CL exercise for
Tutorial 4

Introduction

Objectives

In this tutorial, you will:

- learn more about *sequents* and *combining predicates*;
- derive de Morgan's second law;
- do proofs in *sequent calculus*.

Tasks

Exercises 1 and 2 are mandatory. Exercise 3 is optional. Exercise 4 and 5 are for your interest only.

Submit

a file called `cl-tutorial-4` with your answers (image or pdf). You do *not* need to submit `Things-QuickCheck.hs` – it's just for fun.

Deadline

12:00 Tuesday 17 October

Reminder

Good Scholarly Practice

Please remember the good scholarly practice requirements of the University regarding work for credit.

You can find guidance at the School page

https://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct.

This also has links to the relevant University pages. Please do not publish solutions to these exercises on the internet or elsewhere, to avoid others copying your solutions.

Exercise 1 –mandatory—marked–

Read Chapter 14 (*Sequent Calculus*) of the textbook.

Derive the second of de Morgan's laws

$$\neg(a \wedge b) = \neg a \vee \neg b$$

using a similar argument to the one presented in the textbook for the first law on page 122.

Exercise 2 –mandatory—marked–

Write a proof which reduces the conclusion

$$(x \wedge y) \vee (x \wedge z) \models x \wedge (y \vee z)$$

to premises that can't be reduced further.

Is it universally valid? If not, give a counterexample.

Exercise 3 –optional—marked–

Write a proof which reduces the conclusion

$$\models (x \wedge \neg y) \vee (\neg(x \vee z) \vee (y \vee z))$$

to premises that can't be reduced further.

Expressions φ like $(x \wedge \neg y) \vee (\neg(x \vee z) \vee (y \vee z))$ used in the antecedents and succedents of sequents are called:

- *tautologies* when $\models \varphi$ is valid (the antecedent is empty);
- *contradictions* when $\varphi \models$ is valid (the succedent is empty);

Is $(x \wedge \neg y) \vee (\neg(x \vee z) \vee (y \vee z))$ a tautology, a contradiction, or neither?

~~Exercise 4~~ –optional—not marked–

Do not submit a solution for this exercise. Discuss in tutorials if you wish!

Write proofs which reduce the conclusions

$$\neg a \wedge \neg b \models \neg(a \wedge b)$$

and

$$\neg(a \wedge b) \models \neg a \wedge \neg b$$

to premises that can't be reduced further.

Is one or both universally valid?

- If not, give a counterexample.
- If so, explain how that shows that $\neg a \wedge \neg b = \neg(a \wedge b)$.

~~Exercise 5~~ –optional—not marked–

The file `Things-QuickCheck.hs` contains a template for verifying the validity of sequents using `QuickCheck`. Read the file, pay attention to the comments, and don't worry if there are lines in the first part of the file that you don't understand; those are needed for setting up `QuickCheck`.

We have already provided in the file definitions of the functions `(|=)` and `(||=)` discussed in Tutorial 3.

Define an infix function:

```
(|||=) :: [Predicate Thing] -> [Predicate Thing] -> Bool
```

for checking whether a sequent involving a list of antecedents and a list of succedents is true or false.

`(|=)` and `(||=)` are special cases of `(|||=)`, meaning that:

1. `p |= q` should give the same result as `[p] |||= [q]`;
2. `ps ||= q` should give the same result as `ps |||= [q]`

for any two predicates `p` and `q` and any list of predicates `ps`.

Encode the two properties above as Boolean-valued functions

```
prop1 :: Predicate Thing -> Predicate Thing -> Bool
prop2 :: [Predicate Thing] -> Predicate Thing -> Bool
```

and test them with `QuickCheck`.

Can you use `(|||=)` and `QuickCheck` to verify your answers to Exercises 3 and 4 (if you did them)?