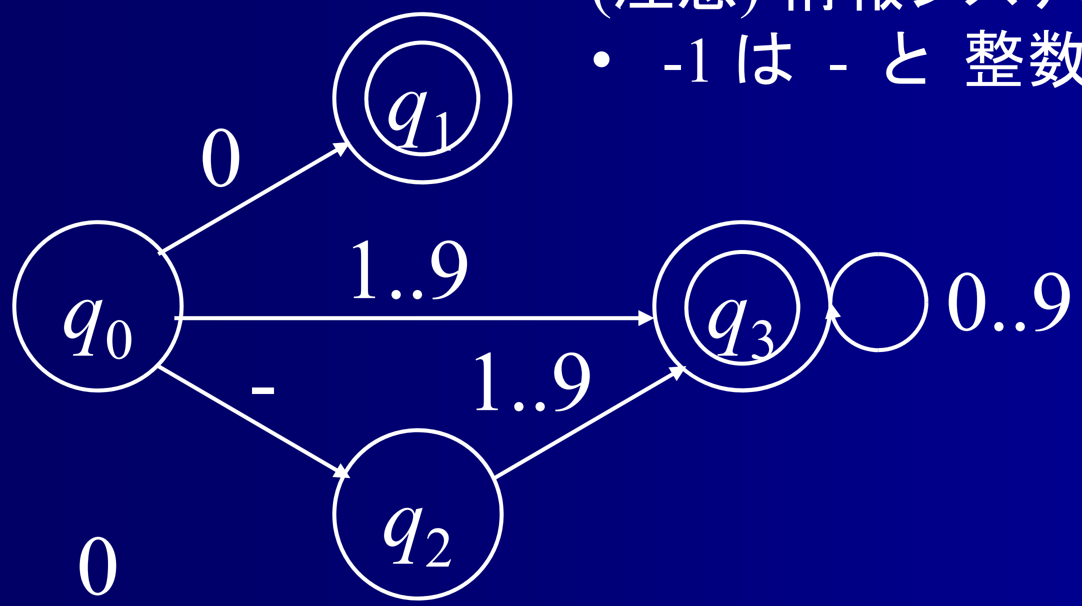
例1 : $\{a^{2n+1} \mid n \geqq 1\ \}$ {a, aaa, aaaaa, aaaaaaa,...}

例2 : a{b|c}* {a, ab, ac, abb,abc, acb, acc,…}

正規言語の例

- 整数を受理するオートマトン

(注意) 情報システムプロジェクトIでは
- -1 は - と 整数1 と判別



0

1 2 3 4 5 6 7 8 9 　 -1 -2 -3 -4 -5 -6 -7 -8 -9

10 11 12 13 14 … 　 -10 -11 -12 -13 -14 …

100 101 102 103 104…

正規言語の例

- 整数を生成する正規文法
  - **N** $=\{S, P, D\}$
  - **T** $=\{0,1,2,3,4,5,6,7,8,9,-\}$
  - **P** $=\{S\rightarrow 0,\ S\rightarrow -P,$
    $S\rightarrow 1,\ S\rightarrow 2,\ S\rightarrow 3,\ \ldots,\ S\rightarrow 9,$
    $S\rightarrow 1D,\ S\rightarrow 2D,\ S\rightarrow 3D,\ \ldots,\ S\rightarrow 9D,$
    $P\rightarrow 1,\ P\rightarrow 2,\ P\rightarrow 3,\ \ldots,\ P\rightarrow 9,$
    $P\rightarrow 1D,\ P\rightarrow 2D,\ P\rightarrow 3D,\ \ldots,\ P\rightarrow 9D,$
    $D\rightarrow 0,\ D\rightarrow 1,\ D\rightarrow 2,\ D\rightarrow 3,\ \ldots,\ D\rightarrow 9,$
    $D\rightarrow 0D,\ D\rightarrow 1D,\ D\rightarrow 2D,\ S\rightarrow 3D,\ \ldots,\ D\rightarrow 9D\}$

BNF記法(Buckus Naur form)

- 文法の記述法

	生成規則	BNF記法
生成則	→	::=
終端記号	$a \in T$	“文字列”
非終端記号	$A \in N$	<文字列>

例 : E→T+T, <exp> ::= <term> “+” <term>
T→0 <term> ::= “0”

非終端記号 終端記号

BNF記法の例

または

<Integer> ::= “0” | <Pdeclist> | “-” <Pdeclist>

<Pdeclist> ::= <Pdec> | <Pdec> <Declist>

<Declist> ::= <Dec> | <Dec> <Declist>

<Pdec> ::= “1” | “2” | “3” | “4” | “5”
| “6” | “7” | “8” | “9”

再帰

<Dec> ::= “0” | <Pdec>

BNF記法では繰り返しの定義には再帰が必要

EBNF記法 (extended Buckus Naur form)

- BNF記法の拡張

EBNF記法	意味
α,β	記号列αβ
α|β	αまたはβ
[α]	省略可能 (0回または1回)
{α}	省略可能な繰り返し (0回以上)

EBNF記法の例

0回または1回　　0回以上

<Integer> ::= “0” | [“-”] <Pdec> { <Dec> }

<Pdec> ::= “1” | “2” | “3” | “4” | “5”
| “6” | “7” | “8” | “9”

<Dec> ::= “0” | <Pdec>

0

1 2 3 4 5 6 7 8 9　　-1 -2 -3 -4 -5 -6 -7 -8 -9

10 11 12 13 14 …　-10 -11 -12 -13 -14 …

100 101 102 103 104…

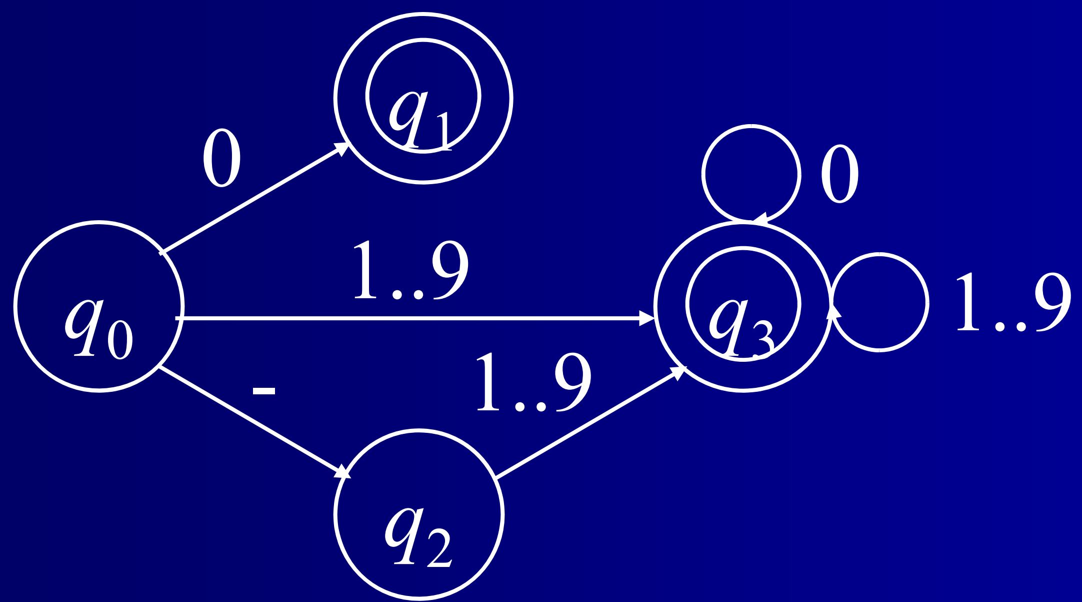
1000 1001 1002 1003 1004…

EBNF記法の例

<Integer> ::= “0” | [“-”] <Pdec> { <Dec> }

<Pdec> ::= “1” | “2” | “3” | “4” | “5”
| “6” | “7” | “8” | “9”

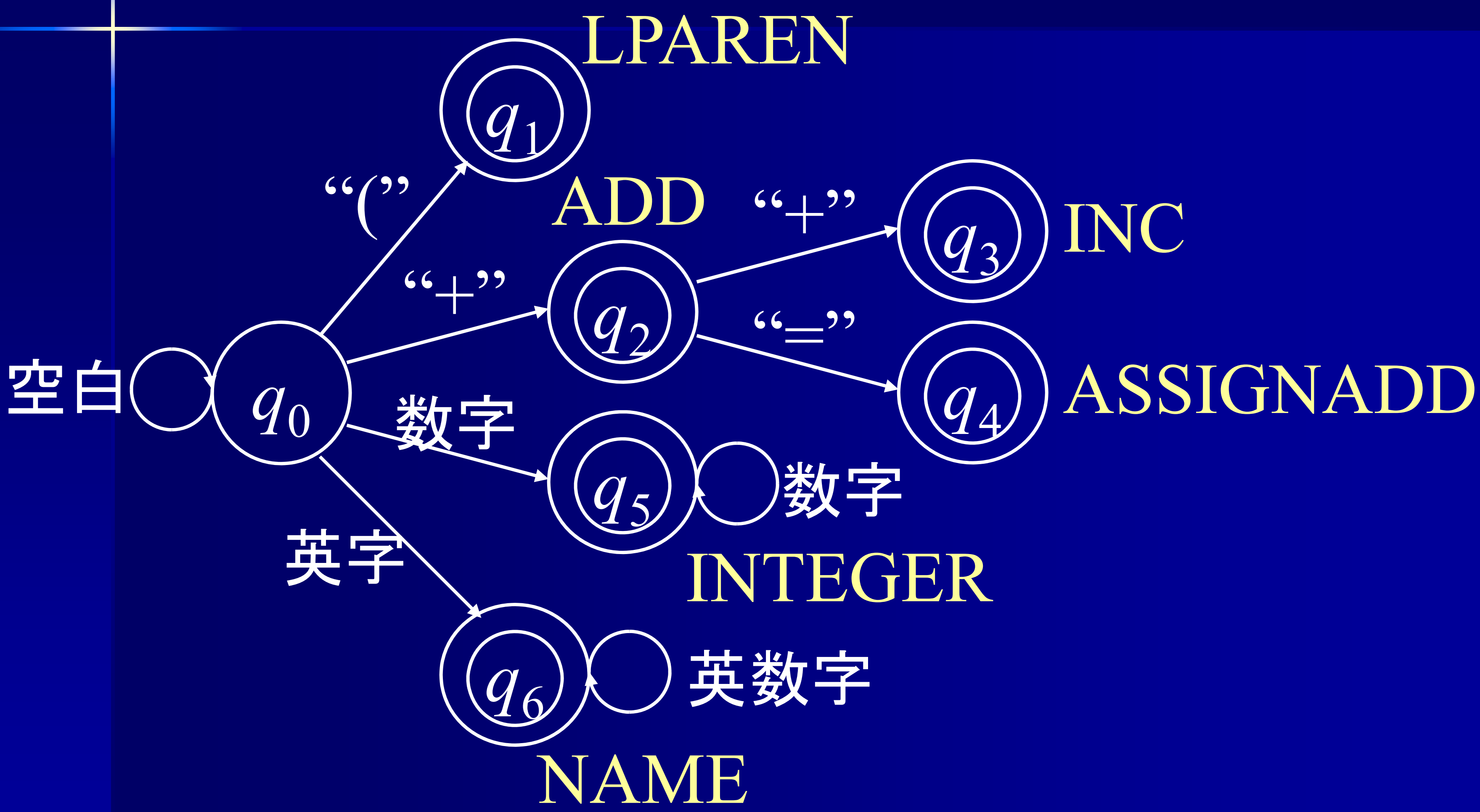
<Dec> ::= “0” | <Pdec>



正規表現とEBNF記法

意味	正規表現	EBNF記法
連接 a の後に b	ab	a , b
選択 a または b	a|b	a | b
省略可能 (0回または1回)	a?	[a]
省略可能な繰り返し (0回以上)	a*	{ a }
省略不可能な繰り返し (1回以上)	a+	a { a }

字句解析オートマトン(一部)



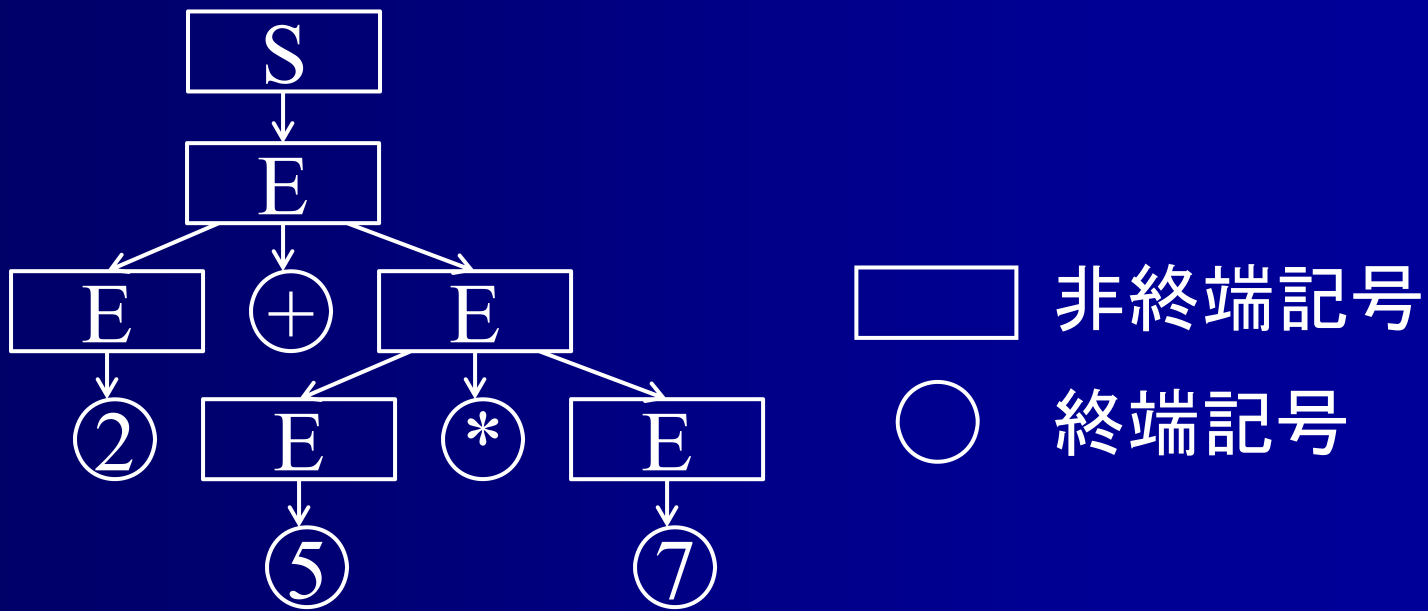
導出木(derivation tree)

- 導出を木の形で表わしたもの

例：**N**={S,E} **T**={2,5,7,+,*}

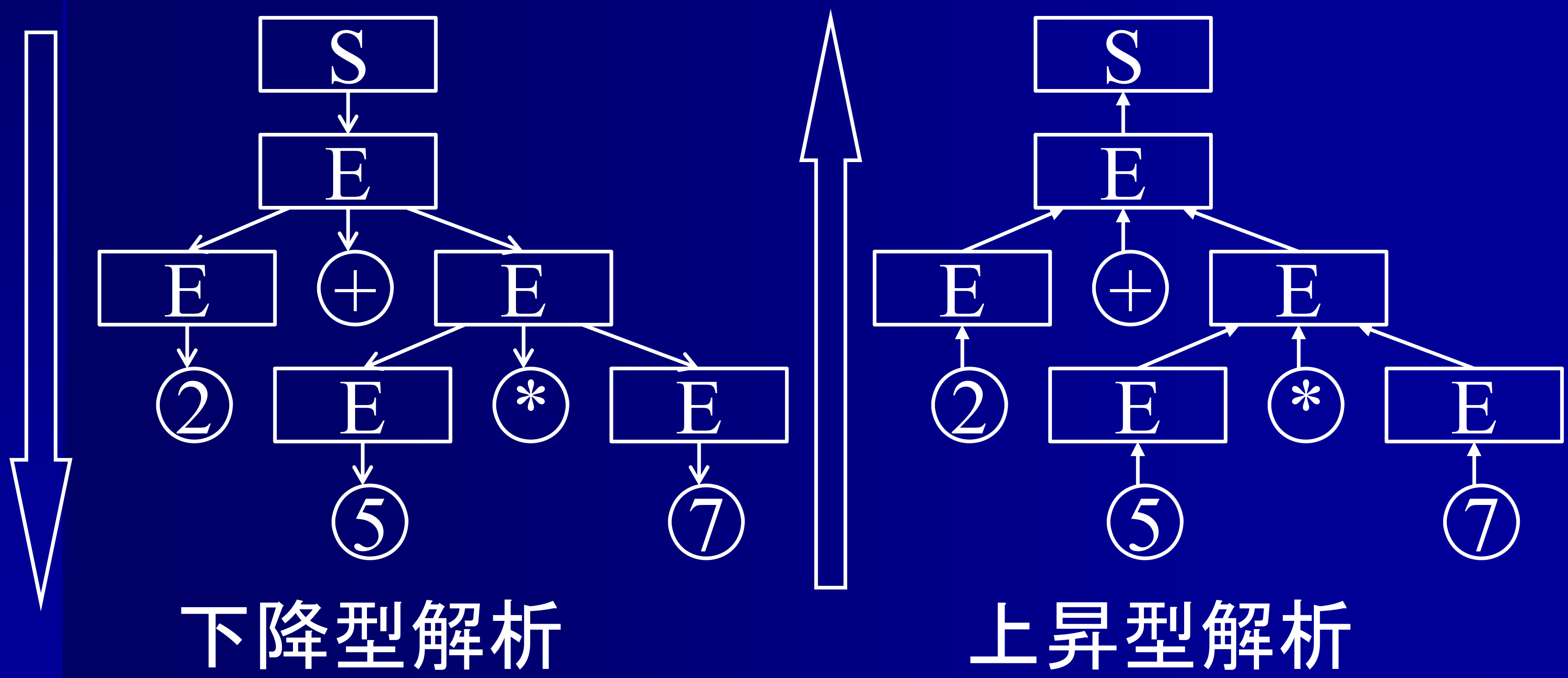
P={S→E, E→E+E, E→E*E, E→2, E→5,E→7}

S→E→E+E→2+E→2+E*E→2+5*E→2+5*7



構文解析

- $\omega \in T^*$ に対して
  $S \Rightarrow \omega$ であるか判定, その導出木を得る

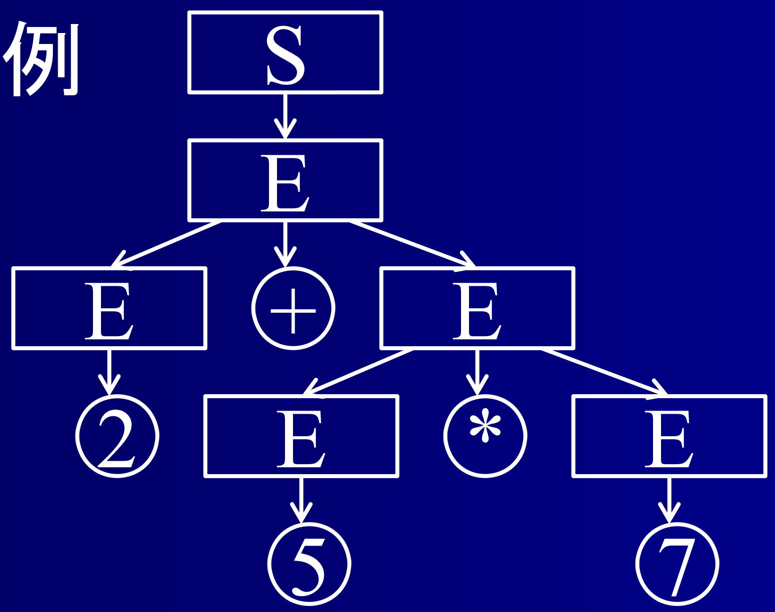


導出木

- 同一の導出木

⇒同一の文字列が得られる導出

例



最終的には
同一の文字列

S→E→E+E→2+E→2+E*E→2+5*E→2+5*7

S→E→E+E→E+E*E→E+E*7→E+5*7→2+5*7

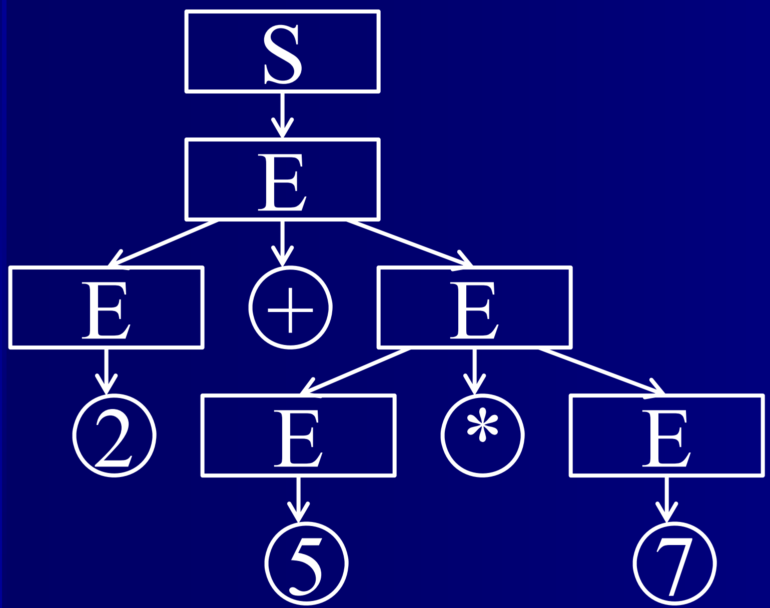
導出木

- 同一の文字列が得られる導出
  ≠同一の導出木

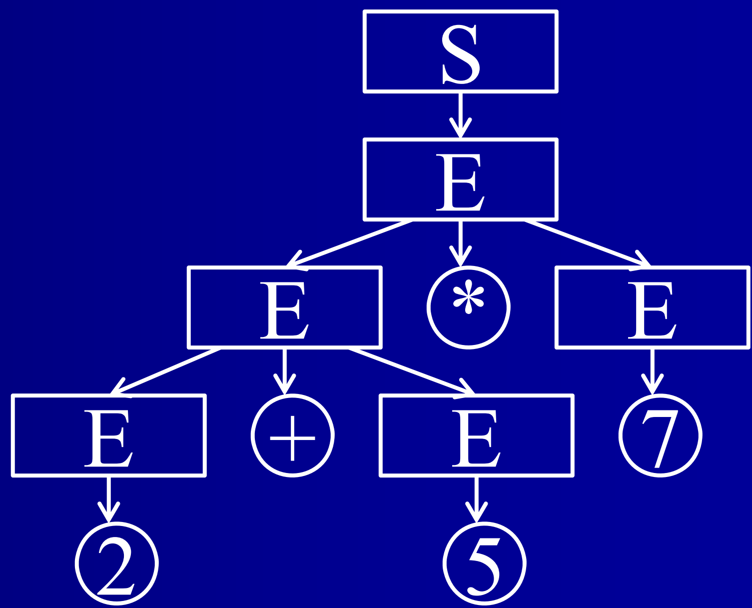
曖昧な文法

$S \to E \to E+E \to E+E*E \to 2+E*E \to 2+5*E \to 2+5*7$

$S \to E \to E*E \to E+E*E \to 2+E*E \to 2+5*E \to 2+5*7$

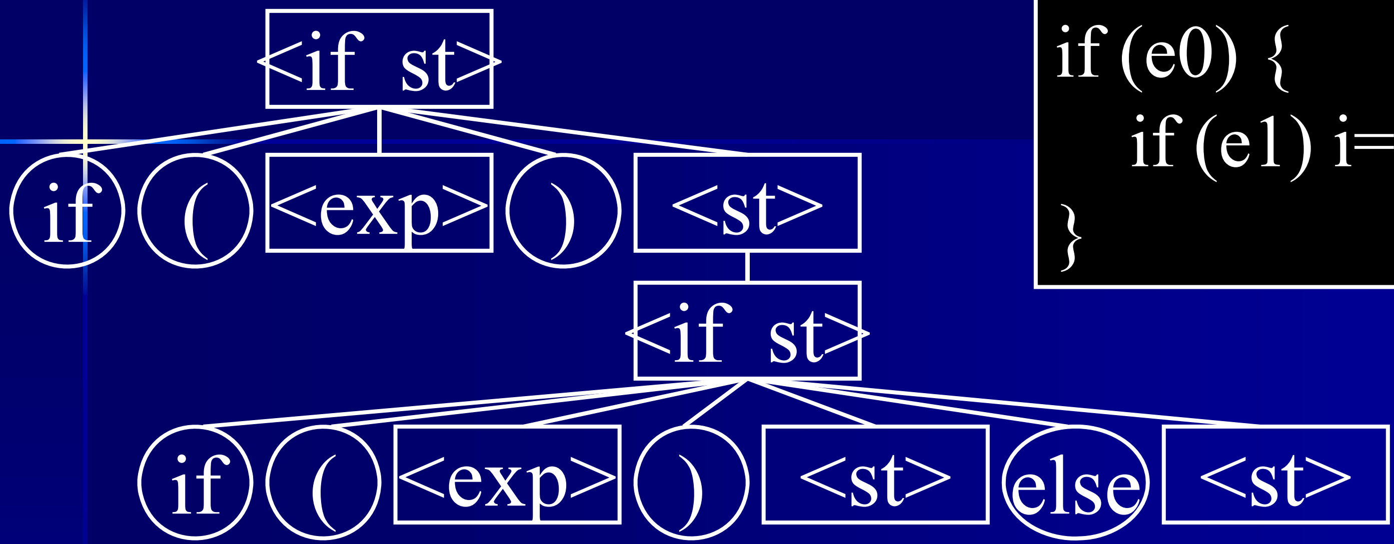


2+(5*7)

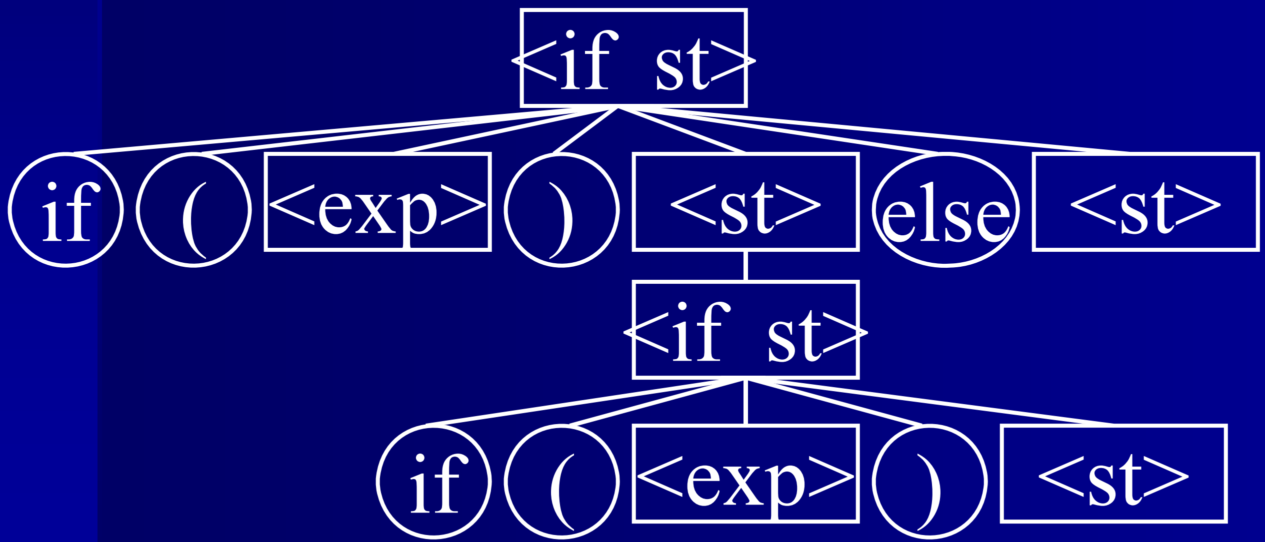


(2+5)*7

```
if (e0) if (e1) i=0; else i=1;
```




```
if (e0) {
    if (e1) i=0; else i=1;
}
```




```
if (e0) {
    if (e1) i=0;
} else i=1;
```

曖昧な文法(ambiguous)

- 曖昧な文法
  - 同一の文字列に対して異なる導出木

例：N={S,E} **T**={2,5,7,+,*}

P={S→E, E→E+E, E→E*E, E→2, E→5,E→7}

S→E→E+E→E+E*E→2+E*E→2+5*E→2+5*7

S→E→E*E→E+E*E→2+E*E→2+5*E→2+5*7

2+(5*7) と (2+5)*7 の区別ができない

⇒ 曖昧性の除去が必要

曖昧性の除去

- 手法1
  - 曖昧性が無いように文法を書き換える
- 手法2
  - 新たな制約を加える
    - 演算子の優先順位, 結合性等

曖昧性の除去

- 手法1
  - 曖昧性が無いように文法を書き換える

例：**N**={S,E} **T**={2,5,7,+,*}
P={S→E, E→E+E, E→E*E, E→2, E→5,E→7}

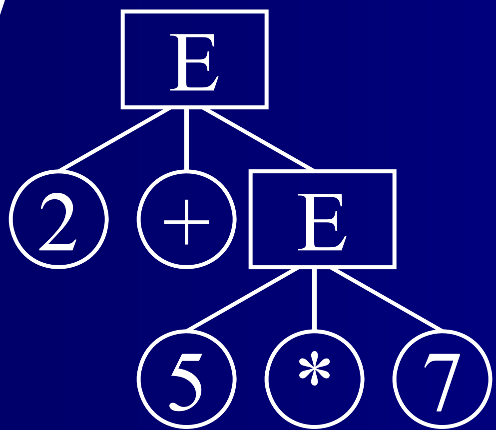
N={S,E,T,F} **T**={2,5,7,+,*}
P={S→E, E→T, E→T+E,
T→F, T→T*F,
F→2, F→5, F→7}

曖昧性の除去

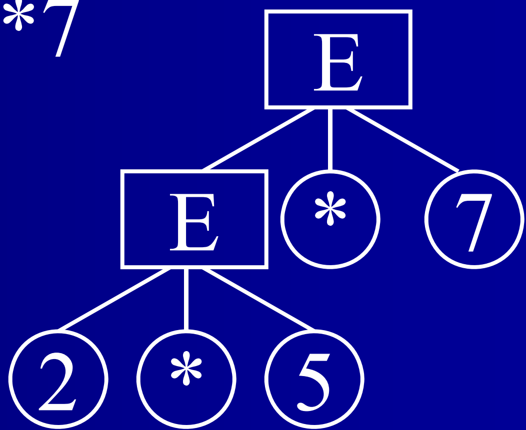
- 手法2
  - 新たな制約を加える
    - 演算子の優先順位, 結合性等

例 : 優先順位: * は + より先に計算
結合性:　* 同士, +同士は左から先に計算

2+5*7



2*5*7

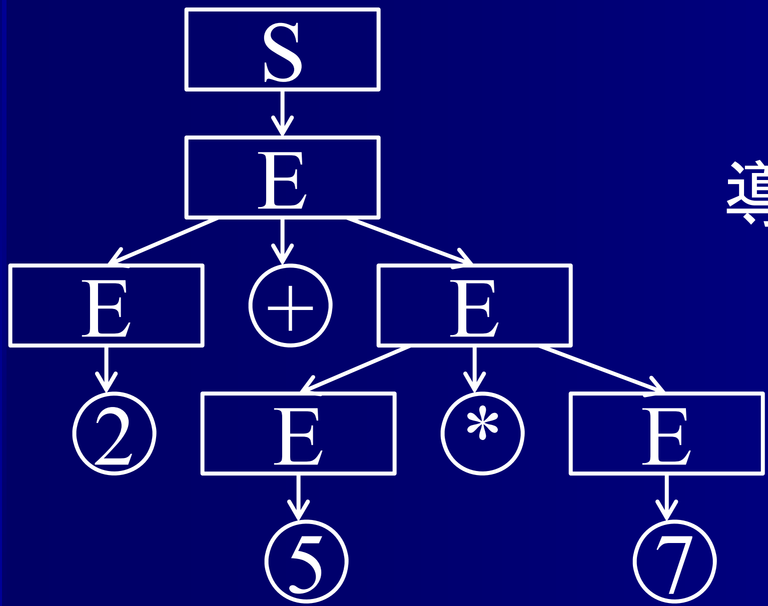


導出木と最左導出

- 同一の導出木 = 同一の最左導出

S→E→E+E→2+E→2+E*E→2+5*E→2+5*7

S→E→E+E→E+E*E→E+E*7→E+5*7→2+5*7



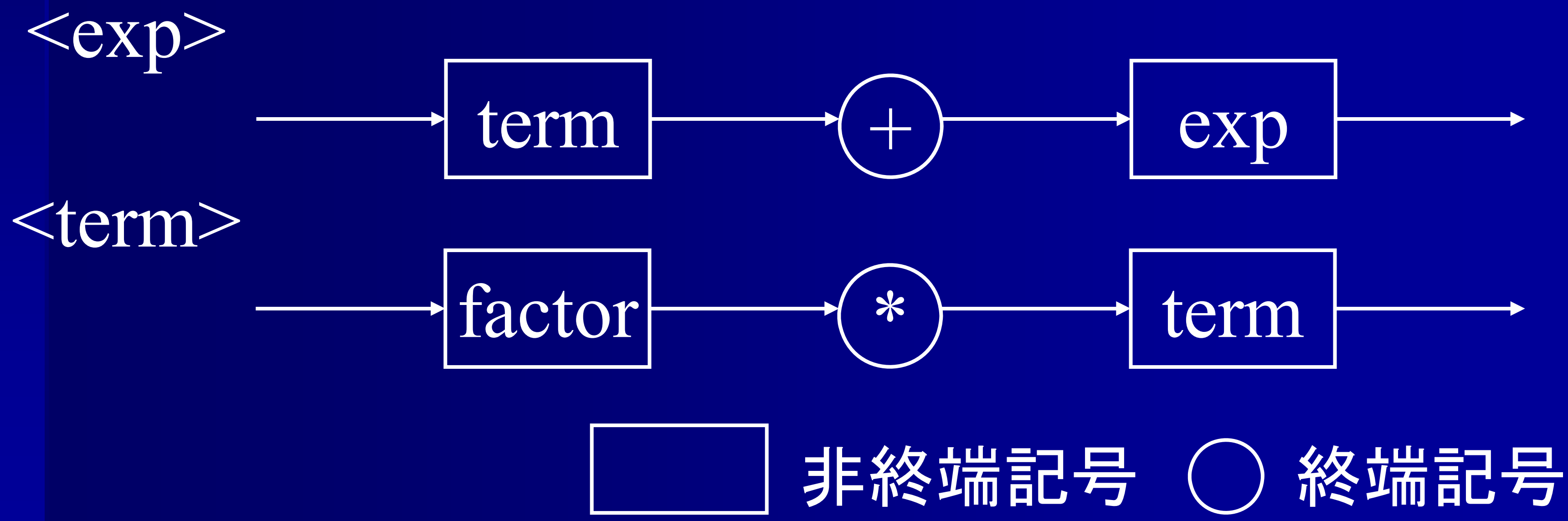
導出が異なっても同じ導出木

一つの導出木には
一つの最左導出が対応

構文図

- 構文解析の流れを示した図

例 : <exp> ::= <term> “+” <exp>
<term> ::= <factor> “*” <term>



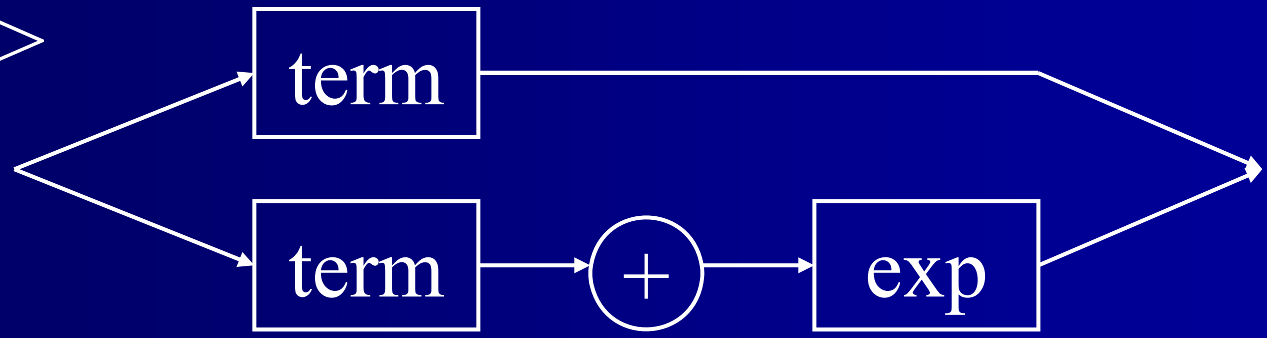
構文図

または

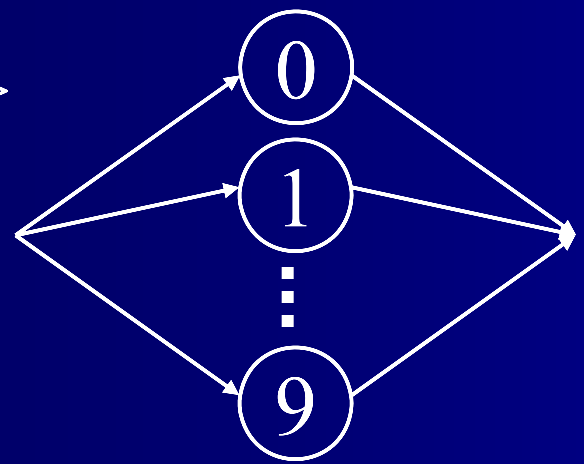
例：<exp> ::= <term> | <term> “+” <exp>

<dec> ::= “0” | “1” | “2” | “3” | … | “9”

<exp>



<dec>



構文図

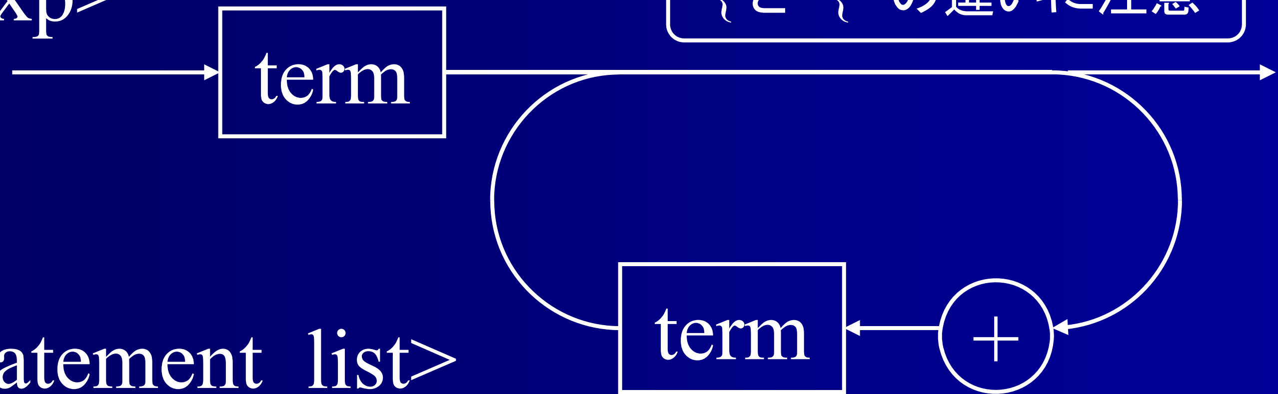
0回以上の繰り返し

例：<exp> ::= <term> { “+” <term> }

<statement_list> ::= “{” { <statement> } “}”

{ と “{” の違いに注意

<exp>



<statement_list>

構文図

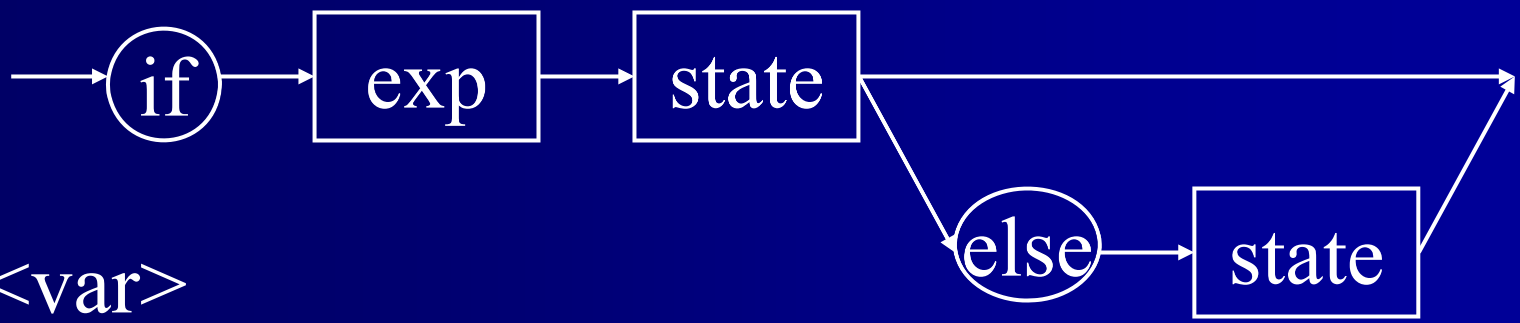
0回,または1回(省略可能)

例 : <if_state> ::= “if” <exp> <state> [“else” <state>]

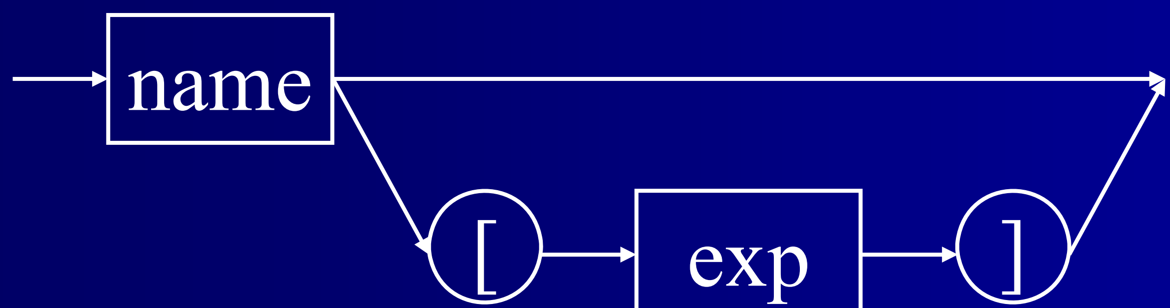
<var> ::= <name> [“[” <exp> “]”]

[と “[” の違いに注意

<if_state>



<var>



構文図

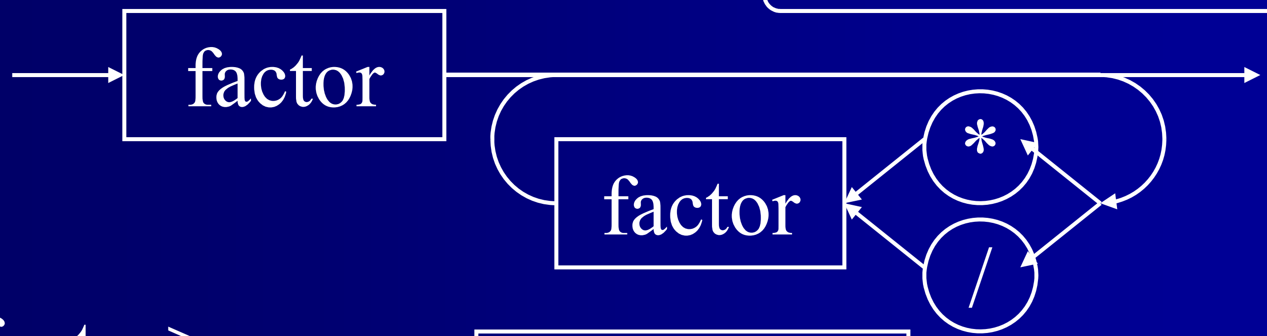
結合の優先順序を表すための括弧

例 : <term> ::= <factor> { (“*” | “/”) <factor> }

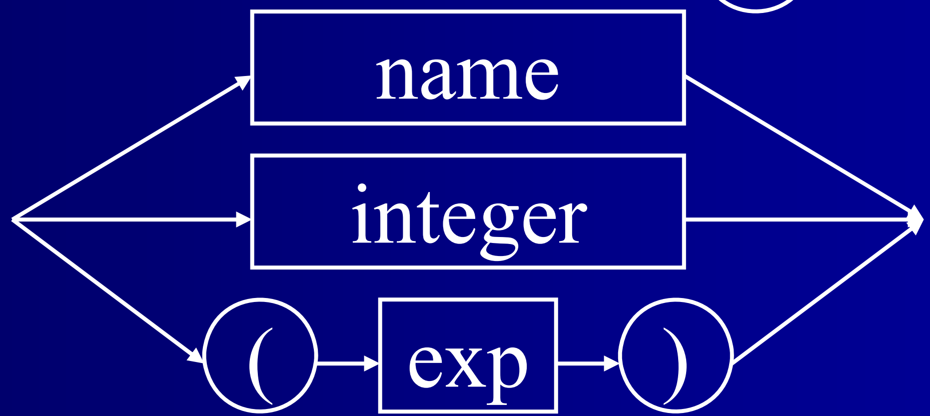
<factor> ::= <name> | <integer> | “(” <exp> “)”

(と “(” の違いに注意

<term>



<factor>



課題テスト

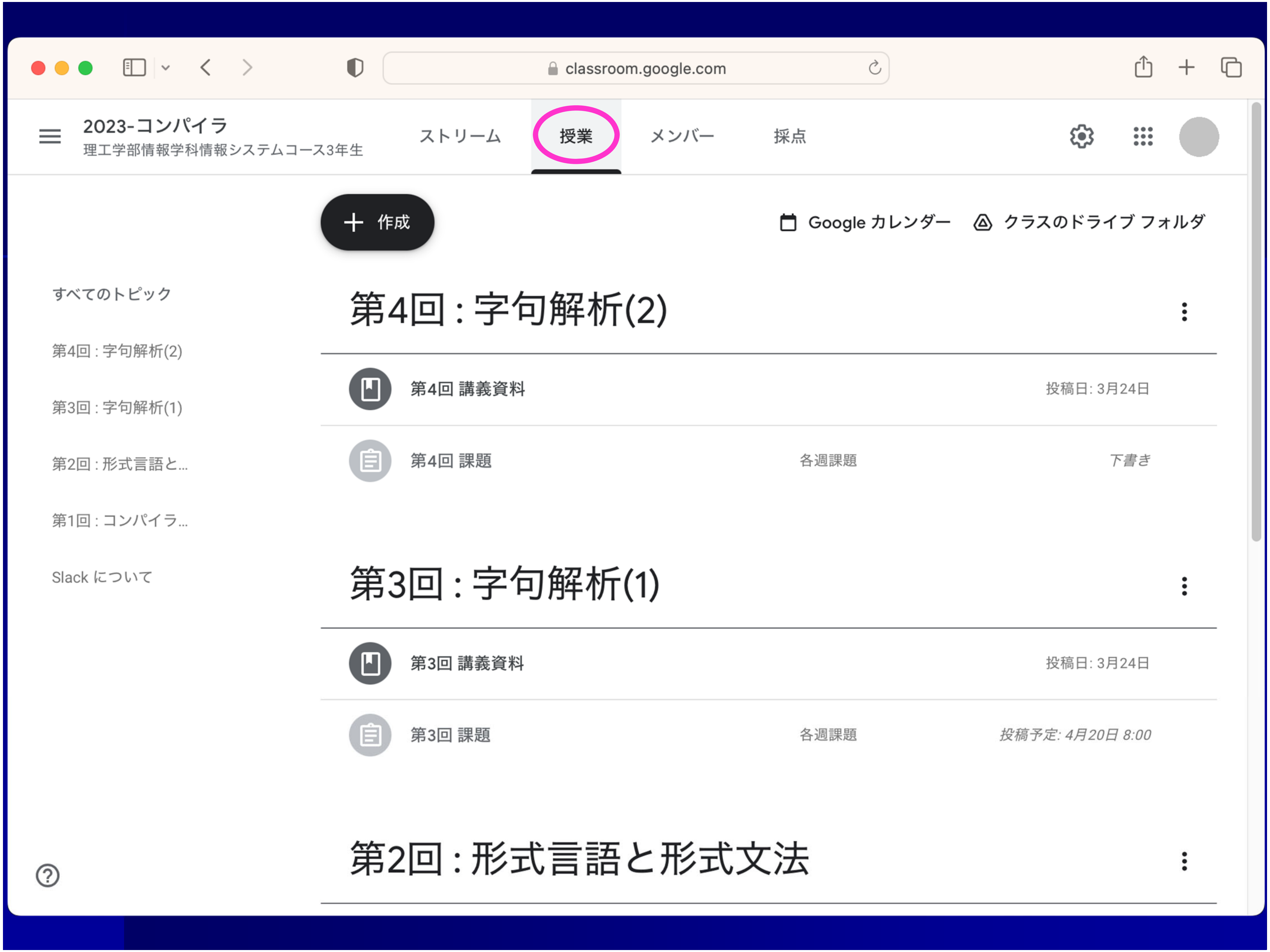
- 毎週 GoogleClassroom上で課題テストを行う
  - 授業後～翌週の授業開始まで
- GoogleClassroomで

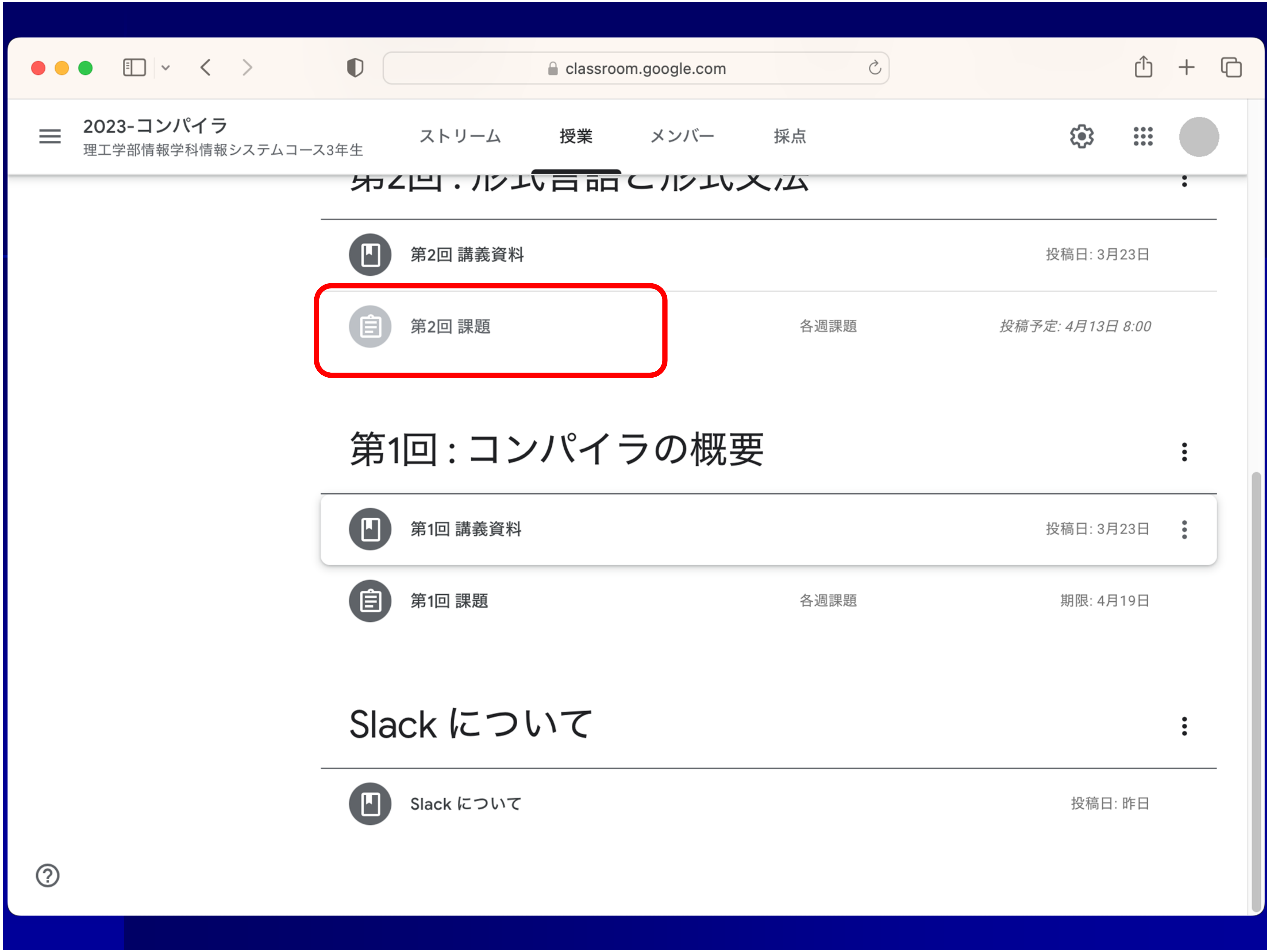
  コンパイラ

  ⇒授業

  ⇒その回の課題

  と辿る


classroom.google.com
2023-コンパイラ
理工学部情報学科情報システムコース3年生
ストリーム
授業
メンバー
採点
作成
Google カレンダー
クラスのドライブ フォルダ
すべてのトピック
第4回 : 字句解析(2)
第3回 : 字句解析(1)
第2回 : 形式言語と...
第1回 : コンパイラ...
Slack について
第4回 : 字句解析(2)
第4回 講義資料
投稿日: 3月24日
第4回 課題
各週課題
下書き
第3回 : 字句解析(1)
第3回 講義資料
投稿日: 3月24日
第3回 課題
各週課題
投稿予定: 4月20日 8:00
第2回 : 形式言語と形式文法


classroom.google.com
2023-コンパイラ
理工学部情報学科情報システムコース3年生
ストリーム
授業
メンバー
採点
第2回 講義資料
投稿日: 3月23日
第2回 課題
各週課題
投稿予定: 4月13日 8:00
第1回：コンパイラの概要
第1回 講義資料
投稿日: 3月23日
第1回 課題
各週課題
期限: 4月19日
Slack について
Slack について
投稿日: 昨日