Introduction to Modal Logic

B. Nebel, S. Wölfl — University of Freiburg
Summer 2015 — Department of Computer Science

Exercise Sheet 11

Due: 15-07-2015

The aim of this sheet is to implement a tableaux solver for the modal logic **K**. You may worked, and hand in, your solutions in groups of at most *three* students! In that case please indicate all names in your solution. This week the solution must be handed in on Wednesday *before* the lecture by email to `wenzelmf@tf.uni-freiburg.de`.

You may use one of the programming languages Python 3, Java, C++. On the website of the lecture you will find a working Python 3 implementation of a parser and the internal representation of formulae (see (a) and (d)). You may use this code, but you don't need to.

Exercise 11.1 (1+1+2+1+5 points)

(a) Implement an internal representation of $\mathscr{L}_{\Box}(P)$-formulae of the form

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \varphi \leftrightarrow \varphi \mid \Box\varphi \mid \Diamond\varphi \, .$$

(b) Implement a procedure that eliminates from a given $\mathscr{L}_{\Box}(P)$-formula all occurrences of $\rightarrow$ and $\leftrightarrow$, by replacing formulae of the form $(\varphi \rightarrow \psi)$ by $(\neg\varphi \vee \psi)$ and $(\varphi \leftrightarrow \psi)$ by $((\neg\varphi \vee \psi) \wedge (\varphi \vee \neg\psi))$, respectively.

(c) Implement a procedure that transforms a given formula into negation normal form (NNF). You may assume that the input formula does not have any occurrences of $\rightarrow$ and $\leftrightarrow$.

(d) Write a parser that reads formualae and transforms them into the internal representation (implemented in (a)). The formulae are input as strings with the following notation:

$S \longrightarrow$ `p1` | `p2` | `p3` | ...	for propositional variables
$S \longrightarrow$ `|` $S\ S$	for disjunctions
$S \longrightarrow$ `&` $S\ S$	for conjunctions
$S \longrightarrow$ `-->` $S\ S$	for implications
$S \longrightarrow$ `<->` $S\ S$	for equivalences
$S \longrightarrow$ `!` S	for negations
$S \longrightarrow$ `[]` S | `<>` S	for boxes and diamonds.

(e) Implement the tableaux method for deciding the **K**-validity of an input formula.

Input:
Your program should be called from the command line by :

```
# tableau <formula>
```

as in the following example (which checks the validity of axiom **K**):

```
# tableau "--> [] --> p1 p2 --> [] p1 [] p2"
```

Output:
The last line of the output of your program should be "TRUE", when the input formula is **K**-valid, and "FALSE" otherwise.