POLITECNICO MILANO 1863

Model Checking of Warehouse Robotics

Formal Methods for Concurrent and Real-Time Systems
a.y. 20/21

Prof. Pierluigi San Pietro
Livia Lestingi

1 Introduction

Automated warehouses are growing increasingly popular in industrial and large-scale retail settings. Experts generally agree that automation in warehouse management leads to a productivity surge. Commonly observed improvements occur in terms of order handling speed and storage capabilities.
Such facilities employ large fleets of custom-tailored **mobile robots** (most notably, the Amazon Kiva[1]) to dispatch goods to their destination on the shop floor. These devices can **synchronize** with each other and inspect the environment to avoid collisions while carrying large storage pods.
Human operators are still involved in residual **manual** tasks at the edge of the system, such as collecting or stowing products from and in the pods. Minimizing human presence in the core section of the warehouse reduces the likelihood of error while also increasing safety on the shop floor.
The goal of the project is to exploit formal verification techniques—specifically model-checking—to:

1. **model** the core entities of an automated warehouse (expanded in Section 2);
2. **verify** critical properties concerning traffic management and efficiency (expanded in Section 3).

2 Model

In the following, we describe:

- the entities have to be **mandatorily** featured by the model;
- the **mandatory** features and behavioral aspects;
- the simplifying **assumptions**;
- **optional** extra features.

2.1 Environment

As mentioned in Section 1, robots are deployed on warehouse shop floors, and items are stored in storage pods.
Such floor layouts usually feature a larger area called **highway** where robots can roam more freely. Storage pods are lined up in a separate area, and pod **rows** are separated by smaller lanes that allow for robot movement.
We assume that the layout can be represented as a **grid**. Each cell is roughly as big as a mobile robot. Therefore, only **one robot** at a time can occupy a cell (except for the entry point). Some cells are **free**, some are **occupied** by a pod. Figure 1 represents a sample layout, constituted by a $\mathbf{10} \times \mathbf{10}$ grid, to be considered for the project.

[1]Kiva Robots in action

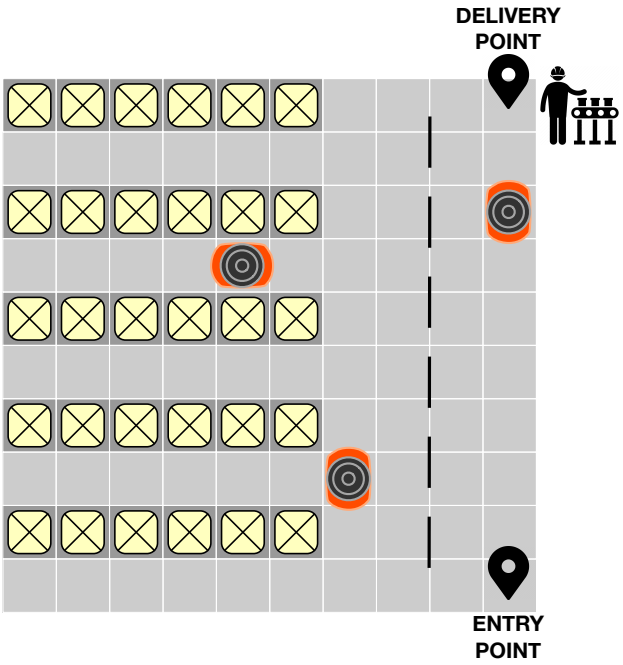

Figure 1: Schematic grid-like representation of a sample floor layout. The robot *highway* is on the right (the dashed line is purely ornamental), whereas the area with storage pods (the yellow squares) is on the left. We assume that all robots enter the system from the *entry point*, and that all pods need to be delivered to the *delivery point*.

💡 *Optional: Design your model so that it potentially supports any rectangular grid-like layout.*

Assume that there is only one **entry point** to the system, which is the starting position for all robots. Furthermore, assume that there is only **one** human **operator** collecting items from the pods. Robots must deliver all pods to the single **delivery point**. Both the delivery and entry point cell positions are known a priori.

2.2 Tasks

In this context, a **task** entails the retrieval and dispatch of an item stored in a specific pod. Assume that a centralized dispatching unit receives new tasks periodically each T time units, where T is a constant system parameter. Tasks are all collected in a single **queue** (visible to all robots), up to a maximum $\mathsf{MAX\_T}$, which is also a system parameter. If new tasks arrive when the queue is already full, they will be lost. A task is removed by the queue when a robot *claims* it (i.e., starts working on it).

💡 *Optional: The time elapsed between two tasks is a sample of a Normal distribution* $\mathcal{N} \sim (\mu_{\mathsf{T}}, \sigma_{\mathsf{T}})$ *(therefore, task arrival times are not predictable).*

The following **phases** make up each task:

1. a robot **claims** the task;
2. the robot **fetches** the requested storage pod;
3. the robot **delivers** the pod to the delivery point;
4. the human **picks up** the item;
5. the robot **returns** the pod to its starting position.

When a new task arrives, its corresponding pod is *randomly* chosen among the ones that are **available**. Therefore, it **never** happens that:

- two *unclaimed* tasks require the same pod;
- a new task requires a pod that is already being carried by another robot.

Robots claim tasks following the FIFO rule (the oldest one is the first one to go as soon as a robot is available).
Assume that there is **no time constraint** on task completion (i.e., they do not need to be completed in a set amount of time).

2.3 Robots

Robots can only perform two *physical* actions:

- lift (or release) and carry a pod;
- move between cells.

Assume that robot movement between cells is **discrete**. A robot can advance by **one** cell every K time units, where K is the same for all areas of the layout and all robots. After time K has elapsed, the robot picks the **next** cell among the adjacent ones (assume that they cannot move diagonally).
When choosing the next cell, the robot **must** follow these rules:

- it cannot occupy a cell already occupied by another robot (except for the entry point);
- it can move under pods only if it is not carrying a pod;
- if no adjacent cell is available, it does not move.

Assume that a robot will claim a task (if at least one is available) as soon as it goes back to idle.

💡 *Optional: As soon as it goes back to idle, the robot claims a new task with a delay described by Exponential distribution* $\mathcal{X} \sim (\lambda)$.

Assume that robots have *endless* battery life and do not need to recharge.

2.4 Human Operator

Assume that there is only one human operator in the system, and that he/she cannot perform any physical action (they do not move).
When a robot is in the delivery cell, it takes H time units for the human to **pick up** the item and **release** the robot. The robot can return the pod only after the human has completed his task and released the robot. Assume that H is a constant parameter.

💡 *Optional: The operator reacts to the arrival of the robot and picks up the object in a time interval which is not fully predictable. Assume that it is described by a Normal distribution* $\mathcal{N} \sim (\mu_{\mathsf{H}}, \sigma_{\mathsf{H}})$.

3 Properties

We want to **formally verify** the **efficiency** of the warehouse as a function of the task queue size ($\mathsf{MAX\_T}$) and system parameters.
The **mandatory** property to be verified is: it never happens that tasks in the queue exceed its maximum size (hence, that incoming tasks get lost).

💡 *If you have included stochastic features, verify the probability that tasks in the queue exceed its maximum size.*

Deliver and critically describe at least 2 system configurations: one that violates the mandatory property and one that does not. Each configuration should feature at least 3 robots and non-trivial values of system parameters (task arrival rate, task queue size, robot speed, human reaction time). Trivial values are considered so when they do not stress the system (hence making the verification irrelevant): for example, if the operator has null reaction time or if new tasks arrive every $60min$ when it takes $1min$ for a robot to complete a task.

💡 *If you have included stochastic features, the two configurations should yield different probability ranges for the mandatory property (in one case the probability of exceeding queue size should be high, in the other it should be low).*

Given the **unboundedness** of exhaustive model-checking, assume there is a maximum amount of tasks that can be received ($> \mathsf{MAX\_T}$).

💡 *Statistical Model-Checking is* ***bounded****. Assume that infinite tasks can be received, but properties should be checked with a time-bound* TAU*. The value of* TAU *is up to you (it can also vary across system configurations), but it should be high enough to illustrate system behavior.*

4 Modeling Tool

The project will be carried out using the Uppaal[2] tool.

[2]Uppaal.org

If you have included stochastic features, make sure you use the latest version of Uppaal with SMC extension.

The system must be modeled as a network of (*optionally probabilistic*) **Timed Automata** through the Uppaal GUI, and properties must be expressed in (*optionally PCTL*) **CTL** logic and verified through the Uppaal engine.

A hands-on introductory demo of the tool will be provided to you in the upcoming days.

5 Delivery Instructions

It is **mandatory** to deliver:

- a .pdf report (max 10 pages excluding front cover and bibliography, no constraint on the template) describing:
  - the model (emphasis on **critical** modeling choices you have made);
  - the system configurations you have chosen;
  - experimental results;
- the .xml Uppaal model. Make sure:
  - it is the same as what you present in the report
  - it is test-ready (i.e., it includes the queries you ran for the experiments and system configurations are easily selectable).

The deadline for the delivery is:

- **June 25** for UIC students. UIC students must enroll in the June exam session; otherwise, the grade cannot be registered in time;
- **July 2** for non-UIC students.