

付録A プログラムリスト

1D-RPCA の可逆シミュレーション

```
1 procedure main()
2   //Number of transitions
3   int t_end = 3
4   //configuration at t=0
5   int t0_num = 12 int t0[12] = {0,0,1,0,1,0,0,0,0,1,1,0}
6   //transition rule
7   int rule[8][3] =
8     {{0,0,0},{0,0,1},{0,1,0},{0,1,1},{1,0,0},{1,0,1},{1,1,0},{1,1,1}}
9   stack conf
10
11   call init_config(conf, t0, t0_num)
12   call CA(t_end, conf, rule)
13 //initialize configuration at t=0
14 procedure init_config(stack sl, int t0[], int t0_num)
15   local int i = 0, int temp = 0
16   from i = t0_num loop
17     temp ^= t0[i - 1]
18     call mypush(temp, sl)
19     i -= 1
20   until i = 0
21   delocal int i = 0, int temp = 0
22
23 //extended push pop
24 procedure mypush(int x, stack s)
25   if !(empty(s) && x=0) then
26     push(x, s)
27   fi !empty(s)
28
29 procedure mypop(int x, stack s)
30   uncall mypush(x, s)
31
32 procedure rewind(stack sl, stack sr)
33   call mypush(0, sr) // the last l
34   from empty(sr) do
35     local int t = 0
36     call mypop(t,sl) call mypush(t,sr) //move R cell
37     call mypop(t,sl) call mypush(t,sr) //move C cell
38     call mypop(t,sl) call mypush(t,sr) //move L cell
39   delocal int t = 0
40   until empty(sl)
41
42 //simulation of cellular automaton
43 procedure CA(int t_end, stack conf, int rule[][] )
44   iterate int t = 1 to t_end
45     call gmap(conf, rule)
46   end
47
48 procedure gmap(stack sr, int rule[][] )
49   local stack sl=nil, int l=0, int c=0, int r=0, int next_r=0
50   call mypop(l, sr)
51   from empty(sl) && c=0 && r=0 && next_r=0 loop
```

```

52  call lmap(l, c, r, rule)
53  call mypush(l,s1) call mypush(c,s1) call mypush(r,s1)
54  call mypop(c,sr) call mypop(r,sr) call mypop(l,sr)
55  r <=> next_r
56  until empty(sr) && l=0 && c=0 && r=0 && next_r=0
57  call rewind(s1, sr)
58  delocal stack s1=nil,int l=0,int c=0,int r=0,int next_r=0
59
60  procedure lmap(int l, int c, int r, int rule[][] )
61  local int no = (l<<2) + (c<<1) + (r<<0)
62  l ^= ((no&0x4)>>2) ^ rule[no][0]
63  c ^= ((no&0x2)>>1) ^ rule[no][1]
64  r ^= ((no&0x1)>>0) ^ rule[no][2]
65  from l = rule[no][0] &&
66  c = rule[no][1] &&
67  r = rule[no][2] loop
68  no -= 1 // zero clear no
69  until no = 0
70  delocal int no = 0

```

2D-RPCA の可逆シミュレーション

```

1 procedure main()
2   int t_end = 3 // Number of transitions
3   int conf0[][] = {{0,12}, // initial configuration
4                   {3,0}}
5   int rule[16][4] = {{0,0,0,0}, {0,0,0,1},
6                     {0,0,1,0}, {1,1,0,0},
7                     {0,1,0,0}, {1,0,1,0},
8                     {1,0,0,1}, {1,1,0,1},
9                     {1,0,0,0}, {0,1,1,0},
10                    {0,1,0,1}, {1,1,1,0},
11                    {0,0,1,1}, {0,1,1,1},
12                    {1,0,1,1}, {1,1,1,1}}
13
14   call CA(t_end, conf0, rule)
15
16 // copy source s to target t
17 procedure copy(int s[][], int t[][])
18   iterate int i = 0 to size(s) - 1
19     iterate int j = 0 to size(s) - 1
20       t[i][j] ^= s[i][j]
21     end
22   end
23
24 procedure swapd(int s[][], int t[][], int d)
25   iterate int i = 0 to size(s) - 1
26     iterate int j = 0 to size(s) - 1
27       t[i+d][j+d] <=> s[i][j]
28     end
29   end
30
31 //simulation of cellular automaton
32 procedure CA(int t_end, int conf[][], int rule[][])
33   iterate int t = 1 to t_end
34     call gmap(conf, rule)
35   end
36
37 //transition of cell
38 procedure lmap(int t, int u, int l, int r, int d, int rule[][])
39   local int no = (l<<3) + (d<<2) + (r<<1) + (u<<0)
40   l ^= ((no & 0x8) >> 3) ^ rule[no][0]
41   d ^= ((no & 0x4) >> 2) ^ rule[no][1]
42   r ^= ((no & 0x2) >> 1) ^ rule[no][2]
43   u ^= ((no & 0x1) >> 0) ^ rule[no][3]
44   from l = rule[no][0] && // zero clear no
45     d = rule[no][1] &&
46     r = rule[no][2] &&
47     u = rule[no][3] loop
48     no -= 1
49   until no = 0
50   t ^= (l<<3) + (d<<2) + (r<<1) + (u<<0)
51   l ^= (t & 0x8) >> 3
52   d ^= (t & 0x4) >> 2
53   r ^= (t & 0x2) >> 1
54   u ^= (t & 0x1) >> 0
55   delocal int no = 0
56
57 procedure gmap(int sr[][], int rule[][])
58   local int sl[size(sr) + 2][size(sr) + 2] = {{0}}, int u = 0, int l =
59     0, int r = 0, int d = 0
60   call swapd(sr, sl, 1)
61   delocal int sr[size(sl) - 2][size(sl) - 2] = {{0}}
62   local int sr[size(sl)][size(sl)] = {{0}}
63   iterate int y = 0 to size(sl) - 1
64     iterate int x = 0 to size(sl) - 1

```

```

64         call get_ulrd(u, l, r, d, sl, x, y)
65         local int t = 0
66         call lmap(t, u, l, r, d, rule)
67         t <=> sr[x][y]
68         delocal int t = 0
69     end
70 end
71 delocal int sl[size(sr)][size(sr)] = {{0}}, int u = 0, int l = 0,
    int r = 0, int d = 0
72
73 procedure get_ulrd(int u, int l, int r, int d, int s[][], int x, int y)
74     call get_l(l, s, x, y+1)
75     call get_d(d, s, x-1, y)
76     call get_r(r, s, x, y-1)
77     call get_u(u, s, x+1, y)
78
79 procedure get_l(int l, int s[][], int x, int y)
80     if !(y = size(s)) then
81         l ^= (s[x][y] & 0x8) >> 3
82         s[x][y] ^= (l << 3)
83     fi !(y = size(s))
84
85 procedure get_d(int d, int s[][], int x, int y)
86     if !(x < 0) then
87         d ^= (s[x][y] & 0x4) >> 2
88         s[x][y] ^= (d << 2)
89     fi !(x < 0)
90
91 procedure get_r(int r, int s[][], int x, int y)
92     if !(y < 0) then
93         r ^= (s[x][y] & 0x2) >> 1
94         s[x][y] ^= (r << 1)
95     fi !(y < 0)
96
97 procedure get_u(int u, int s[][], int x, int y)
98     if !(x = size(s)) then
99         u ^= (s[x][y] & 0x1) >> 0
100        s[x][y] ^= (u << 0)
101    fi !(x = size(s))

```