Computer Supported Modeling and Reasoning

David Basin, Achim D. Brucker, Jan-Georg Smaus, and Burkhart Wolff

April 2005

http://www.infsec.ethz.ch/education/permanent/csmr/

Propositional Logic

David Basin

Propositional Logic: Overview

- System for formalizing certain valid patterns of reasoning
- Expressions built by combining "atomic propositions" using not, if . . . then . . . , and, or, etc.
- Validity means no counterexample. Depends on form of the expressions $\Rightarrow$ can make patterns explicit by replacing words by symbols

$$\frac{A \rightarrow B \quad A}{B}$$

From if A then B and A it follows that B.

- What about

  From if A then B and B it follows that A?

More Examples (Which are Valid?)

1. If it is Sunday, then I don't need to work.
   It is Sunday.
   Therefore I don't need to work.
2. It will rain or snow.
   It will not snow.
   Therefore it will rain.
3. The Butler is guilty or the Maid is guilty.
   The Maid is guilty or the Cook is guilty.
   Therefore either the Butler is guilty or the Cook is guilty.

History

- Propositional logic was developed to make this all precise.
- Laws for valid reasoning were known to the Stoic philosophers (about 300 BC).
- The formal system is often attributed to George Boole (1815-1864).

Formal Systems

Formalization allows us to "turn the crank".

Phrases like "from . . . it follows" or "therefore" are formalized as derivation rules, e.g.

$$\frac{A \to B \quad A}{B} \to\text{-}E$$

Rules are grafted together to build trees called derivations.

This defines a proof system in the style of natural deduction.

Formalizing Propositional Logic

- We must formalize
  (a) Language
  (b) Deductive system
- Here we will focus on formalizing the deductive machinery and implicitly assume semantics and metatheorems (soundness and completeness).
- For labs we will carry out proofs using the Isabelle System. Isabelle supports a Natural Deduction deductive system.

Propositional Logic: Language

Propositions are built from a collection of (propositional) variables and closed under disjunction, conjunction, implication, . . .

Propositional Logic: Language (2)

More formally: Let a set V of variables be given. L_P, the language of propositional logic over V, is the smallest set where:

- X in L_P if X in V.
- $\bot$ in L_p.
- $(A \wedge B)$ in L_P if A in L_P and B in L_P.
- $(A \vee B)$ in L_P if A in L_P and B in L_P.
- $(A \rightarrow B)$ in L_P if A in L_P and B in L_P.
- $(\neg A)$ in L_P if A in L_P.

The elements of L_P are called (propositional) formulas.

We omit unnecessary brackets.

More Detailed Explanations

What is Validity (of a Pattern of Reasoning)?

Let A and B are symbols (variables) standing for arbitrary propositions. Then

From if A then B and A it follows that B

is valid because it is true regardless of what A and B are. In other words, there is no pair of propositions for A and B that makes this rule false (this would be a counter example). The validity of this (propositional) pattern of reasoning is based on the fact that any proposition must be either true or false; therefore, we have only to check for all combinations in order to establish validity.

A formalization of a "pattern of reasoning" is a logical rule. Therefore, we also speak of a valid rule.

An Invalid Pattern

From if A then B and B it follows that A

is invalid because there is a counterexample:

Let A be "Kim is a man" and B be "Kim is a person".

More Examples (Which are Valid?)

1. If it is Sunday, then I don't need to work.
   It is Sunday.
   Therefore I don't need to work. VALID
2. It will rain or snow.
   It is too warm for snow.
   Therefore it will rain. VALID
3. The Butler is guilty or the Maid is guilty.
   The Maid is guilty or the Cook is guilty.
   Therefore either the Butler is guilty or the Cook is guilty. NOT VALID

Turning the Crank

By formalizing patterns of reasoning, we make it possible for such reasoning to be checked or even carried out by a computer.

From known patterns of reasoning new patterns of reasoning can be constructed.

What does Formalization Mean?

At this stage, we are content with a formalization that builds on geometrical notions like "above" or "to the right of". In other words, our formalization consists of geometrical objects like trees.

We study formalization in more detail later.

Proof Systems

By a proof system or deductive system we mean a mechanism that allows for the construction logical statements (e.g. valid formulae) from other statements by purely syntactic means.

In particular, a deductive system can be given by a collection of rules, or a "calculus".

We call the rules in that particular set basic rules. Later we will see one can also derive rules.

We will see for example natural deduction calculi and sequent calculi for various logics in this course.

Soundness and Completeness

A proof system is sound if only valid propositions can be derived in it.

A proof system is complete if all valid propositions can be derived in it.

What is a Meta-theorem?

A metatheorem is a theorem about a proof system, as opposed to a theorem derived within the proof system. The statements "proof system XYZ is sound" or "proof system XYZ is complete" are meta-theorems.

What is a Language?

By language we mean the language of formulae. We can also say that we define the (object) logic. Here "logic" is used in the narrower sense.

What is Semantics?

As we mentioned earlier, we will not say much about semantics in this lecture.

The standard semantics for propositional logic builds on the semantic domain of truth values $\{True, False\}$, i.e., each formula is interpreted as either $True$ or $False$. To interpret a formula, we need an assignment of the propositional variables. Each variable must be assigned one of the values $True$ or $False$. For each syntactic construct of propositional logic, it is specified how it must be interpreted as a function from truth values to truth values.

We go into more detail later.

Two formulae are equivalent if they yield the same truth value for any assignment of the propositional variables.

What does Open Assumption Mean?

For example, all logical statements at the leaves of the proof:

$$\dfrac{\dfrac{A \to (B \to C) \quad A}{B \to C}\,{\to}\text{-}E \qquad B}{C}\,{\to}\text{-}E$$

are open. For the moment, it suffices to know that when an assumption is made, it is initially an open assumption.

What is $\vdash$?

This symbol is used to mark a form of logical statement. By writing $A \to (B \to C), A, B \vdash C$, we assert that C can be derived in this proof system under the open assumptions $A \to (B \to C)$, A, B.

This form of logical statements gives rise to explained later.

Why is this Example Abstract?

Natural deduction is not just about propositional logic! We explain here the general principles of natural deduction, not just the application to propositional logic.

In order to emphasize that applying natural deduction is a completely mechanical process, we give an example that is void of any intuition.

It is important that you understand this process. Applying rules mechanically is one thing. Understanding why this process is semantically justified is another.

How to Read these Rules

The first rule reads: if at some root of a tree in the forest you have constructed so far, there is a ♦, then you are allowed to draw a line underneath that ♦ and write ♣ underneath that line.

The third rule reads: if the forest you have constructed so far contains two neighboring trees, where the left tree has root ♣ and the right tree has root ♠, then you are allowed to draw a line underneath those two roots and write ♥ underneath that line.

How to Read these Rules (2)

The last rule reads: if at some root of a tree in the forest you have constructed so far, there is a ♥, then you are allowed to draw a line underneath that ♥ and write ♦ underneath that line. Moreover you are allowed to discharge (eliminate, close) 0 or more occurrences of ♦ at the leaves of the tree.

Discharging is marked by writing [] around the discharged formula.

Note that generally, the tree may contain assumptions other than ♦ at the leaves. However, these must not be discharged in this rule application. They will remain open until they might be discharged by some other rule application later.

Making Assumptions

In everyday language, "making an assumption" has a connotation of "claiming". This is not the case here. By making an assumption, we are not claiming anything.

When interpreting a derivation tree, we must always consider the open assumptions. We must say: under the assumptions . . . , we derived . . . .

It is thus unproblematic to "make" assumptions.

Propositional Variables

In mathematics, logic and computer science, there are various notions of variable. In propositional logic, a variable stands for a proposition, i.e., a variable can be interpreted as $True$ or $False$.

This will be different in logics that we will learn about later.

What is a Formula?

In logic, the word "formula" has a specific meaning. Formulae are a syntactic category, namely the expressions that stand for a statement. So formulas are syntactic expressions that are interpreted (on the semantic level) as $True$ or $False$.

We will later learn about another syntactic category, that of terms.

Associativity and Precedences

To save brackets, we use standard associativity and precedences. All binary connectives are right-associative:

$$A \circ B \circ C \equiv A \circ (B \circ C)$$

The precedences are $\neg$ before $\wedge$ before $\vee$ before $\rightarrow$. So for example

$$A \rightarrow B \wedge \neg C \vee D \equiv A \rightarrow ((B \wedge (\neg C)) \vee D)$$

Why Smallest Set?

The language of propositional logic is a set of formulae, defined by induction. Note the following points about the definition, which are important characteristics of any inductive definition:

- By the second item in the definition, L_P is non-empty (also, one would usually have that V is non-empty, since otherwise L_P is not very interesting);
- L_P is required to be the smallest set meeting the above conditions. Otherwise, anything (a number, a dog, the pope) could be a propositional formula.
- All conditions (or rules) defining L_P have the form: if ψ_1 and . . . and ψ_n are in L_P, then some formula built from ψ_1 and . . . and ψ_n is in L_P.

It is crucial that no negation is involved here. If for example, there was a rule stating: if A is in L_P then A is not in L_P, then there could be no L_P fulfilling such a rule.

More detail on inductive definitions can be found in an article by Aczel [Acz77].

Introduction and Elimination

It is typical that the basic rules of a proof system can be classified as introduction or elimination rules for a particular connective.

This classification provides obvious names for the rules and may guide the search for proofs.

The rules for conjunction are pronounced and-introduction, and-elimination-left, and and-elimination-right.

Apart from the basic rules, we will later see that there are also derived rules.

Validity Revisited

A rule is valid if for any assignment under which the assumptions of the formula are true, the conclusion is true as well.

The notation $\mathcal{A} \models A \wedge B$ stands for: under the assignment $\mathcal{A}$, the formula $A \wedge B$ is interpreted as $True$.

This is consistent with the

earlier intuitive explanation of validity of a formula. Details can be found in any textbook on logic [vD80].

Note that while the notation $\mathcal{A} \models \ldots$ will be used again later, there $\mathcal{A}$ will not stand for an assignment, but rather for a construct having an assignment as one constituent. This is because we will generalize, and in the new setting we need something more complex than just an assignment. But in spirit $\mathcal{A} \models \ldots$ will still mean the same thing.

Schematic Rules

The letters A and B in the rules are not propositional variables. Instead, they can stand for arbitrary propositional formulas. One can also say that A and B are metavariables, i.e., they are variables of the proof system as opposed to object variables, i.e., variables of the language that we reason about (here: propositional logic).

When a rule is applied, the metavariables of it must be replaced with actual formulae. We say that a rule is being instantiated.

We will see more about the use of metavariables later.

Can we Prove Anything . . . ?

All three rules have a non-empty sequence of assumptions. Thus to build a tree using these rules, we must first make some assumptions.

None of the rules involves discharging an assumption.

We have said earlier that a proof is a derivation with no open assumptions.

Consequently, the answer is no.

Object vs. Meta

In these examples, you may regard A, B, C as propositional variables. On the other hand, the proofs are schematic, i.e., they go through for any formula replacing A, B, and C.

So you Find this Strange!

When we make the assumption P, we obtain a forest consisting of one tree. In this tree, P is at the same time a leaf and the root. Thus the tree P is a degenerate example of the schema

$$\begin{array}{l}[A]\\ \vdots\\ B\end{array}$$

where both A and B are replaced with P.

Therefore we may apply rule $\rightarrow$-*I*, similarly as in our abstract example.

$$A \to B \to A$$

The rule(s):

$$\frac{\begin{array}{c}[A]\\ \vdots \\ B\end{array}}{A \to B}\ {\to}\text{-}I$$

The proof:

$$\frac{\dfrac{[A]^1}{B \to A}\ {\to}\text{-}I}{A \to B \to A}\ {\to}\text{-}I^1$$

$(A \wedge (B \wedge C)) \rightarrow (A \wedge C)$

The rules:

$$\frac{A \quad B}{A \wedge B}\ \wedge\text{-}I$$

$$\frac{A \wedge B}{A}\ \wedge\text{-}EL$$

$$\frac{A \wedge B}{B}\ \wedge\text{-}ER$$

$$\frac{\begin{array}{c}[A]\\ \vdots \\ B\end{array}}{A \rightarrow B}\ \rightarrow\text{-}I$$

The proof:

$$\dfrac{\dfrac{\dfrac{[A \wedge (B \wedge C)]^1}{A}\ \wedge\text{-}EL \qquad \dfrac{\dfrac{[A \wedge (B \wedge C)]^1}{B \wedge C}\ \wedge\text{-}ER}{C}\ \wedge\text{-}ER}{A \wedge C}\ \wedge\text{-}I}{(A \wedge (B \wedge C)) \rightarrow (A \wedge C)}\ \rightarrow\text{-}I^1$$

$$(A \to B \to C) \to (A \to B) \to A \to C$$

The rules:

$$\dfrac{\begin{array}{c}[A]\\ \vdots \\ B\end{array}}{A \to B}\ \to\text{-}I$$

$$\dfrac{A \to B \quad A}{B}\ \to\text{-}E$$

The proof:

$$\dfrac{\dfrac{\dfrac{[(A \to B \to C)]^1 \quad [A]^3}{B \to C}\ \to\text{-}E \quad \dfrac{[(A \to B)]^2 \quad [A]^3}{B}\ \to\text{-}E}{C}\ \to\text{-}E}{A \to C}\ \to\text{-}I^3$$

$$\dfrac{(A \to B) \to A \to C}{(A \to B \to C) \to (A \to B) \to A \to C}\ \to\text{-}I^2$$

$$\to\text{-}I^1$$

Falsity

The symbol $\bot$ stands for "false".

No Introduction Rule for $\bot$

The symbol $\bot$ stands for "false".

It should be intuitively clear that since the purpose of a proof system is to derive true formulae, there is no introduction rule for falsity. One may wonder: what is the role of $\bot$ then? We will see this soon. The main role is linked to negation.

Connectives

The connectives are called conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$) and negation ($\neg$).

The connectives $\wedge, \vee, \rightarrow$ are binary since they connect two formulas, the connective $\neg$ is unary (most of the time, one only uses the word connective for binary connective).

Negation

"Officially", negation does not exist in our language and proof system. Negation is only used as a shorthand, or syntactic sugar, for reasons of convenience. In paper-and-pencil proofs, we are allowed to erase any occurrence of $\neg P$ and replace it with $P \rightarrow \bot$, or vice versa, at any time. However, we shall see that when proofs are automated, this process must be made explicit.

The Rules for $\neg$

The rule

$$\frac{\neg A \quad A}{\bot}\ \neg\text{-}E$$

is simply an instance of $\rightarrow$-*E* (since $\neg A$ is shorthand for $A \rightarrow \bot$).

Likewise, the rule

$$\frac{\begin{array}{c}[A]\\ \vdots \\ \bot\end{array}}{\neg A}\ \neg\text{-}I$$

is simply an instance of $\rightarrow$-*I*. Therefore, we will not introduce these as special rules. But there is a special rule $\neg$-*E*.

The Rule $\neg$-E

For negation, it is common to have a rule

$$\frac{\neg A \quad A}{B}\ \neg\text{-}E$$

We have seen how this rule can be derived. The concept of deriving rules will be explained more systematically later.

This rule is also called ex falsum quod libet (from the false whatever you like).

Peirce's Law Valid?

Yes, simply check the truth table:

A	B	$((A \to B) \to A) \to A$
$True$	$True$	$True$
$True$	$False$	$True$
$False$	$True$	$True$
$False$	$False$	$True$

Peirce's Law Provable?

In the proof system given so far, this is not provable. To prove that it is not provable requires an analysis of so-called normal forms of proofs. However, we do not do this here.

Intuitionistic versus Classical Logic

The proof system we have given so far is a proof system for intuitionistic logic. The main point about intuitionistic logic is that one cannot claim that every statement is either true or false, but rather, evidence must be given for every statement.

In classical reasoning, the law of the excluded middle holds.

One also says that proofs in intuitionistic logic are constructive whereas proofs in classical logic are not necessarily constructive.

The difference between intuitionistic and classical logic has been the topic of a fundamental discourse in the literature on logic [PM68]. Often proofs contain case distinctions, assuming that for any statement ψ, either ψ or $\neg\psi$ holds. This reasoning is classical; it does not apply in intuitionistic logic.

Axiom of the Excluded Middle

$A \vee \neg A$ is called axiom of the excluded middle.

Reductio ad absurdum

The rule

$$\dfrac{\begin{array}{c}[\neg A]\\ \vdots\\ \bot\end{array}}{A}\ RAA$$

is called reduction ad absurdum.

The Classical Rule

The rule

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

corresponds to a formulation in Isabelle.

Sequent Notation

A logical statement like $A \to (B \to C), A, B \vdash C$ is called a derivability judgement. We explained it earlier as simply asserting the fact that there exists a derivation tree with C at its root and open assumptions $A \to (B \to C), A, B$.

However, it is also possible to make such judgements the central objects of the deductive system, i.e., have rules involving such objects.

The notation $\Gamma \vdash A$ is called sequent notation. However, this should not be confused with the sequent calculus (we will consider it later). The sequent calculus is based on sequents, which are syntactic entities of the form $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$, where the $A_1, \ldots, A_n, B_1, \ldots, B_m$ are all formulae. You see that this definition is more general than the derivability judgements we consider here.

What we are about to present is a kind of hybrid between natural

deduction and the sequent calculus, which we might call natural deduction using a sequent notation.

Axioms vs. Rules

An axiom is a rule without premises. We call a rule with premises proper.

Note that the natural deduction rules for propositional logic contain no axioms. In the sequent style formalization, having the assumption rule (axiom) is essential for being able to prove anything, but in the natural deduction style we learned first, we can construct proofs without having any axioms.

Assumptions

The special rule for assumptions takes the role in this sequent style notation that the process of making and discharging assumptions had in natural deduction based on trees.

It is not so obvious that the two ways of writing proofs are equivalent, but we shall become familiar with this in the exercises by doing proofs on paper as well as in Isabelle.

Weakening

The rule *weaken* is

$$\frac{\Gamma \vdash B}{A, \Gamma \vdash B}\ \textit{weaken}$$

Intuitively, the soundness of rule *weaken* should be clear: having an additional assumption in the context cannot hurt since there is no proof rule that requires the absence of some assumption.

We will see an application of that rule later.

Deriving $\wedge$-E

As an example, consider

$$\frac{A, B, \Gamma \vdash C \quad \Gamma \vdash A \wedge B}{\Gamma \vdash C}\ \wedge\text{-}E$$

This rule can be derived as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{A, B, \Gamma \vdash C}{A, \Gamma \vdash B \to C}\ \to\text{-}I}{\Gamma \vdash A \to B \to C}\ \to\text{-}I \quad \dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\text{-}EL}{\Gamma \vdash B \to C}\ \to\text{-}E \quad \dfrac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\text{-}ER}{\Gamma \vdash C}\ \to\text{-}E$$

Which Rule to Choose?

In general, statements about which rule to choose when building a proof are heuristics, i.e., they are not guaranteed to work. Building a proof means searching for a proof. However, there are situations where the choice is clear. E.g., when the topmost connective of a formula is $\rightarrow$, then $\rightarrow$-*I* is usually the right rule to apply.

The question will be addressed more systematically later.

Goals to Axioms

As you saw in our animation, we worked from the root of the tree to the leaves.

Working with Assumptions

One aspect you might have noted in the proof is that the steps at the top, where $\wedge$-*EL* and $\wedge$-*ER* were used, required non-obvious choices, and those choices were based on the assumptions in the current derivability judgement.

In Isabelle, we will apply other rules and proof techniques that allow us to manipulate assumptions explicitly. These techniques make the process of finding a proof more deterministic.

References

[Acz77] Peter Aczel. *Handbook of Mathematical Logic*, chapter An Introduction to Inductive Definitions, pages 739–782. North-Holland, 1977.

[PM68] Dag Prawitz and Per-Erik Malmnäs. A survey of some connections between classical, intuitionistic and minimal logic. In A. Schmidt and H. Schütte, editors, *Contributions to Mathematical Logic*, pages 215–229. North-Holland, 1968.

[vD80] Dirk van Dalen. *Logic and Structure*. Springer-Verlag, 1980. An introductory textbook on logic.