レポート７

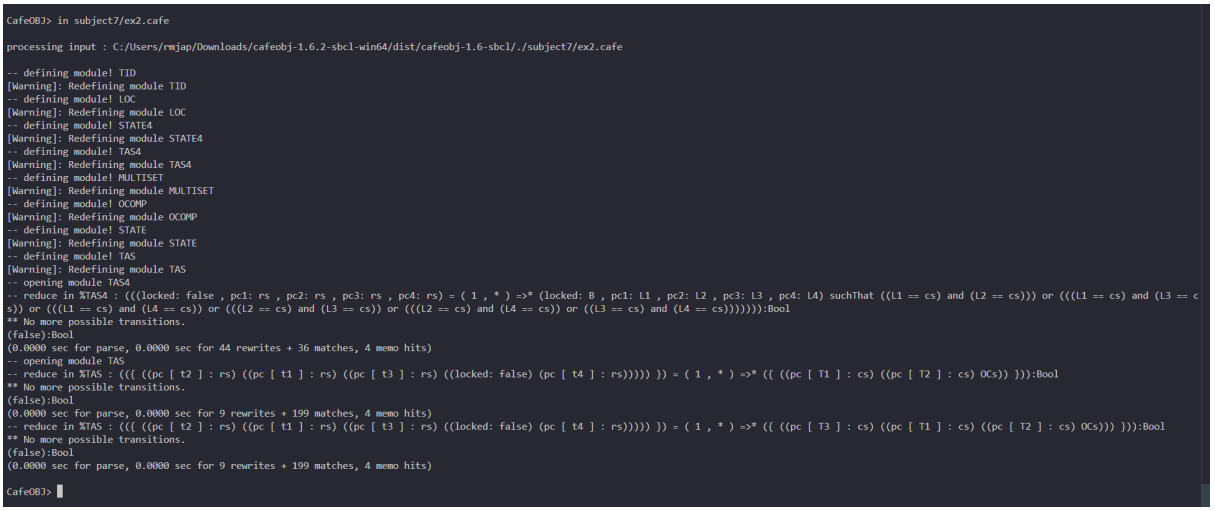
課題１

```
CafeOBJ>

in subject7/ex1.cafe

processing input : C:/Users/rmjap/Downloads/cafeobj-1.6.2-sbcl-win64/dist/cafeobj-1.6-sbcl/./subject7/ex1.cafe

-- defining module! TID
[Warning]: Redefining module TID
-- defining module! LOC
[Warning]: Redefining module LOC
-- defining module! STATE2
-- defining module! FMUTEX2
-- opening module FMUTEX2
-- reduce in %FMUTEX2 : ((locked: false , pc1: rs , pc2: rs) = ( 1 , * ) =>* (locked: B , pc1: cs , pc2: cs)):Bool
** Found [state 0-8] (locked: true , pc1: cs , pc2: cs):State2
-- target: (locked: B , pc1: cs , pc2: cs)
    { B |-> true, B |-> true, L1 |-> cs }

-- found required number of solutions 1.
(true):Bool
(0.0000 sec for parse, 0.0000 sec for 15 rewrites + 1 matches, 6 memo hits)
[state 0-0] (locked: false , pc1: rs , pc2: rs):State2
  trans [want1]: (locked: false , pc1: rs , pc2: L2) => (locked: false , pc1: ms , pc2: L2)
[state 0-1] (locked: false , pc1: ms , pc2: rs):State2
  trans [want2]: (locked: false , pc1: L1 , pc2: rs) => (locked: false , pc1: L1 , pc2: ms)
[state 0-4] (locked: false , pc1: ms , pc2: ms):State2
  trans [try1]: (locked: B , pc1: ms , pc2: L2) => (locked: true , pc1: cs , pc2: L2)
[state 0-6] (locked: true , pc1: cs , pc2: ms):State2
  trans [try2]: (locked: B , pc1: L1 , pc2: ms) => (locked: true , pc1: L1 , pc2: cs)
[state 0-8] (locked: true , pc1: cs , pc2: cs):State2
    { B |-> true, B |-> true, L1 |-> cs }
-- defining module! MULTISET
[Warning]: Redefining module MULTISET
-- reading in file  : nat

processing input : C:/Users/rmjap/Downloads/cafeobj-1.6.2-sbcl-win64/dist/cafeobj-1.6-sbcl/lib/nat.cafe

-- defining module! NAT
-- reading in file  : nznat

processing input : C:/Users/rmjap/Downloads/cafeobj-1.6.2-sbcl-win64/dist/cafeobj-1.6-sbcl/lib/nznat.cafe

-- defining module! NZNAT
-- done reading in file: nznat

-- done reading in file: nat

-- opening module MULTISET(E <= NAT)
-- reduce in %MULTISET(E <= NAT) : ((1 (2 3)) == (1 (2 3))):Bool
(true):Bool
(0.0000 sec for parse, 0.0000 sec for 1 rewrites + 9 matches)
-- reduce in %MULTISET(E <= NAT) : ((1 2) == (2 1)):Bool
(true):Bool
(0.0000 sec for parse, 0.0000 sec for 1 rewrites + 5 matches)
-- reduce in %MULTISET(E <= NAT) : ((1 (1 (2 (1 2)))) == (1 (2 (1 (1 2))))):Bool
(true):Bool
(0.0000 sec for parse, 0.0000 sec for 1 rewrites + 45 matches)
-- reduce in %MULTISET(E <= NAT) : ((1 2) emp):MSet
(2 1):MSet
```

```
(2 1):MSet
(0.0000 sec for parse, 0.0000 sec for 1 rewrites + 5 matches)
-- reduce in %MULTISET(E <= NAT) : ((1 2) (1 (2 (1 (emp (emp (emp (emp (emp emp)))))))):MSet
(1 (1 (2 (2 1)))):MSet
(0.0000 sec for parse, 0.0000 sec for 6 rewrites + 14 matches)
-- reduce in %MULTISET(E <= NAT) : (emp):MSet
(emp):MSet
(0.0000 sec for parse, 0.0000 sec for 0 rewrites + 0 matches)
-- reduce in %MULTISET(E <= NAT) : (emp emp):MSet
(emp):MSet
(0.0000 sec for parse, 0.0000 sec for 1 rewrites + 1 matches)
-- defining module! OCOMP
[Warning]: Redefining module OCOMP
-- opening module MULTISET(E <= OCOMP)
-- reduce in %MULTISET(E <= OCOMP) : (((pc [ t1 ] : rs) (locked: false)) == ((locked: false) (pc [ t1 ] : rs))):Bool
(true):Bool
(0.0000 sec for parse, 0.0000 sec for 1 rewrites + 5 matches)
-- reduce in %MULTISET(E <= OCOMP) : (((pc [ t1 ] : rs) ((locked: false) (pc [ t2 ] : cs))) == ((locked: false) ((pc [ t2 ] : cs) (pc [ t1 ] : rs)))):Bool
(true):Bool
(0.0000 sec for parse, 0.0000 sec for 1 rewrites + 9 matches)
-- defining module! STATE
[Warning]: Redefining module STATE
-- defining module! FMUTEX
-- opening module FMUTEX
-- reduce in %FMUTEX : (({ ((locked: false) ((pc [ t1 ] : rs) (pc [ t2 ] : rs))) }) = ( 1 , * ) =>* ({ ((pc [ T1 ] : cs) ((pc [ T2 ] : cs) OCs)) })):Bool
** Found [state 0-8] ({ ((pc [ t1 ] : cs) ((pc [ t2 ] : cs) (locked: true))) }):State
-- target: ({ ((pc [ T1 ] : cs) ((pc [ T2 ] : cs) OCs)) })
    { T2 |-> t2, T1 |-> t1, OCs |-> (locked: true), T |-> t1, B |-> true, OCs |-> (pc [ t2 ] : cs) }
    { T2 |-> t1, T1 |-> t2, OCs |-> (locked: true), T |-> t1, B |-> true, OCs |-> (pc [ t2 ] : cs) }

-- found required number of solutions 1.
(true):Bool
(0.0000 sec for parse, 0.0000 sec for 15 rewrites + 61 matches, 6 memo hits)
[state 0-0] ({ ((pc [ t1 ] : rs) ((pc [ t2 ] : rs) (locked: false))) }):State
  trans [want]: ({ ((locked: false) ((pc [ T ] : rs) OCs)) }) => ({ ((locked: false) ((pc [ T ] : ms) OCs)) })
[state 0-1] ({ ((pc [ t1 ] : ms) ((pc [ t2 ] : rs) (locked: false))) }):State
  trans [want]: ({ ((locked: false) ((pc [ T ] : rs) OCs)) }) => ({ ((locked: false) ((pc [ T ] : ms) OCs)) })
[state 0-3] ({ ((pc [ t2 ] : ms) ((pc [ t1 ] : ms) (locked: false))) }):State
  trans [try]: ({ ((locked: B) ((pc [ T ] : ms) OCs)) }) => ({ ((locked: true) ((pc [ T ] : cs) OCs)) })
[state 0-6] ({ ((pc [ t2 ] : cs) ((pc [ t1 ] : ms) (locked: true))) }):State
  trans [try]: ({ ((locked: B) ((pc [ T ] : ms) OCs)) }) => ({ ((locked: true) ((pc [ T ] : cs) OCs)) })
[state 0-8] ({ ((pc [ t1 ] : cs) ((pc [ t2 ] : cs) (locked: true))) }):State
    { T2 |-> t2, T1 |-> t1, OCs |-> (locked: true), T |-> t1, B |-> true, OCs |-> (pc [ t2 ] : cs) }
    { T2 |-> t1, T1 |-> t2, OCs |-> (locked: true), T |-> t1, B |-> true, OCs |-> (pc [ t2 ] : cs) }
-- defining module! STATE3
-- defining module! MUTEX3
-- opening module MUTEX3
-- reduce in %MUTEX3 : (((locked: false , pc1: rs , pc2: rs , pc3: rs) = ( 1 , * ) =>* (locked: B , pc1: L1 , pc2: L2 , pc3: L3) suchThat ((L1 == cs) and (L2 == cs))) or ((((L1 == cs) and (L3 == cs)) or ((L2 ==
cs) and (L3 == cs)))):Bool
** No more possible transitions.
(false):Bool
(0.0000 sec for parse, 0.0000 sec for 27 rewrites + 21 matches, 3 memo hits)
-- defining module! MUTEX
-- opening module MUTEX
-- reduce in %MUTEX : (({ ((pc [ t3 ] : rs) ((pc [ t2 ] : rs) ((locked: false) (pc [ t1 ] : rs)))) }) = ( 1 , * ) =>* ({ ((pc [ T1 ] : cs) ((pc [ T2 ] : cs) OCs)) })):Bool
** No more possible transitions.
```

```
** No more possible transitions.
(false):Bool
(0.0000 sec for parse, 0.0000 sec for 7 rewrites + 85 matches, 3 memo hits)
-- defining module! QUEUE
-- opening module QUEUE(E <= TID)
-- reduce in %QUEUE(E <= TID) : (((emq ; t1) ; (t2 ; emq)) == ((t1 ; t2) ; emq)):Bool
(true):Bool
(0.0000 sec for parse, 0.0000 sec for 4 rewrites + 36 matches)
-- reduce in %QUEUE(E <= TID) : ((t1 ; t2) == (t2 ; t1)):Bool
(false):Bool
(0.0000 sec for parse, 0.0000 sec for 1 rewrites + 17 matches)
-- reduce in %QUEUE(E <= TID) : (((t1 ; emq) ; (t2 ; emq)) == ((t1 ; t2) ; emq)):Bool
(true):Bool
(0.0000 sec for parse, 0.0000 sec for 4 rewrites + 40 matches)
-- defining module! OCOMP
[Warning]: Redefining module OCOMP
-- defining module! STATE
[Warning]: Redefining module STATE
-- defining module! FQLOCK
-- reduce in %FQLOCK : (({ ((pc [ t2 ] : rs) ((pc [ t1 ] : rs) ((tmp [ t1 ] : emq) ((queue: emq) (tmp [ t2 ] : emq))))) }) = ( 1 , * ) =>* ({ ((pc [ T1 ] : cs) ((pc [ T2 ] : cs) OCs)) })):Bool
** Found [state 0-33] ({ ((pc [ t1 ] : cs) ((queue: t1) ((pc [ t2 ] : cs) ((tmp [ t2 ] : t2) (tmp [ t1 ] : t1))))) }):State
-- target: ({ ((pc [ T1 ] : cs) ((pc [ T2 ] : cs) OCs)) })
    { T2 |-> t2, T1 |-> t1, OCs |-> ((queue: t1) ((tmp [ t2 ] : t2) (tmp [ t1 ] : t1))), Q1 |-> emq, Q2 |-> t1, T |-> t1, OCs |-> ((tmp [ t2 ] : t2) (pc [ t2 ] : cs)) }
    { T2 |-> t1, T1 |-> t2, OCs |-> ((queue: t1) ((tmp [ t2 ] : t2) (tmp [ t1 ] : t1))), Q1 |-> emq, Q2 |-> t1, T |-> t1, OCs |-> ((tmp [ t2 ] : t2) (pc [ t2 ] : cs)) }

-- found required number of solutions 1.
(true):Bool
(0.0000 sec for parse, 0.0156 sec for 77 rewrites + 2367 matches, 33 memo hits)
-- defining module! QLOCK
-- defining module! QLOCK
[Warning]: Redefining module QLOCK
-- opening module QLOCK
-- reduce in %QLOCK : (({ ((queue: emq) ((pc [ t1 ] : rs) (pc [ t2 ] : rs))) }) = ( 1 , * ) =>* ({ ((pc [ T1 ] : cs) ((pc [ T2 ] : cs) OCs)) })):Bool
** No more possible transitions.
(false):Bool
(0.0000 sec for parse, 0.0000 sec for 19 rewrites + 125 matches, 10 memo hits)
-- reduce in %QLOCK : (({ ((pc [ t3 ] : rs) ((pc [ t2 ] : rs) ((queue: emq) (pc [ t1 ] : rs)))) }) = ( 1 , * ) =>* ({ ((pc [ T1 ] : cs) ((pc [ T2 ] : cs) OCs)) })):Bool
** No more possible transitions.
(false):Bool
(0.0000 sec for parse, 0.0156 sec for 64 rewrites + 1729 matches, 33 memo hits)
-- reduce in %QLOCK : (({ ((pc [ t2 ] : rs) ((pc [ t1 ] : rs) ((pc [ t3 ] : rs) ((queue: emq) (pc [ t4 ] : rs))))) }) = ( 1 , * ) =>* ({ ((pc [ T1 ] : cs) ((pc [ T2 ] : cs) OCs)) })):Bool
** No more possible transitions.
(false):Bool
(0.0000 sec for parse, 0.0313 sec for 261 rewrites + 14383 matches, 132 memo hits)

CafeOBJ>
```

課題２



課題３

状態表現方法の比較分析

プロセスの状態をマルチセットで表現する方法は、レコードで表現する方法と比較して、より汎用的で保守しやすい状態管理を実現できる。

スケーラビリティの観点では、課題2のようにプロセス数が増加した場合、マルチセット版では既存の遷移規則をそのまま使用できるのに対し、レコード版ではプロセス数に応じて個別の状態遷移規則を新たに定義する必要がある。具体的には、4スレッド対応のレコード版では8個の遷移規則（try1-4, exit1-4）が必要となるが、マルチセット版では2個の汎用的な遷移規則で全スレッドに対応可能である。

この違いの根本的な理由は、マルチセットが交換法則を満たす**ことである。特に交換法則により**、状態要素の順序に依存せず、1つの汎用的な遷移規則`{(locked: false) (pc[T]: rs) OCs}`で任意のスレッドの動作を表現できる。ここで変数`T`は任意のスレッドにマッチし、`OCs`はその他の状態要素すべてを表すため、柔軟なパターンマッチングが実現される。

一方、レコード版では各スレッドのフィールド位置が構造的に固定されており、`pc1`はスレッドt1専用、`pc2`はスレッドt2専用というように、スレッドと位置の対応関係が変更できない。そのため、各スレッド専用の遷移規則（try1, try2, try3, try4など）を個別に定義する必要がある。

このように、マルチセット版の方が、プロセスを増やす際にも**容易に**拡張できる**という利点を持つ。**

課題４

```
stateDiagram-v2
    [*] --> S0 : Initialize
```

```
state S0 {
   [*] --> idle
   idle : queue=emq, both_rs
}

S0 --> S1 : [want] T=t2
S0 --> S4 : [want] T=t1

state S1 {
   [*] --> t2_waiting
   t2_waiting : queue=t2, t1=rs, t2=ws
}

state S4 {
   [*] --> t1_waiting
   t1_waiting : queue=t1, t1=ws, t2=rs
}

S1 --> S2 : [want] T=t1
S4 --> S2 : [want] T=t2

state S2 {
   [*] --> both_waiting
   both_waiting : queue=(first;second), both_ws
}

S2 --> S3 : [try] T=t2 (if t2 first)
S2 --> S6 : [try] T=t1 (if t1 first)

state S3 {
   [*] --> t2_critical
   t2_critical : queue=(t2;t1), t1=ws, t2=cs
}

state S6 {
   [*] --> t1_critical
   t1_critical : queue=(t1;t2), t1=cs, t2=ws
}
```

```
S3 → S7 : [exit1] T=t2
S6 → S8 : [exit1] T=t1

S7 → S5 : [try] T=t1
S8 → S9 : [try] T=t2

S5 → [*] : [exit2] T=t1
S9 → [*] : [exit2] T=t2
```