```
THEOREM Spec => DeadlockFree

<1> DEFINE T0 == Trying(0)
           T1 == Trying(1)
           Success == InCS(0) \/ InCS(1)

<1>1. Spec => [] LInv                                                    PROOF
  (*************************************************************************)
  (* This is a standard invariance proof.                                  *)
  (*************************************************************************) OMITTED

<1>2. SUFFICES ASSUME [] LInv /\ [] [Next]_vars /\ Fairness
      PROVE  DeadlockFree                                                PROOF
  (*************************************************************************)
  (* By 1 and the definition of Spec.                                      *)
  (*************************************************************************) OMITTED

<1>3. SUFFICES ASSUME [] ~Success
      PROVE  (T0 \/ T1) ~> FALSE                                         PROOF
  (*************************************************************************)
  (* This is a standard temporal proof by contradiction, since             *)
  (* DeadlockFree equals (T0 \/ T1) ~> Success.                            *)
  (*************************************************************************) OMITTED

<1>4. T0 ~> FALSE
  <2>1. T0 ~> [] (pc[0] = "e2")                                          PROOF
    (*************************************************************************)
    (* Assumption [] LInv implies process 0 is never at e3 or e4.          *)
    (* Therefore, by the code and assumption Fairness, we see that if T0   *)
    (* is true and process 0 never reaches cs (which is implied by the     *)
    (* assumption [] ~Success, then process 0 eventually reaches e2 and    *)
    (* stays there forever.                                                *)
    (*************************************************************************) OMITTED
  <2>2. [] (pc[0] = "e2") ~> [] ((pc[0] = "e2") /\ ~x[1])
    <3>1. SUFFICES ASSUME [] (pc[0] = "e2")
          PROVE  [] ((pc[0] = "e2") /\ ~x[1])                            PROOF
      (*********************************************************************)
      (* By the [] ~ Rule.                                                 *)
      (*********************************************************************) OMITTED
    <3>2. TRUE ~> ([] (pc[1] = "ncs") \/ [] T1)                          PROOF
      (*********************************************************************)
      (* The code and assumption Fairness imply that if process 1 never    *)
      (* reaches cs (by the assumption [] ~Success), then eventually it    *)
      (* must either reach and remain forever at ncs, or T1 must become    *)
      (* true and remain true forever.                                     *)
      (*********************************************************************) OMITTED
    <3>3. [] (pc[1] = "ncs") => [] ~x[1]                                 PROOF
      (*********************************************************************)
      (* [] LInv implies x[1] equals FALSE when process 1 is at ncs.       *)
      (*********************************************************************) OMITTED
    <3>4. [] T1 ~> [] ~x[1]                                              PROOF
      (*********************************************************************)
      (* pc[0] = "e2" implies x[0], so the step <3>1 assumption implies    *)
      (* [] x[0]. The code, Fairness, [] ~Success, and [] x[0] imply that  *)
      (* leads to process 1 reaching and remaining forever at e4 with x[1] *)
      (* equal to FALSE.                                                   *)
      (*********************************************************************) OMITTED
```

```
    <3>5. QED                                                           PROOF
      (*****************************************************************)
      (* By <3>1-<3>4 and Leads-To Induction.                          *)
      (*****************************************************************) OMITTED
  <2>3. []((pc[0] = "e2") /\ ~x[1]) ~> FALSE                            PROOF
    (*******************************************************************)
    (* The code and Fairness implythat (pc[0] = "e2") /\ []~x[1] leads to  *)
    (* process 0 reaching cs, contradicting []~Success.                *)
    (*******************************************************************) OMITTED
  <2>4. QED                                                             PROOF
    (*******************************************************************)
    (* BY <2>1 - <2>3 and Leads-To Induction.                          *)
    (*******************************************************************) OMITTED

<1>5. T1 ~> FALSE
  <2>1. T1 => []T1                                                      PROOF
    (*******************************************************************)
    (* From the code, we see that if T1 is true and process 1 never        *)
    (* reaches cs (which is implied by the assumption []~Success), then T1 *)
    (* remains forever true.                                           *)
    (*******************************************************************) OMITTED
  <2>2. []T1  ~>  (T0 \/ [](T1 /\ ~T0))
    (*******************************************************************) PROOF
    (* By the tautologies F ~> (G \/ (F /\ []~G)) and                  *)
    (* []F /\ []G <=> [](F /\ G).                                      *)
    (*******************************************************************) OMITTED
  <2>3. [](T1 /\ ~T0)  ~>  [](T1 /\ ~x[0])                              PROOF
    (*******************************************************************)
    (* By the code and Fairness, []~T0 implies that eventually process 0   *)
    (* is always at ncs, which implies that x[0] always equals FALSE.      *)
    (*******************************************************************) OMITTED
  <2>4. [](T1 /\ ~x[0]) ~> FALSE                                        PROOF
    (*******************************************************************)
    (* The code, Fairness, and []~x[0] imply that process 1 eventually     *)
    (* reaches e2.  Assumption Fairness and []~x[0] then imply that        *)
    (* process 1 reaches cs, contradicting the assumption []~Success.      *)
    (*******************************************************************) OMITTED
  <2>5. QED                                                             PROOF
    (*******************************************************************)
    (* BY <2>1 - <2>4 and Leads-To Induction.                          *)
    (*******************************************************************) OMITTED

<1>6. QED                                                               PROOF
  (*********************************************************************)
  (* By <1>3 - <1>5 and Leads-To Induction.                            *)
  (*********************************************************************) OMITTED
```

CLOSE