2010年度 博士前期課程 入試問題 解答例

【試験日時】2009年8月1日(土)　9:00〜12:00

【作成者】樋口 雄大 (t-higuti@ist.osaka-u.ac.jp)

【作成日】2009年8月16日

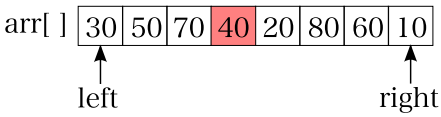
1 アルゴリズムとプログラミング

(1)	クイックソート								
(2)	1 回目	30	10	20	40	70	80	60	50
	3 回目	10	20	30	40	70	80	60	50
(3)		20	40	60	80	10	50	30	70
(4-1)	$2n-1$								
(4-2)	$O(n^2)$								
(4-3)	最悪の状況								

解説：

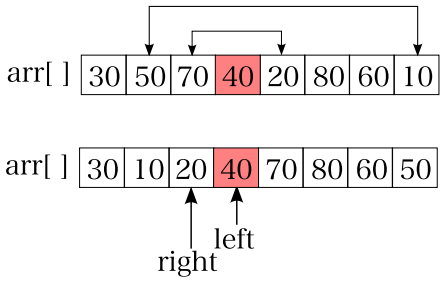
(2)

初期状態において，配列 `arr[]` には，次のような値が格納されている．



`arrange` 関数の 1 回目の実行では，14 行目で，基準値 `bd` として 40 が選択される．まず，17〜19 行目の while ループで，`a[left]` $<$ `bd` が成り立つ間，`left` を左端から順次右へスライドしていく．その結果，最初にこの条件が成り立たなくなる `arr[1]` でループを脱出する．20〜22 行目の while ループでは，`a[right]` $\geq$ `bd` が成り立つ間，`right` を右端から順次左へスライドし，`arr[7]` でループを脱出する．`right` ≥ 0 かつ `left` $\leq$ `right` なので，27 行目の if 条件式が真になり，`arr[1] = 50` と `arr[7] = 10` が交換される．

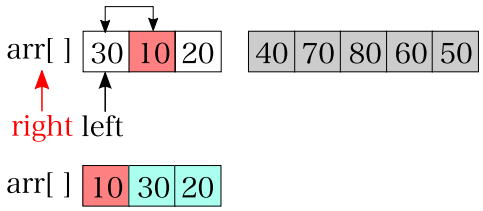
同様に 70 と 20 が交換され，最終的に，`left` $= 3$，`right` $= 2$ で `left` $>$ `right` となり，33 行目の `break` 文が実行される．最後に，37 行目で `left` の値が返り値 (分割の境界点) として返され，`arange` 関数の 1 回目の実行が終了する．



`arrange` 関数の返り値は，45 行目で，変数 `k` に格納される．その後，46 行目で `display` 関数の 1 回目の呼び出しが行われ，47〜48 行目で $[a, k-1]$ ([0,2])，$[k, b]$ ([3, 7]) の部分列に対して，`sort` 関数が再帰的に適用される．

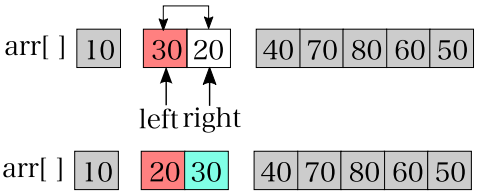
先に実行されるのは，$[0, 2]$ の部分列に対する処理である．入力列の中央にある，`arr[1] =`

10 が分割の基準値として選択される．続いて，left と right を左右にスライドさせる．arr[0] = 30 ≥ 10 が成り立つので，left は left = 0 で止まる．一方，arr[2] ≥ 10，arr[1] ≥ 10，arr[0] ≥ 10 がいずれも成り立つので，right は，入力列の左端を突き抜け，−1 で止まる．したがって，基準値 10 と，入力列の左端値 30 が交換され，26 行目で，返り値として $a+1=1$ が返される．



この後 display 関数の 2 回目の実行が行われ，続いて，部分列 [0, 0]，[1, 2] に対して sort 関数が再帰的に実行される．[0, 0] については，入力列の長さが 0 となるので，43 行目の if 条件式が真となり，sort 関数は，display 関数を実行する事なく即座に終了する．

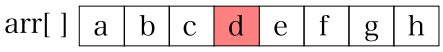
部分列 [2, 3] に対する sort 関数の実行では，基準値として 30 が選ばれ，30 と 20 が交換される．その後，display 関数が実行され，3 回目の出力結果は下図のようになる．



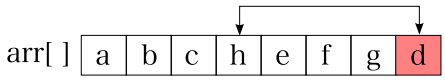
(3)

49 行目の sort 関数の呼び出しにおいて常に k と b が等しくなり，かつ 25 行目の処理が行われないような入力の並びを考える．

25 行目，すなわち，right が入力列の左端を突き抜けた時のエラー処理が行われないことから，k の値は，必ず arrange 関数実行終了時の left の値に等しくなる．left が b，すなわち入力の右端に達するのは，基準値として，常に入力列の最大値が選ばれる場合である．入力列の要素を記号 a 〜 h に置き換えて，このことを確かめてみよう．



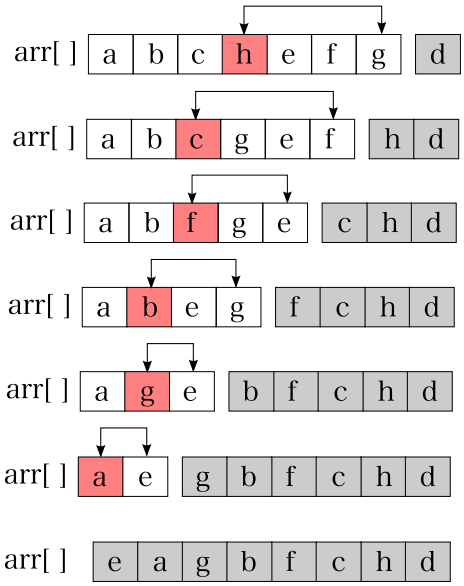
最初に，d が基準値として選ばれる．先ほどおいた仮定より，d は，a 〜 h の中で最大の要素である．要素値に重複がないとすれば，$a < d$，$b < d$，$c < d$ が成り立ち，17〜19 行目のループで，left は d が格納された left = 3 までスライドする．一方，$h < d$ なので，right は入力列の左端から動かない．その結果，29 行目で d と h が交換され，left = 4，right = 6 となる．



次のループでは，$e < d$，$f < d$，$g < d$ が成り立つので，`left` は 右端の 7 まで移動する．一方，`right` は 6 のまま動かない．`left` > `right` となったので，33 行目の `break` 文が実行され，`arange` 関数の実行が終わる．返り値は，確かに入力列の右端の 7 となっている (すなわち，`k` = `b`)．

続いて，部分列 [0, 6] に対して `sort` 関数が呼び出される．基準値としては h が選ばれ，先ほどと同様の議論によって基準値は入力列の右端の要素 g と交換され，`arrange` 関数の実行が終了する (h が，a, b, c, e, f, g, h の中で最大の要素であるという仮定による)．`arange` 関数の返り値は，やはり入力列の右端の 6 となる．

以下同様に，ソーティングの実行の様子を示したのが下図である．基準値として選ばれる要素を赤く塗りつぶしている．

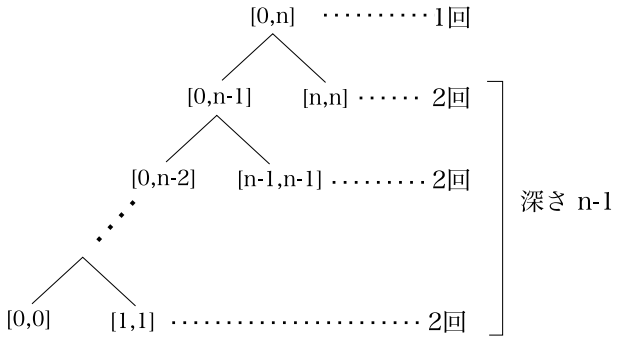


このトレース結果から，基準値として選ばれる順序は，d，h，c，f，b，g，a，e となることが分かる．この結果と，基準値として常に入力列の最大値が選ばれるという条件から，10〜80 を，大きいものから順に，各変数に割り当てていけばよいことになる．すなわち，$d = 80$，$h = 70$，$c = 60$，$f = 50$，$b = 40$, $g = 30$，$a = 20$，$e = 10$．したがって，求める並び替えの結果は，20，40，60，80，10，50，30，70 である．

(4-1)

49 行目の `sort` 関数の呼び出しにおいて `k` と `b` が等しくなるとき，長さ n の入力列は，長さ $(n-1)$ の部分列と，長さ 1 の部分列に分割される．このそれぞれに対して，再帰的に `sort` 関数が適用される．したがって，部分列の長さが 1 短くなるごとに，`sort` の合計呼び出し回数は 2 ずつ増えていく．長さ n から始めて，最終的にはすべての部分列の長さが 1 になるまで `sort`

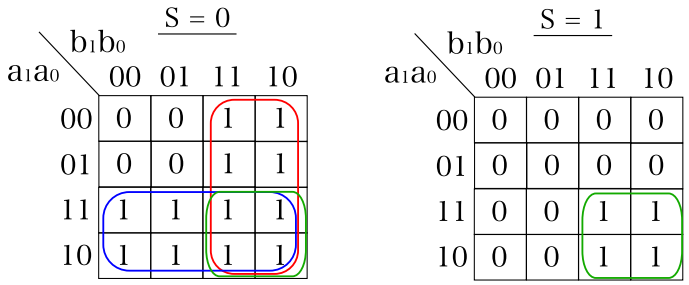
関数の再帰呼び出しが行われること，および，最初の長さ n の入力列に対しては，`sort` 関数の呼び出しは 1 回であることを考慮すると，呼び出しの合計回数は $1 + 2(n-1) = 2n - 1$ 回となる．



2 論理回路

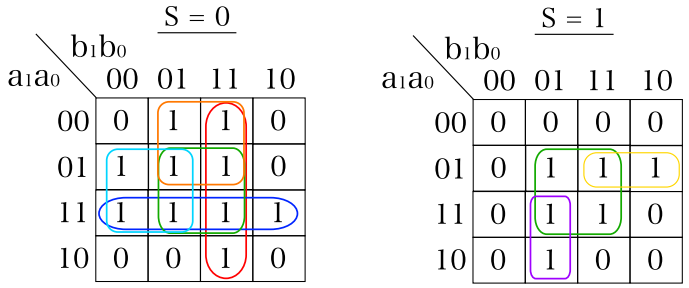
(1-1)

c_1:



$$c_1 = a_1 b_1 + \bar{S} a_1 + \bar{S} a_2$$

c_0:



$$c_0 = a_0 b_0 + \bar{S} a_1 a_0 + \bar{S} b_1 b_0 + \bar{S} a_0 \bar{b_1} + \bar{S} \bar{a_1} b_0 + S \bar{a_1} a_0 b_1 + S a_1 \bar{b_1} b_0$$

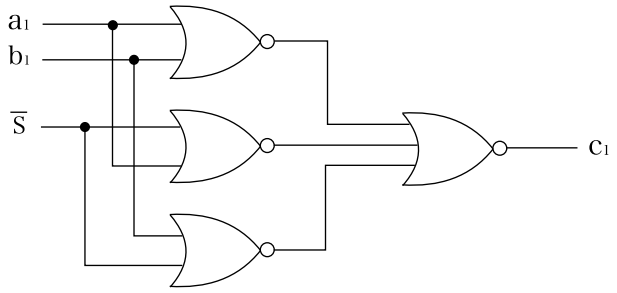
解説:

回路の機能としてはそれほど複雑ではないが，入力信号が S, a_1，a_0，b_1，b_0 の 5 つあるので，少し工夫が必要．5 変数の論理式の最小積和形を求めるには，どれか一つの変数の値によって場合分けし，4 変数のカルノー図を 2 つ作る．解答例に示した 2 つのカルノー図は，左側が $S = 0$ の場合，右側が $S = 1$ の場合を表している．

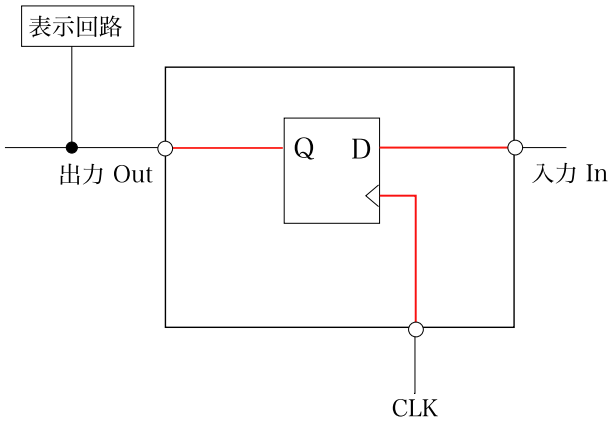
c_1 のカルノー図において緑で囲んだ部分は，$S = 0$ の場合にも，$S = 1$ の場合にも共通に囲まれている．この部分から得られる項は $a_1 b_1$ となり，入力信号 S に依存しない．すなわち，1 つのカルノー図上で，8 個の領域を囲んだのと同じ効果が得られる．直感的には，2 枚のカルノー図を上下に重ね，4 マス $\times$ 4 マス $\times$ 2 マス の 3 次元の表を考えればよい．一方，左側のカルノー図において青や赤で囲んだ部分は，$S = 0$ の時のみ 1 となるため，対応する項は，それぞれ $\bar{S} a_1$，$\bar{S} b_1$ となる．

(1-2)

$$
\begin{aligned}
c_1 &= (a_1 + b_1)(\bar{S} + a_1)(\bar{S} + b_1) \qquad \cdots c_1\text{の最小和積形論理式} \\
&= \overline{\overline{(a_1 + b_1)(\bar{S} + a_1)(\bar{S} + b_1)}} \\
&= \overline{\overline{(a_1 + b_1)} + \overline{(\bar{S} + a_1)} + \overline{(\bar{S} + b_1)}}
\end{aligned}
$$



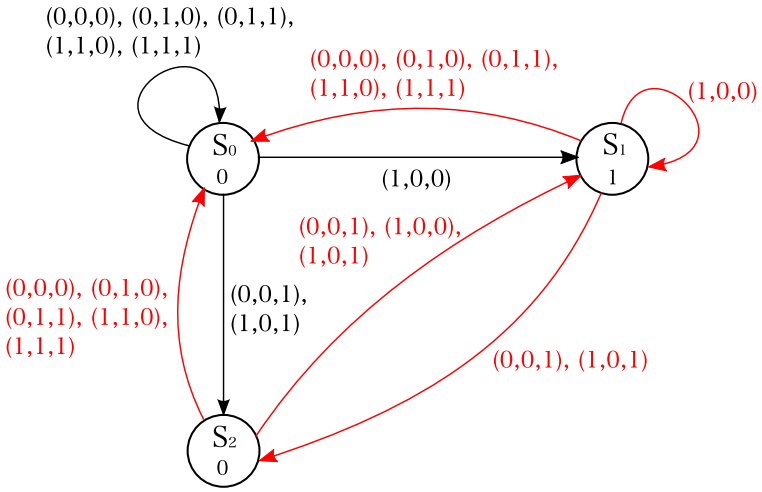
(2-1)



※ 赤線が，解答用紙に新たに書き加える部分

(2-2)

(2-2-1)



※ 赤線・赤文字が，解答用紙に新たに書き加える部分

(2-2-2)

$Out = Q_1\bar{Q_0}$

(2-2-3)

$D_1 = \bar{Q_1}\bar{R}C$

$D_0 = Q_1\bar{R}C + In\bar{R}\bar{C}$

解説：

(2-2-1) の状態遷移表をもとにカルノー図を作成すると，以下のようになる．

D_1:

In = 0

Q_1Q_0 \ RC	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	d	d	d	d
10	0	0	0	0

In = 1

Q_1Q_0 \ RC	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	d	d	d	d
10	0	0	0	0

D_0:

In = 0

Q_1Q_0 \ RC	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	d	d	d	d
10	0	1	0	0

In = 1

Q_1Q_0 \ RC	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	d	d	d	d
10	1	1	0	0

前問と同様に 5 入力なので，一つの論理式についてカルノー図を 2 つ書き，それらを立体的に見る．例えば，上の D_1 の場合には，$In = 0$ と $In = 1$ で，$\bar{Q_1}\bar{R}C$ の部分が共に 1 になっているため，$Q_1 = 0, R = 0, C = 1$ の時には，入力信号 In の値によらず，$D_1 = 1$ となる．D_2 についても同様の考え方により，解答例に示した論理式が導かれる．

3 計算機システムとシステムプログラム

(1-1)	(X1) 0101	(X2) 1010	(X3) 0101	(X4) 1011
	(Y1) $-2^{n-1}+1$	(Y2) $2^{n-1}-1$	(Y3) -2^{n-1}	(Y4) $2^{n-1}-1$
(1-2-1)	$c_1 = a_0b_0 + (a_0 + b_0)c_0$			
(1-2-2)	$G_0 = g_1 + p_1g_0$			
	$P_0 = p_1p_0$			
(1-2-3)	G_0 : $3T$			
	P_0 : $2T$			
(1-2-4)	桁上げ信号	c_3		
	遅延と算出根拠 (1-2-3) より，LC_3 の出力 G_0 を求めるのに $3T$，P_0 を求めるのに $2T$ を要する．$c_2 = G_0 + P_0c_0$ であり，P_0 は 時刻 $2T$ で求まっているから，P_0c_0 は G_0 と同時に求まる．よって，時刻 $4T$ で c_2 が得られる．さらに，$c_3 = g_2 + p_2c_2$. g_2 および p_2 は，時刻 T ですでに得られているから，ここでの新たな遅延は $2T$．以上，c_3 が求まるまでに合計 $6T$ の遅延が生じる．			
(1-3)	(Z1) (エ)	(Z2) (イ)	(Z3) (オ)	(Z4) (ス)
	(Z5) (ウ)	(Z6) (サ)		

(2-1)	(a) (シ)	(b) (ク)	(c) (タ)	(d) (チ)
	(e) (タ)	(f) (チ)	(g) (タ)	(h) (タ)
	(i) (キ)	(j) (オ)	(k) (サ)	(l) (コ)
	(m) (サ)	(n) (コ)		
(2-2)	手法 1	①，②		
	手法 2	①，③		
	手法 3	①，②		

解説：

手法 1

変数 turn は，プロセス P1 とプロセス P2 のどちらのプロセスが危険領域を実行できるかを表している．turn = 2 の時 (P2 のターン) には，プロセス P1 は 5 行目の while ループで待たされるため，危険領域を実行することはできない．同様に，turn = 1 の時 (P1 のターン) には，プロセス P2 は 5 行目の while ループで待たされるため，危険領域を実行できない．それぞれのプロセスは，危険領域の実行を終了したら，8 行目で turn の値を更新して，次に相手のプロセスが危険領域を実行できるようにする．

- 自分のターンでないプロセスが危険領域の実行を待たされる事から，相互排除の条件 1 は満たされる．
- 変数 turn の値は，必ず 1 または 2 の値をとるので，プログラムの実行のどの時点においても，P1 と P2 のいずれかのプロセスは危険領域を実行できることになり，デッドロックに陥る事はない．したがって，相互排除の条件 2 は満たされる．
- 各プロセスは，必ず P1 → P2 → P1 → ··· というように，交互に危険領域を実行することになるため，例えば，P1 は，1 度危険領域を実行すると，P2 が危険領域の実行を要求していないとしても，P2 が危険領域を実行するまで，再度危険領域を実行することはできない．したがって，相互排除の条件 3 は満たされない．

手法 2

この手法では，c1 = 0 とすることにより，プロセス P1 が危険領域を実行しようとしている，または実行中であることを宣言する．このとき，プロセス P2 は危険領域を実行することはできない．また同様に，c2 = 0　により，プロセス P2 が危険領域を実行しようとしている，または実行中であることを表している．

- 各プロセスは，危険領域の実行前に，実行する事を変数を用いて宣言し，他方のプロセスが実行を宣言中ならば，そのプロセスは危険領域の実行を待つため，相互排除の条件 1 は満たされる．
- いま，初期状態において，c1 = 1，c2 = 1 であるとする．プロセス P1 が 6 行目で c1 を 0 にし，その直後に割り込みが起こってプロセス P2 に実行が移ったとする．P2 は 6 行目で c2 を 0 にした後，危険領域を実行しようとするが，c1 = 0 となっているので，7 行目の while ループで待たされる．再び P1 に実行が移ったとしても，c2 = 0 になっているので，同様に 6 行目の while ループで待たされる．したがって，デッドロックが発生し，相互排除の条件 2 は満たされない．
- プロセス P1 が待ち状態のままプロセス P2 が実行され続けたとしても，変数 c2 の値を変化させて，危険領域の実行を宣言するだけで，危険領域を繰り返し実行することができるため，手法 1 のように P2 の実行が待たされる事はなく，相互排除の条件 3 は満たされる．

手法 3

手法 3 は，手法 1 と手法 2 を組み合わせたものである．P1 は，7 行目で `c1` を 0 にして，危険領域を実行しようとしていることを宣言するが，`turn` = 2 (P2 のターン) の場合には，`c1` を再び 1 に戻して宣言を撤回し，11 行目の while ループで，自分のターン (`turn` = 1) になるまで待つ．

- 自分のターンでないプロセスは，11 行目の while ループで待たされるため，相互排除の条件 1 は満たされる．
- P1 が 7 行目で `c1` を 0 にした後に割り込みがかかって P2 に実行が移り，`c2` が 0 になったとしても，`turn` = 1 ならば，10 行目で `c2` が再び 1 に戻されるため，手法 2 のように，デッドロックに陥ることはない．したがって，相互排除の条件 2 は満たされる．
- 手法 1 と同様，P1 と P2 は，必ず交互に危険領域を実行することになるため，相互排除の条件 3 は満たされない．

8 情報論理学

(1-1) (b)

– 評価値を 1 とする解釈
- 対象領域 $D = \{0, 1\}$
- $a = 0$
- $p(0) = 0,\ \ p(1) = 1$

– 評価値を 0 とする解釈
- 対象領域 $D = \{0, 1\}$
- $a = 1$
- $p(0) = 0,\ \ p(1) = 1$

(1-2) (a)

(1-3) (b)

– 評価値を 1 とする解釈
- 対象領域 $D = \{0, 1\}$
- $p(0) = 0,\ \ p(1) = 0$

– 評価値を 0 とする解釈
- 対象領域 $D = \{0, 1\}$
- $p(0) = 1,\ \ p(1) = 1$

(1-4) (b)

– 評価値を 1 とする解釈
- 対象領域 $D = \{0, 1\}$
- $p(0, 0) = 0,\ \ p(0, 1) = 0,$
  $p(1, 0) = 0,\ \ p(1, 1) = 0$

– 評価値を 0 とする解釈
- 対象領域 $D = \{0, 1\}$
- $p(0, 0) = 1,\ \ p(0, 1) = 0,$
  $p(1, 0) = 0,\ \ p(1, 1) = 1$

(1-5) (a)

(2-1) ※ 変形の対象となる部分に下線を引いた

$$
\begin{aligned}
\neg A \ &= \ p(a,b) \wedge \forall x \forall y \exists z (p(x,y) \rightarrow p(g(x,z),y)) \wedge \forall x \forall y (p(x,y) \rightarrow p(y,x)) \\
&\quad \wedge \underline{\neg (\exists z \exists w \exists v \ p(g(g(z,w),v),a)))} \\
&= \ p(a,b) \wedge \forall x \forall y \exists z (p(x,y) \rightarrow p(g(x,z),y)) \wedge \forall x \forall y (p(x,y) \rightarrow p(y,x)) \\
&\quad \wedge \underline{\forall z \forall w \forall v \ \neg p(g(g(z,w),v),a))} \\
&= \ p(a,b) \wedge \underline{\forall x \forall y} \exists z \ (p(x,y) \rightarrow p(g(x,z),y)) \wedge \underline{\forall x \forall y} \ (p(x,y) \rightarrow p(y,x)) \\
&\quad \wedge \underline{\forall x \forall y} \forall w \ \neg p(g(g(x,y),w),a)) \\
&= \ \forall x \forall y \ [\ p(a,b) \wedge \underline{\exists z} \ (p(x,y) \rightarrow p(g(x,z),y)) \wedge (p(x,y) \rightarrow p(y,x)) \\
&\quad \wedge \forall w \ \neg p(g(g(x,y),w),a)) \ ] \\
&= \ \forall x \forall y \exists z \ [\ p(a,b) \wedge (p(x,y) \rightarrow p(g(x,z),y)) \wedge (p(x,y) \rightarrow p(y,x)) \\
&\quad \wedge \underline{\forall w} \ \neg p(g(g(x,y),w),a)) \ ] \\
&= \ \forall x \forall y \exists z \forall w \ [\ p(a,b) \wedge \underline{(p(x,y) \rightarrow p(g(x,z),y))} \wedge \underline{(p(x,y) \rightarrow p(y,x))} \\
&\quad \wedge \ \neg p(g(g(x,y),w),a)) \ ] \\
&= \ \forall x \forall y \exists z \forall w \ [\ p(a,b) \wedge (\neg \ p(x,y) \vee p(g(x,z),y)) \wedge (\neg \ p(x,y) \vee p(y,x)) \\
&\quad \wedge \ \neg p(g(g(x,y),w),a)) \ ]
\end{aligned}
$$

(2-2)

変数 z にスコーレム関数 $h(x,y)$ を導入すると，下記のスコーレム連言標準形論理式が得られる．

$$\forall x\forall y\forall z\ [\ p(a,b) \wedge (\neg\ p(x,y) \vee p(g(x,h(x,y)),y)) \wedge (\neg\ p(x,y) \vee p(y,x)) \wedge\ \neg p(g(g(x,y),w),a))\ ]$$

(2-3)

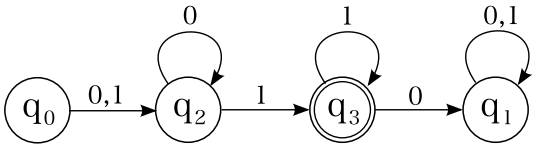
(2-2) で求めたスコーレム連言標準形論理式の各節 (下記 1. 〜 4.) から，導出節を求めていく．

1. $p(a,b)$
2. $\neg\ p(x,y) \vee p(g(x,h(x,y)),y)$
3. $\neg\ p(x,y) \vee p(y,x)$
4. $\neg p(g(g(x,y),w),a)$
5. $\neg\ p(a,b) \vee p(b,a)$ ⋯ 3. で x に a，y に b を代入
6. $p(b,a)$ ⋯ 1. と 5. の導出節
7. $\neg\ p(b,a) \vee p(g(b,h(b,a)),a)$ ⋯ 2. で x に b，y に a を代入
8. $p(g(b,h(b,a)),a)$ ⋯ 6. と 7. の導出節
9. $\neg\ p(g(b,h(b,a)),a) \vee p(g(g(b,h(b,a)),h(g(b,h(b,a)),a)),a)$ ⋯ 2. で x に $g(b,h(b,a))$，y に a を代入
10. $p(g(g(b,h(b,a)),h(g(b,h(b,a)),a)),a)$ ⋯ 8. と 9. の導出節
11. $\neg p(g(g(b,h(b,a)),h(g(b,h(b,a)),a)),a)$ ⋯ 4. で x に b，y に $h(b,a)$，w に $h(g(b,h(b,a)))$ を代入
12. 0 ⋯ 10. と 11. の導出節

導出節として空節 0 が導かれたので，$\neg A$ は充足不能．よって，論理式 A は恒真である．

9 計算理論

(1-1)



(1-2)　M_3 : (e)　M_4 : (b)　M_5 : (h)

(1-3)　(1) $(0, A)/AAA$　(2) $(1, A)/A$　(3) $(0, A)/\epsilon$

(2-1)

$G_2 = (V_2, T_2, P_2, S_2)$

- 非終端記号の集合 $V_2 = V_1$
- 終端記号の集合 $T_2 = \{a,\ b,\ c,\ d\}$
- 開始記号 $S_2 = S$
- 生成規則の集合 $P_2 = \{S \to A,\ S \to AB,\ A \to AD,\ A \to ABD,\ A \to ACD,$
  $A \to ABCD,\ A \to a,\ B \to C,\ B \to b,\ C \to B,\ C \to c,\ D \to d\}$

(2-2)

チョムスキー標準形とは，すべての生成規則が $A \to BC$，$A \to a$，$S \to \epsilon$ のいずれかの形をした，ϵ なし文脈自由文法である (S, A, B, C は非終端記号，a は終端記号，S は開始記号).

(2-3)

$G_4 = (V_4, T_4, S_4, P_4)$

- 非終端記号の集合 $V_4 = \{S,\ A,\ B,\ C,\ D,\ E\}$
- 終端記号の集合 $T_4 = \{a,\ b\}$
- 開始記号 $S_4 = S$
- 生成記号の集合 $P_4 = \{S \to AB,\ B \to DC,\ C \to AE,\ D \to a,\ E \to b\}$

解説:

(1-1)

決定性有限オートマトン M_1 の各状態を，0-等価性，すなわち各々の状態自身が受理状態であるかどうかにもとづいて分類すると，$(\{p, q, s, t, u, v\}, \{\mathrm{r}\})$ となる．これを Π_0 とおく.

続いて，Π_0 の 1-等価性に基づく細分 Π_1 を考える．Π_1 を求めるには，同じ同値類に入っている状態がすべての 1 文字入力について，遷移した後 Π_0 の同じ同値類に入るかどうかを見て決めれば良いことになる.

0						1
p	q	s	t	u	v	r
00	00	00	01	00	01	01

上の表から，$\Pi_1 = (\{p,q,s,u\},\{t,v\},\{r\})$ であることが分かる．同様に，Π_1 の 2-等価性に基づく細分 Π_2 を求める．

0				1		2
p	q	s	u	t	v	r
11	00	00	00	12	12	02

上の表から，$\Pi_2 = (\{p\},\{q,s,u\},\{t,v\},\{r\})$ が得られる．さらに，Π_2 の 3-等価性による細分 Π_3 を求める．

0	1			2		3
p	q	s	u	t	v	r
22	11	11	11	23	23	13

上の表から，$\Pi_3 = (\{p\},\{q,s,u\},\{t,v\},\{r\})$ となる．$\Pi_3 = \Pi_2$ であることから，これ以上細分を行うことはできない．よって，$L(M_1) = L(M_2)$ を満たす状態数最小の決定性有限オートマトン M_2 の状態遷移図は上記の解答例のようになる．

(2-1)

ϵ なし文法とは，$X \to \epsilon$ の形の生成規則 (ϵ 規則) を一つも含まないか，1 つだけ $S \to \epsilon$ の形の ϵ 規則を許すが，S はどの規則の右辺にも現れないような文法である．文脈自由文法を ϵ なし文法に変換するためには，まず，空語 ϵ を生成しうる非終端記号の集合 N_ϵ を求める．これは，教科書で紹介されている，マーク付けの方法で判定することができる．ここでは，$N_\epsilon = \{B,C\}$ となる．

あとは，生成規則集合から $B \to \epsilon$ と $C \to \epsilon$ を取り除き，教科書の方針に従って適切な規則を加えれば，求める文法が得られる．この問題では，開始記号 $S \notin N_\epsilon$ なので，(b) の考慮は不要である．

(2-3)

ここでは，$< >$ で囲んだ記号列を新たな非終端記号と考えて，変換の過程を示す．

- $S \to A < aAb >$
- $< aAb > \to < a >< Ab >$
- $< Ab > \to A < b >$
- $< a > \to a$
- $< b > \to b$

$< aAb > = B$，$< Ab > = C$，$< a > = D$，$< b > = E$ と記号の置き換えを行えば，解答例で示したチョムスキー標準形文法が得られる．