形式言語とオートマトン

http://ritswiki.ath.cx/?/

情報理工/情コミ?

- オートマンスーパーリンク！[*1]
  - 講義ノート
- 形式言語とオートマトン
  - 形式言語の種類
  - オートマトンの種類
  - 形式言語とオートマトンの関係
- 過去問解答案
  - 2002年後期中間 2002.11.26
  - 2002年後期期末 2003.01.14
  - 2003年後期期末 2004.01.27
  - 2004年後期中間 2005.01.08
  - 2004年後期期末 2005.02.01
  - 2006年後期期末 2007.01.29

形式言語とオートマトン [↑]

オートマトン(Automaton)
抽象機械(Abstract Machine)または自動機械(Automaton)とは、自律的に機械で計算を行う過程を形式的に取扱うために考えられた数学的モデルのことである。形式的に記号列が言語の要素かどうかを判定する手段として、このようなモデルを利用する。(講義ノート 第6講 抽象機械 P1参考)

↑

形式言語の種類 [↑]

言語(Language)
言語はアルファベット上の**記号列の集合**として定義されている。この記号列の選び方により、言語の類別が可能である。
(講義ノート 第5講 正規言語と正規表現 P1より引用)

文法(Grammar)
文法 G は4つの要素からなる組 G = (N,Σ,P,S) として定義される。

- N:**非終端記号**(nonterminal symbol)**の有限集合**。
- Σ:**終端記号**(terminal symbol)**の有限集合**。(N∩Σ=φ)
- P:**生成規則**(production rule)。
  {λ→ρ|λ∈(N∪Σ)+, ρ∈(N∪Σ)*}
- S:**開始記号**(start symbol)。(S∈N)

文法 G が与える言語 L(G) は次のように定義される。
$L(G) = \{\sigma \mid S \Rightarrow^* \sigma \wedge \sigma \in \Sigma^*\}$

正規言語(Regular Language, RL)
生成規則にλ∈Nかつρ∈Σ*N∪Σ*という制限を加えた場合の文法を正規文法(Regular Grammar, RG)といい、定義される言語を正規言語(Regular Language, RL)という。

正規言語は取扱うことが最も容易な言語の種類であり、言語に関する様々な性質を決定的に判断することができる。正規言語と正規表現が定義する言語のクラスは同じ言語のクラスである。(講義ノート 第5講 正規言語と正規表現 P2,P7参考)

正規表現(Regular Expression, RE)
正規表現とは、テキストパターンに適合(マッチ)する記号列の集合である。
正規表現はアルファベット上の記号列 Σ∪{(,),|,*} で定義されている。
連結演算子は暗黙に与えられている。優先順位:* ≫ 連結 ≫ |

- 各演算子の意味
  - * は0回以上の繰り返し

- (と) はグループ化
- | は選択

文脈自由言語(Context Free Language, CFL)
生成規則にλ∈Nという制限を加えた場合の文法を文脈自由文法(Context Free Grammar, CFG)といい、定義される言語を文脈自由言語(Context Free Language, CFL)という。

文脈依存言語(Context Sensitive Language, CSL)
生成規則に|λ|≦|ρ|という制限を加えた場合の文法を文脈依存文法(Context Sensitive Grammar, CSG)という。CSGが定義する言語は空列Λを含まない。CSGに空列を追加した言語のクラスを文脈依存言語(Context Sensitive Language, CSL)という。L(G_CSG)∪{Λ}

帰納可算集合(Recursively Enumerable Set)
生成規則に制限のない場合の言語のことを帰納可算言語(Recursively Enumerable Language)という。この言語はチューリング機械が定義する言語のクラスと同じである。

アルファベット{a,b,c}上の典型的な言語の例
RL - {a^n|n≧0}={Λ,a,aa,...}
CFL - {a^nb^n|n≧0}={Λ,ab,aabb,...}
CSL - {a^nb^nc^n|n≧0}={Λ,abc,aabbcc,...}

↑

オートマトンの種類 [t]

抽象機械
抽象機械の動作とは、制御部の現在の状態とヘッドが参照するテープ上の記号を入力として、次状態やテープ操作などの出力を決定することである。機械は、一定の状態になるか、次状態が定義されていないため動作が継続できなくなると停止する。(講義ノート 第6講 抽象機械 P3,P4参考)

機械 M の動作状態 C
機械 M の動作状態の集合 C(M)
⊢_M 機械動作状態集合間で定義される関係演算
⊢_M^* 機械 M の計算過程(computation, trace)を示す.

抽象機械 M = (Q,Σ,δ,q_0,F)

- Q:状態集合
- Σ:終端記号集合
- δ:状態遷移関数(関係)
- q_0:初期状態(initial state, start state) q_0∈Q
- F:終了状態集合(final state set, accepting state set) F⊆Q

テープ構成(tape configuration)
テープ上の記号列はテープアルファベット Σ_T = Σ∪{<,>} で表現する。
Σはアルファベット、<,>はテープマーカ(テープの両端を示すための特別な記号)
テープ構成は [p,x] で表現する。現在のテープ記号は σ(p,x) で表現する。

- p:ヘッドの位置
- x:テープ上の記号列
- [1, x]:初期テープ構成
- [|x|+1, x]:最終テープ構成

決定性と非決定性
与えられた情報から機械の動作がユニークに決められる場合を決定的(deterministic)といい、そうでない場合を非決定的(nondeterministic)という。非決定的な機械の実装は、決定的な機械より複雑である。(講義ノート 第6講 抽象機械 P4,P5より引用)

有限状態オートマトン(Finite State Automaton, FSA)
決定的なFSAでは、ある引数について、次状態は高々1個までである。すなわち、現状態と入力記号を決めれば、次状態が1通りだけ決定するか、次状態が定義されないかのどちらかである。(講義ノート 第7講 決定的有限状態機械 P2より引用)
記号列が空列の場合はテープは進まない。この場合を特に空遷移(Λ-transition)という。 NFAでは動作状態の列が枝分かれして、いくつかの列が並列することになる。(講義ノート 第9講 非決定的有限状態機械 P2,P3より引用)

決定的有限状態機械(Deterministic Finite state Automaton, DFA)
非決定的有限状態機械(Nondeterministic Finite state Automaton, NFA)

DFA M = (Q,Σ,δ,q_0,F)
NFA M = (Q,Σ,Δ,q_0,F)
$\mathcal{L}_{DFA} = \mathcal{L}_{NFA}$

状態遷移関数(state transition function) δ:(q,σ)→q'
引数は現状態と入力記号で、戻り値は次状態
δ={(q,σ,q')|q,q'∈Q,σ∈Σ}
状態遷移関係(state transition relation) Δ
Δ={(q,σ,q')|q,q'∈Q,σ∈Σ*}
q**は現状態、**q'**は次状態、**σ**は入力記号列**
σ**には空列**Λ**が含まれることに注意。

機械 M **の動作状態** C = (q,[p,x])
q**は現状態、**[p,x]**はテープ構成**
$(q_0,[1,x]) \vdash_M^* (f,[|x|+1,x])$
f∈F **なら** M **は** x **を受理**(accept)
そうでない M **は** x **を拒否**(reject)

講義ノート **第**9**講 非決定的有限状態機械** P2**より引用**

	状態遷移	入力	次状態
DFA	関数	現状態と1記号	高々1個の状態
NFA	関係	現状態と記号列	複数の状態可

NFAからDFA

NFA→DFA**変換** 【講義ノート **第**10**講** DFAとNFA **を参照のこと】**

- **当初の** NFA **を** M = (Q, Σ, Δ, q_0, F)
  a. **記号列遷移の**1**記号遷移化**
     M_1 = (Q_1, Σ, Δ_1, q_0, F_1)
  b. **空遷移の除去**
     M_2 = (Q_2, Σ, Δ_2, q_0, F_2)
  c. **部分集合構成法**(subset construction)**により**DFA**の構成**
     ある入力記号に対して、現状態から複数の遷移先を統合して一つの状態と見なす。
     M_3 = (Q_3, Σ, δ_3, {q_0}, F_3)
     Q_3 **は集合を要素とする集合で、**Q_2**のべき集合の部分集合である。**
     M_3 **の初期状態は** q_0 **のみからなる集合** {q_0} **である。**
     F_3 = {Q' | Q' ∈ Q_3 ∧ Q' ∩ F_2 ≠ φ(**空集合**)}
     δ_3 **は組織的に構成するアルゴリズムと順次的に構成するアルゴリズムがある。**

プッシュダウンオートマトン(Pushdown Automaton, PDA)

作業テープはスタック(push down stack)**として使われる。**

決定的プッシュダウン機械(Deterministic Pushdown Automaton, DPDA)
非決定的プッシュダウン機械(Nondeterministic Pushdown Automaton, NPDA)
DPDA M = (Q,Σ,δ,q_0,F)
NPDA M = (Q,Σ,Δ,q_0,F)
$\mathcal{L}_{DPDA} \subset \mathcal{L}_{NPDA}$

状態遷移関数δ:(q,t,s)→(q',i,d)
引数は現状態と入力テープ上の記号と作業テープ上の記号で、
戻り値は次状態と入力テープ操作と作業テープ操作
状態遷移関係Δ
Δ={(q,t,s,q',i,d)|q,q'∈Q, s,t∈Σ_T, i∈{o,+}, d∈{o,pop}∪{push a|a∈Σ}}
$\Delta \subseteq Q \times \Sigma_T \times \Sigma_T \times Q \times \{o,+\} \times (\{o,pop\} \cup \{push\ a \mid a \in \Sigma\})$
q**は現状態、**q'**は次状態、**t**は入力テープ上の現在の記号、**s**は作業テープ上の現在の記号**
i**は入力テープ操作集合**{o,+}**の要素、**d**は作業テープ操作集合**{o,pop}∪{push a|a∈Σ}**の要素**

入力記号列x∈Σ*****についての動作状態** C = (q,[p,x],[1,y])
q**は現状態、**[p,x]**は入力テープ構成、**[1,y]**は作業テープ構成(ヘッドは常にテープの先頭)**
$(q_0,[1,x],[1,\Lambda]) \vdash_M^* (f,[|x|+1,x],[1,\Lambda])$
f∈F **なら** M **は** x **を受理**(accept)
そうでない M **は** x **を拒否**(reject)

チューリング機械(Turing Machine, TM)
作業テープに制限はない。

決定的チューリング機械(Deterministic Turing Machine, DTM)
非決定的チューリング機械(Nondeterministic Turing Machine, NTM)
DTM M = (Q,Σ,δ,q_0,F)
NTM M = (Q,Σ,Δ,q_0,F)
$\mathcal{L}_{DTM} = \mathcal{L}_{NTM}$

状態遷移関数δ:(q,a,b)→(q',d_1,c,d_2)
引数は現状態と入力テープ上の記号と作業テープ上の記号で、
戻り値は次状態と入力テープ操作とテープ書き込み記号と作業テープ操作
状態遷移関係Δ
Δ={(q,a,b,q',d_1,c,d_2)|q,q'∈Q, a,b,c∈Σ_T, d_1,d_2∈{+1,0,-1}}
$\Delta \subseteq Q \times \Sigma_T \times \Sigma_T \times Q \times \{+1,0,-1\} \times \Sigma_T \times \{+1,0,-1\}$
qは現状態、q'は次状態、aは入力テープ上の現在の記号、bは作業テープ上の現在の記号
cは作業テープに書き込まれる記号でbを書き換える、d_1,d_2はテープ操作集合{+1,0,-1}の要素

入力記号列x∈Σ*についての動作状態 C = (q,[p,x],[w,y])
qは現状態、[p,x]は入力テープ構成、[w,y]は作業テープ構成
$(q_0,[1,x],[1,\Lambda]) \vdash_M^* (f,[h,x],[k,s]) = C_f$
C_fは停止動作状態で
$f \in F$ の場合 M は x を受理(accept)
$f \notin F$の場合 M は x を拒否(reject)

各オートマトンの構成

種類	構成
FSA	制御部＋入力テープ
PDA	制御部＋入力テープ＋作業テープ(スタック)
TM	制御部＋入力テープ＋作業テープ(無制限)

↑

形式言語とオートマトンの関係

言語のクラス間には以下のような関係が成り立つ。

$\mathcal{L}_{TM} \supset \mathcal{L}_{CSL} \supset \mathcal{L}_{CFL} \supset \mathcal{L}_{RL}$

チョムスキー階層についてもっと詳しく知りたい場合は Chomsky hierarchy - Wikipedia が便利です。

チョムスキー階層

型	文法	言語	オートマトン
Type-0	-	帰納的可算言語	TM(Turing Machine)
Type-1	文脈依存文法	文脈依存言語	LBA(Linear Bounded Automaton)
Type-2	文脈自由文法	文脈自由言語	PDA(Pushdown Automaton)
Type-3	正規文法	正規言語	FSA(Finite State Automaton)

↑

過去問解答案

基本は自己責任。間違いを探す気で
Λ(ラムダ)と∧(または)の区別が付きにくいので mimeTeX[*2] を利用してみました。

空集合と空列についての理解は必須です。必ず覚えておきましょう。
要素の数が0である集合$|\emptyset| = 0$を空集合 (empty set) といい、$\emptyset, \{\}$などで表す。
列の長さが0である記号列$|x| = 0$を空列 (empty string) といい、Λ, ϵなどで表す。
また、要素がない言語を空言語 (empty language) といい、空集合と同じ記号で表す。
注意:空列のみからなる言語は、1つの要素を含むので空集合ではない。$\{\Lambda\} \neq \emptyset$
※同じ記号||でも意味が若干異なります。注意してください。

以下に基本的なことを書いておきます。いろいろ省略しているので、必要なものが抜け落ちてるかも…

ある記号列xをn回連結した記号列をx^nと表す。$x^0=\Lambda$
記号列x中の記号aの個数を$\#_a(x)$と表す。例:$\#_a(aaabab)=4$
記号列xの逆列をx^Rと表す。例:$(x^R)^R=x,\ (xy)^R=y^Rx^R$
$\circ$は連結演算子を意味する。例:$aaab\circ ab=aaabab,\ x\circ y\neq y\circ x$
空列を扱った単純な例:$\Lambda^R=\Lambda,\ \Lambda\circ x=x\circ\Lambda=x$

Σ上の記号列の有限あるいは無限集合をΣ上の言語という。
アルファベットΣ上の言語をL_1,L_2とする。またLを任意の言語とする。
$L_1=\{ab,abb\},L_2=\{bb,bbb\}$ → $L_1\circ L_2=\{abbb,abbbb,abbbbb\}$
$L\circ\emptyset=\emptyset\circ L=\emptyset,\ L\circ\{\Lambda\}=\{\Lambda\}\circ L=L,\ \{\}^*=\emptyset^*=\{\Lambda\}$
逆列言語 (reversal) L^Rの定義は$L^R=\{x^R|x\in L\}$で表される。

あるアルファベットΣ上の正規表現Eが定義する言語を$L(E)$とする。
$E=()$ → $L(E)=\emptyset,\ E=()^*$ → $L(E)=\emptyset^*=\{\Lambda\}$
$E=(E_1|E_2)$ → $L(E)=L(E_1)\cup L(E_2)$
$E=(E_1E_2)$ → $L(E)=L(E_1)\circ L(E_2)$
注意:正規表現$(a|b)^*$の正規言語は$(\{a\}\cup\{b\})^*$または$\{a,b\}^*$である。

連結操作(concatenation operation)
記号列x^nの連結:

$$x^n=\begin{cases}x^0 &=\Lambda\\ x^{i+1}&=x\circ x^i\ \ (i\geq 0)\end{cases}$$

言語L^nの連結:

$$L^n=\begin{cases}L^0 &=\{\Lambda\}\\ L^{i+1}&=L\circ L^i\ \ (i\geq 0)\end{cases}$$

star-閉包(star-closure, Kleene star):

$$L^*=L^0\cup L^1\cup L^2\cup\cdots=\bigcup_{i\geq 0}L^i$$

テストでは、文法 G の非終端記号集合と終端記号集合の並びが逆になっている。
白色で書かれている解答は曖昧なものである。自分で考えるなり教授に聞くなりしてねっと。

↑

2002年後期中間 2002.11.26

問題1

1 2 **2** 8 **3** 7 **4** 3 **5** 1

問題2

6	2	**7**	2	**8**	2	**9**	2	**10**	2	**11**	2	**12**	1
13	1	**14**	1	**15**	1	**16**	1	**17**	2	**18**	2	**19**	1
20	2	**21**	2	**22**	1	**23**	1	**24**	2	**25**	2	**26**	2
27	2	**28**	2	**29**	1	**30**	1	**31**	2	**32**	1	**33**	2

問題3

34 2 **35** 1 **36** 2 **37** 2 **38** 2

解説

i. 1〜5
言語L1とL2がどのような記号列の集合なのかがわかれば解けます。
$L_1=\{0\}^*,\ L_2=\{1\}^*$
$L_1=\{0^i|i\geq 0\},\ L_2=\{1^j|j\geq 0\}$
$L_1=\{\Lambda,0,00,000,\ldots\},\ L_2=\{\Lambda,1,11,111,\ldots\}$
$L_1\circ L_2=\{\Lambda,0,1,00,11,01,000,111,001,011,\ldots\}$
$L_2\circ L_1=\{\Lambda,0,1,00,11,10,000,111,110,100,\ldots\}$

なぜ上記のような要素があるのかについては、連結操作を参照してください。

ii. 6〜33
$\{xx^R | x \in \{0,1\}^*\}$ に注意して解きましょう。
ある記号列xを01と決めたとき、記号列 x^R は逆列なので10となります。
ちなみに xx^R のような形式の記号列を回文(palindrome)といいます。

iii. 34〜38
34 $\{\}^* = \emptyset^* = \{\Lambda\}$ $\{\} = \emptyset$
35 $(0^*1^*)^* = \{\Lambda,0,1,00,11,01,10,\cdots\}$ $\{0,1\}^* = \{\Lambda,0,1,00,11,01,10,\cdots\}$
36 $0^* \cap 1^* = \{\Lambda\}$ $0^* = \{\Lambda,0,00,\cdots\}$ $1^* = \{\Lambda,1,11,\cdots\}$
37 $0^* \cup 1^* = \{\Lambda,0,1,00,11,000,111,\cdots\}$
38 $0^*(0^* \cup 1^*) = 0^*\{0,1\}^*$

↑

2002年後期期末 2003.01.14

問題1

1 1 **2** 2 **3** 4 **4** 2 **5** 4 **6** 2 **7** 3

問題2

8 1 **9** 1 **10** 4
11 9 **12** 9 **13** 9
14 9 **15** 9 **16** 9
17 1 **18** 1 **19** 7
20 9 **21** 9 **22** 9
23 9 **24** 9 **25** 9
26 1 **27** 1 **28** 7
29 1 **30** 7

問題3

31 1 **32** 2 **33** 8 **34** 5

解説

i. 1〜7
問題1は、各オートマトンの特徴について理解していれば解けるはずです。
ちなみに TM で計算可能なものはコンピュータで計算可能なものです。
以下に参考になりそうなサイトなどを挙げて置きます。
Wikipedia「Church-Turing thesis」「Halting problem」「Traveling Salesman Problem」
あとは「ポストの対応問題」「決定不能」「万能チューリング」について調べてください。

1 FSAで受理可能。正規表現は 0* となる。

2 DPDAで受理可能。
1).入力記号列の前半で0が連続する間、スタックに0を格納する。
2).入力に1が出現すると、1が1個出現する毎に0を1個スタックから取除く。
この最終状態(final configuration)を考慮することで、記号の個数を数えられる。

3 TMで受理可能。詳しくは講義ノートでも見てね。

4 DPDAで受理可能。
0と2の個数が一致すればこの言語を受理することができる。
1).0が入力される間は、0をスタックにプッシュ
2).1が入力される間は、スタックの操作はしない
3).2が入力される間は、0をスタックからポップ

5 TMで受理可能。プログラムではキューで実現可能。

6 DPDAで受理可能。
折り返し位置を決定的に求めることができるため、DPDAとなる。
1).0または1が出現する度に、その記号をスタックにプッシュする。
2).入力記号列に2が出現すれば、折り返しと判断し状態を移行する。
3).0または1が出現する度に、スタックからポップした記号を比較。

7 NPDAで受理可能。
どちらか一方のみならDPDAで受理可能だが、和集合により遷移先が決定的ではなくなっている。

ii. 8〜30
問題文に『初期状態から到達できない状態については、その全ての解答欄に次状態として(9)をマークせよ』とある。
これに注意して問題を解こう。ちなみに上記を考慮しない場合は以下のようになる。これは正規表現 (a|b|c)*cc を受理する。

state	a	b	c
{0}	{0}	{0}	{0,1}
{1}	{ }	{ }	{2}
{2}	{ }	{ }	{ }
{0,1}	{0}	{0}	{0,1,2}
{0,2}	{0}	{0}	{0,1}
{1,2}	{ }	{ }	{2}
{0,1,2}	{0}	{0}	{0,1,2}

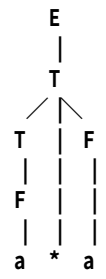
この問題は、『順次的に構成するアルゴリズム』を用いて説明します。NFAからDFAを参照のこと。
「遷移(q_0,a_i,q')がΔ_2に含まれていれば、遷移({q_0},a_i,Q')の遷移先Q'にq'を追加する。」

初期状態0のみからなる集合{0}が既に書かれている遷移表を初期値として、δを構成します。
遷移(0,a,0)がΔに含まれているので、遷移({0},a,{0})と0を追加する。
遷移(0,b,0)がΔに含まれているので、遷移({0},b,{0})と0を追加する。
遷移(0,c,0)がΔに含まれているので、遷移({0},c,{0})と0を追加する。
遷移(0,c,1)がΔに含まれているので、遷移({0},c,{0,1})と1を追加する。
新規の状態の集合{0,1}が追加されたので、次にその状態遷移について調べます。
{0,1}は、その集合の要素である0と1の各状態について調べればよい。
aによる遷移は(0,a,0)のみなので、遷移({0,1},a,{0})を表に追加する。
bによる遷移は(0,b,0)のみなので、遷移({0,1},b,{0})を表に追加する。
cによる遷移は(0,c,0)(0,c,1)(1,c,2)なので、遷移({0,1},c,{0,1,2})を表に追加する。
新規の状態の集合{0,1,2}が追加されたので、同様にその状態遷移について調べると
({0,1,2},a,{0}) ({0,1,2},b,{0}) ({0,1,2},c,{0,1,2}) で、新たな状態はないため終了。

初期状態から到達可能な状態というのは、{0}からその状態に遷移できる、ということです。
以上より初期状態{0}から到達可能な状態は{0},{0,1},{0,1,2}ということがわかります。
また、終了状態は状態2を含む集合から構成されるので、表から{0,1,2}となります。

iii. 31〜34
最左導出ということに注意して解くと、E⇒T⇒T*F⇒F*F⇒a*F⇒a*aとなる。
ちなみに最右導出の場合は、E⇒T⇒T*F⇒T*a⇒F*a⇒a*aとなる。

解析木(parse tree)



↑

2003年後期期末 2004.01.27 [t]

問題1

1 1 **2** 1 **3** 4 **4** 1

5 8 **6** 8 **7** 3 **8** 9

9 8 **10** 8 **11** 8 **12** 9

13 1 **14** 1 **15** 7 **16** 1

17 1 **18** 1 **19** 4 **20** 9

21 8 **22** 8 **23** 3 **24** 9

25 1 **26** 1 **27** 7 **28** 1

29 8 **30** 8 **31** 8 **32** 9

33 1 **34** 7

問題2

35 4 **36** 2 **37** 2 **38** 2 **39** 1 **40** 2 **41** 3 **42** 4

問題3

43 1 **44** 9 **45** 5 **46** 3

解説

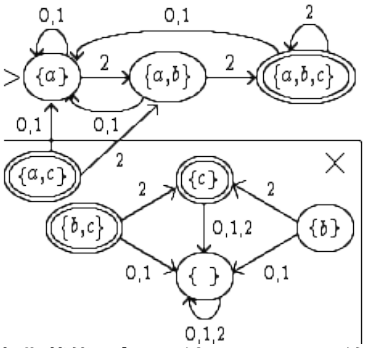
i. 1～34

NFAからDFAを参照のこと。部分集合構成法はサブセット構成法とも呼ばれているようです。
与えられたNFAの状態遷移図を描いてみましょう。そうすることで理解しやすくなるはずです。
状態遷移図については「講義ノート 第7講 決定的有限状態機械 P5,6」を参考にしてください。
ちなみにアルゴリズムに従って文章を書くと長くなりそうなのでしていません。と一応注意書き。

1).与えられた状態遷移関係Δから、{a},{b},{c}の各状態の集合の遷移先を求めます。
(a,0,b)を状態aで入力記号が0のときの遷移先がbだと解釈すればよくわかるはずです。
(実際にはaは現状態、0は入力記号列、bは次状態となります。既に1記号遷移化済みなの)
{a}に関わりのあるものをΔから抜き出すと(a,0,a),(a,1,a),(a,2,a),(a,2,b)が該当します。
{b}に関わりのあるものをΔから抜き出すと(b,2,c)が該当します。
{c}に関わりのあるものは無いので無視します。(空集合で埋める)
それらの状態の集合から、0,1,2の各記号を入力したときの遷移先を求めます。
もし、同じ記号で複数の遷移先がある場合はそれらを統合し1つの状態と見なします。

2).求めた各状態の集合{a},{b},{c}の状態遷移表から、他の状態の集合の遷移先を求めます。
例えば状態の集合{a,b,c}は状態aと状態bと状態cを1つにまとめたものなので、
{a}∪{b}∪{c}と見なすことができ、既に埋まっている表から直ぐに求まります。
この操作を各状態の集合に当てはめることで、全ての状態について遷移先が求まります。

初期状態{a}から巡って移行できるのは{a},{a,b},{a,b,c}です。
終了状態は「初期状態から到達できない状態」を削除した図で考える？
講義ノート 第10講 DFAとNFA P7「終了状態は…」を参考にすると、削除する？



初期状態は「>」の付いているもの。終了状態は二重丸◎のもの。
※書いたノートには空集合{}への遷移は描かれていませんが、ここでは描いています。
※講義ノート 第10講 DFAとNFA『組織的に構成するアルゴリズム』P5,6 を参考

構成された DFA $M=(Q,\Sigma,\delta,q_0,F)$ を参考に書きます。

※初期状態から到達できない状態は削除しています。
$Q=\{\{a\},\{a,b\},\{a,b,c\}\}\ \Sigma=\{0,1,2\}\ q_0=\{a\}\ F=\{\{a,b,c\}\}$
$\delta=\{(\{a\},0,\{a\}),(\{a\},1,\{a\}),(\{a\},2,\{a,b\}),$
$(\{a,b\},0,\{a\}),(\{a,b\},1,\{a\}),(\{a,b\},2,\{a,b,c\}),$
$(\{a,b,c\},0,\{a\}),(\{a,b,c\},1,\{a\}),(\{a,b,c\},2,\{a,b,c\})\}$
δは状態遷移関数です。関係じゃないです。
あと、要素なのか集合なのか、その辺をしっかり理解！

ii. 35〜42
37 DPDA. (1) 0 -> no operation. (2) 1 -> push. (3) 2 -> pop.
38 DPDA. (1) 0 -> push. (2) 1 -> pop. (3) 2 -> no operation.
39 FSA. Regular Expression: 0*1*2*
40 and 41 @see <講義ノート 第15講 DPDAとNPDA P7>
42 TM. use Queue.

iii. 43〜46
導出(derive) 講義ノート 第17講 文法 P2を参考にしてください。
導出の繰返しを導出列(derivation)という。文(sentence)になるまで行う。
終端記号と非終端記号の両方からなる記号列を文形式(sentential form)という。
『遷移δ(p,a)=p'』は『規則p→ap'』と見なすことができる。明確な共通点！
「曖昧さ」についても考えておくと幸せに…？
43 $\{\Lambda,ab,aabb,aaabbb,\cdots\}$
$S\Rightarrow\Lambda, S\Rightarrow aSb\Rightarrow ab, S\Rightarrow aSb\Rightarrow aaSbb\Rightarrow aabb$
44 $\{\Lambda,ab,ba,abab,baba,abba,baab,\cdots\}$
aabb,bbaa -> reject!
45 $\{\Lambda,a,b,aa,bb,ab,aaa,bbb,aab,abb,\cdots\}$
46 $\{\Lambda,aa,bb,aaaa,bbbb,abba,baab,\cdots\}$

↑

2004年後期中間 2005.01.08 [t]

問題1

1 2 **2** 7 **3** 6 **4** 3 **5** 6 **6** 6 **7** 1 **8** 3

問題2

9 2 **10** 2 **11** 2 **12** 2 **13** 2 **14** 2 **15** 1
16 1 **17** 1 **18** 1 **19** 1 **20** 2 **21** 2 **22** 1
23 2 **24** 2 **25** 1 **26** 1 **27** 2 **28** 2 **29** 2
30 2 **31** 2 **32** 1 **33** 1 **34** 2 **35** 1 **36** 2

問題3

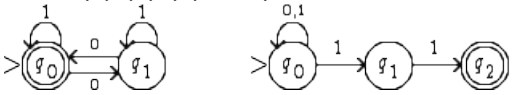
37 9 **38** 3 **39** 1 **40** 8

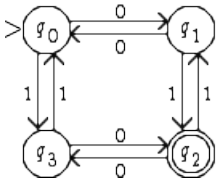
解説

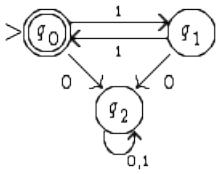
i. 1〜8
2002年後期中間 2002.11.26 問題1の解説を参照してください。

ii. 9〜36
2002年後期中間 2002.11.26 問題2の解説を参照してください。

iii. 37〜40
アルファベットΣ={0,1}上の記号列
(1)の正規表現は 1*(01*01*)* で、これは記号列に含まれる0の個数が偶数個のものを受理する。
(2)の正規表現は (0|1)*11 で、これは記号列が11で終わるものを受理する。
(3)の正規表現は (0(00|11)*1|1(00|11)*0)(00|11|(01|10)(00|11)*(01|10))* で、
これは記号列に含まれる0と1の個数がそれぞれ奇数個のものを受理する。1+2n
(4)の正規表現は (11)* で、これは1のみからなる記号列でなおかつ偶数個のものを受理する。
40 -> (1),(2),(4) accept!







↑

2004年後期期末 2005.02.01 [t]

問題1

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

9 1 **10** 1 **11** ?

12 1 **13** 3 **14** 9

15 3 **16** 5 **17** 1

18 1 **19** 1 **20** 9

21 3 **22** 5 **23** 1

24 1 **25** 3 **26** 9

27 3 **28** 5 **29** 9

30 3 **31** 5 **32** 9

33 2 **34** 5

問題2

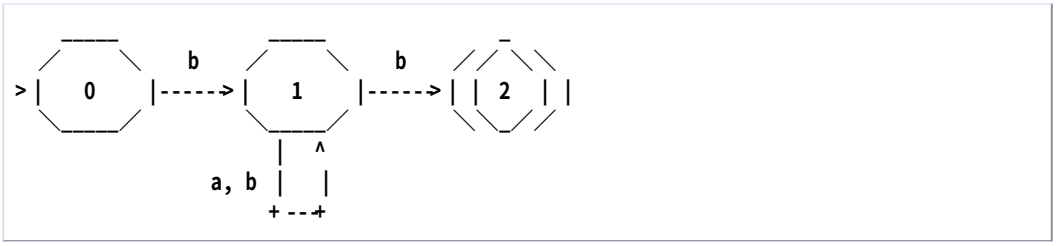
35 9 **36** 9 **37** 9 **38** 9 **39** 9 **40** 1 **41** 9 **42** 1

問題3

43 9 **44** 1 **45** 1 **46** 1 **47** 3 **48** 1 **49** 2 **50** 9

解説

i. 1〜8
NFAとあるので、正規表現で表すことができる。
この正規表現は b(a|b)*b で、先頭と末尾の記号がbである記号列を受理する。
NFAの状態遷移図



9〜34
空集合{}を1つの状態としてみなすのかについてはわかりません。
以下に参考になりそうな部分を挙げて置きます。完全化という言葉が鍵？
講義ノート【第10講 DFAとNFA P5の一番下,P9の考察】【第11講 FSAが定義する言語 P3の捕集合】【第12講 FSAと正規言語 P2の部分】

NFA→DFA変換については、ここを参考にしてください。
Subset construction は「部分集合構成法」や「サブセット構成法」と訳すようです。
Regular Expression を「正則表現」と訳すのか「正規表現」と訳すのかと同じこと。
状態遷移図(state transition diagram)は、ちゃんと描けるようになっておきましょう。
以下、与えられた抽象機械Mと状態遷移関係Δを元に説明します。例によって適当です。

まず、{0},{1},{2}の各状態の集合を調べます。これは与えられたNFAを元に考えます。
状態0でbを入力した場合は遷移(0,b,0)より次状態は0となります。a,cの場合、遷移先は未定義。
状態1でa,cを入力した場合、それぞれ遷移(1,a,1),(1,c,1)が該当し、次状態は1となります。
また、状態1でbを入力すると(1,b,1)(1,b,2)が該当します。そこで部分集合構成法にあるように、
出てきた複数の次状態(遷移先)をまとめて1つの状態と見なす、という作業をします。{1,2}
状態2では遷移関係Δに該当するものはなく、遷移先は未定義となります。(遷移先を空集合とする)
ここで新たに構成するDFAの状態集合Qは『集合を要素とする集合』なので{0},{1},{2}にします。

次に、状態0と状態2で出てきた遷移先が未定義の場合の遷移先の空集合{}について考えます。
{}は抜け出すことができない状態なのでトラップ状態(trap state)といい、さらにこの場合は
決して受理されることはない状態となるので死状態(dead state)というようです。一部海外で。
そういうことなので、空集合は入力記号が何であっても次状態(遷移先)は自分自身となります。

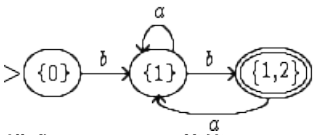
さらに、求めた{0},{1},{2}から{0,1},{1,2},{2,0},{0,1,2}の各状態の集合を求めます。
{0,1}というのは状態0と状態1を要素とする集合なので、{0}と{1}の和集合であると考えられます。
この考えは他の状態の集合についても同じように適用できます。(L_1∪L_2みたいなものです。)

これで、各状態の集合の遷移先がそれぞれ求まりました。次に初期状態について考えます。
元のNFAでは初期状態は0(要素)ですが、構成されたDFAでは{0}(集合)となります。
また、状態0が含まれるからといって{0,1}が初期状態とはなりません。以下その説明です。
NFAの初期状態0から**空遷移**で到達できる状態からなる**集合**が構成されたDFAの初期状態です。
この場合は空遷移で到達できるものはないので、初期状態は「状態0のみからなる集合」となります。
(状態0のε-閉包[*3]というらしい。「ε-NFA」を空列を含むNFAとしているところもある。)

さて、初期状態から到達可能な状態についてですが、これは状態遷移図を元に考えてみます。
オートマトンは初期状態から開始するので、「>」の付いている{0}からスタートします。
そこから出ている線に沿って終了状態を目指す感じです。つまり、初期状態から出ている線から
到達できるものは{0},{1},{1,2}が当てはまります。空集合についてはわかりません…。

最後は終了状態についてです。これはNFAの終了状態集合{2}の要素を含む集合となります。
この際、暗黙の了解で**初期状態から到達できない状態を削除**しているのかは知りません。
ちなみに初期状態から到達できない状態を削除することは「DFAの最小化」に関係しています。

構成されたDFAの状態遷移図(最小の状態数)



構成されたDFAで状態を{~~おっぱい括弧~~}でくくっていないような状態遷移図を描かないよーに！
部分集合構成法と書いてある意味を全く理解していないことになりますので、気をつけてね。
要素の名前を適切に付け直している場合は問題ないけどね。っと、付けたし。

ii. 35～42
正規文法とあるので、正規表現で表すことができる。
この正規表現は b(a|b)*b で、先頭と末尾の記号がbである記号列を受理する。
このことから、問題1と問題2は関連性があるといえます。なかなか…
iii. 43～50
過去問で既に解説済み。

↑

2006年後期期末 2007.01.29 [†]

問題1

		1	1	**2**	1	**3**	1
4	2			**5**	1	**6**	1
7	2	**8**	1			**9**	1
10	2	**11**	1	**12**	1		
13	5	**14**	5	**15**	5	**16**	5

問題2

17 1 **18** 1

19 7 **20** 2

21 4 **22** 1

23 1 **24** 1

25 4 **26** 1

27 7 **28** 2

29 8 **30** 2

31 8 **32** 2

33 2

問題3

34 1 **35** 5 **36** 3 **37** 6 **38** 2

問題4

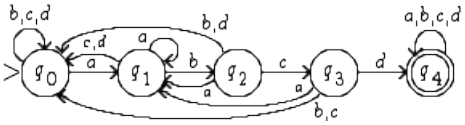
39 2 **40** 6 **41** 3 **42** 3 **43** 2 **44** 1 **45** 3

問題5

46 ?

解説

i. 1〜16
講義ノート 第8講 有限状態機械の設計 によく似た例があります。
その部分を参考にすれば解けるはずなので、解説は省略します。



ii. 17〜33
過去問に類似問題があるので、解説は省略します。

iii. 34〜38
過去問に類似問題があるので、参考にしてください。
問題文に最も適切な言語についての説明があります。

iv. 39〜45
過去問やノートなどを参考にしてください。

v. 46
このDPDAが受理する言語は、(1)と(2)が該当します。
(1)を選択する際の根拠となるのは、「最も適切な言語」という記述がある場合が考えられる。
そう考えると、ここでは単に「受理する言語」を選べとあるので両方が該当することになる。
しかし、(1)のCFGからオートマトンを構成する場合には、0101という記号列を受理する
必要はないと考えられ、与えられた状態遷移表とは異なったものを作成する。だから、(2)???
(問題3と問題5の文法の部分は(2)のところだけが違っているという…。なかなか…)

---

*1 オートマトンに関連するリンクのこと。オートマトンをオートマンと間違える人がいるためこの名が付けられた。
*2 mimeTeXとmimetex.inc.phpを参照のこと。LaTeX形式で書けるようです。勉強がてら無駄に使用。
*3 NFAからDFAの「空遷移の除去」の部分を参考にしてください。ε、Λは空列を表す記号です。

RitsWiki は、いかなる大学・組織とも無関係に運用されており、記載されている情報の信憑性および内容は保証されません。各自の責任の下でご利用下さい。
RitsWiki is UNOFFICIAL. Any information provided on this site is not guaranteed in any way. USE AT YOUR OWN RISK.

PukiWiki 1.4.6 Copyright © 2001-2005 PukiWiki Developers Team. License is GPL.
Based on "PukiWiki" 1.3 by yu-ji. Powered by PHP 4.4.7

HTML convert time to 0.050 sec.