Propositional Logic

CS 3234: Logic and Formal Systems

Martin Henz and Aquinas Hobor

August 26, 2010

Generated on Friday 1añ October, 2010, 14:40

1 Motivation

In traditional logic, terms represent sets, and therefore, propositions are limited to stating facts on sets. We have no straightforward way of formulating propositions such as:

- "$1 + 1 = 3$"
- "The sun is shining today."
- "Earth has more mass than Mars."

In this section, we therefore generalize propositions to state anything we like, leading to the notion of *propositional atoms*, described in the following section.

For traditional logic, we have considered particular forms of arguments that combine propositions. For example,

> If p_1 then p_2.

We would like to study in what ways propositions can be combined into arguments, which will also be propositions. This leads us to the notion of *logical operators* in Section 3. Using these operators, we will define *formulas* in Section 4 and their semantics in Section 5. Section 6 gives a proof theory, and Section **??** states its soundness and completeness.

2 Propositional Atoms

We allow any kind of proposition, for example "The sun is shining today", as long as in a particular context (model), we can unambiguously state whether the proposition holds or not. It is convenient to give short names to propositions, such as p, q, p_1, etc, instead of sentences like "The sun is shining today". More formally, we fix a set A of propositional atoms.

In Coq, it is convenient to identify atoms with numerals.

```
Inductive Num : Type :=
  | Zero : Num
  | Succ : Num -> Num.

Definition Atom : Type := Num.
```

This provides us with a sufficiently large (namely countably infinite) supply of propositional atoms. When we need to, we can define Coq variables to refer to particular atoms, such as

```
Definition p_0 : Atom := Zero.
Definition p_1 : Atom := Succ Zero.
Definition p_2 : Atom := Succ (Succ Zero).
```

How can we give meaning to propositional atoms? As in traditional logic, a *model* assigns truth values to each atom. Thus a model for a propositional logic for the set A of atoms is a mapping from A to $\{T, F\}$. Models for propositional logic are called *valuations*.

Example 1. *Let $A = \{p, q, r\}$. Then a valuation v_1 might assign p to T, q to F and r to T. Thus, more formally, $p^{v_1} = T$, $q^{v_1} = F$, and $r^{v_1} = T$. To emphasize that v_1 is a function, we also write $v_1(p)$ instead of p^{v_1}.*

Another valuation v_2 might assign p to F, q to F and r to F, more formally $v_2(p) = v_2(q) = v_2(r) = F$.

In Coq, we define two truth values as constants of the type `Bool`, as follows.

```
Record Bool : Type :=
  | T : Bool
  | F : Bool.
```

Then a valuation can be defined as a function from `atom` to `Bool`.

```
Definition Valuation : Type := Atom -> Bool.

Definition exampleValuation : Valuation :=
  fun a : Atom => match a with
    | Zero                    => T   (* p_0       *)
    | Succ Zero               => F   (* p_1       *)
    | Succ (Succ Zero)        => T   (* p_2       *)
    | _                       => F   (* all other *)
  end.
```

3 Constructing Propositions

We would like to build larger propositions, such as arguments, out of smaller ones, such as propositional atoms. We do this using *operators* that can be applied to propositions, and yield propositions.

3.1 Unary Operators

Let us consider a unary operator op for propositions. We said that a valuation v unambiguously fixes the meaning of each propositional atom. If the meaning of a propositional atom p is fixed to $v(p)$—T or F—as given by the chosen valuation v, then the meaning of $op\ p$ should depend only on $v(p)$. For a unary operator, there are only four choices of such a meaning, and thus four possible unary operators:

1. $(op(p))^v = F$ if $p^v = F$ and $(op(p))^v = F$ if $p^v = T$
2. $(op(p))^v = T$ if $p^v = F$ and $(op(p))^v = T$ if $p^v = T$
3. $(op(p))^v = F$ if $p^v = F$ and $(op(p))^v = T$ if $p^v = T$
4. $(op(p))^v = T$ if $p^v = F$ and $(op(p))^v = F$ if $p^v = T$

The first and second do not depend on their argument, and thus the argument can be omitted, leading to them being *nullary* operators, as described in the next section.

The third one is the identity function ($(op(p))^v = p^v$) and therefore can be omitted.

The fourth operator *negates* its argument, T becomes F and F becomes T. Thus, we call this operator *negation*, and write $\neg p$ (pronounced "not p").

3.2 Nullary Operators

We have seen that an operator assigns T to its argument, regardless of the valuation, and regardless of the argument. Since it does not depend on its argument, we may omit any argument. This means that it is just a constant. We denote the nullary operator whose value is T in every valuation by the symbol $\top$.

Similarly, we denote the nullary operator, whose value is F in every valuation by the symbol $\bot$.

3.3 Binary Operations

For binary operations, we have the following 16 choices.

p	q	$op_1(p,q)$	$op_2(p,q)$	$op_3(p,q)$	$op_4(p,q)$
F	F	F	F	F	F
F	T	F	F	F	F
T	F	F	F	T	T
T	T	F	T	F	T

p	q	$op_5(p,q)$	$op_6(p,q)$	$op_7(p,q)$	$op_8(p,q)$
F	F	F	F	F	F
F	T	T	T	T	T
T	F	F	F	T	T
T	T	F	T	F	T

p	q	$op_9(p,q)$	$op_{10}(p,q)$	$op_{11}(p,q)$	$op_{12}(p,q)$
F	F	T	T	T	T
F	T	F	F	F	F
T	F	F	F	T	T
T	T	F	T	F	T

p	q	$op_{13}(p,q)$	$op_{14}(p,q)$	$op_{15}(p,q)$	$op_{16}(p,q)$
F	F	T	T	T	T
F	T	T	T	T	T
T	F	F	F	T	T
T	T	F	T	F	T

We single out three of these 16 operators, and give them particular names:

$\boldsymbol{op_2}$: The operator application $op_2(p,q)$ evaluates to T when p evaluates to T *and* q evaluates to T, and F otherwise. Therefore, we say "p and q". This operator is called *conjunction*, and traditionally, the symbol $\wedge$ is used to denote it.

$$p \wedge q := op_2(p,q)$$

$\boldsymbol{op_8}$: The operator application $op_8(p,q)$ evaluates to T when p evaluates to T *or* q evaluates to T, and F otherwise. Therefore, we say "p or q". This operator is called *disjunction*, and traditionally, the symbol $\vee$ is used to denote it.

$$p \vee q := op_8(p,q)$$

$\boldsymbol{op_{14}}$: The operator application $op_{14}(p,q)$ evaluates to T when p evaluates to F *or* q evaluates to T, and F otherwise. Therefore, we say "p implies q". This operator is called *implication*, and traditionally, the symbol $\rightarrow$ is used to denote it.

$$p \rightarrow q := op_{14}(p,q)$$

4 Syntax of Propositional Logic

In propositional logic, we would like to apply operators not only to atomic propositions, but also to the result of applying other operators. This means that our language of *well-formed formulas in propositional logic* should be inductively defined as follows.

Definition 1. *For a given set A of propositional atoms, the set of* well-formed formulas in propositional logic *is the least set F that fulfills the following rules:*

- *The constant symbols $\bot$ and $\top$ are in F.*
- *Every element of A is in F.*
- *If ϕ is in F, then $(\neg\phi)$ is also in F.*
- *If ϕ and ψ are in F, then $(\phi \wedge \psi)$ is also in F.*
- *If ϕ and ψ are in F, then $(\phi \vee \psi)$ is also in F.*
- *If ϕ and ψ are in F, then $(\phi \to \psi)$ is also in F.*

Example 2. *The expression*

$$(((\neg p) \wedge q) \to (\top \wedge (q \vee (\neg r))))$$

is a well-formed formula in propositional logic.

A more compact definition of the syntax of propositional logic is given in Backus Naur Form (BNF), as follows:

$$\phi ::= p \mid \bot \mid \top \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \to \phi)$$

where p stands for any atomic proposition.

In Coq, we use the keyword `Inductive` to introduce the inductively defined set `PropFormula`.

```
Inductive PropFormula : Type :=
  | propTop : PropFormula
  | propBot : PropFormula
  | propAtom : Atom -> PropFormula
  | propNeg : PropFormula -> PropFormula
  | propConj : PropFormula -> PropFormula -> PropFormula
  | propDisj : PropFormula -> PropFormula -> PropFormula
  | propImpl : PropFormula -> PropFormula -> PropFormula.

Definition exampleFormula : PropFormula :=
  propDisj (propNeg (propAtom p_2))
                (propImpl (propConj (propAtom p_1) propTop)
                                    propBot).
```

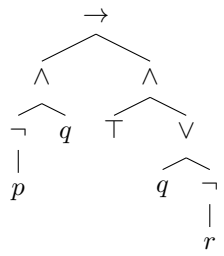
The grammar above is non-ambiguous. We can show that a particular formula is well-formed by looking for its outermost operator (the one belonging to the outermost parentheses). There is only one rule applicable for this operator, and thus we know whether the formula is an atom, a negation, a conjunction, a disjunction, or an implication. This reasoning is called the *inversion principle*. We then apply the same process recursively to the arguments of the operator.

This leads to a hierarchical structure for every formula, where the outermost operator appears at the root, and its arguments are written as branches below it. This structure is called the *parse tree* of the formula.

Example 3. *The formula*

$$(((\neg p) \wedge q) \rightarrow (\top \wedge (q \vee (\neg r))))$$

has the following parse tree.



A formula ψ whose root operator appears below the root operator of a formula ϕ is called a *sub-formula* of ϕ. In the example above, $(q \vee (\neg r))$ is a sub-formula of $(\top \wedge (q \vee (\neg r)))$. We define that every formula is a sub-formula of itself.

Exercise 1. *List all sub-formulas of the formula in Example 3.*

The definition of well-formed formulas surrounds every operator with parentheses, which leads to a proliferation of symbols and makes formulas unnecessarily hard to read. As in arithmetic, where $*$ binds more tightly than $+$, we introduce the following conventions.

Convention 1. *The negation symbol $\neg$ binds more tightly than $\wedge$ and $\vee$, and $\wedge$ and $\vee$ bind more tightly than $\rightarrow$. Moreover, $\rightarrow$ is* right-associative*: The formula $p \rightarrow q \rightarrow r$ is read as $p \rightarrow (q \rightarrow r)$.*

5 Semantics of Propositional Logic

Recall that in propositional logic, a model is a valuation that maps every propositional atom to one of the truth values T or F. In order to give meaning to well-formed propositional formulas, we need to be able to apply such a valuation to a formula. This process is called *evaluation*.

In order to define evaluation, we introduce the following functions on truth values.

Definition 2. *The function* $\setminus : \{F, T\} \to \{F, T\}$ *is given in the following truth table:*

B	$\setminus B$
F	T
T	F

Definition 3. *The function* $\& : \{F, T\} \times \{F, T\} \to \{F, T\}$ *is given in the following truth table:*

B_1	B_2	$B_1 \& B_2$
F	F	F
F	T	F
T	F	F
T	T	T

Definition 4. *The function* $| : \{F, T\} \times \{F, T\} \to \{F, T\}$ *is given in the following truth table:*

B_1	B_2	$B_1 \mid B_2$
F	F	F
F	T	F
T	F	F
T	T	T

Definition 5. *The function* $\Rightarrow : \{F, T\} \times \{F, T\} \to \{F, T\}$ *is given in the following truth table:*

B_1	B_2	$B_1 \Rightarrow B_2$
F	F	T
F	T	T
T	F	F
T	T	T

Definition 6. *The result of* evaluating *a well-formed propositional formula* ϕ *with respect to a valuation* v*, denoted* $v(\phi)$ *is defined as follows:*

- *If* ϕ *is the constant* $\bot$*, then* $v(\phi) = F$*.*
- *If* ϕ *is the constant* $\top$*, then* $v(\phi) = T$*.*
- *If* ϕ *is an propositional atom* p*, then* $v(\phi) = p^v$*.*
- *If* ϕ *has the form* $(\neg\psi)$*, then* $v(\phi) = \setminus v(\psi)$*.*
- *If* ϕ *has the form* $(\psi \wedge \tau)$*, then* $v(\phi) = v(\psi) \& v(\tau)$*.*
- *If* ϕ *has the form* $(\psi \vee \tau)$*, then* $v(\phi) = v(\psi) \mid v(\tau)$*.*
- *If* ϕ *has the form* $(\psi \to \tau)$*, then* $v(\phi) = v(\psi) \Rightarrow v(\tau)$*.*

In Coq, we define a function `eval` that performs evaluation according to the given rules.

```
Fixpoint eval (f : PropFormula) (v : Valuation) : Bool :=
  match f with
  | propTop =>
    T
  | propBot =>
    F
  | propAtom a =>
    v(a)
  | propNeg f1 =>
    match (eval f1 v) with
      | F => T
      | T => F
     end
  | propConj f1 f2 =>
    match (eval f1 v, eval f2 v) with
     | (F, F) => F
     | (F, T) => F
     | (T, F) => F
     | (T, T) => T
    end
  | propDisj f1 f2 =>
    match (eval f1 v, eval f2 v) with
     | (F, F) => F
     | (F, T) => T
     | (T, F) => T
     | (T, T) => T
    end
  | propImpl f1 f2 =>
    match (eval f1 v, eval f2 v) with
     | (F, F) => T
     | (F, T) => T
     | (T, F) => F
     | (T, T) => T
    end
  end.
```

Definition 7. *A formula is called* valid *if it evaluates to T with respect to every possible valuation.*

Definition 8. *A formula is called* satisfiable *if it evaluates to T with respect to at least one possible valuation.*

A safe way to check whether a particular formula is valid or satisfiable, is to evaluate with respect to *every* possible valuation.

Example 4. *For example, we can verify the validity of the formula*

$$\phi = p \wedge q \rightarrow p \vee q$$

by contructing the following truth table.

p	q	$p \wedge q$	$p \vee q$	$p \wedge q \rightarrow p \vee q$
F	F	F	F	T
F	T	F	T	T
T	F	F	T	T
T	T	T	T	T

The last column of the table contains the results of evaluating ϕ with respect to the valuation corresponding to the respective row. Since there are two distinct variables in ϕ, the truth table has $|\{F,T\}|^2 = 4$ rows. From the fact that the last row contains only T, we conclude that the formula is valid.

Exercise 2. *Is the formula in Example 3 valid? If not, find a counter-example, namely a valuation for which it evaluates to F.*

Exercise 3. *Find a valid formula that contains the propositional atoms p, q, r and w.*

6 Proof Theory

In this section, we introduce a formalism that allows us to construct a proof (graphical and text-based) for a given argument, called *sequent*.

Definition 9. *A sequent consists of propositional formulas $\phi_1, \phi_2, \ldots, \phi_n$, called* premises*, where $n \geq 0$, and a propositional formula ψ called* conclusion*. We write a sequent as follows:*

$$\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$$

and say "ψ is provable using the premises $\phi_1, \phi_2, \ldots, \phi_n$".

6.1 Introducing $\top$

The simplest rule involves our constant $\top$. Since $\top$ always holds, we can introduce it as an axiom.

Axiom 1 ($\top i$)**.**

$$\frac{}{\top}[\top i]$$

In Coq, we simply state:

```
Axiom Top_I :
  holds propTop.
```

6.2 Rules for Conjunction

We can prove a conjunction by proving both parts of the conjunction. Thus we can state the following conjunction introduction axiom.

Axiom 2 ($\wedge$i)**.**

$$\frac{\phi \qquad \psi}{\phi \wedge \psi}[\wedge i]$$

Since conjunction expresses that both its arguments hold, we can just ignore either one, leading to the following conjunction elimination rules.

Axiom 3 ($\wedge e_1$)**.**

$$\frac{\phi \wedge \psi}{\phi}[\wedge e_1]$$

Axiom 4 ($\wedge e_2$)**.**

$$\frac{\phi \wedge \psi}{\psi}[\wedge e_2]$$

Example 5. *Let us to prove the following sequent.*

$$p \wedge q, r \vdash q \wedge r$$

The following proof in graphical notation uses conjunction elimination and conjunction introduction.

$$\dfrac{\dfrac{p \wedge q}{q}[\wedge e_2] \qquad r}{q \wedge r}[\wedge i]$$

The formulas $p \wedge q$ and r are premises of the sequent and therefore do not require any further proof. The text-based version of this proof follows.

1	$(p \wedge q)$	*premise*
2	q	$\wedge e$ *1*
3	r	*premise*
4	$q \wedge r$	$\wedge i$ *2,3*

In Coq, conjunction introduction and elimination rules are expressed as follows.

```
Axiom Conj_I : forall f1 f2,
  holds f1 ->
  holds f2 ->
  holds (propConj f1 f2).


Axiom Conj_E1 : forall f1 f2,
  holds (propConj f1 f2) ->
  holds f1.


Axiom Conj_E2 : forall f1 f2,
  holds (propConj f1 f2) ->
  holds f2.
```

6.3 Double Negation Elimination

We should be able prove ϕ by proving $\neg\neg\phi$, which gives rise to the following elimination rule for double negation.

Axiom 5 ($\neg\neg$e)**.**

$$\frac{\neg\neg\phi}{\phi}[\neg\neg i]$$

```
Axiom NegNeg_E : forall f,
  holds (propNeg (propNeg f)) ->
  holds f.
```

6.4 Implication Elimination

If an implication $\phi \to \psi$ and its premise ϕ are known, we can infer the conclusion ψ.

Example 6. *Assume we know "If it rained, then the street is wet" ($\phi \to \psi$), and also "It rained.". Then, we can conclude that the street is indeed wet (ψ).*

Axiom 6 ($\to$e)**.**

$$\frac{\phi \qquad \phi \to \psi}{\psi}[\to e]$$

This rule is often called "Modus Ponens" (or MP).[1]

6.5 Implication Introduction

The sequent

$$p \to q \vdash \neg\neg p \to q$$

should be provable, but we don't have a rule to introduce implication yet! Our goal is to allow a proof of the following form.

1	$p \to q$	premise
2	$\neg\neg p$	assumption
3	p	$\neg\neg e$ 2
4	q	$\to e$ 1, 3
5	$\neg\neg p \to q$	$\to_i$ 2–4

Thus we start a box with an *assumption*, and use previously proven propositions (including premises) from the outside in the box. Of course, we cannot use assumptions from inside the box in rules outside the box; the purpose of the box is to constrain the assumption to conclusions within the box.

This example gives rise to the following axiom for implication introduction.

Axiom 7 ($\to$i)**.**

$$\frac{\boxed{\begin{array}{c}\phi \\ \vdots \\ \psi\end{array}}}{\phi \to \psi}\,[\to i]$$

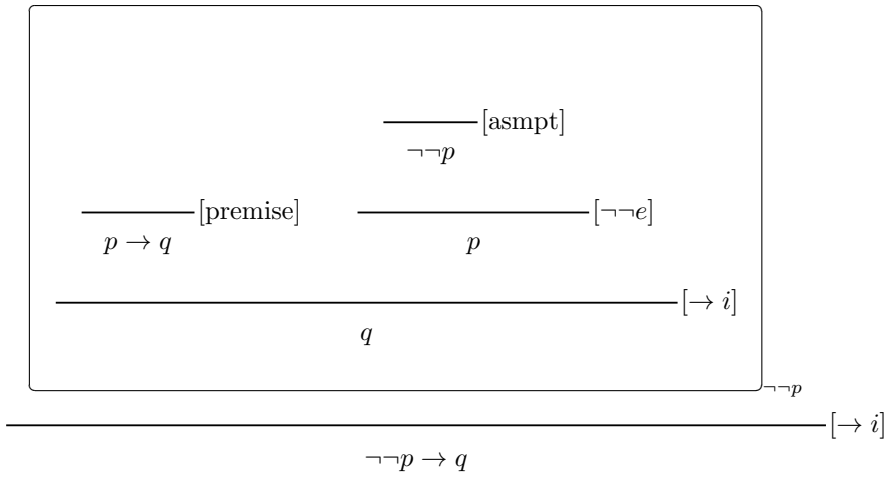
```
Axiom Impl_I : forall f1 f2,
  (holds f1 -> holds f2) ->
  holds (propImpl f1 f2).


Axiom Impl_E : forall f1 f2,
  holds (propImpl f1 f2) ->
  holds f1 ->
  holds f2.
```

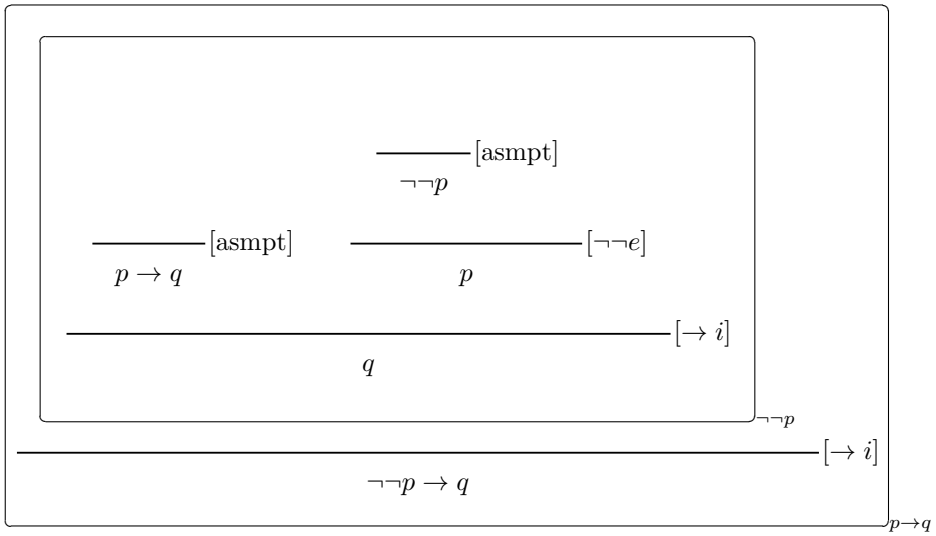
In the text-based notation of the proof, given above, the first line in the box states the assumption, which can be used anywhere inside the box.

[1] "Modus ponens" is an abbreviation of the Latin "modus ponendo ponens" which means in English "mode that affirms by affirming". More precisely, we could say "mode that affirms the antecedent of an implication".

In graphical notation of actual proofs, we annotate the box with the assumption, which can then be introduced anywhere within the box.



One way to visualize a premise of a sequent in a graphical proof is to surround the proof with a box, annotated by the premises as assumptions.



This notation makes clear that assumptions can be introduced in the proof wherever needed, as long as there is a surrounding box annotated by the assumption.

6.6 Rules for Disjunction

Since a disjunction holds if either of its arguments holds, we can introduce disjunction as follows.

Axiom 8 ($\vee i_1$)**.**

$$\frac{\phi}{\phi \vee \psi}\,[\vee i_1]$$

Axiom 9 ($\vee i_2$)**.**

$$\frac{\psi}{\phi \vee \psi}\,[\vee i_2]$$

Disjunction elimination makes use of the arguments as assumptions. If a disjunction is proven, and we can derive the same formula χ using either one of the assumptions, we have proven χ.

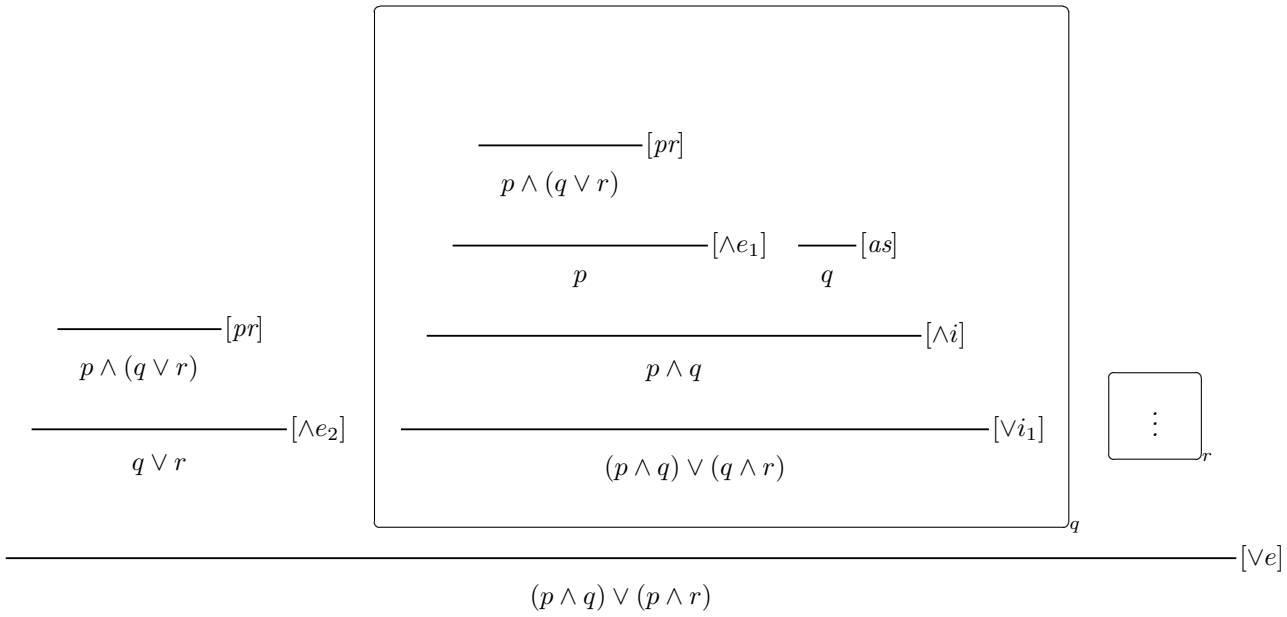
Axiom 10 ($\vee e$)**.**

$$\frac{\phi \vee \psi \qquad \begin{array}{|c|}\hline \phi \\ \vdots \\ \chi \\ \hline \end{array} \qquad \begin{array}{|c|}\hline \psi \\ \vdots \\ \chi \\ \hline \end{array}}{\chi}\,[\vee e]$$

Example 7.

1	$p \wedge (q \vee r)$	*premise*
2	p	$\wedge e_1$ *1*
3	$q \vee r$	$\wedge e_2$ *1*
4	q	*assumption*
5	$p \wedge q$	$\wedge i$ *2,4*
6	$(p \wedge q) \vee (p \wedge r)$	$\vee i_1$ *5*
7	r	*assumption*
8	$p \wedge r$	$\wedge i$ *2,7*
9	$(p \wedge q) \vee (p \wedge r)$	$\vee i_2$ *8*
10	$(p \wedge q) \vee (p \wedge r)$	$\vee e$ *3, 4–6, 7–9*

In graphical notation, we annotate the boxes with the corresponding assumption.



The second box does not fit on the page and is left to the reader. This example demonstrates the practical limitations of the graphical notation. From now onwards, we therefore focus on the text-based notation for proofs.

```
Axiom Disj_I1 : forall f1 f2,
  holds f1 ->
  holds (propDisj f1 f2).


Axiom Disj_I2 : forall f1 f2,
  holds f2 ->
  holds (propDisj f1 f2).


Axiom Disj_E : forall f1 f2 f3,
  holds (propDisj f1 f2) ->
  (holds f1 -> holds f3) ->
  (holds f2 -> holds f3) ->
  holds f3.
```

6.7 Axioms for $\bot$ and Negation

Often during a proof, we make a sufficient number of assumptions such that we can derive $\bot$. This means that the assumptions are mutually contradictory, since $\bot$ can never hold. In such a context, any formula should be provable; we state:

Axiom 11 ($\bot e$)**.**

$$\frac{\bot}{\phi}\,[\bot e]$$

Exercise 4. *A formula of the form $\bot \to \phi$ should be derivable; an implication is false only when its first argument holds, and $\bot$ never holds. Prove $\bot \to \phi$ without any premises.*

```
Axiom Bot_E : forall f,
  holds propBot ->
  holds f.
```

If we can prove a formula and its negation, we clearly have contradictory assumptions, and thus can conclude $\bot$.

Axiom 12 ($\neg e$)**.**

$$\frac{\phi \qquad \neg\phi}{\bot}\,[\neg e]$$

```
Axiom Neg_E : forall f,
  holds f ->
  holds (propNeg f) ->
  holds propBot.
```

If by assuming a formula ϕ, we can derive a contradiction, its negation must hold.

Axiom 13 ($\neg i$)**.**

$$\frac{\boxed{\begin{array}{c}\phi\\ \vdots\\ \bot\end{array}}}{\neg\phi}\,[\neg i]$$

```
Axiom Neg_I : forall f,
  (holds f -> holds propBot) ->
  holds (propNeg f).
```

6.8 Derived Rules

Lemma 1 ($\neg\neg i$)**.** *The following sequent holds for any formula ϕ:*

$$\phi \vdash \neg\neg\phi$$

Proof. We use ϕ in the following proof to represent an arbitrarily chosen formula.

1	ϕ	premise
2	$\neg\phi$	assumption
3	$\bot$	$\neg$e 1,2
4	$\neg\neg\phi$	$\neg$i 2–3

This proof is correct, regardless of the shape of ϕ, and thus the sequent holds for any formula ϕ. □

Such a generalized sequent, once proven, can be used like an axiom. We can state it as follows.

$$\frac{\phi}{\neg\neg\phi}[\neg\neg i]$$

```
Lemma NegNegI : forall f v,
   holds v f -> holds v (propNeg (propNeg f)).

Proof.
 intros.
 apply Neg_I.
 intro.
 apply Neg_E with f.
 trivial.
 trivial.
Qed.
```

A useful result in practice is that for any formula ϕ, the disjunction of ϕ and its negation is always provable.

Lemma 2 (LEM)**.**

$$\frac{}{\phi \vee \neg\phi}[LEM]$$

The twin sister of MP eliminates an implication by infers the negation of the premise from the negation of the conclusion.

Lemma 3 (MT)**.**

$$\dfrac{\phi \to \psi \qquad \neg\psi}{\neg\phi}[MT]$$

Exercise 5. *Prove MT using text-based notation.*

This rule is called "Modus Tollens" (which explains the abbreviation MT).[2]

6.9 Summary

Basic Rules

$$\dfrac{\phi \qquad \psi}{\phi \wedge \psi}[\wedge i] \qquad \dfrac{\phi \wedge \psi}{\phi}[\wedge e_1] \qquad \dfrac{\phi \wedge \psi}{\psi}[\wedge e_2]$$

$$\dfrac{\phi}{\phi \vee \psi}[\vee i_1] \qquad \dfrac{\psi}{\phi \vee \psi}[\vee i_2] \qquad \dfrac{\phi \vee \psi \qquad \boxed{\begin{array}{c}\phi\\ \vdots\\ \chi\end{array}} \qquad \boxed{\begin{array}{c}\psi\\ \vdots\\ \chi\end{array}}}{\chi}[\vee e]$$

$$\dfrac{\boxed{\begin{array}{c}\phi\\ \vdots\\ \psi\end{array}}}{\phi \to \psi}[\to i] \qquad \dfrac{\phi \qquad \phi \to \psi}{\psi}[\to e]$$

[2] "Modus tollens" is an abbreviation of the Latin "modus tollendo tollens" which means in English "mode that denies by denying". More precisely, we could say "mode that denies the consequent of an implication".

$$\dfrac{\boxed{\begin{array}{c}\phi\\ \vdots\\ \bot\end{array}}}{\neg\phi}[\neg i] \qquad \dfrac{\phi \quad \neg\phi}{\bot}[\neg e]$$

$$\dfrac{\bot}{\phi}[\bot e] \qquad \dfrac{\neg\neg\phi}{\phi}[\neg\neg e]$$

Some Derived Rules

$$\dfrac{\phi}{\neg\neg\phi}[\neg\neg i]$$

$$\dfrac{}{\phi \vee \neg\phi}[\text{LEM}]$$

$$\dfrac{\phi \to \psi \qquad \neg\psi}{\neg\phi}[MT]$$

7 Soundness and Completeness

Intuitively, the notions of provability (Section 6), and validity (Section 5) should be closely related. Of course, we expect that any formula the we can prove, to be valid. On the other hand, we also would like to guarantee that a proof exists, whenever a formula is valid.

In order to formally state the corresponding results, we first introduce the semantic notion of entailment.

Definition 10. *If, for all valuations in which all $\phi_1, \phi_2, \ldots, \phi_n$ evaluate to* `T`*, the formula ψ evaluates to* `T` *as well, we say that $\phi_1, \phi_2, \ldots, \phi_n$ semantically entail ψ, written:*

$$\phi_1, \phi_2, \ldots, \phi_n \models \psi$$

Now we can state the main results on the proof theory of propositional logic.

Theorem 1 (Soundness of Propositional Logic)**.** *Let* $\phi_1, \phi_2, \ldots, \phi_n$ *and* ψ *be propositional formulas. If* $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$*, then* $\phi_1, \phi_2, \ldots, \phi_n \models \psi$*.*

Theorem 2 (Completeness of Propositional Logic)**.** *Let* $\phi_1, \phi_2, \ldots, \phi_n$ *and* ψ *be propositional formulas. If* $\phi_1, \phi_2, \ldots, \phi_n \models \psi$*, then* $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$*.*