TUTORIAL 4 · [COMPUTATION AND LOGIC]

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 1–4 are mandatory. Exercise 5 is optional.

SUBMIT a file called `cl-tutorial-4` with your answers (image or pdf) and the file `Things-QuickCheck.hs` with your code.

DEADLINE Saturday, 17th 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 Please do not publish solutions to these exercises on the internet or elsewhere, to avoid others copying your solutions.

EXERCISE 1

MANDATORY | BEFORE TUTORIAL SESSION

Read Chapter 10 (*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 83.

SOLUTION TO EXERCISE 1

We start with the following proofs:

$$\dfrac{\dfrac{\neg a \vDash c}{\neg c \vDash a}\,\text{contra-position} \qquad \dfrac{\neg b \vDash c}{\neg c \vDash b}\,\text{contra-position}}{\dfrac{\neg c \vDash a \wedge b}{\neg(a \wedge b) \vDash c}\,\text{contraposition}}\,\wedge$$

and

$$\dfrac{\neg a \vDash c \qquad \neg b \vDash c}{\neg a \vee \neg b \vDash c}\,\vee$$

SOLUTION TO EXERCISE 1 (CONT.)

Combining them gives the equivalence

$$\begin{array}{c} \neg(a \wedge b) \vDash c \\ \hline\hline \neg a \vee \neg b \vDash c \end{array}$$

from which we get

$$\dfrac{\dfrac{}{\neg a \vee \neg b \vDash \neg a \vee \neg b}\ \text{immediate}}{\neg(a \wedge b) \vDash \neg a \vee \neg b} \qquad \dfrac{\dfrac{}{\neg(a \wedge b) \vDash \neg(a \wedge b)}\ \text{immediate}}{\neg a \vee \neg b \vDash \neg(a \wedge b)}$$

Interpreting $\vDash$ as set inclusion, we obtain the desired equation:

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

EXERCISE 2

MANDATORY | BEFORE TUTORIAL SESSION

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.

SOLUTION TO EXERCISE 2

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{I}
\UnaryInfC{$x, y \vDash x$}
\doubleLine
\RightLabel{$\wedge L$}
\UnaryInfC{$x \wedge y \vDash x$}
\AxiomC{}
\RightLabel{I}
\UnaryInfC{$x, y \vDash y, z$}
\doubleLine
\RightLabel{$\vee R$}
\UnaryInfC{$x, y \vDash y \vee z$}
\doubleLine
\RightLabel{$\wedge L$}
\UnaryInfC{$x \wedge y \vDash y \vee z$}
\doubleLine
\RightLabel{$\wedge R$}
\BinaryInfC{$x \wedge y \vDash x \wedge (y \vee z)$}
\AxiomC{}
\RightLabel{I}
\UnaryInfC{$x, z \vDash x$}
\doubleLine
\RightLabel{$\wedge L$}
\UnaryInfC{$x \wedge z \vDash x$}
\AxiomC{}
\RightLabel{I}
\UnaryInfC{$x, z \vDash y, z$}
\doubleLine
\RightLabel{$\vee R$}
\UnaryInfC{$x, z \vDash y \vee z$}
\doubleLine
\RightLabel{$\wedge L$}
\UnaryInfC{$x \wedge z \vDash y \vee z$}
\doubleLine
\RightLabel{$\wedge R$}
\BinaryInfC{$x \wedge z \vDash x \wedge (y \vee z)$}
\doubleLine
\RightLabel{$\vee L$}
\BinaryInfC{$(x \wedge y) \vee (x \wedge z) \vDash x \wedge (y \vee z)$}
\end{prooftree}
$$

Because the conclusion has been shown to follow from the empty set of premises, it is universally valid.

EXERCISE 3

Write a proof which reduces the conclusion

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

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

Is $(x \wedge \neg y) \vee (\neg(x \vee z) \vee (y \vee z)$ a tautology?
What about a contradiction?

SOLUTION TO EXERCISE 3

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{I}
\UnaryInfC{$x \vDash x, y, z$}
\AxiomC{}
\RightLabel{I}
\UnaryInfC{$x, y \vDash y, z$}
\doubleLine
\RightLabel{$\neg R$}
\UnaryInfC{$x \vDash \neg y, y, z$}
\doubleLine
\RightLabel{$\wedge R$}
\BinaryInfC{$x \vDash x \wedge \neg y, y, z$}
\AxiomC{}
\RightLabel{I}
\UnaryInfC{$z \vDash x \wedge \neg y, y, z$}
\doubleLine
\RightLabel{$\vee L$}
\BinaryInfC{$x \vee z \vDash x \wedge \neg y, y, z$}
\doubleLine
\RightLabel{$\neg R$}
\UnaryInfC{$\vDash x \wedge \neg y, \neg(x \vee z), y, z$}
\doubleLine
\RightLabel{$\vee R$}
\UnaryInfC{$\vDash (x \wedge \neg y), \neg(x \vee z), (y \vee z)$}
\doubleLine
\RightLabel{$\vee R$}
\UnaryInfC{$\vDash (x \wedge \neg y), (\neg(x \vee z) \vee (y \vee z))$}
\doubleLine
\RightLabel{$\vee R$}
\UnaryInfC{$\vDash (x \wedge \neg y) \vee (\neg(x \vee z) \vee (y \vee z))$}
\end{prooftree}
$$

Because the conclusion follows from the empty set, it is a tautology.

EXERCISE 4

MANDATORY | BEFORE TUTORIAL SESSION

Write proofs which reduce the conclusions

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

and

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

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

SOLUTION TO EXERCISE 4

We first reduce $\neg a \wedge \neg b \vDash \neg(a \wedge b)$:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg b, a, b \vDash a}\;I}{\neg b, a \wedge b \vDash a}\;\wedge L}{\neg b \vDash a, \neg(a \wedge b)}\;\neg R}{\neg a, \neg b \vDash \neg(a \wedge b)}\;\neg L}{\neg a \wedge \neg b \vDash \neg(a \wedge b)}\;\wedge L
$$

Since the conclusion has been shown to follow from the empty set of premises, it is universally valid.

SOLUTION TO EXERCISE 4 (CONT.)

We now reduce $\neg(a \wedge b) \vDash \neg a \wedge \neg b$:

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{I}
\UnaryInfC{$a \vDash a$}
\doubleLine
\RightLabel{$\neg R$}
\UnaryInfC{$\vDash a, \neg a$}
\AxiomC{$b \vDash a$}
\doubleLine
\RightLabel{$\neg R$}
\UnaryInfC{$\vDash a, \neg b$}
\doubleLine
\RightLabel{$\wedge R$}
\BinaryInfC{$\vDash a, \neg a \wedge \neg b$}
\AxiomC{$a \vDash b$}
\doubleLine
\RightLabel{$\neg R$}
\UnaryInfC{$\vDash b, \neg a$}
\AxiomC{}
\RightLabel{I}
\UnaryInfC{$b \vDash b$}
\doubleLine
\RightLabel{$\neg R$}
\UnaryInfC{$\vDash b, \neg b$}
\doubleLine
\RightLabel{$\wedge R$}
\BinaryInfC{$\vDash b, \neg a \wedge \neg b$}
\doubleLine
\RightLabel{$\wedge R$}
\BinaryInfC{$\vDash a \wedge b, \neg a \wedge \neg b$}
\doubleLine
\RightLabel{$\neg L$}
\UnaryInfC{$\neg(a \wedge b) \vDash \neg a \wedge \neg b$}
\end{prooftree}
$$

The proof shows that the conclusion follows from the two premises $a \vDash b$ and $b \vDash a$, meaning that it is true whenever both of those sequents are true. One counterexample is a universe containing a thing x for which $a\,x$ is true and $b\,x$ is false. Another counterexample is a universe containing a thing x for which $b\,x$ is true and $a\,x$ is false.

SOLUTION TO EXERCISE 4 (CONT.)

If both conclusions were universally valid, then $\neg a \wedge \neg b = \neg(a \wedge b)$ would hold since $\models$ corresponds to set inclusion and we would have shown that each side of the equation is a subset of the other.

EXERCISE 5

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.

SOLUTION TO EXERCISE 5

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

gamma |||= delta =
    and [ or [d x | d <- delta]
        | x <- things, and [g x | g <- gamma] ]
```

EXERCISE 5 (CONT.)

OPTIONAL | BEFORE TUTORIAL SESSION

`(|=)` 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?

SOLUTION TO EXERCISE 5 (CONT.)

```haskell
prop1 :: Predicate Thing -> Predicate Thing -> Bool
prop1 p q =
    (p |= q) == ([p] |||= [q])
-- quickCheck prop1

prop2 :: [Predicate Thing] -> Predicate Thing -> Bool
prop2 ps q =
    (ps ||= q) == (ps |||= [q])
-- quickCheck prop2
```