WORKSHEET

# A6.10 Project: the mission robot

Theory of computation · A level · AQA 7517 4.4.2.1 · about 30 min

Name _____ Class _____ Date _____

## What this lesson is about

Validate a mission with a regular expression, carry it out with a Mealy machine, and report with sets.

- The mission language
- The controller
- What the robot saw
- Where the limits are

## Questions

**5 marks in all**

**1.** A mission is one or more ids separated by commas, defined by <mission> ::= <id> | <id>,<mission>. Why is a regular expression enough to check it? [1 mark]

- ☐ **A** The rules only recurse at the end, so nothing needs to be counted or matched: the language is regular
- ☐ **B** Regular expressions can check any BNF grammar
- ☐ **C** It has fewer than ten rules
- ☐ **D** It uses digits

**2.** Which strings are matched in full by [0-9]+(,[0-9]+)*? [1 mark]

Tick every answer that is true.

- ☐ **A** 3,8,5
- ☐ **B** 12
- ☐ **C** 3,,8
- ☐ **D** 3,8,
- ☐ **E** 007

**3.** The robot saw markers {3, 5, 6, 8}. The mission was {3, 5, 8}. What is seen \ mission? Give the member. [1 mark]

**4.** The mission names a marker that is not on the mat, and the controller stays in SEARCH forever. What is the practical answer? [1 mark]

- ☐ **A** Write a checker that detects every possible infinite loop in advance
- ☐ **B** Give the state a time limit and treat running out as a failure
- ☐ **C** Use a faster processor
- ☐ **D** Remove the SEARCH state

**5.** The robot is asked to visit 20 markers in whatever order is shortest. Which approach is most sensible? [1 mark]

- ☐ **A** Try every order, since the problem is tractable
- ☐ **B** Use a heuristic such as nearest neighbour, since the problem is intractable
- ☐ **C** Give up, since the problem is non-computable
- ☐ **D** Use a regular expression

## The task: the mission

Carry out a mission typed in by the user. - Ask for the mission with `input("Mission? ")`. A valid mission matches `[0-9]+(,[0-9]+)*` exactly (check with `re.fullmatch`). While it is not valid, print `invalid mission` and ask again. The input box holds `3,,8` then `3,8,5`, so `invalid mission` is printed exactly once. - For each id in the mission, in order, run the Mealy machine with a table dictionary like the one in lesson A6.2: keys `(state, input)`, values `(next_state, output)`, start state `"SEARCH"`, stopping at `"STOP"`. Each tick: sense, look up, act, change state, `wait(0.1)`. When it reaches STOP, print `reached` and the id, for example `reached 3`. - `sense(target)` and `act(output, steer)` are written for you. `sense` also returns, as a third value, the list of every marker id in view on that tick. - Keep a set of every marker id seen during the whole run. At the end, print `seen but not in mission:` followed by the ids in the set difference, ascending, separated by single spaces, or `none` if it is empty. - Do not touch any marker.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

import re

def bearing_of(cx):
    return (cx - 160) * 120 / 320      # pixels to degrees

def sense(target):
    tags = apriltags()
    ids = [t[0] for t in tags]
    mine = [t for t in tags if t[0] == target]
    if not mine:
        return "none", 0, ids
    if mine[0][3] <= 15:
        return "near", 0, ids
    return "far", bearing_of(mine[0][1]), ids

def act(output, steer):
    if output == "spin":
        turn_right(40)
    elif output == "drive":
        drive(60, 0, steer * 3)
    elif output == "halt":
        stop()

set_cv("apriltag")
```

**Plan your program here, then type it in and press Run.**

**Do it on the robot**

www.bugbotlab.com/learn/a6-10-project-the-mission-robot/

The simulator checks it and tells you when it passes. Nothing to install, no account.

## Challenges

1. Give every state a time limit: if SEARCH lasts more than 10 seconds, print `marker <id> not found` and move on to the next id.
2. Reject missions that name the same marker twice. Can a regular expression check that? What can?
3. Let the user type `nearest` instead of a list, and visit every marker the robot has seen in nearest neighbour order.