Middle East Technical University | Department of Computer Engineering

CENG 242

Programming Language Concepts

Spring '2016-2017

Programming Assignment 5

Due date: 02 June 2017, Saturday, 23:55

1 Introduction

In this assignment, you are going to solve a task where you use tiles to fill a rectangle-shaped hole. Basically, you are going to provide a set of tiles that covers the width of a given hole when they are put side by side. An example sketch is given in Figure 1.

2 Specifications

2.1 Tiles

1. The knowledge base containing the tiles will be given in the file `tiles.pl`.
2. Each tile is a fact defined as;

```
tile(TILE_NAME, TILE_WIDTH, TILE_HEIGHT, TILE_COLOR).
```

   where `TILE_NAME` represents the name of the tile, `TILE_WIDTH` and `TILE_HEIGHT` represent the size of the tile and `TILE_COLOR` represents the color of the tile.
3. The fields `TILE_NAME` and `TILE_COLOR` can be any arbitrary atom. However, `TILE_NAME` field is unique for each tile fact. That is, a tile with a name cannot have different sizes or colors in the knowledge base.
4. `TILE_WIDTH` and `TILE_HEIGHT` of a tile are integers and they are guaranteed to be greater than or equal to 1, i.e. `TILE_WIDTH` ≥ 1 and `TILE_HEIGHT` ≥ 1.
5. Tiles cannot be rotated, i.e., they are going to be used as they are defined in the knowledge base.

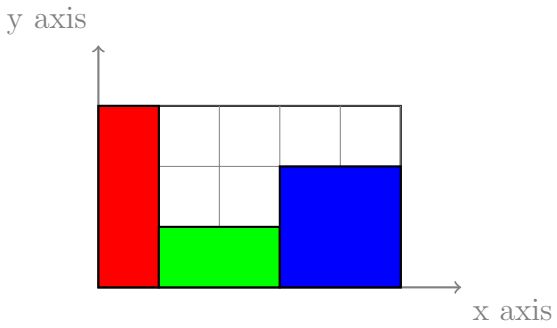

Figure 1: An example 5x3 rectangle-shaped hole filled with 1x3 red, 2x1 green and 2x2 blue tile. The width of the hole is completely filled. However, as the total area of the hole is 15 units and the covered area is $3 + 2 + 4 = 9$, $9/15 = 60\%$ of the area of the hole is filled.

2.2 Task

2.2.1 filled_area (40 points)

1. You are going to implement a predicate called **filled_area/2** taking 2 arguments.

```
filled_area(TILES, AREA).
```

2. `TILES` argument is a list of tile names.
3. `AREA` argument is an integer which represents the sum of the areas of the `TILES`.
4. It is guaranteed that the `TILES` argument will always be provided in any kind of query and the given tile names will be present in the knowledge base.
5. A query missing `AREA` value should evaluate the total area covered by the given set of `TILES`. If the set of `TILES` is empty, it should evaluate the `AREA` as 0.
6. A query providing all of the arguments should evaluate as `true` if the total area covered by the given set of `TILES` is equal to `AREA`. It should evaluate as `false`, otherwise.

2.2.2 filling_list (60 points)

1. You are going to implement a predicate called **filling_list/4** taking 4 arguments.

```
filling_list(WIDTH, HEIGHT, PERCENTAGE, TILES).
```

2. `WIDTH` and `HEIGHT` of the hole are integers and they are guaranteed to be greater than or equal to 1, i.e. `WIDTH` ≥ 1 and `HEIGHT` ≥ 1.
3. `TILES` argument is a list of tile names.
4. `PERCENTAGE` is a float which satisfies the condition $0 \leq$ `PERCENTAGE` ≤ 1.
5. It is guaranteed that the arguments `WIDTH`, `HEIGHT` and `PERCENTAGE` are always provided in any kind of query.
6. A query missing `TILES` value should evaluate a list of tiles from the knowledge base that can fill the rectangle-shaped hole with the given `WIDTH` and `HEIGHT` by a percentage $\geq$ `PERCENTAGE`. Such a query should evaluate all possible tile sets and orderings when the backtracking is forced.

7. A query missing `TILES` value should evaluate as `false` if it is not possible to fill the rectangle-shaped hole with the given constraints.

8. A query providing all of the arguments should evaluate as `true` if the given set of `TILES` fills the rectangle-shaped hole with the given `WIDTH` and `HEIGHT` by a percentage $\geq$ `PERCENTAGE`. It should evaluate as `false`, otherwise.

9. Each tile is unique, that is, a tile name cannot occur more than once in the `TILES` list.

10. It is assumed that the tiles are put side by side (along the x-axis) and it is not possible to put one on top of the other.

11. It is assumed that the tiles are not overlapping.

12. The given set of `TILES` should fit into the rectangle-shaped hole. That is, a tile in the list cannot have a bigger height than the `HEIGHT` of the hole and the total width of the `TILES` set must be directly equal to the `WIDTH` of the hole. The total width of the `TILES` cannot exceed the `WIDTH` of the hole.

13. The set of `TILES` does not have to be the best set of tiles that covers the most of the hole. It is enough that the set covers the hole by a percentage $\geq$ `PERCENTAGE`.

14. You may use the predicate `filled_area/2` to calculate the area covered by a filling list.

2.3 An Example

2.3.1 Example Knowledge Base

```
:- module(tiles, [tile/4]).

tile(t1, 3, 5, blue).
tile(t2, 1, 2, blue).
tile(t3, 4, 3, blue).

tile(t4, 2, 4, red).
tile(t5, 4, 1, red).

tile(t6, 2, 3, green).
tile(t7, 1, 5, green).
tile(t8, 3, 3, green).

tile(t9, 1, 1, purple).
tile(t10, 2, 5, purple).
```

2.3.2 Example Runs for filled_area/2

```
?- filled_area([], A).
A = 0.

?- filled_area([t1], A).
A = 15.

?- filled_area([t1, t2, t3], A).
A = 29.

?- filled_area([t1, t2, t3, t4, t5], A).
A = 41.
```

2.3.3 Example Runs for filling_list/4

```
?- filling_list(1, 1, 1.0, FL).
FL = [t9] ;
false.

?- filling_list(3, 3, 1.0, FL).
FL = [t8] ;
false.

?- filling_list(3, 5, 1.0, FL).
FL = [t1] ;
FL = [t10, t7] ;
FL = [t7, t10] ;
false.

?- filling_list(3, 5, 0.8, FL).
FL = [t1] ;
FL = [t10, t2] ;
FL = [t7, t4] ;
FL = [t4, t7] ;
FL = [t10, t7] ;
FL = [t2, t10] ;
FL = [t7, t10] ;
false.

?- filling_list(2, 1, 0.4, FL).
false.

?- filling_list(1, 2, 0.4, FL).
FL = [t2] ;
FL = [t9] ;
false.
```

3 Regulations

1. **Programming Language:** You should write your code using SWI Prolog.
2. **Late Submission:** See the syllabus for the details.
3. **Cheating:** All the work should be done individually. **We have zero tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations.

4. **Newsgroup:** You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.

5. **Evaluation:** Your program will be evaluated automatically using "black-box" technique so make sure to obey the specifications.

4 Submission

Submission will be done via Ceng Class. Submit a single file called "hw5.pl" through the Ceng Class system. The file must start with the following line.

```
:- module(hw5, [filled_area/2, filling_list/4]).
:- [tiles].
```