Mathematical and Logical Foundations of Computer Science

Lecture 1 - Introduction

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

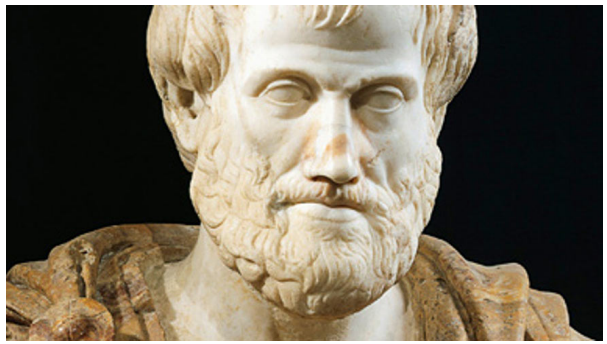
University of Birmingham

Today

- What is logic?
- Why study logic?
- This module
- Basic concepts

What is logic?

An old science developed in many cultures, most notably in Greece by **Aristotle** in 350 B.C.



In his *Organon*, Aristotle provided rules to conduct logical reasoning, and derive correct statements.

As such, logic provides reasoning techniques that enable deriving knowledge in a systematic way.

In the 19th century, mathematicians such as **Boole** and **Frege** further revolutionized the field of logic, and their contributions led to modern mathematical logic, which we will study in this module.

What is logic?

What sort of reasoning can logic help us with?

A puzzle:

- There are 4 cards, each with a letter on one side and a number on the other
- Rule: "every card with a vowel has an even number on the other side"

$$\boxed{Q} \qquad \boxed{E} \qquad \boxed{6} \qquad \boxed{3}$$

- Which card(s) must you turn over in order to check this rule?
- E and 3
- Why do we not need to turn over Q and 6?

What is logic?

Another puzzle:

- There are 4 cards, each with name of a drink on one side and an age on the other
- Rule: "if the age is under 18, then the drink on the other side of the card is non-alcoholic"

Juice	35	Beer	16

- Which card(s) must you turn over in order to check this rule?
- Beer and 16
- Why do we not need to turn over Juice and 35?

What is logic?

Reasoning techniques for deriving knowledge

An informal argument:

- All men are mortal
- Socrates is a man
- Therefore, Socrates is mortal

In what is called Predicate Logic:

- $\forall x.\text{Man}(x) \rightarrow \text{Mortal}(x)$
- Socrates is a man, i.e., $\text{Man}(Socrates)$
- Hence, $\text{Mortal}(\text{Socrates})$

What is logic?

Logic is about formalising knowledge and reasoning

in a precise, unambiguous, rigorous way

Today

- What is logic?
- **Why study logic?**
- This module
- Basic concepts

Why study logic?

- Logic is fundamental in computer science
  - also in philosophy, mathematics, psychology, ...
- Logic in computer science:
  - understanding/modelling, formalisation/rigour, correctness/proof, computation/automation, ..
- Logic plays a key role in many areas of computer science:
  - correctness and formal verification
    - self-driving cars
  - theory of computation
    - what can be computed? how fast?
  - SAT solvers
    - solving "every hard" problem
  - AI, databases, etc ...

Today plan

- What is logic?
- Why study logic?
- **This module**
- Basic concepts

Syllabus of the logic part of this module

- Propositional logic
  - syntax
  - proofs (natural deduction)
  - semantics, truth tables
  - satisfiability
- First order logic (predicate calculus)
  - syntax
  - proofs (natural deduction)
  - semantics

Learning outcomes

- Understand and apply algorithms for key problems in logic such as satisfiability.
- Write formal proofs for propositional and predicate logic
- Apply mathematical and logical techniques to solve a problem within a computer science setting

Organization

- lectures: optional pre-recorded lectures & on-campus lectures
- tutorials
- assessments
- office hours
- resources:
  - Canvas page
  - Further reading:
    - http://leanprover.github.io/logic_and_proof/index.html
    - https://www.paultaylor.eu/stable/prot.pdf
    - https://research.tue.nl/en/publications/logical-reasoning-a-first-course

Today

- What is logic?
- Why study logic?
- This module
- **Basic concepts**

Basic concepts: Propositions

A **proposition** is a sentence which states a fact

i.e. a statement that can (in principle) be true or false

Example sentences:

- Birmingham is north of London
  proposition, and true
- $8 \times 7 = 42$
  proposition, and false
- Please mind the gap
  not a proposition!
- Every even natural number > 2 is the sum of two primes
  proposition
  Goldbach Conjecture: unknown whether it is true or false!
- Is black the opposite of white?
  not a proposition!

Basic concepts: Arguments

An **argument** is a list of propositions

- the last of which is called the **conclusion**
- and the others are called **premises**

Example: 2 premises and 1 conclusion

1. Premise 1: **If** there is smoke, **then** there is a fire
2. Premise 2: There is no fire
3. Conclusion: **Therefore**, there is no smoke

Basic concepts: Validity of Arguments

An argument is **valid** if (and only if), whenever the premises are true, then so is the conclusion

Is the argument from the previous slide valid?

1. Premise 1: **If** there is smoke, **then** there is a fire
2. Premise 2: There is no fire
3. Conclusion: **Therefore**, there is no smoke

Yes, it is valid!

If an argument is not valid, then it is invalid

Basic concepts: Example Arguments

Is this valid?

1. If John is at home, then his television is on.
2. His television is not on.
3. Therefore, John is not at home.

Valid

Is this valid?

1. You can eat a burger or pasta.
2. You ate a burger.
3. Therefore, you did not eat pasta.

Invalid

Why not both?

OR in English is usually exclusive

Basic concepts: More Example Arguments

Is this valid? Invalid

1. If the control software crashes, then the car's brakes will fail.
2. The car's brakes failed.
3. Therefore, the control software crashed.

Is this valid? Invalid (for the same reason as above)

1. If (2+2=5) then (3+3=6).
2. 3+3=6.
3. Therefore, 2+2=5.

More generally (with **symbols**) this argument is not valid (we saw 2 counterexamples):

1. If P then Q.
2. Q.
3. Therefore, P.

Basic concepts: More Example Arguments

Is this valid? Invalid

1. If the control software crashes, then the car's brakes will fail.
2. The control software did not crash.
3. Therefore, the car's brakes did not fail.

Is this valid? Invalid (for the same reason as above)

1. If (2+2=5) then (3+3=6).
2. 2+2 is not 5.
3. Therefore, 3+3 is not 6.

More generally (with **symbols**) this argument is not valid (we saw 2 counterexamples):

1. If P then Q.
2. $\neg P$.
3. Therefore, $\neg Q$.

Conclusion

What did we cover today?

- what and why logic
- organization of the logic part of the module
- basic logic concepts

Next time?

- Symbolic logic

Mathematical and Logical Foundations of Computer Science

Lecture 2 - Symbolic Logic

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- **Symbolic logic**
- Propositional logic
- Predicate logic

Today

We will introduce some useful concepts to deal with logical systems. Some of them will make more sense as we experience them during the course of this module.

- Symbolic logic
- Grammars
- (Meta)variables
- Axiom schemata
- Substitution

Symbolic Logics

Symbolic logics are **formal languages** that allow conducting logical reasoning through the **manipulation of symbols**.

> *"Symbolic logic is the development of the most general principles of rational procedure, in ideographic symbols, and in a form which exhibits the connection of these principles one with another." (Irving Lewis in A Survey of Symbolic Logic)*

Pioneered for example by Leibniz, Boole, Frege, etc.

For example:

- **Propositional logic**
- **Predicate logic**
- Higher-order logic

Grammars - BNFs

Two important aspects of a language are:

- its **syntax** describing the well-formed sequences of symbols denoting objects of the language;
- and its **semantics** assigning meaning to those symbols.

This lecture focuses on syntax.

The syntax of a language is defined through a **grammar**.

In particular, the language of a symbolic logic is defined by a grammar that allows deriving formulas from collections of symbols (we will see an example in a few slides).

Grammars - BNFs

The grammar of such a language is often defined using a **Backus Naur Form** (BNF). BNFs allow defining **context-free grammars** (i.e., where production rules are context independent). They are collections of **rules** of the form:

$$lhs ::= rhs_1 \mid \cdots \mid rhs_n$$

Meaning: this rule means that the left-hand-side lhs (a non-terminal symbol) can expand to any of the forms rhs_1 to rhs_n on the right-hand-side.

Each rhs_i is a sequence of non-terminal and terminal symbols.

The **arity** of a terminal symbol is the number of arguments it takes.

The **Fixity** of a terminal symbol is the place where it occurs w.r.t. its arguments: **infix** if it occurs in-between its arguments, **prefix** if it occurs before, and **postfix** if it occurs after.

Grammars - BNF example

Example of a BNF for (some) arithmetic expressions:

$$exp ::= num \mid exp + exp \mid exp \times exp$$

where a numeral num is a sequence of digits. Here exp is a non-terminal symbol and $+$, $\times$, 0, 1, etc., are terminal symbols.

Arity & fixity:

- 0, 1, etc. are nullary (arity 0) operators (they are called **constants**).
- $+$ and $\times$ are binary (arity 2) infix operators

Derivations:

$$exp \mapsto exp + exp \mapsto 1 + exp \mapsto 1 + 2$$
$$exp \mapsto exp \times exp \mapsto exp \times 0 \mapsto 2 \times 0$$

How to extend this language to allow for conditional expressions?

$$
\begin{array}{lcl}
exp & ::= & num \mid exp + exp \mid exp \times exp \mid \texttt{if}\ b\ \texttt{then}\ exp\ \texttt{else}\ exp \\
b & ::= & \texttt{true} \mid \texttt{false} \mid b\ \&\ b \mid b \parallel b
\end{array}
$$

Fixity: all the above operators are infix.

Grammars - BNF example

Example of a BNF for propositional logic formulas:

$$P ::= a \mid P \to P \mid P \vee P \mid P \wedge P \mid \neg P$$

where a ranges over a set of atomic propositions (e.g., *"it is raining"*, or *"it is sunny"*). Here P is a non-terminal symbol and $\wedge$, $\vee$, $\to$, and $\neg$, as well as the atomic propositions, are terminal symbols.

Arity & Fixity: $\wedge$, $\vee$, $\to$ are binary infix operators, $\neg$ is a unary (arity 1) prefix operator

Example: let s stand for "it is sunny", and r for "it is rainy"

Derivation:

$$P \mapsto P \vee P \mapsto r \vee P \mapsto r \vee \neg P \mapsto r \vee \neg s$$

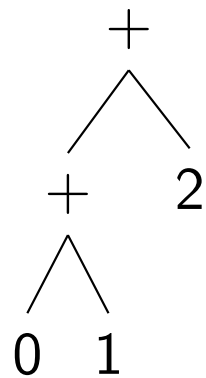
Grammars - abstract syntax trees

An expression derived from a BNF grammar can then be seen as a tree, called an **abstract syntax tree**.

For example, given the grammar:

$$exp ::= num \mid exp + exp \mid exp \times exp$$

an abstract syntax tree corresponding to $0 + 1 + 2$ is:



Grammars - associativity

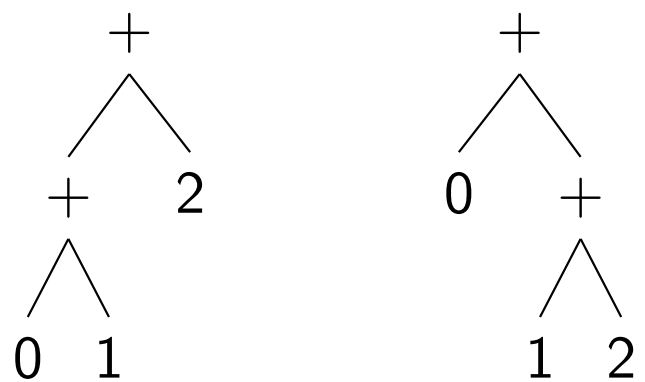
Note the **ambiguity** in our example: $0 + 1 + 2$.
Does it stand for $(0 + 1) + 2$ or $0 + (1 + 2)$?

We need to define the **associativity** of the terminal symbols to avoid ambiguities.

- left associativity: $(0 + 1) + 2$
- right associativity: $0 + (1 + 2)$

We will consider the first but we will sometimes use parentheses to avoid ambiguities.

Those have different abstract syntax trees:



Grammars - precedence

What about: $0 + 1 \times 2$? This is again ambiguous.

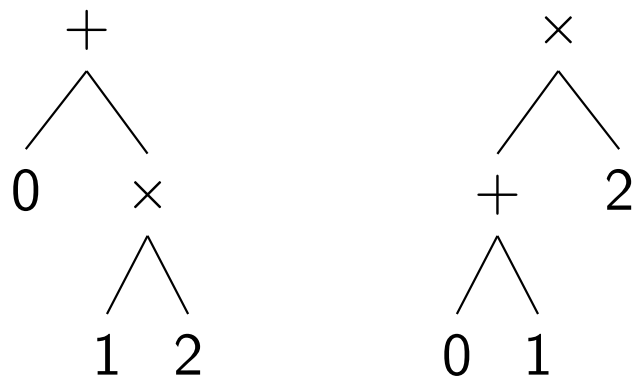
Does it stand for $(0 + 1) \times 2$ or $0 + (1 \times 2)$?

We need to define the **precedence** of the terminal symbols to avoid ambiguities.

- $\times$ has higher precedence: $0 + (1 \times 2)$
- $+$ has higher precedence: $(0 + 1) \times 2$

We will consider the first.

Those have different abstract syntax trees:



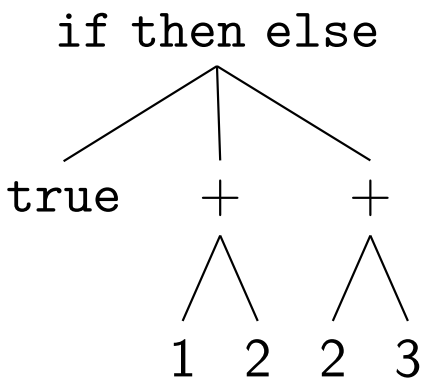
Grammars - example

What is the abstract syntax tree for?

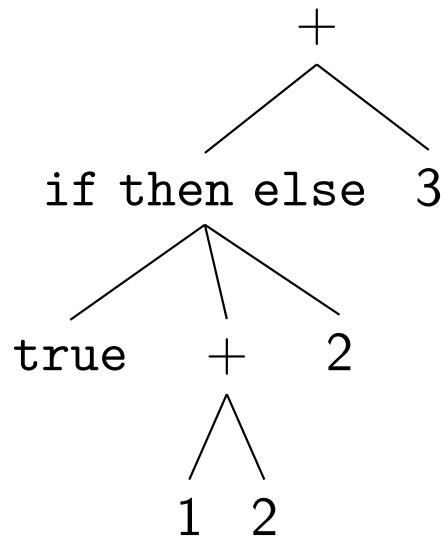
$$\texttt{if true then } 1 + 2 \texttt{ else } 2 + 3$$

Again this is ambiguous. Without knowing which operator has precedence over the other, it could be either of the two:

$\texttt{if true then } (1 + 2) \texttt{ else } (2 + 3)$



$(\texttt{if true then } 1 + 2 \texttt{ else } 2) + 3$



Grammars - associativity, precedence, parentheses

To avoid ambiguities:

- define the associativity of symbols
- define the precedence between symbols
- use parentheses to avoid ambiguities or for clarity

Parentheses are sometimes necessary:

- using left associativity $0 + 1 + 2$ stands for $(0 + 1) + 2$
- we need parentheses to express $0 + (1 + 2)$

Grammars - example

Given the grammar:

$$P ::= a \mid P \to P \mid P \vee P \mid P \wedge P \mid \neg P$$

what is the abstract syntax tree for $(\neg P) \wedge (Q \vee R)$?

(Meta)variables

Some of these concepts will start making more sense when we come to experience them during the course of this module

We sometimes want to write down expressions/formulas such as $exp + exp$ or $P \rightarrow P$, where exp and P are non-terminals.

In that case exp and P act as **variables** that can range over **all possible** expressions/formulas.

Such variables are typically called, **metavariables** or **schematic variables**, and act as placeholders for any element derivable from a given grammar rule.

For example, we might write $P \rightarrow P$ to mean that P implies P whatever the proposition P is: "it is rainy" $\rightarrow$ "it is rainy" is true; "it is sunny" $\rightarrow$ "it is sunny" is true; etc.

(Meta)variables

Notation. Given the grammar:

$$exp ::= num \mid exp + exp \mid exp \times exp$$

one typically allows exp, exp_0, exp_1, ..., exp', exp'', ..., as variables ranging over all possible arithmetic expressions derivable using the above rule.

Technical details:

- The expressions of the language captured by the above grammar, has all the ones that cannot be derived further, i.e., that do not contain **non-terminal** symbols.
- $exp + exp$ is not part of this language but is useful to capture a **collection** of expressions.
- Why is it called a "metavariable"? A metavariable is a variable within the language, called the **metatheory**, used to describe and study a theory at hand.

(Meta)variables

For example, let us consider the following grammar:

$$\begin{array}{lll} exp & ::= & num \mid exp + exp \mid exp \times exp \\ eq & ::= & exp = exp \end{array}$$

where equalities are used to state that two expressions are equal.

This defines the syntax of a simple symbolic logic to reason about arithmetic expressions.

We use this language to state laws of arithmetic by describing what equalities hold using variables that act as placeholders for any possible expressions.

Some equalities are assumed to hold in our simple logic through **axioms**, such as $0 + 0 = 0$, $1 + 0 = 1$, $2 + 0 = 2$, etc.

Axiom schemata

For example, as part of a "number theory" one may want to assume that the following equality holds:

$$exp + 0 = exp$$

A standard law of arithmetic that states: 0 is an additive identity.

It stands for an infinite number of axioms, which can be obtained by **instantiating** the variable exp with any arithmetic expression. This is called an **axiom schemata**.

For example, the following equality is such an instance:

$$1 + 0 = 1$$

Other examples of instances?

- $2 + 0 = 2$
- $(1 + 2) + 0 = 1 + 2$
- etc.

Axiom schemata

As another example, take again propositional logic, whose syntax is:

$$P ::= a \mid P \to P \mid P \vee P \mid P \wedge P \mid \neg P$$

Variables are useful to state axioms of the logic.

For example, we can state:

$$(P \wedge Q) \to P$$

using the variables P and Q.

By replacing P by "2 is prime" and Q by "2 is even", we can obtain the following instance of this formula:

$$(2 \text{ is prime} \wedge 2 \text{ is even}) \to 2 \text{ is prime}$$

Substitution

How do we obtain the equality:

$$1 + 0 = 1$$

from the axiom schema:

$$exp + 0 = exp$$

This is done by instantiating the schema, i.e., by substituting the variable exp with an arithmetic expression. For example here, we substituted exp with 1.

A **substitution** is a mapping (e.g., a key/value map), that maps metavariables to arithmetic expressions.
The **substitution operation** is the operation that replaces all occurrences of the keys by the corresponding values (the 1st key/value pair is considered if a key occurs more than once).

Substitution

We write $k_0 \backslash v_0, \ldots, k_n \backslash v_n$ for the substitution that maps k_i to v_i for $i \in \{0, \ldots, n\}$.

For example:

- The substitution $exp \backslash 1$ maps exp to 1.
- $exp_1 \backslash 0, exp_2 \backslash 1$ maps exp_1 to 0 and exp_2 to 1.
- $exp_1 \backslash 0, exp_2 \backslash 1, exp_1 \backslash 1$ also maps exp_1 to 0 and exp_2 to 1.

The substitution operation, written $eq[s]$, takes an equality eq and a substitution s, and replaces all occurrences of the keys of s by the corresponding values in eq.

For example: $(exp + 0 = exp)[exp \backslash 1]$ returns $1 + 0 = 1$.

Substitution - formally

Formally, the substitution operation is defined recursively on the syntactic forms they are applied to.

For example, the substitution operation computes as follows on arithmetic expressions:

$$
\begin{array}{lcl}
num[s] & = & num \\
(exp_1 + exp_2)[s] & = & exp_1[s] + exp_2[s] \\
(exp_1 \times exp_2)[s] & = & exp_1[s] \times exp_2[s] \\
(exp_1 = exp_2)[s] & = & exp_1[s] = exp_2[s]
\end{array}
$$

and as we allow variables in expressions:

$$
\begin{array}{lcl}
v[s] & = & v, \text{ if } v \text{ is not a key of } s \\
v[s] & = & e, \text{ if } s \text{ maps } v \text{ to } e
\end{array}
$$

Substitution - further examples

Consider the following commutativity schema:

$$exp_1 + exp_2 = exp_2 + exp_1$$

What does $(exp_1 + exp_2 = exp_2 + exp_1)[exp \backslash 1]$ return?
$exp_1 + exp_2 = exp_2 + exp_1$

What does $(exp_1 + exp_2 = exp_2 + exp_1)[exp_1 \backslash 1]$ return?
$1 + exp_2 = exp_2 + 1$

What does $(exp_1 + exp_2 = exp_2 + exp_1)[exp_1 \backslash 1, exp_2 \backslash 2]$ return?
$1 + 2 = 2 + 1$

Conclusion

What did we cover today?

- A formal language such as a symbolic logic has a syntax captured by a grammar (e.g., a BNF).
- (Meta)variables are used to capture collections of axioms (as axiom schemata) of symbolic logics.
- Substitution is used to derive instances of axiom schemata.

Next time?

- Propositional logic - Syntax

Mathematical and Logical Foundations of Computer Science

Lecture 3 - Propositional Logic (Syntax)

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- Symbolic logic
- **Propositional logic**
- Predicate logic

Today

- Propositional logic
- Syntax of the language
- Informal semantics
- Simple proofs

Propositions - informal presentation

Propositional logic is a **symbolic logic** to reason about logical statements called **propositions** that can (in principle) be true or false.

Propositions are built by combining atomic propositions using the and, or, not, and implies logical connectives.

Are these examples of propositions?

- Birmingham is north of London Yes
- Is Birmingham north of London? No
- $8 \times 7 = 42$ Yes
- Every even natural number > 2 is the sum of two primes Yes
- Please mind the gap No

Arguments - informal presentation

Let an **argument** be a list of propositions, the last of which is called the conclusion and the others are called premises.

An argument is **valid** if and only if (iff) whenever the premises are true, then so is the conclusion

In propositional logic true propositions can be derived from other true propositions through the use of derivation rules.

For example:

1. If John is at home, then his television is on.
2. His television is not on.
3. Therefore, John is not at home.

Valid? Yes

Arguments - informal presentation

More examples:

1. You can eat a burger or pasta.
2. You ate a burger.
3. Therefore, you did not eat pasta.

Valid? No Because you could eat both. In propositional logic, or is not exclusive as it is often the case in English.

1. If the control software crashes, then the car's brakes will fail.
2. The car's brakes failed.
3. Therefore, the control software crashed.

valid? No The car's brakes could have failed for another reason.

1. If the control software crashes, then the car's brakes will fail.
2. The control software did not crash.
3. Therefore, the car's brakes did not fail.

valid? No The car's brakes could have failed for another reason.

Formalizing logical statements and arguments

We want to formalise such statements and arguments.

We will take a **symbolic approach**.

It will allow us proving the (in)validity of statements generally.

Advantages of formal symbolic language over natural languages are:

- **unambiguous**
- **more concise**

Propositional Logic

Symbols:

- atomic propositions (true/false atomic statements)
- combined using logical connectives

Atomic propositions (atoms)

- propositions that cannot be broken into smaller parts
- Let $p, q, r, \ldots$ be atomic propositions
- two special atoms: $\top$ stands for True, $\bot$ stands for False

Logical Connectives

- conjunction: $\wedge$ (and)
- disjunction: $\vee$ (or)
- implication: $\rightarrow$ (if .... then / implies)
- negation: $\neg$ (not) — can be defined using $\rightarrow$ and $\bot$

Propositions - informal examples

What are the atomic propositions and connectives?

- The car's brakes failed
  an atomic proposition
- The control software crashed and the car's brakes failed
  a conjunction of 2 atomic propositions
- If the control software crashes, then the car's brakes will fail
  an implication connecting 2 atomic propositions

Propositional logic

The syntax of propositional logic formulas (called propositions) is defined by the following grammar:

$$P ::= a \mid P \wedge P \mid P \vee P \mid P \rightarrow P \mid \neg P$$

where a ranges over **atomic propositions**.

Atomic propositions are formulas.

If P and Q are formulas, then

- $P \wedge Q$ is a formula
- $P \vee Q$ is a formula
- $P \rightarrow Q$ is a formula
- $\neg P$ is a formula

Those are called **compound formulas**.

Example of a compound formula: $\neg p \wedge q \wedge q \wedge \neg r$.

Connectives - informal semantics

Conjunction: $P \wedge Q$, i.e., P and Q

- true if both individual propositions P and Q are true

Disjunction: $P \vee Q$, i.e., P or Q

- true if one or both individual propositions P and Q are true
- also sometimes called "inclusive or"
- Note: Or in English is often an "exclusive or" (i.e. where one or the other is true, but not both)
- e.g., "Your mark will be pass or fail"
- but logical disjunction is always defined as above

Connectives - informal semantics

Implication: $P \rightarrow Q$, i.e., P implies Q

- means: if P is true then Q must be true too
- if P is false, we can conclude nothing about Q
- P is the antecedent, Q is the consequent

Negation: $\neg P$, i.e., not P

- it can be defined as $P \rightarrow \bot$
- if P is true, then $\bot$ (False)
- true iff P is false

Avoiding ambiguities

$P \wedge Q \vee R$

- Is this a well-formed formula? Yes
- what does it mean?
- $(P \wedge Q) \vee R$?
- $P \wedge (Q \vee R)$?
- We don't know.

In general use parentheses to avoid ambiguities.

Use either $(P \wedge Q) \vee R$ or $P \wedge (Q \vee R)$.

Precedence: in decreasing order of precedence $\neg, \wedge, \vee, \rightarrow$.

For example, $\neg P \vee Q$ means $(\neg P) \vee Q$.

Associativity: all operators are right associative

For example, $P \vee Q \vee R$ means $P \vee (Q \vee R)$.

However use parentheses around compound formulas for clarity.

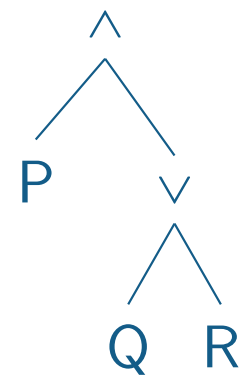
Parse Trees

Parentheses help clarify how formulas are derived given the propositional logic's grammar:

$$P ::= a \mid P \wedge P \mid P \vee P \mid P \rightarrow P \mid \neg P$$

The parse tree for $(P \wedge Q) \vee R$ is:

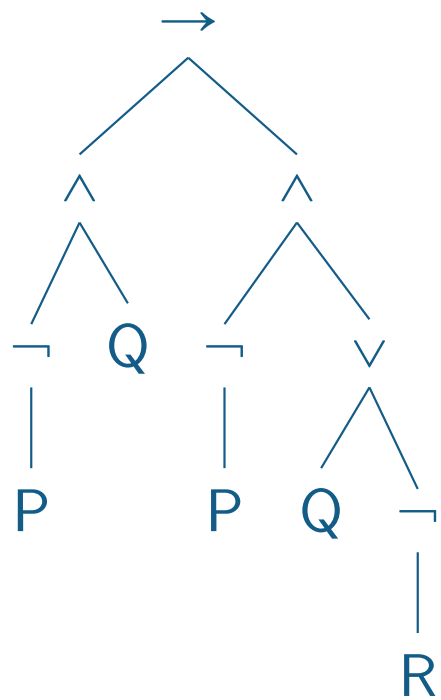
while the parse tree for $P \wedge (Q \vee R)$ is:



Leaves are atomic propositions and the other nodes are connectives.

Parse Trees

What it the parse tree for: $(\neg P \wedge Q) \rightarrow (\neg P \wedge (Q \vee \neg R))$?



Scope and Main connective

Scope of a connective

- The connective itself, plus what it connects
- That is, the sub-tree of the parse tree rooted at the connective
- The scope of $\wedge$ in $(P \wedge Q) \vee R$ is $P \wedge Q$

Main connective of a formula

- The connective whose scope is the whole formula
- That is, the root node of the parse tree
- The main connective of $(P \wedge Q) \vee R$ is $\vee$

Arguments in Propositional Logic

Example argument

1. If John is at home, then his television is on
2. His television is not on
3. Therefore, John is not at home

Identify atomic propositions:

- p =“John is at home”
- q =“John’s television is on”

How do we write this argument in propositional logic?

- Premise 1: $p \rightarrow q$
- Premise 2: $\neg q$
- Conclusion: $\neg p$

Arguments in Propositional Logic

Example argument

- Premise 1: $p \rightarrow q$
- Premise 2: $\neg q$
- Conclusion: $\neg p$

Notation: written as a **sequent**

- $p \rightarrow q, \neg q \vdash \neg p$
- i.e., set of premises separated by commas, then a **turnstile** followed by the conclusion.
- Recall that premises and conclusions are both formulas.
- A sequent is **valid** if the argument has been proven, i.e., if the conclusion is true assuming that the premises are true.

Proofs in Propositional Logic

For **formal proofs** we need two things

1. A **formal** language
   - for representing propositions, arguments
   - here we are using propositional logic
2. A **proof** theory
   - to prove ("infer", "deduce") whether an argument is valid
   - we'll see several different approaches in this module
   - for now (next few lectures): Natural Deduction

Natural Deduction

Natural Deduction

- "natural" style of constructing a proof (like a human would)
- syntactic (rather than semantic) proof method
- proofs are constructed by applying inference rules

Basic idea to prove an argument is valid:

- start with the premises (we can assume these are true)
- repeatedly apply inference rules (which "preserve truth")
- until we have inferred the conclusion

What are inference rules?

Inference rules are the tools we have/are allowed to use

Example of an inference rule:

$$\frac{A \quad B}{A \wedge B} \; [\wedge I]$$

Notation

- Premise(s) at the top
- Conclusion at the bottom
- Name of the inference rule on the right

Some simple inference rules

And-introduction:

$$\frac{A \quad B}{A \wedge B} \; [\wedge I]$$

Implication-elimination

$$\frac{A \quad A \to B}{B} \; [\to E]$$

False-elimination

$$\frac{\bot}{A} \; [\bot \; E]$$

True-introduction

$$\frac{}{\top} \; [\top I]$$

A simple proof

Negation-elimination, i.e., both A and $\neg A$ cannot be true at same time

Formally, want to prove $A, \neg A \vdash \bot$

A **proof** is a tree of instances of inference rules.

Assuming that $\neg A$ is defined as $A \rightarrow \bot$, a proof of the above sequent (or argument) is:

$$\frac{A \quad \neg A}{\bot} \; [\rightarrow E]$$

Another simple proof

Given three hypotheses A, B, C, how can we prove $(A \wedge B) \wedge (A \wedge C)$?

Here is a proof:

$$\dfrac{\dfrac{A \quad B}{A \wedge B}\,[\wedge I] \qquad \dfrac{A \quad C}{A \wedge C}\,[\wedge I]}{(A \wedge B) \wedge (A \wedge C)}\,[\wedge I]$$

The rule used at each step is **and-introduction**, i.e., $\wedge I$

Conclusion

What did we cover today?

- Syntax of propositional logic
- Informal semantics of propositional logic formulas
- Simple Natural Deduction proofs

Next time?

- Natural Deduction

Mathematical and Logical Foundations of Computer Science

Lecture 4 - Propositional Logic (Natural Deduction)

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- Symbolic logic
- **Propositional logic**
- Predicate logic

Today

- Natural Deduction proofs

Recap: Connectives & Special Atomic Propositions

Syntax

$$P ::= a \mid P \wedge P \mid P \vee P \mid P \to P \mid \neg P$$

Two special atoms:

- $\top$ which stands for True
- $\bot$ which stands for False

We also introduced four connectives:

- $P \wedge Q$: we have a proof of both P and Q
- $P \vee Q$: we have a proof of at least one of P and Q
- $P \to Q$: if we have a proof of P then we have a proof of Q
- $\neg P$: stands for $P \to \bot$

Recap: Proofs in Propositional Logic

For **formal proofs**, we need two things

1. A **formal** language
   - for representing propositions, arguments
   - here we are using propositional logic
2. A **proof** theory
   - to prove ("infer", "deduce") whether an argument is valid
   - inference rules, which are the building blocks of proofs

Recap: What are inference rules?

Inference rules are the tools we are allowed to use

Careful with the rules you assume otherwise you might be able to prove false statements!

Example of an inference rule (**and-introduction** rule):

$$\frac{A \qquad B}{A \wedge B} \; [\wedge I]$$

These are **rule schemata**, where here A and B are **metavariables** ranging over all possible propositions.

Notation

- Premise(s) at the top
- Conclusion at the bottom
- Name of the inference rule on the right

Recap: Some simple inference rules

And-introduction

$$\frac{A \qquad B}{A \wedge B} \; [\wedge I]$$

implication-elimination

$$\frac{A \to B \qquad A}{B} \; [\to E]$$

False-elimination

$$\frac{\bot}{A} \; [\bot \; E]$$

True-introduction

$$\frac{}{\top} \; [\top I]$$

Recap: A simple proof

Negation-elimination, i.e., both A and $\neg A$ cannot be true at same time

Formally, want to prove $A, \neg A \vdash \bot$

A **proof** is a tree of instances of inference rules.

Assuming that $\neg A$ is defined as $A \rightarrow \bot$, a proof of the above sequent (or argument) is:

$$\frac{A \quad \neg A}{\bot} \; [\rightarrow E]$$

Recap: Another simple proof

Given three hypotheses A, B, C, how can we prove $(A \wedge B) \wedge (A \wedge C)$?

Here is a proof:

$$\dfrac{\dfrac{A \quad B}{A \wedge B}\,[\wedge I] \qquad \dfrac{A \quad C}{A \wedge C}\,[\wedge I]}{(A \wedge B) \wedge (A \wedge C)}\,[\wedge I]$$

The rule used at each step is **and-introduction**, i.e., $[\wedge I]$

Natural Deduction

Framework

- "natural" style of constructing a proof
- start with the given premises
- repeatedly apply the given inference rules
- until you obtain the conclusion

Two key points:

- Can work both forwards and backwards
- Natural doesn't mean there is unique proof

Introduced by **Gentzen** in 1934
and further studied by **Prawitz** in 1965.

Slightly confusing aspect of natural Deduction

Discharging/cancellation of hypothesis

$$\dfrac{\dfrac{}{A}\,1}{\vdots}\\ \dfrac{B}{A \to B}\,1\ [\to I]$$

This is the "implication-introduction" rule.

We don't have to make use of A in which case we can just omit it:

$$\dfrac{B}{A \to B}$$

Cancelling hypothesis continued

Given the hypothesis A, C how can we prove $B \to ((A \wedge B) \wedge (A \wedge C))$?

Here is a proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{A \quad \overline{B}^{\,1}}{A \wedge B}\,[\wedge I] \quad \dfrac{A \quad C}{A \wedge C}\,[\wedge I]}{(A \wedge B) \wedge (A \wedge C)}\,[\wedge I]}{B \to ((A \wedge B) \wedge (A \wedge C))}\,1\,[\to I]$$

At this point, we can also cancel another hypothesis, say A

This gives a proof of

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

using the hypothesis C only

Cancelling hypothesis continued

We proved it forward, but we can also prove it backward:

$$\dfrac{\dfrac{\dfrac{\dfrac{A \quad \overline{B}^{\,1}}{A \wedge B}\,[\wedge I] \quad \dfrac{A \quad C}{A \wedge C}\,[\wedge I]}{(A \wedge B) \wedge (A \wedge C)}\,[\wedge I]}{B \to ((A \wedge B) \wedge (A \wedge C))}\,1\,[\to I]$$

Comprehensive set of inference rules

Rules for $\rightarrow$ (implication)

- implication-introduction

$$\dfrac{\dfrac{\overline{A}^{\,1}}{\vdots}}{\dfrac{B}{A \rightarrow B}\,1\,[\rightarrow I]}$$

- implication-elimination

$$\dfrac{A \rightarrow B \qquad A}{B}\,[\rightarrow E]$$

Comprehensive set of inference rules

Rules for $\neg$ (not)

- Negation-introduction

$$\dfrac{\dfrac{}{A}\,1}{\vdots}\\ \dfrac{\bot}{\neg A}\,1\ [\neg I]$$

- Negation-elimination

$$\dfrac{A \qquad \neg A}{\bot}\ [\neg E]$$

Comprehensive set of inference rules

Rules for $\vee$ (or)

- or-introduction (for any formula B)

$$\frac{A}{A \vee B}\ [\vee I_L] \qquad \frac{A}{B \vee A}\ [\vee I_R]$$

- or-elimination

$$\frac{A \vee B \qquad A \to C \qquad B \to C}{C}\ [\vee E]$$

More comprehensive set of inference rules

Rules for $\wedge$ (and)

- and-introduction

$$\frac{A \qquad B}{A \wedge B}\ [\wedge I]$$

- and-elimination

$$\frac{A \wedge B}{B}\ [\wedge E_R] \qquad \frac{A \wedge B}{A}\ [\wedge E_L]$$

A simple natural Deduction proof

Given $A \to B$ and $B \to C$, give a proof of $A \to C$

Here is a proof:

$$\dfrac{\dfrac{\overline{A}^{\,1} \qquad A \to B}{B}\,[\to E] \qquad B \to C}{\dfrac{C}{A \to C}\,1\,[\to I]}\,[\to E]$$

And backward?

$$\dfrac{\dfrac{\overline{A}^{\,1} \qquad A \to B}{B}\,[\to E] \qquad B \to C}{\dfrac{C}{A \to C}\,1\,[\to I]}\,[\to E]$$

We also need to go forward to prove C

Another simple natural Deduction proof

Given $\neg A \vee B$ and A, how do we derive B?

Here is a proof:

$$\dfrac{\neg A \vee B \qquad \dfrac{\dfrac{\dfrac{A \qquad \neg A}{\bot}\,[\neg E]}{B}\,[\bot\, E]}{\neg A \to B}\,1\,[\to I] \qquad \dfrac{\dfrac{B}{B \to B}\,2\,[\to I]}{}}{B}\,[\vee E]$$

Backward? We go forward because we are left with just B

$$\dfrac{\neg A \vee B \qquad \dfrac{\dfrac{\dfrac{A \qquad \neg A}{\bot}\,[\neg E]}{B}\,[\bot\, E]}{\neg A \to B}\,1\,[\to I] \qquad \dfrac{\dfrac{B}{B \to B}\,2\,[\to I]}{}}{B}\,[\vee E]$$

Forward & backward reasoning in Natural Deduction

We typically go both **forward and backward** in proofs

Show $(B \wedge A)$ given the hypothesis $(A \wedge B)$

Here is a proof:

$$\dfrac{\dfrac{A \wedge B}{B}\,[\wedge E_R] \qquad \dfrac{A \wedge B}{A}\,[\wedge E_L]}{B \wedge A}\,[\wedge I]$$

Complicated looking question

Prove the following:

$R\ , (P \to Q) \wedge (Q \to P)\ , Q \to Z\ , R \to P \vdash Z$

Here is a proof:

$$\dfrac{\dfrac{Q \to Z \qquad \dfrac{\dfrac{R \quad R \to P}{P}\,[\to E] \qquad \dfrac{P \to Q \wedge Q \to P}{P \to Q}\,[\wedge E]}{Q}\,[\to E]}{Z}\,[\to E]}$$

Conclusion

What did we cover today?

- Natural Deduction rules for propositional logic
- Natural Deduction proofs
- Forward & backward reasoning

Next time?

- Classical Reasoning

Mathematical and Logical Foundations of Computer Science

Lecture 6 - Propositional Logic (Classical Reasoning)

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- Symbolic logic
- **Propositional logic**
- Predicate logic

Today

- Classical Reasoning
- Constructive vs. Classical Natural Deduction

Further reading

- Chapter 5 of
  http://leanprover.github.io/logic_and_proof/

Recap: Propositional logic syntax

Syntax:

$$P ::= a \mid P \wedge P \mid P \vee P \mid P \rightarrow P \mid \neg P$$

Two special atoms:

- $\top$ which stands for True
- $\bot$ which stands for False

We also introduced four connectives:

- $P \wedge Q$: we have a proof of both P and Q
- $P \vee Q$: we have a proof of at least one of P and Q
- $P \rightarrow Q$: if we have a proof of P then we have a proof of Q
- $\neg P$: stands for $P \rightarrow \bot$

Recap: Proofs

Natural Deduction

introduction/elimination rules

natural proofs

$$\dfrac{\begin{array}{c}\overline{A}^{\,1}\\ \vdots\\ B\end{array}}{A \to B}\ 1\ [\to I]$$

Classical Reasoning

The proof systems we have seen so far are sometimes called **constructive** or **intuitionistic**, i.e., **proofs** can be viewed as **programs**:

- A proof of $A \wedge B$ can be viewed as a **pair** of a proof of A and a proof of B
- A proof of $A \to B$ can be viewed as a **procedure** which transforms evidence for A into evidence for B
- A proof of $A \vee B$ is either a proof of A or a proof of B, which indicates which one it is

There are other proof systems, called **classical**, which

- rely on Boolean truth values
- introduce additional reasoning principles

Classical Reasoning: Proof by Contradiction

A typical classical reasoning principal is the **"proof by contradiction"** proof technique

Example: Euclid's proof of infinitude of primes

- Assume the negation: Suppose there are only finitely many primes, say $p_1, p_2, \ldots, p_r$
- Consider the number $n = (p_1 \times p_2 \times \ldots \times p_r) + 1$
- Then n cannot be a prime (by assumption)
- But none of the primes $p_1, p_2, \ldots, p_r$ can divide n
- Contradiction

Proof by Contradiction:

- If $\neg A \rightarrow \bot$ then A
- That is, $\neg\neg A \vdash A$

Negation of a negation is?

Can we deduce A and $\neg\neg A$ from each other? That is, are they equivalent?

One direction is easy: $A \vdash \neg\neg A$

Here is the proof:

$$\dfrac{\dfrac{A \quad \dfrac{}{\neg A}\,1}{\bot}\,[\neg E]}{\neg\neg A}\,1\,[\neg I]$$

Can we show the other direction, i.e., $\neg\neg A \vdash A$?

Not using the current set of inference rules we have!

Classical vs. Intutionistic Reasoning in Natural Deduction

Two more (equivalent) assumptions/rules

Law of Excluded Middle (LEM)

- For each A, we can always prove one of A or $\neg A$
- i.e., $\vdash A \vee \neg A$
- E.g., we can assume every even natural number > 2 is the sum of two primes, or not, without knowing which one is true

Double Negation Elimination (DNE)

- $\neg\neg A \vdash A$
- Equivalently, $(\neg A) \rightarrow \bot \vdash A$
- "proof by contradiction"

Classical vs. Intutionistic Reasoning in Natural Deduction

Two more (equivalent) assumptions/rules

As rules:

$$\frac{}{A \vee \neg A}\ [LEM] \qquad \frac{\neg\neg A}{A}\ [DNE]$$

Classical reasoning allows using these two rules

We so far have not used them, and were therefore using what is called **constructive** or **intuitionistic** logic

LEM implies DNE

Assuming $A \vee \neg A$, infer $\neg\neg A \vdash A$

Here is a proof:

$$\dfrac{A \vee \neg A \qquad \dfrac{\dfrac{}{A}\,1}{A \to A}\,1\,[\to I] \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\neg A}\,2 \qquad \neg\neg A}{\bot}\,[\neg E]}{A}\,[\bot\, E]}{\neg A \to A}\,2\,[\to I]}{A}\,[\vee E]$$

DNE implies LEM

Assuming $\neg\neg A \vdash A$, infer $\vdash A \vee \neg A$

Here is a proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(A \vee \neg A)}\,1 \quad \dfrac{\dfrac{}{A}\,2}{A \vee \neg A}\,[\vee I_L]}{\bot}\,[\neg E]}{\neg A}\,2\,[\neg I] \quad \dfrac{\dfrac{}{\neg(A \vee \neg A)}\,1}{A \vee \neg A}\,[\vee I_R]}{\bot}\,[\neg E]}{\neg\neg(A \vee \neg A)}\,1\,[\neg I]}{A \vee \neg A}\,[DNE]$$

Contrapositive

Given an implication $A \to B$, the formula $\neg B \to \neg A$ is called the **"contrapositive"**

Can we prove that an implication implies its contrapositive?

$$A \to B \vdash \neg B \to \neg A$$

Here is a proof (intuitionistic):

$$\dfrac{\dfrac{\dfrac{A \to B \quad \overline{A}^{\,2}}{B}\,[\to E] \qquad \dfrac{}{\neg B}\,1}{\dfrac{\bot}{\neg A}\,2\,[\neg I]}\,[\neg E]}{\neg B \to \neg A}\,1\,[\to I]$$

The other direction holds in classical logic (next slide)

Contrapositive

Given an implication $A \to B$, the formula $\neg B \to \neg A$ is called the **"contrapositive"**

Can we prove that an implication follows from its contrapositive?
$\neg B \to \neg A \vdash A \to B$

Here is a proof (classical):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A}\,1 \qquad \dfrac{\neg B \to \neg A \qquad \dfrac{}{\neg B}\,2}{\neg A}\,[\to E]}{\bot}\,[\neg E]}{\neg\neg B}\,2\,[\neg I]}{B}\,[DNE]}{A \to B}\,1\,[\to I]$$

We used DNE, and hence this proof uses classical reasoning!

Classical Reasoning Through Examples

We will present classical proofs of:

- $(A \to B) \vee (B \to A)$
- $(\neg B \to \neg A) \to (A \to B)$

We saw a classical proof of $(\neg B \to \neg A) \to (A \to B)$ before using DNE – we will present an alternative proof that uses LEM instead

Which we will prove in classical Natural Deduction.

Example 1

Provide a classical Natural Deduction proof of $(A \to B) \vee (B \to A)$

$$\dfrac{\dfrac{}{A \vee \neg A}\,[LEM] \quad \dfrac{\dfrac{\dfrac{\dfrac{}{A}\,1}{B \to A}\,2\,[\to I]}{(A \to B) \vee (B \to A)}\,[\vee I_R]}{A \to (A \to B) \vee (B \to A)}\,1\,[\to I] \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg A}\,3 \quad \dfrac{}{A}\,4}{\bot}\,[\neg E]}{B}\,[\bot\, E]}{A \to B}\,4\,[\to I]}{(A \to B) \vee (B \to A)}\,[\vee I_L]}{\neg A \to (A \to B) \vee (B \to A)}\,3\,[\to I]}{(A \to B) \vee (B \to A)}\,[\vee E]$$

Hypotheses:

- hyp. 1: A
- hyp. 2: B
- hyp. 3: $\neg A$
- hyp. 4: A

Example 2

Provide a classical Natural Deduction proof of
$(\neg B \to \neg A) \to (A \to B)$
Here is a proof:

$$\dfrac{\dfrac{}{B \vee \neg B}\,[LEM] \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg B \to \neg A}\,1 \qquad \dfrac{}{\neg B}\,4}{\neg A}\,[\to E] \qquad \dfrac{}{A}\,2}{\bot}\,[\neg E]}{B}\,[\bot\, E]}{\neg B \to B}\,4\,[\to I] \qquad \dfrac{}{B}\,3}{B}\,[\vee E]}{A \to B}\,3\,[\to I]}{(\neg B \to \neg A) \to (A \to B)}\,2\,[\to I]}\,1\,[\to I]$$

Hypotheses:

- hyp. 1: $\neg B \to \neg A$
- hyp. 2: A
- hyp. 3: B
- hyp. 4: $\neg B$

Conclusion

What did we cover today?

- Classical Reasoning
- Constructive vs. Classical Natural Deduction

Further reading

- Chapter 5 of
  http://leanprover.github.io/logic_and_proof/
- "Proofs and Types", Girard, Taylor, and Lafont, Chapter 5

Next time

- Propositional logic's (classical) semantics

Mathematical and Logical Foundations of Computer Science

Lecture 7 - Propositional Logic (Semantics)

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- Symbolic logic
- **Propositional logic**
- Predicate logic

Today

- semantics of propositional logic
- satisfiability & validity
- truth tables
- soundness & completeness

Further reading:

- Chapter 6 of
  http://leanprover.github.io/logic_and_proof/

Recap: Propositional logic syntax

Syntax:

$$P ::= a \mid P \wedge P \mid P \vee P \mid P \rightarrow P \mid \neg P$$

Two special atoms:

- $\top$ which stands for True
- $\bot$ which stands for False

We also introduced four connectives:

- $P \wedge Q$: we have a proof of both P and Q
- $P \vee Q$: we have a proof of at least one of P and Q
- $P \rightarrow Q$: if we have a proof of P then we have a proof of Q
- $\neg P$: stands for $P \rightarrow \bot$

Syntax vs. Semantics

Syntax

- Rules for allowable formulas in the language
- Syntax for propositional logic:

$$P ::= a \mid P \wedge P \mid P \vee P \mid P \rightarrow P \mid \neg P$$

Semantics

- Assigning meaning/interpretations with formulas
- Semantics for propositional logic: This lecture!

Syntax and Semantics for the English language?

- Syntax: alphabet and grammar
- Semantics: meanings for words

Semantics for Propositional Logic

Semantics assigns **meanings/interpretrations** with **formulas**

The basic notion we use is **"truth value"**

The two standard truth values are "true" and "false"
We use the symbols **T** and **F** respectively

This is a **classical** notion of truth

- i.e., interpretation of each proposition is either true or false
- **Excluded Middle**: for each A we have $A \vee \neg A$
- Here it means for each A, we have that A is either true or false.

WARNING: This is just one possible way to assign meanings!

Semantics for Propositional Logic (continued)

Truth assignment

- Function assigning a truth value for each atomic proposition
- E.g., given 2 atomic propositions p, q, if the formula is $p \vee q$
- then one truth assignment ϕ is $\phi(p) = \mathbf{T}$ and $\phi(q) = \mathbf{F}$
- Also called an "interpretation" or a "valuation"

How many truth valuations do we need to consider for $p \vee q$?

- $2^2 = 4$
- $\phi(p) = \mathbf{T}, \phi(q) = \mathbf{T}$ and $\phi(p) = \mathbf{T}, \phi(q) = \mathbf{F}$ and $\phi(p) = \mathbf{F}, \phi(q) = \mathbf{T}$ and $\phi(p) = \mathbf{F}, \phi(q) = \mathbf{F}$

Conventions:

- The atoms $\top, \bot$ have the interpretations $\mathbf{T}, \mathbf{F}$ respectively
- $\phi(\top) = \mathbf{T}$ and $\phi(\bot) = \mathbf{F}$

Semantics of logical connectives

How to extend the notion of semantics to **compound formulas**?

Define semantics for the four logical connectives: $\vee, \wedge, \rightarrow, \neg$

This is done **recursively bottom-up** over the structure of propositions.

For example given a conjunction $A \wedge B$, we first have to evaluate the truth-values of A and B to compute the truth-value of $A \wedge B$.

I.e., $\phi(A \wedge B) = \mathbf{T}$ iff both $\phi(A) = \mathbf{T}$ and $\phi(B) = \mathbf{T}$.

Semantics of logical connectives

The **extended valuation function** is recursively defined as follows:

- $\phi(\top) = \mathbf{T}$
- $\phi(\bot) = \mathbf{F}$
- $\phi(A \vee B) = \mathbf{T}$ iff either $\phi(A) = \mathbf{T}$ or $\phi(B) = \mathbf{T}$
- $\phi(A \wedge B) = \mathbf{T}$ iff both $\phi(A) = \mathbf{T}$ and $\phi(B) = \mathbf{T}$
- $\phi(A \to B) = \mathbf{T}$ iff $\phi(B) = \mathbf{T}$ whenever $\phi(A) = \mathbf{T}$
- $\phi(\neg A) = \mathbf{T}$ iff $\phi(A) = \mathbf{F}$

Semantics of logical connectives

What is $\phi(2 > 1 \wedge 1 > 0)$? (inequalities are atomic propositions)

$\phi(2 > 1 \wedge 1 > 0) = \mathbf{T}$ because $\phi(2 > 1) = \mathbf{T}$ and $\phi(1 > 0) = \mathbf{T}$

What is $\phi(2 > 1 \wedge 0 > 1)$?

$\phi(2 > 1 \wedge 0 > 1) = \mathbf{F}$ because $\phi(0 > 1) = \mathbf{F}$

What is $\phi(x > 1 \wedge 3 > x)$?

we don't know: it depends on $\phi(x > 1)$ and $\phi(3 > x)$

What is $\phi(x > 1 \vee 2 > x)$?

it depends on $\phi(x > 1)$ and $\phi(2 > x)$

$\phi(x > 1 \vee 2 > x) = \mathbf{T}$ for all combinations

only 2 possible combinations (the atoms are interdependent):
$\phi(x > 1) = \mathbf{T}, \phi(2 > x) = \mathbf{F}$ and $\phi(x > 1) = \mathbf{F}, \phi(2 > x) = \mathbf{T}$

Semantics of logical connectives

What is $\phi(2 > 0 \to 1 > 0)$? (inequalities are atomic propositions)

$\phi(2 > 0 \to 1 > 0) =$ **T** because $\phi(1 > 0) =$ **T**

What is $\phi(0 > 2 \to 1 > 0)$?

still $\phi(0 > 2 \to 1 > 0) =$ **T** because $\phi(1 > 0) =$ **T**

What is $\phi(2 > 0 \to 0 > 1)$?

$\phi(2 > 0 \to 0 > 1) =$ **F** because $\phi(0 > 1) =$ **F** while $\phi(2 > 0) =$ **T**

What is $\phi(0 > 2 \to 0 > 1)$?

$\phi(0 > 2 \to 0 > 1) =$ **T** because $\phi(0 > 2) =$ **F**

What is $\phi(x > 2 \to x > 1)$? it depends on $\phi(x > 2)$ and $\phi(x > 1)$

$\phi(x > 2 \to x > 1) =$ **T** for all possible combinations (the atoms are interdependent): $\phi(x > 2) =$ **T**, $\phi(x > 1) =$ **T** and $\phi(x > 2) =$ **F**, $\phi(x > 1) =$ **T** and $\phi(x > 2) =$ **F**, $\phi(x > 1) =$ **F**

Satisfiability & Validity

The above technique allows answering the following question:

What is the truth value of a formula w.r.t. a given valuation of its atoms?

To analyze the meaning of a formula, we also want to analyze its truth value w.r.t. **all possible combinations** of assignments of truth values with its atoms.

Satisfaction & validity

- Given a valuation ϕ on all atomic propositions, we say that ϕ **satisfies** A if $\phi(A) = \mathbf{T}$.
- A is **satisfiable** if there exists a valuation ϕ on atomic propositions such that $\phi(A) = \mathbf{T}$.
- A is **valid** if $\phi(A) = \mathbf{T}$ for all possible valuations ϕ.

A method to check satisfiability and validity: **truth tables**

Truth tables

Semantics for "or"

$$\phi(A \vee B) = \mathbf{T} \text{ iff either } \phi(A) = \mathbf{T} \text{ or } \phi(B) = \mathbf{T}$$

Truth table for "or"

A	B	$A \vee B$
T	**T**	**T**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

- One row for each valuation
- Last column has the truth value for the corresponding valuation

Truth tables

Semantics for "and"

$$\phi(A \wedge B) = \textbf{T} \text{ iff both } \phi(A) = \textbf{T} \text{ and } \phi(B) = \textbf{T}$$

Truth table for "and"

A	B	$A \wedge B$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

Truth tables

Semantics for "implies"

$$\phi(A \to B) = \textbf{T} \text{ iff } \phi(B) = \textbf{T} \text{ whenever } \phi(A) = \textbf{T}$$

Truth table for "implies"

A	B	$A \to B$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

Truth tables

Semantics for "not"

$$\phi(\neg A) = \mathbf{T} \text{ iff } \phi(A) = \mathbf{F}$$

Truth table for "not"

A	$\neg A$
T	**F**
F	**T**

Semantics for compound formulas

We can now construct a truth table for any propositional formula

- consider all possible truth assignments for the atoms
- then use truth tables for each connective recursively

What is the truth table for $(p \to q) \wedge \neg q$?

p	q	$p \to q$	$\neg q$	$(p \to q) \wedge \neg q$
T	**T**	**T**	**F**	**F**
T	**F**	**F**	**T**	**F**
F	**T**	**T**	**F**	**F**
F	**F**	**T**	**T**	**T**

- 2 atoms, and hence $2^2 = 4$ rows (one per interpretation)
- Use intermediate columns to evaluate sub-formulas
- 2 atoms and 3 connectives hence $2 + 3 = 5$ columns
- Rightmost column gives values of the formula

Satisfiability & validity

A formula is **satisfiable** iff there is a valuation that satisfies it

i.e., if there is a **T** in the rightmost column of its truth table

example: $p \wedge q$ because of the valuation $\phi(p) = \mathbf{T}, \phi(q) = \mathbf{T}$

A formula is **falsifiable** iff there is a valuation that makes it false

i.e., if there is a **F** in the rightmost column of its truth table

example: $p \wedge q$ because of the valuation $\phi(p) = \mathbf{F}, \phi(q) = \mathbf{T}$

A formula is **unsatisfiable** iff no valuation satisfies it

i.e., the cells of the rightmost column of its truth table all contain **F**

example: $p \wedge \neg p$ **(contradiction)**

A formula is **valid** iff every valuation satisfies it

i.e., the cells of the rightmost column of its truth table all contain **T**

example: $p \vee \neg p$ **(tautology)**

Validity of arguments using semantics

Validity of an argument

- **syntactically:** we can derive the conclusion from the premises
- **semantically:** the conclusion is true whenever the premises are

Formally, we write

$$P_1, \ldots, P_n \models C$$

if the corresponding argument is **semantically valid**

i.e., every valuation that evaluates each of the premises $P_1, \ldots, P_n$ to **T** also evaluates the conclusion C to **T**

Checking validity

- Already seen how to do this using "natural deduction"
- Truth tables is yet another way
- Bonus: yields counterexample if argument is invalid

Checking (semantic) validity

Is $P \to Q, \neg Q \models \neg P$ (semantically) valid?

P	Q	$P \to Q$	$\neg Q$	$\neg P$
T	**T**	**T**	**F**	**F**
T	**F**	**F**	**T**	**F**
F	**T**	**T**	**F**	**T**
F	**F**	**T**	**T**	**T**

Argument is valid: any row where conclusion is **F** then at least one of the premises is also **F**

Note that checking $P_1, \ldots, P_n \models C$ is equivalent to checking the validity of $P_1 \to \cdots P_n \to C$

i.e., that the cells of the rightmost column of the truth table for $P_1 \to \cdots P_n \to C$ all contain **T**

Checking (semantic) validity

Is $\neg P \rightarrow \neg R, R \models \neg P$ (semantically) valid?

P	R	$\neg P$	$\neg R$	$\neg P \rightarrow \neg R$	R	$\neg P$
T	**T**	**F**	**F**	**T**	**T**	**F**
T	**F**	**F**	**T**	**T**	**F**	**F**
F	**T**	**T**	**F**	**F**	**T**	**T**
F	**F**	**T**	**T**	**T**	**F**	**T**

Argument is invalid

- Look at the first row
- Conclusion is **F**, but both premises are **T**
- Can we add a premise to make the argument valid?
  - Yes, we can add $\neg R$, which would be **F** in the first row

Proving anything using contradictions!

Is $P, \neg P \models C$ is (semantically) valid?

P	C	$\neg P$	C
T	**T**	**F**	**T**
T	**F**	**F**	**F**
F	**T**	**T**	**T**
F	**F**	**T**	**F**

Argument is (trivially) valid:

- Look at any row (we only have to look at rows where the conclusion is **F**)
- One of P and $\neg P$ is **F**

Truth Tables vs. Natural Deduction

Pros and cons of two ways of checking validity

Truth tables	**Natural deduction**
shows validity in a restricted setting (Boolean truth values)	checks validity in general setting (by an actual proof!)
simple, easy to automate	more difficult to automate
size of truth table is huge: exponential in number of atoms	typically scales better than brute force search
generates counterexamples if invalid	no easy way to check validity (other than actually proving)

Soundness & Completeness

Given a deduction system such as Natural deduction, a formula is said to be **provable** if there is a proof of it in that deduction system

- This is a **syntactic** notion
- it asserts the existence of a syntactic object: a proof
- typically written $\vdash A$

A formula A is **valid** if $\phi(A) = \mathbf{T}$ for all possible valuations ϕ

- it is a **semantic** notion
- it is checked w.r.t. valuations that give meaning to formulas

Soundness: a deduction system is sound w.r.t. a semantics if every provable formula is valid

- i.e., if $\vdash A$ then $\models A$

Completeness: a deduction system is complete w.r.t. a semantics if every valid formula is provable

- i.e., if $\models A$ then $\vdash A$

Soundness & Completeness

Classical Natural Deduction is

- **sound** and
- **complete**

w.r.t. the **truth table semantics**

Proving those properties is done within the **metatheory**

- Soundness is easy. It requires proving that each rule is valid.

  For example:

$$\frac{A \quad B}{A \wedge B} \; [\wedge I]$$

  is valid because $A, B \models A \wedge B$

- Completeness is harder

We will not prove them here

Conclusion

What did we cover today?

- semantics of propositional logic
- satisfiability & validity
- truth tables
- soundness & completeness

Further reading

- Chapter 6 of
  http://leanprover.github.io/logic_and_proof/

Next time?

- equivalences
- normal forms

Mathematical and Logical Foundations of Computer Science

Lecture 8 - Propositional Logic (Equivalences & Normal Forms)

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- Symbolic logic
- **Propositional logic**
- Predicate logic

Today

- Logical Equivalences
- Proving logical Equivalences in Natural Deduction
- Proving logical Equivalences using truth tables
- Normal forms

Further reading:

- Chapter 3 of
  http://leanprover.github.io/logic_and_proof/

Recap: Propositional logic syntax

Syntax:

$$P ::= a \mid P \wedge P \mid P \vee P \mid P \to P \mid \neg P$$

Lower-case letters are atoms: p, q, r, etc.

Upper-case letters stand for any proposition: P, Q, R, etc.

Two special atoms:

- $\top$ which stands for True
- $\bot$ which stands for False

We also introduced four connectives:

- $P \wedge Q$: we have a proof of both P and Q
- $P \vee Q$: we have a proof of at least one of P and Q
- $P \to Q$: if we have a proof of P then we have a proof of Q
- $\neg P$: stands for $P \to \bot$

Recap: Proofs

Natural Deduction

introduction/elimination rules

natural proofs

$$\dfrac{\begin{array}{c}\overline{A}^{\,1}\\ \vdots\\ B\end{array}}{A \to B}\ 1\ [\to I]$$

Recap: Classical Reasoning

Two (equivalent) classical rules

Law of Excluded Middle (LEM)

- $\vdash A \vee \neg A$
- We will write LEM for $A \vee \neg A$

Double Negation Elimination (DNE)

- "proof by contradiction"
- $\neg\neg A \vdash A$
- Equivalently, $(\neg A) \to \bot \vdash A$
- Equivalently, $\vdash (\neg\neg A) \to A$
- We will write DNE for $(\neg\neg A) \to A$

Classical system:

- Classical Natural Deduction with LEM and DNE rules

Recap: Semantics

Semantics for "implies"

$$\phi(A \to B) = \textbf{T} \text{ iff } \phi(B) = \textbf{T} \text{ whenever } \phi(A) = \textbf{T}$$

Truth table for "implies"

P	Q	$P \to Q$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

Logical equivalences

Let $A \leftrightarrow B$ be defined as $(A \rightarrow B) \wedge (B \rightarrow A)$

- it means that A and B are logically equivalent
- A and B have the same semantics
- $\phi(A) = \mathbf{T}$ if and only if $\phi(B) = \mathbf{T}$
- A is provable if and only if B is provable
- this is called a "bi-implication"
- read as "A if and only if B"

Example: we showed that DNE and LEM are equivalent

We have already proved:

- DNE $\rightarrow$ LEM
- LEM $\rightarrow$ DNE

It is then straightforward to derive a proof of DNE $\leftrightarrow$ LEM

Logical equivalences

Another example: we showed that implications are classically equivalent to their contrapositives (in classical logic)

We have proved:

- $(A \to B) \to (\neg B \to \neg A)$ in intuitionistic logic
- $(\neg B \to \neg A) \to (A \to B)$ in classical logic

It is then straightforward to derive a proof of
$(A \to B) \leftrightarrow (\neg B \to \neg A)$ in classical Natural Deduction

Equivalences are for example useful in proofs to "replace" a formula by another equivalent formula

We will now present some standard ones

Logical equivalences

We are going to prove:

- De Morgan's law (I): $\neg(A \vee B) \leftrightarrow (\neg A \wedge \neg B)$
- De Morgan's law (II): $\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B)$
- implication elimination: $(A \to B) \leftrightarrow (\neg A \vee B)$

Some of these proofs are **intuitionistic**, while some are **classical**

In addition you can try to prove:

- Commutativity of $\wedge$: $(A \wedge B) \leftrightarrow (B \wedge A)$
- Commutativity of $\vee$: $(A \vee B) \leftrightarrow (B \vee A)$
- Associativity of $\wedge$: $((A \wedge B) \wedge C) \leftrightarrow (A \wedge (B \wedge C))$
- Associativity of $\vee$: $((A \vee B) \vee C) \leftrightarrow (A \vee (B \vee C))$
- Distributivity of $\wedge$ over $\vee$: $(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$
- Distributivity of $\vee$ over $\wedge$: $(A \vee (B \wedge C)) \leftrightarrow ((A \vee B) \wedge (A \vee C))$
- Double negation elimination: $(\neg\neg A) \leftrightarrow A$
- Idempotence: $(A \wedge A) \leftrightarrow A$ and $(A \vee A) \leftrightarrow A$

Logical equivalences

As our Natural Deduction equivalence proofs will all be as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A}\,1}{\vdots}}{B}}{A \to B}\,1\,[\to I] \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{B}\,2}{\vdots}}{A}}{B \to A}\,2\,[\to I]$$
$$\dfrac{\qquad\qquad}{A \leftrightarrow B}\,[\wedge I]$$

then, we will focus on proving

- $A \vdash B$ (left-to-right implication)
- $B \vdash A$ (right-to-left implication)

De Morgan's Laws (I): Negation of OR

Show the logical equivalence $\neg(A \vee B) \leftrightarrow (\neg A \wedge \neg B)$ in Natural Deduction

We first prove the left-to-right implication:
$\neg(A \vee B) \vdash (\neg A \wedge \neg B)$

Here is a proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg(A \vee B) \quad \dfrac{\dfrac{}{A}\,1}{A \vee B}\,[\vee I_L]}{\bot}\,[\neg E]}{\neg A}\,1\,[\neg I] \quad \dfrac{\dfrac{\dfrac{\neg(A \vee B) \quad \dfrac{\dfrac{}{B}\,2}{A \vee B}\,[\vee I_R]}{\bot}\,[\neg E]}{\neg B}\,2\,[\neg I]}{\neg A \wedge \neg B}\,[\wedge I]$$

Proof only uses intuitionistic rules!

Other direction on the next slide

De Morgan's Laws (I): Negation of OR

Show the logical equivalence $\neg(A \vee B) \leftrightarrow (\neg A \wedge \neg B)$ in Natural Deduction

We now prove the right-to-left implication:
$(\neg A \wedge \neg B) \vdash \neg(A \vee B)$

Here is a proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\neg A \wedge \neg B}{\neg A}\,[\wedge E]}{A}\,2 \quad \dfrac{\neg A \wedge \neg B}{\neg A}\,[\wedge E]}{\bot}\,[\neg E]}{A \to \bot}\,2\,[\to I] \quad \dfrac{\dfrac{\dfrac{\neg A \wedge \neg B}{\neg B}\,[\wedge E]}{B}\,3 \quad \dfrac{\neg A \wedge \neg B}{\neg B}\,[\wedge E]}{\bot}\,[\neg E]}{B \to \bot}\,3\,[\to I]}{A \vee B}\,1}{\bot}\,[\vee E]}{\neg(A \vee B)}\,1\,[\neg I]$$

Again, we only used intuitionistic rules!

De Morgan's Laws (II): Negation of AND

Show the logical equivalence $\neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$ in Natural Deduction

We first prove the right-to-left implication: $\neg A \vee \neg B \vdash \neg(A \wedge B)$

Here is a proof:

$$\dfrac{\neg A \vee \neg B \qquad \dfrac{\dfrac{\dfrac{\neg A \quad 2 \qquad \dfrac{\overline{A \wedge B}^{\,1}}{A}\,[\wedge E_L]}{\bot}\,[\neg E]}{\neg A \to \bot}\,2\,[\to I] \qquad \dfrac{\dfrac{\dfrac{\neg B \quad 3 \qquad \dfrac{\overline{A \wedge B}^{\,1}}{B}\,[\wedge E_R]}{\bot}\,[\neg E]}{\neg B \to \bot}\,3\,[\to I]}{\bot}\,[\vee E]}{\neg(A \wedge B)}\,1\,[\neg I]$$

Proof uses intuitionistic rules!

De Morgan's Laws (II): Negation of AND

Show the logical equivalence $\neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$ in Natural Deduction

We now prove the left-to-right implication: $\neg(A \wedge B) \vdash \neg A \vee \neg B$

Here is a proof (classical—we use DNE thrice):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg A}\,2}{\neg A \vee \neg B}\,[\vee I_L] \quad \dfrac{}{\neg(\neg A \vee \neg B)}\,1}{\bot}\,[\neg E]}{\neg\neg A}\,2\,[\neg I]}{A}\,[DNE] \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg B}\,3}{\neg A \vee \neg B}\,[\vee I_R] \quad \dfrac{}{\neg(\neg A \vee \neg B)}\,1}{\bot}\,[\neg E]}{\neg\neg B}\,3\,[\neg I]}{B}\,[DNE]}{A \wedge B}\,[\wedge I] \quad \neg(A \wedge B)}{\bot}\,[\neg E]}{\neg\neg(\neg A \vee \neg B)}\,1\,[\neg I]}{\neg A \vee \neg B}\,[DNE]$$

Expressing $\rightarrow$ using $\neg$ and $\vee$

Show the logical equivalence: $A \rightarrow B \leftrightarrow \neg A \vee B$

We first prove the left-to-right implication $A \rightarrow B \vdash \neg A \vee B$

Here is a proof (classical—it uses LEM):

$$\dfrac{\dfrac{}{A \vee \neg A}\,[LEM] \qquad \dfrac{\dfrac{\dfrac{\dfrac{\overline{A}^{\,1} \qquad A \rightarrow B}{B}\,[\rightarrow E]}{\neg A \vee B}\,[\vee I_R]}{A \rightarrow (\neg A \vee B)}\,1\,[\rightarrow I] \qquad \dfrac{\dfrac{\dfrac{\overline{}^{\,2}}{\neg A}\,[\vee I_L]}{\neg A \vee B}}{\neg A \rightarrow (\neg A \vee B)}\,2\,[\rightarrow I]}{\neg A \vee B}\,[\vee E]$$

The other direction holds intuitionistically (next slide)

Expressing $\rightarrow$ using $\neg$ and $\vee$

Show the logical equivalence: $A \rightarrow B \leftrightarrow \neg A \vee B$

We now prove the right-to-left implication $\neg A \vee B \vdash A \rightarrow B$

Here is a proof (intuitionistic):

$$\dfrac{\neg A \vee B \qquad \dfrac{\dfrac{\dfrac{\dfrac{\neg A \quad^{2} \qquad A \quad^{1}}{\bot}\,[\neg E]}{B}\,[\bot\, E]}{\neg A \rightarrow B}\,2\,[\rightarrow I] \qquad \dfrac{\dfrac{B}{B}\,^{3}}{B \rightarrow B}\,3\,[\rightarrow I]}{B}\,[\vee E]}{A \rightarrow B}\,1\,[\rightarrow I]$$

Logical equivalences using truth tables

Classically, two formulas are logically equivalent if they have the same semantics.

I.e., they have the same truth values for all valuations.

E.g., an implication and its contrapositive are logically equivalent:

Show that $(A \to B) \leftrightarrow (\neg B \to \neg A)$ using a truth table

A	B	$A \to B$	$\neg B$	$\neg A$	$\neg B \to \neg A$
T	**T**	**T**	**F**	**F**	**T**
T	**F**	**F**	**T**	**F**	**F**
F	**T**	**T**	**F**	**T**	**T**
F	**F**	**T**	**T**	**T**	**T**

The two formulas are equivalent because the two columns for $A \to B$ and $\neg B \to \neg A$ are identical

Normal forms

Among the formulas equivalent to a given formula, some are of particular interest:

- **Conjunctive Normal forms** (CNF)
  - $(A \vee B \vee C) \wedge (D \vee X) \wedge (\neg A)$
  - ANDs of ORs of literals (atoms or negations of atoms)
  - A **clause** in this context is a disjunction of literals
- **Disjunctive Normal Form** (DNF)
  - $(P \wedge Q \wedge A) \vee (R \wedge \neg Q) \vee (\neg A)$
  - ORs of ANDs of literals
  - A **clause** in this context is a conjunction of literals

All the variables above and the ones used in the rest of this lecture stand for atomic propositions

Every formula can be expressed in DNF

Every proposition is equivalent to a formula in DNF (OR of ANDs)!

Can you find propositions in DNF that are logically equivalent to:

- $(A \wedge \neg B \wedge \neg C) \vee X$

  Already in DNF

- Z

  Already in DNF

- $A \rightarrow B$

  Logically equivalent to $\neg A \vee B$

- $\neg(A \wedge B)$

  Logically equivalent (by De Morgan's law) to $\neg A \vee \neg B$

Every formula can be expressed in CNF

Every proposition is equivalent to a formula in CNF (AND of ORs)!

Can you find propositions in CNF that are logically equivalent to:

- $(A \vee \neg B \vee \neg C) \wedge X$

  Already in CNF

- Z

  Already in CNF

- $A \rightarrow B$

  Logically equivalent to $\neg A \vee B$

- $\neg(A \vee B)$

  Logically equivalent (by De Morgan's law) to $\neg A \wedge \neg B$

Every proposition can be expressed in DNF

Every proposition can be expressed in DNF (ORs of ANDs)!

Express $(P \to Q) \wedge Q$ in DNF

We do it using a truth table

P	Q	$(P \to Q)$	$(P \to Q) \wedge Q$
T	**T**	**T**	**T**
T	**F**	**F**	**F**
F	**T**	**T**	**T**
F	**F**	**T**	**F**

- Enumerate all the **T** rows from the conclusion column
  - Row 1 gives $P \wedge Q$
  - Row 3 gives $\neg P \wedge Q$
- Take **OR** of these formulas
- **Final answer** is $(P \wedge Q) \vee (\neg P \wedge Q)$

Every formula can be expressed in CNF

Every proposition can be expressed in CNF (ANDs of ORs)!

Express $(P \to Q) \wedge Q$ in CNF

We do it by using a truth table

P	Q	$(P \to Q)$	$(P \to Q) \wedge Q$
T	**T**	**T**	**T**
T	**F**	**F**	**F**
F	**T**	**T**	**T**
F	**F**	**T**	**F**

- Enumerate all the **F** rows from the conclusion column
  - Row 2 gives $P \wedge \neg Q$
  - Row 4 gives $\neg P \wedge \neg Q$
- Do **AND** of negations of each of these formulas
- We obtain $\neg(P \wedge \neg Q) \wedge \neg(\neg P \wedge \neg Q)$
- **Finally**: equivalent to $(\neg P \vee Q) \wedge (P \vee Q)$ by De Morgan

Making use of equivalences to convert to CNF/DNF

If $P \leftrightarrow Q$ and P occurs in A, then replacing P by Q in A leads to a proposition B, such that $A \leftrightarrow B$

Example:

- consider the formula $P \rightarrow Q \rightarrow (P \wedge Q)$
- we know that classically $(Q \rightarrow (P \wedge Q)) \leftrightarrow (\neg Q \vee (P \wedge Q))$
- this is an instance of $(A \rightarrow B) \leftrightarrow (\neg A \vee B)$
- when replacing $Q \rightarrow (P \wedge Q)$ by $\neg Q \vee (P \wedge Q)$ in $P \rightarrow Q \rightarrow (P \wedge Q)$, we obtain $P \rightarrow (\neg Q \vee (P \wedge Q))$
- $P \rightarrow Q \rightarrow (P \wedge Q)$ and $P \rightarrow (\neg Q \vee (P \wedge Q))$ are equivalent

Making use of equivalences to convert to CNF/DNF

We can convert a formula to an equivalent formula in CNF or DNF using the equivalences presented above (slide 10)

Example: express $(P \to Q) \wedge Q$ in CNF using known equivalences

- $(P \to Q) \wedge Q$ $(\boxed{P \to Q}) \wedge Q$
- $\leftrightarrow (\neg P \vee Q) \wedge Q$ – using $(A \to B) \leftrightarrow (\neg A \vee B)$

Example: express $\neg(P \wedge \neg Q) \wedge \neg(\neg P \wedge \neg Q)$ in CNF using known equivalences

- $\neg(P \wedge \neg Q) \wedge \neg(\neg P \wedge \neg Q)$ $\boxed{\neg(P \wedge \neg Q)} \wedge \neg(\neg P \wedge \neg Q)$
- $\leftrightarrow (\neg P \vee \neg\neg Q) \wedge \neg(\neg P \wedge \neg Q)$ $(\neg P \vee \neg\neg Q) \wedge \boxed{\neg(\neg P \wedge \neg Q)}$ – using de Morgan
- $\leftrightarrow (\neg P \vee \neg\neg Q) \wedge (\neg\neg P \vee \neg\neg Q)$
  $(\neg P \vee \boxed{\neg\neg Q}) \wedge (\neg\neg P \vee \neg\neg Q)$ – using de Morgan
- $\leftrightarrow (\neg P \vee Q) \wedge (\neg\neg P \vee \neg\neg Q)$ $(\neg P \vee Q) \wedge (\boxed{\neg\neg P} \vee \neg\neg Q)$ – using double negation elim.
- $\leftrightarrow (\neg P \vee Q) \wedge (P \vee \neg\neg Q)$ $(\neg P \vee Q) \wedge (P \vee \boxed{\neg\neg Q})$ – using

Making use of equivalences to convert to CNF/DNF

Example: express $(P \to Q) \wedge Q$ in DNF using known equivalences

- $(P \to Q) \wedge Q$
- $\leftrightarrow (\neg P \vee Q) \wedge Q$ – using $(A \to B) \leftrightarrow (\neg A \vee B)$
- $\leftrightarrow Q \wedge (\neg P \vee Q)$ – using commutativity of $\wedge$
- $\leftrightarrow (Q \wedge \neg P) \vee (Q \wedge Q)$ – using distributivity of $\wedge$ over $\vee$

Conclusion

What did we cover today?

- Logical Equivalences
- Proving logical Equivalences in Natural Deduction
- Proving logical Equivalences using truth tables
- Normal forms

Further reading:

- Chapter 3 of
  http://leanprover.github.io/logic_and_proof/

Next time

- SAT

Mathematical and Logical Foundations of Computer Science

Lecture 9 - Propositional Logic (SAT)

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- Symbolic logic
- **Propositional logic**
- Predicate logic

Today

- History of Computing
- SAT (first $\mathcal{NP}$-hard problem)
- Algorithms for SAT

Recap: Propositional logic syntax

Syntax:

$$P ::= a \mid P \wedge P \mid P \vee P \mid P \rightarrow P \mid \neg P$$

Two special atoms:

- $\top$ which stands for True
- $\bot$ which stands for False

We also introduced four connectives:

- $P \wedge Q$: we have a proof of both P and Q
- $P \vee Q$: we have a proof of at least one of P and Q
- $P \rightarrow Q$: if we have a proof of P then we have a proof of Q
- $\neg P$: stands for $P \rightarrow \bot$

Recap: Normal forms

Among the formulas equivalent to a given formula, some are of particular interest (the variables here stand for atoms):

- **Conjunctive Normal forms** (CNF)
  - $(A \vee B \vee C) \wedge (D \vee X) \wedge (\neg A)$
  - ANDs of ORs of literals (atoms or negations of atoms)
  - A **clause** in this context is a disjunction of literals
- **Disjunctive Normal Form** (DNF)
  - $(P \wedge Q \wedge A) \vee (R \wedge \neg Q) \vee (\neg A)$
  - ORs of ANDs of literals
  - A **clause** in this context is a conjunction of literals

Theorem: Every proposition is equivalent to a formula in CNF!

Theorem: Every proposition is equivalent to a formula in DNF!

Recap: Every proposition can be expressed in DNF

Every proposition can be expressed in DNF (ORs of ANDs)!

Express $(P \to Q) \wedge Q$ in DNF

We do it using a truth table

P	Q	$(P \to Q)$	$(P \to Q) \wedge Q$
T	**T**	**T**	**T**
T	**F**	**F**	**F**
F	**T**	**T**	**T**
F	**F**	**T**	**F**

- Enumerate all the **T** rows from the conclusion column
  - Row 1 gives $P \wedge Q$
  - Row 3 gives $\neg P \wedge Q$
- Take **OR** of these formulas
- **Final answer** is $(P \wedge Q) \vee (\neg P \wedge Q)$

Recap: Every formula can be expressed in CNF

Every proposition can be expressed in CNF (ANDs of ORs)!

Express $(P \to Q) \wedge Q$ in CNF

We do it by using a truth table

P	Q	$(P \to Q)$	$(P \to Q) \wedge Q$
T	**T**	**T**	**T**
T	**F**	**F**	**F**
F	**T**	**T**	**T**
F	**F**	**T**	**F**

- Enumerate all the **F** rows from the conclusion column
  - Row 2 gives $P \wedge \neg Q$
  - Row 4 gives $\neg P \wedge \neg Q$
- Do **AND** of negations of each of these formulas
- We obtain $\neg(P \wedge \neg Q) \wedge \neg(\neg P \wedge \neg Q)$
- **Finally**: equivalent to $(\neg P \vee Q) \wedge (P \vee Q)$ by De Morgan

Satisfiability of CNF formulas

Problem definition: Given a CNF formula can we set **T** or **F** value to each variable to satisfy the formula?

- **Example**: Consider the formula $(A \vee \neg B) \wedge (C \vee B)$
- Is it satisfiable?
- Satisfiable by setting $A =$ **T**, $B =$ **F** and $C =$ **T**
- Known as **CNF Satisfiability** or simply **SAT**

First a bit of history

History of Computing

1930s

- Alan Turing invented the Turing Machine in 1936
- Mathematical model of computable functions (as abstract machines)
- Basis of modern computers
- Biography: *Alan Turing: The Enigma*
- Movie: *The Imitation Game*

1940s and 1950s

- Code-breaking by Allies in Bletchley Park

  Go visit the *National Museum of Computing*
- Should not really have been breakable

  Made use of manual & hardware errors
- Alan Turing was heavily involved

History of Computing

1960s

- People began to look at **general** ways to solve a problem, rather than solving given instance!
  - Is $(A \vee \neg B) \wedge (\neg A) \wedge (B \vee Z \vee \neg X)$ satisfiable?
  - How (fast) can we check in general if a CNF formula is satisfiable?
- Many **known** problems had **polynomial** running time
  - Actually even n^4 or smaller
- Polynomial time became accepted as **standard of efficiency**
  - $\mathcal{P}$: The class of problems solvable in polynomial time (in size of input)
- **Claim**: Any exponential (ultimately) **beats** any polynomial
  - $1.0000000001^n > n^{10000000000000000000000000}$ if n is large enough

History of Computing

1970s

- But still **many** problems no one knew how to solve in polynomial time!
- **CNF satisfiability (SAT)**
  - Say we have N atoms and M clauses
  - No known algorithm to solve in time polynomial in N and M
  - **Brute force**: does 2^N truth assignments, and checks in N time if each of the M clauses is satisfied
  - So, total running time is $2^N \cdot N \cdot M$
  - Note that the input size is $N + M$
- Can we design a polynomial time algorithm for SAT?
- Or show that such an algorithm cannot exist?
- $\mathcal{NP}$: class of problems where we can verify a potential solution in polynomial time

$\mathcal{P}$ vs. $\mathcal{NP}$

$\mathcal{P}$: the class of problems which we can solve in polynomial time
$\mathcal{NP}$: the class of problems where we can verify a potential solution/answer in polynomial time

Clearly, $\mathcal{P} \subseteq \mathcal{NP}$ (solving is a (hard) way of verifying)

What about the other direction? Is $\mathcal{P} = \mathcal{NP}$?

- Status unknown!
- Million dollar question

What do most people believe?

- $\mathcal{P}$ is not equal to $\mathcal{NP}$

Why haven't we been able to prove it then?

- Hard to rule out all possible polytime algorithms?

Hardness for the class $\mathcal{NP}$

$\mathcal{NP}$: the class of problems where we can verify a potential solution/answer in polynomial time

Definition: A problem is $\mathcal{NP}$-hard if it is **at least as hard as** any problem in $\mathcal{NP}$.

More precisely, a problem X is $\mathcal{NP}$-hard if any problem $Y \in \mathcal{NP}$ can be solved

- using an oracle for solving X
- plus a polynomial overhead for translating between X and Y

If $\mathcal{P} \neq \mathcal{NP}$ then a problem being $\mathcal{NP}$-hard means it cannot be solved in polynomial time!

Great, except no one knew how to show existence of a single $\mathcal{NP}$-hard problem!

The first $\mathcal{NP}$-hard problem

Cook-Levin Theorem (1971/1973):
CNF-Satisfiability (**SAT**) is $\mathcal{NP}$-hard

How do you show a problem, say X, is $\mathcal{NP}$-hard?

- A polytime reduction from any of the known $\mathcal{NP}$-hard problems, say SAT, to X
- That is, show how you can solve SAT using an oracle for X
- Plus a polynomial overhead for the translation

Tens of thousands of problems known to be $\mathcal{NP}$-hard

Significance of SAT

Many practical problems can be encoded into SAT
(e.g., formal verification, planning/scheduling, etc.)

A possible solution (valuation) can be verified "efficiently"

No known algorithm to solve the problem "efficiently" in all cases

In practice, SAT solvers are very efficient
($\mathcal{NP}$-hardness is the worst case)

Special cases

Let n-**SAT** be the SAT problem restricted to n-CNFs, i.e., where clauses are disjunctions of n literals

- 1-SAT is in $\mathcal{P}$
- 2-SAT is in $\mathcal{P}$
- 3-SAT is $\mathcal{NP}$-hard

Why not consider DNF instead of CNF?

Theorem: Any propositional formula can be expressed in CNF

Theorem: Any propositional formula can be expressed in DNF

Theorem: CNF satisfiability is $\mathcal{NP}$-hard

How hard is DNF satisfiability?

- Example of a DNF formula:
  $(A \wedge \neg B \wedge C) \vee (\neg X \wedge Y) \vee (Z)$
- Is it satisfiable?
- **Trivial** to check in polytime!
- Just pick any clause, and set variables to **T** or **F**.

Why not use DNFs then?

Because changing a formula from CNF to DNF can cause exponential blowup!

Why not consider DNF instead of CNF?

Because changing a formula from CNF to DNF can cause exponential blowup!

Convert $(A \vee B) \wedge (C \vee D)$ into DNF

Remember: $P \wedge (Q \vee R) \leftrightarrow (P \wedge Q) \vee (P \wedge R)$

$$
\begin{aligned}
& (A \vee B) \wedge (C \vee D) \\
\leftrightarrow \quad & ((A \vee B) \wedge C) \vee ((A \vee B) \wedge D) \\
\leftrightarrow \quad & (C \wedge (A \vee B)) \vee (D \wedge (A \vee B)) \\
\leftrightarrow \quad & (C \wedge A) \vee (C \wedge B) \vee (D \wedge A) \vee (D \wedge B)
\end{aligned}
$$

Consider the CNF formula: $(P_1 \vee Q_1) \wedge \cdots \wedge (P_n \vee Q_n)$

Expressing this formula in DNF requires 2^n clauses

Algorithms for SAT?

Brute force for SAT with N variables and M clauses needs $2^N \cdot N \cdot M$ time

- There are 2^N truth assignments
- For each truth assignment and each clause, verify if it is satisfied in N time

Can we solve SAT faster than 2^N? Say 1.999999999^N?

Conjecture (Strong Exponential Time Hypothesis (SETH)): SAT cannot be solved in $(2-\alpha)^N \cdot \text{poly}(N+M)$ time for any constant $\alpha > 0$

SAT solvers

Many state-of-the-art SAT solvers are based on the **Davis-Putman-Logemann-Loveland** algorithm (DPLL)

Basic idea (does a lot of pruning instead of brute force):

1. **Easy** cases
   - Atom p only appears as either p or $\neg p$ (but not both): assign truth value accordingly
2. **Branch** on choosing a variable p and set a truth value to it
   - This choice needs to be done **cleverly**
   - If $p = \mathbf{T}$: remove all clauses containing p and remove all literals $\neg p$ from clauses
   - If $p = \mathbf{F}$: remove all clauses containing $\neg p$ and remove all literals p from clauses
3. Keep running the above steps **until**
   - All clauses have been removed (all true): return SAT
   - One clause is empty (one is false): **backtrack** in Step 2 and choose a different truth value for p; if it is not possible to backtrack, return UNSAT

SAT solvers

Apply the DPLL algorithm to

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

Here is a possible run of the algorithm:

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

$p = \mathbf{T}$

$(q \vee r) \wedge (\neg q \vee r)$

$q = \mathbf{T}$

(r)

$r = \mathbf{T}$

SAT

SAT Solvers

Let us use this SAT solver: https://jfmc.github.io/z3-play/

two variables, two clauses:

$(p \vee q) \wedge (\neg q)$

```
(declare-const p Bool)
(declare-const q Bool)
(define-fun conjecture () Bool
  (and (or p q) (not q))
)
(assert conjecture)
(check-sat)
(get-model)
```

SAT Solvers

Let us use this SAT solver: https://jfmc.github.io/z3-play/

three variables, three clauses:

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

```
(declare-const p Bool)
(declare-const q Bool)
(declare-const r Bool)
(define-fun conjecture () Bool
  (and (or p q r) (or (not p) (not q)) (or q (not r)))
)
(assert conjecture)
(check-sat)
(get-model)
```

SAT Solvers

Let us use this SAT solver: https://jfmc.github.io/z3-play/

four variables, five clauses:

$(p \vee q \vee \neg r) \wedge (q \vee r \vee \neg s) \wedge (\neg p \vee q \vee r) \wedge (\neg p) \wedge (\neg r \vee s)$

```
(declare-const p Bool)
(declare-const q Bool)
(declare-const r Bool)
(declare-const s Bool)
(define-fun conjecture () Bool
  (and (or p q (not r)) (or q r (not s)) (or (not p) q r)
       (not p) (or (not r) s)
  )
)
(assert conjecture)
(check-sat)
(get-model)
```

SAT Solvers

Let us use this SAT solver: https://jfmc.github.io/z3-play/

five variables, eight clauses:

$$(p \vee t \vee s) \wedge (q \vee r \vee \neg s \vee \neg t) \wedge (\neg t \vee r) \wedge (p \vee \neg q \vee s)$$
$$\wedge (p \vee q \vee r \vee \neg t) \wedge (q \vee r \vee \neg s) \wedge (p \vee \neg s) \wedge (\neg p \vee q \vee s \vee t)$$

```
(declare-const p Bool)
(declare-const q Bool)
(declare-const r Bool)
(declare-const s Bool)
(declare-const t Bool)
(define-fun conjecture () Bool
  (and (or p t s) (or q r (not s) (not t)) (or (not t) r) (or p (not q) s)
       (or p q r (not t)) (or q r (not s)) (or p (not s)) (or (not p) q s t)
  )
)
(assert conjecture)
(check-sat)
(get-model)
```

Conclusion

What did we cover today?

- History of Computing
- SAT (first $\mathcal{NP}$-hard problem)
- Algorithms for SAT

Next time?

- Propositional logic (wrap-up)

Mathematical and Logical Foundations of Computer Science

Lecture 10 - Propositional Logic (Wrap-up)

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- Symbolic logic
- **Propositional logic**
- Predicate logic

Today

- Syntax of propositional logic
- Natural Deduction
- Classical reasoning
- Semantics
- Equivalences
- Provability/Validity

Syntax & Informal Semantics

Syntax:

$$P ::= a \mid P \wedge P \mid P \vee P \mid P \to P \mid \neg P$$

Lower-case letters are atoms: p, q, r, etc.

Upper-case letters are (meta-)variables: P, Q, R, etc.

Two special atoms:

- $\top$ which stands for True
- $\bot$ which stands for False

We also introduced four connectives:

- $P \wedge Q$: we have a proof of both P and Q
- $P \vee Q$: we have a proof of at least one of P and Q
- $P \to Q$: if we have a proof of P then we have a proof of Q
- $\neg P$: stands for $P \to \bot$

Syntax

Example of propositions:

- "if x is a number then it is even or odd"
  - atom p: "x is a number"
  - atom q: "x is even"
  - atom r: "x is odd"
  - $p \to q \vee r$
- "if x is even then it is not odd"
  - atom p: "x is even"
  - atom q: "x is odd"
  - $p \to \neg q$
- "if $a = b$ and $b = c$ then $a = c$"
  - atom p: "$a = b$"
  - atom q: "$b = c$"
  - atom r: "$a = c$"
  - $(p \wedge q) \to r$
  - or equivalently: $p \to q \to r$

Precedence & Associativity

Precedence: in decreasing order of precedence $\neg, \wedge, \vee, \rightarrow$.

For example:

- $\neg P \vee Q$ means $(\neg P) \vee Q$
- $P \wedge Q \vee R$ means $(P \wedge Q) \vee R$
- $P \wedge Q \rightarrow Q \wedge P$ means $(P \wedge Q) \rightarrow (Q \wedge P)$

Associativity: all operators are right associative

For example:

- $P \vee Q \vee R$ means $P \vee (Q \vee R)$.
- $P \wedge Q \wedge R$ means $P \wedge (Q \wedge R)$.
- $P \rightarrow Q \rightarrow R$ means $P \rightarrow (Q \rightarrow R)$.

However use parentheses around compound formulas for clarity.

Constructive Natural Deduction

Constructive Natural Deduction rules:

$$\frac{\bot}{A}\ [\bot\, E] \qquad \frac{}{\top}\ [\top I] \qquad \frac{\begin{array}{c}\overline{A}^{\,1}\\ \vdots\\ B\end{array}}{A \to B}\ 1\ [\to I] \qquad \frac{A \to B \quad A}{B}\ [\to E]$$

$$\frac{\begin{array}{c}\overline{A}^{\,1}\\ \vdots\\ \bot\end{array}}{\neg A}\ 1\ [\neg I] \qquad \frac{\neg A \quad A}{\bot}\ [\neg E]$$

$$\frac{A}{A \vee B}\ [\vee I_L] \qquad \frac{A}{B \vee A}\ [\vee I_R] \qquad \frac{A \vee B \quad A \to C \quad B \to C}{C}\ [\vee E]$$

$$\frac{A \quad B}{A \wedge B}\ [\wedge I] \qquad \frac{A \wedge B}{B}\ [\wedge E_R] \qquad \frac{A \wedge B}{A}\ [\wedge E_L]$$

Classical Reasoning

Classical Natural Deduction includes all the Constructive Natural Deduction rules, plus:

$$\frac{}{A \vee \neg A}\ [LEM] \qquad \frac{\neg\neg A}{A}\ [DNE]$$

Semantics

A **valuation** ϕ assigns **T** or **F** with each atom

A valuation is **extended** to all formulas as follows:

- $\phi(\top) = \mathbf{T}$
- $\phi(\bot) = \mathbf{F}$
- $\phi(A \vee B) = \mathbf{T}$ iff either $\phi(A) = \mathbf{T}$ or $\phi(B) = \mathbf{T}$
- $\phi(A \wedge B) = \mathbf{T}$ iff both $\phi(A) = \mathbf{T}$ and $\phi(B) = \mathbf{T}$
- $\phi(A \to B) = \mathbf{T}$ iff $\phi(B) = \mathbf{T}$ whenever $\phi(A) = \mathbf{T}$
- $\phi(\neg A) = \mathbf{T}$ iff $\phi(A) = \mathbf{F}$

Satisfaction & validity:

- Given a valuation ϕ, we say that ϕ **satisfies** A if $\phi(A) = \mathbf{T}$
- A is **satisfiable** if there exists a valuation ϕ on atomic propositions such that $\phi(A) = \mathbf{T}$
- A is **valid** if $\phi(A) = \mathbf{T}$ for all possible valuations ϕ

Truth Tables

We can use **truth tables** to check whether propositions are valid:

A	B	$A \vee B$
T	**T**	**T**
T	**F**	**T**
F	**T**	**T**
F	**F**	**F**

A	B	$A \wedge B$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

P	Q	$P \to Q$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

A	$\neg A$
T	**F**
F	**T**

A proposition is (semantically) valid if the last column in its truth table only contains **T**

Validity

These techniques can be used to prove the validity of propositions:

- a **Natural Deduction** proof (syntactic validity)
- a **truth table** with only **T** in the last column (semantical validity)

We saw that:

- a formula A is provable in **Natural Deduction**
- **iff** A is **semantically valid**

This is true about the classical versions of these deduction systems

Logical equivalences

Let $A \leftrightarrow B$ be defined as $(A \rightarrow B) \wedge (B \rightarrow A)$

- it means that A and B are logically equivalent
- this is called a "bi-implication"
- read as "A if and only if B"

We will now prove:

- Distributivity of $\wedge$ over $\vee$:
  $(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$
- Double negation elimination as an equivalence: $\neg\neg A \leftrightarrow A$

You can also try proving the distributivity of $\vee$ over $\wedge$:
$(A \vee (B \wedge C)) \leftrightarrow ((A \vee B) \wedge (A \vee C))$

Provability/Validity

Provide a constructive Natural Deduction proof of the following equivalence: $(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$

Left-to-right implication:

$$\dfrac{\dfrac{\dfrac{}{A \wedge (B \vee C)}\,1}{B \vee C}\,[\wedge E_R] \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A \wedge (B \vee C)}\,1}{A}\,[\wedge E_L] \quad \dfrac{}{B}\,2}{A \wedge B}\,[\wedge I]}{(A \wedge B) \vee (A \wedge C)}\,[\vee I_L]}{B \to (A \wedge B) \vee (A \wedge C)}\,2\,[\to I] \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A \wedge (B \vee C)}\,1}{A}\,[\wedge E_L] \quad \dfrac{}{C}\,3}{A \wedge C}\,[\wedge I]}{(A \wedge B) \vee (A \wedge C)}\,[\vee I_R]}{C \to (A \wedge B) \vee (A \wedge C)}\,3\,[\to I]$$

$$\dfrac{(A \wedge B) \vee (A \wedge C)}{(A \wedge (B \vee C)) \to ((A \wedge B) \vee (A \wedge C))}\,1\,[\to I] \qquad [\vee E]$$

Provability/Validity

Right-to-left implication:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(A \wedge B) \vee (A \wedge C)}\,1 \quad \Pi_1 \quad \Pi_2}{A}\,[\vee E] \quad \dfrac{\dfrac{\dfrac{}{(A \wedge B) \vee (A \wedge C)}\,1 \quad \Pi_3 \quad \Pi_4}{B \vee C}\,[\vee E]}{A \wedge (B \vee C)}\,[\wedge I]}{((A \wedge B) \vee (A \wedge C)) \to (A \wedge (B \vee C))}\,1\,[\to I]$$

where Π_1 is:

$$\dfrac{\dfrac{\dfrac{}{A \wedge B}\,2}{A}\,[\wedge E_L]}{(A \wedge B) \to A}\,2\,[\to I]$$

where Π_2 is:

$$\dfrac{\dfrac{\dfrac{}{A \wedge C}\,3}{A}\,[\wedge E_L]}{(A \wedge C) \to A}\,3\,[\to I]$$

where Π_3 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \wedge B}\,4}{B}\,[\wedge E_R]}{B \vee C}\,[\vee I_L]}{(A \wedge B) \to (B \vee C)}\,4\,[\to I]$$

where Π_4 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \wedge C}\,5}{C}\,[\wedge E_R]}{B \vee C}\,[\vee I_R]}{(A \wedge C) \to (B \vee C)}\,5\,[\to I]$$

Provability/Validity

Prove that $(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$ is valid using a truth table

A	B	C	$B \vee C$	$A \wedge (B \vee C)$	$A \wedge B$	$A \wedge C$	$(A \wedge B) \vee (A \wedge C)$
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	**T**	**F**	**T**	**T**	**T**	**F**	**T**
T	**F**	**T**	**T**	**T**	**F**	**T**	**T**
T	**F**	**F**	**F**	**F**	**F**	**F**	**F**
F	**T**	**T**	**T**	**F**	**F**	**F**	**F**
F	**T**	**F**	**T**	**F**	**F**	**F**	**F**
F	**F**	**T**	**T**	**F**	**F**	**F**	**F**
F	**F**	**F**	**F**	**F**	**F**	**F**	**F**

The 5th and last columns are identical, so the two formulas are equivalent

Provability/Validity

Provide a classical Natural Deduction proof of the following equivalence: $\neg\neg A \leftrightarrow A$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\neg\neg A}\,1}{A}\,[DNE]}{\neg\neg A \to A}\,1\,[\to I] \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\neg A}\,3 \quad \dfrac{}{A}\,2}{\bot}\,[\neg E]}{\neg\neg A}\,3\,[\neg I]}{A \to \neg\neg A}\,2\,[\to I]}{\neg\neg A \leftrightarrow A}\,[\wedge I]$$

Provability/Validity

Prove that $\neg\neg A \leftrightarrow A$ is valid using a truth table

A	$\neg A$	$\neg\neg A$
T	**F**	**T**
F	**T**	**F**

The 1st and last columns are identical, so the two formulas are equivalent

Conclusion

What did we cover today?

- Syntax of propositional logic
- Natural Deduction
- Classical reasoning
- Semantics
- Equivalences
- Provability/Validity

Next time?

- Predicate logic (syntax)

Mathematical and Logical Foundations of Computer Science

Lecture 11 - Predicate Logic (Syntax)

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- Symbolic logic
- Propositional logic
- **Predicate logic**

Today

- Syntax of Predicate Logic

Further reading:

- Chapter 7 of
  http://leanprover.github.io/logic_and_proof/

Recap: Propositional Logic

Propositions: Facts (that can in principle be true or false)

- 2 is an even number
- 2 is an odd number
- $\mathcal{P} = \mathcal{NP}$
- Mind the gap! (not a proposition)

Grammar: $P ::= a \mid P \wedge P \mid P \vee P \mid P \to P \mid \neg P$
where a ranges over **atomic propositions**.

Two special atoms: $\top$ stands for True, $\bot$ stands for False

Four connectives:

- $P \wedge Q$: we have a proof of both P and Q
- $P \vee Q$: we have a proof of at least one of P and Q
- $P \to Q$: if we have a proof of P then we have a proof of Q
- $\neg P$: stands for $P \to \bot$

Recap: Proofs

Natural Deduction

introduction/elimination rules

natural proofs

$$\dfrac{\begin{array}{c}\overline{A}^{\,1}\\ \vdots\\ B\end{array}}{A \to B}\ 1\ [\to I]$$

Expressiveness of Propositional Logic

Famous derivation in logic:

- All men are mortal
- Socrates is a man
- Therefore, Socrates is mortal

Can we express this in propositional logic?

Another example:

- Every even natural number is not odd
- x is even
- x is not odd

Can we express this in propositional logic?

Beyond Propositional Logic

Propositional logic allows us to state facts

- does not allow stating **properties of** and **relations between** "objects"
- e.g., the property of numbers of being even, or odd

This brings us to a **richer** logic called **predicate logic**

- **contains** propositional logic
- also known as **first-order logic**
- Predicate logic allows us to reason about members of a (non-empty) domain

Beyond Propositional Logic

For example, the argument:

- All men are mortal
- Socrates is a man
- Therefore, Socrates is mortal

includes the following components:

- Domain = Men
- Socrates is one member of this domain
- Predicates are “being a man” and “being mortal”

Beyond Propositional Logic

Another example: consider a database with 3 tables

Student	
sid	**name**
0	Alice
1	Bob

Module	
mid	**name**
0	Math
1	OOP

Enroll	
sid	**mid**
0	0
1	1

These 3 tables can be seen as 3 relations:

- $Student(sid, name)$: predicate $Student$ relates student ids and names
- $Module(mid, name)$: predicate $Module$ relates module ids and names
- $Enroll(sid, mid)$: predicate $Enroll$ relates student and module ids

Domain = all possible values

A formula can be seen as a query

For example: find the Students x enrolled in the Math module

- $\exists y.\exists z.Student(y, x) \wedge Module(z, \mathsf{Math}) \wedge Enroll(y, z)$

Key ingredients of Predicate Logic

The key ingredients of predicate logic are

- predicates, quantifiers, variables, functions, and constants

Famous derivation in logic:

- All men are mortal
- Socrates is a man
- Therefore, Socrates is mortal

We can write this argument as $\forall x.(p(x) \rightarrow q(x)), p(s) \vdash q(s)$

- **Predicates**:
  - $p(x)$ which states that x is a man
  - $q(x)$ which states that x is mortal
- **Quantifier**: The "for all" symbol $\forall$
- **Variable**: x to denote an element of the domain
- **Constant**: s which stands for Socrates

Key ingredients of Predicate Logic

Domain (also called universe)

- Non-empty set of objects/entities (individuals) to reason about
- Example: set of 1st year students

Variables

- Symbols to represent (as yet unknown) objects in the domain
- Usually denoted by $x, y, z, \ldots$
- Similar to variables from programming languages

Quantifiers

- **universal** quantifier
  $\forall x. \cdots$: "for all elements x of the domain"
- **existential** quantifier
  $\exists x. \cdots$: "there exists an element x of the domain such that"
- quantify over elements of the domain
- **precedence**: lower than the other connectives

Key ingredients of Predicate Logic

Functions

- Build an element of the domain from elements of the domain
- Usually denoted by $f, g, h, \ldots$
- Different functions can have different numbers of arguments
- The number of arguments of a function is called its **arity**
- A function symbol of arity 1 can only be applied to 1 argument, A function symbol of arity 2 can only be applied to 2 arguments, etc.
- **Notation**: We sometimes write f^k when we want to indicate that the function symbol f has arity k

Constants

- Specific objects in the domain
- Functions of arity 0
- Usually denoted by $a, b, c, \ldots$

Key ingredients of Predicate Logic

Let the domain be $\mathbb{N}$.

Provide examples of function symbols, along with their arities

- $0, 1, 2, \ldots$ are constant symbols (nullary function symbols)
- `add`: the binary addition function
- `add`(m, n): addition applied to the two expressions m and n
- `square`: the unary square function
- `square`(m): square applied to the expression m

Key ingredients of Predicate Logic

Predicates

- Propositions are facts/statements, which may be true or false
- A predicate evaluates to true/false depending on its arguments
- Predicates can be seen as functions from elements of the domain to propositions
- **Example**: $p(x)$ means "predicate p is true for variable x"
- **Example**: $p(a)$ means "predicate p is true for constant a"

Examples of formulas in predicate logic

- $\forall x.(p(x) \wedge q(x))$
  - for all x it is true that $p(x)$ and $q(x)$
- $(\forall x.p(x)) \rightarrow \neg \forall x.q(x)$
  - if $p(x)$ is true for all x, then $q(x)$ is not true for all x
- $\exists x.(p(x) \vee \neg q(x))$
  - there is some x for which $p(x)$ is true or $q(x)$ is not true

More examples in predicate calculus

Domain is cars, and we have 3 predicate symbols

- $f(x) =$ "x is fast"
- $r(x) =$ "x is red"
- $p(x) =$ "x is purple"

How to express the following sentences in predicate logic?

- All cars are fast: $\forall x.f(x)$
- All red cars are fast: $\forall x.r(x) \rightarrow f(x)$
- Some red cars are fast: $\exists x.r(x) \wedge f(x)$
  - Wrong answer: $\exists x.r(x) \rightarrow f(x)$
- There are no red cars: $\neg\exists x.r(x)$
  - Alternative answer: $\forall x.\neg r(x)$
- No fast cars are purple: $\neg\exists x.f(x) \wedge p(x)$
  - Alternative answer: $\forall x.f(x) \rightarrow \neg p(x)$

Connections between $\exists$ and $\forall$

To disprove a **"for all"** proposition, we need to find an x for which the predicate is false

- $\neg(\forall x.p(x))$ is the same as $\exists x.\neg p(x)$

To disprove a **"there exists"** proposition, we need to show that the predicate is false for all x

- $\neg(\exists x.p(x))$ is the same as $\forall x.\neg p(x)$

Arity of predicates

The **arity** of a predicate is the number of arguments it takes

Unary predicates (arity 1) represent facts about individuals

- $p(x) =$ "x is prime"

Binary predicates (arity 2) represent relationships between individuals, i.e., they represent relations

- Example: $m(a, b) =$ "a is married to b"
- Doesn't have to be symmetric!
- Example: $l(a, b) =$ "a likes b"

What are **nullary** predicates (arity 0)?

- Atomic propositions!

Notation: We sometimes write p^k when we want to indicate that the predicate symbol p has arity k

Syntax

The syntax of predicate logic is defined by the following grammar:

$$\begin{array}{lll} t & ::= & x \mid f(t, \ldots, t) \\ P & ::= & p(t, \ldots, t) \mid \neg P \mid P \wedge P \mid P \vee P \mid P \rightarrow P \mid \forall x.P \mid \exists x.P \end{array}$$

where:

- x ranges over variables
- f ranges over function symbols
- $f(t_1, \ldots, t_n)$ is a well-formed term only if f has arity n
- p ranges over predicate symbols
- $p(t_1, \ldots, t_n)$ is a well-formed formula only if p has arity n

The pair of a collection of function symbols, and a collection of predicate symbols, along with their arities, is called a **signature**.

The scope of a quantifier extends as far right as possible. E.g., $P \wedge \forall x.p(x) \vee q(x)$ is read as $P \wedge \forall x.(p(x) \vee q(x))$

Examples

Consider the following domain and signature:

- Domain: $\mathbb{N}$
- Functions: $0, 1, 2, \ldots$ (arity 0); $+$ (arity 2)
- Predicates: $\texttt{prime}$, $\texttt{even}$, $\texttt{odd}$ (arity 1); $=$, $>$, $\geqslant$ (arity 2)

Express the following sentences in predicate logic

- All prime numbers are either 2 or odd.
  $\forall x.\texttt{prime}(x) \rightarrow x = 2 \vee \texttt{odd}(x)$
- Every even number is equal to the sum of two primes.
  $\forall x.\texttt{even}(x) \rightarrow \exists y.\exists z.\texttt{prime}(y) \wedge \texttt{prime}(z) \wedge x = y + z$
- There is no number greater than all numbers.
  $\neg\exists x.\forall y.x > y$
- All numbers have a number greater than them.
  $\forall x.\exists y.y > x$

Natural Deduction rules for $\forall$ and $\exists$?

Propositional logic: Each connective has two inference rules

- One for introduction
- One for elimination

Introduction and elimination rules for $\forall$ and $\exists$?

$$\frac{?}{\forall x.P}\ [\forall I] \qquad \frac{\forall y.P}{?}\ [\forall E]$$

$$\frac{?}{\exists x.P}\ [\exists I] \qquad \frac{\exists y.P}{?}\ [\exists E]$$

Conclusion

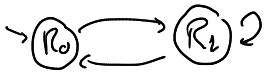
What did we cover today?

- Predicate logic (syntax)

Next time?

- Predicate logic (Natural Deduction)

- A robot can move between 2 rooms R_0 and R_1 as follows:



i.e., (a) it starts in room R_0
(b) when in room R_0 it can only move to room R_1
(c) when in room R_1 it can move to R_1 or R_0

- To model this system, let us consider the following atomic propositions: $p_0, p_1, p_2, \dots$
such that p_i means that the robot is in room R_0 at step i (i.e., after i moves)
and $\neg p_i$ means that the robot is in room R_1 at step i
- we can model the initial state of the robot as follows: p_0 , i.e, the robot is initially in room R_0 (a)
- we can model a transition as follows: $(p_i \to \neg p_{i+1}) \wedge (\neg p_i \to (\neg p_{i+1} \vee p_i))$, i.e., (b) $\wedge$ (c)
- we can capture two moves as follows: $p_0 \wedge (p_0 \to \neg p_1) \wedge (\neg p_0 \to (\neg p_1 \vee p_2)) \wedge (p_1 \to \neg p_2) \wedge (\neg p_1 \to (p_2 \vee p_2))$
initial position, 1st move, 2nd move
- Could we prove that when in room R_1, the robot will be in room R_1 next? ← call it P
No : when in R_1, the robot can move to R_0
How do we formally prove that this property fails? we prove its negation, i.e $\neg(\neg p_i \to \neg p_{i+1})$
- let's show that this fails after 2 steps, i.e: $p_0 \wedge (p_0 \to \neg p_1) \wedge (\neg p_0 \to (\neg p_1 \vee p_1)) \wedge (p_1 \to \neg p_2) \wedge (p_1 \to (\neg p_2 \vee p_2)) \wedge \neg(\neg p_1 \to \neg p_2)$
- We'll show this using a SAT solver: (1) convert the formula to a CNF using logical equivalences
(2) use DPLL to check whether it is satisfiable

(1) $p_0 \wedge (p_0 \to \neg p_1) \wedge (\neg p_0 \to (\neg p_1 \vee p_1)) \wedge (p_1 \to \neg p_2) \wedge (\neg p_1 \to (\neg p_2 \vee p_2)) \wedge \neg(\neg p_1 \to \neg p_2)$
$\leftrightarrow p_0 \wedge (\neg p_0 \vee \neg p_1) \wedge (\neg\neg p_0 \vee \neg p_1 \vee p_1) \wedge (\neg p_1 \vee \neg p_2) \wedge (\neg\neg p_1 \vee \neg p_2 \vee p_2) \wedge \neg(\neg\neg p_1 \vee \neg p_2)$ - elim →
$\leftrightarrow p_0 \wedge (\neg p_0 \vee \neg p_1) \wedge (\neg\neg p_0 \vee \neg p_1 \vee p_1) \wedge (\neg p_1 \vee \neg p_2) \wedge (\neg\neg p_1 \vee \neg p_2 \vee p_2) \wedge \neg\neg\neg p_1 \wedge \neg\neg p_2$ - de-Morg.
$\leftrightarrow p_0 \wedge (\neg p_0 \vee \neg p_1) \wedge (p_0 \vee \neg p_1 \vee p_1) \wedge (\neg p_1 \vee \neg p_2) \wedge (p_1 \vee \neg p_2 \vee p_2) \wedge \neg p_1 \wedge p_2$ - ¬¬ elim
CNF →

(2) use DPLL: $p_0 \wedge (\neg p_0 \vee \neg p_1) \wedge (p_0 \vee \neg p_1 \vee p_2) \wedge (\neg p_1 \vee \neg p_2) \wedge (p_1 \vee \neg p_2 \vee p_2) \wedge \neg p_1 \wedge p_2$
$p_0 = T$ $p_1 = F$ $p_2 = T$: the formula is satisfiable
This gives us a run of our robot: $R_0 \to R_1 \to R_0$, which shows that the property P does not hold, i.e, we obtained a counterexample.

Mathematical and Logical Foundations of Computer Science

Lecture 12 - Predicate Logic (Natural Deduction Proofs)

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- Symbolic logic
- Propositional logic
- **Predicate logic**

Today

- Natural Deduction proofs for Predicate Logic
- $\forall/\exists$ rules
- substitution

Further reading:

- Chapter 8 of
  http://leanprover.github.io/logic_and_proof/

Recap: Beyond Propositional Logic

Famous derivation in logic:

- All men are mortal
- Socrates is a man
- Therefore, Socrates is mortal

Cannot be expressed in propositional logic

We introduced:

- predicates, quantifiers, variables, functions, and constants

We can write this argument as $\forall x.(p(x) \rightarrow q(x)), p(s) \vdash q(s)$

- **Domain**: people
- **Predicates**: $p(x) =$ "x is a man"; $q(x) =$ "x is mortal"
- **Quantifier**: The "for all" symbol $\forall$
- **Variable**: x to denote an element of the domain
- **Constant**: s which stands for Socrates

Recap: Syntax

The syntax of predicate logic is defined by the following grammar:

$$\begin{array}{lll} t & ::= & x \mid f(t, \ldots, t) \\ P & ::= & p(t, \ldots, t) \mid \neg P \mid P \wedge P \mid P \vee P \mid P \rightarrow P \mid \forall x.P \mid \exists x.P \end{array}$$

where:

- x ranges over variables
- f ranges over function symbols
- $f(t_1, \ldots, t_n)$ is a well-formed term only if f has arity n
- p ranges over predicate symbols
- $p(t_1, \ldots, t_n)$ is a well-formed formula only if p has arity n

The pair of a collection of function symbols, and a collection of predicate symbols, along with their arities, is called a **signature**.

The scope of a quantifier extends as far right as possible. E.g., $P \wedge \forall x.p(x) \vee q(x)$ is read as $P \wedge \forall x.(p(x) \vee q(x))$

Recap: Examples

Consider the following domain and signature:

- Domain: $\mathbb{N}$
- Functions: $0, 1, 2, \ldots$ (arity 0); $+$ (arity 2)
- Predicates: $\texttt{prime}$, $\texttt{even}$, $\texttt{odd}$ (arity 1); $=$, $>$, $\geqslant$ (arity 2)

Express the following sentences in predicate logic

- All prime numbers are either 2 or odd.
  $\forall x.\texttt{prime}(x) \rightarrow x = 2 \vee \texttt{odd}(x)$
- Every even number is equal to the sum of two primes.
  $\forall x.\texttt{even}(x) \rightarrow \exists y.\exists z.\texttt{prime}(y) \wedge \texttt{prime}(z) \wedge x = y + z$
- There is no number greater than all numbers.
  $\neg\exists x.\forall y.x \geqslant y$
- All numbers have a number greater than them.
  $\forall x.\exists y.y > x$

One more example (from the book – section 7.6.2)

Domain is people, and we have 6 predicates

politician(x) rich(x) crazy(x) trusts(x, y) knows(x, y) related-to(x, y)

Express the following sentences in predicate logic

- Nobody trusts a politician.
  $\neg\exists x.\exists y.\text{politician}(y) \wedge \text{trusts}(x, y)$
- Anyone who trusts a politician is crazy.
  $\forall x.(\exists y.\text{politician}(y) \wedge \text{trusts}(x, y)) \rightarrow \text{crazy}(x)$
- Everyone knows someone who is related to a politician.
  $\forall x.\exists y.\text{knows}(x, y) \wedge \exists z.\text{politician}(z) \wedge \text{related-to}(y, z)$
- Everyone who is rich is either a politician or knows a politician.
  $\forall x.\text{rich}(x) \rightarrow \text{politician}(x) \vee \exists y.\text{knows}(x, y) \wedge \text{politician}(y)$

Inference rules for $\forall$ and $\exists$?

Propositional logic: Each connective has at least 2 inference rules

- At least 1 for introduction
- At least 1 for elimination

Introduction and elimination rules for $\forall$ and $\exists$?

$$\frac{?}{\forall y.P}\ [\forall I] \qquad \frac{\forall x.P}{?}\ [\forall E]$$

$$\frac{?}{\exists y.P}\ [\exists I] \qquad \frac{\exists x.P}{?}\ [\exists E]$$

Free & Bound Variables

Free variables and **Bound** variables:

Bound variables:

- Consider the formula $\forall x.\text{even}(x) \vee \text{odd}(x)$
  Here the variable x is **bound** by the quantifier $\forall$
- $\forall x.\text{even}(x) \vee \text{odd}(x)$ is considered the same as $\forall y.\text{even}(y) \vee \text{odd}(y)$
  Renaming a **bound** variable **doesn't** change the meaning!

Free variables:

- Consider the formula $\forall y.x \leqslant y$
- y is a **bound** variable and x is a **free** variable
- variables are **free** if they are not bound
- $\forall y.x \leqslant y$ is the **same** as $\forall z.x \leqslant z$
- $\forall y.x \leqslant y$ is **not the same** as $\forall y.w \leqslant y$
- Renaming a **free** variable **changes** the meaning!

Free & Bound Variables

The **scope** of a quantified formula of the form $\forall x.P$ or $\exists x.P$ is P. The quantifier are said to **bind** x.

Bound variables: a variable x occurs bound in a formula, if it occurs in the scope of a quantifier quantifying x

Free variables: a variable x occurs free in a formula, if it does not occur in the scope of a quantifier quantifying x

The set of variables occurring free/bound in a terms and formulas is recursively computed as follows:

$$
\begin{array}{lcl}
\mathtt{fv}(x) & = & \{x\} \\
\mathtt{fv}(f(t_1,\ldots,t_n)) & = & \mathtt{fv}(t_1) \cup \ldots \cup \mathtt{fv}(t_n) \\
\mathtt{fv}(p(t_1,\ldots,t_n)) & = & \mathtt{fv}(t_1) \cup \ldots \cup \mathtt{fv}(t_n) \\
\hline
\mathtt{fv}(\neg P) & = & \mathtt{fv}(P) \\
\mathtt{fv}(P_1 \wedge P_2) & = & \mathtt{fv}(P_1) \cup \mathtt{fv}(P_2) \\
\mathtt{fv}(P_1 \vee P_2) & = & \mathtt{fv}(P_1) \cup \mathtt{fv}(P_2) \\
\mathtt{fv}(P_1 \rightarrow P_2) & = & \mathtt{fv}(P_1) \cup \mathtt{fv}(P_2) \\
\hline
\mathtt{fv}(\forall x.P) & = & \mathtt{fv}(P) \backslash \{x\} \\
\mathtt{fv}(\exists x.P) & = & \mathtt{fv}(P) \backslash \{x\}
\end{array}
$$

$$
\begin{array}{lcl}
\mathtt{bv}(p(t_1,\ldots,t_n)) & = & \emptyset \\
\hline
\mathtt{bv}(\neg P) & = & \mathtt{bv}(P) \\
\mathtt{bv}(P_1 \wedge P_2) & = & \mathtt{bv}(P_1) \cup \mathtt{bv}(P_2) \\
\mathtt{bv}(P_1 \vee P_2) & = & \mathtt{bv}(P_1) \cup \mathtt{bv}(P_2) \\
\mathtt{bv}(P_1 \rightarrow P_2) & = & \mathtt{bv}(P_1) \cup \mathtt{bv}(P_2) \\
\hline
\mathtt{bv}(\forall x.P) & = & \mathtt{bv}(P) \cup \{x\} \\
\mathtt{bv}(\exists x.P) & = & \mathtt{bv}(P) \cup \{x\}
\end{array}
$$

Free & Bound Variables

What are the free variables of the following formulas

- $P_1 = (\texttt{odd}(x) \wedge \exists y.y < x \wedge \texttt{odd}(y))$
  $\texttt{fv}(P_1) = \{x\}$
- $P_2 = (\texttt{odd}(x) \wedge x > y \wedge \exists y.y < x \wedge \texttt{odd}(y))$
  $\texttt{fv}(P_2) = \{x, y\}$
- $P_3 = (\forall x.\texttt{odd}(x) \wedge x > y \wedge \exists y.y < x \wedge \texttt{odd}(y))$
  $\texttt{fv}(P_3) = \{y\}$

Note: In $(\texttt{odd}(x) \wedge x > y \wedge \exists y.y < x \wedge \texttt{odd}(y))$ the green occurrence of y is **not** the same variable as the red occurrence of y.

The formula $(\texttt{odd}(x) \wedge x > y \wedge \exists y.y < x \wedge \texttt{odd}(y))$ is considered the same as $(\texttt{odd}(x) \wedge x > y \wedge \exists z.z < x \wedge \texttt{odd}(z))$

Inference rules for $\forall$ and $\exists$?

Propositional logic: Each connective has at least 2 inference rules

- At least 1 for introduction
- At least 1 for elimination

Introduction and elimination rules for $\forall$ and $\exists$?

$$\frac{?}{\forall y.P}\ [\forall I] \qquad \frac{\forall x.P}{?}\ [\forall E]$$

$$\frac{?}{\exists y.P}\ [\exists I] \qquad \frac{\exists x.P}{?}\ [\exists E]$$

WARNING ⚠

Trickier than inference rules from propositional logic!

We need to be careful with free and bound variables!

Inference Rule for "for all elimination" – 1st attempt

$$\frac{\forall x.P}{?}\ [\forall E]$$

What can we conclude from the fact that P is true for all x?

Predicate P is true for all elements x of the domain

- For any element of the domain t, we can deduce that P is true where x is replaced by t is true
- This "replacing" operation is a **substitution** operation as seen in lecture 2.
- However, we now have to be careful with free/bound variables.

Substitution

Substitution is defined recursively on terms and formulas:

$P[x\backslash t]$ substitute all the free occurrences of x in P with t.

1st attempt (WRONG)

$$
\begin{array}{lcl}
x[x\backslash t] & = & t \\
x[y\backslash t] & = & x \\
(f(t_1,\ldots,t_n))[x\backslash t] & = & f(t_1[x\backslash t],\ldots,t_n[x\backslash t]) \\
(p(t_1,\ldots,t_n))[x\backslash t] & = & p(t_1[x\backslash t],\ldots,t_n[x\backslash t]) \\
\hline
(\neg P)[x\backslash t] & = & \neg P[x\backslash t] \\
(P_1 \wedge P_2)[x\backslash t] & = & P_1[x\backslash t] \wedge P_2[x\backslash t] \\
(P_1 \vee P_2)[x\backslash t] & = & P_1[x\backslash t] \vee P_2[x\backslash t] \\
(P_1 \rightarrow P_2)[x\backslash t] & = & P_1[x\backslash t] \rightarrow P_2[x\backslash t] \\
\hline
(\forall x.P)[x\backslash t] & = & \forall x.P \\
(\exists x.P)[x\backslash t] & = & \exists x.P \\
(\forall y.P)[x\backslash t] & = & \forall y.P[x\backslash t] \\
(\exists y.P)[x\backslash t] & = & \exists y.P[x\backslash t]
\end{array}
$$

Why is this wrong? $(\forall y.y > x)[x\backslash y]$ would return $\forall y.y > y$, where the free y is now bound! The free y got **captured**! The red occurrences of y stand for different variables than the green ones.

Substitution

Substitution is defined recursively on terms and formulas:

$P[x\backslash t]$ substitute all the free occurrences of x in P with t.

2nd attempt (CORRECT)

$$
\begin{array}{lcl}
x[x\backslash t] & = & t \\
x[y\backslash t] & = & x \\
(f(t_1,\ldots,t_n))[x\backslash t] & = & f(t_1[x\backslash t],\ldots,t_n[x\backslash t]) \\
(p(t_1,\ldots,t_n))[x\backslash t] & = & p(t_1[x\backslash t],\ldots,t_n[x\backslash t]) \\
\hline
(\neg P)[x\backslash t] & = & \neg P[x\backslash t] \\
(P_1 \wedge P_2)[x\backslash t] & = & P_1[x\backslash t] \wedge P_2[x\backslash t] \\
(P_1 \vee P_2)[x\backslash t] & = & P_1[x\backslash t] \vee P_2[x\backslash t] \\
(P_1 \to P_2)[x\backslash t] & = & P_1[x\backslash t] \to P_2[x\backslash t] \\
\hline
(\forall x.P)[x\backslash t] & = & \forall x.P \\
(\exists x.P)[x\backslash t] & = & \exists x.P \\
(\forall y.P)[x\backslash t] & = & \forall y.P[x\backslash t], \text{ if } y \notin \mathtt{fv}(t) \\
(\exists y.P)[x\backslash t] & = & \exists y.P[x\backslash t], \text{ if } y \notin \mathtt{fv}(t)
\end{array}
$$

The additional **conditions** ensure that **free variables do not get captured**.

These conditions can always be met by silently renaming bound variables before substituting.

Inference Rule for "for all elimination" – 2nd attempt

The correct rule is:

$$\frac{\forall x.P}{P[x\backslash t]}\ [\forall E]$$

Condition: $\mathtt{fv}(t)$ must not clash with any bound variables of P

Example: consider the formula $\forall x.\exists y.y > x$

- True over domain of natural numbers
- P is $\exists y.y > x$
- Let t be y
- This condition guarantees that we can do the substitution
- Substituting x with y without renaming bound variables would give the wrong answer (see previous slide)
- Therefore, we first rename bound variables that clash with $\mathtt{fv}(t)$, i.e., with y: $\exists z.z > x$
- Then, we substitute: $\exists z.z > y$

Inference Rule for "for all introduction"

$$\frac{?}{\forall x.P}\ [\forall I]$$

When can we conclude P is true for all x?

If we have proved P for a "**general/representative/typical**" variable

$$\frac{P[x\backslash y]}{\forall x.P}\ [\forall I]$$

Condition: y must not be free in any not-yet-discharged hypothesis or in $\forall x.P$

What could go wrong without this condition?

Otherwise, given the assumption $y > 2$, we could derive $\forall x.x > 2$, which is clearly wrong.

Inference Rule for "exists introduction"

$$\frac{?}{\exists x.P}\ [\exists I]$$

When can we conclude P is true for some x?

If we have proved predicate P for an element of the domain

$$\frac{P[x\backslash t]}{\exists x.P}\ [\exists I]$$

Condition: $\mathtt{fv}(t)$ must not clash with $\mathtt{bv}(P)$

Example: Consider the predicate $P = (\forall y.y = x)$

- Without the substitution conditions $P[x\backslash y]$ would be true
- We could then deduce $\exists x.\forall y.y = x$, i.e., numbers are all equal to each other — obviously incorrect!
- The substitution conditions prevents such captures
- $[\exists I]$'s condition guarantees that the substitution conditions hold

Inference Rule for "exists elimination"

$$\frac{\exists x.P}{?}\ [\exists E]$$

What can we conclude from the fact that P is true for some x?

We know that it holds about some element of the domain, but we do not know which

$$\dfrac{\exists x.P \qquad \dfrac{\dfrac{}{P[x\backslash y]}\,1}{\vdots\ Q}}{Q}\ 1\ [\exists E]$$

Condition: y must not be free in Q or in not-yet-discharged hypotheses or in $\exists x.P$

This rule is similar to OR-elimination!

All four inference rules in one slide

$$\frac{P[x\backslash y]}{\forall x.P}\ [\forall I]$$

Condition: y must not be free in any not-yet-discharged hypothesis or in $\forall x.P$

$$\frac{\forall x.P}{P[x\backslash t]}\ [\forall E]$$

Condition: $\mathtt{fv}(t)$ must not clash with $\mathtt{bv}(P)$

$$\frac{P[x\backslash t]}{\exists x.P}\ [\exists I]$$

Condition: $\mathtt{fv}(t)$ must not clash with $\mathtt{bv}(P)$

$$\dfrac{\exists x.P \qquad \begin{array}{c}\dfrac{}{P[x\backslash y]}\,1\\ \vdots \\ Q\end{array}}{Q}\ 1\ [\exists E]$$

Condition: y must not be free in Q or in not-yet-discharged hypotheses or in $\exists x.P$

A simple proof

Prove that $(\forall z.p(z)) \rightarrow \forall x.p(x) \vee q(x)$

We use backward reasoning

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall z.p(z)}\,1}{p(y)}\,[\forall E]}{p(y) \vee q(y)}\,[\vee I_L]}{\forall x.p(x) \vee q(x)}\,[\forall I]}{(\forall z.p(z)) \rightarrow \forall x.p(x) \vee q(x)}\,1\ [\rightarrow I]$$

Conditions:

- y does not occur free in not-yet-discharged hypotheses or in $\forall x.p(x) \vee q(x)$
- y does not clash with bound variables in $p(z)$

A simple proof

More generally, we can prove:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall z.P}\,1}{P[x\backslash y]}\,[\forall E]}{P[x\backslash y] \vee Q[x\backslash y]}\,[\vee I_L]}{\forall x.P \vee Q}\,[\forall I]}{(\forall z.P) \rightarrow \forall x.P \vee Q}\,1\ [\rightarrow I]$$

We assume that y does not occur in P or Q

Conclusion

What did we cover today?

- Natural Deduction proofs for Predicate Logic
- $\forall/\exists$ rules
- substitution

Next time?

- Natural Deduction proofs for Predicate Logic – continued

Mathematical and Logical Foundations of Computer Science

Lecture 13 - Predicate Logic (Natural Deduction Proofs – Continued)

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- Symbolic logic
- Propositional logic
- **Predicate logic**

Today

- Natural Deduction proofs for Predicate Logic
- side conditions

Further reading:

- Chapter 8 of
  http://leanprover.github.io/logic_and_proof/

Recap: Syntax

The syntax of predicate logic is defined by the following grammar:

$$\begin{array}{lll} t & ::= & x \mid f(t, \ldots, t) \\ P & ::= & p(t, \ldots, t) \mid \neg P \mid P \wedge P \mid P \vee P \mid P \rightarrow P \mid \forall x.P \mid \exists x.P \end{array}$$

where:

- x ranges over variables
- f ranges over function symbols
- $f(t_1, \ldots, t_n)$ is a well-formed term only if f has arity n
- p ranges over predicate symbols
- $p(t_1, \ldots, t_n)$ is a well-formed formula only if p has arity n

The pair of a collection of function symbols, and a collection of predicate symbols, along with their arities, is called a **signature**.

The scope of a quantifier extends as far right as possible. E.g., $P \wedge \forall x.p(x) \vee q(x)$ is read as $P \wedge \forall x.(p(x) \vee q(x))$

Recap: Substitution

Substitution is defined recursively on terms and formulas:

$P[x\backslash t]$ substitute all the free occurrences of x in P with t.

$$
\begin{array}{lcl}
x[x\backslash t] & = & t \\
x[y\backslash t] & = & x \\
(f(t_1,\ldots,t_n))[x\backslash t] & = & f(t_1[x\backslash t],\ldots,t_n[x\backslash t]) \\
(p(t_1,\ldots,t_n))[x\backslash t] & = & p(t_1[x\backslash t],\ldots,t_n[x\backslash t]) \\
\hline
(\neg P)[x\backslash t] & = & \neg P[x\backslash t] \\
(P_1 \wedge P_2)[x\backslash t] & = & P_1[x\backslash t] \wedge P_2[x\backslash t] \\
(P_1 \vee P_2)[x\backslash t] & = & P_1[x\backslash t] \vee P_2[x\backslash t] \\
(P_1 \to P_2)[x\backslash t] & = & P_1[x\backslash t] \to P_2[x\backslash t] \\
\hline
(\forall x.P)[x\backslash t] & = & \forall x.P \\
(\exists x.P)[x\backslash t] & = & \exists x.P \\
(\forall y.P)[x\backslash t] & = & \forall y.P[x\backslash t], \text{ if } y \notin \mathtt{fv}(t) \\
(\exists y.P)[x\backslash t] & = & \exists y.P[x\backslash t], \text{ if } y \notin \mathtt{fv}(t)
\end{array}
$$

The additional **conditions** ensure that **free variables do not get captured**.

These conditions can always be met by silently renaming bound variables before substituting.

Recap: $\forall$ & $\exists$ elimination and introduction rules

$$\frac{P[x\backslash y]}{\forall x.P}\ [\forall I]$$

Condition: y must not be free in any not-yet-discharged hypothesis or in $\forall x.P$

$$\frac{\forall x.P}{P[x\backslash t]}\ [\forall E]$$

Condition: $\texttt{fv}(t)$ must not clash with $\texttt{bv}(P)$

$$\frac{P[x\backslash t]}{\exists x.P}\ [\exists I]$$

Condition: $\texttt{fv}(t)$ must not clash with $\texttt{bv}(P)$

$$\frac{\exists x.P \qquad \begin{array}{c}\dfrac{}{P[x\backslash y]}\ 1\\ \vdots \\ Q\end{array}}{Q}\ 1\ [\exists E]$$

Condition: y must not be free in Q or in not-yet-discharged hypotheses or in $\exists x.P$

Inference Rule for "for all elimination"

$$\frac{\forall x.P}{P[x\backslash t]}\ [\forall E]$$

Condition: $\mathtt{fv}(t)$ must not clash with $\mathtt{bv}(P)$

Example: consider the formula $\forall x.\exists y.y > x$

- True over domain of natural numbers
- P is $\exists y.y > x$
- Let t be y
- This condition guarantees that we can do the substitution
- Substituting x with y without renaming bound variables would give the wrong answer
- Therefore, we first rename bound variables that clash with $\mathtt{fv}(t)$, i.e., with y: $\exists z.z > x$
- Then, we substitute: $\exists z.z > y$

Inference Rule for "for all elimination"

More precisely: Assume that from $\forall x.\exists y.y > x$, we want to derive a number greater than y.

We would use the following rule:

$$\frac{\forall x.\exists y.y > x}{(\exists y.y > x)[x \backslash y]} \; [\forall E]$$

However, without renaming the bound y, $P[x \backslash y]$ is undefined

Therefore, we rename the bound variable just before performing the substitution:

$$\frac{\forall x.\exists y.y > x}{\exists z.z > y} \; [\forall E]$$

Inference Rule for "for all introduction"

$$\frac{P[x\backslash y]}{\forall x.P}\ [\forall I]$$

We conclude P is true for all x if we have proved P for a "**general/representative/typical**" variable

Condition: y must not be free in any not-yet-discharged hypothesis or in $\forall x.P$

What could go wrong without this condition?

- Otherwise, given the assumption $x > 2$, we could derive $\forall x.x > 2$, which is clearly wrong.
- We could also derive $\forall x.\forall y.x > 0 \rightarrow y > 0$, which is also clearly wrong.

Inference Rule for "for all introduction"

More precisely: without this condition we would be able to derive

$$\dfrac{\dfrac{\dfrac{}{x > 2}\,1}{\forall x.x > 2}\,\cancel{[\forall I]}}{x > 2 \to \forall x.x > 2}\,1\,[\to I]$$

WARNING ⚠ Note that this is **not a correct use** of the $[\forall I]$ rule because x is free in $x > 2$, which is not-yet-discharged when the $[\forall I]$ rule is applied

However, it is okay for the variable to appear in an assumption that is discharged **above** the $[\forall I]$ rule:

$$\dfrac{\dfrac{\dfrac{}{x > 2}\,1}{x > 2 \to x > 2}\,1\,[\to I]}{\forall x.x > 2 \to x > 2}\,[\forall I]$$

Inference Rule for "for all introduction"

How can we make checking this condition more tractable?

Going backward, we must ensure such variables

- are not free in the hypotheses we have introduced and discharged at the time $[\forall I]$ is used,
- are not free in the universally quantified formula.

We record those hypotheses in a **context** as follows:

$$\dfrac{\dfrac{y > 2}{\forall x.x > 2}\ [\forall I]}{x > 2 \to \forall x.x > 2}\ 1\ [\to I]$$

Context:

- 1: $x > 2$

We cannot pick x as it occurs in our **context**
We must pick a "fresh" variable not free in the **context** or in $\forall x.x > 2$
We cannot finish this proof now

Inference Rule for "for all introduction"

Prove $\forall x.x > 2 \to x > 2$ backward using contexts

Here is a proof:

$$\dfrac{\dfrac{\dfrac{}{x > 2}\,1}{x > 2 \to x > 2}\,1\ [\to I]}{\forall x.x > 2 \to x > 2}\ [\forall I]$$

Context:

- 1: $x > 2$

We can pick any variable we want as the context is empty and our conclusion does not have any free variables

Inference Rule for "for all introduction"

What could happen if we could pick a variable free in the conclusion?

If we could pick a variable free in the conclusion, we could derive:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x > 0}\,1}{x > 0 \to x > 0}\,1\ [\to I]}{\forall y.x > 0 \to y > 0}\ \cancel{[\forall I]}}{\forall x.\forall y.x > 0 \to y > 0}\ [\forall I]$$

WARNING ⚠ Note that this is **not a correct use** of the $[\forall I]$ rule because x is free the conclusion $\forall y.x > 0 \to y > 0$

Inference Rule for "for all introduction"

The rule's condition forces us to pick a **different** variable:

$$\dfrac{\dfrac{\dfrac{\overline{y > 0}}{x > 0 \to y > 0}\,1\ [\to I]}{\forall y.x > 0 \to y > 0}\ [\forall I]}{\forall x.\forall y.x > 0 \to y > 0}\ [\forall I]$$

We cannot finish this proof now

Inference Rule for "exists introduction"

$$\frac{P[x\backslash t]}{\exists x.P}\ [\exists I]$$

We conclude P is true for some x if we have proved predicate P for an element of the domain

Condition: $\mathtt{fv}(t)$ must not clash with $\mathtt{bv}(P)$

Example: Consider the predicate $P = (\forall y.y = x)$

- Without the substitution conditions $P[x\backslash y]$ would be true
- We could then deduce $\exists x.\forall y.y = x$, i.e., numbers are all equal to each other — obviously incorrect!
- The substitution conditions prevents such captures
- $[\exists I]$'s condition guarantees that the substitution conditions hold

Inference Rule for "exists introduction"

As for "for all elimination", we rename the bound variable just before performing the substitution.

For example if we know that y is the smallest number:

$$\frac{\forall z.y \leqslant z}{\exists x.\forall y.x \leqslant y}\ [\exists I]$$

Inference Rule for "exists elimination"

$$\dfrac{\exists x.P \qquad \begin{array}{c}\dfrac{}{P[x\backslash y]}\,1\\ \vdots \\ Q\end{array}}{Q}\,1\ [\exists E]$$

From the fact that P is true for some x we know that it holds about some element of the domain, but we do not know which

Condition: y must not be free in Q or in not-yet-discharged hypotheses or in $\exists x.P$

This rule is similar to OR-elimination!

Inference Rule for "exists elimination"

What could go wrong without this condition?

Assume for the sake of this example that $x \leqslant y$ is defined as $\neg y < x$

Without the condition we could prove:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\exists x.\forall y.x \leqslant y}\,2 \qquad \dfrac{\dfrac{}{0 < z}\,1 \qquad \dfrac{\dfrac{}{\forall y.z \leqslant y}\,3}{z \leqslant 0}\,[\forall E]}{\bot}\,[\neg E]}{\bot}\,3\ [\exists E]}{\neg\exists x.\forall y.x \leqslant y}\,2\ [\neg I]}{0 < z \to \neg\exists x.\forall y.x \leqslant y}\,1\ [\to I]$$

WARNING ⚠ Note that this is **not a correct use** of the $[\exists E]$ rule because z is free in $0 < z$, which is not-yet-discharged when the $[\exists E]$ rule is applied

Inference Rule for "exists elimination"

Similarly, without the condition we could prove:

$$\dfrac{\dfrac{\dfrac{}{0 < z}\,1 \qquad \dfrac{\dfrac{}{\exists x.\forall y.x \leqslant y}\,2 \qquad \dfrac{\dfrac{}{\forall y.z \leqslant y}\,3}{z \leqslant 0}\,[\forall E]}{z \leqslant 0}\,\cancel{3\ [\exists E]}}{\bot}\,[\neg E]}{\neg\exists x.\forall y.x \leqslant y}\,2\ [\neg I]}{0 < z \to \neg\exists x.\forall y.x \leqslant y}\,1\ [\to I]$$

WARNING ⚠ Note that this is **not a correct use** of the $[\exists E]$ rule because z is free in $z \leqslant 0$, the conclusion of the instance of the $[\exists E]$ rule

Inference Rule for "exists elimination"

We use contexts to make checking this condition more tractable
For example:

$$\dfrac{\dfrac{\dfrac{}{\exists x.\forall y.x \leqslant y}\,2 \qquad \dfrac{\dfrac{}{0 < z}\,1 \qquad z \leqslant 0}{\bot}\,[\neg E]}{\bot}\,3\ [\exists E]}{\dfrac{\bot}{\neg\exists x.\forall y.x \leqslant y}\,2\ [\neg I]}\,1\ [\to I]$$
$$0 < z \to \neg\exists x.\forall y.x \leqslant y$$

Context:

- 1: $0 < z$
- 2: $\exists x.\forall y.x \leqslant y$
- 3: $\forall y.w \leqslant y$

We cannot pick z anymore as it occurs free in the context
We must pick a fresh variable w not free in the context (1 and 2), the conclusion $\bot$, or $\exists x.\forall y.x \leqslant y$

We cannot conclude our proof anymore

Inference Rule for "exists elimination"

What could happen if we could pick a variable free in the $\exists$ formula?

Let us assume for the sake of this example that we can use the following rule

$$\frac{t < t}{\bot}\ [IRREFL]$$

If we could pick a variable free in the $\exists$ formula, we could derive:

$$\dfrac{\dfrac{}{\exists x.\exists y.x < y}\,1 \qquad \dfrac{\dfrac{}{\exists y.x < y}\,2 \qquad \dfrac{\dfrac{}{x < x}\,3}{\bot}\,[IRREFL]}{\bot}\,\cancel{3\ [\exists E]}}{\bot}\,2\ [\exists E]$$

$$\frac{\bot}{\neg\exists x.\exists y.x < y}\,1\ [\neg I]$$

WARNING ⚠ Note that this is **not a correct use** of the $[\exists E]$ rule because x is free in $\exists y.x < y$

Another Natural Deduction proof with contexts

Prove that $(\forall x.p(x)) \to (\forall y.q(y)) \to \forall z.p(z) \wedge q(z)$

Here is a proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall x.p(x)}\,1}{p(z)}\,[\forall E] \quad \dfrac{\dfrac{}{\forall y.q(y)}\,2}{q(z)}\,[\forall E]}{p(z) \wedge q(z)}\,[\wedge I]}{\forall z.p(z) \wedge q(z)}\,[\forall I]}{(\forall y.q(y)) \to \forall z.p(z) \wedge q(z)}\,2\ [\to I]}{(\forall x.p(x)) \to (\forall y.q(y)) \to \forall z.p(z) \wedge q(z)}\,1\ [\to I]$$

Context:

- 1: $\forall x.p(x)$
- 2: $\forall y.q(y)$

z does not occur free in the context or in the conclusion

Formal verification

Predicate Logic is more expressive and more convenient than Propositional Logic

- to do **Mathematics**
- to do **program verification**, i.e., to formally/mathematically verify that a program satisfies some formal/mathematical specification

Simple example: let the domain be $\mathbb{N}$ and the signature be:

- predicates: $\geqslant$ of arity 2
- functions: $\mathtt{max}$ of arity 2; and $0, 1, 2, \ldots$ of arity 0

Let us define the following function:

$$\mathtt{max3}(t_1, t_2, t_3) \text{ stands for } \mathtt{max}(t_1, \mathtt{max}(t_2, t_3))$$

A specification for $\mathtt{max}$ might be:

$$\forall x. \forall y. \mathtt{max}(x, y) \geqslant x \wedge \mathtt{max}(x, y) \geqslant y$$

Formal verification

While a specification for `max3` might be:

$$\forall x.\forall y.\forall z.\texttt{max3}(x,y,z) \geqslant x \wedge \texttt{max3}(x,y,z) \geqslant y \wedge \texttt{max3}(x,y,z) \geqslant z$$

Prove that `max3` satisfies this specification using Natural Deduction

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall x.\forall y.\texttt{max}(x,y) \geqslant x}{\forall y.\texttt{max}(u,y) \geqslant u}\,[\forall E]}{\texttt{max3}(u,v,w) \geqslant u}\,[\forall E] \quad \ldots}{\texttt{max3}(u,v,w) \geqslant u \wedge \texttt{max3}(u,v,w) \geqslant v \wedge \texttt{max3}(u,v,w) \geqslant w}\,[\wedge I]}{\forall z.\texttt{max3}(u,v,z) \geqslant u \wedge \texttt{max3}(u,v,z) \geqslant v \wedge \texttt{max3}(u,v,z) \geqslant z}\,[\forall I]}{\forall y.\forall z.\texttt{max3}(u,y,z) \geqslant u \wedge \texttt{max3}(u,y,z) \geqslant y \wedge \texttt{max3}(u,y,z) \geqslant z}\,[\forall I]}{\forall x.\forall y.\forall z.\texttt{max3}(x,y,z) \geqslant x \wedge \texttt{max3}(x,y,z) \geqslant y \wedge \texttt{max3}(x,y,z) \geqslant z}\,[\forall I]$$

We skipped some parts of the proof. For the missing part, we also need to assume that $\geqslant$ is transitive.

Conclusion

What did we cover today?

- Natural Deduction proofs for Predicate Logic
- side conditions

Further reading:

- Chapter 8 of
  http://leanprover.github.io/logic_and_proof/

Mathematical and Logical Foundations of Computer Science

Predicate Logic (Semantics)

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- Symbolic logic
- Propositional logic
- **Predicate logic**

Today

- Semantics of Predicate Logic
- Models
- Variable valuations
- Satisfiability & validity

Further reading:

- Chapter 10 of
  http://leanprover.github.io/logic_and_proof/

Recap: Syntax

The syntax of predicate logic is defined by the following grammar:

$$\begin{array}{lll} t & ::= & x \mid f(t, \ldots, t) \\ P & ::= & p(t, \ldots, t) \mid \neg P \mid P \wedge P \mid P \vee P \mid P \rightarrow P \mid \forall x.P \mid \exists x.P \end{array}$$

where:

- x ranges of variables
- f ranges over function symbols
- $f(t_1, \ldots, t_n)$ is a well-formed term only if f has arity n
- p ranges over predicate symbols
- $p(t_1, \ldots, t_n)$ is a well-formed formula only if p has arity n

The pair of a collection of function symbols, and a collection of predicate symbols, along with their arities, is called a **signature**.

The scope of a quantifier extends as far right as possible. E.g., $P \wedge \forall x.p(x) \vee q(x)$ is read as $P \wedge \forall x.(p(x) \vee q(x))$

Recap: Substitution

Substitution is defined recursively on terms and formulas:

$P[x\backslash t]$ substitute all the free occurrences of x in P with t.

$$
\begin{array}{lcl}
x[x\backslash t] & = & t \\
x[y\backslash t] & = & x \\
(f(t_1,\ldots,t_n))[x\backslash t] & = & f(t_1[x\backslash t],\ldots,t_n[x\backslash t]) \\
(p(t_1,\ldots,t_n))[x\backslash t] & = & p(t_1[x\backslash t],\ldots,t_n[x\backslash t]) \\
\hline
(\neg P)[x\backslash t] & = & \neg P[x\backslash t] \\
(P_1 \wedge P_2)[x\backslash t] & = & P_1[x\backslash t] \wedge P_2[x\backslash t] \\
(P_1 \vee P_2)[x\backslash t] & = & P_1[x\backslash t] \vee P_2[x\backslash t] \\
(P_1 \to P_2)[x\backslash t] & = & P_1[x\backslash t] \to P_2[x\backslash t] \\
\hline
(\forall x.P)[x\backslash t] & = & \forall x.P \\
(\exists x.P)[x\backslash t] & = & \exists x.P \\
(\forall y.P)[x\backslash t] & = & \forall y.P[x\backslash t], \text{ if } y \notin \mathtt{fv}(t) \\
(\exists y.P)[x\backslash t] & = & \exists y.P[x\backslash t], \text{ if } y \notin \mathtt{fv}(t)
\end{array}
$$

The additional **conditions** ensure that **free variables do not get captured**.

These conditions can always be met by silently renaming bound variables before substituting.

Recap: $\forall$ & $\exists$ elimination and introduction rules

Natural Deduction rules for quantifiers:

$$\frac{P[x\backslash y]}{\forall x.P}\ [\forall I] \qquad \frac{\forall x.P}{P[x\backslash t]}\ [\forall E] \qquad \frac{P[x\backslash t]}{\exists x.P}\ [\exists I] \qquad \frac{\exists x.P \qquad \begin{array}{c}\overline{P[x\backslash y]}^{\,1}\\ \vdots \\ Q\end{array}}{Q}\ 1\ [\exists E]$$

Condition:

- for $[\forall I]$: y must not be free in any not-yet-discharged hypothesis or in $\forall x.P$
- for $[\forall E]$: $\texttt{fv}(t)$ must not clash with $\texttt{bv}(P)$
- for $[\exists I]$: $\texttt{fv}(t)$ must not clash with $\texttt{bv}(P)$
- for $[\exists E]$: y must not be free in Q or in not-yet-discharged hypotheses or in $\exists x.P$

Recap: Example of a simple proof

here is a proof of $(\forall z.p(z)) \to \forall x.p(x) \vee q(x)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall z.p(z)}\,1}{p(y)}\,[\forall E]}{p(y) \vee q(y)}\,[\vee I_L]}{\forall x.p(x) \vee q(x)}\,[\forall I]}{(\forall z.p(z)) \to \forall x.p(x) \vee q(x)}\,1\ [\to I]$$

Conditions:

- y does not occur free in not-yet-discharged hypotheses or in $\forall x.p(x) \vee q(x)$
- y does not clash with bound variables in $p(z)$

Interpretation of Predicate & Function Symbols

Semantics: Assigning meaning/interpretations to formulas

Earlier in the module: a **particular semantics** for propositional logic

- Each proposition has a meaning (**a truth value**) of **T** or **F**
- Used truth tables to check **semantic validity**

We now **extend** this particular semantics to predicate logic

- Propositional logic constructs are interpreted similarly
- In addition, we need to interpret
  - **predicate & function symbols**
  - **quantifiers**

Predicate symbols: for example, given the domain $\mathbb{N}$ and a unary predicate symbol `even`, what is the meaning of `even`?

- to state that a number is $0, 2, 4, \ldots$?
- is it always obvious?
- what if we had a predicate symbol `small`?
- what does that mean?

Interpretation of Predicate & Function Symbols

Given a domain D and a predicate symbol p of arity n

- p is interpreted by a n-ary relation $\mathcal{R}_p$
- of the form $\{\langle d_1^1, \ldots, d_n^1 \rangle, \langle d_1^2, \ldots, d_n^2 \rangle, \ldots\}$
- where each d_j^i is in D
- we write: $\mathcal{R}_p \in 2^{D^n}$ or $\mathcal{R}_p \subseteq D^n$

For example

- a meaningful interpretation for `even` would be
  - $\{\langle 0 \rangle, \langle 2 \rangle, \langle 4 \rangle, \ldots\}$
- a meaningful interpretation for `odd` would be
  - $\{\langle 1 \rangle, \langle 3 \rangle, \langle 5 \rangle, \ldots\}$
- a meaningful interpretation for `prime` would be
  - $\{\langle 2 \rangle, \langle 3 \rangle, \langle 5 \rangle, \ldots\}$

Interpretation of Predicate & Function Symbols

Function symbols: for example, given the domain $\mathbb{N}$ and a binary function symbol `add`, what is the meaning of `add`?

- is it addition?
- is it always obvious?
- what if we had a binary function symbol `combine`?
- what does that mean?

Given a domain D and a function symbol f of arity n

- f is interpreted by a function $\mathcal{F}_f$ from D^n to D
- we write: $\mathcal{F}_f \in D^n \to D$

For example

- a meaningful interpretation for `add` would be
  - $+$ (formally: $\langle n, m \rangle \mapsto n + m$)
- a meaningful interpretation for `mult` would be
  - $\times$ (formally: $\langle n, m \rangle \mapsto n \times m$)

Interpretation of Predicate & Function Symbols

WARNING ⚠: sometimes for convenience we will use the same symbol for a function symbol and its interpretation

For example:

1. we have used 0 in our examples as a **constant symbol**, which has no meaning on its own
2. this constant symbol would be interpreted by the natural number 0, which is an **object of the domain** $\mathbb{N}$

Even though we used the same symbols, these symbols stand for different entities:

1. a **constant symbol**
2. an **object of the domain**

If we want to distinguish them, we might use:

1. $\overline{0}$ or `zero` for the **constant symbol**
2. 0 for the **object of the domain**

Models

Models: a model provides the interpretation of all symbols

Given a **signature** $\langle\langle f_1^{k_1}, \ldots, f_n^{k_n}\rangle, \langle p_1^{j_1}, \ldots, p_m^{j_m}\rangle\rangle$

- of function symbols f_i of arity k_i, for $1 \leqslant i \leqslant n$
- of predicate symbols p_i of arity j_i, for $1 \leqslant i \leqslant m$

a **model** is a structure $\langle D, \langle \mathcal{F}_{f_1}, \ldots, \mathcal{F}_{f_n}\rangle, \langle \mathcal{R}_{p_1}, \ldots, \mathcal{R}_{p_m}\rangle\rangle$

- of a non-empty domain D
- interpretations $\mathcal{F}_{f_i}$ for function symbols f_i ($\in D^{k_i} \to D$)
- interpretations $\mathcal{R}_{p_i}$ for predicate symbols p_i ($\subseteq D^{j_i}$)

Models of predicate logic replace **truth assignments** for propositional logic

For example:

- we might interpret the signature $\langle\langle \texttt{add}\rangle, \langle \texttt{even}\rangle\rangle$
  - where `add` is a binary function symbol
  - and `even` is a unary predicate symbol
- by the model $\langle \mathbb{N}, \langle\langle +\rangle, \langle \{\langle 0\rangle, \langle 2\rangle, \langle 4\rangle, \ldots\}\rangle\rangle\rangle$

Models

A **model** assigns meaning to function and predicate symbols

Variable valuations: In addition, we need to assign meaning to **variables**:

- this is done using a partial function v
- that maps variables to D
- i.e., a mapping of the form $x_1 \mapsto d_1, \ldots, x_n \mapsto d_n$
- which maps each x_i to d_i, i.e., to $v(x_i)$
- $\mathtt{dom}(v) = \{x_1, \ldots, x_n\}$
- let $\cdot$ be the empty mapping
- we write $v, x \mapsto d$ for the mapping that
  - maps x to d
  - and maps each $y \in \mathtt{dom}(v)$ such that $x \neq y$ to $v(y)$

For example

- $(x_1 \mapsto d_1), x_2 \mapsto d_2$ maps x_1 to $?d_1$ and x_2 to $?d_2$
- $(x_1 \mapsto d_1, x_2 \mapsto d_2), x_1 \mapsto d_3$ maps x_1 to $?d_3$ and x_2 to $?d_2$

Semantics of Predicate Logic

Given a **model** M with domain D and a **variable valuation** v, to assign **meaning** to Predicate Logic formulas, we define two operations:

- $[\![t]\!]_v^M$, which gives meaning to the term t w.r.t. M and v
- $\models_{M,v} P$, which gives meaning to the formula P w.r.t. M and v

Meaning of terms:

- $[\![x]\!]_v^M = v(x)$
- $[\![f(t_1, \ldots, t_n)]\!]_v^M = \mathcal{F}_f(\langle [\![t_1]\!]_v^M, \ldots, [\![t_n]\!]_v^M \rangle)$

Semantics of Predicate Logic

Given a **model** M with domain D and a **variable valuation** v, to assign **meaning** to Predicate Logic formulas, we define two operations:

- $[\![t]\!]_v^M$, which gives meaning to the term t w.r.t. M and v
- $\models_{M,v} P$, which gives meaning to the formula P w.r.t. M and v

Meaning of formulas:

- $\models_{M,v} p(t_1, \ldots, t_n)$ iff $\langle [\![t_1]\!]_v^M, \ldots, [\![t_n]\!]_v^M \rangle \in \mathcal{R}_p$
- $\models_{M,v} \neg P$ iff $\neg \models_{M,v} P$
- $\models_{M,v} P \wedge Q$ iff $\models_{M,v} P$ and $\models_{M,v} Q$
- $\models_{M,v} P \vee Q$ iff $\models_{M,v} P$ or $\models_{M,v} Q$
- $\models_{M,v} P \rightarrow Q$ iff $\models_{M,v} Q$ whenever $\models_{M,v} P$
- $\models_{M,v} \forall x.P$ iff for every $d \in D$ we have $\models_{M,(v,x \mapsto d)} P$
- $\models_{M,v} \exists x.P$ iff there exists a $d \in D$ such that $\models_{M,(v,x \mapsto d)} P$

Semantics of Predicate Logic

For example:

- consider the signature $\langle\langle \texttt{zero}, \texttt{succ}, \texttt{add}\rangle, \langle \texttt{even}, \texttt{odd}\rangle\rangle$
- the model M: $\langle \mathbb{N}, \langle 0, +1, +\rangle, \langle \{\langle 0\rangle, \langle 2\rangle, \langle 4\rangle, \dots\}, \{\langle 1\rangle, \langle 3\rangle, \langle 5\rangle, \dots\}\rangle\rangle$
- we write $+1$ for the function that given a number increments it by 1
- $+(n, m)$ stands for $n + m$

What is $\models_{M,\cdot} \texttt{even}(\texttt{succ}(\texttt{zero})) \vee \texttt{odd}(\texttt{succ}(\texttt{zero}))$?

- iff $\models_{M,\cdot} \texttt{even}(\texttt{succ}(\texttt{zero}))$ or $\models_{M,\cdot} \texttt{odd}(\texttt{succ}(\texttt{zero}))$
- iff $\langle [\![\texttt{succ}(\texttt{zero})]\!]^M_\cdot \rangle \in \{\langle 0\rangle, \langle 2\rangle, \langle 4\rangle, \dots\}$ or $\langle [\![\texttt{succ}(\texttt{zero})]\!]^M_\cdot \rangle \in \{\langle 1\rangle, \langle 3\rangle, \langle 5\rangle, \dots\}$
- iff $\langle 1\rangle \in \{\langle 0\rangle, \langle 2\rangle, \langle 4\rangle, \dots\}$ or $\langle 1\rangle \in \{\langle 1\rangle, \langle 3\rangle, \langle 5\rangle, \dots\}$
- iff True

Semantics of Predicate Logic

For example:

- consider the signature $\langle\langle \texttt{zero}, \texttt{succ}, \texttt{add}\rangle, \langle \texttt{even}, \texttt{odd}\rangle\rangle$
- the model M: $\langle \mathbb{N}, \langle 0, +1, +\rangle, \langle \{\langle 0\rangle, \langle 2\rangle, \langle 4\rangle, \dots\}, \{\langle 1\rangle, \langle 3\rangle, \langle 5\rangle, \dots\}\rangle\rangle$
- we write $+1$ for the function that given a number increments it by 1
- $+(n, m)$ stands for $n + m$

What is $\models_{M,\cdot} \forall x.\texttt{even}(x)$?

- iff for all $n \in \mathbb{N}$, $\models_{M, x\mapsto n} \texttt{even}(x)$
- iff for all $n \in \mathbb{N}$, $\langle [\![x]\!]^M_{x\mapsto n}\rangle \in \{\langle 0\rangle, \langle 2\rangle, \langle 4\rangle, \dots\}$
- iff for all $n \in \mathbb{N}$, $\langle n\rangle \in \{\langle 0\rangle, \langle 2\rangle, \langle 4\rangle, \dots\}$
- iff False, because $1 \notin \{0, 2, 4, \dots\}$

Semantics of Predicate Logic

For example:

- consider the signature $\langle\langle \mathtt{zero}, \mathtt{succ}, \mathtt{add}\rangle, \langle \mathtt{even}, \mathtt{odd}\rangle\rangle$
- the model M: $\langle \mathbb{N}, \langle 0, +1, +\rangle, \langle \{\langle 0\rangle, \langle 2\rangle, \langle 4\rangle, \dots\}, \{\langle 1\rangle, \langle 3\rangle, \langle 5\rangle, \dots\}\rangle\rangle$
- we write $+1$ for the function that given a number increments it by 1
- $+(n, m)$ stands for $n + m$

What is $\models_{M,\cdot} \forall x.\mathtt{even}(x) \to \neg\mathtt{odd}(x)$?

- iff for all $n \in \mathbb{N}$, $\models_{M,x\mapsto n} \mathtt{even}(x) \to \neg\mathtt{odd}(x)$
- iff for all $n \in \mathbb{N}$, $\models_{M,x\mapsto n} \neg\mathtt{odd}(x)$ whenever $\models_{M,x\mapsto n} \mathtt{even}(x)$
- iff for all $n \in \mathbb{N}$, $\neg \models_{M,x\mapsto n} \mathtt{odd}(x)$ whenever $\models_{M,x\mapsto n} \mathtt{even}(x)$
- iff for all $n \in \mathbb{N}$, $\langle [\![x]\!]^M_{x\mapsto n}\rangle \notin \{\langle 1\rangle, \langle 3\rangle, \langle 5\rangle, \dots\}$ whenever $\langle [\![x]\!]^M_{x\mapsto n}\rangle \in \{\langle 0\rangle, \langle 2\rangle, \langle 4\rangle, \dots\}$
- iff for all $n \in \mathbb{N}$, $\langle n\rangle \notin \{\langle 1\rangle, \langle 3\rangle, \langle 5\rangle, \dots\}$ whenever $\langle n\rangle \in \{\langle 0\rangle, \langle 2\rangle, \langle 4\rangle, \dots\}$
- iff for all $n \in \mathbb{N}$, $n \notin \{1, 3, 5, \dots\}$ whenever $n \in \{0, 2, 4, \dots\}$
- iff True

Semantics of Predicate Logic

For example:

- consider the signature $\langle\langle \mathtt{zero}, \mathtt{succ}, \mathtt{add}\rangle, \langle \mathtt{lt}, \mathtt{ge}\rangle\rangle$
- the model M:
  $\langle \mathbb{N}, \langle 0, +1, +\rangle, \langle \{\langle 0,1\rangle, \langle 0,2\rangle, \langle 1,2\rangle, \dots\}, \{\langle 0,0\rangle, \langle 1,1\rangle, \langle 1,0\rangle, \dots\}\rangle\rangle$
- we write $+1$ for the function that given a number increments it by 1
- $+(n,m)$ stands for $n+m$

What is $\models_{M,\cdot} \forall x.\forall y.\mathtt{lt}(x,y) \to \mathtt{ge}(y,x)$?

- iff for all $n, m \in \mathbb{N}$, $\models_{M, x\mapsto n, y\mapsto m} \mathtt{lt}(x,y) \to \mathtt{ge}(y,x)$
- iff for all $n, m \in \mathbb{N}$, $\models_{M, x\mapsto n, y\mapsto m} \mathtt{ge}(y,x)$ whenever $\models_{M, x\mapsto n, y\mapsto m} \mathtt{lt}(x,y)$
- iff for all $n, m \in \mathbb{N}$, $\langle [\![y]\!]^M_{x\mapsto n, y\mapsto m}, [\![x]\!]^M_{x\mapsto n, y\mapsto m}\rangle \in \{\langle 0,0\rangle, \langle 1,1\rangle, \langle 1,0\rangle, \dots\}$ whenever $\langle [\![x]\!]^M_{x\mapsto n, y\mapsto m}, [\![y]\!]^M_{x\mapsto n, y\mapsto m}\rangle \in \{\langle 0,1\rangle, \langle 0,2\rangle, \langle 1,2\rangle, \dots\}$
- iff for all $n, m \in \mathbb{N}$, $\langle m, n\rangle \in \{\langle 0,0\rangle, \langle 1,1\rangle, \langle 1,0\rangle, \dots\}$ whenever $\langle n, m\rangle \in \{\langle 0,1\rangle, \langle 0,2\rangle, \langle 1,2\rangle, \dots\}$
- iff True

Satisfiability & Validity

We write $\models_M P$ for $\models_{M,.} P$

Truth: P is **true** in the model M if $\models_M P$

We also say that M is a model of P

Satisfiability: P is **satisfiable** if there is a model M such that P is true in M, i.e., $\models_M P$

Validity: P is **valid** if for all model M, P is true in M

Example: $\models_{M,.} \forall x.\texttt{even}(x) \rightarrow \neg\texttt{odd}(x)$ is satisfiable (see above) but not valid because not true for example in the model $\langle \mathbb{N}, \langle 0, +1, + \rangle, \langle \{\langle 0 \rangle, \langle 2 \rangle, \langle 4 \rangle, \dots\}, \{\langle 0 \rangle, \langle 2 \rangle, \langle 4 \rangle, \dots\} \rangle \rangle$

Decidability: Validity is not decidable for predicate logic, i.e., there is no algorithm that given a formula P either returns "yes" if P is valid, and otherwise returns "no", while it is decidable for propositional logic

Recap: Soundness & Completeness

Given a deduction system such as Natural deduction, a formula is said to be **provable** if there is a proof of it in that deduction system

- This is a **syntactic** notion
- it asserts the existence of a syntactic object: a proof
- typically written $\vdash A$

A formula A is **valid** if for all model M, A is true in M, i.e., $\models_M P$

- it is a **semantic** notion
- it is checked w.r.t. valuations/models that give meaning to formulas
- written $\models A$

Soundness: a deduction system is sound w.r.t. a semantics if every provable formula is valid

- i.e., if $\vdash A$ then $\models A$

Completeness: a deduction system is complete w.r.t. a semantics if every valid formula is provable

- i.e., if $\models A$ then $\vdash A$

Soundness & Completeness

Natural Deduction for Predicate Logic is

- **sound** and
- **complete**

w.r.t. the **model semantics of Predicate Logic**

Proving those properties is done within the **metatheory**

We will not prove them here

Conclusion

What did we cover today?

- Semantics of Predicate Logic
- Models
- Variable valuations
- Satisfiability & validity

Further reading:

- Chapter 10 of
  http://leanprover.github.io/logic_and_proof/

Next time?

- Equivalences in Predicate Logic

Mathematical and Logical Foundations of Computer Science

Predicate Logic (Equivalences)

Vincent Rahli

(some slides were adapted from Rajesh Chitnis' slides)

University of Birmingham

Where are we?

- Symbolic logic
- Propositional logic
- **Predicate logic**

Today

Equivalences:

- in Natural Deduction
- using semantics

Further reading:

- Chapter 8 of
  http://leanprover.github.io/logic_and_proof/

Recap: Syntax

The syntax of predicate logic is defined by the following grammar:

$$\begin{array}{lll} t & ::= & x \mid f(t, \ldots, t) \\ P & ::= & p(t, \ldots, t) \mid \neg P \mid P \wedge P \mid P \vee P \mid P \rightarrow P \mid \forall x.P \mid \exists x.P \end{array}$$

where:

- x ranges over variables
- f ranges over function symbols
- $f(t_1, \ldots, t_n)$ is a well-formed term only if f has arity n
- p ranges over predicate symbols
- $p(t_1, \ldots, t_n)$ is a well-formed formula only if p has arity n

The pair of a collection of function symbols, and a collection of predicate symbols, along with their arities, is called a **signature**.

The scope of a quantifier extends as far right as possible. E.g., $P \wedge \forall x.p(x) \vee q(x)$ is read as $P \wedge \forall x.(p(x) \vee q(x))$

Recap: Substitution

Substitution is defined recursively on terms and formulas:

$P[x\backslash t]$ substitute all the free occurrences of x in P with t.

$$
\begin{array}{lcl}
x[x\backslash t] & = & t \\
x[y\backslash t] & = & x \\
(f(t_1,\ldots,t_n))[x\backslash t] & = & f(t_1[x\backslash t],\ldots,t_n[x\backslash t]) \\
(p(t_1,\ldots,t_n))[x\backslash t] & = & p(t_1[x\backslash t],\ldots,t_n[x\backslash t]) \\
\hline
(\neg P)[x\backslash t] & = & \neg P[x\backslash t] \\
(P_1 \wedge P_2)[x\backslash t] & = & P_1[x\backslash t] \wedge P_2[x\backslash t] \\
(P_1 \vee P_2)[x\backslash t] & = & P_1[x\backslash t] \vee P_2[x\backslash t] \\
(P_1 \to P_2)[x\backslash t] & = & P_1[x\backslash t] \to P_2[x\backslash t] \\
\hline
(\forall x.P)[x\backslash t] & = & \forall x.P \\
(\exists x.P)[x\backslash t] & = & \exists x.P \\
(\forall y.P)[x\backslash t] & = & \forall y.P[x\backslash t], \text{ if } y \notin \mathtt{fv}(t) \\
(\exists y.P)[x\backslash t] & = & \exists y.P[x\backslash t], \text{ if } y \notin \mathtt{fv}(t)
\end{array}
$$

The additional **conditions** ensure that **free variables do not get captured**.

These conditions can always be met by silently renaming bound variables before substituting.

Recap: $\forall$ & $\exists$ elimination and introduction rules

Natural Deduction rules for quantifiers:

$$\frac{P[x\backslash y]}{\forall x.P}\ [\forall I] \qquad \frac{\forall x.P}{P[x\backslash t]}\ [\forall E] \qquad \frac{P[x\backslash t]}{\exists x.P}\ [\exists I] \qquad \frac{\exists x.P \qquad \begin{array}{c}\overline{P[x\backslash y]}^{\,1}\\ \vdots \\ Q\end{array}}{Q}\ 1\ [\exists E]$$

Condition:

- for $[\forall I]$: y must not be free in any not-yet-discharged hypothesis or in $\forall x.P$
- for $[\forall E]$: $\texttt{fv}(t)$ must not clash with $\texttt{bv}(P)$
- for $[\exists I]$: $\texttt{fv}(t)$ must not clash with $\texttt{bv}(P)$
- for $[\exists E]$: y must not be free in Q or in not-yet-discharged hypotheses or in $\exists x.P$

Recap: Example of a proof

here is a proof of $(\forall z.p(z)) \to \forall x.p(x) \vee q(x)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\forall z.p(z)}\,1}{p(y)}\,[\forall E]}{p(y) \vee q(y)}\,[\vee I_L]}{\forall x.p(x) \vee q(x)}\,[\forall I]}{(\forall z.p(z)) \to \forall x.p(x) \vee q(x)}\,1\ [\to I]$$

Conditions:

- y does not occur free in not-yet-discharged hypotheses or in $\forall x.p(x) \vee q(x)$
- y does not clash with bound variables in $p(z)$

Recap: Models

Models: a model provides the interpretation of all symbols

Given a **signature** $\langle\langle f_1^{k_1}, \ldots, f_n^{k_n}\rangle, \langle p_1^{j_1}, \ldots, p_m^{j_m}\rangle\rangle$

- of function symbols f_i of arity k_i, for $1 \leqslant i \leqslant n$
- of predicate symbols p_i of arity j_i, for $1 \leqslant i \leqslant m$

a **model** is a structure $\langle D, \langle \mathcal{F}_{f_1}, \ldots, \mathcal{F}_{f_n}\rangle, \langle \mathcal{R}_{p_1}, \ldots, \mathcal{R}_{p_m}\rangle\rangle$

- of a non-empty domain D
- interpretations $\mathcal{F}_{f_i}$ for function symbols f_i
- interpretations $\mathcal{R}_{p_i}$ for function symbols p_i

Models of predicate logic replace **truth assignments** for propositional logic

Variable valuations:

- a partial function v
- that map variables to D
- i.e., a mapping of the form $x_1 \mapsto d_1, \ldots, x_n \mapsto d_n$

Recap: Semantics of Predicate Logic

Given a **model** M with domain D and a **variable valuation** v:

- $[\![t]\!]_v^M$ gives meaning to the term t w.r.t. M and v
- $\models_{M,v} P$ gives meaning to the formula P w.r.t. M and v

Meaning of terms:

- $[\![x]\!]_v^M = v(x)$
- $[\![f(t_1, \ldots, t_n)]\!]_v^M = \mathcal{F}_f(\langle [\![t_1]\!]_v^M, \ldots, [\![t_n]\!]_v^M \rangle)$

Meaning of formulas:

- $\models_{M,v} p(t_1, \ldots, t_n)$ iff $\langle [\![t_1]\!]_v^M, \ldots, [\![t_n]\!]_v^M \rangle \in \mathcal{R}_p$
- $\models_{M,v} \neg P$ iff $\neg \models_{M,v} P$
- $\models_{M,v} P \wedge Q$ iff $\models_{M,v} P$ and $\models_{M,v} Q$
- $\models_{M,v} P \vee Q$ iff $\models_{M,v} P$ or $\models_{M,v} Q$
- $\models_{M,v} P \rightarrow Q$ iff $\models_{M,v} Q$ whenever $\models_{M,v} P$
- $\models_{M,v} \forall x.P$ iff for every $d \in D$ we have $\models_{M,(v,x \mapsto d)} P$
- $\models_{M,v} \exists x.P$ iff there exists a $d \in D$ such that $\models_{M,(v,x \mapsto d)} P$

Recap: Logical equivalences for Propositional Logic

The same equivalences hold as in Propositional Logic:

- De Morgan's law (I): $\neg(A \vee B) \leftrightarrow (\neg A \wedge \neg B)$
- De Morgan's law (II): $\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B)$
- Implication elimination: $(A \rightarrow B) \leftrightarrow (\neg A \vee B)$
- Commutativity of $\wedge$: $(A \wedge B) \leftrightarrow (B \wedge A)$
- Commutativity of $\vee$: $(A \vee B) \leftrightarrow (B \vee A)$
- Associativity of $\wedge$: $((A \wedge B) \wedge C) \leftrightarrow (A \wedge (B \wedge C))$
- Associativity of $\vee$: $((A \vee B) \vee C) \leftrightarrow (A \vee (B \vee C))$
- Distributivity of $\wedge$ over $\vee$: $(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$
- Distributivity of $\vee$ over $\wedge$: $(A \vee (B \wedge C)) \leftrightarrow ((A \vee B) \wedge (A \vee C))$
- Double negation elimination: $(\neg\neg A) \leftrightarrow A$
- Idempotence: $(A \wedge A) \leftrightarrow A$ and $(A \vee A) \leftrightarrow A$

Logical Equivalences

In addition, the following hold (some hold only classically):

- $(\forall x.A \wedge B) \leftrightarrow ((\forall x.A) \wedge (\forall x.B))$
- $(\exists x.A \vee B) \leftrightarrow ((\exists x.A) \vee (\exists x.B))$
- $(\neg\forall x.A) \leftrightarrow (\exists x.\neg A)$
- $(\neg\exists x.A) \leftrightarrow (\forall x.\neg A)$
- $(\forall x.A) \leftrightarrow A$ if $x \notin \mathtt{fv}(A)$
- $(\exists x.A) \leftrightarrow A$ if $x \notin \mathtt{fv}(A)$
- $(\forall x.A \vee B) \leftrightarrow ((\forall x.A) \vee B)$ if $x \notin \mathtt{fv}(B)$
- $(\exists x.A \wedge B) \leftrightarrow ((\exists x.A) \wedge B)$ if $x \notin \mathtt{fv}(B)$
- $(\forall x.A \rightarrow B) \leftrightarrow ((\exists x.A) \rightarrow B)$ if $x \notin \mathtt{fv}(B)$
- $(\exists x.A \rightarrow B) \leftrightarrow ((\forall x.A) \rightarrow B)$ if $x \notin \mathtt{fv}(B)$
- $(\forall x.A \rightarrow B) \leftrightarrow (A \rightarrow \forall x.B)$ if $x \notin \mathtt{fv}(A)$
- $(\exists x.A \rightarrow B) \leftrightarrow (A \rightarrow \exists x.B)$ if $x \notin \mathtt{fv}(A)$

Logical Equivalences

As before to prove a logical equivalence $A \leftrightarrow B$, we will prove:

- that we can derive B form A
- that we can derive A form B

We will prove:

- $(\forall x.A \wedge B) \leftrightarrow ((\forall x.A) \wedge (\forall x.B))$
- $(\exists x.A \vee B) \leftrightarrow ((\exists x.A) \vee (\exists x.B))$
- $(\neg \forall x.A) \leftrightarrow (\exists x.\neg A)$
- $(\neg \exists x.A) \leftrightarrow (\forall x.\neg A)$

Logical Equivalences

Prove the logical equivalence $(\forall x.A \wedge B) \leftrightarrow ((\forall x.A) \wedge (\forall x.B))$ in Natural Deduction

Here is a proof of the left-to-right implication (constructive):

$$\dfrac{\dfrac{\dfrac{\dfrac{\forall x.A \wedge B}{A[x\backslash y] \wedge B[x\backslash y]}\,[\forall E]}{A[x\backslash y]}\,[\wedge E_L]}{\forall x.A}\,[\forall I] \qquad \dfrac{\dfrac{\dfrac{\dfrac{\forall x.A \wedge B}{A[x\backslash y] \wedge B[x\backslash y]}\,[\forall E]}{B[x\backslash y]}\,[\wedge E_R]}{\forall x.B}\,[\forall I]}{(\forall x.A) \wedge (\forall x.B)}\,[\wedge I]$$

- pick y such that it does not occur in A or B
- y must not be free in $\forall x.A \wedge B$ or in $\forall x.A$
- y must not clash with $\mathtt{bv}(A \wedge B)$
- y must not be free in $\forall x.A \wedge B$ or in $\forall x.B$
- y must not clash with $\mathtt{bv}(A \wedge B)$

Logical Equivalences

Prove the logical equivalence $(\forall x.A \wedge B) \leftrightarrow ((\forall x.A) \wedge (\forall x.B))$ in Natural Deduction

Here is a proof of the right-to-left implication (constructive):

$$\dfrac{\dfrac{\dfrac{\dfrac{(\forall x.A) \wedge (\forall x.B)}{\forall x.A}\,[\wedge E_L]}{A[x\backslash y]}\,[\forall E] \quad \dfrac{\dfrac{(\forall x.A) \wedge (\forall x.B)}{\forall x.B}\,[\wedge E_R]}{B[x\backslash y]}\,[\forall E]}{A[x\backslash y] \wedge B[x\backslash y]}\,[\wedge I]}{\forall x.A \wedge B}\,[\forall I]$$

- pick y such that it does not occur in A or B
- y must not be free in $(\forall x.A) \wedge (\forall x.B)$ or in $\forall x.A \wedge B$
- y must not clash with $\mathtt{bv}(A)$
- y must not clash with $\mathtt{bv}(B)$

Logical Equivalences

Prove the logical equivalence $(\exists x.A \vee B) \leftrightarrow ((\exists x.A) \vee (\exists x.B))$ in Natural Deduction

Here is a proof of the left-to-right implication (constructive):

$$\dfrac{\exists x.A \vee B \qquad \dfrac{\dfrac{A[x\backslash y] \vee B[x\backslash y]}{(\exists x.A) \vee (\exists x.B)}\,1 \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A[x\backslash y]}\,2}{\exists x.A}\,[\exists I]}{(\exists x.A) \vee (\exists x.B)}\,[\vee I_L]}{A[x\backslash y] \to (\exists x.A) \vee (\exists x.B)}\,2\,[\to I] \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{B[x\backslash y]}\,3}{\exists x.B}\,[\exists I]}{(\exists x.A) \vee (\exists x.B)}\,[\vee I_R]}{B[x\backslash y] \to (\exists x.A) \vee (\exists x.B)}\,3\,[\to I]}{(\exists x.A) \vee (\exists x.B)}\,[\vee E]}{(\exists x.A) \vee (\exists x.B)}\,1\,[\exists E]$$

- pick y such that it does not occur in A or B
- 1: $A[x\backslash y] \vee B[x\backslash y]$
- 2: $A[x\backslash y]$
- 3: $B[x\backslash y]$

Logical Equivalences

Prove the logical equivalence $(\exists x.A \vee B) \leftrightarrow ((\exists x.A) \vee (\exists x.B))$ in Natural Deduction

Here is a proof of the right-to-left implication (constructive):

$$\dfrac{(\exists x.A) \vee (\exists x.B) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\exists x.A}\,1 \qquad \dfrac{\dfrac{\dfrac{}{A[x\backslash y]}\,2}{A[x\backslash y] \vee B[x\backslash y]}\,[\vee I_L]}{\exists x.A \vee B}\,[\exists I]}{\exists x.A \vee B}\,2\,[\exists E]}{\exists x.A \to \exists x.A \vee B}\,1\,[\to I] \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\exists x.B}\,3 \qquad \dfrac{\dfrac{\dfrac{}{B[x\backslash y]}\,4}{A[x\backslash y] \vee B[x\backslash y]}\,[\vee I_R]}{\exists x.A \vee B}\,[\exists I]}{\exists x.A \vee B}\,4\,[\exists E]}{\exists x.B \to \exists x.A \vee B}\,3\,[\to I]}{\exists x.A \vee B}\,[\vee E]$$

- 1: $\exists x.A$
- pick y such that it does not occur in A or B
- 2: $A[x\backslash y]$
- 3: $\exists x.B$
- 4: $B[x\backslash y]$

Logical Equivalences

Prove the logical equivalence $(\neg\forall x.A) \leftrightarrow (\exists x.\neg A)$ in Natural Deduction

Here is a proof of the left-to-right implication (classical):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg A[x\backslash y]}\,2}{\exists x.\neg A}\,[\exists I] \qquad \dfrac{}{\neg(\exists x.\neg A)}\,1}{\bot}\,[\neg E]}{\neg\neg A[x\backslash y]}\,2\,[\neg I]}{A[x\backslash y]}\,[DNE]}{\forall x.A}\,[\forall I] \qquad \neg\forall x.A}{\bot}\,[\neg E]}{\neg\neg(\exists x.\neg A)}\,1\,[\neg I]}{\exists x.\neg A}\,[DNE]$$

- 1: $\neg(\exists x.\neg A)$
- pick y such that it does not occur in A
- 2: $\neg A[x\backslash y]$

Logical Equivalences

Prove the logical equivalence $(\neg\forall x.A) \leftrightarrow (\exists x.\neg A)$ in Natural Deduction

Here is a proof of the right-to-left implication (constructive):

$$\dfrac{\dfrac{\exists x.\neg A \qquad \dfrac{\dfrac{\dfrac{}{\neg A[x\backslash y]}\,2 \qquad \dfrac{\dfrac{}{\forall x.A}\,1}{A[x\backslash y]}\,[\forall E]}{\bot}\,[\neg E]}{\bot}\,2\,[\exists E]}{\neg\forall x.A}\,1\,[\neg I]$$

- 1: $\forall x.A$
- pick y such that it does not occur in A
- 2: $\neg A[x\backslash y]$

Logical Equivalences

Prove the logical equivalence $(\neg\exists x.A) \leftrightarrow (\forall x.\neg A)$ in Natural Deduction

Here is a proof of the left-to-right implication (constructive):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A[x\backslash y]}\,1}{\exists x.A}\,[\exists I] \quad \neg\exists x.A}{\bot}\,[\neg E]}{\neg A[x\backslash y]}\,1\,[\neg I]}{\forall x.\neg A}\,[\forall I]$$

- pick y such that it does not occur in A
- 1: $A[x\backslash y]$

Logical Equivalences

Prove the logical equivalence $(\neg\exists x.A) \leftrightarrow (\forall x.\neg A)$ in Natural Deduction

Here is a proof of the right-to-left implication (constructive):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall x.\neg A}{\neg A[x\backslash y]}\,[\forall E] \quad \dfrac{}{A[x\backslash y]}\,2}{\bot}\,[\neg E]}{\bot}\,2\;[\exists E]}{\exists x.A}\,1}{\bot}\,1\;[\neg I]}{\neg\exists x.A}$$

- 1: $\exists x.A$
- pick y such that it does not occur in A
- 2: $A[x\backslash y]$

Logical Equivalences

As before: if ($P \leftrightarrow Q$ or $Q \leftrightarrow P$) and P occurs in A, then replacing P by Q in A leads to a formula B, such that $A \leftrightarrow B$

Also,

Semantical equivalence: two formulas P and Q are equivalent if for all models M and valuations v, $\models_{M,v} P$ iff $\models_{M,v} Q$

Logical Equivalences

Example: prove $(\neg\exists x.A) \leftrightarrow (\forall x.\neg A)$

- if $\models_{M,v} \neg\exists x.A$ then $\models_{M,v} \forall x.\neg A$
  - to prove: $\models_{M,v} \forall x.\neg A$, i.e., for every $d \in D$ it is not the case that $\models_{M,v,x\mapsto d} A$
  - assume $d \in D$ and $\models_{M,v,x\mapsto d} A$, and prove a contradiction
  - assumption: $\models_{M,v} \neg\exists x.A$, i.e., it is not the case that there exists a $e \in D$ such that $\models_{M,v,x\mapsto e} A$
  - contradiction! there is one: take $e = d$
- if $\models_{M,v} \forall x.\neg A$ then $\models_{M,v} \neg\exists x.A$
  - to prove: $\models_{M,v} \neg\exists x.A$, i.e., it is not the case that there exists a $e \in D$ such that $\models_{M,v,x\mapsto e} A$
  - assume that there exists a $e \in D$ such that $\models_{M,v,x\mapsto e} A$, and prove a contradiction
  - assumption: $\models_{M,v} \forall x.\neg A$, i.e., for every $d \in D$ it is not the case that $\models_{M,v,x\mapsto d} A$
  - therefore, instantiating this assumption with e: it is not the case that $\models_{M,v,x\mapsto e} A$
  - contradiction!

Conclusion

What did we cover today?

- Equivalence using Natural Deduction
- Equivalences using semantics

Further reading:

- Chapter 8 of
  http://leanprover.github.io/logic_and_proof/

Next time?

- Predicate Logic – Equivalences