D5: Databases and Information Systems
Knowledge Representation
for the Semantic Web, WS 2017
Project 2

*The deadline for submitting your project is on **26.01.2018**. Detailed information on the submission process is given in Section 2. Please do not wait until the deadline, and start preparing your solutions as early as possible! You can get up to **10 points** for this project. For questions on the assignment please come during the office hours or send an email (see the course website for further information).*

1 Introduction

In this assignment you will encode a logic puzzle using answer set programming. To test and run your program you will use `DLV`, which is a solver for extended logic programs. To start with, you should install the most current version of `DLV` (2012-12-17), which can be obtained from its official homepage.[1]

The objective of this project is to construct suitable extended logic programs for the problem of solving a given logic puzzle such that the solutions of this problem are determined by the answer sets of the program. The encoding of the problem should rely on the *guess-and-check* paradigm of answer-set programming, by using *default negation* and *integrity constraints* as essential formalization techniques.

Before you start designing the logic program we recommend to read the following tutorials to make yourself familiar with `DLV` and Answer Set Programming in general.

- `http://www.dlvsystem.com/html/DLV_User_Manual.html`
- `http://www.dbai.tuwien.ac.at/proj/dlv/tutorial/`
- In addition to these online-resources the article "Answer Set Programming: A Primer" by T. Eiter, G. Ianni, T. Krennwallner is available on the course website.

1.1 General Problem Description

Five dukes live in five different castles, each with a different color. Every duke owns a different vehicle and has a different pet. Near every castle there is a park (belonging to a castle), where different types of fruits grow. Given the following background knowledge, the goal is to find out all valid results of the form `result(Castle, Color, Vehicle, Pet, Fruit)`, which reflect the properties of each castle according to the constraints (1)–(14).

1. The fifth castle is purple.
2. Apples grow directly to the right of oranges.
3. The duke driving motocycle lives in the yellow castle.
4. Mandarins grow to the right of bananas.
5. The third castle is not blue.
6. Deers are the pets of the duke from castle five.
7. There are two castles between the park with cherry trees and bananas on the right.

[1] `http://www.dlvsystem.com/dlv/`

8. The scooter is driven by the duke living in castle two.
9. There is one castle between the castle where camels live and the castle whose owner has rides a bike on the right.
10. Oranges grow to the left of the castle owned by a motocycle driver.
11. The duke with monkeys drives a scooter.
12. Pandas live directly to the left of deers.
13. The boat belongs to the duke living in castle five.
14. The apples grow directly next to the pink castle.

Formulate this problem as an Answer Set Program for computing the solution to this logic puzzle.

1.2 Subtask 1: Definition of the Guess-Program

The first task is to construct a program `guess.dl` which calculates all possible answer sets for valid combinations of castles, colors, vehicles, pets and fruits. In general, such a combination is valid if and only if each castle has a different color, and dukes in different castles have different vehicles and different pets. Moreover, different fruits grow in their parks.

Use the following predicates to encode this task:

- `castle(C)`: `C` is an integer value ($1 \leq$ `C` ≤ 5) that represents a specific castle;
- `color(CO)`: `CO` is a color;
- `vehicle(V)`: `V` is a type of vehicle;
- `pet(P)`: `P` is a kind of pet;
- `fruit(F)`: `F` is a kind of fruit;
- `has_color(C, CO)`: castle `C` has color `CO`;
- `has_vehicle(C, V)`: the duke living in castle `C` owns the vehicle `V`;
- `has_pet(C, P)`: the duke living in castle `C` has pets `P`;
- `has_fruit(C,F)`: fruit `F` grows in the park of castle `C`;
- `result(C, CO, V, P, F)`: the result for each castle `C` contains the respective color `CO`, the vehicle `V`, the pets `P`, and the fruits `F` growing in the respective park of the castle.

We provide a definition of all facts (5 castles, 5 colors, 5 vehicles, 5 pets, and 5 fruits) in the file `facts.dl`, which is available on the website next to the link with this assignment.

Create a file `guess.dl` for guessing possible assignments of colors, vehicles, pets and fruits to castles. In the same file define rules that ensure the uniqueness of assignments of colors, vehicles, pets, and fruits to castles, e.g., no castle has two different colors, and no color is assigned to two different castles. You may need some auxiliary predicates (with arbitrary names) to do so. In addition, define a rule for `result(C, CO, V, P, F)` in `guess.dl`, that collects the assigned poperties for each castle.

Run your program `guess.dl` together with `facts.dl` using `DLV` and compute possible answer sets. You can limit the number of generated answer sets to a specific number using the command-line option `-n`, (e.g., `-n=20`). You may want to use the `DLV` command-line option `-filter` or `-pfilter` for selecting only specific predicates (e.g., `-filter=result`) to appear in the answer sets. Generate enough answer sets to make sure that all of them are correct with respect to the uniqueness of assignments of properties to castles within each answer set.

1.3 Subtask 2: Definiton of the Check-Program

The standalone test of `guess.dl` produced a lot of possible answer sets but not all of them meet the requirements of our logic puzzle from Section 1.1.
Formulate integrity constraints in a program `check.dl` that kill all answer sets that are not valid with respect to the requirements given in (1)–(14). You may need some auxiliary predicates (with arbitrary names) to define the positions of the castles. Since the castles are represented by integer values n with $1 \leq n \leq 5$, where `castle(1)` is the leftmost castle and `castle(5)` is the rightmost one, you can use addition and subtraction to define the constraints. The positions have the following meaning:

- castle X is left of castle $Y \Rightarrow X < Y$
- castle X is directly left of castle $Y \Rightarrow X = Y - 1$
- castle X is right of castle $Y \Rightarrow X > Y$
- castle X is directly right of castle $Y \Rightarrow X = Y + 1$
- castle X is directly next to castle $Y \Rightarrow X = Y - 1$ **or** $X = Y + 1$
- there is one castle between castle X and castle $Y \Rightarrow X = Y + 2$ **or** $X = Y - 2$
- analogous for 'two castles between'

Now run all three programs `facts.dl`, `guess.dl`, and `check.dl` using `DLV` and compute answer set(s). Again you can use the command-line option `-filter` to select relevant predicates. Do not include the command `#maxint` in your files (if `DLV` asks you to do so)—rather, execute `DLV` with command-line option `-N` to specifiy the maximum range of integers.

How many answer sets do you get now? Does the puzzle have a unique solution?

Hint: Taking all constraints into account, the puzzle should have a unique solution (otherwise something is wrong).

2 Submission

The strict deadline for the submission of this project is **January, 26, 23:55**. Compress your files `guess.dl`, `check.dl` in one ZIP-file (without subdirectories!) named

`puzzle_XXXXXXX.zip` ,

where XXXXXXX is your matriculation number. Send the zipped file to gadelrab@mpi-inf.mpg.de and dstepano@mpi-inf.mpg.de before the deadline.

It is important to document your answer set program (relate the requirements to the relevant rules in your submitted files `guess.dl`, `check.dl` using comments). The maximum number of points cannot be achieved if the source code is not documented sufficiently.