

## 演習課題3

提出日：2022年7月27日

所属：知能システム科学コース

名前：織田 康太郎

1. トランザクション T、S、U に対して、以下のスケジュールで実行されたとする。これらのスケジュールが競合直列化可能か否か判定せよ。

(1)  $R_T(x)R_S(x)R_U(x)R_T(y)R_S(y)W_U(x)W_T(x)W_S(z)$

(2)  $R_T(x)R_S(y)R_U(z)R_T(y)R_S(x)R_U(y)W_T(z)W_S(z)W_U(z)$

(3)  $R_T(x)R_S(x)R_U(x)R_T(y)R_S(z)W_T(y)R_U(z)W_S(z)W_T(z)W_U(x)$

以下に示す db3.py を用いてスケジュールが競合直列化可能であるかの判定を行った。

また、競合直列化可能か正しく判定できているかを確認するために、オリジナルのスケジュールを作成して、それについても判定を行った。

オリジナルのスケジュール

$R_S(x)R_S(y)R_T(x)R_T(z)W_S(y)R_U(y)R_U(z)W_U(y)W_T(x)W_T(z)$

ソースコード 1: db3.py

```

1 import itertools
2
3 # DB に対する操作のこと、 Rt(x) だと
4 # R: 操作の内容 ( R/W のいずれか)
5 # t: どのトランザクションに属する操作か
6 # x: 操作の対象となるデータ
7 # position は元の課題に記載されているスケジュールでの位置
8 class Sousa:
9     position = 0
10
11     def __init__(self, sousa, transaction, data):
12         self.sousa = sousa
13         self.transaction = transaction
14         self.data = data
15         self.position = Sousa.position
16         Sousa.position += 1
17
18 # 描画準備
19 def print_sousa(sousa):
20     return sousa.sousa.upper() + sousa.transaction + '(' + sousa.data + ')'
21
22 # 描画
23 def print_schedule(schedule):
24     N = len(schedule)
25     for i in range(N):
26         print(print_sousa(schedule[i]), end=' ')

```

```

27     print()
28
29 # 競合しない操作のペアのリストを作成
30 def check_conflict(schedule):
31     N = len(schedule)
32     a, b = 0, 1
33     ans = []
34     for i in range(N*(N-1)//2):
35         if not check(schedule[a], schedule[b]):
36             ans.append([schedule[a], schedule[b]])
37         b += 1
38         if b == N:
39             a += 1
40             b = a + 1
41     return ans
42
43 # 競合の確認
44 # #04.pdf の 28, 29 ページを元にして競合の確認を行う。
45 def check(a, b):
46     if a.sousa == 'r' and b.sousa == 'r':
47         return False
48     if a.data != b.data:
49         if a.sousa != b.sousa and a.transaction == b.transaction:
50             return True
51         return False
52     return True
53
54 # 目的となる直列スケジュールの作成
55 # ここで直列スケジュールはトランザクションごとに操作をまとめたものをつなげたものとする。
56 # そのため、今回のようにトランザクションが
57 # 3つある場合、直列化可能スケジュールは 3! 個つまり 6 個考えられる。
58 def make_goal(schedule):
59     transaction_T = []
60     transaction_S = []
61     transaction_U = []
62     for i in schedule:
63         if i.transaction == 't':
64             transaction_T.append(i)
65         elif i.transaction == 's':
66             transaction_S.append(i)
67         else:
68             transaction_U.append(i)
69     transactions = [transaction_S, transaction_T, transaction_U]
70     goals = []
71     for i in list(itertools.permutations(transactions)):
72         serial_schedule = []
73         for j in i:
74             serial_schedule += j
75         goals.append(serial_schedule)
76     return goals

```

```

77
78 # スケジュールを構成するデータベースに対する操作を並べ替えて、
79 # 新たなスケジュールを構築する。
80 # そのとき、操作の順番が入れ替わっているものについては、競合を確認する。
81 # 全ての操作について競合しないスケジュールを返す。
82 def search_schedule(schedule, nconflicts):
83     schedules = list(itertools.permutations(schedule))
84     enable_schedules = []
85     for s in schedules:
86         for i in range(1, len(s)):
87             for j in range(i):
88                 if s[j].position > s[i].position:
89                     if [s[i], s[j]] not in nconflicts:
90                         break
91             else:
92                 continue
93             break
94         else:
95             enable_schedules.append(s)
96     return enable_schedules
97
98 # 競合直列化可能であるかを確認する。
99 # まず、競合の起きない操作のペアのリストを check_conflict を用いて作成する。
100 # 次に、 search_schedule を用いて、元のスケジュールと競合の起きないスケジュールのリストを作成する。
101 # その後、 make_goal で元のスケジュールから直列スケジュールを作成する。
102 # 最後に、直列スケジュールのいずれかが元のスケジュールと競合の起きないスケジュールのリストに
103 # 含まれるかを確認して、含まれる場合、「競合直列化可能」と表示する。
104 # また、含まれない場合、「競合直列化不可能」と表示する。
105 def solve(schedule):
106     print(' 元のスケジュール：', end='')
107     print_schedule(schedule)
108     nconflicts = check_conflict(schedule)
109     es = search_schedule(schedule, nconflicts)
110     gs = make_goal(schedule)
111     hantei = False
112     for g in gs:
113         print(' goal: ', end='')
114         print_schedule(g)
115         if tuple(g) in es:
116             print(' 可能')
117             hantei = True
118         else:
119             print(' 不可能')
120
121     print(' 結果：', end='')
122     if hantei:
123         print(' 競合直列化可能')
124     else:
125         print(' 競合直列化不可能')
126     print()

```

```

127
128
129 S1 = [ # (1) のスケジュール
130     Sousa("r", "t", "x"),
131     Sousa("r", "s", "x"),
132     Sousa("r", "u", "x"),
133     Sousa("r", "t", "y"),
134     Sousa("r", "s", "y"),
135     Sousa("w", "u", "x"),
136     Sousa("w", "t", "x"),
137     Sousa("w", "s", "z")
138 ]
139
140 S2 = [ # (2) のスケジュール
141     Sousa("r", "t", "x"),
142     Sousa("r", "s", "y"),
143     Sousa("r", "u", "z"),
144     Sousa("r", "t", "y"),
145     Sousa("r", "s", "x"),
146     Sousa("r", "u", "y"),
147     Sousa("w", "t", "z"),
148     Sousa("w", "s", "z"),
149     Sousa("w", "u", "z")
150 ]
151
152 S3 = [ # (3) のスケジュール
153     Sousa("r", "t", "x"),
154     Sousa("r", "s", "x"),
155     Sousa("r", "u", "x"),
156     Sousa("r", "t", "y"),
157     Sousa("r", "s", "z"),
158     Sousa("w", "t", "y"),
159     Sousa("r", "u", "z"),
160     Sousa("w", "s", "z"),
161     Sousa("w", "t", "z"),
162     Sousa("w", "u", "x")
163 ]
164
165 S4 = [ # オリジナルのスケジュール
166     Sousa('r', 's', 'x'),
167     Sousa('r', 's', 'y'),
168     Sousa('r', 't', 'x'),
169     Sousa('r', 't', 'z'),
170     Sousa('w', 's', 'y'),
171     Sousa('r', 'u', 'y'),
172     Sousa('r', 'u', 'z'),
173     Sousa('w', 'u', 'y'),
174     Sousa('w', 't', 'x'),
175     Sousa('w', 't', 'z')
176 ]

```

```

177
178 print(' (1) のスケジュール')
179 solve(S1)
180
181 print(' (2) のスケジュール')
182 solve(S2)
183
184 print(' (3) のスケジュール')
185 solve(S3)
186
187 print(' オリジナルのスケジュール')
188 solve(S4)

```

## (2) Result

```

1  (1) のスケジュール
2  元のスケジュール：Rt(x) Rs(x) Ru(x) Rt(y) Rs(y) Wu(x) Wt(x) Ws(z)
3  goal: Rs(x) Rs(y) Ws(z) Rt(x) Rt(y) Wt(x) Ru(x) Wu(x)
4  不可能
5  goal: Rs(x) Rs(y) Ws(z) Ru(x) Wu(x) Rt(x) Rt(y) Wt(x)
6  不可能
7  goal: Rt(x) Rt(y) Wt(x) Rs(x) Rs(y) Ws(z) Ru(x) Wu(x)
8  不可能
9  goal: Rt(x) Rt(y) Wt(x) Ru(x) Wu(x) Rs(x) Rs(y) Ws(z)
10 不可能
11 goal: Ru(x) Wu(x) Rs(x) Rs(y) Ws(z) Rt(x) Rt(y) Wt(x)
12 不可能
13 goal: Ru(x) Wu(x) Rt(x) Rt(y) Wt(x) Rs(x) Rs(y) Ws(z)
14 不可能
15 結果： 競合直列化不可能
16
17 (2) のスケジュール
18 元のスケジュール：Rt(x) Rs(y) Ru(z) Rt(y) Rs(x) Ru(y) Wt(z) Ws(z) Wu(z)
19 goal: Rs(y) Rs(x) Ws(z) Rt(x) Rt(y) Wt(z) Ru(z) Ru(y) Wu(z)
20 不可能
21 goal: Rs(y) Rs(x) Ws(z) Ru(z) Ru(y) Wu(z) Rt(x) Rt(y) Wt(z)
22 不可能
23 goal: Rt(x) Rt(y) Wt(z) Rs(y) Rs(x) Ws(z) Ru(z) Ru(y) Wu(z)
24 不可能
25 goal: Rt(x) Rt(y) Wt(z) Ru(z) Ru(y) Wu(z) Rs(y) Rs(x) Ws(z)
26 不可能
27 goal: Ru(z) Ru(y) Wu(z) Rs(y) Rs(x) Ws(z) Rt(x) Rt(y) Wt(z)
28 不可能
29 goal: Ru(z) Ru(y) Wu(z) Rt(x) Rt(y) Wt(z) Rs(y) Rs(x) Ws(z)
30 不可能
31 結果： 競合直列化不可能
32
33 (3) のスケジュール
34 元のスケジュール：Rt(x) Rs(x) Ru(x) Rt(y) Rs(z) Wt(y) Ru(z) Ws(z) Wt(z) Wu(x)
35 goal: Rs(x) Rs(z) Ws(z) Rt(x) Rt(y) Wt(y) Wt(z) Ru(x) Ru(z) Wu(x)

```

```

36 不可能
37 goal: Rs(x) Rs(z) Ws(z) Ru(x) Ru(z) Wu(x) Rt(x) Rt(y) Wt(y) Wt(z)
38 不可能
39 goal: Rt(x) Rt(y) Wt(y) Wt(z) Rs(x) Rs(z) Ws(z) Ru(x) Ru(z) Wu(x)
40 不可能
41 goal: Rt(x) Rt(y) Wt(y) Wt(z) Ru(x) Ru(z) Wu(x) Rs(x) Rs(z) Ws(z)
42 不可能
43 goal: Ru(x) Ru(z) Wu(x) Rs(x) Rs(z) Ws(z) Rt(x) Rt(y) Wt(y) Wt(z)
44 不可能
45 goal: Ru(x) Ru(z) Wu(x) Rt(x) Rt(y) Wt(y) Wt(z) Rs(x) Rs(z) Ws(z)
46 不可能
47 結果： 競合直列化不可能
48
49 オリジナルのスケジュール
50 元のスケジュール：Rs(x) Rs(y) Rt(x) Rt(z) Ws(y) Ru(y) Ru(z) Wu(y) Wt(x) Wt(z)
51 goal: Rs(x) Rs(y) Ws(y) Rt(x) Rt(z) Wt(x) Wt(z) Ru(y) Ru(z) Wu(y)
52 不可能
53 goal: Rs(x) Rs(y) Ws(y) Ru(y) Ru(z) Wu(y) Rt(x) Rt(z) Wt(x) Wt(z)
54 可能
55 goal: Rt(x) Rt(z) Wt(x) Wt(z) Rs(x) Rs(y) Ws(y) Ru(y) Ru(z) Wu(y)
56 不可能
57 goal: Rt(x) Rt(z) Wt(x) Wt(z) Ru(y) Ru(z) Wu(y) Rs(x) Rs(y) Ws(y)
58 不可能
59 goal: Ru(y) Ru(z) Wu(y) Rs(x) Rs(y) Ws(y) Rt(x) Rt(z) Wt(x) Wt(z)
60 不可能
61 goal: Ru(y) Ru(z) Wu(y) Rt(x) Rt(z) Wt(x) Wt(z) Rs(x) Rs(y) Ws(y)
62 不可能
63 結果： 競合直列化可能

```