

実践情報セキュリティとアルゴリズム 第六回課題

28G23027 川原尚己

実行結果のスクリーンショットを以下に示す.

Status[Check again](#)[Full output](#)

Checking philosophers.tla / philosophers.cfg

Errors: 1 error(s)

Start: 15:16:32 (Jan 31), end: 15:16:33 (Jan 31)

States

| Time | Diameter | Found | Distinct | Queue |
|----------|----------|-------|----------|-------|
| 00:00:00 | 5 | 29 | 20 | 6 |

Coverage

| Module | Action | Total | Distinct |
|--------------|-----------------------------|-------|----------|
| philosophers | Init | 2 | 2 |
| philosophers | Take1 | 19 | 9 |
| philosophers | Take2 | 8 | 5 |
| philosophers | Eat | 5 | 4 |
| philosophers | Put1 | 1 | 1 |
| philosophers | Put2 | 0 | 0 |
| philosophers | Terminating | 0 | 0 |

Errors

Deadlock reached.

Error Trace

Filter

[Hide unmodified](#)

▼ 1: Initial predicate

▼ chopsticks (3)

1

0 := FALSE @@ 1 := FALSE @@ 2 := FALSE

2

1 := FALSE

3

2 := FALSE

▼ left (3)

1

0 := FALSE @@ 1 := FALSE @@ 2 := FALSE

2

1 := FALSE

3

2 := FALSE

▼ pc (3)

1

0 := "Take1" @@ 1 := "Take1" @@ 2 := "Take1"

2

1 := "Take1"

3

2 := "Take1"

▼ right (3)

1

0 := FALSE @@ 1 := FALSE @@ 2 := FALSE

2

1 := FALSE

3

2 := FALSE

▼ 2: Take1 in philosophers >>

▶ chopsticks (3) M

0 := TRUE @@ 1 := FALSE @@ 2 := FALSE

▶ left (3)

0 := FALSE @@ 1 := FALSE @@ 2 := FALSE

▶ pc (3) M

0 := "Take2" @@ 1 := "Take1" @@ 2 := "Take1"

▶ right (3) M

0 := TRUE @@ 1 := FALSE @@ 2 := FALSE

▼ 3: Take1 in philosophers >>

▶ chopsticks (3) M

0 := TRUE @@ 1 := TRUE @@ 2 := FALSE

▶ left (3)

0 := FALSE @@ 1 := FALSE @@ 2 := FALSE

▶ pc (3) M

0 := "Take2" @@ 1 := "Take2" @@ 2 := "Take1"

▶ right (3) M

0 := TRUE @@ 1 := TRUE @@ 2 := FALSE

▼ 4: Take1 in philosophers >>

▶ chopsticks (3) M

0 := TRUE @@ 1 := TRUE @@ 2 := TRUE

▶ left (3)

0 := FALSE @@ 1 := FALSE @@ 2 := FALSE

▶ pc (3) M

0 := "Take2" @@ 1 := "Take2" @@ 2 := "Take2"

▶ right (3) M

0 := TRUE @@ 1 := TRUE @@ 2 := TRUE

初期状態について、philosophers.cfg において” CONSTANTS PHILOSOPHERS = {0, 1, 2}”となっているため、哲学者の人数は三名である。また、philosophers.tla において

```

fair process philosopher
  variables
    left = FALSE,
    right = FALSE;

```

とあるので、哲学者は誰も一本も箸を持っておらず、各哲学者の両隣に箸が一本ずつ置かれている状況である。エラートレースに対してもこれらのことが確認でき、状況が一致している。

次に、デッドロックが起こる理由について説明する。

TLA+のソースコード及びエラートレースにおいて、"Take1"は右側の箸を取る操作と一致している。つまり、初期状態においては各哲学者が""Take1"を実行することになっているから、一人目から順に自分の右手側の箸をとっていくことになる。ソースコードによると、"Take1"を実行した後は、左手側の箸を取る行為に相当する"Take2"を実行することになっているが、哲学者全員が"Take1"を実行した後は卓上に箸は一本も残っていない。よって、誰も"Take2"を行うことができず、デッドロックが発生する。