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 3

1 Exercises

The exercises will be discussed on 7th November 2023. There are unit tests for each programming exercise in the ILIAS.

Exercise 1 (Lecture – Inverted Classroom)
Watch the lecture videos *5 LP PropLogic Resolution*[1] and *6 LP DPLL*[2] in the HHU Mediathek. The corresponding slides are uploaded in ILIAS: 4_PropProofTheory.pdf and 5_DPLL_SLD.pdf

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!

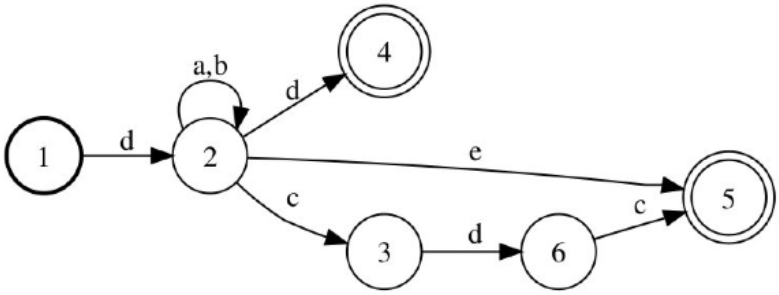
Exercise 2 (Simplification of Propositional Formulae)
Simplify the following two propositional formulae by rewriting implications to disjunctions and moving negations inwards to the literals.

1. $(a \Rightarrow (b \wedge \neg a)) \Rightarrow b$
2. $(\neg(a \vee b)) \wedge (\neg b \Rightarrow a)$

Exercise 3 (Automata)
Represent the automata shown as facts `delta(-In, -Literal, -Out)` in Prolog. For instance, `delta(1, d, 2).` Implement a predicate `accept(+L)` which receives a list of atoms as argument and is true if the automata accepts the word represented by the list.

```
?- accept([d,a,b,a,b,b,b,c,d,c]).
true
?- accept([d,a,b,a,e,d,c]).
false
```

[1] `https://mediathek.hhu.de/watch/c93518b6-9996-4c3b-802c-9fb1f599f6dc`
[2] `https://mediathek.hhu.de/watch/74b8ef41-230e-4df9-a6cc-75fb7f1744d3`



We provide a partial implementation to start with in the Prolog file containing the unit tests for this exercise.

Exercise 4 (SAT Solving in Prolog)
In the following we represent propositional formulae as Prolog compound terms. For the constants true and false we use the terms `cst(true)` and `cst(false)` while the logical operators are represented by `and/2`, `or/2` and `not/1`. For instance, the formula

$$\neg(true \wedge false) \vee false \tag{1}$$

in propositional logic is represented by the following Prolog compound term:

$$or(not(and(cst(true), cst(false))), cst(false)) \tag{2}$$

Implement a predicate `is_true(+F)` which is true if the formula `F` is true.

Examples:

```
?- is_true( or(not(and(cst(true), cst(false))), cst(false)) ).
true

?- is_true( or(not(cst(true)), cst(false)) ).
false
```

If variables are used in a formula, possible solutions for the variables should be found which make the formula true.

```
?- is_true( or(not(and(cst(true), cst(A))), cst(B))).
A = false ? ;
B = true ? ;
false.
```

Hint: Do not use Prolog's negation for the interpretation of the logical negation. Implement a second predicate `is_false(+F)` instead, which is true if the formula `F` is false.

We provide two clauses of `is_true/1` to start with in the Prolog file containing the unit tests for this exercise.