LISP 실습 과제 : Cellular Automata

작성자 : 2019134006 김준섭

1. cellularAutomata1d
   1. 구성
      1. 문법 : (cellularAutomata1d [initialBoard] [time])
      2. 기능 : 초기 배열(initialBoard)로부터 정해진 규칙에 따라 time만큼의 시간 뒤의 상황 출력
      3. 입출력 :
         1. 입력 : 리스트의 길이는 제한 없음. 리스트의 양 끝은 0으로 취급. 초기 시간은 0으로 고려.
            1. 초기 상태를 표현한 리스트 initialBoard
            2. 세대를 의미하는 time, 초기는 0이며 세대가 증가할 수록 time은 1만큼 증가
         2. 출력 : time 만큼의 세대가 지난 후 리스트의 모습 반환

텍스트, 폰트, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* 1. 구현 방법
     1. cellularAutomata1d를 구현하기 위해 3개의 함수 buffer, scan, timeAction을 정의하였다.
        1. buffer :   
           append, cons를 이용, 입력받은 리스트의 양 끝에 0을 붙인다. 이는 문제의 조건상 리스트의 양 끝은 0으로 한다는 가정을 구현하기 위해서이다.
        2. scan :   
           car, cadr, caddr을 이용해 리스트의 맨 앞의 3개의 원소를 추출, cond, and를 이용해 문제상황으로 주어진 규칙에 따라 3개의 원소의 배열에 대응하는 숫자를 출력한다.
        3. timeAction :   
           if문과 cdr을 이용, 재귀적인 방법으로 입력받은 리스트 각 원소에 대해 scan을 수행하여 얻은 값을 순서대로 붙여 리스트로 만든다.
     2. cellularAutomata1d : if문을 이용, time = 0일 때 자기 자신을 반환함을 base로 삼아 recursion을 수행하여, 입력한 리스트를 규칙에 따라 변화시킨 list를 출력하는 과정을 time만큼 반복한다.
  2. 코드

1. (defun cellularAutomata1d (initialBoard time)

1. (if (eq time 0)
2. initialBoard
3. (cellularAutomata1d (timeAction (buffer initialBoard)) (- time 1))
4. )
5. )
6. (defun timeAction (board)
7. (if (eq (length board) 2)
8. nil
9. (cons
10. (scan board) (timeAction (cdr board))
11. )
12. )
13. )
14. (defun buffer (board)
15. (append (cons 0 board) '(0))
16. )
17. (defun scan (board)
18. (cond
19. ((and (eq (car board) 0) (eq (cadr board) 0) (eq (caddr board) 0)) 0)
20. ((and (eq (car board) 0) (eq (cadr board) 0) (eq (caddr board) 1)) 1)
21. ((and (eq (car board) 0) (eq (cadr board) 1) (eq (caddr board) 0)) 1)
22. ((and (eq (car board) 0) (eq (cadr board) 1) (eq (caddr board) 1)) 0)
23. ((and (eq (car board) 1) (eq (cadr board) 0) (eq (caddr board) 0)) 1)
24. ((and (eq (car board) 1) (eq (cadr board) 0) (eq (caddr board) 1)) 1)
25. ((and (eq (car board) 1) (eq (cadr board) 1) (eq (caddr board) 0)) 0)
26. ((and (eq (car board) 1) (eq (cadr board) 1) (eq (caddr board) 1)) 0)
27. )
28. )
29. cellularAutomata2d
    1. 구성
       1. 문법 : (cellularAutomata2d [initialBoard] [time])
       2. 기능 : 초기 2차원 판(initialBoard)에서부터 주어진 세대(time)만큼 지난 후의 판의 모습을 출력
       3. 입출력 :
          1. 입력 : 리스트는 2차원 리스트이며 리스트의 크기는 제한 없음, 리스트의 양 끝은 0으로 취급, 초기 시간은 0으로 고려
             1. 2d판이 표현된 리스트 initialBoard
             2. 세대를 의미하는 time, 초기는 0이며 세대가 증가할 수록 time은 1만큼 증가
          2. 출력 : 주어진 시간이 지난 후 보드 판의 문자열로 리턴

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

* 1. 구현 방법
     1. cellularAutomata2d를 구현하기 위해 6개의 함수 timeAction2, scan2, copy, Arrayidentify, list2arr, printer를 정의하였다.
        1. timeAction2 :  
           recursion을 통해 각 셀별 scan2를 통해 얻어낸 값들을 모아 array로 만든다. copy를 통해 구해낸 결과를 원래 보드에 복사한다.
        2. scan2 :   
           if문을 활용하여 입력된 board의 rr번째 행, cc번째 열에 해당하는 셀과 이웃한 8개의 셀에서 값이 1인 이웃의 개수와 자기 자신의 셀을 비교하여 새롭게 그 자리에 들어갈 값을 출력한다. 이 과정에서 if를 이용한 recursion을 통해 반복을 수행한다.
        3. copy :   
           recursion을 활용해 original의 값을 newb로 복사
        4. Arrayidentify :   
           and, arrayp, array-rank를 이용해 입력값이 2차원 배열인지 여부를 출력한다.
        5. list2arr :  
           if를 이용, Arrayidentify로부터 입력값이 2차원 array인지 여부를 판단하고, 아니라면 입력받은 list를 2차원 array로 바꾸어 출력한다.
        6. printer :  
           cond, concatenate, format을 이용하여 reursion을 통해 array로 입력받은 값을 행렬처럼 표시하도록 만든다.
     2. cellularAutomata2d : 1d에서와 마찬가지로 if문을 이용, time = 0일 때 자기 자신을 반환함을 base로 삼아 recursion을 수행하여, 입력한 리스트를 규칙에 따라 변화시킨 list를 출력하는 과정을 time만큼 반복한다.
  2. 코드

1. (defun cellularAutomata2d (initialBoard time)

1. (let ((convertedArr (list2arr initialBoard)))
2. (if (eq time 0)
3. (printer convertedArr 0 0 0)
4. (progn
5. (timeAction2 convertedArr)
6. (cellularAutomata2d convertedArr (- time 1))
7. )
8. )
9. )
10. )
11. (defun timeAction2 (board)
12. (let ((row (array-dimension board 0))(col (array-dimension board 1)))
13. (let ((newBoard (make-array (list row col))))
14. (labels (
15. (fill\_new (rr cc)
16. (when (< rr row)
17. (when (< cc col)
18. (progn
19. (setf (aref newBoard rr cc) (scan2 board rr cc 0 1))
20. (fill\_new rr (+ cc 1)))
21. )
22. (fill\_new (+ rr 1) cc)
23. )
24. )
25. )
26. (fill\_new 0 0)
27. (copy newBoard board 0 0)
28. )
29. )
30. )
31. )
32. (defun scan2 (board rr cc counter neighbor\_no)
33. (if (= (aref board rr cc) 0)
34. (cond
35. ((= neighbor\_no 1)
36. (if (and (>= rr 1) (>= cc 1) (= (aref board (- rr 1) (- cc 1)) 1))
37. (progn
38. (setq counter (+ counter 1))
39. (scan2 board rr cc counter (+ neighbor\_no 1))
40. )
41. )
42. (scan2 board rr cc counter (+ neighbor\_no 1))
43. )
44. ((= neighbor\_no 2)
45. (if (and (>= cc 1) (= (aref board rr (- cc 1)) 1))
46. (progn
47. (setq counter (+ counter 1))
48. (scan2 board rr cc counter (+ neighbor\_no 1))
49. )
50. )
51. (scan2 board rr cc counter (+ neighbor\_no 1))
52. )
53. ((= neighbor\_no 3)
54. (if (and (< rr (- (array-dimension board 0) 1))(>= cc 1) (= (aref board (+ rr 1) (- cc 1)) 1))
55. (progn
56. (setq counter (+ counter 1))
57. (scan2 board rr cc counter (+ neighbor\_no 1))
58. )
59. )
60. (scan2 board rr cc counter (+ neighbor\_no 1))
61. )
62. ((= neighbor\_no 4)
63. (if (and (>= rr 1) (= (aref board (- rr 1) cc) 1))
64. (progn
65. (setq counter (+ counter 1))
66. (scan2 board rr cc counter (+ neighbor\_no 1))
67. )
68. )
69. (scan2 board rr cc counter (+ neighbor\_no 1))
70. )
71. ((= neighbor\_no 5)
72. (if (and (< rr (- (array-dimension board 0) 1))(= (aref board (+ rr 1) cc) 1))
73. (progn
74. (setq counter (+ counter 1))
75. (scan2 board rr cc counter (+ neighbor\_no 1))
76. )
77. )
78. (scan2 board rr cc counter (+ neighbor\_no 1))
79. )
80. ((= neighbor\_no 6)
81. (if (and (>= rr 1)(< cc (- (array-dimension board 1) 1))(= (aref board (- rr 1) (+ cc 1)) 1))
82. (progn
83. (setq counter (+ counter 1))
84. (scan2 board rr cc counter (+ neighbor\_no 1))
85. )
86. )
87. (scan2 board rr cc counter (+ neighbor\_no 1))
88. )
89. ((= neighbor\_no 7)
90. (if (and (< cc (- (array-dimension board 1) 1))(= (aref board rr (+ cc 1)) 1))
91. (progn
92. (setq counter (+ counter 1))
93. (scan2 board rr cc counter (+ neighbor\_no 1))
94. )
95. )
96. (scan2 board rr cc counter (+ neighbor\_no 1)))
97. ((= neighbor\_no 8)
98. (if (and (< rr (- (array-dimension board 0) 1)) (< cc (- (array-dimension board 1) 1)))
99. (if (= (aref board (+ rr 1) (+ cc 1)) 1)
100. (progn
101. (setq counter (+ counter 1))
102. (if (= counter 3) 1 0)
103. )
104. (if (= counter 3) 1 0)
105. )
106. (if (= counter 3) 1 0)
107. )
108. )
109. )
110. (cond
111. ((= neighbor\_no 1)
112. (if (and (>= rr 1) (>= cc 1) (= (aref board (- rr 1) (- cc 1)) 1))
113. (progn (setq counter (+ counter 1)) (scan2 board rr cc counter (+ neighbor\_no 1)))
114. )
115. (scan2 board rr cc counter (+ neighbor\_no 1))
116. )
117. ((= neighbor\_no 2)
118. (if (and (>= cc 1) (= (aref board rr (- cc 1)) 1))
119. (progn (setq counter (+ counter 1)) (scan2 board rr cc counter (+ neighbor\_no 1)))
120. )
121. (scan2 board rr cc counter (+ neighbor\_no 1))
122. )
123. ((= neighbor\_no 3)
124. (if (and (< rr (- (array-dimension board 0) 1))(>= cc 1) (= (aref board (+ rr 1) (- cc 1)) 1))
125. (progn (setq counter (+ counter 1))(scan2 board rr cc counter (+ neighbor\_no 1)))
126. )
127. (scan2 board rr cc counter (+ neighbor\_no 1))
128. )
129. ((= neighbor\_no 4)
130. (if (and (>= rr 1) (= (aref board (- rr 1) cc) 1))
131. (progn (setq counter (+ counter 1))(scan2 board rr cc counter (+ neighbor\_no 1)))
132. )
133. (scan2 board rr cc counter (+ neighbor\_no 1))
134. )
135. ((= neighbor\_no 5)
136. (if (and (< rr (- (array-dimension board 0) 1))(= (aref board (+ rr 1) cc) 1))
137. (progn (setq counter (+ counter 1))(scan2 board rr cc counter (+ neighbor\_no 1)))
138. )
139. (scan2 board rr cc counter (+ neighbor\_no 1))
140. )
141. ((= neighbor\_no 6)
142. (if (and (>= rr 1)(< cc (- (array-dimension board 1) 1))(= (aref board (- rr 1) (+ cc 1)) 1))
143. (progn (setq counter (+ counter 1))(scan2 board rr cc counter (+ neighbor\_no 1)))
144. )
145. (scan2 board rr cc counter (+ neighbor\_no 1))
146. )
147. ((= neighbor\_no 7)
148. (if (and (< cc (- (array-dimension board 1) 1))(= (aref board rr (+ cc 1)) 1))
149. (progn (setq counter (+ counter 1))(scan2 board rr cc counter (+ neighbor\_no 1)))
150. )
151. (scan2 board rr cc counter (+ neighbor\_no 1)))
152. ((= neighbor\_no 8)
153. (if (and (< rr (- (array-dimension board 0) 1)) (< cc (- (array-dimension board 1) 1)))
154. (if (= (aref board (+ rr 1) (+ cc 1)) 1)
155. (progn (setq counter (+ counter 1))
156. (if (or (= counter 2) (= counter 3)) 1 0))
157. (if (or (= counter 2) (= counter 3)) 1 0)
158. )
159. (if (or (= counter 2) (= counter 3)) 1 0)
160. )
161. )
162. )
163. )
164. )
165. (defun copy (original newb rr cc)
166. (let ((row (array-dimension original 0)) (col (array-dimension original 1)))
167. (when (< rr row)
168. (when (< cc col)
169. (progn (setf (aref newb rr cc) (aref original rr cc))
170. (copy original newb rr (+ cc 1))
171. )
172. )
173. (copy original newb (+ rr 1) cc)
174. )
175. )
176. )
177. (defun Arrayidentify (arr)
178. (and (arrayp arr)(= (array-rank arr) 2))
179. )
180. (defun list2arr (lst)
181. (if (Arrayidentify lst)
182. lst
183. (let\* ((row (length lst))(col (if (zerop row) 0 (length (first lst)))))
184. (make-array (list row col):initial-contents lst)
185. )
186. )
187. )
188. (defun printer (board rr cc nextline)
189. (let ((row (array-dimension board 0))(col (array-dimension board 1)))
190. (cond
191. ((= nextline 0) (concatenate 'string (format nil "~%") (printer board rr cc 1)))
192. ((= rr row) "")
193. ((= cc col)(concatenate 'string (format nil "~%") (printer board (+ rr 1) 0 1)))
195. (t (concatenate 'string (format nil "~a " (aref board rr cc))(printer board rr (+ cc 1) 1)))
196. )
197. )
198. )

3. 전체 코드 (“2019134006.lisp”)

1. (defun cellularAutomata1d (initialBoard time)

2. (if (eq time 0)

1. initialBoard
2. (cellularAutomata1d (timeAction (buffer initialBoard)) (- time 1))
3. )
4. )
5. (defun timeAction (board)
6. (if (eq (length board) 2)
7. nil
8. (cons
9. (scan board) (timeAction (cdr board))
10. )
11. )
12. )
13. (defun buffer (board)
14. (append (cons 0 board) '(0))
15. )
16. (defun scan (board)
17. (cond
18. ((and (eq (car board) 0) (eq (cadr board) 0) (eq (caddr board) 0)) 0)
19. ((and (eq (car board) 0) (eq (cadr board) 0) (eq (caddr board) 1)) 1)
20. ((and (eq (car board) 0) (eq (cadr board) 1) (eq (caddr board) 0)) 1)
21. ((and (eq (car board) 0) (eq (cadr board) 1) (eq (caddr board) 1)) 0)
22. ((and (eq (car board) 1) (eq (cadr board) 0) (eq (caddr board) 0)) 1)
23. ((and (eq (car board) 1) (eq (cadr board) 0) (eq (caddr board) 1)) 1)
24. ((and (eq (car board) 1) (eq (cadr board) 1) (eq (caddr board) 0)) 0)
25. ((and (eq (car board) 1) (eq (cadr board) 1) (eq (caddr board) 1)) 0)
26. )
27. )
28. ;-------------------------------------------------------------------------------------
29. (defun cellularAutomata2d (initialBoard time)
30. (let ((convertedArr (list2arr initialBoard)))
31. (if (eq time 0)
32. (printer convertedArr 0 0 0)
33. (progn
34. (timeAction2 convertedArr)
35. (cellularAutomata2d convertedArr (- time 1))
36. )
37. )
38. )
39. )
40. (defun timeAction2 (board)
41. (let ((row (array-dimension board 0))(col (array-dimension board 1)))
42. (let ((newBoard (make-array (list row col))))
43. (labels (
44. (fill\_new (rr cc)
45. (when (< rr row)
46. (when (< cc col)
47. (progn
48. (setf (aref newBoard rr cc) (scan2 board rr cc 0 1))
49. (fill\_new rr (+ cc 1)))
50. )
51. (fill\_new (+ rr 1) cc)
52. )
53. )
54. )
55. (fill\_new 0 0)
56. (copy newBoard board 0 0)
57. )
58. )
59. )
60. )
61. (defun scan2 (board rr cc counter neighbor\_no)
62. (if (= (aref board rr cc) 0)
63. (cond
64. ((= neighbor\_no 1)
65. (if (and (>= rr 1) (>= cc 1) (= (aref board (- rr 1) (- cc 1)) 1))
66. (progn
67. (setq counter (+ counter 1))
68. (scan2 board rr cc counter (+ neighbor\_no 1))
69. )
70. )
71. (scan2 board rr cc counter (+ neighbor\_no 1))
72. )
73. ((= neighbor\_no 2)
74. (if (and (>= cc 1) (= (aref board rr (- cc 1)) 1))
75. (progn
76. (setq counter (+ counter 1))
77. (scan2 board rr cc counter (+ neighbor\_no 1))
78. )
79. )
80. (scan2 board rr cc counter (+ neighbor\_no 1))
81. )
82. ((= neighbor\_no 3)
83. (if (and (< rr (- (array-dimension board 0) 1))(>= cc 1) (= (aref board (+ rr 1) (- cc 1)) 1))
84. (progn
85. (setq counter (+ counter 1))
86. (scan2 board rr cc counter (+ neighbor\_no 1))
87. )
88. )
89. (scan2 board rr cc counter (+ neighbor\_no 1))
90. )
91. ((= neighbor\_no 4)
92. (if (and (>= rr 1) (= (aref board (- rr 1) cc) 1))
93. (progn
94. (setq counter (+ counter 1))
95. (scan2 board rr cc counter (+ neighbor\_no 1))
96. )
97. )
98. (scan2 board rr cc counter (+ neighbor\_no 1))
99. )
100. ((= neighbor\_no 5)
101. (if (and (< rr (- (array-dimension board 0) 1))(= (aref board (+ rr 1) cc) 1))
102. (progn
103. (setq counter (+ counter 1))
104. (scan2 board rr cc counter (+ neighbor\_no 1))
105. )
106. )
107. (scan2 board rr cc counter (+ neighbor\_no 1))
108. )
109. ((= neighbor\_no 6)
110. (if (and (>= rr 1)(< cc (- (array-dimension board 1) 1))(= (aref board (- rr 1) (+ cc 1)) 1))
111. (progn
112. (setq counter (+ counter 1))
113. (scan2 board rr cc counter (+ neighbor\_no 1))
114. )
115. )
116. (scan2 board rr cc counter (+ neighbor\_no 1))
117. )
118. ((= neighbor\_no 7)
119. (if (and (< cc (- (array-dimension board 1) 1))(= (aref board rr (+ cc 1)) 1))
120. (progn
121. (setq counter (+ counter 1))
122. (scan2 board rr cc counter (+ neighbor\_no 1))
123. )
124. )
125. (scan2 board rr cc counter (+ neighbor\_no 1)))
126. ((= neighbor\_no 8)
127. (if (and (< rr (- (array-dimension board 0) 1)) (< cc (- (array-dimension board 1) 1)))
128. (if (= (aref board (+ rr 1) (+ cc 1)) 1)
129. (progn
130. (setq counter (+ counter 1))
131. (if (= counter 3) 1 0)
132. )
133. (if (= counter 3) 1 0)
134. )
135. (if (= counter 3) 1 0)
136. )
137. )
138. )
139. (cond
140. ((= neighbor\_no 1)
141. (if (and (>= rr 1) (>= cc 1) (= (aref board (- rr 1) (- cc 1)) 1))
142. (progn (setq counter (+ counter 1)) (scan2 board rr cc counter (+ neighbor\_no 1)))
143. )
144. (scan2 board rr cc counter (+ neighbor\_no 1))
145. )
146. ((= neighbor\_no 2)
147. (if (and (>= cc 1) (= (aref board rr (- cc 1)) 1))
148. (progn (setq counter (+ counter 1)) (scan2 board rr cc counter (+ neighbor\_no 1)))
149. )
150. (scan2 board rr cc counter (+ neighbor\_no 1))
151. )
152. ((= neighbor\_no 3)
153. (if (and (< rr (- (array-dimension board 0) 1))(>= cc 1) (= (aref board (+ rr 1) (- cc 1)) 1))
154. (progn (setq counter (+ counter 1))(scan2 board rr cc counter (+ neighbor\_no 1)))
155. )
156. (scan2 board rr cc counter (+ neighbor\_no 1))
157. )
158. ((= neighbor\_no 4)
159. (if (and (>= rr 1) (= (aref board (- rr 1) cc) 1))
160. (progn (setq counter (+ counter 1))(scan2 board rr cc counter (+ neighbor\_no 1)))
161. )
162. (scan2 board rr cc counter (+ neighbor\_no 1))
163. )
164. ((= neighbor\_no 5)
165. (if (and (< rr (- (array-dimension board 0) 1))(= (aref board (+ rr 1) cc) 1))
166. (progn (setq counter (+ counter 1))(scan2 board rr cc counter (+ neighbor\_no 1)))
167. )
168. (scan2 board rr cc counter (+ neighbor\_no 1))
169. )
170. ((= neighbor\_no 6)
171. (if (and (>= rr 1)(< cc (- (array-dimension board 1) 1))(= (aref board (- rr 1) (+ cc 1)) 1))
172. (progn (setq counter (+ counter 1))(scan2 board rr cc counter (+ neighbor\_no 1)))
173. )
174. (scan2 board rr cc counter (+ neighbor\_no 1))
175. )
176. ((= neighbor\_no 7)
177. (if (and (< cc (- (array-dimension board 1) 1))(= (aref board rr (+ cc 1)) 1))
178. (progn (setq counter (+ counter 1))(scan2 board rr cc counter (+ neighbor\_no 1)))
179. )
180. (scan2 board rr cc counter (+ neighbor\_no 1)))
181. ((= neighbor\_no 8)
182. (if (and (< rr (- (array-dimension board 0) 1)) (< cc (- (array-dimension board 1) 1)))
183. (if (= (aref board (+ rr 1) (+ cc 1)) 1)
184. (progn (setq counter (+ counter 1))
185. (if (or (= counter 2) (= counter 3)) 1 0))
186. (if (or (= counter 2) (= counter 3)) 1 0)
187. )
188. (if (or (= counter 2) (= counter 3)) 1 0)
189. )
190. )
191. )
192. )
193. )
194. (defun copy (original newb rr cc)
195. (let ((row (array-dimension original 0)) (col (array-dimension original 1)))
196. (when (< rr row)
197. (when (< cc col)
198. (progn (setf (aref newb rr cc) (aref original rr cc))
199. (copy original newb rr (+ cc 1))
200. )
201. )
202. (copy original newb (+ rr 1) cc)
203. )
204. )
205. )
206. (defun Arrayidentify (arr)
207. (and (arrayp arr)(= (array-rank arr) 2))
208. )
209. (defun list2arr (lst)
210. (if (Arrayidentify lst)
211. lst
212. (let\* ((row (length lst))(col (if (zerop row) 0 (length (first lst)))))
213. (make-array (list row col):initial-contents lst)
214. )
215. )
216. )
217. (defun printer (board rr cc nextline)
218. (let ((row (array-dimension board 0))(col (array-dimension board 1)))
219. (cond
220. ((= nextline 0) (concatenate 'string (format nil "~%") (printer board rr cc 1)))
221. ((= rr row) "")
222. ((= cc col)(concatenate 'string (format nil "~%") (printer board (+ rr 1) 0 1)))
224. (t (concatenate 'string (format nil "~a " (aref board rr cc))(printer board rr (+ cc 1) 1)))
225. )
226. )
227. )
228. 실험  
     작성한 코드가 잘 동작하는지 확인하는 작업을 수행하였다.
     1. cellularAutomata1d
        1. (cellularAutomata1d '(0 1 1 0) 1)
           1. 결과 : (1 0 0 1)



* + 1. (cellularAutomata1d '(0 0 0 0 0 0 0 1 0 0 0 0 0 0 0) 0)
       1. 결과 : (0 0 0 0 0 0 0 1 0 0 0 0 0 0 0)



* + 1. (cellularAutomata1d '(0 0 0 0 0 0 0 1 0 0 0 0 0 0 0) 1)
       1. 결과 : (0 0 0 0 0 0 1 1 1 0 0 0 0 0 0)



* + 1. (cellularAutomata1d '(0 0 0 0 0 0 0 1 0 0 0 0 0 0 0) 2)
       1. 결과 : (0 0 0 0 0 1 0 0 0 1 0 0 0 0 0)



* + 1. (cellularAutomata1d '(0 0 0 0 0 0 0 1 0 0 0 0 0 0 0) 3)
       1. 결과 : (0 0 0 0 1 1 1 0 1 1 1 0 0 0 0)



* + 1. (cellularAutomata1d '(0 0 0 0 0 0 0 1 0 0 0 0 0 0 0) 4)
       1. 결과 : (0 0 0 1 0 0 0 1 0 0 0 1 0 0 0)



* + 1. (cellularAutomata1d '(0 0 0 0 0 0 0 1 0 0 0 0 0 0 0) 5)
       1. 결과 : (0 0 1 1 1 0 1 1 1 0 1 1 1 0 0)



* + 1. (cellularAutomata1d '(0 0 0 0 0 0 0 1 0 0 0 0 0 0 0) 6)
       1. 결과 : (0 1 0 0 0 1 0 0 0 1 0 0 0 1 0)



실험결과, 강의자료의 예제와 잘 일치하는 것을 확인할 수 있다.

|  |  |
| --- | --- |
| Time | Output |
| 0 | (0 0 0 0 0 0 0 1 0 0 0 0 0 0 0) |
| 1 | (0 0 0 0 0 0 1 1 1 0 0 0 0 0 0) |
| 2 | (0 0 0 0 0 1 0 0 0 1 0 0 0 0 0) |
| 3 | (0 0 0 0 1 1 1 0 1 1 1 0 0 0 0) |
| 4 | (0 0 0 1 0 0 0 1 0 0 0 1 0 0 0) |
| 5 | (0 0 1 1 1 0 1 1 1 0 1 1 1 0 0) |
| 6 | (0 1 0 0 0 1 0 0 0 1 0 0 0 1 0) |

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

* 1. cellularAutomata2d
     1. (cellularAutomata2d '((0 0 0 0 0 0) (0 0 0 0 0 0) (0 0 0 1 0 0) (0 1 0 1 0 0) (0 0 1 1 0 0) (0 0 0 0 0 0)) 0)
        1. 결과:

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

* + 1. (cellularAutomata2d '((0 0 0 0 0 0) (0 0 0 0 0 0) (0 0 0 1 0 0) (0 1 0 1 0 0) (0 0 1 1 0 0) (0 0 0 0 0 0)) 1)
       1. 결과 :

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

* + 1. (cellularAutomata2d '((0 0 0 0 0 0) (0 0 0 0 0 0) (0 0 0 1 0 0) (0 1 0 1 0 0) (0 0 1 1 0 0) (0 0 0 0 0 0)) 2)
       1. 결과 :

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명



* + 1. (cellularAutomata2d '((0 0 0 0 0 0) (0 0 0 0 0 0) (0 0 0 1 0 0) (0 1 0 1 0 0) (0 0 1 1 0 0) (0 0 0 0 0 0)) 3)
       1. 결과 :

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

위의 강의자료와 잘 일치하는 것을 확인할 수 있다.

* + 1. (cellularAutomata2d '((0 0 0 0 0 0) (0 0 0 0 0 0) (0 0 0 1 0 0) (0 1 0 1 0 0) (0 0 1 1 0 0) (0 0 0 0 0 0)) 4)
       1. 결과 :

텍스트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

결과를 자세히 살펴보면, time = 0일 때와 time = 4일 때는 1의 배치가 우측 하단으로 평행이동한 것과 일치하는 것을 관찰할 수 있다. 위 배치는 즉 전진하는 진동자로, 우주선, 특히 ‘글라이더’라고 불린다.

* 1. cellularAutomata2d를 이용한 콘웨이의 생명게임 패턴 구현  
     2차원 세포자동자의 셀의 생존 및 탄생 규칙은 콘웨이의 생명게임과 일치하므로, 콘웨이의 생명게임에서 유명한 패턴 몇가지를 구현해보았다.
     1. 글라이더 (전진하는 진동자, 주기 : 4)

스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

* + 1. 막대기 (진동자, 주기 = 2)

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

* + 1. 두꺼비 (진동자, 주기 = 2)

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

* + 1. 비콘 (진동자, 주기 = 2)

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + 1. Lightweight spaceship, LWSS (전진하는 진동자, 주기 : 4)

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

1. 소감 및 고찰
   1. cellularAutomata1d를 구현할 때에는 변수 선언 및 반복문 사용 없이 구현할 수 있었다. 이를 통해 실제로 common lisp를 이용하면 변수선언, 반복문 없이도 recursion만을 이용해 문제상황을 해결할 수 있음을 확인할 수 있었다, 다만 cellularAutomata2d를 구현할 때에는 변수선언 없이 구현하기에는 어려움을 느껴 불가피하게 변수선언을 도입하여 구현하였다. 이에 향후에는 lisp의 취지에 맞게 변수선언 및 반복문 사용을 사용하지 않고 프로그램을 작성할 수 있을만큼의 역량을 길러야 할 필요성을 느꼈다.
   2. 2차원 세포 자동자를 이용해 콘웨이의 생명게임의 패턴을 실제로 구현해보며 프로그램이 잘 작성되었는지 확인함과 동시에 세포자동자의 다양한 패턴을 알아보는 흥미로운 경험이 되었다.