

TUTORIAL 5 · [COMPUTATION AND LOGIC]



OBJECTIVES

In this tutorial, you will:

- learn more about *sequents* and *combining predicates*;
- derive new rules in the sequent calculus for *exclusive or*, *implication*, and *equivalence*;
- do proofs in *sequent calculus*.



TASKS

Exercises 1–5 are mandatory. Exercise 6 is optional.



SUBMIT a file called `cl-tutorial-5` with your answers (image or pdf) and the file `cl-tutorial-5-QuickCheck.hs` with your code.



DEADLINE Saturday, 24th of October, 4 PM UK time


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

<http://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 Please do not publish solutions to these exercises on the internet or elsewhere, to avoid others copying your solutions.

 For any two predicates a and b , we define **exclusive or** as:


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

We derive a rule for \oplus by reducing the sequent $\vdash (a \wedge \neg b) \vee (\neg a \wedge b)$:

$$\frac{\vdash a, b \quad a, b \vdash}{\vdash a \oplus b}$$

We add this derived rule to sequent calculus as:

$$\frac{\Gamma \vdash a, b, \Delta \quad \Gamma, a, b \vdash \Delta}{\Gamma \vdash a \oplus b, \Delta} \oplus R$$

 Reduce the sequent $(a \wedge \neg b) \vee (\neg a \wedge b) \vdash$ and use the result to derive the rule $(\oplus L)$.

EXERCISE 2

MANDATORY | ⌚ BEFORE TUTORIAL SESSION

🧠 For any two predicates a and b , we define **implication** (also known as *conditional*) as:

$$a \rightarrow b = \neg a \vee b$$

✎ Reduce the sequents

$$\vdash \neg a \vee b$$


and

$$\neg a \vee b \vdash$$


and use the results to write rules ($\rightarrow R$) and ($\rightarrow L$), respectively.

EXERCISE 3

MANDATORY |  BEFORE TUTORIAL SESSION

 For any two predicates a and b , we define **equivalence** (also known as *biconditional*) as:

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

 Reduce the sequents

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


and

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


and use the results to write rules ($\leftrightarrow R$) and ($\leftrightarrow L$), respectively.


EXERCISE 4

MANDATORY |  BEFORE TUTORIAL SESSION

 Use reductions to decide whether the following equations are universally valid:

1. $a \leftrightarrow b = (a \rightarrow b) \wedge (b \rightarrow a)$
2. $(a \rightarrow b) \rightarrow c = a \rightarrow (b \rightarrow c)$
3. $(a \leftrightarrow b) \leftrightarrow c = a \leftrightarrow (b \leftrightarrow c)$
4. $(a \oplus b) \oplus c = a \oplus (b \oplus c)$

 Recall that in order to decide whether an equation $x = y$ is universally valid, we need to write proofs which reduce the sequents $x \vDash y$ and $y \vDash x$.

 We can write proofs to reduce sequents of the form $\models \varphi$ in order to derive simpler equivalent expressions to φ . To achieve this, we need to move every predicate in the premises of the derived rule to the right of the turnstile, using negation.

For example, for the rule for exclusive or

$$\frac{\models a, b \quad a, b \models}{\models (a \wedge \neg b) \vee (\neg a \wedge b)} \quad \text{we obtain} \quad \frac{\models a, b \quad \models \neg a, \neg b}{\models (a \wedge \neg b) \vee (\neg a \wedge b)}$$


which is equivalent to

$$\frac{\models a \vee b \quad \models \neg a \vee \neg b}{\models (a \wedge \neg b) \vee (\neg a \wedge b)}$$


It follows that $(a \wedge \neg b) \vee (\neg a \wedge b)$ is equivalent to $(a \vee b) \wedge (\neg a \vee \neg b)$.

EXERCISE 5 (CONT.)

MANDATORY | 1 BEFORE TUTORIAL SESSION

 In this way we produce a conjunction of disjunctions of literals, where a **literal** is a predicate or the negation of a predicate.


We say that an expression is in **conjunctive normal form (CNF)** if it consists of a conjunction of disjunctions of literals.

 Using the technique outlined on the previous slide, find equivalent CNFs for the following expressions:

1. $r \leftrightarrow (a \wedge b)$
2. $r \leftrightarrow (a \vee b)$
3. $r \leftrightarrow (a \rightarrow b)$
4. $r \leftrightarrow (\neg a)$

EXERCISE 6

OPTIONAL |  BEFORE TUTORIAL SESSION

 The file `cl-tutorial-5-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 (as in Tutorial 4).

 Add to `cl-tutorial-5-QuickCheck.hs` three functions:

```
(+::+) :: Predicate u -> Predicate u -> Predicate u
(-:>) :: Predicate u -> Predicate u -> Predicate u
(<:>) :: Predicate u -> Predicate u -> Predicate u
```

that compute the exclusive or (`+::+`), implication (`-:>`), and biconditional (`<:>`) of two predicates.


EXERCISE 6 (CONT.)

OPTIONAL |  BEFORE TUTORIAL SESSION

 Define a function

```
(|=|) :: Predicate u -> Predicate u -> Bool
```

to check, for every predicates p and q , whether both sequents $p \models q$ and $q \models p$ are valid.

 Write a property and use [QuickCheck](#) to verify the equivalence of $p \models q$ and $\models p \leftrightarrow q$, where p and q are predicates.

 Can you use [QuickCheck](#) to verify your answers to Exercises 4 and 5?