

EXTENDS Integers

⊢

CONSTANT N

ASSUME N ⊢ in Nat

Procs == 1..N

⊢

a \prec b == \ / a[1] < b[1]

⊢ \ / (a[1] = b[1]) ⊢ \ / (a[2] < b[2])

⊢

(\*

--algorithm AtomicBakery

{ ⊢ variable num = [i \ in Procs | -> 0] ⊢;

⊢

⊢ process (p \ in Procs)

⊢ ⊢ variables unchecked, ⊢ max;

⊢ ⊢ { ncs: ⊢ while (TRUE)

⊢ ⊢ ⊢ { e1: ⊢ ⊢ unchecked := Procs \ {self} ⊢;

⊢ ⊢ ⊢ max := 0;

⊢ ⊢ ⊢ e2: ⊢ ⊢ while (unchecked # {})

⊢ ⊢ ⊢ { ⊢ with (i \ in unchecked)

⊢ ⊢ ⊢ { unchecked := unchecked \ {i};

⊢ ⊢ ⊢ if (num[i] > max) ⊢ { max := num[i] }

⊢ ⊢ ⊢ }

⊢ ⊢ ⊢ };

⊢ ⊢ ⊢ e3: ⊢ ⊢ with (i \ in {j \ in Nat: ⊢ j > max}) ⊢ { num[self] :=

⊢ ⊢ ⊢ unchecked := Procs \ {self} ⊢;

⊢ ⊢ ⊢ wait: ⊢ ⊢ while (unchecked # {})

⊢ ⊢ ⊢ { ⊢ with (i \ in unchecked)

⊢ ⊢ ⊢ { ⊢ await \ / num[i] = 0

⊢ ⊢ ⊢ \ / << num[self], ⊢ self >> \ prec << num

⊢ ⊢ ⊢ unchecked := unchecked \ {i}

⊢ ⊢ ⊢ }

⊢ ⊢ ⊢ };

⊢ ⊢ ⊢ cs: ⊢ ⊢ skip; ⊢ ⊢ \* the critical section;

⊢ ⊢ ⊢ exit: num[self] := 0

⊢ ⊢ ⊢ }

⊢ ⊢ }

}

\*)

⊢

MutualExclusion == \ A i, j \ in Procs: ⊢ (i # j) ⊢ => \ / \ pc[i] = "cs"

⊢ \ / \ pc[j] = "cs"

