Formal Methods for Concurrent and Real Time Systems

Class Project

Politecnico di Milano
Dipartimento di Elettronica,
Informazione e Bioingegneria

University of Illinois
at Chicago

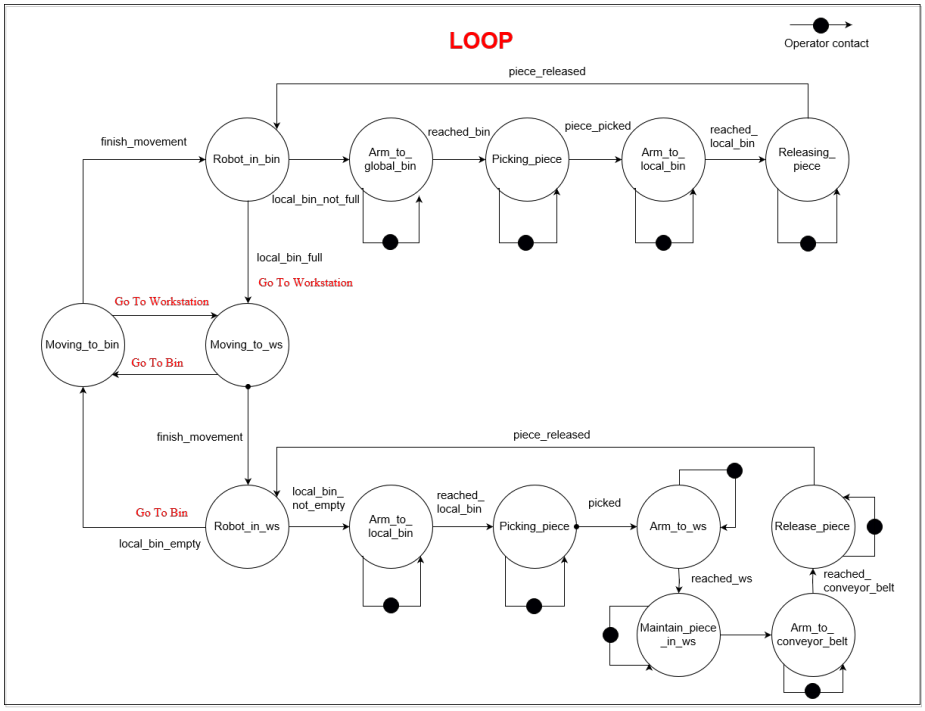
Menchetti Guglielmo - mat. 899865
Montanari Claudio - mat. 899980
Norcini Lorenzo - mat. 882549

Contents

1 Introduction

1.1 Problem Definition

The objective of the following document is to describe a formal model of the interaction of an Human Operator and a Robot.
The Robot to be modelled is analogous to the KUKA KMR-iiwa (`https://www.kuka.com/en-us/products/robotics-systems`) which is a manipulator and mobile robot, used mainly for industrial use: welding, assembly, loading and unloading.
The scenario is one of Robot Human Collaboration where the Robot has to aid the operator in performing a given task.
Specifically the Robot has to bring Work Pieces from a Global Bin to a Work Station.
After being elaborated in some fashion by either additional machinery or the Operator, the pieces are placed on a Conveyor Belt.
Normally the Robot operates in a sort of loop, going back and forth from the Global Bin to the Work Station in order to bring pieces.
It's also possible for the Operator to give the Robot additional commands.
For a more precise description of the Robot tasks it's possible to refer to the graph below.



1.2 Assumptions and Conventions

- All the signals are considered instantaneous and valid only in the time instant in which they are asserted.
- We assume a reasonable behaviour of the operator when using the remote controller. Specifically, no more than one signal can be sent in the same time instant.
- We assume that all the information related to the position of entities are retrieved by sensors deployed in the environment.
- We also assume that the contact of the operator is retrieved by sensors positioned in the robot arm. Specifically, the operator contact signal is active when the robot arm is near to the operator (e.g. less than 5 cm). Furthermore, the operator contact has no effect if it occurs in non-working zones, since the arm is locked in position (retracted).
  Finally, at the end of operator contact, the state of the robot body and arm are in a valid state modeled by the controller.
- For what concerns the specification of time bounded formulas (i.e. $UpToNow_{ie}$) we adopted the following convention: the first subscript refers to the left part of the time interval while the second one refers to the right part of the time interval.
- We assume that all logic formulas are implicitly valid for all time instants (i.e. as they are contained in an $Always()$ predicate)

1.3 Document Structure

The document is structured in 6 main sections describing the main components of the system and their interactions:

- **Environment**: This section describes the Environment in which the Entities operate.
  The Environment is represented as a tiled Map.
  Each Tile of the Map has certain characteristics that are used in modelling the behaviour of the entities.
- **Safety Policy**: This section contains the axioms that define the characteristics that our model has to satisfy in order to guarantee a reasonable degree of safety.
- **Controller**: This section describes the Controller for the Robot Body and Robot Arm behaviour.
  The Controller receives information coming from Operator, Robot Body and Robot Arm.
  The Controller sends the signals that determine the behaviour of the Robot Arm and Robot Body.

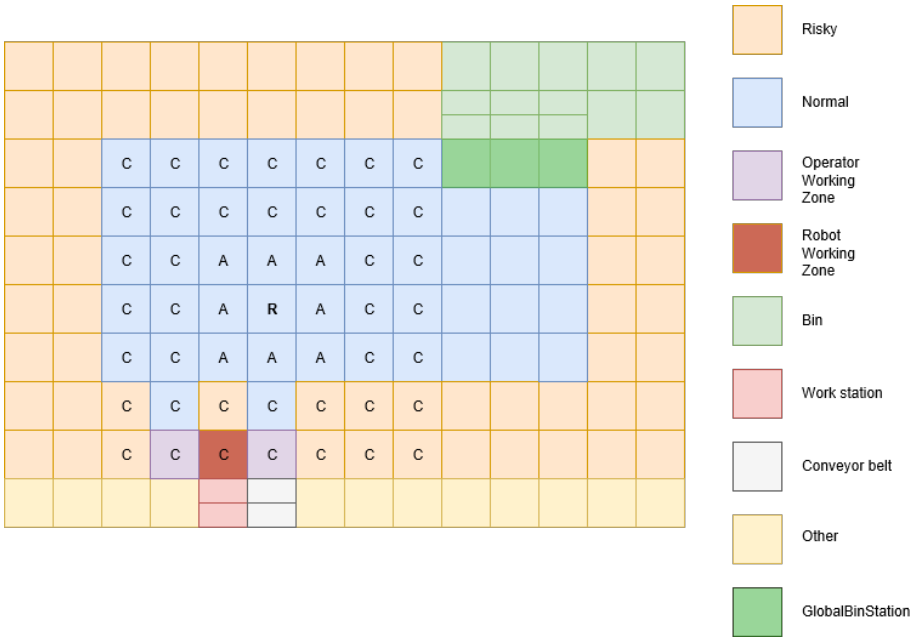
- **RobotBody**: This section describes the characteristics and the behaviour of the RobotBody.
  Such behaviour is modelled by describing the responses of the Robot Body to control signals.

- **RobotArm**: This section describes the characteristics and the behaviour of the RobotArm.
  Such behaviour is modelled by describing the responses of the Robot Arm to control signals.

- **Operator**: This section describes the characteristics and the behaviour of the Operator.
  In this section the behaviour is not modelled in much detail in order to account for different types of Operator behaviour.

2 Environment

The environment where the Robot will be moving has been modeled accordingly to the picture below. Each tile has the dimension of the Robot (almost 1m x 1m), thus the Robot perfectly fits one tile and under simplifying assumptions, also the Operator occupies one tile.
The *Light Blue* tiles (i.e. the ones marked as *Normal*) are regions where the Robot will be moving with less restricted safety policy; while the *Orange* ones are still regions where the Robot will be moving but more carefully since we are close to the walls.
The *Dark Green* tiles instead, will be devoted to the Robot for loading pieces on its local bin; notice that the *Light Green* tiles (in particular the ones that have been split in two parts) are the regions where the Robot arm will go to take the pieces from the global bin.
The *Dark Red* tile is where the Robot will stay when working at the working station, whereas the *Light Purple* next to it are the tiles where the Operator will stay when collaborating with the Robot. Again, the tiles split in two parts in front of the *Dark Red* (i.e. *Conveyorbelt* and *Workstation*) are the regions where the arm of the robot may be when in the working station.
For clarity, notice that the *Orange* tile next to the *Workstation* tile has been intentionally positioned there so that, when the Robot has to leave the working station and it has the Operator adjacent, it is forced to maximize the distance from the operator going in a tile marked as *Normal*.



2.0.1 Constants and domain

- **Map**: Tile[0], . . . , Tile[N]
- **TileType**: { risky, normal, operatorWorkingZone, robotWorkingZone, bin, workStation, conveyorBelt, globalBinStation }
- **TileWalkability**: { walkable, notWalkable }
- **Entity**: { robotBody, robotArm, operator }
- **N**: integer

2.1 Predicates

What follows is the formalization in terms of logic predicates of the Environment where the Robot and the Operator will work.
First we define the notion of distance with respect to the concept of Tile and we also define the main feature of a Tile. Finally, we extend the notion of distance with respect to the Entities.
The following definitions have to be considered valid for all time instant.

2.1.1 Maps predicates

Left(Tile[i], Tile[j])
True if and only if Tile[j] is on the left side of Tile[i]:

$$\begin{aligned} &\forall\ i,j\ Left(Tile[i], Tile[j]) \\ &\iff \\ &j = (i-1)\ \wedge \\ &(i-1) = (i-1)\ mod\ (\lceil \frac{j}{J} \rceil \cdot J)) \end{aligned} \tag{1}$$

Right(Tile[i], Tile[j])
True if and only if Tile[j] is on the right side of Tile[i]:

$$\begin{aligned} &\forall\ i,j\ Right(Tile[i], Tile[j]) \\ &\iff \\ &j = (i+1)\ \wedge \\ &(i+1) = (i+1)\ mod\ (\lceil \frac{j}{J} \rceil \cdot J)) \end{aligned} \tag{2}$$

Up(Tile[i], Tile[j])
True if and only if Tile[j] is on the upper side of Tile[i]:

$$\begin{aligned} &\forall\ i,j\ Up(Tile[i], Tile[j]) \\ &\iff \\ &j = (i-J)\ \wedge \\ &(i-J) > 0) \end{aligned} \tag{3}$$

Down(Tile[i], Tile[j])
True if and only if Tile[j] is on the lower side of Tile[i]:

$$\begin{aligned} &\forall\ i,j\ Down(Tile[i], Tile[j]) \\ &\iff \\ &j = (i+J)\ \wedge \\ &(i+J) < N \end{aligned} \tag{4}$$

LeftUp(Tile[i], Tile[j])
True if and only if Tile[j] is on the upper-left side of Tile[i]:

$$\begin{aligned}
&\forall\ i, j\ LeftUp(Tile[i], Tile[j]) \\
&\iff \\
&j = (i - J - 1)\ \wedge \\
&(i - J - 1) = (i - J - 1)\ mod\ (\lceil \frac{j}{J} \rceil) \cdot J\ \wedge \\
&(i - J - 1) > 0)
\end{aligned} \tag{5}$$

LeftDown(Tile[i], Tile[j])
True if and only if Tile[j] is on the lower-left side of Tile[i]:

$$\begin{aligned}
&\forall\ i, j\ LeftDown(Tile[i], Tile[j]) \\
&\iff \\
&(j = (i + J - 1)\ \wedge \\
&(i + J - 1) = (i + J - 1)\ mod\ (\lceil \frac{j}{J} \rceil \cdot J)\ \wedge \\
&(i + J - 1) < N)
\end{aligned} \tag{6}$$

RightUp(Tile[i], Tile[j])
True if and only if Tile[j] is on the upper-right side of Tile[i]:

$$\begin{aligned}
&\forall\ i, j\ RightUp(Tile[i], Tile[j]) \\
&\iff \\
&j = (i - J + 1)\ \wedge \\
&(i - J + 1) = (i - J + 1)\ mod\ (\lceil \frac{j}{J} \rceil \cdot J)\ \wedge \\
&(i - J + 1) > 0
\end{aligned} \tag{7}$$

RightDown(Tile[i], Tile[j])
True if and only if Tile[j] is on the lower-right side of Tile[i]:

$$\begin{aligned}
&\forall\ i, j\ RightDown(Tile[i], Tile[j]) \\
&\iff \\
&j = (i + J + 1)\ \wedge \\
&(i + J + 1) = (i + J + 1)\ mod\ (\lceil \frac{j}{J} \rceil \cdot J)\ \wedge \\
&(i + J + 1) < N
\end{aligned} \tag{8}$$

Adjacent(Tile[i], Tile[j])
True if and only if Tile[j] and Tile[i] are adjacent:

$$\begin{aligned}
&\forall\, i, j :\ i \neq j,\ Adjacent(Tile[i], Tile[j]) \\
&\iff \\
&Up(Tile[i], Tile[j]) \vee \\
&Down(Tile[i], Tile[j]) \vee \\
&Left(Tile[i], Tile[j]) \vee \\
&Right(Tile[i], Tile[j]) \vee \\
&LeftUp(Tile[i], Tile[j]) \vee \\
&RightUp(Tile[i], Tile[j]) \vee \\
&LeftDown(Tile[i], Tile[j]) \vee \\
&RightDown(Tile[i], Tile[j])
\end{aligned} \tag{9}$$

AreAtDistance(Tile[i], Tile[j], N)
True if and only if Tile[j] and Tile[i] shares a Tile[k] that is at distance $(N-1)$ to Tile[i] and adjacent to Tile[j] or, if N = 1, if the two tiles are adjacent:

$$\begin{aligned}
&\forall\, i, j :\ i \neq j,\ AreAtDistance(Tile[i], Tile[j], N) \\
&\iff \\
&(N > 1 \wedge \\
&\quad \exists k\ :\ AreAtDistance(Tile[i], Tile[k], N-1) \wedge \\
&\qquad Adjacent(Tile[j], Tile[k]) \wedge \\
&\qquad\quad \neg Adjacent(Tile[i], Tile[j])) \\
&\vee \\
&(N = 1 \wedge Adjacent(Tile[i], Tile[j])
\end{aligned} \tag{10}$$

HasType(Tile[i])
Returns the $TileType$ value of a Tile[i]:

$$\forall\, i,\ HasType(Tile[i]) = t \in TileType \tag{11}$$

HasWalkability(Tile[i])
Returns the $Walkability$ value of a Tile[i]:

$$\forall\, i,\ HasWalkability(Tile[i]) = w \in TileWalkability \tag{12}$$

Tile properties
Each Tile[i] has a $TileType$:

$$\forall\, k,\ Tile[k] \in Map,\ \exists!\ p_k \in TileType : HasType(Tile[k]) = p_k \tag{13}$$

and a $TileWalkability$:

$$\begin{aligned}
&\forall\, k,\ Tile[k] \in Map,\ \exists!\ p_k \in TileWalkability : \\
&\quad HasWalkability(Tile[k]) = p_k
\end{aligned} \tag{14}$$

RobotWorkingZone
A Tile is *robotWorkingZone* if and only if its lower Tile is *workStation*:

$$\begin{aligned} &\forall\ i,\ HasType(Tile[i]) = robotWorkingZone \\ &\iff \\ &\exists\ j:\ Down(Tile[i], Tile[j]) \wedge \\ &\qquad HasType(Tile[j]) = workStation \end{aligned} \tag{15}$$

OperatorWorkingZone
A Tile is *operatorWorkingZone* if and only if its lower-left or lower-right Tile is *workStation*:

$$\begin{aligned} &\forall\ i:\ HasType(Tile[i]) = operatorWorkingZone \\ &\iff \\ &\exists\ j:\ (DownLeft(Tile[i], Tile[j]) \vee \\ &\qquad\qquad DownRight(Tile[i], Tile[j])) \wedge \\ &\qquad HasType(Tile[j]) = workStation \end{aligned} \tag{16}$$

2.1.2 Entities predicates and function

IsIn(Entity)
Defines the *IsIn* function that returns the Tile in which the Entity is located and this Tile is unique:

$$\forall e \in Entity,\ \exists!\ i:\ IsIn(e) = Tile[i] \tag{17}$$

EntitiesAreAdjacent(Entity, Entity)
Given two different entities, *EntitiesAreAdjacent* is True if and only if the two entities are in adjacent Tiles:

$$\begin{aligned} &\forall e, e' \in Entity,\ AreAdjacent(e, e') \\ &\iff \\ &\exists\ i, j:\ i \neq j \wedge \\ &IsIn(e) = Tile[i] \wedge \\ &IsIn(e') = Tile[j] \wedge \\ &Adjacent(Tile[i], Tile[j]) \end{aligned} \tag{18}$$

EntitiesAreClose(Entity, Entity)
Given two different entities, *EntitiesAreClose* is True if and only if the two entities are at distance 2 or 3:

$$\begin{aligned}
&\forall e, e' \in Entity,\ EntitiesAreClose(e, e') \\
&\iff \\
&\exists\, i, j :\ i \neq j \wedge \\
&IsIn(e) = Tile[i] \wedge \\
&IsIn(e') = Tile[j] \wedge \\
&(AreAtDistance(Tile[i], Tile[j], 2) \vee \\
&\qquad AreAtDistance(Tile[i], Tile[j], 3))
\end{aligned} \tag{19}$$

EntitiesAreFar(Entity, Entity)
Given two different entities, *EntitiesAreFar* is True if and only if the two entities are at a distance greater than 3:

$$\begin{aligned}
&\forall e, e' \in Entity,\ EntitiesAreFar(e, e') \\
&\iff \\
&\exists\, i, j :\ i \neq j \wedge \\
&IsIn(e) = Tile[i] \in Map \wedge \\
&IsIn(e') = Tile[j] \in Map \wedge \\
&AreAtDistance(Tile[i], Tile[j], n) \wedge \\
&n > 3
\end{aligned} \tag{20}$$

3 Safety Policy

3.1 Body Movement Safety Policy

The following formulas define the Safety Policy for what concerns the Robot Body movements, valid for all time instant.

If the Operator is close and the Robot is in a position where there is no risk of "trapping" the operator, the Robot is moving slow.

$$\begin{aligned} &OperatorProximity = close \wedge \\ &Risk = low \\ &\implies \\ &BodyState = moving_{slow} \end{aligned} \tag{21}$$

If the Operator is adjacent and the Robot is in a position where there is no risk of "trapping" the operator, the Robot is still.

$$\begin{aligned} &OperatorProximity = adjacent \wedge \\ &Risk = low \\ &\implies \\ &BodyState = still \end{aligned} \tag{22}$$

If the Operator is neither adjacent nor close and the Robot is in a position where there is a risk of "trapping" the operator, the Robot is moving slow.

$$\begin{aligned} &\neg(OperatorProximity = close \vee \\ &OperatorProximity = adjacent) \wedge \\ &Risk = high \\ &\implies \\ &BodyState = moving_{slow} \end{aligned} \tag{23}$$

If the Operator is either adjacent or close and the Robot is in a position where there is a risk of "trapping" the operator, the Robot is still.

$$\begin{aligned} &(OperatorProximity = close \vee \\ &OperatorProximity = adjacent) \wedge \\ &Risk = high \\ &\implies \\ &BodyState = still \end{aligned} \tag{24}$$

If the Robot Body state is $inWorkStation$ or $inGlobalBin$ then it must be in a Tile that is coherent with such state.

$$\begin{array}{l} WorkingState = inWorkStation \\ \Longrightarrow \\ HasType(isIn(robotBody)) = robotWorkingZone \end{array} \tag{25}$$

$$\begin{array}{l} WorkingState = inGlobalBin \\ \Longrightarrow \\ HasType(isIn(robotBody)) = globalBinStation \end{array} \tag{26}$$

If the Robot Body state is $inWorkStation$ or $inGlobalBin$ the Body State is $Still$.

$$\begin{array}{l} WorkingState = inWorkStation \\ \Longrightarrow \\ BodyState = Still \end{array} \tag{27}$$

$$\begin{array}{l} WorkingState = inGlobalBin \\ \Longrightarrow \\ BodyState = Still \end{array} \tag{28}$$

3.2 Arm Movement Safety Policy

The following formulas define the Safety Policy for what concerns the Robot Arm movement and the Operator contact, valid for all time instants.

If the Robot is transferring between Global Bin and Work Station or viceversa, then the Arm is retracted over the Robot Body. This guarantees that if the arm is extended then the Robot must be either in the $inWorkingStation$ or in the $inGlobalBin$ working state.

$$\begin{array}{l} WorkingState = inTransit \\ \Longrightarrow \\ ArmState = Retracted \end{array} \tag{29}$$

If the Operator is in contact with the Robot Arm, no signal is sent from the controller which in turn means that no action is performed.

$$\begin{array}{l} OPContact \\ \Longrightarrow \\ \neg(NewArmSignal) \end{array} \tag{30}$$

If the Robot extends or retracts its Arm fast, then the Operator must not be adjacent (these also guarantees that there is no contact with the Operator).

$$
\begin{array}{l}
(\ ExtendFast\ \vee\ RetractFast\ ) \\
\Longrightarrow \\
\neg(OperatorProximity = adjacent)
\end{array}
\tag{31}
$$

If the Robot extends or retracts its Arm slow, then the Operator can be adjacent but there must not be any contact with him.

$$
\begin{array}{l}
(\ ExtendSlow\ \vee\ RetractSlow\ ) \\
\Longrightarrow \\
\neg(OpContact)
\end{array}
\tag{32}
$$

4 Controller

4.1 Time independent

constants

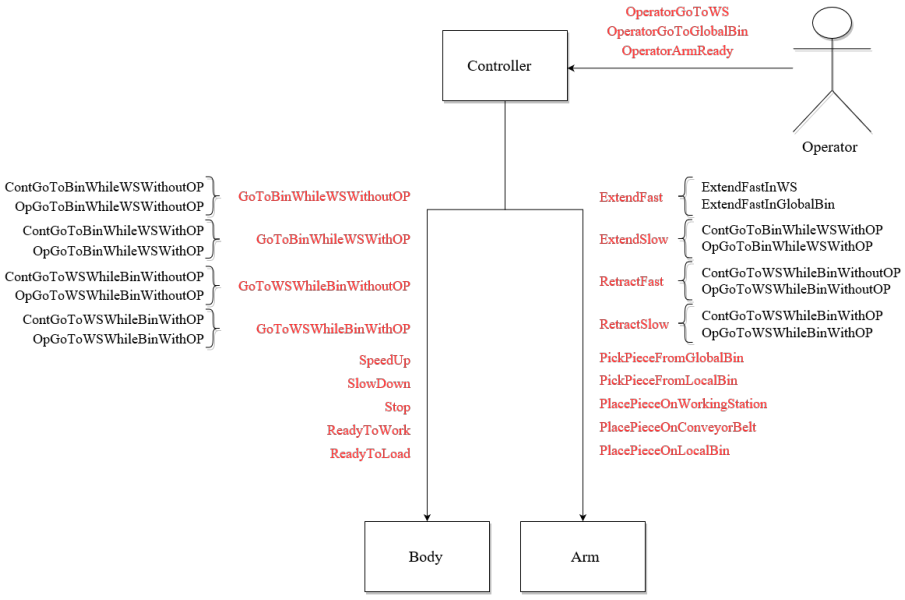
$taskWS_{max}$ = maximum time required to complete an arm task when the robot is in work station (i.e. the time required to extend, pick and retract).

$taskBin_{max}$ = maximum time required to complete the task when the robot is working at the global bin station (i.e. the time required to extend, release and retract).

We assume that such constants are sufficiently large in order to complete the given task. When a task completion depends upon an operator signal we assume that such signal will arrive within a reasonable time frame.

4.2 Controller predicates

What follows is a formalization in terms of logic predicates of the Controller component of the Robot. First we translate all the signals that can be sent by the Operator in terms of Controller signals.
Finally, we define the control model for the Robot Body and for the Robot Arm separately.



4.2.1 Mapping of controller and operator task signals directed to body

The following signals maps the operator and controller signals that are sent to the body in order to accomplish a task.
The following signals are defined both in section (4.4) and (4.3).

GoToBinWhileWSWithoutOP

$$\begin{aligned} &OpGoToBinWhileWSWithoutOP \vee \\ &ContGoToBinWhileWSWithoutOP \\ &\Longleftrightarrow \\ &GoToBinWhileWSWithoutOP \end{aligned} \tag{33}$$

GoToBinWhileWSWithOP

$$\begin{aligned} &OpGoToBinWhileWSWithOP \vee \\ &ContGoToBinWhileWSWithOP \\ &\Longleftrightarrow \\ &GoToBinWhileWSWithOP \end{aligned} \tag{34}$$

GoToWSWhileBinWithoutOP

$$\begin{aligned} &OpGoToWSWhileBinWithoutOP \vee \\ &ContGoToWSWhileBinWithoutOP \\ &\Longleftrightarrow \\ &GoToWSWhileBinWithoutOP \end{aligned} \tag{35}$$

GoToWSWhileBinWithOP

$$\begin{aligned} &OpGoToWSWhileBinWithOP \vee \\ &ContGoToWSWhileBinWithOP \\ &\Longleftrightarrow \\ &GoToWSWhileBinWithOP \end{aligned} \tag{36}$$

4.2.2 Other operator task signals directed to body and arm

The operator can also send signals to the body while the robot is moving in order to change its direction. This signals are:

- $GoToBinWhileTransit$
- $GoToWSWhileTransit$

Another signal is sent by the operator to the arm when the robot is in the working zone and has to move the piece that the arm is holding, from the working zone to the conveyor belt. This signal is:

- $OpArmReady$

This signals are defined in section (4.4).

4.2.3 Mapping of controller signals directed to body to manage robot speed

The controller can also send the following signals to the body in order to control its speed.
This signals are defined in the section (4.3).

SpeedUp

$$\begin{aligned} & SpeedUpSafeZone \vee \\ & SpeedUpStillState \vee \\ & SpeedUpRiskyZone \\ & \Longleftrightarrow \\ & SpeedUp \end{aligned} \tag{37}$$

Stop

$$\begin{aligned} & StopSafeZone \vee \\ & StopRiskyZone \\ & \Longleftrightarrow \\ & Stop \end{aligned} \tag{38}$$

SlowDown

$$\begin{aligned} & SlowDownRiskyZone \vee \\ & SlowDownSafeZone \\ & \Longleftrightarrow \\ & SlowDown \end{aligned} \tag{39}$$

4.2.4 Other controller task signals directed to the body

The controller also send the following signals to the body in order to change the $WorkingState$ of the Robot Body:

- $ReadyToWork$
- $ReadyToLoad$
- $ReadyToLeaveWorkingStation$
- $ReadyToLeaveGlobalBin$

This signals are defined in section (4.3).

4.2.5 Controller task signals directed to the arm

The controller can send signals to the Arm in order to accomplish some tasks. We need to have the duplicate of each command, one for the $GlobalBin$ and one for the $WorkStation$, whereas the final signal that is seen by the Arm is just one.

This signals are defined in section (4.5).

ExtendFast

$$\begin{aligned} & ExtendFastInGlobalBin \vee \\ & ExtendFastInWorkStation \\ & \Longleftrightarrow \\ & ExtendFast \end{aligned} \tag{40}$$

ExtendSlow

$$\begin{aligned} & ExtendSlowInGlobalBin \vee \\ & ExtendSlowInWorkStation \\ & \Longleftrightarrow \\ & ExtendSlow \end{aligned} \tag{41}$$

RetractFast

$$\begin{aligned} & RetractFastInGlobalBin \vee \\ & RetractFastInWorkStation \\ & \Longleftrightarrow \\ & RetractFast \end{aligned} \tag{42}$$

RetractSlow

$$\begin{aligned} & RetractSlowInGlobalBin \vee \\ & RetractSlowInWorkStation \\ & \Longleftrightarrow \\ & RetractSlow \end{aligned} \tag{43}$$

4.3 Body predicates

4.3.1 Body task signals

ContGoToWSWhileBinWithoutOP

The $ContGoToWSWhileBinWithOP$ signal is sent to the body if and only if up to now the body $WorkingState$ was in bin and the body was not moving. Furthermore, at the current time, the local bin is full, the arm is retracted with the end effector in idle and the operator not adjacent:

$$\begin{aligned} & UpToNow_{ie}(WorkingState = inGlobalBin \wedge \\ & \quad BodyState = still) \wedge \\ & \quad ReadyToLeaveGlobalBin \wedge \\ & \quad \neg(OperatorProximity = adjacent) \\ & \quad \Longleftrightarrow \\ & \quad ContGoToWSWhileBinWithoutOP \end{aligned} \tag{44}$$

ContGoToWSWhileBinWithOP

The $ContGoToWSWhileBinWithOP$ signal is like the previous one but is sent to the body when the operator is adjacent to the robot but there is no contact:

$$\begin{aligned} &UpToNow_{ie}(WorkingState = inGlobalBin \wedge \\ &\quad BodyState = still) \wedge \\ &\quad ReadyToLeaveGlobalBin \wedge \\ &\quad OperatorProximity = adjacent \wedge \\ &\quad \Longleftrightarrow \\ &\quad ContGoToWSWhileBinWithOP \end{aligned} \tag{45}$$

ContGoToBinWhileWSWithoutOP

The $ContGoToBinWhileWSWithoutOP$ signal is sent to the body if and only if up to now the body $WorkingState$ was in work station and the body was not moving. Moreover, at the current time, the local bin is empty, the arm is retracted with the end effector in idle and the operator not adjacent:

$$\begin{aligned} &UpToNow_{ie}(WorkingState = inWorkStation \wedge \\ &\quad BodyState = still) \wedge \\ &\quad ReadyToLeaveWorkStation \wedge \\ &\quad \neg(OperatorProximity = adjacent) \\ &\quad \Longleftrightarrow \\ &\quad ContGoToBinWhileWSWithoutOP \end{aligned} \tag{46}$$

ContGoToBinWhileWSWithOP

The $ContGoToBinWhileWSWithOP$ signal is like the previous one but is sent to the body when the operator is adjacent to the robot, it is in the $operatorWorkingZone$ but there is no contact:

$$\begin{aligned} &UpToNow_{ie}(WorkingState = inWorkStation \wedge \\ &\quad BodyState = still) \wedge \\ &\quad ReadyToLeaveWorkStation \wedge \\ &\quad HasType(IsIn(operator)) = operatorWorkingZone \wedge \\ &\quad \Longleftrightarrow \\ &\quad ContGoToBinWhileWSWithOP \end{aligned} \tag{47}$$

4.3.2 Body speed signals

SlowDownSafeZone

If and only if up to now the Robot was moving fast, it is in a safe zone, and the operator is close, the $SlowDownSafeZone$ flag is true:

$$
\begin{aligned}
&WorkingState = inTransit \wedge \\
&UpToNow_{ie}(BodyState = moving_{fast}) \wedge \\
&Risk = low \wedge \\
&OperatorProximity = close \\
&\iff \\
&SlowDownSafeZone
\end{aligned}
\tag{48}
$$

SlowDownRiskyZone

If and only if up to now the Robot is moving fast, it is in a risky zone and the operator is far, the $SlowDownRiskyZone$ flag is true:

$$
\begin{aligned}
&WorkingState = inTransit \wedge \\
&UpToNow_{ie}(BodyState = moving_{fast}) \wedge \\
&Risk = high \wedge \\
&OperatorProximity = far \\
&\iff \\
&SlowDownRiskyZone
\end{aligned}
\tag{49}
$$

SpeedUpSafeZone

If and only if the Robot is moving slow in a safe zone, no signal is received from operator and the operator is far, then the $SpeedUpSafeZone$ flag is true:

$$
\begin{aligned}
&WorkingState = inTransit \wedge \\
&UpToNow_{ie}(BodyState = moving_{slow}) \wedge \\
&Risk = low \wedge \\
&OperatorProximity = far \\
&\iff \\
&SpeedUpSafeZone
\end{aligned}
\tag{50}
$$

SpeedUpRiskyZone

If and only if the Robot is still in a risky zone, no signal is received from operator and the operator is far, then the $SpeedUpRiskyZone$ flag is true:

$$
\begin{aligned}
&WorkingState = inTransit \wedge \\
&UpToNow_{ie}(BodyState = still) \wedge \\
&Risk = high \wedge \\
&OperatorProximity = far \\
&\iff \\
&SpeedUpRiskyZone
\end{aligned}
\tag{51}
$$

SpeedUpStillState

If and only if the Robot is still in a safe zone and no signal is received from operator and the operator is adjacent, then the *SpeedUpStillState* flag is true:

$$\begin{aligned} & WorkingState = inTransit \wedge \\ & UpToNow_{ie}(BodyState = still) \wedge \\ & Risk = low \wedge \\ & OperatorProximity = close \\ & \iff \\ & SpeedUpStillState \end{aligned} \tag{52}$$

StopSafeZone

If and only if the Robot is moving slow in a safe zone, no signal is received from operator and the operator is adjacent, then the $StopSafeZone$ flag is true:

$$\begin{aligned} & WorkingState = inTransit \wedge \\ & UpToNow_{ie}(BodyState = moving_{slow}) \wedge \\ & Risk = low \wedge \\ & OperatorProximity = adjacent \\ & \iff \\ & StopSafeZone \end{aligned} \tag{53}$$

StopRiskyZone

If and only if the Robot is moving slow in a risky zone, no signal is received from operator and the operator is close or adjacent, the *StopRiskyZone* flag is true:

$$\begin{aligned} & WorkingState = inTransit \wedge \\ & UpToNow_{ie}(BodyState = moving_{slow}) \wedge \\ & Risk = high \wedge \\ & (OperatorProximity = close \vee \\ & \quad OperatorProximity = adjacent) \iff \\ & \quad StopRiskyZone \end{aligned} \tag{54}$$

4.3.3 Other body task signals

ReadyToWork

The $ReadyToWork$ signal is sent to the body if and only if up to now the robot was in transit to the work station, it was moving slow and now it is located in a $WorkStation$ Tile:

$$
\begin{aligned}
&UpToNow_{ie}(WorkingState = inTransit \wedge \\
&\quad BodyState = moving_{slow} \wedge \\
&\qquad Direction = toWorkStation) \\
&\wedge \\
&HasType(IsIn(RobotBody)) = robotWorkingZone \\
&\Longleftrightarrow \\
&ReadyToWork
\end{aligned}
\tag{55}
$$

ReadyToLoad

The *ReadyToLoad* signal is sent to the body if and only if up to now the robot was in transit to the global bin, it was moving slow and now it is located in a *globalBinStation* Tile:

$$
\begin{aligned}
&UpToNow_{ie}(WorkingState = inTransit \wedge \\
&\quad BodyState = moving_{slow} \wedge \\
&\qquad Direction = toBin) \\
&\wedge \\
&hasType(IsIn(RobotBody)) = globalBinStation \\
&\Longleftrightarrow \\
&ReadyToLoad
\end{aligned}
\tag{56}
$$

ReadyToLeaveWorkStation

This flag is active if and only if the robot is in working station, the arm is retracted, the local bin is empty (i.e. all the pieces are unloaded), the end effector is idle and there is no operator contact.

$$
\begin{aligned}
&WorkingState = inWorkStation \wedge \\
&ArmState = retracted \wedge \\
&EndEffectorState = idle \wedge \\
&LocalBinStatus = empty \wedge \\
&\neg(OpContact) \\
&\Longleftrightarrow \\
&ReadyToLeaveWorkStation
\end{aligned}
\tag{57}
$$

ReadyToLeaveGlobalBin

The *ReadyToLeaveGlobalBin* flag is active if and only if the arm is retracted, the end effector is in idle and the local bin of the robot is full:
In any case there should not be any operator contact.

$$
\begin{aligned}
&WorkingState = inGlobalBin \wedge \\
&ArmState = retracted \wedge \\
&EndEffectorState = idle \wedge \\
&LocalBinStatus = full \wedge \\
&\neg(OpContact) \\
&\iff \\
&ReadyToLeaveGlobalBin
\end{aligned}
\tag{58}
$$

4.4 Operator predicates

$OPSignal$ is True if and only if an order has been received from the operator. These signals are sent by the operator in order to make the robot move to the $WorkingStation$ or to the $GlobalBin$ and start working there; the last signal instead, is sent by the operator when the piece is ready to be moved from the working zone to the conveyor belt:

$$
\begin{aligned}
&OPSignal \\
&\iff \\
&OperatorGoToWS \vee \\
&OperatorGoToGlobalBin \vee \\
&OpArmReady
\end{aligned}
\tag{59}
$$

4.4.1 Operator task signals

GoToWSWhileTransit

If and only if an $OperatorGoToWS$ signal is received and up to now the robot was in transit (and still it is), moving towards the global bin, then a $GoToWSWhileTransit$ signal is triggered:

$$
\begin{aligned}
&OperatorGoToWS \wedge \\
&UpToNow_{ii}(BodyState = inTransit) \wedge \\
&UpToNow_{ie}(Direction = toBin) \\
&\iff \\
&GoToWSWhileTransit
\end{aligned}
\tag{60}
$$

OpGoToWSWhileBinWithoutOP

If and only if in $taskBin_{max}$ the operator sent an $OperatorGoToWS$ signal while the operator is not adjacent, up to now the robot is in the global bin, the end effector is idle and the arm is retracted, then a $GoToWSWhile BinWithoutOP$ signal is propagated:

$$
\begin{aligned}
&UpToNow_{ie}(WorkingState = inGlobalBin) \wedge \\
&Within(OperatorGoToWS, \\
&\quad -taskBin_{max}) \wedge \\
&EndEffectorState = idle \wedge \\
&ArmState = retracted \wedge \\
&\neg(OperatorProximity = adjacent) \\
&\iff \\
&OpGoToWSWhileBinWithoutOP
\end{aligned}
\tag{61}
$$

OpGoToWSWhileBinWithOP

Same as above but now the operator is adjacent and there is no operator contact, thus an $OpGoToWSWhileBinWithOP$ signal is issued:

$$
\begin{aligned}
&UpToNow_{ie}(WorkingState = inGlobalBin) \wedge \\
&Within(OperatorGoToWS, \\
&\quad -taskBin_{max}) \wedge \\
&EndEffectorState = idle \wedge \\
&ArmState = retracted \wedge \\
&OperatorProximity = adjacent \wedge \\
&\neg(OpContact) \\
&\iff \\
&OpGoToWSWhileBinWithOP
\end{aligned}
\tag{62}
$$

GoToBinWhileTransit

If and only if an $OperatorGoToGlobalBin$ signal is received and up to now the robot was in transit (and still it is), moving towards the working station, then a $GoToGlobalBin\ WhileTransit$ signal is triggered:

$$
\begin{aligned}
&OperatorGoToGlobalBin \wedge \\
&UpToNow_{ii}(WorkingState = inTransit) \wedge \\
&UpToNow_{ie}(Direction = toWorkStation) \iff \\
&GoToBinWhileTransit
\end{aligned}
\tag{63}
$$

OpGoToBinWhileWSWithoutOP

If and only if in $taskWS_{max}$ the operator sent an $OperatorGoToBin$ signal while the operator is not adjacent, up to now the robot was in the working station, the end effector is idle and the arm is retracted, then a $GoToBin\ WhileWSWithoutOP$ signal is propagated:

$$
\begin{aligned}
&UpToNow_{ie}(WorkingState = inWorkingStation) \wedge \\
&Within(OperatorGoToBin, \\
&\quad -taskWS_{max}) \wedge \\
&EndEffectorState = idle \wedge \\
&ArmState = retracted \wedge \\
&\neg(OperatorProximity = adjacent) \\
&\iff \\
&OpGoToBinWhileWSWithoutOP
\end{aligned}
\tag{64}
$$

OpGoToBinWhileWSWithOP

Same as above but now the operator is adjacent and there is no operator contact, thus an $OpGoToBinWhileWSWithOP$ signal is issued:

$$
\begin{aligned}
&UpToNow_{ie}(WorkingState = inWorkingStation) \wedge \\
&Within(OperatorGoToBin, \\
&\quad -taskWS_{max}) \wedge \\
&EndEffectorState = idle \wedge \\
&ArmState = retracted \wedge \\
&UpToNow_{ie}(OperatorProximity = adjacent) \wedge \\
&OperatorWorkingState = inWorkStation \wedge \\
&\neg(OpContact) \\
&\iff \\
&OpGoToBinWhileWSWithOP
\end{aligned}
\tag{65}
$$

4.5 Arm predicates

4.5.1 Arm tasks in global bin

ExtendFastInBin

The $ExtendFastInBin$ is sent if and only if the robot is in global bin, the arm is retracted, the end effector is in idle, the local bin is not full (otherwise the new piece won't be loaded) and the operator is not adjacent to the robot (i.e. there cannot be a contact with the operator):

$$
\begin{aligned}
&WorkingState = inGlobalBin \wedge \\
&ArmState = retracted \wedge \\
&EndEffectorState = idle \wedge \\
&\neg(LocalBinStatus = full) \wedge \\
&\neg(OperatorProximity = adjacent) \\
&\iff \\
&ExtendFastInGlobalBin
\end{aligned}
\tag{66}
$$

ExtendSlowInBin

Like before, but it can be sent only if the operator is adjacent and there is no contact from the operator.

$$\begin{aligned} & WorkingState = inGlobalBin \wedge \\ & (ArmState = retracted \vee \\ & \quad Armstate = extending) \wedge \\ & EndEffectorState = idle \wedge \\ & \neg(LocalBinStatus = full) \wedge \\ & OperatorProximity = adjacent \wedge \\ & \neg(OpContact) \\ & \iff \\ & ExtendSlowInGlobalBin \end{aligned} \tag{67}$$

PickPieceFromGlobalBin

This signal is sent is possible only when the arm is in the global bin (extended), the end effector state is in idle (otherwise it means that the piece it has already been taken), the local bin has space for the piece (not full) and there is no contact of the operator.

$$\begin{aligned} & WorkingState = inGlobalBin \wedge \\ & ArmState = extended \wedge \\ & EndEffectorState = idle \wedge \\ & \neg(LocalBinStatus = full) \wedge \\ & \neg(OpContact) \\ & \iff \\ & PickPieceFromGlobalBin \end{aligned} \tag{68}$$

PlacePieceOnLocalBin

This signal is sent only when the arm is retracted, the end effector state is holding a piece, the local bin has space for the piece (not full) and there is no contact with the operator.

$$\begin{aligned} & WorkingState = inGlobalBin \wedge \\ & ArmState = retracted \wedge \\ & EndEffectorState = holding \wedge \\ & \neg(LocalBinStatus = full) \wedge \\ & \neg(OpContact) \\ & \iff \\ & PlacePieceOnLocalBin \end{aligned} \tag{69}$$

RetractArmFastGlobalBin

The *RetractArmFastGlobalBin* signal is sent if and only if the robot is in global bin, the arm is extended, the effector is in holding and the operator is not adjacent to the robot (i.e. there cannot be a contact with the operator).

$$\begin{aligned} &WorkingState = inGlobalBin \wedge \\ &ArmState = extended \wedge \\ &EndEffectorState = holding \wedge \\ &\neg(OperatorProximity = adjacent) \\ &\iff \\ &RetractArmFastGlobalBin \end{aligned} \tag{70}$$

RetractArmSlowGlobalBin

Like before, but it can be sent only if the operator is adjacent and there is no contact from the operator.

$$\begin{aligned} &WorkingState = inGlobalBin \wedge \\ &(ArmState = extended \vee \\ &\quad ArmState = retracting) \wedge \\ &EndEffectorState = holding \wedge \\ &OperatorProximity = adjacent \wedge \\ &\neg(OpContact) \\ &\iff \\ &RetractArmSlowGlobalBin \end{aligned} \tag{71}$$

4.5.2 Arm tasks in working station

PickPieceFromLocalBin

This signal is sent when the arm is retracted, the end effector state is idle (otherwise it has already took a piece), the local bin is not empty and there is no contact (clearly for safety reasons).

$$\begin{aligned} &WorkingState = inWorkingStation \wedge \\ &ArmState = retracted \wedge \\ &EndEffectorState = idle \wedge \\ &\neg(BinStatus = empty) \wedge \\ &\neg(OpContact) \\ &\iff \\ &PickPieceFromLocalBin \end{aligned} \tag{72}$$

ExtendFastInWS

The *ExtendFastInWS* signal is sent if and only if the robot is in work station, the arm is retracted, the end effector is holding a piece (since it should have

just picked a piece from the local bin) and the operator is not adjacent to the robot (i.e. there cannot be a contact with the operator).

$$\begin{aligned} & WorkingState = inWorkingStation \wedge \\ & ArmState = retracted \wedge \\ & EndEffectorState = holding \wedge \\ & \neg(OperatorProximity = adjacent) \\ & \iff \\ & ExtendFastInWS \end{aligned} \tag{73}$$

ExtendSlowInWS

Like before, but it can be sent only if the operator is adjacent and there is no contact from the operator.

$$\begin{aligned} & WorkingState = inWorkingStation \wedge \\ & (ArmState = retracted \vee \\ & \quad ArmState = retracting) \wedge \\ & EndEffectorState = holding \wedge \\ & OperatorProximity = adjacent \wedge \\ & \neg(OpContact) \\ & \iff \\ & ExtendSlowInWS \end{aligned} \tag{74}$$

PlacePieceOnWorkingStation

The controller sends this signal to the arm if and only if the arm is extended, the end effector state is holding a piece and there is no contact.

$$\begin{aligned} & WorkingState = inWorkingStation \wedge \\ & ArmState = extended \wedge \\ & EndEffectorState = holding \wedge \\ & \neg(OpContact) \\ & \iff \\ & PlacePieceOnWorkingStation \end{aligned} \tag{75}$$

PlacePieceOnConveyorBelt

The controller sends this signal to the arm if and only if the arm is extended, the end effector state is holding a piece and there is no contact (clearly for safety reasons).

$$
\begin{aligned}
&WorkingState = inWorkingStation \wedge \\
&ArmState = extended \wedge \\
&EndEffectorState = inPlace \wedge \\
&OPArmReady \wedge \\
&\neg(OpContact) \\
&\iff \\
&PlacePieceOnConveyorBelt
\end{aligned}
\tag{76}
$$

RetractArmFastWorkingStation

This signal is sent to the arm if and only if the robot is in working station, the arm is extended, the end effector is in idle and the operator is not adjacent to the robot.

$$
\begin{aligned}
&WorkingState = inWorkingStation \wedge \\
&ArmState = extended \wedge \\
&EndEffectorState = idle \wedge \\
&\neg(OperatorProximity = adjacent) \\
&\iff \\
&RetractArmFastWorkingStation
\end{aligned}
\tag{77}
$$

RetractArmSlowWorkingStation

This signal is sent if and only if the robot is in working station, the arm is extended or retracting, the end effector is in idle and the operator is adjacent to the robot but there is no contact.

$$
\begin{aligned}
&WorkingState = inWorkingStation \wedge \\
&(ArmState = extended \vee \\
&\quad ArmState = retracting) \wedge \\
&EndEffectorState = idle \wedge \\
&\neg(OpContact) \wedge \\
&OperatorProximity = adjacent \\
&\iff \\
&RetractArmSlowWorkingStation
\end{aligned}
\tag{78}
$$

5 Robot Body

5.1 Time independent

constants

δ_{slow} = the distance covered in a time instant when the robot is $moving_{slow}$ (i.e. 1 tile)
δ_{fast} = the distance covered in a time instant when the robot is $moving_{fast}$ (i.e. 2 tiles)

In general, we suppose: $\delta_{slow} \leq \delta_{max}$.

5.2 Time dependent

variables

BodyState = {*still*, $moving_{slow}$, $moving_{fast}$ }
Direction = {*none*, *toBin*, $toWorkStation$ }
WorkingState = {$inGlobalBin$, $inWorkStation$, $inTransit$ }
LocalBinStatus = {*empty*, $full$}
OperatorProximity = {*adjacent*, *close*, far}

5.3 Predicates

What follows is a formalization in terms of logic predicates of the behaviour of the Robot Body component. First we define what are the regions of the Map where the Robot can go and we also define the concept of moving with different speed levels. Finally, we define the behaviour of the Body with respect to all the signals that it can receive from the Controller.

5.3.1 Body position

The position of the body is always in a *walkable* tile:

$$HasWalkability(IsIn(robotBody)) = walkable \tag{79}$$

For all instants, the position of the robot body at $t+1$ is the same as t or in an adjacent Tile or in a Tile at distance 2:

$$\begin{aligned} &\exists i \ : \ IsIn(operator) = Tile[i] \\ &\iff \\ &Futr(IsIn(operator) = Tile[i], 1) \vee \\ &(\exists j : Futr(IsIn(operator) = Tile[j], 1) \wedge \\ &\quad (Adjacent(Tile[i], Tile[j]) \vee \\ &\quad AreAtDistance(Tile[i], Tile[j], 2)) \end{aligned} \tag{80}$$

5.3.2 Body working states

The following predicates, define the possible working state of the robot body ($inBin$, $inWorkStation$ or $inTransit$).

inBin

The robot is considered to be $inBin$ only when its position tile is of type $globalBin$, the body is still and without a direction:

$$\begin{aligned} & WorkingState = inBin \\ & \iff \\ & BodyState = still \wedge \\ & Direction = none \\ & HasType(IsIn(robotBody)) = globalBin \end{aligned} \tag{81}$$

inWorkStation

The robot is considered to be $inWorkStation$ only when its position tile is of type $robotWorkingZone$, the body is still and without a direction.

$$\begin{aligned} & WorkingState = inWorkStation \\ & \iff \\ & BodyState = still \wedge \\ & Direction = none \\ & hasType(isIn(RobotBody)) = robotWorkingZone \end{aligned} \tag{82}$$

inTransit

The robot is considered to be $inTransit$ only when it has a valid direction:

$$\begin{aligned} & WorkingState = inTransit \\ & \iff \\ & Direction = toBin \vee \\ & Direction = toWorkStation \end{aligned} \tag{83}$$

5.3.3 Local Bin Behaviour

The local bin of the robot can become *empty* if a piece has been taken from the local bin in the previous time instant.
Similarly, it can become $full$ only if a piece has been previously taken from the global bin:

empty

$$\begin{aligned} & Becomes(LocalBinStatus = empty) \\ & \implies \\ & Past(PickPieceFromLocalBin, 1) \end{aligned} \tag{84}$$

full

$$\begin{aligned} & Becomes(LocalBinStatus = full) \\ & \implies \\ & Past(PlacePieceOnLocalBin, 1) \end{aligned} \tag{85}$$

5.3.4 Operator proximity

The operator proximity is defined with respect to the concept of *adjacent*, *close* and far that have been modeled in the Environment section.

adjacent

$$\begin{aligned} & OperatorProximity = adjacent \iff \\ & EntitiesAreAdjacent(robotBody, operator) \end{aligned} \tag{86}$$

close

$$\begin{aligned} & OperatorProximity = close \iff \\ & EntitiesAreClose(robotBody, operator) \end{aligned} \tag{87}$$

far

$$\begin{aligned} & OperatorProximity = far \iff \\ & EntitiesAreFar(robotBody, operator) \end{aligned} \tag{88}$$

5.3.5 Risk

The risk variable is defined with regard to the position of the robot: if the robot is in a Tile near to the working zone or to the global bin (*operatorWorkingZone*, *robotWorkingZone*, *globalBinStation*), or near the edges of the map (*risky* Tiles) the value of *Risk* is high, otherwise it's low:

high

$$\begin{aligned} & Risk = high \\ & \iff \\ & hasType(isIn(robotBody)) = risky \vee \\ & hasType(isIn(robotBody)) = globalBinStation \vee \\ & hasType(isIn(robotBody)) = operatorWorkingZone \vee \\ & hasType(isIn(robotBody)) = robotWorkingZone \end{aligned} \tag{89}$$

low

$$
\begin{aligned}
&Risk = low \\
&\iff \\
&hasType(isIn(robotBody)) = normal
\end{aligned}
\tag{90}
$$

5.3.6 Body state

Moving slow

The Robot is $moving_{slow}$ if and only if in the next time interval the position of the Robot is in a Tile at distance δ_{slow} with respect to the Tile in which the Robot currently is:

$$
\begin{aligned}
&BodyState = moving_{slow} \\
&\iff \\
&\exists\ i, j:\ Futr(IsIn(Robot) = Tile[i],\ 1) \wedge \\
&IsIn(Robot) = Tile[j] \wedge \\
&areAtDistance(Tile[i], Tile[j], \delta_{slow})
\end{aligned}
\tag{91}
$$

Moving fast

The Robot is $moving_{fast}$ if and only if in the next time tick the position of the Robot is in a Tile at distance δ_{fast} with respect to the Tile in which the Robot currently is:

$$
\begin{aligned}
&BodyState = moving_{fast} \\
&\iff \\
&\exists\ i, j:\ Futr(IsIn(Robot) = Tile[i],\ 1) \wedge \\
&IsIn(Robot) = Tile[j] \wedge \\
&areAtDistance(Tile[i], Tile[j], \delta_{fast})
\end{aligned}
\tag{92}
$$

Still

The Robot is considered to be *still* if and only if its position in the next time tick remain unchanged with respect to the current time tick:

$$
\begin{aligned}
&BodyState = still \\
&\iff \\
&\exists\ i, j:\ Futr(IsIn(Robot, Tile[i]), 1) \wedge \\
&IsIn(Robot) = Tile[j] \wedge \\
&i = j
\end{aligned}
\tag{93}
$$

Speed changes binding

The Robot speed can change if and only if a $NewBodySignal$ has been received:

$$\begin{aligned}
&\exists\ s \in BodyState\ :\ Past(BodyState = s, 1) \wedge \\
&BodyState \neq s \\
&\iff \\
&NewBodySignal
\end{aligned} \tag{94}$$

In general the Robot can have only one speed at a time:

$$\begin{aligned}
&\forall s : BodyState = s \\
&\implies \\
&\nexists\ s' : s' \mathrel{!}= s \wedge \\
&BodyState = s'
\end{aligned} \tag{95}$$

Robot in transit

If the robot is not *still*, then it is in transit. This holds since any time we are not *still* (i.e. $moving_{slow}$ or $moving_{fast}$) then it means we are moving towards the work station or the bin since when the robot arrives to such locations won't be moving (it will be *still*). The reader may notice that few can be said about when the robot is *still*: it may be the case that it is working somewhere or it is in transit but for safety reasons it stopped.

$$\begin{aligned}
&\neg(BodyState = still) \\
&\implies \\
&WorkingState = inTransit
\end{aligned} \tag{96}$$

5.3.7 Robot direction

If the robot is $inTransit$ and the direction is $toWorkStation$, then in a certain time in the future (given a sufficiently large time bound and if the Operator do not impose another direction using the remote) the working state of the robot will be $inWorkStation$.

$$\begin{aligned}
&WorkingState = inTransit \wedge \\
&Direction = toWorkStation \\
&\implies \\
&SomeF(WorkingState = inWorkStation) \\
&\vee \\
&\neg(Until \\
&\quad (\neg GoToBinWhileTransit, \\
&\quad WorkingState = inWorkStation))
\end{aligned} \tag{97}$$

Same if the destination is $toGlobalBin$.

$$
\begin{aligned}
&WorkingState = inTransit \land \\
&Direction = toGlobalBin \\
&\implies \\
&SomeF(WorkingState = inBin) \lor \\
&\neg(Until \\
&\quad (\neg GoToWSWhileTransit, \\
&\quad WorkingState = toGlobalBin))
\end{aligned}
\tag{98}
$$

5.3.8 Signals

Body signals
List of all the possible signals that the body can receive from the controller that are not sent by the operator:

$$
\begin{aligned}
&NewBodySignal \\
&\iff \\
&SpeedUp \lor \\
&SlowDown \lor \\
&Stop
\end{aligned}
\tag{99}
$$

Operator signals
List of all the possible signals that the body can receive from the controller that are sent by the operator:

$$
\begin{aligned}
&NewOPSignal \\
&\iff \\
&GoToWSWhileTransit \lor \\
&GoToBinWhileTransit
\end{aligned}
\tag{100}
$$

Shared operator and body signals
List of all the possible signals that the body can receive from the controller and the operator:

$$
\begin{aligned}
&NewSharedSignal \\
&\iff \\
&GoToWSWhileBinWithoutOP \lor \\
&GoToWSWhileBinWithOP \lor \\
&GoToBinWhileWSWithoutOP \lor \\
&GoToBinWhileWSWithOP
\end{aligned}
\tag{101}
$$

5.3.9 Body movement with controller signals

The $BodyState$ of the Robot changes if and only if a $NewBodySignal$, $NewOPSignal$ or $NewSharedSignal$ is received by the Controler:

$$\begin{aligned} & NewBodySignal \\ & \iff \\ & \exists\ s\ \in\ BodyState\ :\ Past(BodyState = s, 2)\ \wedge \\ & BodyState\ \neq\ s \end{aligned} \tag{102}$$

The same is true for the $Direction$:

$$\begin{aligned} & NewOPSignal\ \vee \\ & NewSharedSignal\ \vee \\ & ReadyToWork\ \vee \\ & ReadyToLoad \\ & \iff \\ & \exists\ d\ \in\ Direction\ :\ Past(Direction = d, 1)\ \wedge \\ & Direction\ \neq\ d \end{aligned} \tag{103}$$

Go to work station while in transit

When a $GoToWSWhileTransit$ signal is received the Robot changes its direction and starts moving towards the work station unless a new signal is received or it finally reaches its destination:

$$\begin{aligned} & GoToWSWhileTransit \\ & \implies \\ & Becomes(Direction = toWorkStation)\ \wedge \\ & Until(\\ & \quad Direction = toWorkStation, \\ & \quad (NewOpSignal\ \vee \\ & \qquad WorkingState = inWorkStation)) \end{aligned} \tag{104}$$

Go to work station while in bin without adjacent operator

When a $GoToWSWhileBinWithoutOP$ signal is received the Robot changes its state into $inTransit$, its direction into $toWorkingStation$ and starts moving since the Operator position has already been checked by the internal Controller and the situation is supposed to be safe, until a $newOpSignal$ is received or it reaches the destination:

$$
\begin{aligned}
&GoToWSWhileBinWithoutOP \\
&\Longrightarrow \\
&Until(\\
&\quad (Direction = toWorkStation \wedge \\
&\qquad WorkingState = inTransit), \\
&(NewOpSignal \vee \\
&\quad WorkingState = inWorkStation))
\end{aligned}
\tag{105}
$$

Go to work station while in bin with adjacent operator

If the Robot body receives a $GoToWSWhileBinWithOP$ it only changes the direction and the working state without moving, until a $NewOpSignal$ is received or it reaches the destination:

$$
\begin{aligned}
&GoToWSWhileBinWithOP \\
&\Longrightarrow \\
&Until(\\
&\quad (Direction = toWorkStation \wedge \\
&\qquad WorkingState = inTransit), \\
&(NewOpSignal \vee \\
&\quad WorkingState = inWorkStation))
\end{aligned}
\tag{106}
$$

Go to bin while in transit

When a $GoToglobalBinWhileTransit$ signal is received the Robot changes its direction and starts moving towards the global bin unless a new signal is received or it finally reaches its destination:

$$
\begin{aligned}
&GoToGlobalBinWhileTransit \\
&\Longrightarrow \\
&Becomes(Direction = toBin) \wedge \\
&Until(\\
&\quad Direction = toBin, \\
&\quad (NewOpSignal \vee \\
&\qquad WorkingState = inGlobalBin))
\end{aligned}
\tag{107}
$$

Go to bin while in work station without adjacent operator

When a $GoToBinWhileWSWithoutOP$ signal is received the Robot changes its direction and starts moving towards the global bin unless a new signal is received or it finally reaches its destination:

$$
\begin{aligned}
&GoToBinWhileWSWithoutOP \\
&\implies \\
&Until(\\
&\quad (Direction = toBin \wedge \\
&\qquad WorkingState = inTransit), \\
&(NewOpSignal \vee \\
&\quad WorkingState = inGlobalBin))
\end{aligned}
\tag{108}
$$

Go to bin while in work station with adjacent operator

When a $GoToBinWhileWSWithOP$ signal is received the Robot changes its direction and starts moving towards the global bin, unless a new signal is received or it finally reaches its destination.
Furthermore, the next time instant it will be at distance 2 to the operator in a *low* risk Tile:

$$
\begin{aligned}
&GoToBinWhileWSWithOP \\
&\implies \\
&Until(\\
&\quad (Direction = toBin \wedge \\
&\qquad WorkingState = inTransit), \\
&(NewOpSignal \vee \\
&\quad WorkingState = inGlobalBin)) \\
&\wedge \\
&NextTime(\\
&\quad AreAtDistance(\\
&\qquad IsIn(robotBody), Tile[j], 2) \\
&\quad \wedge \\
&\quad Risk = low, 1) \\
&\wedge \\
&IsIn(operator) = Tile[j]
\end{aligned}
\tag{109}
$$

Slow down while moving fast

If the Robot body receives a *SlowDown* signal while moving fast, it starts moving slow, until a new Signal is received.

$$
\begin{aligned}
&SlowDown \wedge \\
&UpToNow(BodyState = moving_{fast}) \\
&\implies \\
&Until(BodyState = moving_{slow}, \\
&\quad NewBodySignal)
\end{aligned}
\tag{110}
$$

Stop

If the Robot body receives a Stop signal while moving slow, it stands still until a new Signal is received.

$$\begin{array}{l} Stop \wedge \\ UpToNow(BodyState = moving_{slow}) \\ \Longrightarrow \\ Until(BodyState = still, \\ \quad NewBodySignal) \end{array} \tag{111}$$

Speed up while moving slow

If the Robot body receives a $SpeedUp$ signal while moving slow, it starts moving fast, until a $NewBodySignal$ is received.

$$\begin{array}{l} SpeedUp \ \wedge \ UpToNow(BodyState = moving_{slow}) \\ \Longrightarrow \\ Until(BodyState = moving_{fast}, \\ \quad NewBodySignal) \end{array} \tag{112}$$

Speed up while still

If the Robot body receives a $SpeedUp$ signal while still, it starts moving slow until a $NewBodySignal$ is received.

$$\begin{array}{l} SpeedUp \ \wedge \ UpToNow(BodyState = still) \\ \Longrightarrow \\ Until(BodyState = moving_{slow}, \\ \quad NewBodySignal) \end{array} \tag{113}$$

Start working in Work Station

If the robot is $ReadyToWork$ then the body state must be still, without a direction and in the work station.

$$\begin{array}{l} ReadyToWork \\ \Longrightarrow \\ BodyState = still \wedge \\ Direction = none \wedge \\ WorkingState = inWorkStation \end{array} \tag{114}$$

Start Loading in Global Bin

If the robot is $ReadyToLoad$ then the body state must be still, without a direction and in the work station.

$$
\begin{aligned}
&ReadyToLoad \\
&\Longrightarrow \\
&BodyState = still \wedge \\
&Direction = none \wedge \\
&WorkingState = inGlobalBin
\end{aligned}
\tag{115}
$$

6 Robot Arm

6.1 Time dependent variables

ArmState = { retracted, retracting, extended, extending }
EndEffectorState = { idle, holding, inPlace }

6.2 Predicates

What follows is a formalization in terms of logic predicates of the behaviour of the Robot Arm. First we define what are the the regions where the Arm can be and then we describe the behaviour of the Arm with respect to all the signals it can receive from the Controller.

6.2.1 ArmState

retracted

The Arm is considered to be *retracted* if and only if its position is the same as the Robot Body position:

$$\begin{aligned} & ArmState = retracted \\ & \iff \\ & IsIn(robotArm) = IsIn(robotBody) \end{aligned} \tag{116}$$

retracting/extending

The Arm is considered to be *retracting* or *extending* if and only if its position is *adjacent* to the Robot Body position and if such tile is of type *bin* or *workStation*:

$$\begin{aligned} & ArmState = retracting \vee \\ & ArmState = extending \\ & \iff \\ & (Adjacent(IsIn(robotArm), IsIn(robotBody)) \wedge \\ & (HasType(IsIn(RobotArm)) \ = \ bin \vee \\ & \quad HasType(IsIn(RobotArm)) \ = \ workStation)) \end{aligned} \tag{117}$$

extended

The Arm is considered to be *extended* if and only if its position is at distance 2 from the Robot Body position and if such tile is of type *Bin* or *workStation* or *conveyorBelt*:

$$
\begin{aligned}
&ArmState = extended \\
&\iff \\
&(AreAtDistance(IsIn(robotArm), IsIn(robotBody), 2) \wedge \\
&(HasType(IsIn(robotArm)) \ = \ bin \vee \\
&\quad HasType(IsIn(robotArm)) \ = \ workStation \vee \\
&\quad\quad HasType(IsIn(robotArm)) \ = \ conveyorBelt))
\end{aligned}
\tag{118}
$$

Arm retracted while moving

The Arm is always *retracted* if the Robot $WorkingState$ is $inTransit$. This concept is expressly defined here even though the behaviour of the Controller already guarantees such property.

$$
\begin{aligned}
&WorkingState = inTransit \\
&\implies \\
&ArmState = retracted
\end{aligned}
\tag{119}
$$

6.2.2 OPContact signal

The contact of the operator can happen if and only if the operator is adjacent:

$$
OpContact \implies OperatorProximity = adjacent \tag{120}
$$

6.2.3 Arm generic movements

ExtendFast

If the Robot receives an $ExtendFast$ signal, at the next time instant the Arm is *extended*.

$$
\begin{aligned}
&ExtendFast \\
&\implies \\
&Futr(Becomes(ArmState = extended), 1)
\end{aligned}
\tag{121}
$$

ExtendSlow

If the Robot receives an $ExtendSlow$ signal with the arm *retracted*, at the next time instant the Arm is *extending*.

$$
\begin{aligned}
&ExtendSlow \wedge \\
&UpToNow_{ei}(ArmState = retracted) \\
&\implies \\
&Futr(Becomes(ArmState = extending), 1)
\end{aligned}
\tag{122}
$$

ExtendSlow II

If the Robot receives an $ExtendSlow$ signal with the arm *extending*, at the next time instant the Arm is *extended*.

$$
\begin{array}{l}
ExtendSlow \wedge \\
UpToNow_{ei}(ArmState = extending) \\
\Longrightarrow \\
Futr(Becomes(ArmState = extended), 1)
\end{array}
\tag{123}
$$

RetractFast

If the Robot receives an *RetractFast* signal, at the next time instant the Arm is *retracted*.

$$
\begin{array}{l}
RetractFast \\
\Longrightarrow \\
(Becomes(ArmState = retracted), 1)
\end{array}
\tag{124}
$$

RetractSlow

If the Robot receives a *RetractSlow* signal with the Arm *extended*, at the next time instant the Arm is *retracting*.

$$
\begin{array}{l}
RetractSlow \wedge \\
UpToNow_{ei}(ArmState = extended) \\
\Longrightarrow \\
Futr(Becomes(ArmState = retracting), 1)
\end{array}
\tag{125}
$$

RetractSlow II

If the Robot receives a *RetractSlow* signal with the Arm *retracting*, at the next time instant the Arm is *retracted*.

$$
\begin{array}{l}
RetractSlow \wedge \\
UpToNow_{ei}(ArmState = retracting) \\
\Longrightarrow \\
Futr(Becomes(ArmState = retracted), 1)
\end{array}
\tag{126}
$$

PickPieceFromGlobalBin

If the Robot receives a $PickPieceFromGlobalBin$ signal at the next time instant the end effector will be *holding* a piece.

$$
\begin{array}{l}
PickPieceFromGlobalBin \\
\Longrightarrow \\
Futr(Becomes(EndEffectorState = holding), 1)
\end{array}
\tag{127}
$$

PlacePieceOnLocalBin

If the Robot receives a $PlacePieceOnLocalBin$ signal at the next time instant the end effector will be *idle*, having released the piece it was holding.

$$
\begin{aligned}
&PlacePieceOnLocalBin \\
&\Longrightarrow \\
&Futr(Becomes(EndEffectorState = idle), 1)
\end{aligned}
\tag{128}
$$

PickPieceFromLocalBin

If the Robot receives a $PickPieceFromLocalBin$ signal at the next time instant the end effector will be *holding* a piece.

$$
\begin{aligned}
&PickPieceFromLocalBin \\
&\Longrightarrow \\
&Futr(Becomes(EndEffectorState = holding), 1)
\end{aligned}
\tag{129}
$$

PlacePieceOnWorkingStation

If the Robot receives a $PlacePieceOnWorkingStation$ signal at the next time instant the end effector will be $inPlace$, which means the Operator can start working on the piece.

$$
\begin{aligned}
&PlacePieceOnWorkingStation \\
&\Longrightarrow \\
&Futr(Becomes(EndEffectorState = inPlace, 1)
\end{aligned}
\tag{130}
$$

PlacePieceOnConveyorBelt

If the Robot receives a $PlacePieceOnConveyorBelt$ signal at the next time instant the Hand Effector will be *idle* as the piece will be placed on the conveyor belt.

$$
\begin{aligned}
&PlacePieceOnConveyorBelt \\
&\Longrightarrow \\
&\quad Futr(Becomes(EndEffectorState = idle \wedge \\
&\qquad HasType(IsIn(robotArm)) \ = \ conveyorBelt), 1) \wedge
\end{aligned}
\tag{131}
$$

6.2.4 Arm State behaviour

The state of the arm can change if and only if the Robot has received a $NewArmSignal$ from the controller in the last time instant:

$$
\begin{aligned}
&(\ \exists\ s\ \in\ ArmState\ :\ Past(ArmState = s,\ 1)\ \wedge \\
&\quad ArmState\ \neq\ s)\ \Longleftrightarrow \\
&\qquad (\ Past(\ ExtendFast\ \vee \\
&\qquad\quad ExtendSlow\ \vee \\
&\qquad\qquad RetractFast\ \vee \\
&\qquad\qquad\quad RetractSlow), 1)
\end{aligned}
\tag{132}
$$

6.3 End Effector State behaviour

The state of the end effector can change if and only if the Robot has received a new signal from the Controller in the last time instant:

$$
\begin{aligned}
&(\exists\ s\ \in\ EndEffectorState\ :\ Past(EndEffectorState = s,\ 1) \wedge \\
&\quad EndEffectorState\ \neq\ s)\ \iff \\
&\qquad (Past(\ PickPieceFromGlobalBin \vee \\
&\qquad\quad PickPieceFromLocalBin \vee \\
&\qquad\qquad PlacePieceOnWorkingStation \vee \\
&\qquad\qquad\quad PlacePieceOnConveyorBelt\ ), 1)
\end{aligned}
\tag{133}
$$

7 Operator

The operator movements are not modelled. The environment only keeps track of its position.
As stated in the introduction, we assume that the contact of the operator is retrieved by sensors positioned in the robot arm. Specifically, the operator contact signal is active when the robot arm is near to the operator (e.g. less than 5 cm).
Furthermore, the operator contact has no effect if it occurs in non-working zones, since the arm is locked in position (retracted).
Finally, at the end of operator contact, the state of the robot body and arm are in a valid state modeled by the controller.

7.1 Signals to controller

Operator signals to Controller:

$$\begin{aligned} & OPSignal \\ & \iff \\ & OperatorGoToWS \vee \\ & OperatorGoToGlobalBin \vee \\ & OpArmReady \end{aligned} \tag{134}$$

7.2 Predicates

Operator position
For all instants, the position of the operator at $t+1$ is the same as t or in an adjacent Tile:

$$\begin{aligned} & \exists i \ : \ IsIn(operator) = Tile[i] \\ & \iff \\ & Futr(isIn(Operator) = Tile[i], 1) \vee \\ & (\exists j : Futr(IsIn(operator) = Tile[j], 1) \wedge \\ & \quad adjacent(Tile[i], Tile[j])) \end{aligned} \tag{135}$$

The position of the operator is always in a walkable tile:

$$HasWalkability(IsIn(operator)) = walkable \tag{136}$$