## Appendix

**Algorithm 4. CHB dependency detection**

```
1   procedure begin(t, l)
2   |  H(t) = inc(H(t))
3   |  C(t) = new TransactionNode(l)
4   |  H(C(t).begin) = C(t).currVC
5   |  C(t).clk = 1
6   end procedure

7   procedure end(t, l)
8   |  C(t) = null
9   end procedure

10  procedure r(t, x)
11  |  for each m ∈ HeldLocks(t) do
12  |  |  P(t) = P(t) ⊔ LW(m, x)
13  |  |  R_m = R_m ∪ {x}
14  |  end for
15  |  if t ≠ T(W(x)) and R(t, x) = null then
16  |  |  P(t) = P(t) ⊔ PW(x)
17  |  |  join(W(x), C(t))
18  |  end if
19  |  R(t, x) = H(t)
20  |  Rclk(t, x) = C(t).clk
21  |  PR(t, x) = P(t)
22  end procedure

23  procedure w(t, x)
24  |  for each m ∈ HeldLocks(t) do
25  |  |  P(t) = P(t) ⊔ (LW(m, x) ⊔ LR(m, x))
26  |  |  W_m = W_m ∪ {x}
27  |  end for
28  |  if exist lastRead of x then
29  |  |  for each t' ∈ Tid and t' ≠ t do
30  |  |  |  P(t) = P(t) ⊔ PR(t', x)
31  |  |  |  join(R(t', x), C(t))
32  |  |  end for
33  |  else if t ≠ T(W(x)) then
34  |  |  P(t) = P(t) ⊔ PW(x)
35  |  |  join(W(x), C(t))
36  |  end if
37  |  W(x) = H(t)
38  |  Wclk(x) = C(t).clk
39  |  PW(x) = P(t)
40  |  for each t' ∈ Tid do
41  |  |  R(t', x) = null
42  |  |  PR(t', x) = null
43  |  end for
44  end procedure

46  procedure acq(t, m)
47  |  HeldLocks(t) = HeldLocks(t) ∪ {m}
48  |  C(t).clk = C(t).clk + 1
49  |  currCS(m).acq = H(t)[t]@C(t).clk@t
50  |  if t ≠ T(L(m)) then
51  |  |  P(t) = P(t) ⊔ PL(m)
52  |  |  join(L(m), C(t))
53  |  end if
54  |  for each t' ∈ Tid and t' ≠ t do
55  |  |  Acq_m(t').Enque(P(t)[t:=H(t)[t]])
56  |  end for
57  |  currCS(m).acq.PC = P(t)[t:=H(t)[t]]
58  end procedure

59  procedure rel(t, m)
60  |  currCS(m).rel = H(t)[t]@C(t).clk@t
61  |  if lastCS(m).tid ≠ t and
62  |  |  not lastCS(m).acq.PC ⊑ P(t)[t:=H(t)[t]] then
63  |  |  //CHB dependency detected
64  |  end if
65  |  while Acq_m(t).Front ⊑ P(t)[t:=H(t)[t]] do
66  |  |  Acq_m(t).Deque()
67  |  |  P(t) = P(t) ⊔ Rel_m(t).Deque()
68  |  end while
69  |  for each x ∈ R_m do
70  |  |  LR(m, x) = LR(m, x) ⊔ H(t)
71  |  end for
72  |  for each x ∈ W_m do
73  |  |  LW(m, x) = LW(m, x) ⊔ H(t)
74  |  end for
75  |  C(t).clk = C(t).clk + 1
76  |  L(m) = H(t)
77  |  Lclk(m) = C(t).clk
78  |  PL(m) = P(t)
79  |  R_m ← W_m ← ∅
80  |  for each t' ∈ Tid and t' ≠ t do
81  |  |  Rel_m(t').Enque(H(t))
82  |  end for
83  |  lastCS(m) = currCS(m)
84  |  HeldLocks(t) = HeldLocks(t) / {m}
85  end procedure
```

Algorithm 4 is modified from Algorithm 1 of weak causally-precedes (WCP) to detect CHB dependency. The following notations are used in the algorithm.

- ***C*(*t*)** denotes the current transaction node for thread *t*.
- ***H*(*t*)** denotes the current happens-before vector clock of thread *t*.
- ***P*(*t*)** denotes the current weak causally-precedes vector clock of thread *t*.
- ***H*(*e*)** denotes the HB vector clock of event *e*, which is a shadow state of $H(T(e))$ when executing *e*.
- ***P*(*e*)** denotes the WCP vector clock of event *e*, which is a shadow state of $P(T(e))$ when executing *e*.
- ***W*(*x*)** denotes the happens-before vector clock of the last write event to the variable *x*.
- ***PW*(*x*)** denotes the weak causally-precedes vector clock of the last write event to the variable *x*.

- ***R*(*t*, *x*)** denotes the happens-before vector clock of the last read event of the variable *x* performed by thread *t*.
- ***PR*(*t*, *x*)** denotes the weak causally-precedes vector clock of the last read event of the variable *x* performed by thread *t*.
- ***L*(*m*)** denotes the happens-before vector clock of the last release event of lock *m*.
- ***PL*(*m*)** denotes the weak causally-precedes vector clock of the last release event of lock *m*.
- ***currCS*(*m*)** and ***lastCS*(*m*)** represent the current critical section and last critical section of lock *m*.
- $Acq_m(t)$ and $Rel_m(t)$ denote two FIFO queues of lock *m* and thread *t* to store the times of acquire and release events of lock *m* performed by other threads.
- $R_m$ and $W_m$ denote the set of variables that the critical sections on lock *m* contain a read event and a write event to that variable.
- ***HeldLocks*(*t*)** represents the set of locks that thread *t* currently holds.
- ***Tid*** denotes the set of threads running in the program.