Heinrich Heine Universität Düsseldorf

Software Engineering and Programming Languages
Prof. Dr. Michael Leuschel
David Geleßus, Fabian Vu

Introduction to Logic Programming – WS 2023
Exercise Sheet 8

1 Exercises

Exercise 1 (Lecture – Inverted Classroom)
Watch the lecture videos *LP 15 Informed Search*[1] in the HHU Mediathek. The corresponding slides are uploaded in ILIAS: 9_Prolog_Search.pdf (slides 57-129)

The complete playlist is available at: `https://mediathek.hhu.de/playlist/691`.

Note: you have to log in with your HHU account (Uni-Kennung) to see the lecture videos!

The exercises will be discussed on 12th December 2023.

Exercise 2 (SLD Resolution)
Consider the following Prolog program:

```
1  student(bob).
2  student(alice).
3
4  passes_exam(Student, Date) :-
5      student(Student),
6      has_learned(Student),
7      exam_takes_place(Date),
8      participates_exam(Student, Date).
9  passes_exam(Student, Date) :-
10     student(Student),
11     exam_takes_place(Date),
12     lecturer_has_good_day(Date),
13     participates_exam(Student , Date).
14
15 exam_takes_place(date(16, 2, 2021)).
16 exam_takes_place(date(30, 3, 2021)).
17
18 lecturer_has_good_day(date(30, 3, 2021)).
19 has_learned(alice).
20 participates_exam(alice, date(16, 2, 2021)).
21 participates_exam(kim, date(16, 2, 2021)).
22 participates_exam(kim, date(30, 3, 2021)).
```

[1] `https://mediathek.hhu.de/watch/fae573bd-5254-4701-9bc7-b2cc0f4b8cc4`

Create a *complete* SLD-tree for the following call:

```
?- passes_exam(S, D).
```

Evaluate all choicepoints and specify all most general unificators.

Note: You can use abbreviations for the predicate names.

Exercise 3 (Iterative Deepening - 15-Puzzle)
Implement a Prolog predicate `solve_idfs(+Puzzle)` which solves the 15 puzzle[2] using iterative deepening.

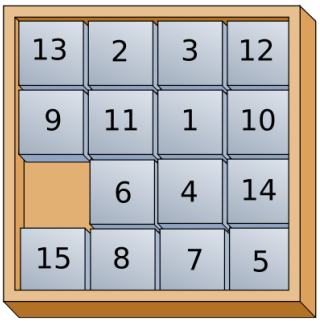

Abbildung 1: Representation as a Prolog predicate `c/4`:
c([13, 2, 3, 12], [9, 11, 1, 10], [x, 6, 4, 14], [15, 8, 7, 5]).

The data structure representing a puzzle is a term `c/4` with four lists. Each list contains a number between 1 and 15 or the atom `x` which represents the empty field in the puzzle. For instance:

```
c([x, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11], [12, 13, 14, 15]).
```

Start by implementing a predicate `s(+Puzzle, -NextPuzzle)` which computes a possible next state `NextPuzzle` for a given puzzle. It should be possible to enumerate all subsequent puzzle states by backtracking.

[2] `http://de.wikipedia.org/wiki/15-Puzzle`