平 成 ３１ 年 度

名古屋大学大学院情報学研究科
情報システム学専攻
入学試験問題（専門）

平成３０年８月８日

注意事項

１．試験開始の合図があるまでは，この問題冊子を開いてはならない．

２．試験終了まで退出できない．

３．外国人留学生は，語学辞書１冊に限り使用してよい．電子辞書の持ち込みは認めない．

４．外国人留学生は，英語での解答を可とする．

５．問題冊子，解答用紙３枚，草稿用紙３枚が配布されていることを確認すること．

６．問題は，(1)確率・統計，(2)プログラミング，(3)計算機理論，(4)ハードウェア，(5)ソフトウェアの５科目がある．このうち３科目を選択して解答すること．なお，選択した科目名を解答用紙の指定欄に記入すること．

７．全ての解答用紙の所定の欄に受験番号を必ず記入すること．解答用紙に受験者の氏名を記入してはならない．

８．解答用紙に書ききれない場合は，裏面を使用してもよい．ただし，裏面を使用した場合は，その旨，解答用紙表面右下に明記すること．

９．解答用紙は試験終了後に３枚とも提出すること．

１０．問題冊子，草稿用紙は試験終了後に持ち帰ること．

確率・統計

解の導出過程も書くこと.

[1] 赤玉が 4 つ，青玉が 1 つ，白玉が 1 つ入った箱 A と，赤玉が 2 つ，青玉が 2 つ，白玉が 2 つ入った箱 B があるとき，以下の問いに答えよ．ただし 2 つの箱は外からは区別が付かず，玉を取り出すとき，箱の中は見えないものとする.

(1) 無作為に 1 つの箱を選び，その箱から玉を 1 つ取り出したとき，それが赤玉である確率を求めよ.

(2) 無作為に 1 つの箱を選び，その箱から玉を 2 つ取り出したとき，それらの色が異なる確率を求めよ.

(3) 1 つの箱から玉を 1 つ取り出したとき，それが赤玉であったとする．その玉を箱に戻さずに，さらに玉を 1 つ取り出すとき，最初に玉を取り出したのと同じ箱から取り出す場合と，別の箱から取り出す場合，それぞれについて新たに取り出した玉が赤玉である確率を求め，どちらの場合の方が赤玉である確率が大きいか答えよ.

[2] 確率変数 X の確率密度関数 $f(x)$ が次式で与えられている．ただし a は定数とする.

$$f(x) = \begin{cases} 12x^2(a-x) & (0 \leq x \leq 1) \\ 0 & (\text{上記以外}) \end{cases}$$

(1) a の値を求めよ.

(2) 確率変数 X の値が $1/3$ 以下となる確率を求めよ.

(3) 確率変数 X の平均 μ と分散 σ^2 を求めよ.

(4) 表が出る確率 θ の確率分布が $f(\theta)$ で与えられたコインがあるとする．このコインを n 回投げたときにすべて裏が出る確率の期待値を求めよ．ただし，n 回の試行の間，θ の値は変動しないものとする.

Translation of technical terms

- 無作為: random
- 確率密度関数: probability density function
- 平均: mean
- 確率分布: probability distribution
- 確率変数: random variable
- 定数: constant
- 分散: variance
- 期待値: expectation

プログラミング

プログラムPは与えられた文字列を操作するC言語プログラムである．プログラムPで扱う文字は，1バイト，または，3バイトで構成される．1バイトで構成される文字は最上位ビットが0であり，残りのビットにより文字を指定する．3バイトで構成される文字は，その1バイト目の最上位ビットが1であり，1バイト目の残りのビットと2バイト目，ならびに，3バイト目により文字を指定する．文字列は8ビット長のchar型の配列に格納され，`'¥0'`を終端とする．16進数は，`0x41`のように先頭に`0x`をつけて表す．すなわち，`0x41`は10進表記の65の16進表記である．

プログラムPにおいて，27行目の関数`concat`は，ポインタ`s1`，ポインタ`s2`がこの順で引数として与えられたとき，`s1`と`s2`が参照する文字列をこの順に連結し，得られた文字列の先頭を参照するポインタを返す．59行目の関数`replace`は，ポインタ`s1`，ポインタ`s2`，正の整数`k`がこの順で引数として与えられたとき，`s1`が参照する文字列の`k`文字目を`s2`が参照する文字列で置き換えて得られる文字列の先頭を参照するポインタを返す．なお，関数`replace`の引数`s2`が参照する文字列の文字数は1とし，1文字は1バイトまたは3バイトで構成されることに注意せよ．

このとき以下の問いに答えよ．なお，ポインタが参照する文字列の内容を解答する場合，下の記述例のようにアドレスと値を対応づけて文字列を表すこと（終端も含めること）．例えば，80行目の`r = str1;`の実行直後のポインタ`r`が参照する文字列の内容は，下の記述例のように解答する．

アドレス	`r`	`r+1`	`r+2`	`r+3`
値	`0x41`	`0x42`	`0x43`	`'¥0'`

(1) 空欄`A`，`B`，`C`，`D`，`E`にあてはまる式を書いて，関数`concat`を完成させよ．

(2) 76行目の`r = index(str1, 1);`を実行した直後の`r`が参照する文字列の内容を記述例にならって書け．

(3) 77行目の`r = index(str2, 2);`を実行した直後の`r`が参照する文字列の内容を記述例にならって書け．

(4) 78行目の`r = index(str3, 3);`を実行した直後の`r`が参照する文字列の内容を記述例にならって書け．

(5) 79行目の`r = replace(str3, str4, 3);`の実行時に呼び出される62行目の`tmp2 = concat_p(string1, string2, tmp1);`を実行した直後の`tmp2`の内容を記述例にならって書け．

(6) 48行目では下線部のように固定長の領域を`result`に割り当てている．しかしながら，関数`concat_p`に与えられる引数によっては領域が不足する．与えられる引数に応じて必要な領域が割り当てられるように下線部を適切な式に変更せよ．

(7) 空欄`F`，`G`，`H`にあてはまる式を書いて関数`replace`を完成させよ．

プログラム P

```
1   #include <stdlib.h>
2
3   char* index(char* string, int p){
4     char *tmp;
5     int i;
6     i = 1;
7     tmp = string;
8
9     while ( (*tmp != '¥0') && (i < p) ){
10      i++;
11      if ((*tmp & 0x80) == 0)
12        tmp++;
13      else
14        tmp += 3;
15    }
16    return tmp;
17  }
18
19  int length_b(char* string){
20    int len;
21    len = 0;
22    while (string[len] != '¥0')
23      len++;
24    return len;
25  }
26
27  char* concat(char* string1, char* string2){
28    char *result, *tmp;
29    result = (char*)malloc(sizeof(char) * (length_b(string1) + length_b(string2) + 1));
30    tmp = result;
31    while (*string1 != '¥0'){
32      [ A ] = [ B ];
33      tmp++;
34      string1++;
35    }
36    while (*string2 != '¥0'){
37      [ C ] = [ D ];
```

```
38      tmp++;
39      string2++;
40    }
41    *tmp = [      E      ];
42    return result;
43  }
44
45  char* concat_p(char* string1, char* string2, char* l){
46    char *result, *tmp1, *tmp2;
47    tmp1 = string1;
48    result = (char*)malloc(sizeof(char) *  100 );
49    tmp2 = result;
50    while ((tmp1 != l) && (*tmp1 != '¥0')){
51      *tmp2 = *tmp1;
52      tmp1++;
53      tmp2++;
54    }
55    *tmp2 = '¥0';
56    return concat(result, string2);
57  }
58
59  char* replace(char* string1, char* string2, int p){
60    char *tmp1, *tmp2;
61    tmp1 = index(string1, p);
62    tmp2 = concat_p(string1, string2, tmp1);
63    if ( [      F      ] == 0)
64      [      G      ];
65    else
66      [      H      ];
67    return concat(tmp2, tmp1);
68  }
69
70  void main(){
71    char str1[4] = {0x41, 0x42, 0x43, '¥0'};
72    char str2[10] = {0xE3, 0x81, 0x82, 0xE3, 0x81, 0x84, 0xE3, 0x81, 0x86, '¥0'};
73    char str3[9] = {0xE3, 0x81, 0x82, 0x41, 0xE3, 0x81, 0x84, 0x42, '¥0'};
74    char str4[4] = {0xE3, 0x81, 0x86, '¥0'};
75    char *r;
```

```
76      r = index(str1, 1);
77      r = index(str2, 2);
78      r = index(str3, 3);
79      r = replace(str3, str4, 3);
80      r = str1;
81    }
```

Translation of technical terms

プログラム	program	関数	function
文字列	string	ポインタ	pointer
C 言語	C programming language	引数	argument
バイト	byte	参照する	refer
最上位ビット	most significant bit	アドレス	address
配列	array	式	expression
16 進数	hexadecimal	固定長	fixed length
10 進表記	decimal notation	割り当てる	allocate
16 進表記	hexadecimal notation		

計算機理論

[1] 決定性有限オートマトンは，アルファベット（記号の有限集合）Σ，状態の有限集合 Q，初期状態 q_0（$\in Q$），遷移関数 $\delta : \Sigma \times Q \to Q$，最終状態の集合 Q_f（$\subseteq Q$）から定められる．Σ に属する記号の列を Σ 上の語と呼び，語の集合 L を Σ 上の言語と呼ぶ．決定性有限オートマトン A が受理する語の集合を $\mathcal{L}(A)$ と記す．すなわち，$\mathcal{L}(A)$ は言語である．言語 L について，ある決定性有限オートマトン A が存在して $L = \mathcal{L}(A)$ を満たすとき，その言語 L を正規言語と呼び，L は A に認識されるという．語 w に対して，w^i は $\overbrace{w \ldots w}^{i}$ の略記である．なお，$i = 0$ のときは w^i は空列 ϵ である．語 w の長さを $|w|$ で表すこととする．例えば，$\{0, 1\}$ をアルファベットとしたとき，語 0100 の長さ $|0100|$ は 4 である．また，10^5 は 100000 を表し，$|10^5| = 6$ である．また，自然数の集合を $\mathbb{N}$ とする（$0 \in \mathbb{N}$ とする）．アルファベット $\{0, 1\}$ 上の言語，決定性有限オートマトンについて以下の問いに答えよ．

(1) 以下のように定義される言語 L_a, L_b, L_c は正規言語である．L_a, L_b, L_c を認識する決定性有限オートマトン A_a, A_b, A_c それぞれの状態遷移図を描け．なお，状態遷移図には初期状態および最終状態を必ず明記すること．

(a) $L_a = \{0^m 10^n \mid m \in \mathbb{N},\ n \in \mathbb{N}\}$

(b) $L_b = \{w \mid w \in L_a,\ |w| \text{ が奇数である}\}$

(c) $L_c = \{0^m 10^m \mid m \in \mathbb{N},\ 0 \leq m \leq 3\}$

(2) 言語 $L_d = \{0^m 10^m \mid m \in \mathbb{N},\ m \geq 1\}$ について考える．

(a) L_d に含まれる長さが 10 以下である語をすべて示せ．

(b) L_d は正規言語ではないことを，以下の反復補題を利用して証明せよ．

反復補題 正規言語 L に対して自然数 n（> 0）が存在して，$|w| \geq n$ である任意の語 $w \in L$ について以下のすべてを満たす語 x, y, z が存在する．

- $w = xyz$
- $|y| > 0$
- $|xy| \leq n$
- 任意の自然数 $i \in \mathbb{N}$ について，$xy^i z \in L$

Translation of technical terms

決定性有限オートマトン	deterministic finite automaton	語	word
		言語	language
アルファベット	alphabet	受理する	accept
記号	symbol	正規言語	regular language
有限集合	finite set	認識する	recognize
状態	state	空列	empty sequence
初期状態	initial state	自然数	natural number
遷移関数	transition function	状態遷移図	state transition diagram
最終状態	final state	奇数	odd number
列	sequence	反復補題	pumping lemma

[2] 自然数の集合を $\mathbb{N}$ とし（$0 \in \mathbb{N}$ とする），自然数の順序対の集合を $\mathbb{N} \times \mathbb{N}$ とする．すなわち，$\mathbb{N} \times \mathbb{N} = \{(n, m) \mid n, m \in \mathbb{N}\}$ とする．さらに，自然数の有限列の集合を $\mathbb{N}^*$ とする．すなわち，$\mathbb{N}^* = \{()\} \cup \{(n_1, \ldots, n_k) \mid k \geq 1 \text{ かつ } n_1 \in \mathbb{N}, \ldots, n_k \in \mathbb{N}\}$ とする．ここで，() は空列を表す．集合 A から B への写像 $f : A \to B$ が

「任意の $a, a' \in A$ について，$a \neq a'$ ならば $f(a) \neq f(a')$」

を満たすとき，f は単射であるという．

(1) $\mathbb{N}$ から $\mathbb{N} \times \mathbb{N}$ への写像で単射であるものを一つ与えよ．

(2) $\mathbb{N} \times \mathbb{N}$ から $\mathbb{N}$ への写像で単射であるものを一つ与えよ．

(3) $\mathbb{N}^*$ から $\mathbb{N}$ への写像で単射であるものを一つ与えよ．

Translation of technical terms

自然数	natural number
集合	set
順序対	ordered pair
有限列	finite sequence
空列	empty sequence
写像	mapping
単射	injection

ハードウェア

[1] キャッシュは，メインメモリの中でアクセス頻度が高いデータをキャッシュメモリに置くことで，メモリの平均アクセス時間を短縮する．また，仮想記憶は，メインメモリの中でアクセス頻度が低いデータを補助記憶（ストレージ）に置くことで，メインメモリの容量を超えるメモリを扱うことを可能にする．このような概念を，記憶の階層と呼ぶ．これに関する以下の問いに答えよ．

(1) 記憶の階層の概念について，ハードウェアの特性を踏まえて 100 文字（英文の場合は 50 語）以内で説明せよ．

(2) 記憶の階層は，参照の局所性を利用して，メモリの平均アクセス時間を短縮する．参照の局所性には，時間的局所性と空間的局所性がある．時間的局所性と空間的局所性について，それぞれ 50 文字（英文の場合は 25 語）以内で説明せよ．

(3) 仮想記憶におけるページは，空間的局所性を活用するためのものである．仮想記憶においてページに該当するものは，キャッシュにおいては何か？ また，これらの仕組みがどのようにメモリの平均アクセス時間を短縮するかについて，100 文字（英文の場合は 50 語）以内で説明せよ．

(4) 記憶の階層において，キャッシュメモリよりも高速な記憶としては何があるか？

(5) 計算機システムの性能の向上のためには，記憶の階層はさらに多くなる傾向にある．具体的に，どのように階層が多くなっているか，50 文字（英文の場合は 25 語）以内で説明せよ．

Translation of technical terms

キャッシュ	cache	記憶の階層	memory hierarchy
メインメモリ	main memory	参照の局所性	locality of reference
平均アクセス時間	average access time	時間的局所性	temporal locality
仮想記憶	virtual memory	空間的局所性	spatial locality
補助記憶	auxiliary memory	ページ	page

[2] 4ビットの符号無し整数値の除算ハードウェアを図1に，このハードウェアのアルゴリズムを図3に示す．除数レジスタの内容は右シフト可能であり，シフト後の最上位ビットは0となる．商レジスタの内容は左シフト可能であり，シフト後の最下位ビットの値は0か1を指定可能である．演算開始時，除数は除数レジスタの上位4ビットに，被除数は剰余レジスタの下位4ビットに格納する．各レジスタのその他のビットは0にする．演算終了時に商は商レジスタに，剰余は剰余レジスタの下位4ビットに格納される．

(1) 図3の（A）（B）（C）を埋めて除算アルゴリズムを完成せよ．

(2) このハードウェアに被除数として十進数の10を入力し，除数として十進数の3を入力した場合の計算過程を下記の形式で示せ．各ステップにおける値は，アルゴリズム中の⑤を実行する際の値とする．なお，各レジスタの値は二進数で記述すること．

ステップ	商レジスタ	除数レジスタ	剰余レジスタ
start	0000	00110000	00001010
1	0000	00011000	00001010
2	....	...	...

(3) 図1のハードウェアの面積と実行サイクルを縮小するためにハードウェア構成を変更した．改良版除算ハードウェアを図2に示す．剰余レジスタの上位4ビットはALUの入力へ，ALUの計算結果は剰余レジスタの上位4ビットに接続されている．剰余レジスタの内容は左シフト可能であり，シフト後の最下位ビットの値は0か1を指定可能である．改良版除算ハードウェアでは，演算開始時に被除数は剰余レジスタ下位4ビットに格納され，除数は除数レジスタに格納され，各レジスタのその他ビットは0とする．演算終了時に商は剰余レジスタの下位4ビットに，剰余は剰余レジスタの上位4ビットに格納されるとする．改良版除算ハードウェアのアルゴリズムを図3と同様の記法で記述せよ．

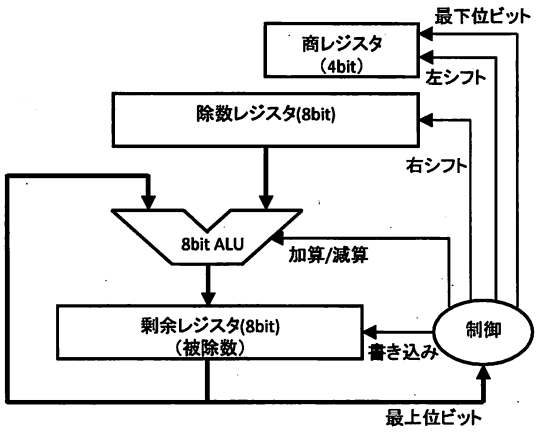


図 1: 除算ハードウェア

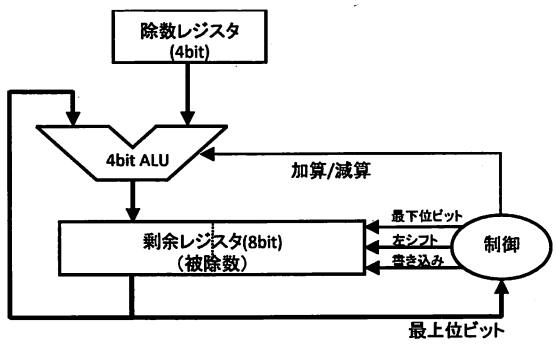


図 2: 改良版除算ハードウェア

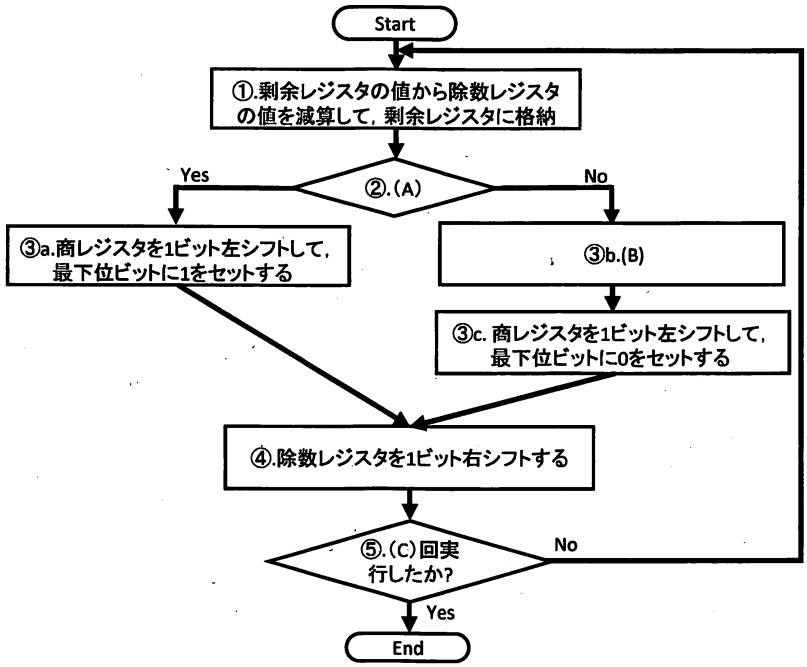


図 3: 除算ハードウェアのアルゴリズム

Translation of technical terms

ビット	bit
符号無し	unsigned
整数値	integer
除算	division
ハードウェア	hardware
アルゴリズム	algorithm
除数	divisor
レジスタ	register
上位	upper
格納	store
被除数	dividend
剰余	remainder
商	quotient
下位	lower
十進数	decimal
ステップ	step
二進数	binary
面積	area
実行サイクル	execution cycle
ハードウェア構成	hardware construction
左シフト	left shift
右シフト	right shift
加算	add
減算	subtract
制御	control
最下位ビット	least significant bit
最上位ビット	most significant bit

ソフトウェア

[1] 次の文脈自由文法 $G = \langle N, T, P, S' \rangle$ の LR構文解析について以下の問いに答えよ．ここで，非終端記号の集合 $N = \{S', S, A, B\}$, 終端記号の集合 $T = \{=, \mathtt{a}, @\}$ とし，生成規則の集合 P を以下のように定める．S' は開始記号である．各規則の前の数字は規則につけられた番号であり，# は，入力文字列の終端を示す特別な記号である．

$$P = \left\{ \begin{array}{llllllll} 1: & S' & \to & S\#, & 4: & A & \to & @B, \\ 2: & S & \to & A = B, & 5: & A & \to & \mathtt{a}, \\ 3: & S & \to & B, & 6: & B & \to & A \end{array} \right\}$$

G に対する LR(0) オートマトンの状態遷移図は図 1 のように構成される．

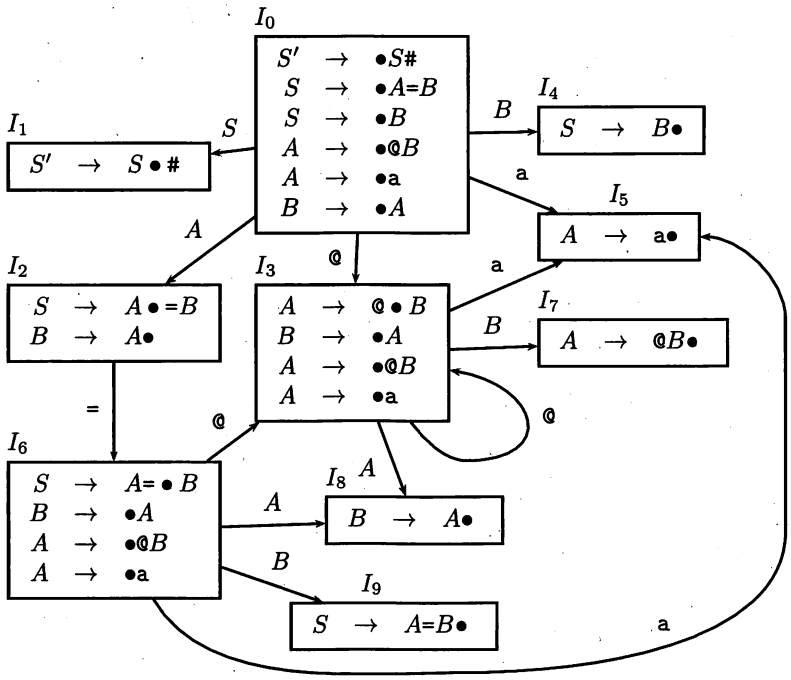


図 1. G に対する LR(0) オートマトン

(1) @a=@a# に対する S' からの最右導出を示せ．一段階の導出は $\alpha \Rightarrow \beta$ で記述せよ．ここで α, β は文形式である．

(2) S, A, B に対する FOLLOW 集合を求めよ．

(3) 表 1 の SLR(1) 構文解析表を解答用紙に書き写して完成させよ．Action 表におけるシフト動作は sn(n は次状態 I_n の n)，還元動作は rn(n は還元に用いる生成規則の番号)，受理は acc と記載する．Goto 表には，状態 I_n の n を記載する．完成した SLR(1) 構文解析表から，G が SLR(1) 構文解析できないことを説明せよ．

(4) 図 2 の G に対する LR(1) オートマトンの状態遷移図の (a) ~ (g) に適当な LR(1) 項を示せ．

(5) 表 1 と同様に表 2 の LR(1) 構文解析表を解答用紙に書き写して完成し，G が LR(1) 構文解析可能であることを説明せよ．

表 1. SLR 構文解析表

状態	Action 表				Goto 表		
	a	@	=	#	S	A	B
I_0							
I_1							
I_2							
I_3							
I_4							
I_5							
I_6							
I_7							
I_8							
I_9							

表 2. LR(1) 構文解析表

状態	Action 表				Goto 表		
	a	@	=	#	S	A	B
I_0							
I_1							
I_2							
I_3							
I_4							
I_5							
I_6							
I_7							
I_8							
I_9							
I_{10}							
I_{11}							
I_{12}							
I_{13}							

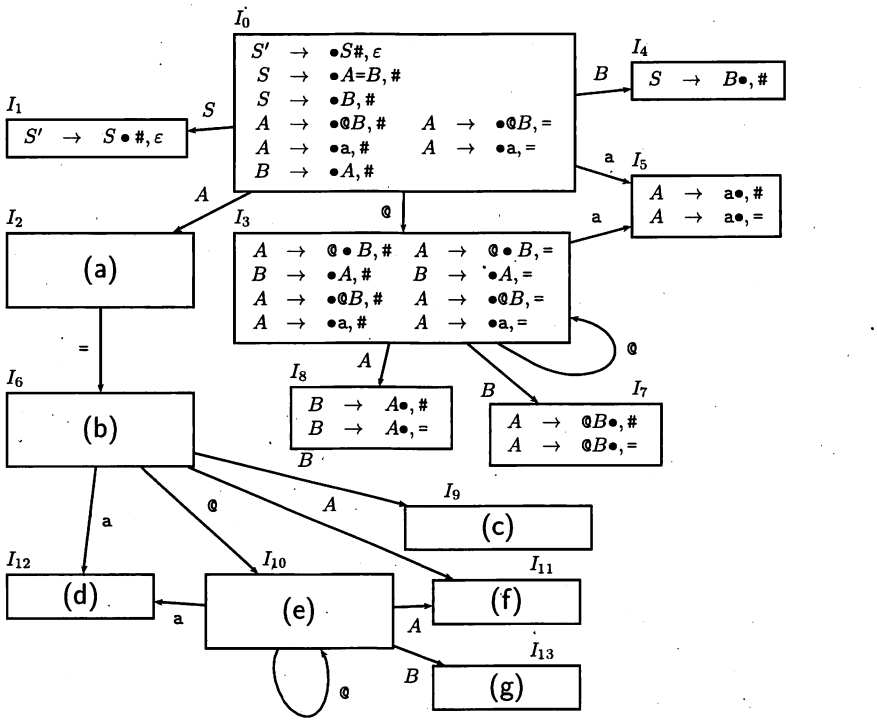


図 2. G に対する LR(1) オートマトン

Translation of Technical terms:

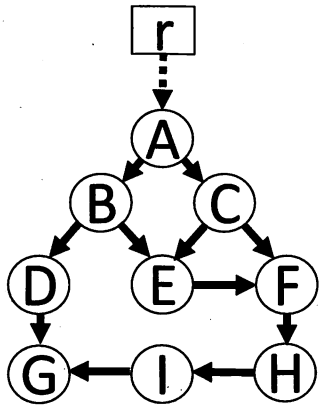
文脈自由文法	context-free grammar	最右導出	rightmost derivation
構文解析	parsing	文形式	sentential form
非終端記号	nonterminal symbol	構文解析表	parsing table
終端記号	terminal symbol	シフト	shift
生成規則	production rule	還元	reduction
開始記号	start symbol	受理	acceptance
状態遷移図	state transition diagram		

[2] ガーベジコレクションに関する以下の問いに答えよ．

(1) メモリリークとは何か？50 文字（英文の場合は 25 語）以内で述べよ．

(2) 手続き内で生成され，その手続きが完了した後も生存するオブジェクト（記憶領域を必要とするデータ項目）を格納するための記憶領域を何と呼ぶか答えよ．

(3) 下図の丸で示す頂点はそれぞれオブジェクト，それらの間の有向辺は始点のオブジェクトから終点のオブジェクトへの参照を表す．四角で示す根 r はプログラムから参照できる変数を，根 r を始点とする有向辺はその変数からオブジェクト A への参照を表す．この図において，オブジェクト H からオブジェクト I への参照が削除されたときに，マーク・アンド・スイープが適用されると，オブジェクト I 以外の全てのオブジェクトがマークされ，オブジェクト I のみがスイープされる．更に，オブジェクト A からオブジェクト B への参照が削除され，マーク・アンド・スイープが適用されたとき，マークされるオブジェクトとスイープされるオブジェクトをそれぞれ全て答えよ．



(4) C言語のように，整数型の値をポインタ型の値に型変換できるプログラミング言語を対象としたガーベジコレクションでは，一部のオブジェクトを解放できないときがある．その理由を 70 文字（英文の場合は 35 語）以内で述べよ．

(5) マーク・アンド・スイープには，記憶領域の断片化を軽減することを目的として，オブジェクトの再配置を行う方式がある．どのように再配置すれば断片化を軽減することができるか 70 文字（英文の場合は 35 語）以内で述べよ．

Translation of Technical terms:

ガーベジコレクション	garbage collection	根	root
メモリリーク	memory leak	プログラム	program
手続き	procedure	変数	variable
生存する	live	削除する	delete
オブジェクト	object	マーク・アンド・スイープ	mark-and-sweep
記憶領域	memory area	言語	language
データ項目	data item	整数型	integer type
格納する	store	ポインタ型	pointer type
頂点	vertex	型変換	type conversion
有向辺	directed edge	解放	deallocation
始点	start vertex	断片化	fragmentation
終点	end vertex	再配置	relocation
参照	reference		