Structured Production Cell verification model in Uppaal

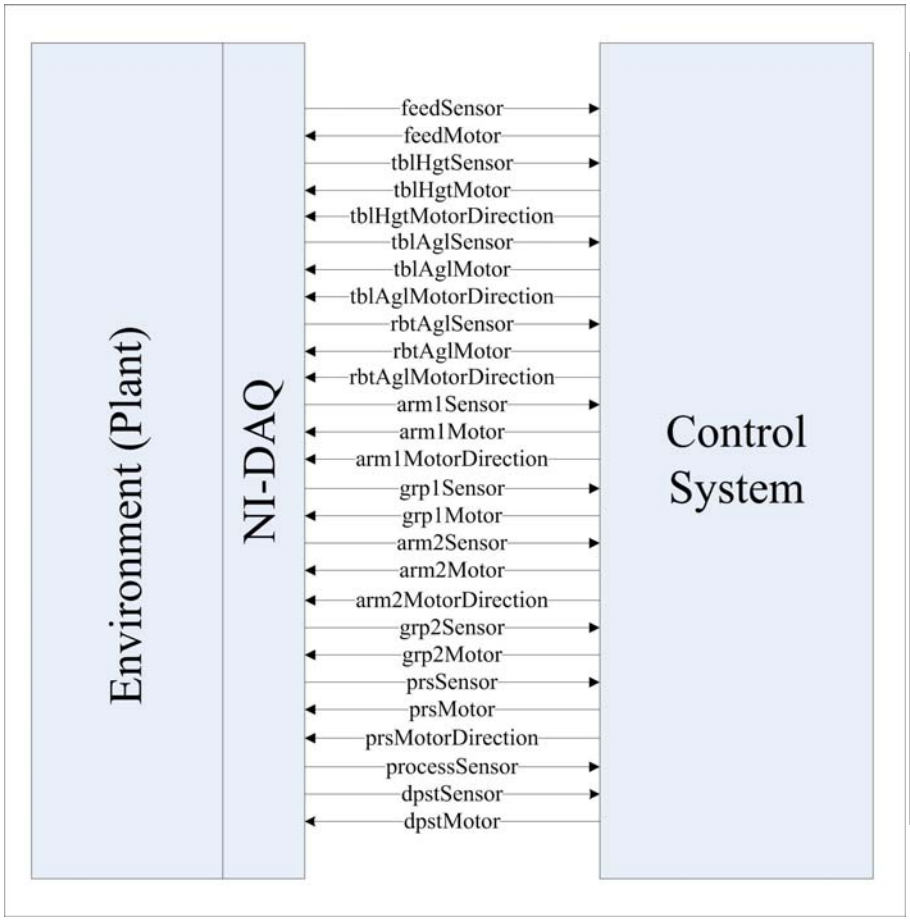
Yu Guo, Ph.D. MoDES

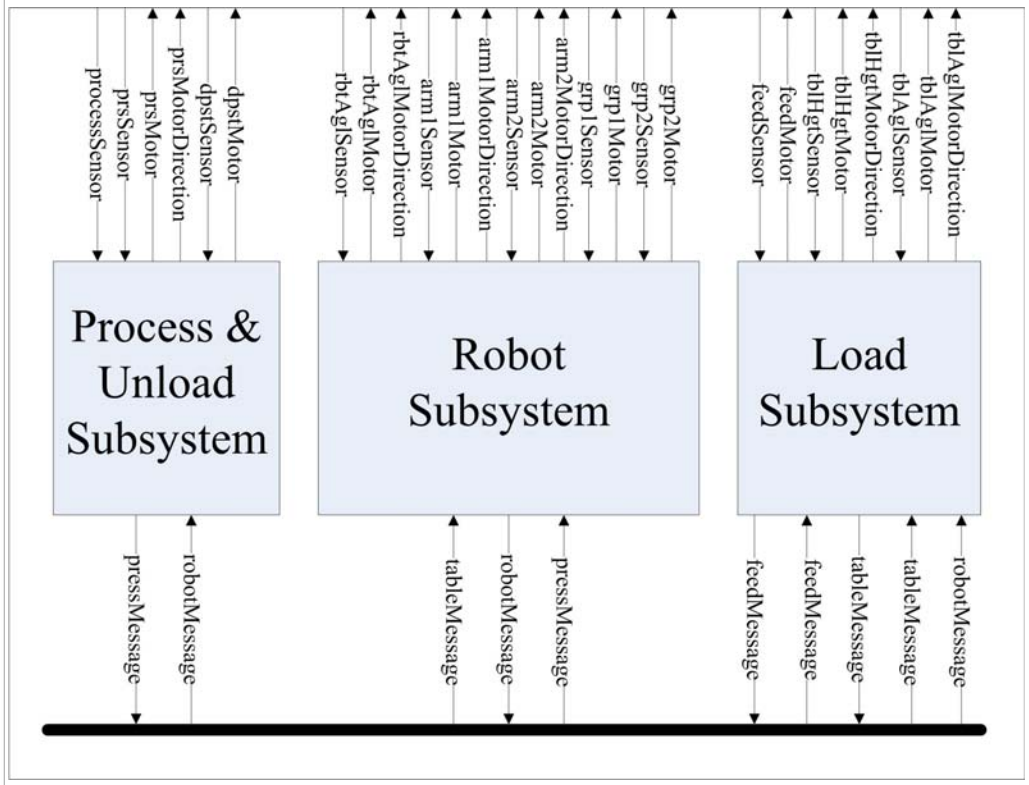
Nicolae Marian

Outline

- COMDES-II design aspects
- Modelling the Production Cell case study in Uppaal
- Property assesment

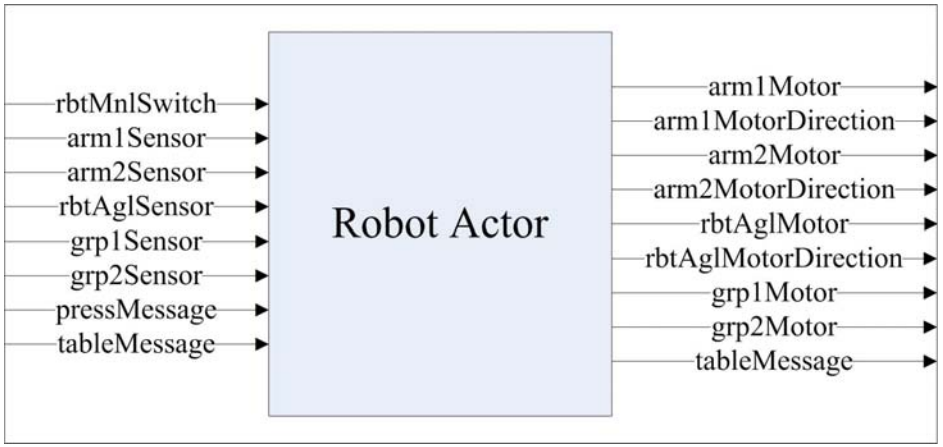
COMDES-II Specification of the Production Cell control system



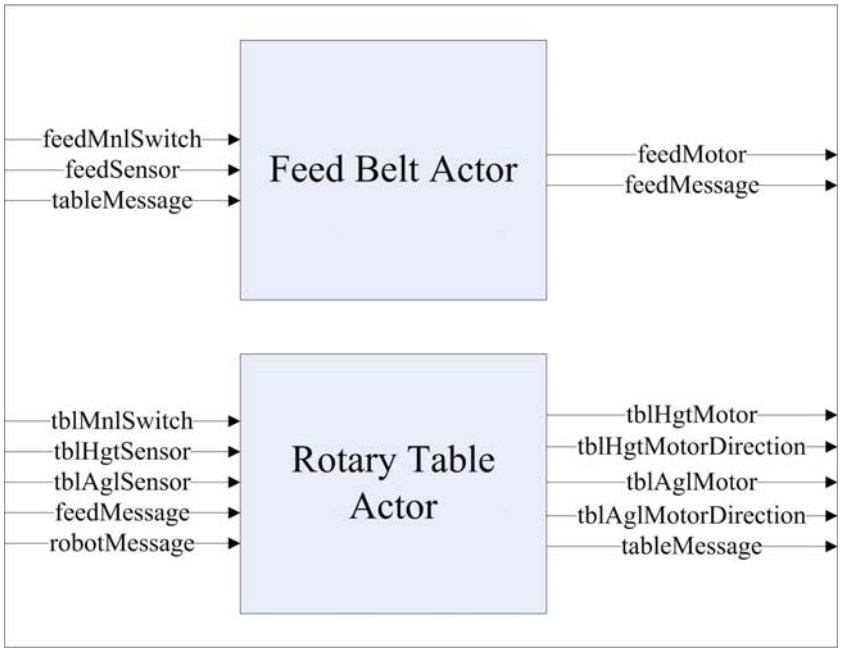


Subsystem specification

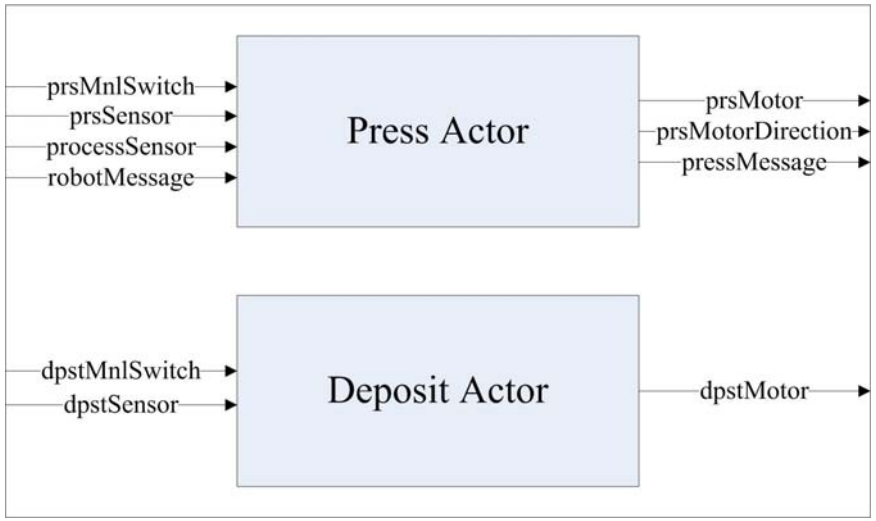
Robot Subsystem



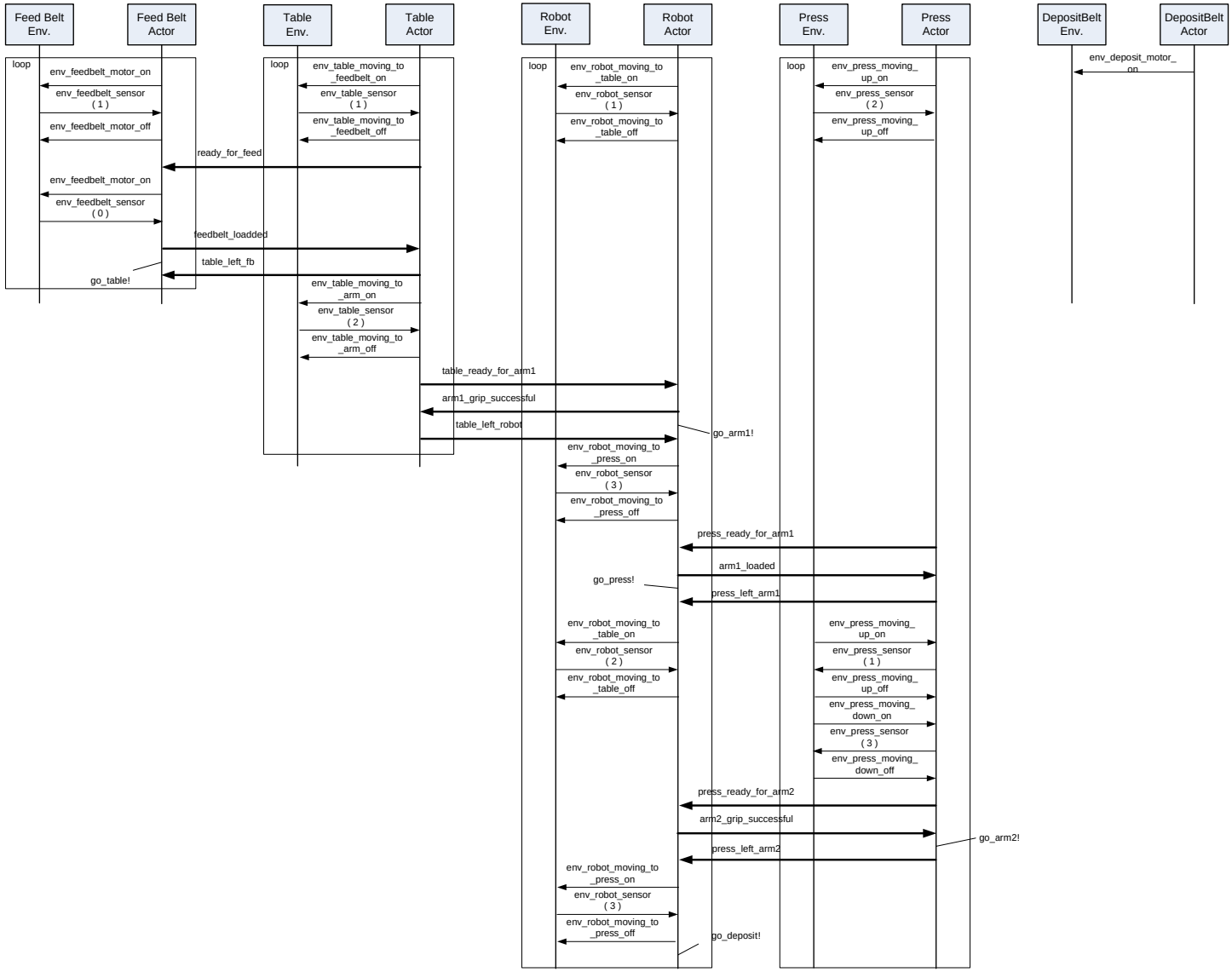
Load Subsystem



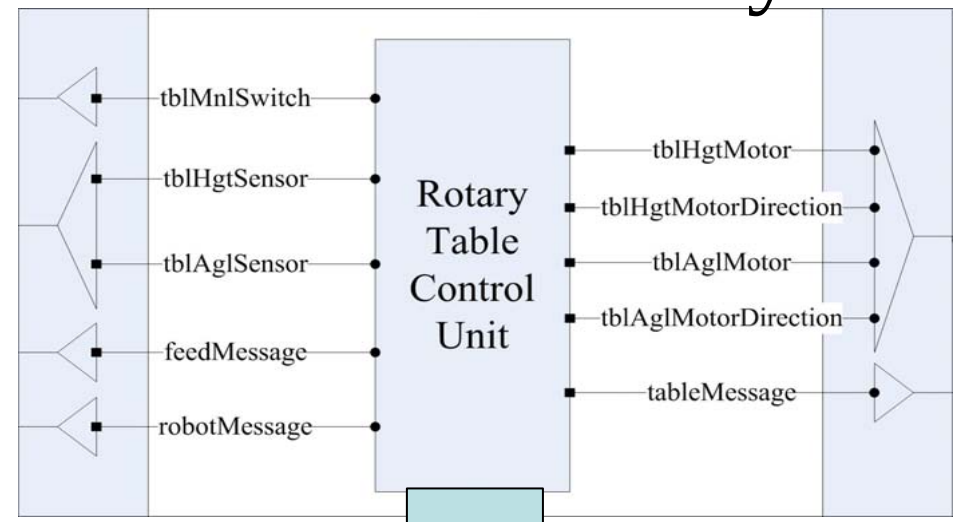
Process & Unload Subsystem

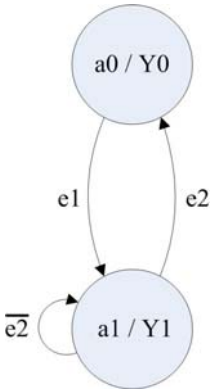


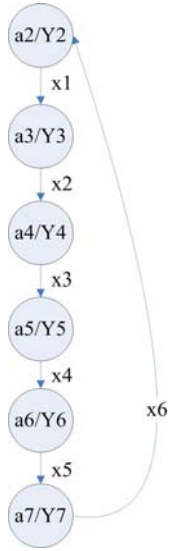
Actor interaction diagram

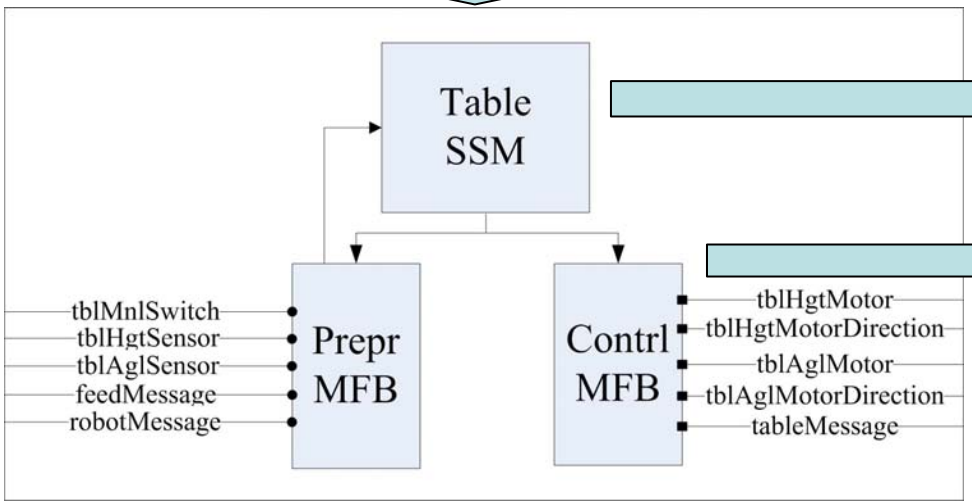


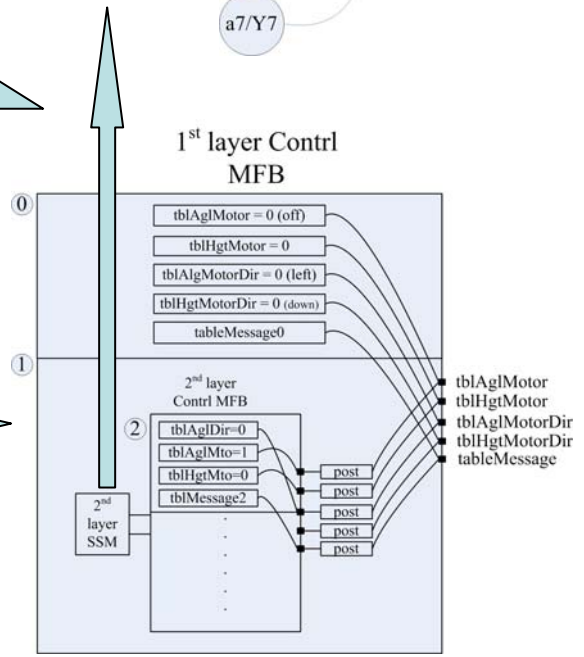
Rotary Table Actor



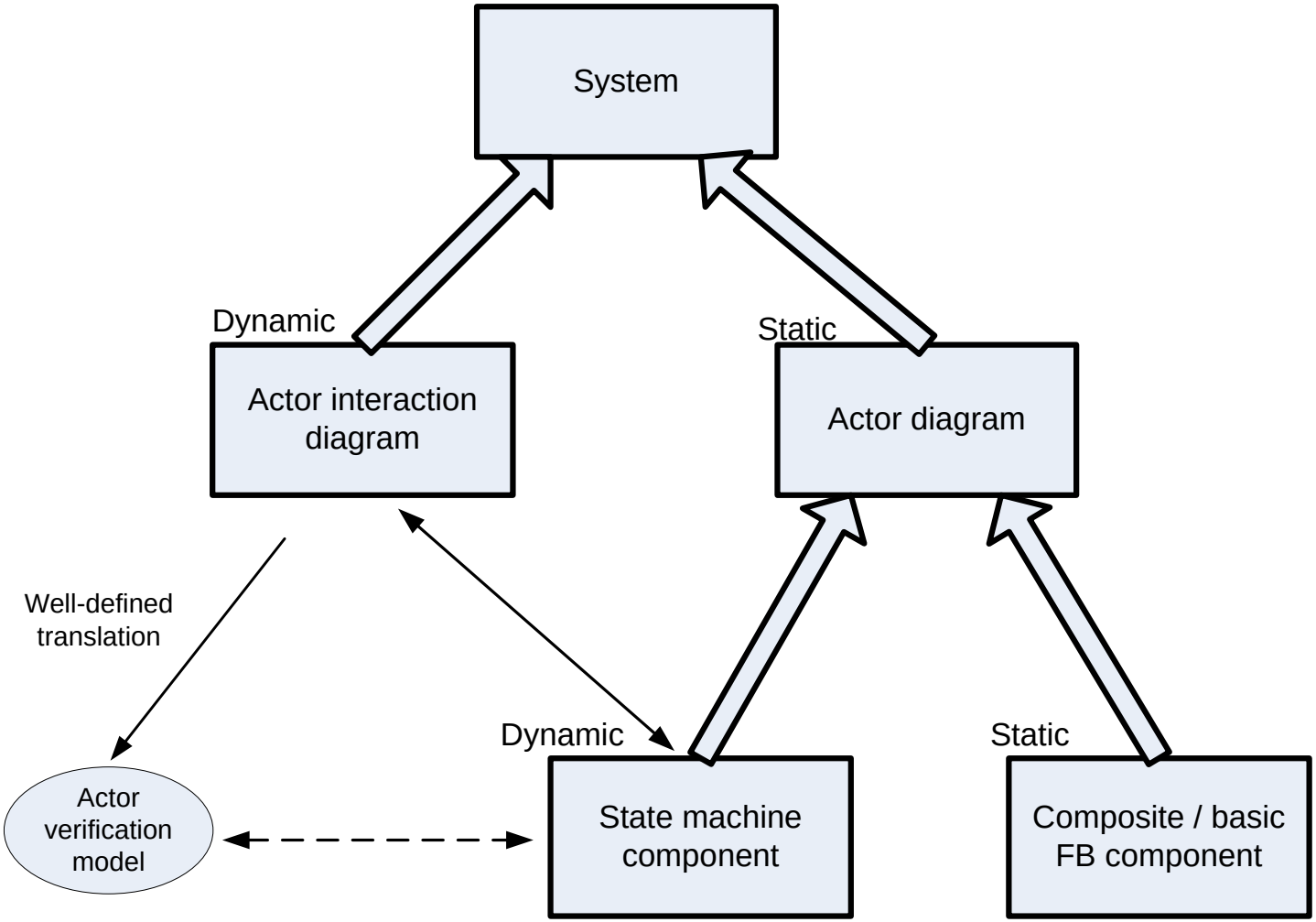








COMDES-II framework



Dynamic Aspect of COMDES-II: Actor interaction diagram

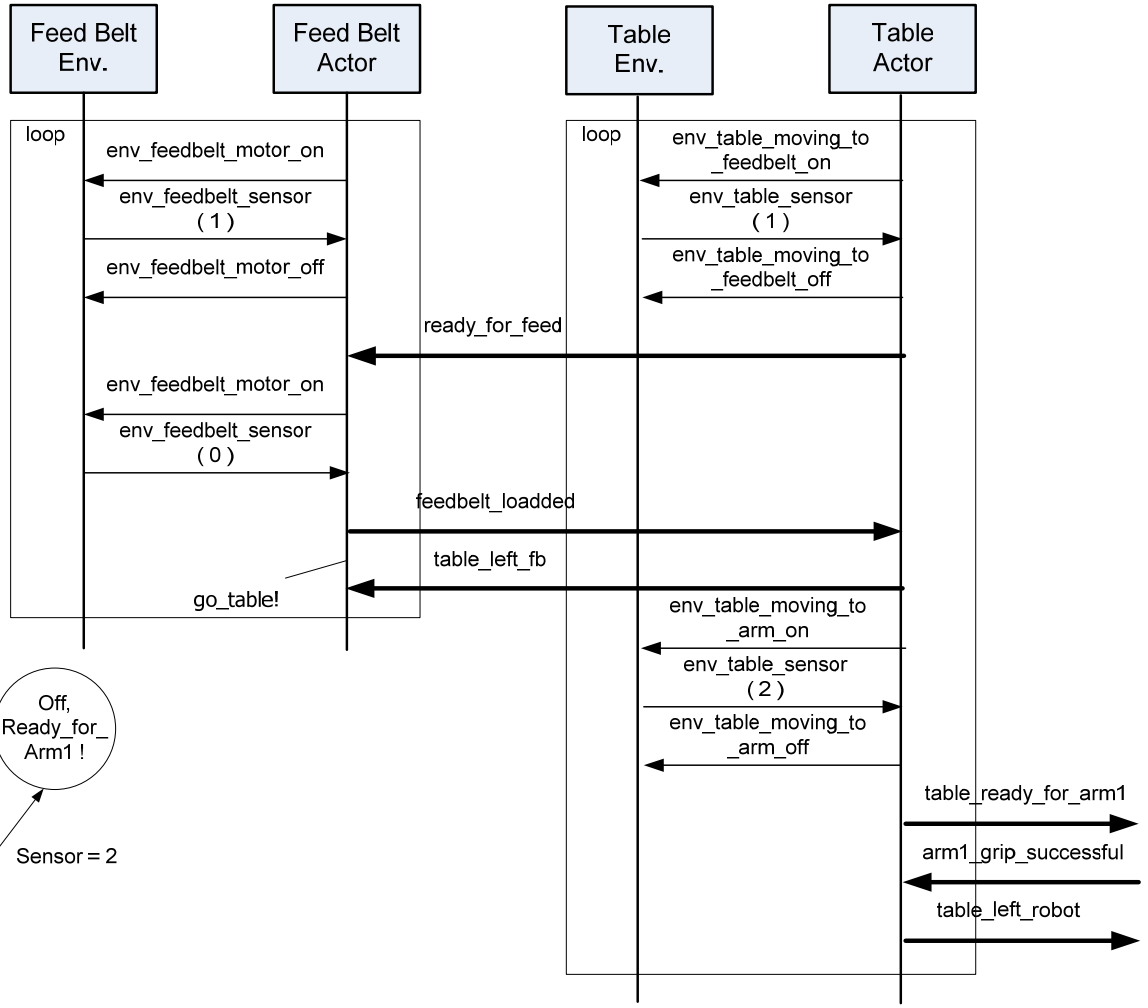
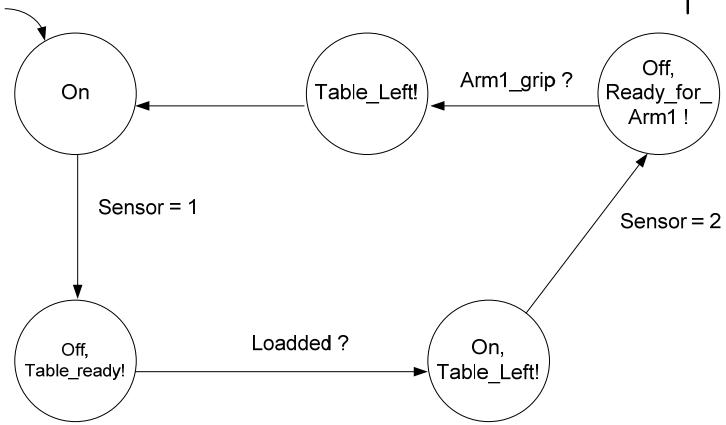


Table actor



Graph transformation of Actor interaction diagram to Uppaal model

- Sensor Signal (trigger from env. to actor) : → variable:

```
int [0,1] env_feedbelt_sensor;
```

Variable included only in a guard.

The environment will set this variable and the controller will read it.

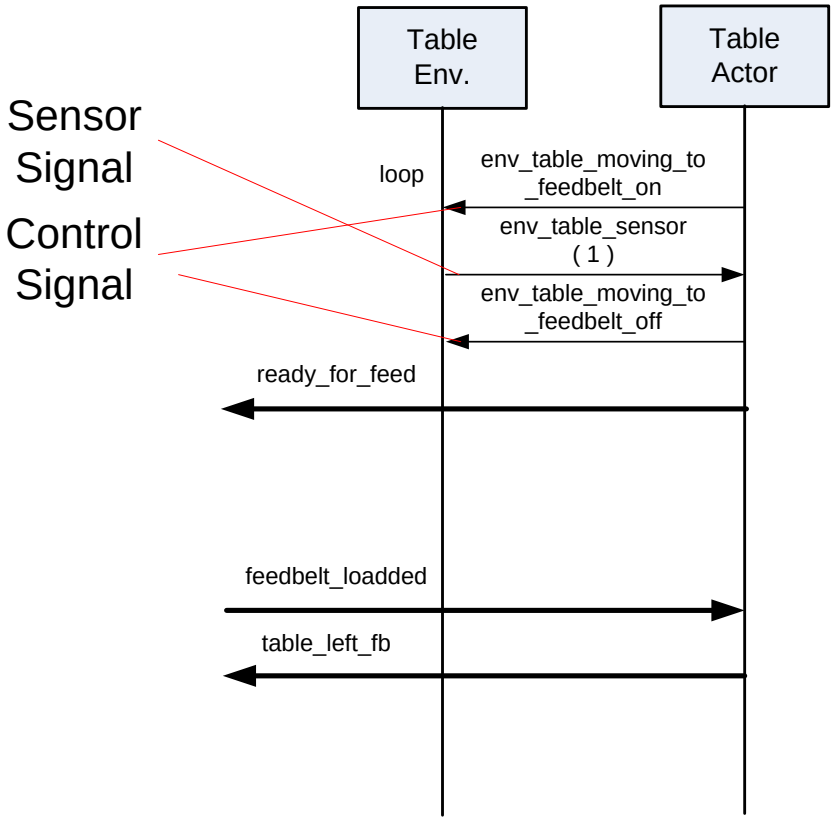
No synchronization is necesarry here.

- Control Signal (reponse from actor to env.) : → broadcast channel:

```
urgent broadcast chan env_feedbelt_motor_on;
```

The controller will issue it.

The Environment waits in a state to receive the control signal. After it gets the control signal, the state of the environment could be changed.



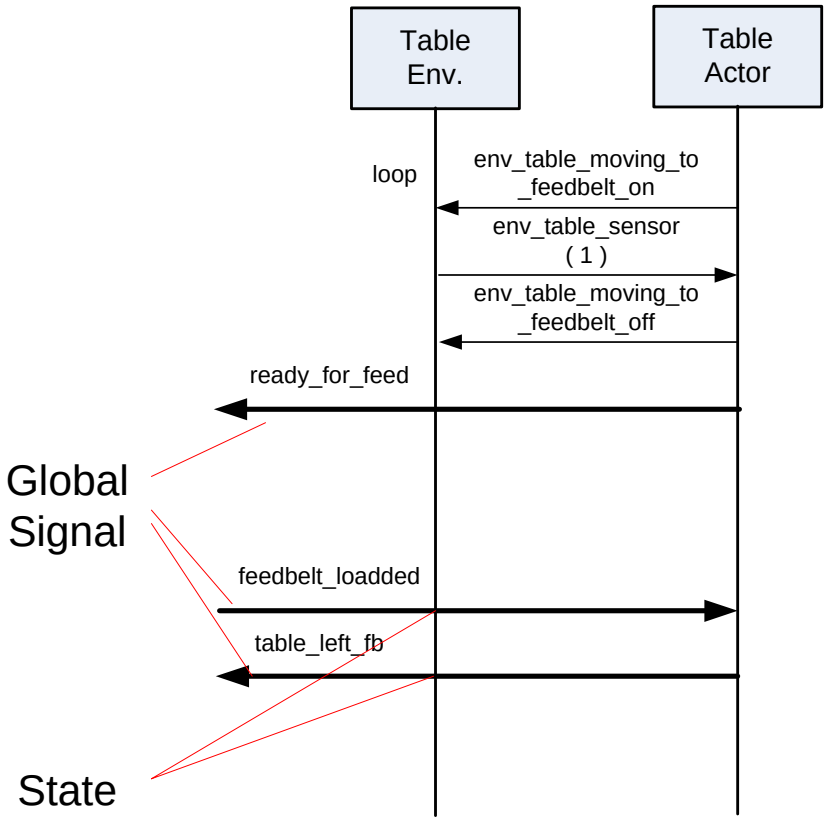
Graph transformation of Actor interaction diagram to Uppaal model cont'd

- Global Signal (pair interaction from actor to actor) :
  → channel:

```
urgent chan loaded,
            ready_for_feed,
            table_left_fb;
```

  Both sender and receiver are blocked if they are not at the rendezvous points.

- States: steps between sensor signal, control signal and global signal.



Priority

- In the real life, the controller is much faster than the environment, because the controller has to react as soon as it gets input from the environment.

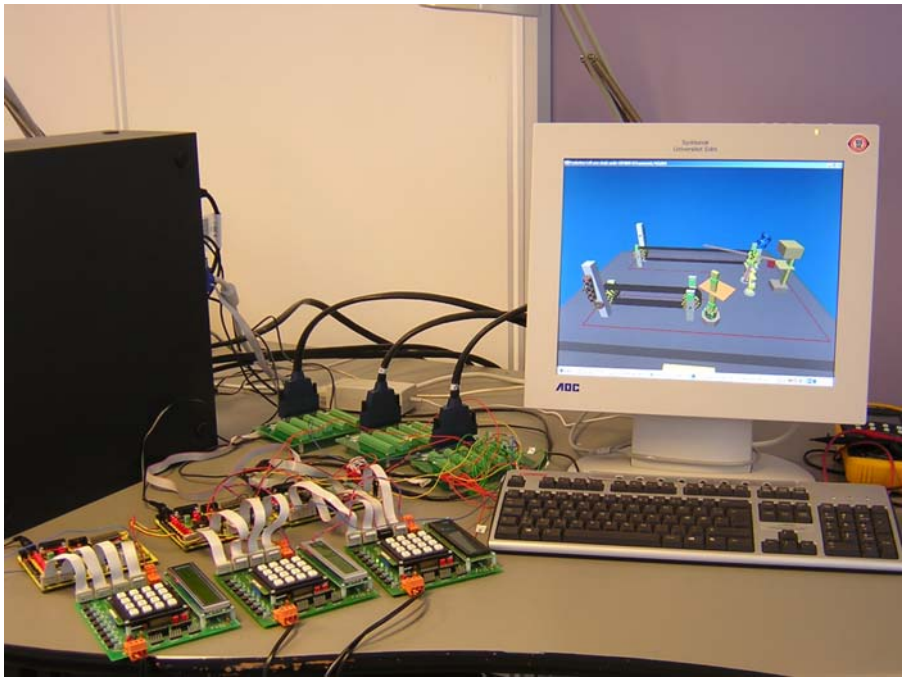
- To model this feature in Uppaal, the controllers will have the highest priority in the system.

```
system  brick1, brick2, brick3, brick4, brick5
        < Env_FeedBelt, Env_Table
        < FeedBeltController, TableController;
```

Production Cell

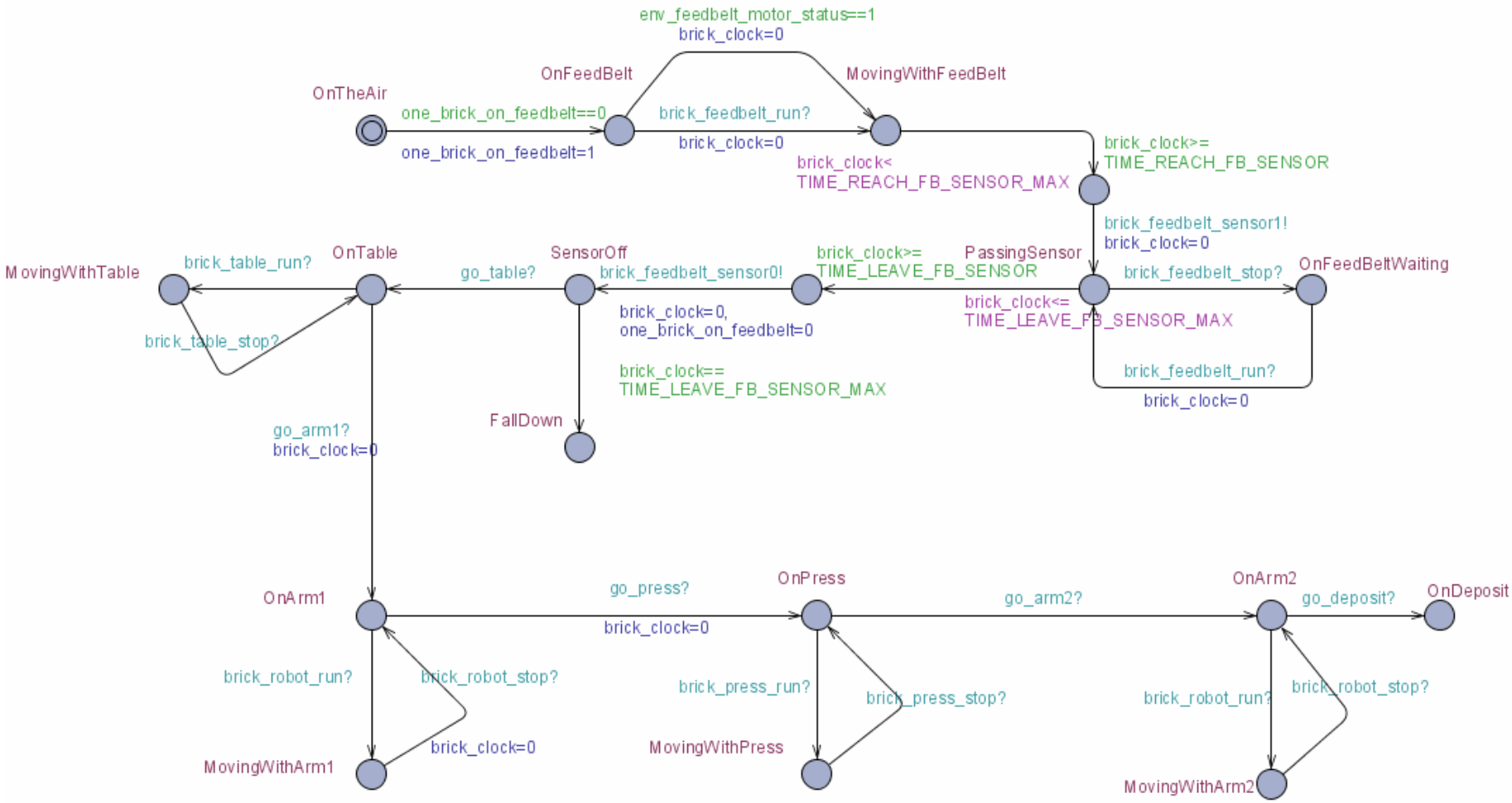
- Environment:
  - Brick
  - Feed Belt
  - Table
  - Robot
  - Press
  - Deposit Belt
- Actor (Controller)
  - Feed Belt
  - Table
  - Robot
  - Press
  - DepositBelt
- Properties
  - safety
  - liveness

$C \parallel E \models \Phi$

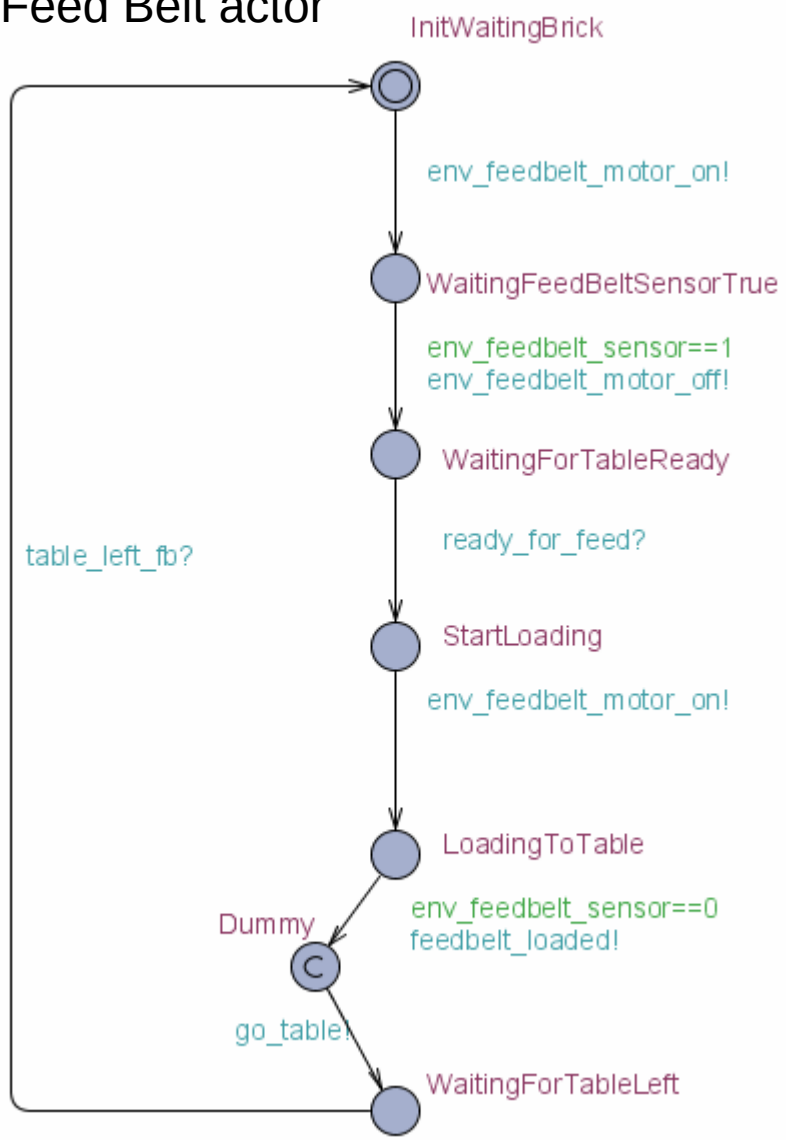


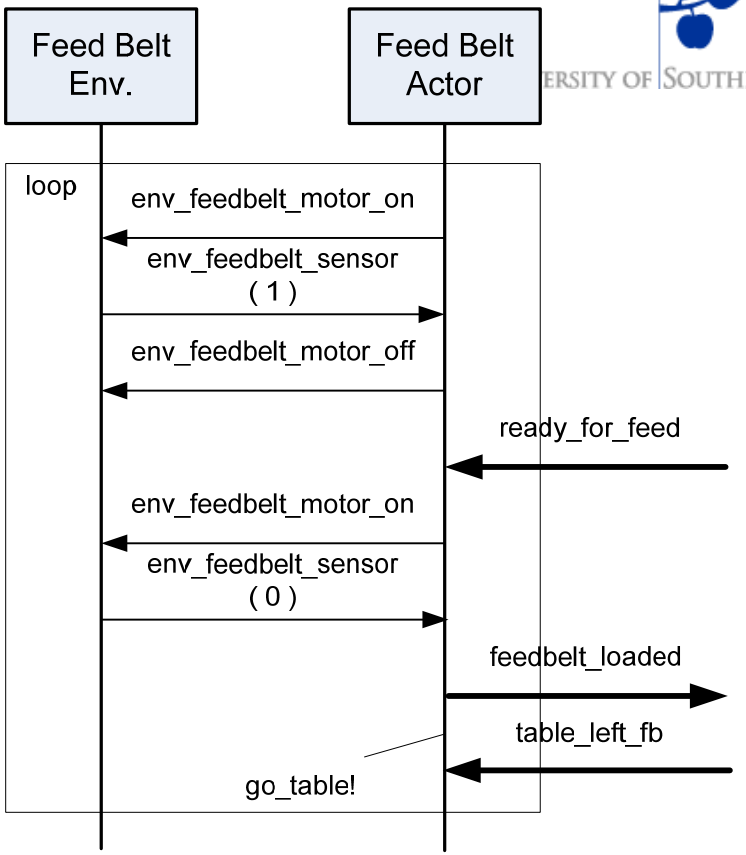
- Channels: 17
- Broadcast channels: 31
- Global variables: 8
- Local variables: 3
- Local clocks: 4
- Locations: 191

Brick Environment



Feed Belt actor





Feed Belt environment

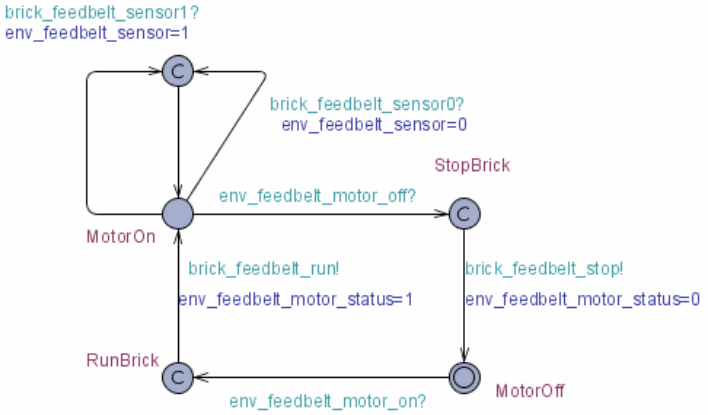


Table Environment

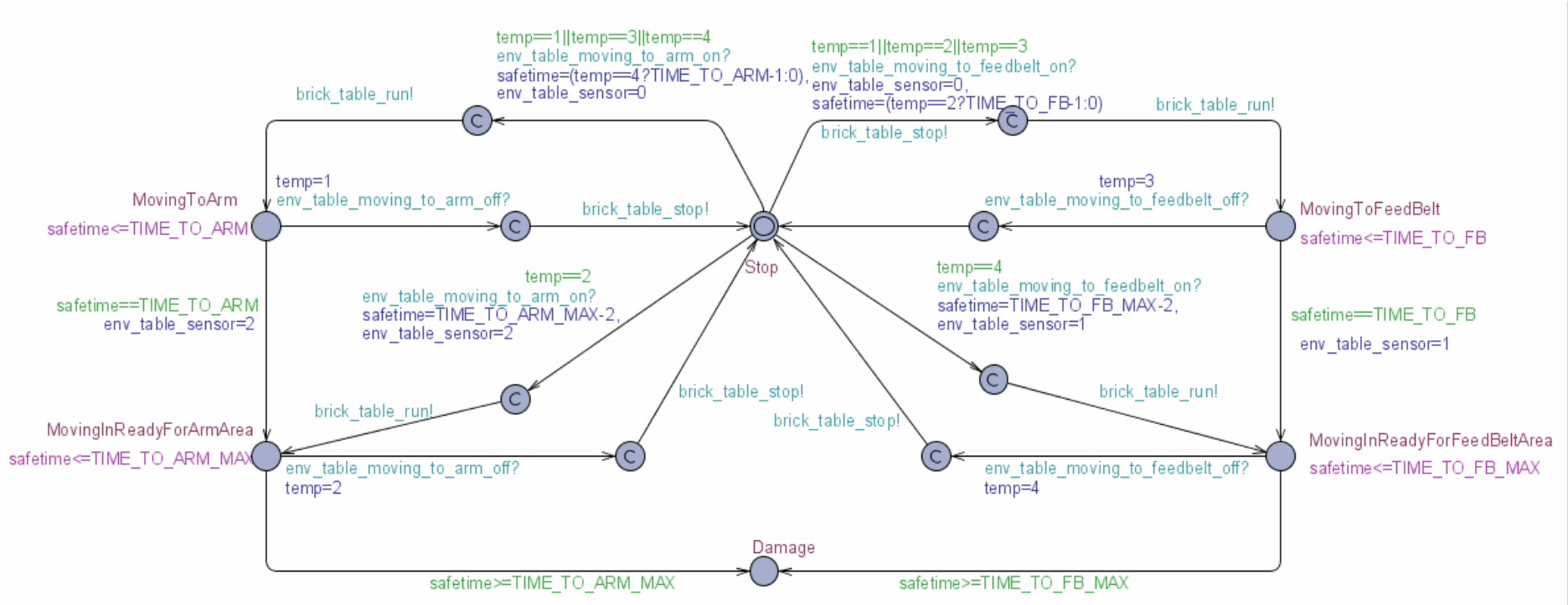
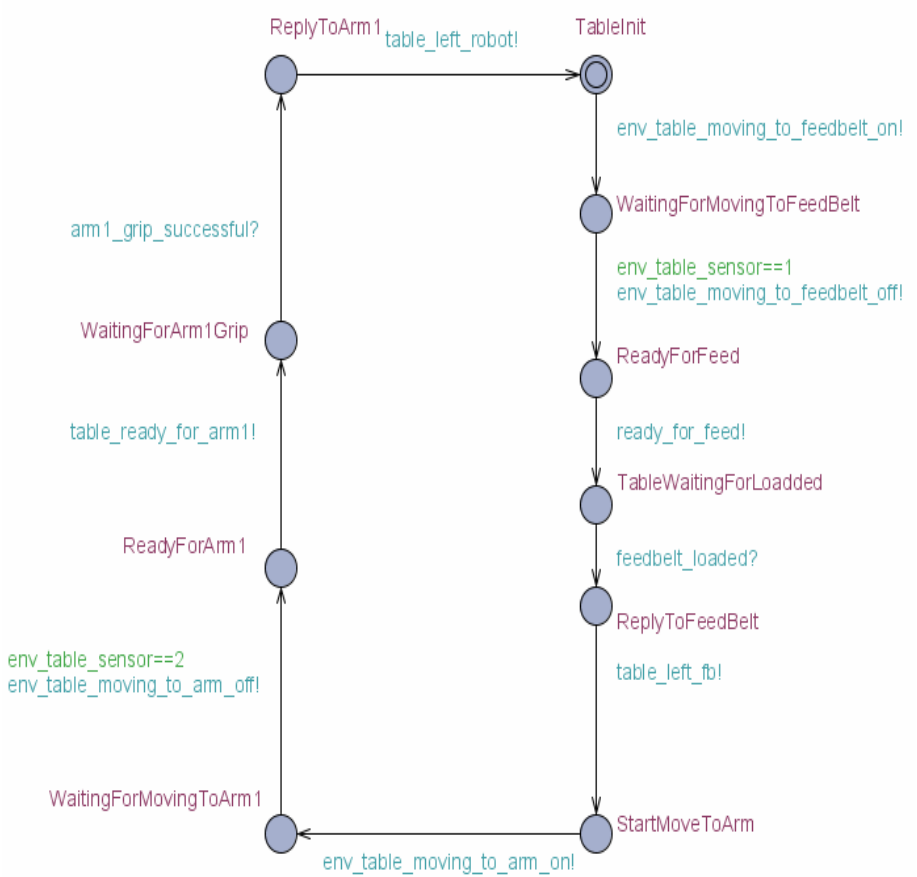


Table Actor


ReplyToArm1
table_left_robot!
TableInit
env_table_moving_to_feedbelt_on!
WaitingForMovingToFeedBelt
arm1_grip_successful?
env_table_sensor==1
env_table_moving_to_feedbelt_off!
WaitingForArm1Grip
ReadyForFeed
table_ready_for_arm1!
ready_for_feed!
TableWaitingForLoadded
ReadyForArm1
feedbelt_loaded?
ReplyToFeedBelt
env_table_sensor==2
env_table_moving_to_arm_off!
table_left_fb!
WaitingForMovingToArm1
StartMoveToArm
env_table_moving_to_arm_on!

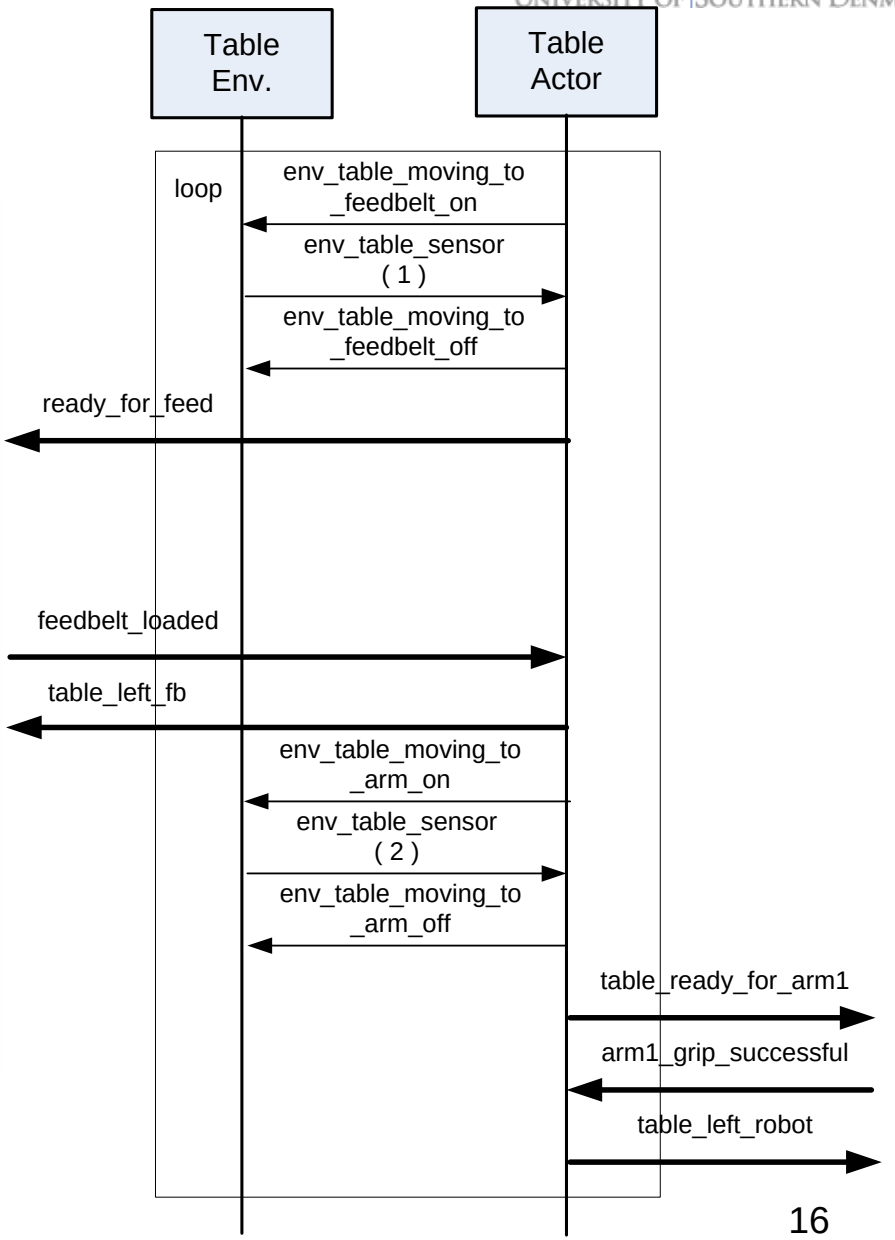
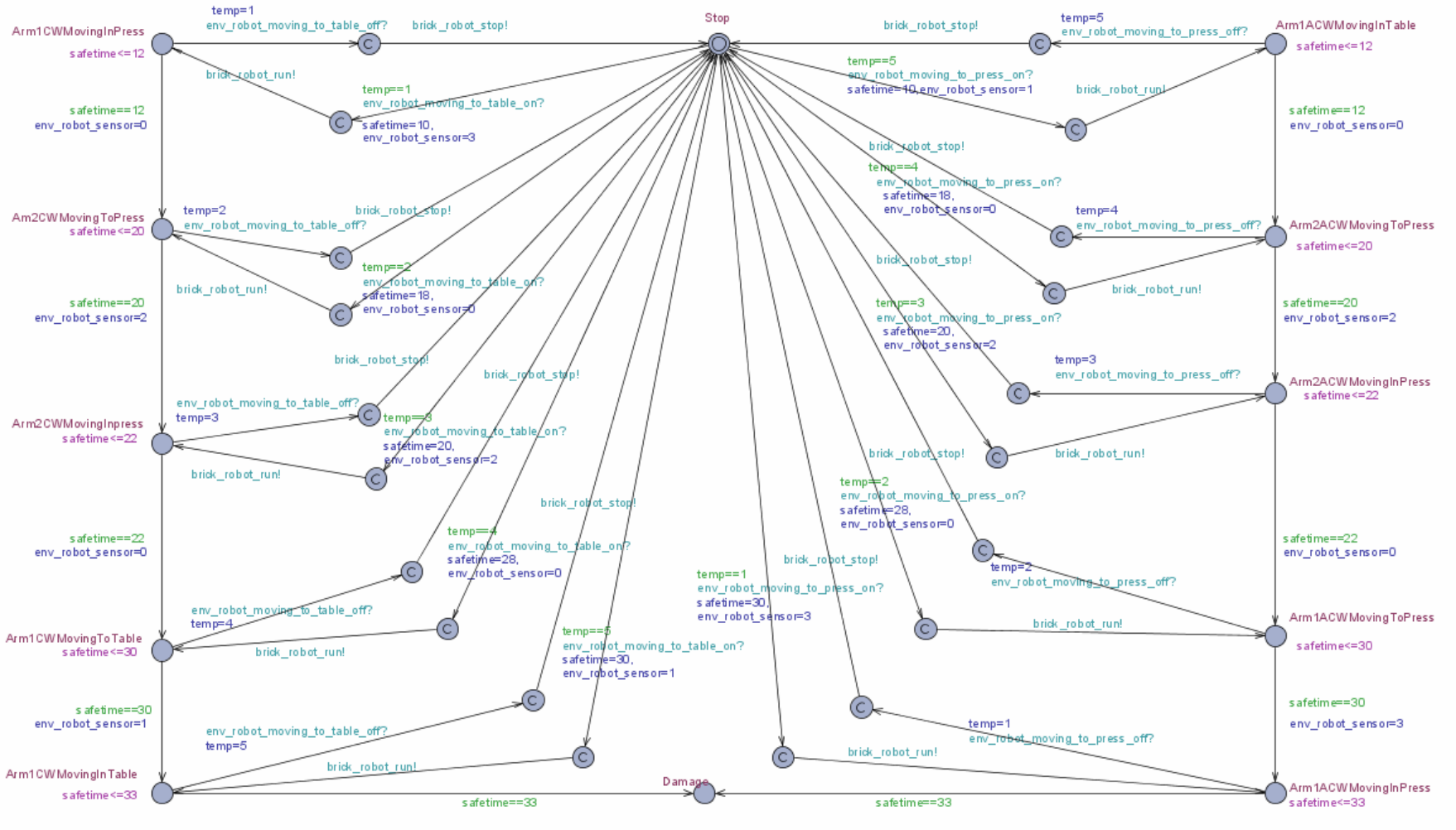


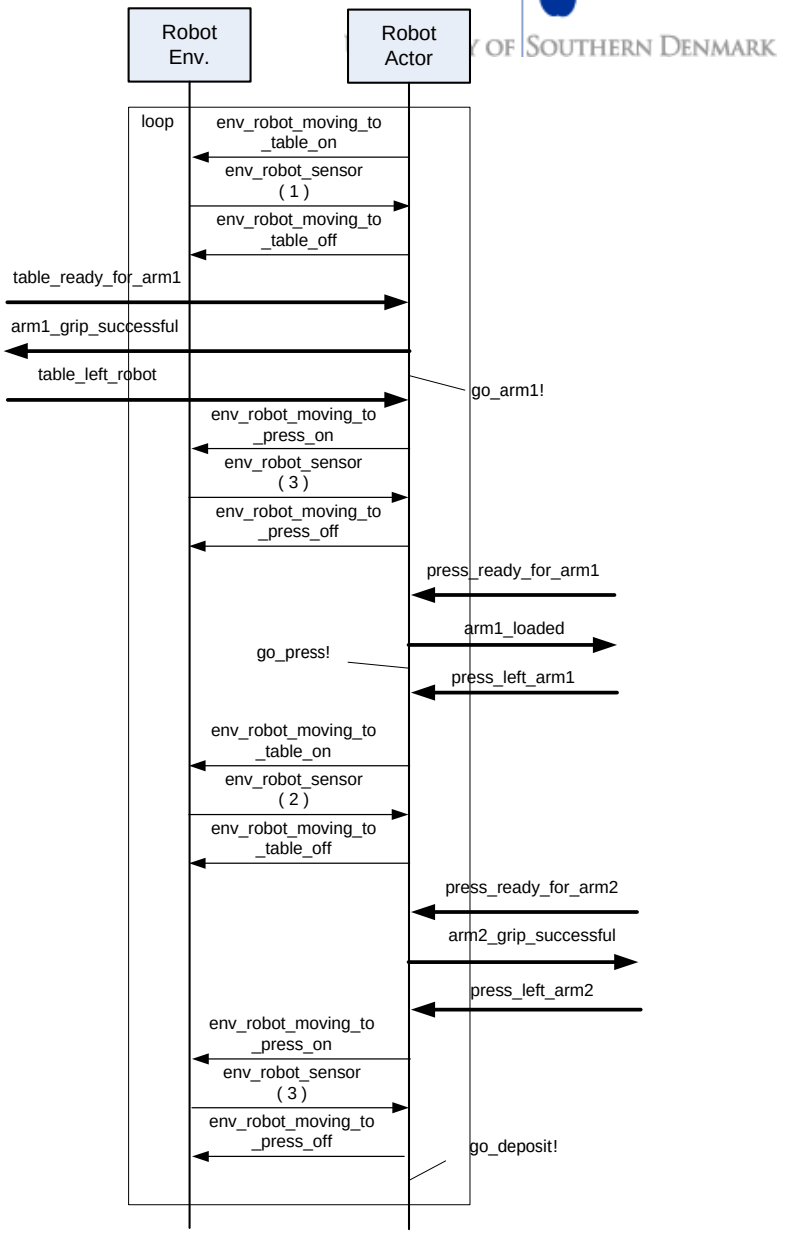
Table Env.
Table Actor
loop
env_table_moving_to_feedbelt_on
env_table_sensor (1)
env_table_moving_to_feedbelt_off
ready_for_feed
feedbelt_loaded
table_left_fb
env_table_moving_to_arm_on
env_table_sensor (2)
env_table_moving_to_arm_off
table_ready_for_arm1
arm1_grip_successful
table_left_robot

Robot Environment



Robot Actor


Arm2LoadingToDeposit
Dummy3
env_robot_moving_to_press_off!
env_robot_sensor==3
WaitingForArm2MovingToDeposit
go_deposit!
env_robot_moving_to_press_on!
StartArm2MoveToDeposit
RobotInit
press_left_arm2?
env_robot_moving_to_table_on!
WaitingForPressLeftArm2
WaitingForArm1MovingToTable
arm2_grip_successful!
env_robot_sensor==1
env_robot_moving_to_table_off!
Arm2GripingFromPress
Arm1ReadyForTable
press_ready_for_arm2?
table_ready_for_arm1?
Arm2ReadyForPress
Arm1GripingFromTable
env_robot_sensor==2
env_robot_moving_to_table_off!
Dummy1
arm1_grip_successful!
WaitingForArm2MovingToPress
go_arm1!
WaitingForTableLeft
env_robot_moving_to_table_on!
table_left_robot?
StartArm2MoveToPress
StartArm1MoveToPress
press_left_arm1?
env_robot_moving_to_press_on!
WaitingForTableLeftArm1
go_press!
WaitingForArm1MovingToPress
Dummy2
env_robot_sensor==3
arm1_loaded!
env_robot_moving_to_press_off!
Arm1ReadyForPress
Arm1LoadingToPress
press_ready_for_arm1?


Robot Env.
Robot Actor
loop
env_robot_moving_to_table_on
env_robot_sensor (1)
env_robot_moving_to_table_off
table_ready_for_arm1
arm1_grip_successful
table_left_robot
go_arm1!
env_robot_moving_to_press_on
env_robot_sensor (3)
env_robot_moving_to_press_off
press_ready_for_arm1
arm1_loaded
go_press!
press_left_arm1
env_robot_moving_to_table_on
env_robot_sensor (2)
env_robot_moving_to_table_off
press_ready_for_arm2
arm2_grip_successful
press_left_arm2
env_robot_moving_to_press_on
env_robot_sensor (3)
env_robot_moving_to_press_off
go_deposit!

Case study requirements

- Safety requirements
  - R1: Keep blanks sufficiently distant

    Do not put blanks on the table, if it is already loaded,
  - R2: The limitations of machine mobility

    The elevating rotary table must not be moved downward, if bottom sensor is true, and it must not be moved upward, if top sensor is true,
  - R3: The avoidance of machine collisions

    The robot, for instance, would collide with the press if arm 1 would extend too far while pointing towards the press;

- Liveness requirements

  Every brick introduced into the system via the feed belt will eventually be put on the deposit belt.

- Claim: controller correctness is implied by the correct behaviour of bricks and environments

  $\Rightarrow$ safety & liveness properties are specified using elements of brick & environment processes

Reachability and Safety

- Is it possible that one brick is moving with table and another is moving with feedbelt? (Reachability)

```
E<> ( brick1.MovingWithTable and brick2.MovingWithFeedBelt )
```

- The table with the brick on it can not move down to the feedbelt. (Safety)

```
A[] not (brick1.MovingWithTable and
Env_Table.MovingToFeedBelt)
```

R1: Keep blanks sufficiently distant

- The bricks can not overtake each other, i.e. it is necessary to show that their locations in the system are occupied under mutual exclusion

- Only one brick at a time can be on the table.

```
A[] (brick1.OnTable imply not
     (brick2.OnTable or brick3.OnTable or
      brick4.OnTable or brick5.OnTable))
```

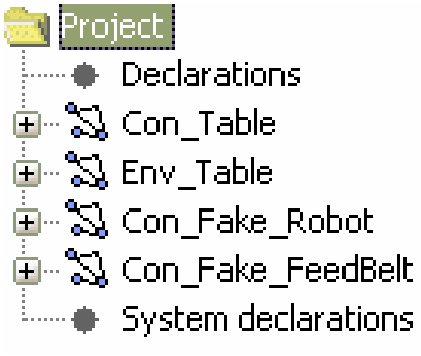
R2: The limitations of machine mobility

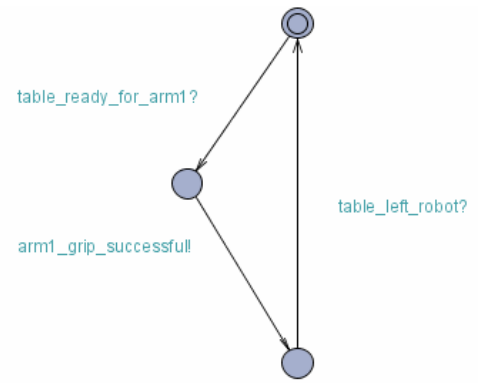
- The table (Robot, Press) never enters Damage state.

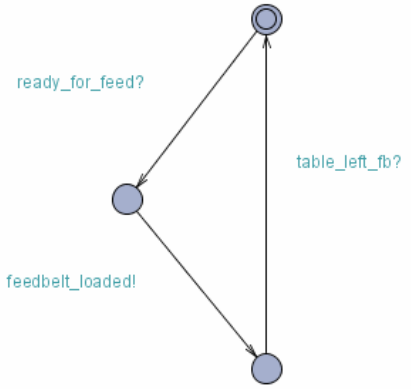
```
A[] not Env_Table.Damage
A[] not Env_Robot.Damage
A[] not Env_Press.Damage
```

- Property Composition

```
A[] not ( Env_Table.Damage or Env_Robot.Damage or
   Env_Press.Damage )
≡
A[] not Env_Table.Damage AND A[] not Env_Robot.Damage AND A[]
   not Env_Press.Damage
```







R3: The avoidance of machine collisions

- Arm1 can not wait in the press area when press is moving up/down close to the process area

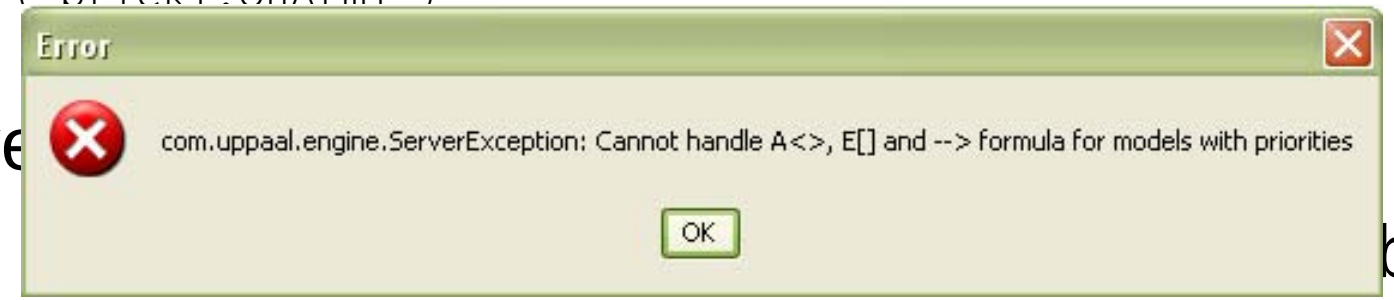
```
A[] not (Env_Robot.Stop and Env_Robot.temp==1
         and Env_Press.DMovingInProcessArea)

A[] not (Env_Robot.Stop and Env_Robot.temp==1
         and Env_Press.UMovingInProcessArea)
```

Liveness Properties

- Liveness : Q is eventually satisfied. A<>Q

  Brick eventually reaches Arm1.

  `A<> ( brick1.OnArm1 )`



- Live

  If a be picked up by arm1.

  `brick1.OnFeedBelt-->brick1.OnArm1`

Bug found in the Actor Interaction Diagram of the Production Cell

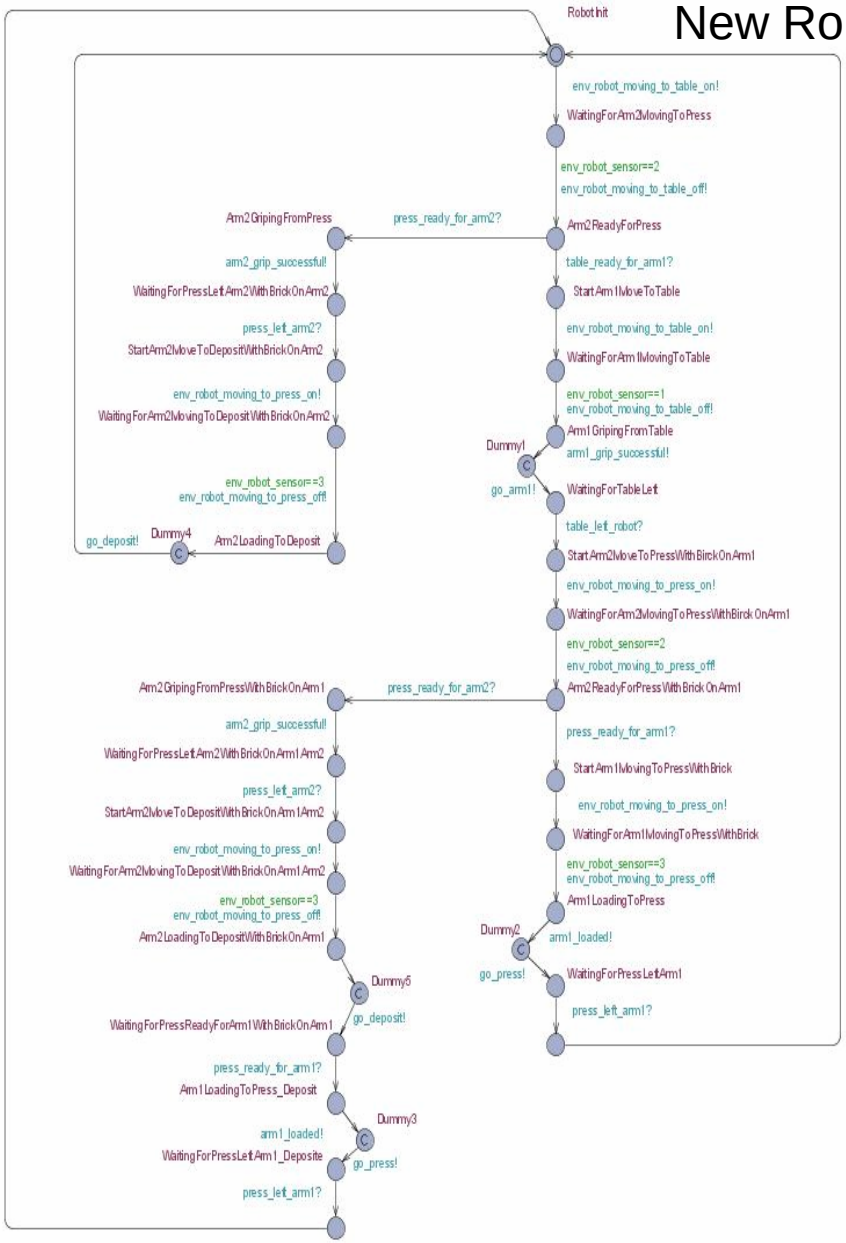
- Is it possible that a brick is moving with arm1 and another is moving with arm2 at the same time?

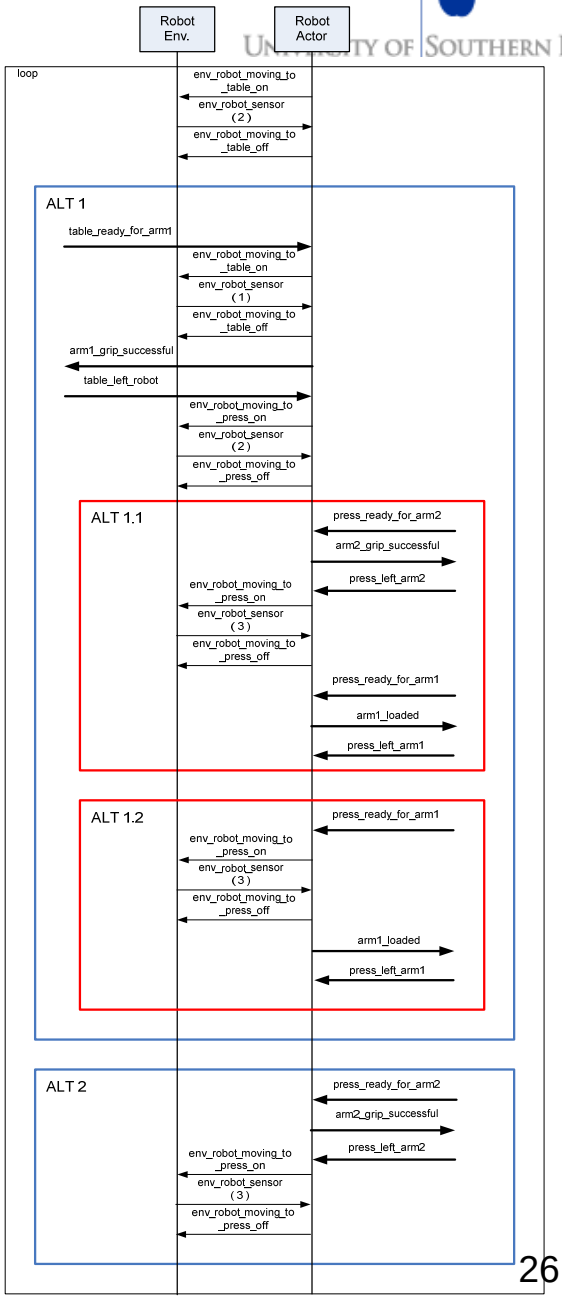
```
E<> ( brick1.MovingWithArm1 and brick2.MovingWithArm2 )
```

- Expected true, actually it is false in this model.

- Reason: according to the current design of the actor interaction diagram, it does not support this feature.

New Robot actor





Conclusions

- The verification model presumes explicit modeling of both environment and actor, within the corresponding plant-controller pairs.

- The global Uppaal model is checked as sub-models in a sequence of chained reactions, concluding the correctness of system behaviour properties from local properties of the plant-controller pairs involved in the transaction.