平成 18 年度

東京大学大学院情報理工学系研究科

コンピュータ科学専攻

入学試験問題

専門科目 I

解答

東京大学理学部情報科学科 2009

2012 年 8 月 7 日

問題 1

問題

一階論理に関して，以下の問いに答えよ．

(1)「充足可能な論理式の否定は，充足不可能な論理式となる」は，正しいか？正しいと思う場合は，簡単な証明を与えよ．また，正しくないと思う場合は，反例を与えよ．
(2)「推論系が完全である」,「推論系が健全である」とは，どういうことかをそれぞれ簡単に述べよ．
(3) 完全な推論系 α がある．推論系 α による，論理式集合 Γ から式 ϕ の証明が存在しないとする．このとき，論理式集合 $\Gamma \cup \{\neg\phi\}$ は充足可能となることを示せ．
(4) 一階論理の完全で健全な推論系を一つ挙げて簡単に説明せよ．さらに，それを使って次のことを証明せよ．
(a) $(\exists x \forall y P(x,y)) \rightarrow (\forall x \exists y P(y,x))$
(b) $(\forall x \forall y (P(x,y) \rightarrow P(y,x))) \rightarrow (\neg\exists x \exists y \neg(\neg P(x,y) \vee P(y,x)))$

解説 (1)

問題

「充足可能な論理式の否定は，充足不可能な論理式となる」は，正しいか？正しいと思う場合は，簡単な証明を与えよ．また，正しくないと思う場合は，反例を与えよ．

解説

一階論理と書いてありますが、この問題はさらにその下の命題論理のレベルで考えて問題ないです。

充足可能　論理式を真にする解釈が存在する、つまり、各命題記号に T/F を代入していったときに全体が T になるような代入の仕方が存在する。
恒真　どんな解釈でも全体が真になる
充足不可能　どんな解釈でも全体が偽になる

当然正しくなくて、最も簡単な反例として、「P」を挙げればいいでしょう。「P」は P=T で真になって充足可能ですし、「$\neg P$」は P=F で真になって充足可能です。
ちなみに、「充足不可能」な論理式の否定は「恒真」になります。
なお、一階論理では「充足可能」⇔「論理式を真にする解釈と付値が存在する」となります。

解説 (2)

問題

「推論系が完全である」,「推論系が健全である」とは，どういうことかをそれぞれ簡単に述べよ．

解説

■健全性　全ての定理は恒真

■弱い完全性　全ての恒真な論理式は定理である

■強い完全性　全ての論理式 A に対して、論理式集合 Γ に対して Γ を充足する解釈が必ず A を充足する ($A \models \Gamma$) ならば、Γ に属する論理式から A が証明可能 ($A \vdash \Gamma$)

ここで、強い完全性は忘れがちなので、しっかり覚えましょう。でないと次の問題ではまります。

解説 (3)

問題

完全な推論系 α がある．推論系 α による，論理式集合 Γ から式 ϕ の証明が存在しないとする．このとき，論理式集合 $\Gamma \cup \{\neg\phi\}$ は充足可能となることを示せ.

解説

強い完全性で答えたほうがいいでしょう。

強い完全性の対偶を考えて、論理式集合 Γ から式 ϕ の証明が存在しないならば Γ を充足する解釈で、ϕ を充足しない、すなわち ϕ を $\bot$ とする解釈 I が存在します。

I は $\neg\phi$ を充足するので、I は Γ に含まれる全ての論理式と $\neg\phi$ を充足する、すなわち $\Gamma \cup \{\neg\phi\}$ を充足することになり、$\Gamma \cup \{\neg\phi\}$ は充足可能であることが示されます。

解説 (4)

問題

一階論理の完全で健全な推論系を一つ挙げて簡単に説明せよ．さらに，それを使って次のことを証明せよ.

(a) $(\exists x \forall y P(x,y)) \to (\forall x \exists y P(y,x))$
(b) $(\forall x \forall y (P(x,y) \to P(y,x))) \to (\neg\exists x \exists y \neg(\neg P(x,y) \vee P(y,x)))$

解説

Hilbert 流も自然演繹も健全で強い意味で完全なので、好きな方を挙げればいいでしょう。

ただ、問題文にある「簡単に説明せよ」で健全性と強い完全性が成立することについて説明しようとするとあまりに時間がかかりすぎるので、(ってか自然演繹の方は知らんしw) 内容に関してだけ説明すればいいと思います。むしろこれを説明するなら他の問題を解き終わったあとでも遅くないでしょうw

Hilbert 流

でもこれ覚えるのキツイ… 自然演繹使うのがおすすめです。

■内容

公理 1. 全ての命題論理のトートロジーの各命題記号を一回論理式で置き換えたもの
公理 2. (代入の公理)$(\forall x.A[x]) \supset A[t]$
公理 3. $(\forall x.(B \supset A[x])) \supset (B \supset (\forall x.A[x]))$

推論規則 1.modus ponens　A と $A \supset B$ から B を導く

推論規則 2. **汎化**　$A[x]$ から $\forall x.A[x]$ を導く

解説　ヒルベルト流では以下の証明の仕方を覚えておくのがおすすめ

$A[t] \supset \exists x.A[x]$ を示す。

トートロジー $(P \to \neg Q$ の対偶$)$ $(P \to \neg Q) \to (Q \to \neg P)$ の P,Q に $\forall x.\neg A[x]$,$A[t]$ を代入した

$((\forall x.\neg A[x]) \to \neg A[t]) \to (A[t] \to \neg(\forall x.\neg A[x]))$

は公理である。

また、代入の公理から $(\forall x.\neg A[x]) \to \neg A[t]$

よって、modus ponens から、$(A[t] \to \neg(\forall x.\neg A[x]))$

すなわち $A[t] \supset \exists x.A[x]$ が示される。

■(a)　$(\exists x \forall y P(x,y)) \to (\forall x \exists y P(y,x))$

$\exists$ を展開する

$(\neg(\forall x.\neg(\forall y.P(x,y)))) \to (\forall x.\neg(\forall y.\neg P(y,x)))$

これを証明していくことになる。

(゜η゜)　トケネーヨ

ってことで慣れている自然演繹の法を使いましょう。

自然演繹

■内容

公理　$\Gamma \vdash A$ when $A \in \Gamma$

推論規則　A から B を導くというのを、

$$\frac{A}{B}$$

と書く事にする。

∧の導入則

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$$

∧の除去則

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}$$

∨の導入則

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}$$

∨の除去則

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$$

→の導入則

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$$

→の除去則

$$\frac{\Gamma \vdash A \quad \Gamma \vdash A \to B}{\Gamma \vdash B}$$

¬の導入則

$$\frac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A}$$

¬の除去則

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}$$

¬の二重除去則

$$\frac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}$$

∀の導入則

$$\frac{\Gamma \vdash A[x]}{\Gamma \vdash \forall x.A[x]}$$

ただし Γ に x は自由に出現しない

∀の除去則

$$\frac{\Gamma \vdash \forall x.A[x]}{\Gamma \vdash A[t]}$$

∃の導入則

$$\frac{\Gamma \vdash A[t]}{\Gamma \vdash \exists x.A[x]}$$

∃の除去則

$$\frac{\Gamma \vdash \exists x.A[x] \quad \Gamma, A[y] \vdash C}{\Gamma \vdash C}$$

ただし、Γ と C に y の自由な出現はない。

復習するときは、「$A \vee B$ を導くときは、$\neg(A \vee B)$ を仮定して A と B をそれぞれ仮定して矛盾を導く」というテクニックを覚えておくと非常に役に立つので、自信のない人は下の 2 つを必ず一度自分でやっておきましょう。

排中律

$$\dfrac{\dfrac{\dfrac{\neg(P \vee \neg P) \vdash \neg(P \vee \neg P) \quad \dfrac{\neg(P \vee \neg P), P \vdash P}{\neg(P \vee \neg P), P \vdash (P \vee \neg P)}}{\neg(P \vee \neg P), P \vdash \bot}}{\neg(P \vee \neg P) \vdash \neg P}}{\dfrac{\neg(P \vee \neg P) \vdash \neg(P \vee \neg P)}{\neg(P \vee \neg P) \vdash \bot}}{\vdash \neg\neg(P \vee \neg P)}{\vdash (P \vee \neg P)}$$

ド・モルガンの法則の∧の否定

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{..., \neg(\neg P \vee \neg Q), \neg P \vdash \neg(\neg P \vee \neg Q)} \quad \dfrac{}{..., ..., \neg P \vdash \neg P}}{..., ..., \neg P \vdash \neg P \vee \neg Q}}{..., \neg(\neg P \vee \neg Q), \neg P \vdash \bot}}{..., \neg(\neg P \vee \neg Q) \vdash \neg\neg P}}{..., \neg(\neg P \vee \neg Q) \vdash P} \quad \dfrac{\text{左に同じ}}{..., \neg(\neg P \vee \neg Q) \vdash Q}}{..., \neg(\neg P \vee \neg Q) \vdash \neg(P \wedge Q)}}{\neg(P \wedge Q), \neg(\neg P \vee \neg Q) \vdash P \wedge Q}}{\neg(P \wedge Q), \neg(\neg P \vee \neg Q) \vdash \bot}}{\neg(P \wedge Q) \vdash \neg\neg(\neg P \vee \neg Q)}}{\neg(P \wedge Q) \vdash (\neg P \vee \neg Q)}$$

■(a) $(\exists x \forall y P(x,y)) \to (\forall x \exists y P(y,x))$

以下の証明図を $\vdash (\exists x \forall y P(x,y)) \to (\forall x \exists y P(y,x))$ の証明とする。

$$\dfrac{\dfrac{\dfrac{(\exists x \forall y P(x,y)) \vdash (\exists x \forall y P(x,y)) \quad \dfrac{\dfrac{\dfrac{}{(\exists x \forall y P(x,y)), \forall y P(s,y) \vdash \forall y P(s,y)}}{(\exists x \forall y P(x,y)), \forall y P(s,y) \vdash P(s,t)}}{(\exists x \forall y P(x,y)), \forall y P(s,y) \vdash \exists y P(y,t)}}{(\exists x \forall y P(x,y)) \vdash \exists y P(y,t)}}{(\exists x \forall y P(x,y)) \vdash (\forall x \exists y P(y,x))}}{\vdash (\exists x \forall y P(x,y)) \to (\forall x \exists y P(y,x))}$$

■(b) $(\forall x \forall y (P(x,y) \to P(y,x))) \to (\neg\exists x \exists y \neg(\neg P(x,y) \vee P(y,x)))$

以下の証明図を $\vdash (\forall x \forall y (P(x,y) \to P(y,x))) \to (\neg\exists x \exists y \neg(\neg P(x,y) \vee P(y,x)))$ の証明とする。

(大きいので、ここでは略記します。入らないので。各自補完してください。)

部分証明図 1

$$\dfrac{\dfrac{\forall x \forall y (P(x,y) \to P(y,x)), \Gamma \vdash \forall x \forall y (P(x,y) \to P(y,x))}{\forall x \forall y (P(x,y) \to P(y,x)), \Gamma \vdash \forall y P(s,y) \to P(y,s)}}{\forall x \forall y (P(x,y) \to P(y,x)), \Gamma \vdash P(s,t) \to P(t,s)}$$

以降 $\forall x \forall y (P(x,y) \to P(y,x)), \Gamma \vdash P(s,t) \to P(t,s)$ の前にはこの部分証明図があるものとする。

部分証明図 2 ($\forall x \forall y (P(x,y) \to P(y,x))$ 仮定下で)

$$\dfrac{\exists x \exists y \neg(\neg P(x,y) \vee P(y,x)) \quad \exists y \neg(\neg P(s,y) \vee P(y,s)) \quad \dfrac{\dfrac{\neg(\neg P(s,y) \vee P(y,s))}{\text{部分証明図 3}}}{\bot}}{\bot}$$

部分証明図 3 ($\forall x \forall y (P(x,y) \to P(y,x))$ 仮定下で)

$$\dfrac{\dfrac{P(s,t) \to P(t,s) \quad \dfrac{\dfrac{\neg(\neg P(s,t) \vee P(t,s)) \quad \dfrac{\neg P(s,t)}{\neg P(s,t) \vee P(t,s)}}{\bot}}{\neg\neg P(s,t)}}{P(s,t)} \quad \dfrac{\neg(\neg P(s,t) \vee P(t,s)) \quad \dfrac{P(t,s)}{\neg P(s,t) \vee P(t,s)}}{\bot}}{P(t,s)} \quad \dfrac{}{\neg P(t,s)}}{\bot}$$

全体

$$\dfrac{\dfrac{\dfrac{\text{(部分証明図 2)}}{\bot}}{\neg\exists x \exists y \neg(\neg P(x,y) \vee P(y,x))}}{(\forall x \forall y (P(x,y) \to P(y,x))) \to (\neg\exists x \exists y \neg(\neg P(x,y) \vee P(y,x)))}$$

にしてもこれ実際に試験の時書くとなると結構時間取られるお。他の問題解いたほうが点になるお

問題 2

(1)

まず、$k = 2$ の場合を説明する。

入力列二つの先頭の要素を比較し、小さい方の列の先頭を取り除いて出力列の末尾に追加する。この操作をどちらかの入力列が空になるまで繰り返し、最後に残った列を出力列の末尾に追加する。このアルゴリズムは二つの列の先頭にしか同時にアクセスしないため、列が巨大で一時記憶に収まらない場合でも動作することが出来る。

$k > 2$ の場合は、2 本ずつ $k = 2$ のアルゴリズムで処理すると、$\lceil \frac{k}{2} \rceil$ 本になり、これを繰り返せばよい。

擬似コードは以下のようになる。

```
function merge2(xs,ys)
  zs <- 空リスト
  while xs も ys も空でない do
    if xs の先頭が ys の先頭以下 then
      xs の先頭を zs の末尾に追加
      xs の先頭を取り除く
    else
      ys の先頭を zs の末尾に追加
      ys の先頭を取り除く
    end if
  end while
  xs を zs の末尾に追加
  ys を zs の末尾に追加
  return zs
end function
function merge(lists)
  while lists の長さが 2 以上 do
    lists2 <- 空リスト
    while lists の長さが 2 以上 do
      merge2(lists の先頭,lists の 2 番目) を lists2 の末尾に追加
      lists の先頭２つを取り除く
    end while
    lists を lists2 の末尾に追加
    lists <- lists2
  end while
  return lists
end funtion
```

(2)

各列をその長さ個の長さ 1 の列に分割した上で、(1) の merge アルゴリズムを適用すればよい。

```
function mergesort(lists)
  lists2 <- 空リスト
  for list in lists do
   for elem in list do
     elem のみからなる長さ 1 のリストを lists2 の末尾に追加
   end for
  end for
  return merge(lists2)
end function
```

(3)

まず最初に長さ 1 の列 kn 個に分割され、それに対して merge アルゴリズムが呼ばれる。各要素を含むリストが merge2 関数の引数になる回数は $O(\log kn)$ 回であり、merge2 において、二つのリストの各要素が二次記憶に読み書きされる回数はちょうど 1 回ずつである。よって、読み書きの回数は $O(kn \log kn)$ 回と見積もれる。

コメント

なぜ k 本もあるのかよく分からない。O 表記でいいのかは知らない。

(4)

この方法だと、長さ m の列 kn/m 本から merge アルゴリズムを開始できる。クイックソート部分は一次記憶を用いるので、ここでの二次記憶の読み書き回数は O(kn) 回である。merge 部分では、$O(\log kn/m)$ 段のマージが行われるので、ここでの二次記憶の読み書き回数は $O(kn \log(kn/m))$ 回である。よって全体では、$O(kn + kn \log kn/m) = O(kn \log kn - kn \log m)$ となり、$kn \log m$ 分読み書き回数が減少する。

問題 4

(1)

Process1, Process2, Process3 がそれぞれ 1 行目の lock まで実行された場合、つまり Process1 が mtxA の lock を、Process2 が mtxB の lock を、Process3 が mtxC の lock を各々取得した状態になると、どの Process も次の lock を取得できなくなり、Deadlock が発生する。

(2)

Process3 を以下のように書き換えればたぶん大丈夫。lock 取得の順番を逆にしただけです。

```
Process 3
1: mutex_lock(mtxA);
2: mutex_lock(mtxC);
   /* 資源 C と資源 A を
      排他利用する */
8: mutex_unlock(mtxC);
9: mutex_unlock(mtxA);
```

(3)

以下のような手法が考えられる

- 複数の資源にアクセスするためのミューテックスを用意する
- lock が取得できないときは既に取得している lock を全て開放する

(4)

とりあえず C 言語っぽく書いてみる。PIDLIST.find(p) は PIDLIST に p と一致するものがあったとき ture を返し、なかったとき false を返すものとする。また、while ループは高々プロセス数分しか回らないので必ず停止する。

```
1  #include <stdbool.h>
2
3  bool check_deadlock(PID p)
4  {
5          PID p1 = p;
6          MID m1;
7
8          do{
9                  PIDLIST.add(p1);
10                 if((m1 = waitFor(p1)) == 0) {
11                         return false;
12                 }
13                 if((p1 = used(m1)) == 0) {
14                         return false;
15                 }
16         } while(!PIDLIST.find(p1))
17
```

```
18          return true;
19  }
```

専門科目 II

問題 1

(1)

$l(u,w)+d(v,u)-d(v,w)<0$ となる点 v と枝 (u,w) が存在したと仮定する。

$d(v,w)>d(v,u)+l(u,w)$ より、v から w への最短路長より、v から u を経由して枝 (u,w) を用いる経路長の方が短いことになり矛盾。

よって任意の点 v と枝 (u,w) に対して $l(u,w)+d(v,u)-d(v,w)\geq 0$ が成り立つ。

(2)

二点 u,v を固定する。任意の u-v パス $v_1=u,v_2,\ldots,v_m=v$ に対して、その G' における長さは、

$$\sum_{i=1}^{m-1}(l(v_i,v_{i+1})+s(v_i)-s(v_{i+1}))=\sum_{i=1}^{m-1}l(v_i,v_{i+1})+s(u)-s(v)$$

となり、G における長さとの差は経路によらず定数 $s(u)-s(v)$ である。

よって、G と G' における二点間の最短路は一致する。

(3)

次のようなベルマンフォードのアルゴリズムを用いればよい。

1. 配列 d を、d(v)=0、v 以外の点 u に対し d(u)=∞と初期化する。
2. 各枝 (u,w) に対し、d(w) よりも d(u)+l(u,w) の方が小さければ、d(w) を d(u)+l(u,w) に更新する。
3. 2 で 1 度でも更新が起きた場合は 2 に戻る。
4. v 以外の各点 u に対して、2. における最後の更新時の枝 (w,u) からなる集合 E' を考えると、T=(V,E') は v を始点とした最短路木

このアルゴリズムの計算量は、最短路の頂点数はたかだか $|V|-1$ であることより、2 が最大で $|V|-1$ 回しか実行されないので、$O(|V||E|)$ である。

コメント

負の辺があるグラフに対してはダイクストラ法は使用できないので注意

(4)

各点 u に対し、$s(u)=d(v,u)$ と定義し、各枝 (u,v) の長さを $l'(u,v)=l(u,v)+s(u)-s(v)$ としたグラフ G' を考える。

$l(u,w)+s(u)-s(v)=l(u,w)+d(v,w)-d(v,w)\geq 0$ なので、G' は負の枝を含まない。

よって次のようなダイクストラのアルゴリズムが適用出来る。

1. 配列 d を d(v)=0,v 以外の点 u に対し d(u)=∞と初期化する。
2. 集合 U を空集合で初期化する。

3. U に含まれない点のうち、もっとも d の小さい点を一つ選んで u とし、U に u を追加する。

4. u から出ている各枝 (u,w) に対し、d(w) よりも d(u)+l'(u,w) の方が小さければ、d(w) を d(u)+l'(u,w) に更新する。

5. U が V と等しくなければ 3 に戻る。

6. v 以外の各点 u に対して、4. における最後の更新時の枝 (w,u) からなる集合 E' を考えると、T=(V,E') は v を始点とした最短路木

このアルゴリズムの計算量は、3 において全部の頂点を調べた場合には一回の反復の計算量が $O(|V|)$ で、3-4 の反復回数が $|V|$ 回であることより、$O(|V|^2)$ となる。

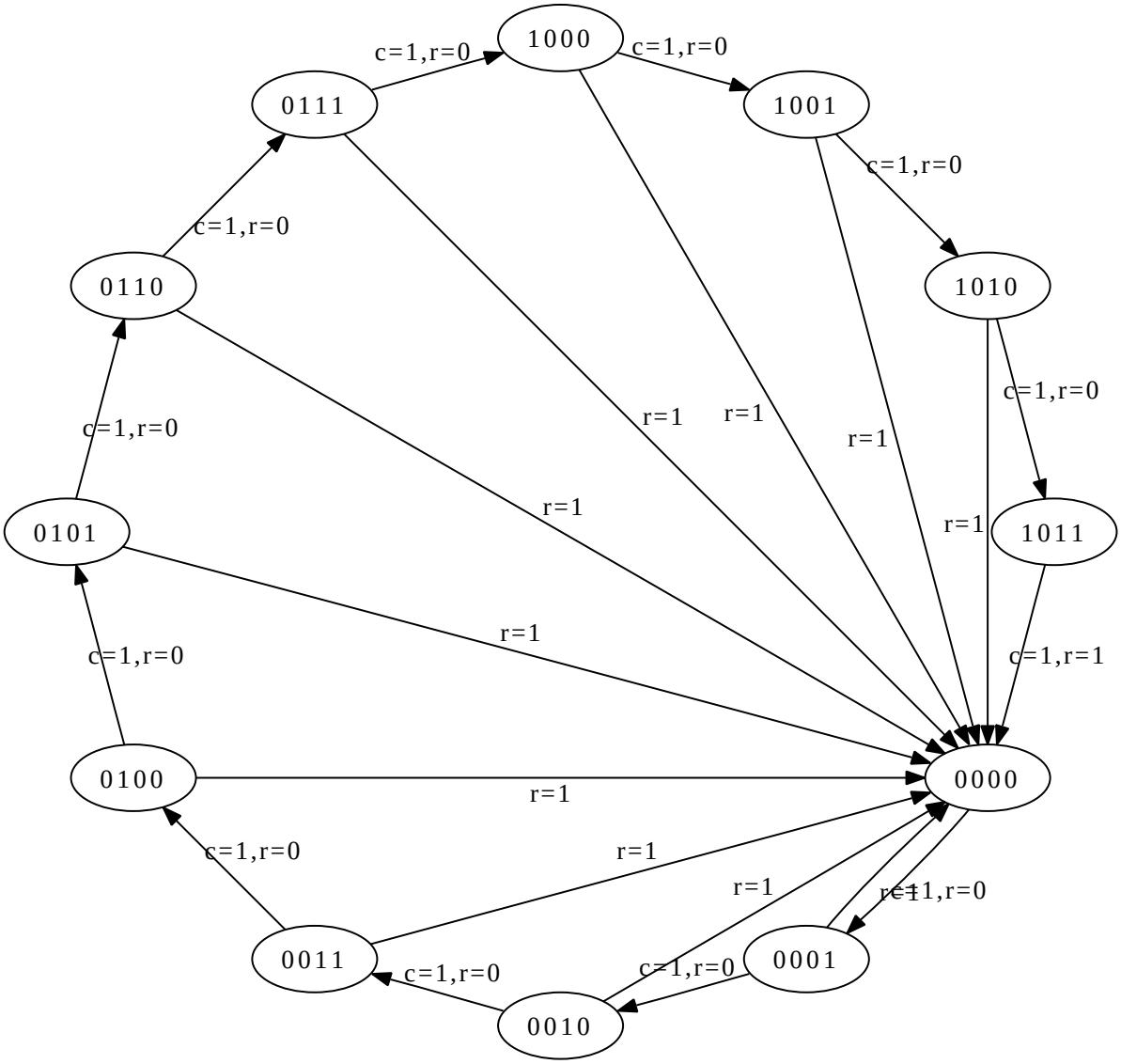
また、順位キューを用いて d の小さいものから取り出せるようにした場合には、4. において各枝について一度ずつしか調べないことより $O(|E| \log |V|)$ となる。

問題 4

(1)

状態遷移表

S	$c=1, r=0$	$r=1$
0000	0001	0000
0001	0010	0000
0010	0011	0000
0011	0100	0000
0100	0101	0000
0101	0110	0000
0110	0111	0000
0111	1000	0000
1000	1001	0000
1001	1010	0000
1010	1011	0000
1011	0000	0000



(2)

カウンタの値をカウンタ + カウンタ信号で毎クロック更新すればよい。カウンタの値が 11 でカウンタ信号が 1 ならリセットをかける。詳しくは小林先生の本を読もう。

(3)

カウント信号をカウンタが 15 の時 0 として扱えばよい。