Airlocks

1.a

[true*] forall rID: robotID . val(rID == R1 || rID == R2) => [lamp_projectWafer]<true* . robot_dropWafer(rID, OUT_STACK)> true

1.b

[true*] forall aID : airlockID . [airlock_setInnerDoorState(aID, OPEN) . (! (airlock_setInnerDoorState(aID, CLOSED)))* . airlock_setOuterDoorState(aID, OPEN)] false

Lamp

2.a

[true*] [lamp_projectWafer . (!robot_pickUpWafer(LAMP))* . lamp_projectWafer] false

Robots

3.a.i

[true*] forall rID: robotID . val(rID == R1 || rID == R2) => [outerRobot_moveToLocation(rID, matchRobotOutputStack(rID)) . robot_checkOutputStackState(matchRobotOutputStack(rID), FULL) . robot_dropWafer(rID, OUT_STACK)] false

3.a.ii

[true*] forall rID: robotID . val(rID == R1 || rID == R2) => [outerRobot_moveToLocation(rID, matchRobotOutputStack(rID)) . (!robot_checkOutputStackState(matchRobotOutputStack(rID), NFULL))* . robot_dropWafer(rID, OUT_STACK)] false

3.b.i

[true*] forall rID: robotID . val(rID == R1 || rID == R2) => [outerRobot_moveToLocation(rID, matchRobotInputStack(rID)) . robot_checkInputStackState(matchRobotInputStack(rID), EMPTY) . robot_pickUpWafer(rID, INP_STACK)] false

3.b.ii

[true*] forall rID: robotID . val(rID == R1 || rID == R2) => [outerRobot_moveToLocation(rID, matchRobotInputStack(rID)) . (!robot_checkInputStackState(matchRobotInputStack(rID), NEMPTY)) . robot_pickUpWafer(rID, INP_STACK)] false

3.c

[true*] forall rID: robotID . val(rID == R1 || rID == R2) => [airlock_setOuterDoorState(matchRobotAirlock(rID), CLOSED) . (! airlock_setOuterDoorState(matchRobotAirlock(rID), OPEN))* . outerRobot_moveToLocation(rID, matchRobotAirlock(rID)) . robot_pickUpWafer(rID, O_AIRLOCK)] false

3.d

[true*] forall rID: robotID . val(rID == R1 || rID == R2) => [airlock_setOuterDoorState(matchRobotAirlock(rID), CLOSED) . (! airlock_setOuterDoorState(matchRobotAirlock(rID), OPEN))* . outerRobot_moveToLocation(rID, matchRobotAirlock(rID)) . robot_dropWafer(rID, O_AIRLOCK)] false

3.e

[true*] forall aID: airlockID . val(aID == A1 || aID == A2) => [airlock_setInnerDoorState(aID, CLOSED) . (!airlock_setInnerDoorState(aID, OPEN))* . robot_pickUpWafer(matchAirlockInnerRobotLocation(aID))] false

3.f

[true*] forall aID: airlockID . val(aID == A1 || aID == A2) => [airlock_setInnerDoorState(aID, CLOSED) . (!airlock_setInnerDoorState(aID, OPEN))* . robot_dropWafer(matchAirlockInnerRobotLocation(aID))] false

3.g

[true*] forall aID: airlockID . val(aID == A1 || aID == A2) => [robot_pickUpWafer(matchAirlockInnerRobotLocation(aID)) . !robot_dropWafer(LAMP)* . robot_dropWafer(matchAirlockInnerRobotLocation(aID))] false

3.h

[true*] forall rID: robotID . val(rID == R1 || rID == R2) => [robot_dropWafer(rID, OUT_STACK) . ! robot_pickUpWafer(rID, INP_STACK)* . outerRobot_moveToLocation(rID, matchRobotAirlock(rID)) . robot_dropWafer(rID, O_AIRLOCK)] false

3.i

[true*] forall rID: robotID . val(rID == R1 || rID == R2) => [robot_dropWafer(rID, O_AIRLOCK) . ! robot_pickUpWafer(rID, O_AIRLOCK)* . outerRobot_moveToLocation(rID, matchRobotOutputStack(rID)) . robot_dropWafer(rID, OUT_STACK)] false

3.j

[true*] forall rID: robotID . val(rID == R1 || rID == R2) => [robot_pickUpWafer(rID, INP_STACK) . (! lamp_projectWafer)* . robot_dropWafer(rID, OUT_STACK)] false

Liveness Requirements

1

[true*] forall rID: robotID . val(rID == R1 || rID == R2) => [robot_pickUpWafer(rID, INP_STACK)]<true* . lamp_projectWafer> true

2

[true*] forall rID: robotID . val(rID == R1 || rID == R2) => [lamp_projectWafer]<true* . robot_dropWafer(rID, OUT_STACK)> true

3

[true*]<true>true