EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATH. - NATURWISS. FAKULTÄT
FACHBEREICH INFORMATIK
KOGNITIVE SYSTEME · PROF. A. ZELL

# Artificial Intelligence

## Assignment 1

Assignment due on: 02.11.2024 Discussion on: 08.11.2024

Notes: Please always justify your answers, even for yes/no questions, unless the description explicitly states otherwise. Full grades can only be awarded when answers are justified. Use the template provided on MOODLE for your submissions and answer in English.

### Question 1 Environments for agents (4 points)

For each of the following task environments identify its properties. To do so, complete the table below. For the column 'Number Agents' possible answers are 'single' or 'multi'. Justify your answer.

- Playing Tennis
- Weather Forecast
- Playing Tic-Tac-Toe
- Baking a cake

| Environment | Observable | Deterministic | Static | Discrete | Number Agents |
|---|---|---|---|---|---|
| Tennis | | | | | |
| Weather Forecast | | | | | |
| Tic-Tac-Toe | | | | | |
| Cake | | | | | |

### Question 2 Properties of Agents (4 points)

For each of the following statements, decide whether it is true or false and support your answer with examples or counterexamples where appropriate.

- An agent that senses only partial information about the state cannot be perfectly rational.
- It is possible for a given agent to be perfectly rational in two distinct task environments.
- A perfectly rational poker-playing agent never loses.
- There exists a task environment in which every agent is rational.

### Question 3 Introduction to Python (3+3+4+2=12 Points)

In preparation for this assignment, you need to install an interpreter for python as discussed in the tutorial session. You are not allowed to use any packages. Make sure to use python $\geq 3.7$. Implement the following tasks using the template provided in our Moodle course. In there you will find a class stub for the class `Polynomial` along with some methods, none of which are implemented. The class is supposed to represent a polynomial in the variable $x$. Once you are done, you can use the test cases at the bottom of the file to test your code.

(a) Implement the function `__init__(self, poly_string:str)` which is the constructor of this class. It will parse the input string to initialize the attribute `self.coefficients`, a python dictionary. It should contain the exponents with non-zero corresponding coefficients as keys, and the coefficients themselves as values.
*Hint: Only consider strings where the exponent sign is ^, without multiplication signs, and where there is either a + or a − sign in front of a variable, but not both. The number of spaces in between symbols should not matter.*
*An example is:* `x^3 + 5 x^2 - 2x +5` *for* $x^2 - 2x + 5$.
*Hint: Make yourself familiar with regular expressions and use the package* `re` *in python. Use it together with the regular expression provided in the python template.*

(b) Now implement `__repr__(self)` like in the template. This method should return a string representation of the polynomial. It should start with the highest power of $x$. There can't be a two consecutive character that both are either + or −. There should be no multiplication symbols or spaces.

(c) Implement the static method `long_division(dividend:Polynomial, divisor:Polynomial)` that performs polynomial division with remainder (dividend over divisor). For this, implement the following pseudo code. Feel free to add whatever methods you need to the class `Polynomial`.

**Algorithm 1** `long_division(dividend:Polynomial, divisor:Polynomial)`

```
quotient ← Polynomial()                          ▷ Initialize relevant variables.
remainder ← copy(dividend)
while degree(remainder) ≥ degree(divisor) do
    result ← leading_term(remainder) / leading_term(divisor)
    quotient ← quotient + result
    remainder ← remainder − result
end while
return quotient, remainder
```

(d) We want to use what we achieved so far to write a function that performs Euclid's algorithm to find the greatest common divisor of two polynomials. For this, implement the following pseudo code. Feel free to add whatever methods you need to the class `Polynomial`. (usually, the gcd of two polynomials is normalized, but we ignore that here.)

**Algorithm 2** `polynomial_gcd(p:Polynomial, q:Polynomial)`

```
if p == 0 then                                   ▷ Check for edge cases.
    return q
end if
if q == 0 then                                   ▷ Check for edge cases.
    return p
end if
while q != 0 do
    _, r ← long_division(p, q)
    p, q ← q, r
end while
return p
```