ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Dipl.-Inf. Achim D. Brucker
Dr. Burkhart Wolff

*Computer-supported Modeling and Reasoning*

http://www.infsec.ethz.ch/education/permanent/csmr/
(rev. 16802)
**Submission date:** –

# Propositional Logic

In this lecture you will deepen your knowledge about *propositional logic*, you will prove your first theorems in an interactive theorem prover (Isabelle) and see how paper-and-pencil proofs are related to interactive theorem proving. In particular you will learn how to do forward-style and backward-style proofs (using Isabelle) and how to combine these two techniques.

## 1 Isabelle in a Nutshell

Isabelle is an interactive theorem prover. During an Isabelle session, you will construct proofs of theorems. A proof consists of a number of proof steps, and the Isabelle system will ensure that each step is correct, and thus ultimately that the entire proof is correct. Various degrees of automation can be realized in Isabelle: you can write each step of a proof yourself, or you can let the system do big subproofs or even the entire proof automatically. In the beginning, we will do the former, because we want to understand in detail what a proof looks like.

In the lecture we will use Isabelle 2004. [1] The graphical user interface (an instance of Proof General) is based on the editor (X)Emacs and can be started Proof General

[1] Isabelle is only supported on Unix-like operating systems (e.g. Linux, Solaris, MacOS

by typing[2]

```
Isabelle-2004
```

in a shell. An special configured (X)Emacs will start showing several Isabelle and Proof General related menus.

X-Symbol

**Hint:** For a nice rendering of mathematical symbols, you should enable the X-Symbol package. For doing so, select the box ⟨`Proof-General` ▷ ⟨`Options` ▷ `X-Symbol`⟩⟩. Now select ⟨`Proof-General` ▷ ⟨`Options` ▷ `Save Options`⟩⟩ to enable X-Symbol automatically on every startup.

Isabelle supports a variety of different logics, thus, before we can prove our first theorem, we have to choose the logic we want. As Isabelle does not provide a special setup for propositional logic, we choose first-order logic (FOL) by selecting ⟨`Isabelle/Isar` ▷ ⟨`Logics` ▷ `FOL`⟩⟩. FOL is a superset of propositional logic.

FOL

**Hint:** If you do not want to use the "default" logic (normally higher-order logic (HOL), you must select the logic on every startup of Isabelle.

We are now ready to prove theorems in propositional logic using Isabelle. While doing so, we have to keep several things in mind:

- The rule names for propositional logic used by Isabelle differ from the names used in the lecture. For an overview of the rule names used by Isabelle, see Tab. 1.

  Isabelle rule names

- Whenever we prove something in Isabelle (or in a paper and pencil fashion), we do so in the context of a *theory*. The essential parts of a theory are the definition of some syntax and judgments that are postulated to be true. In Isabelle, this theory is contained in a file whose name ends in `.thy`. We can start a new theory (building upon FOL) named `ex1` by creating a file with the first line

  *theory*

  **theory** ex1 = FOL:

- Isabelle uses several concrete syntaxes to represent mathematical symbols

  symbol representation

X). You can download Isabelle from `http://isabelle.in.tum.de`. If you use Windows and do not want to install Linux on your hard disk, we recommend IsaMorph (`http://www.brucker.ch/projects/isamorph/`) which is a CD-based Linux that already provides Isabelle.

[2] If you have installed Isabelle yourself or if you are using IsaMorph, just type `Isabelle` instead of `Isabelle-2004`.

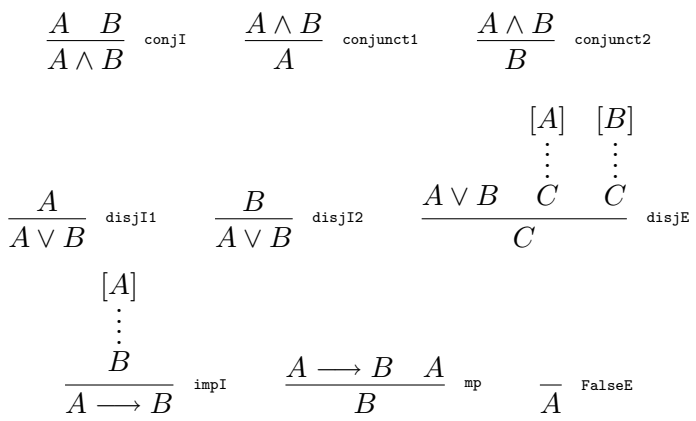


Table 1: Propositional Logic in Isabelle

(see Tab. 2). With enabled X-Symbol mode, you will see the the mathematical symbols in the output of Isabelle. You can enter these symbols either by typing their ASCII representation (most of the symbols will be automatically converted to their mathematical representation) or entering their internal name. Also selecting the symbols within the ⟨`X-Symbol` ▷ ...⟩ menu is possible.

## 2 Our first theorem

Open a new file[3] `simple.thy` and enter the following skeleton of a theory file:

**theory** simple = FOL:

**end**

**Hint:** Isabelle requires, that the file name (without extension) is identical to the theory name.

You can now start the Isabelle process by clicking on the ⟨`Next`⟩ button; after a short startup time, the first line of your theory should be highlighted.

[3] Click on the menu ⟨`File` ▷ `Open`⟩ and enter `simple.thy` into the (X)Emacs Minibuffer (the very last line of the (X)Emacs window).

| | | | | | |
|---|---|---|---|---|---|
| $[\![$ | [\| | \<lbrakk> | $\exists!$ | EX!, :?! | \<exists>! |
| $]\!]$ | \|] | \<rbrakk> | $\epsilon$ | SOME, @ | \<epsilon> |
| $\Longrightarrow$ | ==> | \<Longrightarrow> | $\circ$ | o | \<circ> |
| $\bigwedge$ | !! | \<And> | $\|\|$ | abs | \<bar> \<bar> |
| $\equiv$ | == | \<equiv> | $\leq$ | <= | \<le> |
| $\rightleftharpoons$ | == | \<rightleftharpoons> | $\times$ | * | \<times> |
| $\rightharpoonup$ | => | \<rightharpoonup> | $\in$ | : | \<in> |
| $\leftharpoondown$ | <= | \<leftharpoondown> | $\notin$ | ~: | \<notin> |
| $\lambda$ | % | \<lambda> | $\subseteq$ | <= | \<subseteq> |
| $\Rightarrow$ | => | \<Rightarrow> | $\subset$ | < | \<subset> |
| $\wedge$ | & | \<and> | $\cup$ | Un | \<union> |
| $\vee$ | \| | \<or> | $\cap$ | Int | \<inter> |
| $\longrightarrow$ | --> | \<longrightarrow> | $\bigcup$ | UN,UNION | \<Union> |
| $\neg$ | ~ | \<not> | $\bigcap$ | INT, Inter | \<Inter> |
| $\neq$ | ~= | \<noteq> | $^*$ | ^* | \<^sup>* |
| $\forall$ | ALL, ! | \<forall> | $^{-1}$ | ^-1 | \<inverse> |
| $\exists$ | EX, ? | \<exists> | | | |

Table 2: Mathematical Symbols, Their ASCII-Equivalents and Internal Names

## 2.1 Backward-Style Reasoning in Isabelle

We will now prove $A \longrightarrow (B \longrightarrow A)$ in backward style. Therefore we begin by
lemma entering our proof goal

**lemma** first_theorem: "A −−> (B −−> A)"

as second line of your theory and click on ⟨Next⟩ (this processes your theory one step further). Now, Isabelle will also highlight this line and also will repeat the proof goal in its output window. As you know from the lecture, we have
impI to apply $\rightarrow$-$I$ as first proof step. Using Tab. 1 we see, that $\rightarrow$-$I$ is called impI in Isabelle. You can look up Isabelle's definition of impI by clicking on the
impI ⟨Command⟩ button and entering
thm

**thm** impl

Minibuffer in the (X)Emacs Minibuffer (the very last line of the (X)Emacs window). Isabelle will print its version of the implication introduction rule in its output
apply area. We apply this rule to the current proof state by writing
rule

**apply** ( *rule* impl)

and processing the theory one step. Can you explain the proof step after executing this rule? Applying impI *resolves* the rule impI and the previous goal $A \longrightarrow (B \longrightarrow A)$ to $A \Longrightarrow B \longrightarrow A$. Put very suggestively, our current state says: if we can prove $B \longrightarrow A$ under the assumption $A$, we are done.

At the moment, you may find it difficult to understand the difference between $\Longrightarrow$ and $\longrightarrow$, since both somehow seem to stand for implication. However, $\longrightarrow$ is a symbol of propositional logic, which is our *object logic*, i.e., the language we are talking about. In contrast, $\Longrightarrow$ is a symbol of the *meta-logic*, i.e., Isabelle's built-in logic in which other object logics (PL, FOL, HOL, ...) are formalized.

Now apply `impI` a second time (by repeating the above line), you should end up in a state where Isabelle requires to prove $A$ under the assumption "$A$ $B$". This holds trivially, in Isabelle, this is made explicit by the so-called *assumption* (tactic) method. Type `assumption`

**apply** (*assumption*)

and after executing this line, Isabelle should reply with `No Subgoals` which `No Subgoals`
means, there is nothing to prove anymore. The effect of the assumption method is to remove the first (and in this case only) subgoal provided the conclusion to be proven (in this case $A$) is one of the assumptions. This completes our proof of $A \longrightarrow (B \longrightarrow A)$. Try to see that we built the proof tree starting from the bottom. We can now close the proof be entering `done`

**done**

Now, `My first theorem` is a proven theorem which can be used in the same way as any other rule, e.g. `impI`.

Summarizing, you should end up with the following theory file:

```
theory simple = FOL:
lemma "My first theorem": "A --> (B --> A)"
  apply (rule impI)
  apply (rule impI)
  apply (assumption)
  done
end
```

## 2.2 Forward-Style Reasoning in Isabelle

We will now prove $A \longrightarrow (B \longrightarrow A)$ using forward style; Forward proofs mirror more or less directly the structure of a proof tree. Considering the proof tree for $A \longrightarrow (B \longrightarrow A)$, we conclude that applying `impI` twice is a valid proof. Let's start with: with `lemmas`

**lemmas** forward_proof = impI

Convince yourself by executing the `thm forward_proof` that Isabelle is now aware of this theorem. Now let's undo the last step by clicking on undo and change the above line to

**lemmas** forward_proof = impl [**OF** impl]

OF where `OF` takes a list of theorems and applies them to the premises of the first `impI`. Again, check the result of this step by executing `thm forward_proof`. As you see, we are nearly done, we only have choose the right assignment for
of the meta variables. This can be done by changing the above proof to

**lemmas** forward_proof = impl [**OF** impl, **of** A B A]

This results in:

$$([\![A; B]\!] \Longrightarrow A) \Longrightarrow A \longrightarrow B \longrightarrow A$$

discharging where the first part of this formula is an artefact from discharging the assumptions (it says essentially that $A$ has been introduced as assumption during the proof and that possible "candidates for discharge" are $A$ and $B$.)

A further version of this proof adds a particular clean-up that performs the discharging:

**lemmas** forward_proof = impl [**OF** impl, **of** A B A, simplified]

which completes the discharge:

$$([\![A; B]\!] \Longrightarrow True) \Longrightarrow \ldots$$

but, unfortunately, has also the undesired effect to distroy also our conclusion in this case:

$$([\![A; B]\!] \Longrightarrow True) \Longrightarrow True$$

## 2.3 Combining Forward- and Backward-Style Reasoning

Note that one can arbitrarily mix forward- and backward-style reasoning in Isabelle, e.g.

```
lemma third_proof: "A --> (B --> A)"
  apply (rule forward_proof)
  apply (assumption)
  done
```

or even

```
lemma third_proof: "A --> (B --> A)"
  apply (rule impl [OF impl, of A B A] )
  apply (assumption)
  done
```

are valid proofs for $A \longrightarrow (B \longrightarrow A)$.

# 3 Exercises

## 3.1 Exercise 1

Choose four of the following theorems and prove them

- using paper and pencil,
- in Isabelle using backward style, and
- in Isabelle using forward style.

Choose suitable names for the proven theorems, i.e. choose names based on the exercise number, like `ex1_1` for the first one.

1. $A \longrightarrow B \longrightarrow A$
2. $A \wedge B \longrightarrow B \wedge A$
3. $A \wedge B \longrightarrow A \vee B$
4. $A \vee B \longrightarrow B \vee A$
5. $A \wedge (B \wedge C) \longrightarrow A \wedge C$
6. $(A \longrightarrow B \longrightarrow C) \longrightarrow (A \longrightarrow B) \longrightarrow (A \longrightarrow C)$
7. $(A \wedge B) \vee C \longrightarrow (A \vee C) \wedge (B \vee C)$

## 3.2 Exercise 2

Look up the definition of $\neg$ (hint: it's called `not_def`) and execute step-by-step the following proof script:

`not_def`
`fold`
`unfold`

```
lemma "P ∧¬P ⟶R"
  apply (unfold not_def)
  apply (fold not_def)
  oops
```

Explain the proof script in detail. Here, `oops` just abandons our proof.

`oops`

Now proof the following lemmas using Isabelle (either forward or backward style):

1. $P \wedge \neg P \longrightarrow R$
2. $(A \vee B) \wedge \neg A \longrightarrow B$
3. $(A \vee \neg A) \longrightarrow ((A \longrightarrow B) \longrightarrow A) \longrightarrow A$

Keep the last theorem in mind, it will be useful later.

### 3.3 Exercise 3

So far we only used rules of the intuitionistic propositional logic. We will now add one further rule

$$\dfrac{\begin{array}{c}[\neg A]\\ \vdots\\ A\end{array}}{A}\ \textit{classical}$$

*classical* to obtain *classical* propositional logic. The characteristic of classical logic is that the principle of the excluded middle holds: $P \vee \neg P$.

excluded middle We show that *classical* is equivalent to the principle of the excluded middle. As above, do the proofs both using paper and pencil and in Isabelle.

1. $(\neg Q \longrightarrow P) \longrightarrow P \vee Q$ (hint: the main part of this proof is a proof of $P \vee Q$ using, among others, the assumption $\neg(P \vee Q)$, followed by an application of *classical*).
2. Using the previous theorem, prove $P \vee \neg P$ (hint: first prove $\neg P \vee P$).
3. Prove $P \vee \neg P \longrightarrow ((\neg P \longrightarrow P) \longrightarrow P)$ intuitionistically.

### 3.4 Exercise 4

Prove the following classical theorem called *Peirce's law*, both using paper and pencil and in Isabelle:

$$((A \longrightarrow B) \longrightarrow A) \longrightarrow A$$

Hing: Use the proof of $P \vee \neg P$ from Ex. 3.