Knowledge Representation & Reasoning

(Group C)

Anna Langedijk
annalangedijk@gmail.com

Important dates

- Homework 3: March 10, 23:59
- Exam: March 28, 14:00

Important dates

- Homework 3: March 10, 23:59
- Exam: March 28, 14:00
  - On pen & paper
  - Can bring cheat sheet, just not digital
    - Tip: bring examples of ASP problems and solutions

Homework 3

- You do not have to understand the provided Python code for solving
- Please write your code in terms of the input so we can test it for other inputs
- The given example should take only seconds to solve
- Please restart the kernel + run all cells before submitting

Planning problem, general tips

1. Write out the facts:
   - What objects do I have? How are they connected?
   - What is the starting state of the world at timestep $T = 1$?

Planning problem, general tips

1. Write out the facts:
   - What objects do I have? How are they connected?
   - What is the starting state of the world at timestep $T = 1$?
2. Define all `possible_actions`, regardless of "legality"

Planning problem, general tips

1. Write out the facts:
   - What objects do I have? How are they connected?
   - What is the starting state of the world at timestep $T = 1$?
2. Define all `possible_actions`, regardless of "legality"
3. Define a choice rule:

```
1 { do(T,A) :- possible_action(A) } 1 :-
timestep(T), T<t.
```

Planning problem, general tips

1. Write out the facts:
   - What objects do I have? How are they connected?
   - What is the starting state of the world at timestep $T = 1$?
2. Define all `possible_actions`, regardless of "legality"
3. Define a choice rule:

   ```
   1 { do(T,A) :- possible_action(A) } 1 :-
   timestep(T), T<t.
   ```
4. Describe state (non)changes in terms of the actions
   - `state(T+1, NewState) :- do(T, Action), state(T, OldState), ...`

Planning problem, general tips

1. Write out the facts:
   - What objects do I have? How are they connected?
   - What is the starting state of the world at timestep $T = 1$?
2. Define all `possible_actions`, regardless of "legality"
3. Define a choice rule:

   ```
   1 { do(T,A) :- possible_action(A) } 1 :-
   timestep(T), T<t.
   ```
4. Describe state (non)changes in terms of the actions
   - `state(T+1, NewState) :- do(T, Action), state(T, OldState), ...`
   - Fluents can make these state (non)changes more modular, but they are not required

Planning problem, general tips

1. Write out the facts:
   - What objects do I have? How are they connected?
   - What is the starting state of the world at timestep $T = 1$?
2. Define all `possible_actions`, regardless of "legality"
3. Define a choice rule:

   ```
   1 { do(T,A) :- possible_action(A) } 1 :-
   timestep(T), T<t.
   ```
4. Describe state (non)changes in terms of the actions
   - `state(T+1, NewState) :- do(T, Action), state(T, OldState), ...`
   - Fluents can make these state (non)changes more modular, but they are not required
5. Describe constraints on the chosen actions

   ```
   :- do(T, Action), not legal(T,Action).
   ```

Planning problem, general tips

1. Write out the facts:
   - What objects do I have? How are they connected?
   - What is the starting state of the world at timestep $T = 1$?
2. Define all `possible_actions`, regardless of "legality"
3. Define a choice rule:

   ```
   1 { do(T,A) :- possible_action(A) } 1 :-
   timestep(T), T<t.
   ```
4. Describe state (non)changes in terms of the actions
   - `state(T+1, NewState) :- do(T, Action), state(T, OldState), ...`
   - Fluents can make these state (non)changes more modular, but they are not required
5. Describe constraints on the chosen actions

   ```
   :- do(T, Action), not legal(T,Action).
   ```
6. Define the final state as some `goal_reached(T) :- ...`

Planning problem, general tips

1. Write out the facts:
   - What objects do I have? How are they connected?
   - What is the starting state of the world at timestep $T = 1$?
2. Define all `possible_actions`, regardless of "legality"
3. Define a choice rule:
   `1 { do(T,A) :- possible_action(A) } 1 :- timestep(T), T<t.`
4. Describe state (non)changes in terms of the actions
   - `state(T+1, NewState) :- do(T, Action), state(T, OldState), ...`
   - Fluents can make these state (non)changes more modular, but they are not required
5. Describe constraints on the chosen actions
   `:- do(T, Action), not legal(T,Action).`
6. Define the final state as some `goal_reached(T) :- ...`
7. Add a minimizing statement such as `#minimize {T :- goal_reached(T)}`

Exercises of this week

- (1), (2): Planning problems
- I can start solving (1) classically, (but there are many correct ways to solve it)
- (3): Theoretical ASP