POLITECNICO
MILANO 1863

FMCS – 2019

Project

Francesco Franzini, matr.912857

Document version: 1.0
June 11, 2019

FMCS Project

Francesco Franzini

June 11, 2019

Contents

1 Introduction **2**
1.1 Scope . . . 2
1.2 Document Structure . . . 2

2 Model description **3**
2.1 Introduction . . . 3
2.2 Assumptions and Constraints . . . 3
2.3 Robot Model . . . 4
2.4 Operator Model . . . 5
2.5 Simulation Model . . . 5

3 TRIO Model **6**
3.1 Definitions . . . 6
3.2 Model . . . 6
3.3 Verification . . . 10

1 Introduction

1.1 Scope

This document describes the model that represents the situation given in the FM project specification document. The goal is to verify that in a workplace, where there are an operator and a robot moving simultaneously and manipulating objects, no collision can happen. This model focuses on the movement part of the system, abstracting away the object manipulation as requested in the specification document.

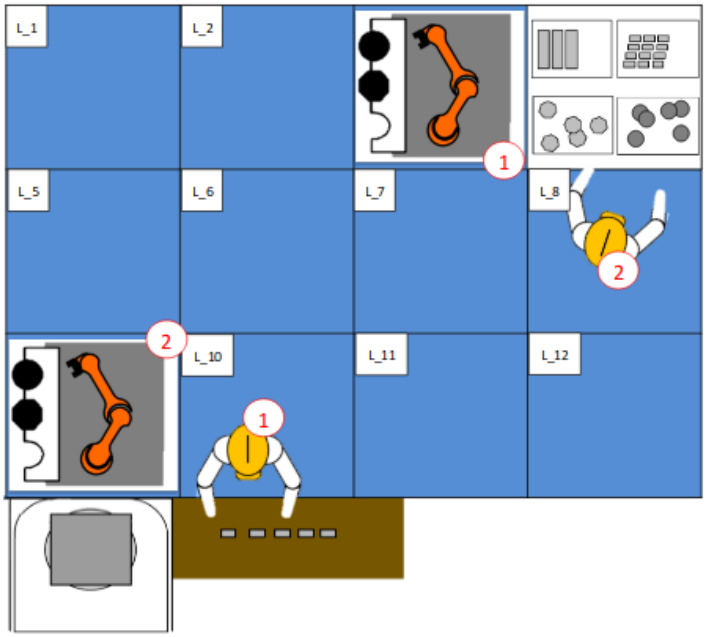

Figure 1: The grid

1.2 Document Structure

This document is composed of three sections.

- **Section1** Contains a general overview of the project
- **Section2** Contains description of the model in natural language
- **Section3** Contains the TRIO model

2 Model description

2.1 Introduction

The model has been split in three main subsections: robot, operator and simulation. The first two model the actors of the system, while the last contains the axioms needed to set the initial conditions, get a meaningful simulation and the Safety property to be verified

2.2 Assumptions and Constraints

Assumptions The following assumptions have been taken into account while designing the model:

- **D1**: Position of the operator at next time instant is known
- **D2**: There is no enforced time limit on the time that operator and robot stay still at their workstations
- **D3**: There is no enforced time limit on the time that operator and robot take to move between their workstations

The domain assumption D1 means that the sensors present on the grid are not only able to detect if the operator is moving, but also *where* it is going. This enables the robot controller to move quite efficiently in cells that are around the operator. This assumption could be removed but, in order to keep the safety property true, every instance in the controller of *next(operatorIn(cell))*) would have to be replaced with a quantification over all the cells that the operator could potentially reach at the next time instant.

2.3 Robot Model

The robot behavior has been modeled as a 3-state FSM that governs the movement around the grid and is managed by a simple controller that is in charge of avoiding collisions. At each time instant the controller+FSM will consider the current state of the grid and force the updates of the robot state in the model.

Here follows a description of the transitions taken for each state by the FSM:

- **RobotToL9**: In this state the robot is going to L9
  - **Next position is L9**: Transition to *RobotWorking*
  - **Next position is not L9**: Self loop
- **RobotToL3**: In this state the robot is going to L3
  - **Next position is L3**: Transition to *RobotWorking*
  - **Next position is not L3**: Self loop
- **RobotWorking**: In this state the robot is working at a station
  - **Working in L9**:
    - **Next state is RobotToL3**: Move away
    - **Next state is not RobotToL3**: Stay still
  - **Working in L3**:
    - **Next state is RobotToL9**: Move away
    - **Next state is not RobotToL9**: Stay still

It is a very simple representation and it is used mainly to keep track of what the robot is doing in the model while not enforcing particular choices on movements.

The controller is made of four main parts:

- **Safety lock**: If the operator is in the same area as the robot, the controller prevents the robot from moving
- **Collision avoidance**: The controller prevents the robot from entering an area where the operator is located
- **Detour avoidance(x2, one for L3 and one for L9)**: If the robot is one cell away from its destination the controller halts it until that cell is free, to avoid detours

As required by the specification document, the controller does not enforce a particular path to destination.

In addition to these, other axioms have been added in order to keep the grid model consistent with the specification:

- **Forbidden place**: The robot cannot be in L4
- **Realistic movement**: The robot can only move by one cell at a time
- **Unique position**: The robot is in one and only one cell at each time instant

2.4 Operator Model

The operator has been modeled in the same way as the robot as far as the FSM is concerned. Since the operator is not an agent that is controllable by our system, it has no controller and can go wherever he or she wants: it will be the robot's duty to prevent accidents.

The FSM description is omitted as it is identical to the one given for the robot, just substituting L9 with L10 and L3 with L8 gives the operator's automaton. As with the robot, axioms for model integrity have been added:

- **Forbidden place**: The operator cannot be in L4
- **Realistic movement**: The operator can only move by one cell at a time
- **Unique position**: The operator is in one and only one cell at each time instant

2.5 Simulation Model

The simulation section of the model consists in four parts that are used to define the predicates used by operator and robot models and the safety property to be verified.

A complete specification is present in Section 3, but the main ones are:

- **Init**: Initial conditions for the system
  - Robot is in L3 in state RobotWorking
  - Operator is in L10 in state OperatorWorking
- **Collision Predicate**: A predicate that is true iff the robot and operator are in the same cell
- **Nice Simulation(Optional)**: Forces the simulation to do at least a loop for both the robot and operator. This is not guaranteed by just the model since the paths taken are left completely free from constraints and could never reach the destination cells within the set bound.
- **Collision Enforcing**: Enforces that sometime in the future there will be a collision with the robot moving into or out from the operator cell. This makes our model unsatisfiable, meaning that no such collision can happen. Since the specification states that collisions are to be avoided when the robot is moving, this model considers a "moving collision" to be the robot moving while the operator is in the same cell or moving to where the operator is (or will be at the next time instant).

3 TRIO Model

3.1 Definitions

In this specification some predicates taken from the *Zot* syntax have been used. They can be easily reduced to basic TRIO predicates as follows:

- **Next(F)** = Dist(F,1)
- **Yesterday(F)** = Dist(F,-1)

3.2 Model

This section contains the TRIO specification of the model, along with the negated safety property.

Auxiliary predicates and functions :

Adjacent checks adjacency of cells while avoiding border issues. A detailed explanation of each condition is given in the Zot file comments.

$$\begin{aligned} Alw(\forall n1, n2(adjacent(n1n2) \iff (\\ (n1 = n2) \vee \\ ((n1 \neq 4) \wedge (n1 \neq 8) \wedge (n1 \neq 12) \wedge ((n2 = n1 + 1) \vee (n2 = n1 - 3))) \vee \\ ((n1 \neq 1) \wedge (n1 \neq 5) \wedge (n1 \neq 9) \wedge ((n2 = n1 - 1) \vee (n2 = n1 + 3))) \vee \\ (n2 = n1 \pm 4) \vee \\ ((n1 \neq 4) \wedge (n2 = n1 + 5)) \vee \\ ((n1 \neq 9) \wedge (n2 = n1 - 5)) \\ ))) \end{aligned}$$

Collision checks if operator and robot are in the same cell.

$$Alw(collision() \iff (\exists p((robotIn(p)) \wedge (operatorIn(p)))))$$

Robot :

Robot Moving is true iff the robot is in another cell at next time instant

$$\begin{aligned} Alw(robotMoving() \iff (\\ (\exists p1(\exists p2(\\ ((p1 \neq p2) \wedge robotIn(p1) \wedge next(robotIn(p2)))))) \\ )) \end{aligned}$$

Robot Always Somewhere asserts that there is one and only one position for the robot at any time instant

$$
\begin{aligned}
&(alw(\exists a(\\
&robotIn(a) \wedge \\
&\neg(\exists b((b \neq a) \wedge (robotIn(b))))\\
&)))
\end{aligned}
$$

Realistic Robot Moving forces the robot to move at maximum 1 cell per time instant (works if position is unique, as guaranteed by the previous invariant)

$$
\begin{aligned}
&(\forall a(alw(\\
&((yesterday(robotIn(a))) \longrightarrow \\
&(\exists b((robotIn(b)) \wedge (adjacent(a,b))))))))
\end{aligned}
$$

Robot State represents the robot FSM

$$
\begin{aligned}
&Alw(\\
&(\neg(robotToL9() \wedge robotToL3())) \wedge \\
&(\neg(robotWorking() \wedge robotToL9())) \wedge \\
&(\neg(robotWorking() \wedge robotToL3())) \wedge \\
&((robotWorking() \vee robotToL3() \vee robotToL9())) \wedge \\
&\\
&((robotToL9() \wedge next(robotIn(9))) \longrightarrow (next(robotWorking()))) \wedge \\
&((robotToL9() \wedge next(\neg robotIn(9))) \longrightarrow (next(robotToL9()))) \wedge \\
&\\
&((robotToL3() \wedge next(robotIn(3))) \longrightarrow (next(robotWorking()))) \wedge \\
&((robotToL3() \wedge next(\neg robotIn(3))) \longrightarrow (next(robotToL3()))) \wedge \\
&\\
&((robotWorking() \wedge next(robotWorking()) \wedge robotIn(3)) \longrightarrow (next(robotIn(3)))) \wedge \\
&((robotWorking() \wedge next(robotToL9()) \wedge robotIn(3)) \longrightarrow (next(\neg robotIn(3)))) \wedge \\
&((robotWorking() \wedge robotIn(3)) \longrightarrow (next(\neg robotToL3))) \wedge \\
&\\
&((robotWorking() \wedge next(robotWorking()) \wedge robotIn(9)) \longrightarrow (next(robotIn(9)))) \wedge \\
&((robotWorking() \wedge next(robotToL3()) \wedge robotIn(9)) \longrightarrow (next(\neg robotIn(9)))) \wedge \\
&((robotWorking() \wedge robotIn(9)) \longrightarrow (next(\neg robotToL9))) \\
&)
\end{aligned}
$$

Robot Controller models a simple controller that drives the robot while avoiding moving collisions

$$(alw$$
;;If collision is true, the robot cannot go in moving state
$$\neg(collision() \wedge robotMoving()) \wedge$$

;;If robot is moving, it cannot go to the next position of the operator
$$(\forall p((next(operatorIn(p)) \wedge robotMoving()) \longrightarrow (\neg next(robotIn(p))))) \wedge$$

;;If going to L9 and adjacent to it, it doesn't go there only if it is blocked(detour avoidance)
$$(robotToL9() \longrightarrow ($$
$$\exists p((robotIn(p) \wedge adjacent(p, 9)) \longrightarrow (next(operatorIn(9)) \vee \neg robotMoving()))) \wedge$$

;;If going to L3 and adjacent to it, the robot doesn't go there only if the operator is blocking
$$(robotToL3() \longrightarrow ($$
$$\exists p((robotIn(p) \wedge adjacent(p, 3)) \longrightarrow (next(operatorIn(3)) \vee \neg robotMoving()))) \wedge$$
$$)$$

Operator :

Operator Moving is true iff the operator is in another cell at next time instant

$$Alw(operatorMoving() \iff ($$
$$(\exists p1(\exists p2($$
$$((p1 \neq p2) \wedge operatorIn(p1) \wedge next(operatorIn(p2))))))$$
$$))$$

Operator Always Somewhere asserts that there is one and only one position for the operator at any time instant

$$(alw(\exists a($$
$$operatorIn(a) \wedge$$
$$\neg(\exists b((b \neq a) \wedge (operatorIn(b))))$$
$$)))$$

Realistic Operator Moving forces the operator to move at maximum 1 cell per time instant

$$(\forall a(alw($$
$$((yesterday(operatorIn(a))) \longrightarrow$$
$$(\exists b((operatorIn(b)) \wedge (adjacent(a, b))))))))$$

Operator State represents the operator FSM

$$
\begin{aligned}
&Alw(\\
&(\neg(operatorToL10() \wedge operatorToL8()))\wedge\\
&(\neg(operatorWorking() \wedge operatorToL10()))\wedge\\
&(\neg(operatorWorking() \wedge operatorToL8()))\wedge\\
&((operatorWorking() \vee operatorToL8() \vee operatorToL10()))\wedge\\
&\\
&((operatorToL10() \wedge next(operatorIn(10))) \longrightarrow (next(operatorWorking())))\wedge\\
&((operatorToL10() \wedge next(\neg operatorIn(10))) \longrightarrow (next(operatorToL10())))\wedge\\
&\\
&((operatorToL8() \wedge next(operatorIn(8))) \longrightarrow (next(operatorWorking())))\wedge\\
&((operatorToL8() \wedge next(\neg operatorIn(8))) \longrightarrow (next(operatorToL8())))\wedge\\
&\\
&((operatorWorking() \wedge next(operatorWorking()) \wedge operatorIn(8)) \longrightarrow (next(operatorIn(8))))\wedge\\
&((operatorWorking() \wedge next(operatorToL10()) \wedge operatorIn(8)) \longrightarrow (next(\neg operatorIn(8))))\wedge\\
&((operatorWorking() \wedge operatorIn(8)) \longrightarrow (next(\neg operatorToL8())))\wedge\\
&\\
&((operatorWorking() \wedge next(operatorWorking()) \wedge operatorIn(10)) \longrightarrow (next(operatorIn(10))))\wedge\\
&((operatorWorking() \wedge next(operatorToL8()) \wedge operatorIn(10)) \longrightarrow (next(\neg operatorIn(10))))\wedge\\
&((operatorWorking() \wedge operatorIn(10)) \longrightarrow (next(\neg operatorToL10())))\wedge\\
&)
\end{aligned}
$$

Collision :

Collision enforcer forces a collision between *moving* robot and operator to happen. The model is made unsatisfiable when CollisionEnforcer() is asserted to be true in the beginning.

$$
\begin{aligned}
&Alw(CollisionEnforcer() \iff (somF(\\
&(collision() \wedge robotMoving())\vee\\
&(next(collision()) \wedge robotMoving())\\
&)))
\end{aligned}
$$

Simulation :

Init describes the model's initial configuration

$$
\begin{aligned}
&Alw(init() \iff (\\
&(robotIn(3)) \wedge robotWorking())\wedge\\
&(operatorIn(10) \wedge operatorWorking())\\
&)
\end{aligned}
$$

Nice Simulation forces the completion of a loop of both operator and robot

$$\begin{aligned} Alw(NiceSimulation() \iff (\\ (next(\neg(robotIn(3))))\wedge \\ (next(\neg(operatorIn(10))))\wedge \\ (next(somF(robotWorking() \wedge robotIn(3))))\wedge \\ (next(somF(operatorWorking() \wedge operatorIn(10))))\wedge \\ )) \end{aligned}$$

Forbidden places invariant prevents robot and operator from going to L4

$$Alw(\neg(robotIn(4)) \wedge \neg(operatorIn(4)))$$

In the Zot file the existential and universal quantifiers have been bound over a range 1..12. In order to have the same result in the TRIO model, we need to add an axiom that forces all position-dependent predicates to be false when the position is out of the grid. An alternative to this would be to add constraints on the quantified variable anytime a quantifier is used, but this would result in very bulky formulas.

$$Alw(\forall p((p \notin [1..12]) \longrightarrow (\neg operatorIn(p) \wedge \neg robotIn(p))))$$

3.3 Verification

Verification has been done by launching the Zot command (ae2sbvzot:zot 30 (yesterday(&& ....))) where the dots are replaced by all the various axioms.

Executing without forcing a moving collision to happen gives a satisfiable model and an example of execution.

```
2. SMT solving: z3...

Evaluation took:
1.027 seconds of real time
0.002517 seconds of total run time (0.000000 user, 0.002517 system)
0.29% CPU
2,459,594,992 processor cycles
32,752 bytes consed

---SAT---
------ time 0 ------
ROBOTIN_3
ROBOTWORKING
OPERATORIN_10
OPERATORMOVING
ROBOTMOVING
OPERATORWORKING
```

```
------ time 1 ------
ROBOTIN_2
ROBOTTOL9
OPERATORIN_7
OPERATORMOVING
OPERATORTOL8
ROBOTMOVING
------ time 2 ------
(...)
------ end ------

Evaluation took:
1.307 seconds of real time
0.279757 seconds of total run time (0.247775 user, 0.031982 system)
[Run times consist of 0.045 seconds GC time,and 0.235 seconds non-GC time. ]
21.42% CPU
3,131,244,576 processor cycles
374,595,232 bytes consed
```

Executing while forcing a moving collision instead gives an unsatisfiable model, so no valid execution is found.

```
2. SMT solving: z3...

Evaluation took:
1.141 seconds of real time
0.002212 seconds of total run time (0.000000 user, 0.002212 system)
0.18% CPU
2,732,598,183 processor cycles
32,736 bytes consed

---UNSAT---
Evaluation took:
1.239 seconds of real time
0.099738 seconds of total run time (0.096896 user, 0.002842 system)
8.07% CPU
2,966,105,379 processor cycles
720,720 bytes consed
```

Since the model is unsatisfiable when a collision is enforced(and normally satisfiable otherwise), it has been proven that no moving collision can happen and so the safety property is always verified.