## Contents

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lecture 01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.1 Introduction to CS 245</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>1.2 Introduction to Propositional Logic</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>1.2.1 Applications of Logic in Computer Science</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>1.2.2 Why We Need Formal Languages</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Lecture 02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.1 Inductive Definitions of Sets</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2.2 Structural Induction</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2.3 Introduction to the Syntax of Propositional Formulas</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Lecture 03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.1 Details Needed for a01</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>3.2 Structural Induction</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>3.3 Unique Readability of Propositional Formulas</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>3.4 Parse Trees</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>Lecture 04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.1 Cleanup from L03</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>4.1.1 Completing the Proof of Unique Readability</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>4.1.2 Generation Sequences for Propositional Formulas</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>4.1.3 Precedence Rules For Connectives</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>4.1.4 Propositions</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>4.1.5 Parse Trees - Correcting Notation</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>4.2 Structural Induction - More Examples</td>
<td>15</td>
</tr>
</tbody>
</table>
5 Lecture 05
5.1 Semantics of Propositional Logic
5.1.1 Truth Valuations
5.1.2 Evaluating Any Propositional Formula
5.1.3 Properties of Propositional Formulae

6 Lecture 06
6.1 Tautological Consequence
6.1.1 Satisfaction of a Set of Formulæ
6.1.2 Definition of Tautological Consequence
6.1.3 Subtleties About Tautological Consequence
6.1.4 Examples
6.2 Translations Between English and Propositional Logic

7 Lecture 07
7.1 Formal Deduction
7.1.1 Introduction to Proof Systems

8 Lecture 08
8.1 Formal Deduction
8.1.1 Formal Deduction - Inference Rules
8.1.2 Proof Examples

9 Lecture 09
9.1 The Soundness of Formal Deduction
9.1.1 How to Prove a Tautological Consequence in General
9.1.2 Soundness

10 Lecture 10
10.1 The Completeness of Formal Deduction
10.1.1 Useful Ingredients
10.2 Proof of the Completeness Theorem

11 Lecture 11
11.1 Soundness Revisited
11.2 Introduction to Predicate Logic

12 Lecture 12
12.1 Informal Definitions
12.2 Ingredients of Predicate Logic
12.3 Syntax of Predicate Logic
12.3.1 Terms
12.3.2 Atomic Formulæ
12.3.3 General Formula
### 13 Lecture 13

13.1 Translations from English into Predicate Logic ........................................ 47
13.2 Translations from Predicate Logic into English ......................................... 48
13.3 Semantics of Predicate Logic ................................................................. 48
  13.3.1 Domains ............................................................................................. 49
  13.3.2 Interpretations ..................................................................................... 49
  13.3.3 Assignments ......................................................................................... 49
  13.3.4 Valuations ............................................................................................ 49
  13.3.5 Terms .................................................................................................. 49
  13.3.6 Formula ............................................................................................... 49

### 14 Lecture 14

14.1 Semantics of Predicate Logic ...................................................................... 51
  14.1.1 Satisfiability and Validity ..................................................................... 51
  14.1.2 Examples ............................................................................................ 52
  14.1.3 Tautological Consequence .................................................................. 53

### 15 Lecture 15

15.1 Definability ................................................................................................. 55
  15.1.1 Definability of a Relation ..................................................................... 55
  15.1.2 Definability of a Set of Structures ....................................................... 56
15.2 More Examples of Tautological Consequence ............................................. 57

### 16 Lecture 16

16.1 Formal Deduction for Predicate Logic ....................................................... 58
  16.1.1 Inference Rules for Quantified Formula ............................................. 58
  16.1.2 Examples ............................................................................................ 59

### 17 Lecture 17

17.1 Soundness and Completeness of Formal Deduction for Predicate Logic .......... 61
  17.1.1 Soundness .......................................................................................... 61
  17.1.2 Rule \( \forall \) Is Sound ............................................................................. 61
  17.1.3 Rule \( \exists^+ \) Is Sound ....................................................................... 61
  17.1.4 Rule \( \forall^+ \) Is Sound ......................................................................... 62
  17.1.5 Rule \( \exists^- \) Is Sound ......................................................................... 63
  17.1.6 Rule \( \approx^- \) Is Sound ........................................................................ 63
  17.1.7 Rule \( \approx^+ \) Is Sound ....................................................................... 63
  17.1.8 Completeness ...................................................................................... 64
  17.1.9 Examples ........................................................................................... 64
1 Lecture 01

Outline

1. Introduction to CS 245
2. Introduction to Propositional Logic
   (a) Applications of Logic in Computer Science
   (b) Why We Need Formal Languages

1.1 Introduction to CS 245

Q: How will the course run?
A: Refer to the course website for high-level answers: https://www.student.cs.uwaterloo.ca/~cs245/

Q: How will iClickers work?

A: Register your iClicker using the link on the course website. Then bring your iClicker to class to answer questions.

- CQ 1

1.2 Introduction to Propositional Logic

1.2.1 Applications of Logic in Computer Science

Setting Course Expectations: You should not expect that CS 245 will (directly) improve your coding skills. You should instead expect that CS 245 will make you a more effective thinker about coding. This will ultimately improve your coding, once you have assimilated CS 245 into your thinking.

1. SAT-solvers (Propositional Logic)
2. Database Analysis (Predicate Logic)
3. Properties of the Natural Numbers - Peano Axioms
4. Program Verification
5. Decidability and Undecidability
6. Definability and Undefinability
7. Provability and Unprovability
8. Artificial Intelligence
9. and many more ...

1.2.2 Why We Need Formal Languages

Motivation: We are often tempted to rely on our own understanding and intuition exclusively. However sometimes an apparently reasonable decision problem has subtleties.

Definition 1.2.1. A **decision problem** is a problem which calls for an answer of either yes or no, given some input.

Definition 1.2.2. A **paradox** is a declarative statement that

1. cannot be true, and
2. cannot be false.

Examples:

1. “This sentence is false.”
2. The Barber Paradox
   There is a barber who is said to shave all men, and only those men, who do not shave themselves. Who shaves the barber?
• As phrased, the barber could be a woman. But what if we insist the barber is a man?
• Then if the barber shaves himself it is because he does not shave himself, and in turn this is because he does shave himself.
• “Does the barber shave himself?” is unanswerable.

3. Barey’s Paradox
Consider the set of natural numbers \( \mathbb{N} = \{0, 1, 2, 3, \ldots\} \).

Remarks:
(a) Unlike what you were likely told in MATH 135, in CS 245 we take 0 to be the first Natural number.

Definition 1.2.3. We say that a Natural Number, \( n \), has a **compact definition** if there is an English sentence of at most 200 characters that uniquely defines the number \( n \).

Examples:
(a) “\( n \) is 3.”
(b) “\( n \) is the difference of 10 and 7.”
(c) “\( n \) is one million.”
(d) “\( n \) is one million to the power of one million.”
(e) “\( n \) is the number of cells in my body.”
(f) “\( n \) is the number of grains of sand on a California beach.”
(g) The sentence “\( n \) is even.” identifies no Natural number.
(h) The sentence “Fruit flies like a banana.” identifies no Natural number.

Let \( B \) be the set of all Natural numbers that have a compact definition.

Q: Is \( B \) a finite set? (CQ 2)
A: Yes, \( |B| \leq 40^{200} < \infty \) (where we get 40 from 26 letters, 10 digits and a few punctuation marks). There are only finitely many compact English descriptions.
Since \( B \) is finite and \( \mathbb{N} \) is infinite, we may consider the first Natural number, \( x \), which does not have a compact definition.

Q: Is \( x \in B \)?
A:
• If \( x \in B \), then by construction \( x \) is the first Natural Number such that \( x \notin B \).
• If \( x \notin B \), then by construction there exists a way to define \( x \) by an English sentence of length \( \leq 200 \). So \( x \in B \).

Then this shows the paradox.

Moral: The presence of such paradoxes, arising from descriptions in some natural language (English in these cases), tells us that we will need formal languages to study logic carefully:

1. Propositional Logic
2. Predicate Logic
2 Lecture 02

Outline

1. Inductive Definitions of Sets
2. Structural Induction
3. Introduction to the Syntax of Propositional Logic

2.1 Inductive Definitions of Sets

In CS 245, structural induction will be a theme of the course. In this lecture we give the setup which will permit us to carry out structural induction correctly in every situation. The first example of such a set is the set of well-formed Propositional formulæ.

You may find this approach is too abstract for your taste, especially so early in the course. While I understand this reaction, I assure you that if we invest the time to understand the general setup, then we will reap the rewards throughout the rest of the course.

What techniques do we have for declaring sets?

1. Explicitly list all elements of the set. For example, \{3, 6, 7\}. Drawback: we cannot do this for infinite sets.
2. Characterizing property. For example, [the set of all even Natural numbers]
3. Inductively, for example, the set of all my blood relatives. Core set = \{me\} Operations = \{ son of, brother of, sister of, wife of, husband of, mother of, father of \}

Formally: Inductive definition sets: consists of 3 components:

1. a Universe of all elements denoted by \(X\),
2. a Core set denoted by \(A\), with \(A \subseteq X\), and
3. a set of operations (functions) denoted by \(F\)

Definition Given any subset \(Y \subseteq X\), and any set \(F\) of operations (functions \(f : X^k \rightarrow X\) for any \(k \geq 1\)), \(Y\) is closed under \(F\) if, for every \(f \in F\), (say \(f\) is a \(k\)-ary function) and every \(y_1, \ldots, y_k \in Y\), \(f(y_1, \ldots, y_k) \in Y\).

Example Let \(Y\) be the set of even Natural numbers. Let \(F\) be the set of addition, and multiplication. Then we know \(Y\) is closed under \(F\).

\(Y\) is not closed if we include subtraction in \(F\). Time permitting, give an example to demonstrate this.

Definition \(Y\) is a minimal set with respect to a property \(R\) if

1. \(Y\) satisfies \(R\), and
2. for every set \(Z\) that satisfies \(R\), \(Y \subseteq Z\).

Now we have the formal definition.
Definition \( I(X, A, F) = \) The minimal subset of \( X \) that

1. contains \( A \), and
2. is closed under the operations in \( F \).

Example The set of Natural numbers:

\[
\mathbb{N} = I \left( \mathbb{R}, \{0\}, \{ f(x) = x + 1 \} \right) .
\]

Exercise: Prove it. One direction is easy. The other direction can be proved using POMI, the way you would have in MATH 135.

Example: \( I(X, A, F) \) is the set of polynomials in variable \( z \) over a field \( K \) (say the field of real numbers), with:

\[
\begin{align*}
X & = \text{the set of all strings that can be written using } \mathbb{R} \cup \{+, \cdot, z\} \\
A & = \{z\} \cup \mathbb{R} \\
F & = \{+, \cdot\}
\end{align*}
\]

2.2 Structural Induction

Here we explain how to use structural induction to prove a property \( R \) holds for every element of a set \( I(X, A, F) \).

1. Prove that \( R(a) \) holds for every \( a \) in the core set \( A \) (the base case).
2. Prove that for every \( k \)-ary \( f \in F \) (for any \( k \geq 1 \)), \( R(f(y_1, \ldots, y_k)) \) holds whenever \( R(y_1), \ldots, R(y_k) \) all hold (the inductive case).

2.3 Introduction to the Syntax of Propositional Logic

1. Let \( P \) be a set of Propositional variables, e.g. \( P = \{p, q, r\} \).
2. Let \( C \) be the set of Propositional connectives, namely \( C = \{\neg, \land, \lor, \rightarrow, \leftrightarrow\} \).
3. Let \( X \) be the set of all strings that can be written using \( P \cup C \cup \{\,(,\,\,)\} \).
4. Let \( F \) be the set containing the following functions defined on \( X \):
   \[
   \begin{align*}
   & (a) \ neg(x) = (\neg x) \text{ (unary)} \\
   & (b) \ and(x, y) = (x \land y) \text{ (binary)} \\
   & (c) \ or(x, y) = (x \lor y) \text{ (binary)} \\
   & (d) \ impl(x, y) = (x \rightarrow y) \text{ (binary)} \\
   & (e) \ equiv(x, y) = (x \leftrightarrow y) \text{ (binary)}
   \end{align*}
   \]

Definition 2.3.1. Using the notation above, the set of well-formed Propositional formulae over \( F \) is defined inductively, as

\[
I(X, P, F)
\]
See also Definitions 2.2.1, 2.2.2 and 2.2.3 in the text.

Examples:

1. Each of the following is a well-formed Propositional formula over \( P = \{p, q, r\} \).
   - (a) \( p \)
   - (b) \( q \)
   - (c) \( (p \land q) \)
   - (d) \( ((p \land q) \land p) \)
   - (e) \( (p \rightarrow (q \lor r)) \)

2. This string is not a well-formed Propositional formula.

\[
)p \rightarrow \land \leftrightarrow rq()
\]

How would you prove this fact? We will discuss this next time.

3 Lecture 03

Outline

1. Details Needed for a01
2. Structural Induction
3. Unique Readability of Propositional Formulae
4. Parse Trees

3.1 Details Needed for a01

Definition 3.1.1. A \textit{proposition} is a declarative sentence that is either true or false, in some context.

Examples:

1. If I feed my fish, and I change my fish’s tank filter, then my fish will be healthy.  
   (compound, not simple)

Definition 3.1.2. An \textit{simple proposition} is a proposition that cannot be broken down into smaller propositions. A proposition that is not simple is called \textit{compound}.

Examples:

1. My fish is healthy.  (simple)

Remarks:

1. We assign Propositional variables like \( p, q \) and \( r \) to represent simple propositions when we translate from English into Propositional Logic.
Order of Precedence for Propositional Connectives: As on p 33 of the text, we may omit some parentheses once we agree on an order of precedence for the connectives. The order is

1. \( \neg \)
2. \( \land \)
3. \( \lor \)
4. \( \rightarrow \)
5. \( \leftrightarrow \)

Examples: On each row of the following table, we give a formula with some (or all) parentheses omitted, followed by the well-formed formula that results from adding parentheses according to the above precedence rules.

<table>
<thead>
<tr>
<th>some (or all) parentheses omitted</th>
<th>well-formed</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p \lor q \land r )</td>
<td>((p \lor (q \land r)))</td>
</tr>
<tr>
<td>( \neg p \lor q )</td>
<td>(((\neg p) \lor q))</td>
</tr>
<tr>
<td>( p \rightarrow q \land r )</td>
<td>((p \rightarrow (q \land r)))</td>
</tr>
<tr>
<td>( p \rightarrow q \leftrightarrow r )</td>
<td>(((p \rightarrow q) \leftrightarrow r))</td>
</tr>
<tr>
<td>( p \land q \rightarrow \neg r )</td>
<td>(((p \land q) \rightarrow (\neg r)))</td>
</tr>
</tbody>
</table>

Remarks:

1. We will say “well-formed Propositional formula” to refer to a formula which is syntactically correct according to the earlier definition.
2. We will say “Propositional formula”, to refer to a formula which may be well-formed, or may have some parentheses omitted, where the correct well-formed formula could be recovered according to the precedence rules.

3.2 Structural Induction

- CQs, as appropriate

Problems:

1. **Problem**: Prove by structural induction that every well-formed Propositional formula contains at least one Propositional variable.
   **Solution**: Exercise.

2. **Problem**: Prove that every well-formed Propositional formula, \( A \), has the same number of \( '(' \) and \( ')' \) symbols.
   **Solution**: Exercise.

Notation:

- \( \text{op}(A) \) denotes the number of \( '(' \) symbols in \( A \).
- \( \text{cl}(A) \) denotes the number of \( ')' \) symbols in \( A \).
- “Without loss of generality” clearly applies to all the binary connectives.
- We prove the result for negations of the form \( A = (\neg B) \) too. This is omitted
from the slides.

\[ op(\neg A) = 1 + op(A) \text{ (inspection)} \]
\[ = 1 + cl(A) \text{ (induction hypothesis: } R(A)) \]
\[ = cl(\neg A) \text{ (inspection)} \]

### 3.3 Unique Readability of Propositional Formulae

**Theorem (Unique Readability of Propositional Formulae) 3.3.1.** Every well-formed Propositional formula, \( A \), is exactly one of an atom, \( \neg B \), \( B \land C \), \( B \lor C \), or \( B \implies C \); and in each case \( A \) is of that form in exactly one way.

Property \( R(A) \): A formula \( A \) has property \( R(A) \) iff it satisfies all three of the following.

1: The first symbol of \( A \) is either ‘(‘ or a variable.
2: \( A \) has an equal number of ‘(‘ and ‘)’, and each proper initial segment of \( A \) has more ‘(‘ than ‘)’.
3: \( A \) has a unique construction as a formula.

(A proper initial segment of \( A \) is a non-empty expression \( x \) such that \( A \) is \( xy \) for some non-empty expression \( y \).)

We prove property \( R(A) \) for all formulas \( A \), by Structural Induction on \( A \).

**Base (\( A \) is \( p \), for some Propositional variable, \( p \)):**

- 1: trivial.
- 2:
  - first part: trivial.
  - second part: vacuous (since \( A \) has no proper initial segments in this case).
- 3: trivial.

The induction step has two sub-cases.

1. \( A \) is \( \neg B \), for some well-formed Propositional formula \( B \):

   The inductive hypothesis is that the formula \( B \) has property \( R \).
   - 1: By construction, \( \neg B \) has Property 1, since it begins with ‘(‘.
   - 2: Since \( B \) has an equal number of left and right parentheses, therefore so does \( \neg B \).

   For the second part of Property 2, we check these subcases for every possible proper initial segment, \( x \), of \( A \).
   - (a) \( x \) is initial segment: Then \( x \) has 1 “(‘ symbol and 0 “)” symbols.
   - (b) \( x \) is “(\neg”:\ Then \( x \) has 1 “(‘ symbol and 0 “)” symbols.
   - (c) \( x \) is “(\neg”\( z \), for some proper initial segment, \( z \), of \( B \): Since \( z \) has more “(‘ than “)” symbols, therefore so does \( x \).
   - (d) \( x \) is “(\neg”\( B \): Since \( B \) equally many “(‘ and “)” symbols, therefore \( x \) has more “(‘ than “)” symbols.
In every case, $x$ has more “(” than “)” symbols. Hence $\neg B$ has Property 2.

\begin{itemize}
\item 3: Because $B$ has Property 3, therefore by construction so does $\neg B$.
\end{itemize}

This shows that $A$ has Property R.

2. $A$ is $(B \star C)$, for some well-formed Propositional formulæ $B, C$ and some binary connective $\star$.

The inductive hypothesis is that each formula $B$ and $C$ has property $R$.

\begin{itemize}
\item 1: Clearly, $A$ has property 1.
\item 2: Since $B$ and $C$ have equal numbers of left and right parentheses, therefore so does $(B \star C)$.
\end{itemize}

For the second part of Property 2, we check these subcases for every possible proper initial segment, $x$, of $A$.

(a) $x$ is “(“: Then $x$ has 1 “(” symbol and 0 “)” symbols.

(b) $x$ is “(“$z$, for some proper initial segment, $z$, of $B$: Since $z$ has more “(” than “)” symbols, therefore so does $x$.

(c) $x$ is “(“$B$: Since $B$ equally many “(” and “)” symbols, therefore $x$ has more “(” than “)” symbols.

(d) $x$ is “(“$B \star$: Since $B$ equally many “(” and “)” symbols, therefore $x$ has more “(” than “)” symbols.

(e) $x$ is “(“$B \star z$, for some proper initial segment, $z$, of $C$: Since $B$ equally many “(” and “)” symbols, and $z$ has more “(” than “)” symbols, therefore $x$ has more “(” than “)” symbols.

(f) $x$ is “(“$B \star C$: Since $B$ and $C$ have equally many “(” and “)” symbols, therefore $x$ has more “(” than “)” symbols.

In every case, $x$ has more “(” than “)” symbols. Hence $\neg B$ has Property 2.

\begin{itemize}
\item 3: We must show
\end{itemize}

If $A$ is $(B' \star' C')$ for formulas $B'$ and $C'$, then $B = B'$, $\star = \star'$ and $C = C'$.

If $|B'| = |B|$, then $B' = B$ (both start at the second symbol of $A$). Thus also $\star = \star'$ and $C = C'$, as required. So we are finished if we can prove that $|B'| = |B|$.

\begin{itemize}
\item For a contradiction, assume that either $B'$ is a proper initial segment of $B$ or $B$ is a proper initial segment of $B'$.
\end{itemize}

\begin{itemize}
\item The inductive hypothesis applies to $B$ and $B'$. In particular, each has property 2.
\item Therefore $B$ and $B'$ have a balanced number of “(” and “)” characters, by property 2.
\end{itemize}

\begin{itemize}
\item But if $B$ is a proper initial segment of $B'$, then $B$ has more “(” than “)” characters, also by property 2. This is a contradiction.
\item We reach a similar contradiction if we assume that $B'$ is a proper initial segment of $B$. Thus neither $B$ nor $B'$ can be a proper initial segment of the other.
\end{itemize}

Therefore $A$ has a unique derivation; it has property 3, as required.

By the principle of structural induction, every Propositional formula has properties 1, 2 and 3.
This shows that Unique Readability (property 3) holds for every Propositional Formula. This is what we set out to prove.

**Explanation of the Connection Between $B$ and $B'$:**

- In the past, some students have been confused about why it holds that either $B = B'$, $B$ is a proper initial segment of $B'$ or vice versa.
- The key fact to remember here is that both $B$ and $B'$ arose from a choice of how to decompose the given formula $A$. In detail,

\[(B \star C) = A = (B' \star C').\]

- Because we actually mean equality of formulae (i.e. symbol-by-symbol equality of the expressions constituting the formulae ) here, we now see that the above fact about $B$ and $B'$ must hold.

### 3.4 Parse Trees

- A **parse tree** for a formula represents the formation sequence as a tree with its root at the top, and each internal node corresponding with an application of one of the formation rules.
- For example, this is a parse tree for the formula $A = ((p \land (\neg q)) \rightarrow r)$:

```
                      (p \land (\neg q)) \rightarrow r
                             /   \         \  \\
                         (p \land (\neg q))       r
                              /   \   \  \\
                          p               (\neg q)
                               |             /
                               q             q
```

- This parse tree has height 3.
- See also p24 of the text.
- Another typical question would be to provide a parse tree, and to ask for the formula that the tree represents.

### 4 Lecture 04

**Outline**

1. Cleanup from L03
   (a) Completing the Proof of Unique Readability
   (b) Generation Sequences for Propositional Formulae
   (c) Precedence Rules For Connectives
4.1 Cleanup from L03

4.1.1 Completing the Proof of Unique Readability

See the rest of the proof in L03.

4.1.2 Generation Sequences for Propositional Formulae

Examples:

1. Give a generation sequence for each of the following well-formed Propositional formulae over $P = \{p, q, r\}$.
   (a) $p$
   Solution:
   i. $p$ is in the core set.
   (b) $q$
   Solution:
   i. $q$ is in the core set.
   (c) $(p \land q)$
   Solution:
   i. $p$ is in the core set.
   ii. $q$ is in the core set.
   iii. Applying $\land$ to lines 1(c)i and 1(c)ii yields $(p \land q)$.
   (d) $(p \rightarrow (q \lor r))$
   Solution:
   i. $p$ is in the core set.
   ii. $q$ is in the core set.
   iii. $r$ is in the core set.
   iv. Applying $\lor$ to lines 1(d)ii and 1(d)iii yields $(q \lor r)$.
   v. Applying $impl$ to lines 1(d)i and 1(d)iv yields $(p \rightarrow (q \lor r))$.

4.1.3 Precedence Rules For Connectives

See the notes in L03.

4.1.4 Propositions

See the notes in L03.
4.1.5 Parse Trees - Correcting Notation

See the notes in L03.

4.2 Structural Induction - More Examples

Examples:

1. Setup:
   - Let \( A \) be the set \( \{(0, 1, 0)\} \).
   - Suppose that we can operate on \( A \) by flipping any two elements from 0 to 1 or from 1 to 0.

   **Problem:** Is it possible that any sequence of such flips applied to \( A \) yields the triple \( (0, 0, 0) \)?

   **Solution:** Let \( X = \{ \text{all triples of binary digits} \} \). Then consider \( I(X, A, F) \), where \( A = \{(0, 1, 0)\}, F = \{ \text{flip 1 and 2, flip 1 and 3, flip 2 and 3} \} \). The problem is then equivalent to asking
   
   Is \((0, 0, 0) \in I(X, A, F)\)?

   I claim that the answer is “No”. I need to prove my answer, by structural induction on \( I(X, A, F) \).

   For any binary triple \((x, y, z)\), define \( R(x, y, z) \) to be
   
   \((x, y, z)\) has an even number of 0 digits.

   We will prove by structural induction that every triple in \( I(X, A, F) \) has property \( R \).
   - **Base:** \( R(0, 1, 0) \) is clear.
   - **Induction:** Let \((x, y, z)\) be any binary triple having property \( R \). Then we check
     that each operation in \( F \) preserves property \( R \), via the following table:

<table>
<thead>
<tr>
<th>input triple</th>
<th>flip 1 and 2</th>
<th>flip 1 and 3</th>
<th>flip 2 and 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 1, 1)</td>
<td>(0, 0, 1)</td>
<td>(0, 1, 0)</td>
<td>(1, 0, 0)</td>
</tr>
<tr>
<td>(0, 0, 1)</td>
<td>(1, 1, 1)</td>
<td>(1, 0, 0)</td>
<td>(0, 1, 0)</td>
</tr>
<tr>
<td>(0, 1, 0)</td>
<td>(1, 0, 0)</td>
<td>(1, 1, 1)</td>
<td>(0, 0, 1)</td>
</tr>
<tr>
<td>(1, 0, 0)</td>
<td>(0, 1, 0)</td>
<td>(0, 0, 1)</td>
<td>(1, 1, 1)</td>
</tr>
</tbody>
</table>

   All the output triples have property \( R \). This completes the inductive step, and the proof.

   Now since \((0,0,0)\) does not have property \( R \), therefore \((0,0,0) \notin I(X, A, F)\).

   **Remarks:**
   - (a) This example provides a strategy for proving that an element of the universe is
     **not** a member of an inductively defined set:
       - i. Prove that all elements of the set have some property \( R \).
       - ii. Prove that the elements of interest does **not** have property \( R \).

   E.g. since all well-formed Propositional formulas have equally many “(’ and “)” symbols, therefore \( )p \rightarrow \land \leftrightarrow rq(\) is not a well-formed Propositional formula.
5 Lecture 05

Outline

1. Semantics of Propositional Logic
   (a) Truth Valuations
   (b) Evaluating Any Propositional Formula
   (c) Properties of Propositional Formulae

5.1 Semantics of Propositional Logic

Remarks:

1. Until now, we have been focussing (mostly) on syntax alone, i.e. on the question, “of all possible strings, which ones constitute well-formed Propositional formulæ?”.
2. Semantics is concerned with the question, “given some well-formed Propositional formula, \( A \), is \( A = 1 \) or \( 0 \) in some semantic context?”.
3. A semantic context in Propositional Logic means a choice of truth valuation, i.e. a choice, for each available Propositional variable, between 0 and 1. (See Definition 5.1.1.)
5. See pp19-20 for the truth tables of each connective.

5.1.1 Truth Valuations

Definition 5.1.1. A truth valuation \( t \) is a function from Propositional variables to \( \{0, 1\} \).

Notation: \( p^t \) denotes the value of \( p \) under \( t \).

In other words, a truth valuation, \( t \), is a function

\[
 t : \mathcal{P} \rightarrow \{0,1\},
\]

where \( \mathcal{P} \) denotes the set of available Propositional variables.

Examples:

1. Let \( \mathcal{P} = \{p, q, r\} \). Define the function

\[
 t : \{p, q\} \rightarrow \{0,1\} \quad \begin{align*}
 p & \mapsto 1 \\
 q & \mapsto 0
\end{align*}
\]

Q: Is \( t \) a truth valuation on \( \mathcal{P} \)? Why or why not?
A: No. \( t \) is not defined for \( r \in \mathcal{P} \), hence \( t \) is not a function \( t : \mathcal{P} \rightarrow \{0,1\} \).

2. Now let \( \mathcal{P} = \{p, q\} \). Define the function \( t \) as in the previous question. Q: Is \( t \) a truth valuation on \( \mathcal{P} \)? Why or why not?
A: Yes. \( t \) is a function \( t : \mathcal{P} \rightarrow \{0,1\} \).
5.1.2 Evaluating Any Propositional Formula

Fix a truth valuation \( t \). Then every formula \( C \) has a value under \( t \), denoted \( C^t \), defined inductively as follows.

1. \( p^t \) is given by the definition of \( t \), for every Propositional variable, \( t \).
2. \( (\neg A)^t = \begin{cases} 1 & \text{if } A^t = 0 \\ 0 & \text{if } A^t = 1 \end{cases} \)
3. \( (A \land B)^t = \begin{cases} 1 & \text{if } A^t = B^t = 1 \\ 0 & \text{otherwise} \end{cases} \)
4. \( (A \lor B)^t = \begin{cases} 1 & \text{if } A^t = 1 \text{ or } B^t = 1 \\ 0 & \text{otherwise} \end{cases} \)
5. \( (A \rightarrow B)^t = \begin{cases} 1 & \text{if } A^t = 0 \text{ or } B^t = 1 \\ 0 & \text{otherwise} \end{cases} \)
6. \( (A \leftrightarrow B)^t = \begin{cases} 1 & \text{if } A^t = B^t \\ 0 & \text{otherwise} \end{cases} \)

This handles every well-formed Propositional forumla unambiguously, because of Unique Readability (Theorem 3.3.1).

Examples:

1. Construct the truth table for each formula given.
   (a) \( (\neg (p \land q)) \).

   **Solution:**
   \[
   \begin{array}{c|c|c|c}
   p & q & (p \land q) & (\neg (p \land q)) \\
   \hline
   0 & 0 & 0 & 1 \\
   0 & 1 & 0 & 1 \\
   1 & 0 & 0 & 1 \\
   1 & 1 & 1 & 0 \\
   \end{array}
   \]

   (b) \( ((\neg p) \lor (\neg q)) \).

   **Solution:**
   \[
   \begin{array}{c|c|c|c|c|c|c|c}
   p & q & (\neg p) & (\neg q) & ((\neg p) \lor (\neg q)) \\
   \hline
   0 & 0 & 1 & 1 & 1 \\
   0 & 1 & 1 & 0 & 1 \\
   1 & 0 & 0 & 1 & 1 \\
   1 & 1 & 0 & 0 & 0 \\
   \end{array}
   \]

   **Remarks:**
   i. This formula is **not equal** to the previous one.
   ii. However this formula is **logically equivalent** to the previous one (Definition 5.1.5), i.e. it has the same truth table, i.e. it evaluates the same way under every possible truth valuation.
   iii. This particular **logically equivalence** \( (\neg (p \land q)) \iff ((\neg p) \lor (\neg q)) \) is an
example of a DeMorgan Law.

(c) \((\neg p) \lor q\).

Solution:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>(\neg p)</th>
<th>((\neg p) \lor q)</th>
<th>(p \rightarrow q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Remarks:

i. Note that \((p \rightarrow q) \iff ((\neg p) \lor q)\).

(d) \(((p \land q) \rightarrow r)\).

Solution:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>((p \land q))</th>
<th>(((p \land q) \rightarrow r))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

5.1.3 Properties of Propositional Formulae

Problems:

1. Let \(P\) be a set of Propositional variables. Let \(t\) be a truth valuation defined on \(P\). Let \(C\) be a well-formed Propositional formula. Prove by structural induction on \(C\) that \(C^t \in \{0, 1\}\).

Solution: For any well-formed Propositional formula, \(C\), define \(R(C)\) to be the statement

\[C^t \in \{0, 1\}\].

Base (\(C = p\), for some Propositional variable \(p \in P\)): Then \(C^t = p^t \in \{0, 1\}\) (i.e. \(R(C)\) holds).

Induction

\(C = (\neg A)\), for some well-formed Propositional formula, \(A\), then the induction hypothesis, \(R(A)\), says that \(A^t \in \{0, 1\}\). Therefore

\[C^t = (\neg A)^t = \begin{cases} 
0 & \text{if } A^t = 1 \\
1 & \text{if } A^t = 0 
\end{cases}\]

This shows that \(C^t \in \{0, 1\}\) (i.e. \(R(C)\) holds), completing this case.
If $C = (A \land B)$, for some well-formed Propositional formulae, $A$, $B$, then the induction hypotheses, $R(A)$ and $R(B)$, say that $A^t \in \{0, 1\}$ and $B^t \in \{0, 1\}$. Therefore

$$C^t = (A \land B)^t = \begin{cases} 1 & \text{if } A^t = B^t = 1 \\ 0 & \text{if } A^t = 0 \text{ or } B^t = 0 \text{ or both} \end{cases}$$

This shows that $C^t \in \{0, 1\}$ (i.e. $R(C)$ holds), completing this case.

The remaining cases for the binary connectives $\lor$, $\rightarrow$ and $\leftrightarrow$ are similar to the case for $\land$, and thus are omitted.

Thus, by the Principle of Structural Induction, $R(C)$ holds for every WFPF, $C$.

**Definition 5.1.2.** A well-formed Propositional formula $C$ is a **tautology** (aka **valid formula**) if $C^t = 1$, for every truth valuation $t$.

**Examples:**

1. $(p \lor (\neg p))$

**Definition 5.1.3.** A well-formed Propositional formula $C$ is **satisfiable** if $C^t = 1$, for some truth valuation $t$.

**Definition 5.1.4.** A well-formed Propositional formula $C$ is a **contradiction** (aka **not satisfiable**) if $C^t = 0$, for every truth valuation $t$.

**Examples:**

1. $(p \land (\neg p))$

**Remarks:**

1. Being satisfiable is the negation of being a contradiction.
2. Being a tautology is **not** the negation of being a contradiction.
3. Every tautology is satisfiable.

**Problems:**

1. For each of the Propositional formulas given below, determine with proof whether the formula is a contradiction (i.e. not satisfiable), satisfiable and not a tautology, or a tautology (i.e. a valid formula). Use truth tables and/or valuation trees to justify each answer.

(a) $C = (p \land (\neg p))$

**Solution:** The given formula has the following truth table.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$(\neg p)$</th>
<th>$(p \land (\neg p))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$C$ is a contradiction (not satisfiable) since there is no truth valuation in which the formula is true.
(b) \( C = (p \lor (\neg p)) \)

**Solution:** The given formula has the following truth table.

<table>
<thead>
<tr>
<th>( p )</th>
<th>( \neg p )</th>
<th>( p \lor (\neg p) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\( C \) is a tautology, since the formula is true in all truth valuations.

(c) \( C = ((p \rightarrow q) \rightarrow r) \)

**Solution 1:** The given formula has the following truth table.

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( r )</th>
<th>( (p \rightarrow q) )</th>
<th>( ((p \rightarrow q) \rightarrow r) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\( C \) is satisfiable since there is at least one truth valuation that makes \( C \) 1 (e.g. row 2).

\( C \) is not a tautology, since at least one truth valuation makes \( C \) 0 (e.g. row 1).

**Solution 2:** As the number of Propositional variables grows, it can become tedious to write down the entire truth table. We can instead solve the problem using a **valuation** tree.

- By properties of implication, a truth valuation \( t \) will make \( C^t = 1 \) if \( t(r) = 1 \). (N.B. This is “if”, not “if and only if”, as we can see from the truth table in Solution 1.) This shows that \( C \) is satisfiable.

- By properties of implication, a truth valuation \( t \) will make \( C^t = 0 \) if and only if \( (p \rightarrow q)^t = 1 \) and \( t(r) = 0 \). Again by properties of implication, a truth valuation \( t \) will make \( (p \rightarrow q)^t = 1 \) if \( t(q) = 1 \). (N.B. Again this is “if”, not “if and only if”.) This shows that \( C \) is not a tautology.

**Definition 5.1.5.** Two well-formed Propositional formulae \( A \) and \( B \) are called **(logically) equivalent** (denoted \( A \equiv B \)) if \( A^t = B^t \) for every truth valuation, \( t \). (Equivalently, if \( A \) and \( B \) have the same truth table.)

### 6 Lecture 06

**Outline**

1. Tautological Consequence
   - (a) Satisfaction of a Set of Formulae
   - (b) Definition of Tautological Consequence
   - (c) Subtleties About Tautological Consequence
6.1 Tautological Consequence

6.1.1 Satisfaction of a Set of Formulae

**Definition 6.1.1.** We say that a truth valuation, \( t \), satisfies a set \( \Sigma \), of well-formed Propositional formulae, (notation: \( \Sigma^t = 1 \)) if, for every \( C \in \Sigma \), we have \( C^t = 1 \).

**Problems:**

1. Analogously to a formula, we say that \( \Sigma \) is satisfiable if there exists a truth valuation \( t \) such that \( \Sigma^t = 1 \). Otherwise, we say that \( \Sigma \) is not satisfiable.
2. If there exists \( C \in \Sigma \) such that \( C^t = 0 \), then we say that \( t \) does not satisfy \( \Sigma \) (notation: \( \Sigma^t = 0 \)).
3. When we write \( \Sigma^t = 0 \), we are not asserting that every \( C \in \Sigma \) is made 0 under \( t \) - just that there is at least one such \( C \).

**Problems:**

1. Verify that \( \Sigma = \{((p \to q) \lor r), ((p \lor q) \lor s)\} \) is satisfiable.

   **Solution 1:** Let \( t \) be a truth valuation such that \( t(q) = 1 \). Then
   - \((p \to q)^t = 1\), so that \(((p \to q) \lor r)^t = 1\), and
   - \((p \lor q)^t = 1\), so that \(((p \lor q) \lor s)^t = 1\).
   This shows that \( t \) satisfies \( \Sigma \). N.B. This defines a family of truth valuations that work. There are \( 2^3 = 8 \) such truth valuations.

   **Solution 2:** Construct a joint truth table for \(((p \to q) \lor r)\) and \(((p \lor q) \lor s)\) and verify that at least one row has 1 in both of its final columns.

2. Is the set \( \Sigma = \{(p \to (p \land q)), ((\neg p) \lor (\neg q)), ((\neg p) \to p)\} \) satisfiable? Prove your answer.

   **Solution 1:** I claim that the set is not satisfiable. For a contradiction, suppose that there exists a truth valuation, \( t \), such that \( \Sigma^t = 1 \). Then
   - \((p \to (p \land q))^t = 1\), so we must have one of
     - \((p \land q)^t = 1\), which requires \( p^t = t(q) = 1 \). However this implies that \( ((\neg p) \lor (\neg q))^t = 0\), which contradicts \( \Sigma^t = 1 \), completing this case.
     - \( p^t = 0\), which implies \(((\neg p) \to p)^t = 0\), which contradicts \( \Sigma^t = 1 \), completing this case.

   **Solution 2:** Construct a joint truth table for \((p \to (p \land q)), ((\neg p) \lor (\neg q))\) and \(((\neg p) \to p)\) and verify that no row has 1 in all three of its final columns.
6.1.2 Definition of Tautological Consequence

Definition 6.1.2. We say that a set \( \Sigma \) of well-formed Propositional formulae \textit{logically implies} a well-formed Propositional formula \( C \) (denoted \( \Sigma \models C \)), if, whenever \( \Sigma^t = 1 \) for some truth valuation \( t \), it follows that \( C^t = 1 \). In this situation, we can equivalently say that \( C \) is a \textbf{tautological consequence} of \( \Sigma \).

Remarks:

1. The notion of \textbf{tautological consequence} will be a key ingredient in defining the \textbf{soundness} and \textbf{completeness} of our proof system(s), later on.
2. It is crucial to understand that Definition 6.1.2 asserts nothing in the situation where \( \Sigma^t = 0 \).
3. To prove that an tautological consequence does not hold, we must exhibit a choice of a truth valuation, \( t \), such that \( \Sigma^t = 1 \) and \( C^t = 0 \).

Problems:

1. Q: Is it true that \( \{ (p \land q) \} \models p \)?
   A: Yes. Any truth valuation, \( t \) such that \( \{ (p \land q) \}^t = 1 \) has \( (p \land q)^t = 1 \), and by properties of \( \land \), this requires \( p^t = 1 = q^t \). Hence \( t \) satisfies \( p \).
   (a) Q: Is it true that \( \{ (p \land q) \} \models r \)?
      A: No. Consider the truth valuation
      \[
      t : \{ p, q, r \} \rightarrow \{ 0, 1 \} \\
      p^t = 1 \\
      q^t = 1 \\
      r^t = 0
      \]
      Then we have that \( \{ (p \land q) \}^t = 1 \), and \( r^t = 0 \).
2. Q: Is it true that \( \{ (p \lor q) \} \models p \)?
   A: No. Consider the truth valuation
   \[
   t : \{ p, q \} \rightarrow \{ 0, 1 \} \\
   p^t = 0 \\
   q^t = 1
   \]
   Then we have that \( \{ (p \lor q) \}^t = 1 \), and \( p^t = 0 \).
3. Prove or disprove the tautological consequence \( \{ (p \to r), (q \to (\neg r)) \} \models (p \to (\neg q)) \).
   \textbf{Solution}: The truth table below shows the valuations of all of the formulæ involved.
   The lines marked with \( \checkmark \) are the ones for which all the assumptions \( \{ (p \to r), (q \to (\neg r)) \} \) are all true. In all such cases, the conclusion \( (p \to (\neg q)) \) is also true.
Thus the tautological consequence holds.

### 6.1.3 Subtleties About Tautological Consequence

1. Any statement like “For all \( x \in \emptyset \), \( x < 6 \)” is true. The empty set, \( \emptyset \), provides no counterexample \( x \in \emptyset \), such that \( x < 6 \) fails to hold.

2. By the same token, “For all \( C \in \emptyset \), \( C^t = 1 \)” is true, for any truth valuation \( t \).
   
   (a) It is important to understand that the negation of the statement in the previous bullet is “There exists \( C \in \emptyset \), such that \( C^t = 0 \)” (which is clearly not true), and not “For all \( C \in \emptyset \), \( C^t = 0 \)” (which is true, as above).

3. Applying Definition 6.1.1 to the empty set, \( \emptyset \), we see that the empty set is satisfied under any truth valuation, \( t \).

4. This shows by Definition 6.1.2 that if \( \emptyset \models C \), then \( C \) is a tautology.
   
   (a) Some authors will write “fresh air” instead of \( \emptyset \). I will always write \( \emptyset \) explicitly.

5. For the reverse implication, note that if \( C \) is a tautology, \( \Sigma \models C \), for any \( \Sigma \).

6. Applying Definition 6.1.2 in the case where \( \Sigma \) is not satisfiable, we see that \( \Sigma \models C \), for any \( C \).

7. An equivalent alternative to Definition 5.1.5, using Definition 6.1.2, is that

\[
A \equiv B \text{ if and only if } (\{A\} \models B \text{ and } \{B\} \models A).
\]

8. In class we developed this summary table about the subtleties of tautological consequence. Make sure that you understand the definition of tautological consequence, and you do not try to rely on this table alone!

<table>
<thead>
<tr>
<th>( \Sigma )</th>
<th>( C )</th>
<th>( \Sigma \models C )?</th>
</tr>
</thead>
<tbody>
<tr>
<td>not satisfiable</td>
<td>contradiction</td>
<td>yes</td>
</tr>
<tr>
<td>satisfiable</td>
<td>satisfiable, not a tautology</td>
<td>yes</td>
</tr>
<tr>
<td>satisfiable</td>
<td>contradiction</td>
<td>no</td>
</tr>
<tr>
<td>satisfiable</td>
<td>satisfiable, not a tautology</td>
<td>maybe</td>
</tr>
<tr>
<td>satisfiable</td>
<td>tautology</td>
<td>yes</td>
</tr>
</tbody>
</table>
6.1.4 Examples
Stuff.

6.2 Translations Between English and Propositional Logic

Remarks:

1. Choose your atomic propositions to be positive statements (i.e. without any embedded negations). E.g. translate “I did not change my fish’s tank filter” as \((\neg q)\), where \(q\) means “I changed my fish’s tank filter”.
2. Negated statements are compound, not atomic.

Translate the following examples from English into Well-Formed Formulae of Propositional Logic:

1. She is clever and she is hard working.
\((c \land h)\), where
   - \(c\): She is clever.
   - \(h\): She is hard working.
2. He is clever but he is not hard working.
\((c \land (\neg h))\), where
   - \(c\): He is clever.
   - \(h\): He is hard working.
3. If it rains, then he will be at home; otherwise he will go to the market or he will go to school.
\((r \rightarrow h) \land ((\neg r) \rightarrow (m \lor s)))\), where
   - \(r\): It rains.
   - \(h\): He will be at home.
   - \(m\): He will go to the market.
   - \(s\): He will go to school.
Note that we need \(\land\) and not \(\lor\) as the last binary connective in this formula!
4. The sum of two numbers is even if and only if both numbers are even or both numbers are odd.
\((e \leftrightarrow ((f \land g) \lor (p \land q)))\)
   - \(e\): The sum of the two numbers is even.
   - \(f\): The first number is even.
   - \(g\): The second number is even.
   - \(p\): The first number is odd.
   - \(q\): The second number is odd.

Translate the following examples from Well-Formed Formulae of Propositional Logic into English: Use the atoms:

- \(s\): Today is Sunday.
• $h$: I do homework.
• $t$: I watch TV.

1. $(h \leftrightarrow (\neg s))$
   • “I do homework if and only if today is not Sunday.”, or
   • “I do homework every day except Sunday.” (a bit more natural).

2. $(h \lor t)$
   • “I do homework or I watch TV.”
   • “I do homework unless I watch TV.” (N.B. “Unless” is messy in English - sometimes it indicates an inclusive or the way we have translated it here, and other times it indicates an exclusive or)

3. $(s \rightarrow t)$
   • “If today is Sunday then I watch TV.”, or
   • “I watch TV if today is Sunday.”, or
   • “Today is Sunday only if I watch TV.”

7 Lecture 07

Outline

1. Formal Deduction
   (a) Introduction to Proof Systems

7.1 Formal Deduction

7.1.1 Introduction to Proof Systems

A major goal of CS 245 is to develop a technique for rigourously checking whether proofs are correct or not (i.e. removing human understanding and intuition from the picture). For example, consider the following argument.

Given the premises:

1. If I study before my mid-term, and I get 8 hours of sleep before my mid-term, then I will pass my mid-term.
2. I studied before my mid-term.
3. I did not pass my mid-term.

We may conclude:

1. Therefore I must not have gotten 8 hours of sleep before my mid-term.

Q: Is this argument “convincing” to you?

A: I find this argument “convincing”. I will justify this assertion, soon.

Define these atomic propositions:
1. $p$: I studied before my mid-term.
2. $q$: I got 8 hours of sleep before my mid-term.
3. $r$: I passed my mid-term.

We then encode the above argument as follows. We will explain all the notation below, soon.

\[
\{((p \land q) \rightarrow r), p, (\neg r)\} \vdash \neg q.
\]

"proves"

We will see soon that the premises on the left of $\vdash$ are sufficient to prove $(\neg q)$, the formula on the right of $\vdash$. This is what I meant by “convincing”, above.

Here is a formal proof of this result, in the proof system of Formal Deduction.

(1) \((p \land q) \rightarrow r), p, (\neg r), q \vdash p\) (by \((\in)\))
(2) \((p \land q) \rightarrow r), p, (\neg r), q \vdash q\) (by \((\in)\))
(3) \((p \land q) \rightarrow r), p, (\neg r), q \vdash (p \land q)\) (by \((\land +)\), (1), (2))
(4) \((p \land q) \rightarrow r), p, (\neg r), q \vdash ((p \land q) \rightarrow r)\) (by \((\in)\))
(5) \((p \land q) \rightarrow r), p, (\neg r), q \vdash r\) (by \((\rightarrow \neg)\) (3), (4))
(6) \((p \land q) \rightarrow r), p, (\neg r), q \vdash (\neg r)\) (by \((\in)\))
(7) \((p \land q) \rightarrow r), p, (\neg r) \vdash (\neg q)\) (by \((\neg +)\), (5), (6))

Remarks:

1. A **proof** in Formal Deduction is a sequence of lines of the form $\Sigma \vdash B$, for some set $\Sigma$ and some well-formed Propositional formula $B$. Each line of the proof is written according to some **inference rule** of Formal Deduction, based on any of the lines already in evidence.
2. If we use our inference rules correctly, then each line asserts a correct fact about provability, from any given assumptions (i.e. suppositions) with which we started.
3. We are finished writing down our proof when the desired assertion appears on the last line.
4. An **inference rule** takes lines as inputs and returns a new line as output, according to predetermined rules.
5. The set of inference rules defines a **proof system**.
6. A proof is 100 % syntactic, and 0 % semantic.
7. **Notation:** $\Sigma \vdash_S C$ reads as “There is a proof, in proof system $S$, with premises $\Sigma$ and conclusion $C$”. Read $\vdash_S$ as “proves in $S$”.
8. The connection between $\vdash$ and $\vdash_S$ will come in the form of the **soundness** and **completeness** of the proof system $S$.
9. Since our only proof system in CS 245 will be Formal Deduction, we will not need to explicitly mention the proof system $S$. There exist other proof systems than Formal Deduction. Books that use other proof systems may need to explicitly mention the proof system for clarity.
8 Lecture 08

Outline

1. Formal Deduction
   (a) Formal Deduction - Inference Rules
   (b) Proof Examples

8.1 Formal Deduction

8.1.1 Formal Deduction - Inference Rules

The Formal Deduction proof system has one axiom (∈) (which can be proved from the basic rules - see p47 in the text), one Reflexive rule (Ref), one Addition of Premises Rule (+) and an introduction and an elimination rule for each Propositional connective.

1. Axiom (∈): If $A \in \Sigma$, then $\Sigma \vdash A$.
2. (Ref): $A \vdash A$.
3. (+):
   If $\Sigma_1 \vdash A$,
   then $\Sigma_1, \Sigma_2 \vdash A$.
4. (¬−):
   If $\Sigma, (\neg A) \vdash B$,
   $\Sigma, (\neg A) \vdash (\neg B)$,
   then $\Sigma \vdash A$.
5. (¬+):
   If $\Sigma, A \vdash B$,
   $\Sigma, A \vdash (\neg B)$,
   then $\Sigma \vdash (\neg A)$.
   This rule is also called Reductio Ad Absurdum (RAA). This rule need not be a basic rule, because it can be proved from the basic rules - see below. We will prove it, then include it as a basic rule for convenience. See also Theorem 2.6.5 [2] in the text.
6. (→ −):
   If $\Sigma \vdash (A \rightarrow B)$,
   $\Sigma \vdash A$,
   then $\Sigma \vdash B$.
7. (→ +):
   If $\Sigma, A \vdash B$,
   then $\Sigma \vdash (A \rightarrow B)$.
8. (∧ −):
   If $\Sigma \vdash (A \land B)$,
   then $\Sigma \vdash A$,
   $\Sigma \vdash B$. 
9. \((\land +)\):
   \[
   \text{If } \Sigma \vdash A, \\
   \Sigma \vdash B, \\
   \text{then } \Sigma \vdash (A \land B).
   \]

10. \((\lor -)\):
   \[
   \text{If } \Sigma, A \vdash C, \\
   \Sigma, B \vdash C, \\
   \text{then } \Sigma, (A \lor B) \vdash C.
   \]

11. \((\lor +)\):
   \[
   \text{If } \Sigma \vdash A, \\
   \text{then } \Sigma \vdash (A \lor B), \\
   \Sigma \vdash (B \lor A).
   \]

12. \((\leftrightarrow -)\):
   \[
   \text{If } \Sigma \vdash (A \leftrightarrow B), \\
   \Sigma \vdash A, \\
   \text{then } \Sigma \vdash B. \\
   \text{If } \Sigma \vdash (A \leftrightarrow B), \\
   \Sigma \vdash B, \\
   \text{then } \Sigma \vdash A.
   \]

13. \((\leftrightarrow +)\):
   \[
   \text{If } \Sigma, A \vdash B, \\
   \Sigma, B \vdash A, \\
   \text{then } \Sigma \vdash (A \leftrightarrow B).
   \]

Remarks:
1. As in the text, we will write \(\Sigma \vdash \Sigma'\) to mean that, for every \(C \in \Sigma'\), \(\Sigma \vdash C\).

8.1.2 Proof Examples

1. Proof of \((p \land q), (r \land s) \vdash (p \land s)\):
   \[
   \begin{align*}
   (1) & \quad (p \land q), (r \land s) \vdash (p \land q) \quad (\text{by } \in) \\
   (2) & \quad (p \land q), (r \land s) \vdash p \quad (\text{by } \land -, (1)) \\
   (3) & \quad (p \land q), (r \land s) \vdash (r \land s) \quad (\text{by } \in) \\
   (4) & \quad (p \land q), (r \land s) \vdash s \quad (\text{by } \land -, (3)) \\
   (5) & \quad (p \land q), (r \land s) \vdash (p \land s) \quad (\text{by } \land +, (2), (4))
   \end{align*}
   \]

2. Proof of \(A, (\neg A) \vdash B\) (Theorem 2.6.5 \[4\] in the text):
   \[
   \begin{align*}
   (1) & \quad A, (\neg A), (\neg B) \vdash A \quad (\text{by } \in) \\
   (2) & \quad A, (\neg A), (\neg B) \vdash (\neg A) \quad (\text{by } \in) \\
   (3) & \quad A, (\neg A) \vdash B \quad (\text{by } \neg-, (1), (2))
   \end{align*}
   \]

3. Proof of \((\neg A) \lor B \vdash (A \rightarrow B)\) (one direction of Theorem 2.6.9 \[5\] in the text):
4. Proof of \((A \rightarrow B) \vdash ((\neg A) \lor B)\) (other direction of Theorem 2.6.9 [5] in the text):

\begin{align*}
1. & \quad (A \rightarrow B), (\neg((\neg A) \lor B)), (\neg A) \vdash (\neg A) \quad \text{(by \((\neg)\))} \\
2. & \quad (A \rightarrow B), (\neg((\neg A) \lor B)), (\neg A) \vdash ((\neg A) \lor B) \quad \text{(by \((\lor +)\)), (1)} \\
3. & \quad (A \rightarrow B), (\neg((\neg A) \lor B)), (\neg A) \vdash (\neg((\neg A) \lor B)) \quad \text{(by \((\neg)\))} \\
4. & \quad (A \rightarrow B), (\neg((\neg A) \lor B)), A \vdash A \quad \text{(by \(\neg(\neg\))}, (2), (3)) \\
5. & \quad (A \rightarrow B), (\neg((\neg A) \lor B)), A \vdash (A \rightarrow B) \quad \text{(by \((\in)\))} \\
6. & \quad (A \rightarrow B), (\neg((\neg A) \lor B)), A \vdash B \quad \text{(by \((\rightarrow \neg)\)), (5), (6)) \\
7. & \quad (A \rightarrow B), (\neg((\neg A) \lor B)), A \vdash ((\neg A) \lor B) \quad \text{(by \((\lor +)\)), (7)) \\
8. & \quad (A \rightarrow B), (\neg((\neg A) \lor B)), A \vdash (\neg((\neg A) \lor B)) \quad \text{(by \((\neg)\))} \\
9. & \quad (A \rightarrow B), (\neg((\neg A) \lor B)) \vdash (\neg A) \quad \text{(by \(\neg(\neg)\)), (8), (9)) \\
10. & \quad (A \rightarrow B), (\neg((\neg A) \lor B)) \vdash ((\neg A) \lor B) \quad \text{(by \((\lor \neg)\)), (4), (10)) \\
\end{align*}

5. Theorem 2.6.2 in the text If \(\Sigma \vdash A\), then there exists a finite subset \(\Sigma_0 \subseteq \Sigma\) such that \(\Sigma_0 \vdash A\).

Proof. Exercise. By structural induction on \(\Sigma \vdash A\). The details are in the text. ■

6. Transitivity - Theorem 2.6.3 [2] in the text If \(\Sigma \vdash \Sigma'\) and \(\Sigma' \vdash A\), then \(\Sigma \vdash A\).

Proof. See the text. It is an exercise to write up the parts with hand-waving properly, using induction. ■

7. Proof of the DeMorgan Law \((\neg(A \land B)) \vdash ((\neg A) \lor (\neg B))\) (one direction of Theorem 2.6.9 [7] in the text):

\begin{align*}
1. & \quad (\neg(A \land B)) \vdash (A \rightarrow (\neg B)) \quad \text{(by Theorem 2.6.8 [5])} \\
2. & \quad (A \rightarrow (\neg B)) \vdash ((\neg A) \lor (\neg B)) \quad \text{(by Theorem 2.6.9 [5])} \\
3. & \quad (\neg(A \land B)) \vdash ((\neg A) \lor (\neg B)) \quad \text{(by Transitivity, (1), (2))} \\
\end{align*}

8. Proof of the DeMorgan Law \(((\neg A) \lor (\neg B)) \vdash (\neg(A \land B))\) (other direction of Theorem 2.6.9 [7] in the text):

\begin{align*}
1. & \quad (\neg A), (A \land B) \vdash (A \land B) \quad \text{(by \((\in)\))} \\
2. & \quad (\neg A), (A \land B) \vdash A \quad \text{(by \((\land \neg)\)), (1))} \\
3. & \quad (\neg A), (A \land B) \vdash (\neg A) \quad \text{(by \((\in)\))} \\
4. & \quad (\neg A) \vdash ((\neg A) \land B) \quad \text{(by \((\neg+)\)), (2), (3))} \\
5. & \quad (\neg B), (A \land B) \vdash (A \land B) \quad \text{(by \((\in)\))} \\
6. & \quad (\neg B), (A \land B) \vdash B \quad \text{(by \((\land \neg)\)), (1))} \\
7. & \quad (\neg B), (A \land B) \vdash (\neg B) \quad \text{(by \((\in)\))} \\
8. & \quad (\neg B) \vdash ((\neg A) \land B) \quad \text{(by \((\neg+)\)), (2), (3))} \\
9. & \quad ((\neg A) \lor (\neg B)) \vdash ((\neg A) \land B) \quad \text{(by \((\lor \neg)\)), (4), (8))} \\
\end{align*}

9. Proof of \((\neg(\neg A)) \vdash A\) (Theorem 2.6.5 [1] in the text):

\begin{align*}
1. & \quad (\neg(\neg A)), (\neg A) \vdash (\neg A) \quad \text{(by \((\in)\))} \\
2. & \quad (\neg(\neg A)), (\neg A) \vdash (\neg(\neg A)) \quad \text{(by \((\in)\))} \\
3. & \quad (\neg(\neg A)) \vdash A \quad \text{(by \((\neg \neg)\)), (1), (2))} \\
\end{align*}

Note the typo in the statement of this result in the text - the \(A\) is missing on the LHS.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>( \Sigma, A \vdash B )</td>
<td>(by supposition)</td>
</tr>
<tr>
<td>(2)</td>
<td>( \Sigma, (\neg(\neg A)) \vdash \Sigma )</td>
<td>(by ((\in)))</td>
</tr>
<tr>
<td>(3)</td>
<td>( (\neg(\neg A)) \vdash A )</td>
<td>(by Theorem 2.6.5 [1])</td>
</tr>
<tr>
<td>(4)</td>
<td>( \Sigma, (\neg(\neg A)) \vdash A )</td>
<td>(by (+), (3))</td>
</tr>
<tr>
<td>(5)</td>
<td>( \Sigma, (\neg(\neg A)) \vdash B )</td>
<td>(by (Tr) (2), (4), (1))</td>
</tr>
<tr>
<td>(6)</td>
<td>( \Sigma, A \vdash (\neg B) )</td>
<td>(by supposition)</td>
</tr>
<tr>
<td>(7)</td>
<td>( \Sigma, (\neg(\neg A)) \vdash \Sigma )</td>
<td>(by ((\in)))</td>
</tr>
<tr>
<td>(8)</td>
<td>( (\neg(\neg A)) \vdash A )</td>
<td>(by Theorem 2.6.5 [1])</td>
</tr>
<tr>
<td>(9)</td>
<td>( \Sigma, (\neg(\neg A)) \vdash A )</td>
<td>(by (+), (8))</td>
</tr>
<tr>
<td>(10)</td>
<td>( \Sigma, (\neg(\neg A)) \vdash (\neg B) )</td>
<td>(by (Tr) (7), (9), (6))</td>
</tr>
<tr>
<td>(11)</td>
<td>( \Sigma \vdash (\neg A) )</td>
<td>(by ((\neg)) (5), (10))</td>
</tr>
</tbody>
</table>

9 Lecture 09

Outline

1. The Soundness of Formal Deduction
   (a) How to Prove a Tautological Consequence in General
   (b) Soundness

9.1 The Soundness of Formal Deduction

9.1.1 How to Prove a Tautological Consequence in General

To prove \( \Sigma \vdash C \), let \( t \) be any truth valuation such that \( \Sigma^t = 1 \), then prove that \( C^t = 1 \).

9.1.2 Soundness

Theorem (Soundness of Propositional FD) 9.1.1. If \( \Sigma \vdash C \), then \( \Sigma \models C \).

Proof.  
- By structural induction on \( \Sigma \vdash C \), where the set of proofs is defined as \( I(X, A, F) \), with
  - \( X \) is all sequences of lines of the form: \( \Sigma \vdash C \) (justification).
  - \( A \) is all single lines of the form: \( \Sigma \vdash C \) (by (\(\in\))).
  - \( F \) is the remaining 12 inference rules of formal deduction.
- **Base:** \( \Sigma \vdash C \) with \( C \in \Sigma \). Any truth valuation \( t \) such that \( \Sigma^t = 1 \) necessarily makes \( C^t = 1 \). This shows that \( \Sigma \models C \) in the base case.
- **Induction:** As we are not in the base case, we have these cases for the last inference rule used in the proof \( \Sigma \vdash C \).
  2. (Ref): \( A \vdash A \) is clear.
  3. (+):
     - The induction hypothesis is \( \Sigma_1 \models A \).
4. ($\neg\neg$):
   - The induction hypothesis is $\Sigma \cup \{\neg A\} \models B$ and $\Sigma \cup \{\neg A\} \models \neg B$.
   - Towards a contradiction, suppose that there exists a truth valuation $t$ such that $\Sigma^t = 1$ and $A^t = 0$.
   - Then by $\neg$-rules, $(\neg A)^t = 1$.
   - Then by the induction hypothesis and the definition of tautological consequence, we have $B^t = 1$ and $(\neg B)^t = 1$.
   - By $\neg$-rules, we have $B^t = 1$ and $B^t = 0$.
   - This contradiction completes the proof of this case.

5. ($\neg+$):
   - The induction hypothesis is $\Sigma \cup \{A\} \models B$ and $\Sigma \cup \{A\} \models \neg B$.
   - Towards a contradiction, suppose that there exists a truth valuation $t$ such that $\Sigma^t = 1$ and $(\neg A)^t = 0$.
   - Then by $\neg$-rules, $A^t = 1$.
   - Then by the induction hypothesis and the definition of tautological consequence, we have $B^t = 1$ and $(\neg B)^t = 1$.
   - By $\neg$-rules, we have $B^t = 1$ and $B^t = 0$.
   - This contradiction completes the proof of this case.

6. ($\rightarrow -$):
   - The induction hypothesis is $\Sigma \models (A \rightarrow B)$ and $\Sigma \models A$.
   - Let $t$ be any truth valuation such that $\Sigma^t = 1$.
   - Then by the definition of tautological consequence, we have $(A \rightarrow B)^t = 1 = A^t$.
   - Thus by the $\rightarrow$-rules, $B^t = 1$.

7. ($\rightarrow +$):
   - The induction hypothesis is $\Sigma \cup \{A\} \models B$.
   - Let $t$ be any truth valuation such that $\Sigma^t = 1$. We have these two cases for $A^t$.
     * If $A^t = 0$, then by the $\rightarrow$-rules, we have $(A \rightarrow B)^t = 1$.
     * If $A^t = 1$, then $(\Sigma \cup \{A\})^t = 1$, so that, by the definition of tautological consequence, we have $B^t = 1$. Then by the $\rightarrow$-rules, we have $(A \rightarrow B)^t = 1$.
   - In either case we have $(A \rightarrow B)^t = 1$.

8. ($\land -$):
   - The induction hypothesis is $\Sigma \models (A \land B)$.
   - Let $t$ be any truth valuation such that $\Sigma^t = 1$.
   - Then by the definition of tautological consequence, we have $(A \land B)^t = 1$.
   - Then by the $\land$-rules, we have $A^t = 1 = B^t$.

9. ($\land +$):
The induction hypothesis is \( \Sigma \models A \) and \( \Sigma \models B \).

Let \( t \) be any truth valuation such that \( \Sigma^t = 1 \).

Then by the definition of tautological consequence, we have \( A^t = B^t = 1 \).

Then by the \( \wedge \)-rules, we have \( (A \land B)^t = 1 \).

10. \((\lor -)\):

The induction hypothesis is \( \Sigma \cup \{ A \} \models C \) and \( \Sigma \cup \{ B \} \models C \).

Let \( t \) be any truth valuation such that \( (\Sigma \cup (A \lor B))^t = 1 \).

Then since \( (A \lor B)^t = 1 \), it follows by \( \lor \)-rules that \( A^t = 1, B^t = 1 \), or both.

* If \( A^t = 1 \), then \( (\Sigma \cup \{ A \})^t = 1 \), so that, by the definition of tautological consequence, we have \( C^t = 1 \).

* If \( B^t = 1 \), then \( (\Sigma \cup \{ B \})^t = 1 \), so that, by the definition of tautological consequence, we have \( C^t = 1 \).

In either case, we have \( C^t = 1 \).

11. \((\lor +)\):

The induction hypothesis is \( \Sigma \models A \).

Let \( t \) be a truth valuation such that \( \Sigma^t = 1 \).

Then by the induction hypothesis, we have that \( A^t = 1 \).

By \( \lor \)-rules, \( (A \lor B)^t = 1 = (B \lor A)^t \).

12. \((\leftrightarrow -)\):

(a) The induction hypothesis is \( \Sigma \models (A \leftrightarrow B) \) and \( \Sigma \models A \).

Let \( t \) be a truth valuation such that \( \Sigma^t = 1 \).

Then by the definition of tautological consequence, we have \( (A \leftrightarrow B)^t = 1 = A^t \).

Then by \( \leftrightarrow \) properties, \( B^t = 1 \).

(b) The induction hypothesis is \( \Sigma \models (A \leftrightarrow B) \) and \( \Sigma \models B \).

Let \( t \) be a truth valuation such that \( \Sigma^t = 1 \).

Then by the definition of tautological consequence, we have \( (A \leftrightarrow B)^t = 1 = B^t \).

Then by \( \leftrightarrow \) properties, \( A^t = 1 \).

13. \((\leftrightarrow +)\):

The induction hypothesis is \( \Sigma \cup \{ A \} \models B \) and \( \Sigma \cup \{ B \} \models A \).

Let \( t \) be a truth valuation such that \( \Sigma^t = 1 \). We have these subcases:

(a) \( A^t = 0 = B^t \). Then \( (A \leftrightarrow B)^t = 1 \).

(b) \( A^t = 0 \) and \( B^t = 1 \). Then \( (\Sigma \cup \{ B \})^t = 1 \), so that by the definition of tautological consequence, \( A^t = 1 \). This is a contradiction, so this case cannot occur.

(c) \( A^t = 1 \) and \( B^t = 0 \). Then \( (\Sigma \cup \{ A \})^t = 1 \), so that by the definition of tautological consequence, \( B^t = 1 \). This is a contradiction, so this case cannot occur.

(d) \( A^t = 1 = B^t \). Then \( (A \leftrightarrow B)^t = 1 \).

All cases are now handled. By the principle of structural induction, we are finished.
Soundness - Remarks on the Proof

- This proof works even if $\Sigma$ is infinite (even uncountable).
- But because any individual proof includes finitely many steps, it can have only finitely many premises.
- So every proof can be written using a finite subset $\Sigma_0 \subset \Sigma$, even if $\Sigma$ is infinite.

Applications of Soundness and Completeness

1. **Problem:** Prove that $\{ (A \rightarrow B) \} \not\models (B \rightarrow A)$.
   **Solution:** The contrapositive of soundness is: If $\Sigma \not\models C$, then $\Sigma \not\vdash C$. For a counterexample, let $A = p, B = q$. Then the valuation $p = 0, q = 1$ witnesses that $\{ (A \rightarrow B) \} \not\models (B \rightarrow A)$. So we are done.

2. **Problem:** Let $\Sigma, C$ satisfy $\Sigma \models C$. Does it follow that $\Sigma \models (\neg C)$?
   **Solution:** No. For example, let $\Sigma = \{ C, (\neg C) \}$. More generally, let $\Sigma$ be any inconsistent set.

3. Similarly, note that, as earlier, an inconsistent set proves any formula.

10 Lecture 10

Outline

1. The Completeness of Formal Deduction
   (a) Useful Ingredients
   (b) Proof of the Completeness Theorem

10.1 The Completeness of Formal Deduction

10.1.1 Useful Ingredients

Definitions

**Definition 10.1.1.** Let $\Sigma$ be a set of well-formed Propositional formulae. We call $\Sigma$ **consistent** if there exists a formula $B$ such that $\Sigma \not\models B$.

**Definition 10.1.2.** Let $\Sigma$ be a set of well-formed Propositional formulae. We call $\Sigma$ **consistent** if, for every Propositional formula $A$, if $\Sigma \models A$ then $\Sigma \not\models (\neg A)$.

**Theorem 10.1.3.** Definitions **10.1.1** and **10.1.2** are equivalent.

**Proof.**
- For the forward direction, suppose that there exists a formula, $B$, such that $\Sigma \not\models B$.
  - Let $A$ be any formula such that $\Sigma \models A$.
  - Towards a contradiction, suppose that $\Sigma \models (\neg A)$.
Then we have

1. \( \Sigma \vdash A, (\neg A) \) (by supposition)
2. \( A, (\neg A) \vdash B \) (by Theorem 2.6.5 [4])
3. \( \Sigma \vdash B \) (by (Tr), (1), (2))

This contradiction completes this direction.

• For the backward direction, suppose that there for every formula, \( A \), if \( \Sigma \vdash A \) then \( \Sigma \not\vdash (\neg A) \).
  
  – Towards a contradiction, suppose that \( \Sigma \vdash B \), for every formula \( B \).
  
  – Let \( A \) be any formula. By our above assumption, \( \Sigma \vdash A \) and \( \Sigma \vdash (\neg A) \).
  
  – This contradiction with the assumption completes the proof in this direction.

\[\text{Definition 10.1.4. Let } \Sigma \text{ be a set of well-formed Propositional formulae. We call } \Sigma \text{ inconsistent if } \Sigma \text{ is not consistent.}\]

\[\text{Lemmas}\]

\[\text{Lemma 10.1.5. Let } \Sigma \text{ be a set of well-formed Propositional formulae. Let } A \text{ be a Propositional formula. Then } \Sigma \vdash A \text{ if and only if } \Sigma \cup \{(\neg A)\} \text{ is unsatisfiable.}\]

\[\text{Proof. For the forward direction, assume that } \Sigma \vdash A \text{. Let } t \text{ be any truth valuation. We have these cases for } \Sigma^t.\]

  • If \( \Sigma^t = 1 \), then because \( \Sigma \vdash A \), it follows that \( A^t = 1 \). Hence \( (\neg A)^t = 0 \). Therefore \( \Sigma \cup \{(\neg A)\}^t = 0 \).
  
  • If \( \Sigma^t = 0 \), then \( \Sigma \cup \{(\neg A)\}^t = 0 \).

In either case, \( \Sigma \cup \{(\neg A)\}^t = 0 \). Since \( t \) was arbitrary, this shows that \( \Sigma \cup \{(\neg A)\} \) is unsatisfiable.

For the backward direction, assume that \( \Sigma \cup \{(\neg A)\} \) is unsatisfiable. Let \( t \) be a truth valuation such that \( \Sigma^t = 1 \). We have these cases for \( A^t \).

  • If \( A^t = 0 \), then \( (\neg A)^t = 1 \). But then \( \Sigma \cup \{(\neg A)\}^t = 1 \). This contradicts the fact that \( \Sigma \cup \{(\neg A)\} \) is unsatisfiable, and so this case cannot occur.
  
  • The only remaining possibility, namely that \( A^t = 1 \), must occur.

This proves that \( \Sigma \vdash A \). ■

\[\text{Lemma 10.1.6. } \Sigma \vdash A \text{ if and only if } \Sigma \cup \{(\neg A)\} \text{ is inconsistent.}\]

\[\text{Proof. For the forward direction, assume that } \Sigma \vdash A \text{. Let } B \text{ be any Propositional formula. Then we have}\]

34
(1) $\Sigma \vdash A$ (by supposition)
(2) $\Sigma, (\neg A) \vdash A$ (by $(+), (1)$)
(3) $\Sigma, (\neg A) \vdash (\neg A)$ (by $(\epsilon)$)
(4) $A, (\neg A) \vdash B$ (by Theorem 2.6.5 [4])
(5) $\Sigma, (\neg A) \vdash B$ (by $(T r) (2), (3), (4)$)

This shows that $\Sigma \cup \{ (\neg A) \}$ violates Definition 10.1.1 of being consistent.

For the backward direction, assume that $\Sigma \cup \{ (\neg A) \}$ is inconsistent. Let $B$ be any Propositional formula. Then we have

(1) $\Sigma, (\neg A) \vdash B$ (by negation of Definition 10.1.1)
(2) $\Sigma, (\neg A) \vdash (\neg B)$ (by negation of Definition 10.1.1)
(3) $\Sigma \vdash A$ (by $(\neg\neg), (1), (2)$)

10.2 Proof of the Completeness Theorem

Theorem (Completeness of FD) 10.2.1. If $\Sigma \vDash A$, then $\Sigma \vdash A$.

Proof. • It suffices to prove that if $\Sigma$ is consistent, then $\Sigma$ is satisfiable. Why?
  – This is because the contrapositive of this statement (replacing $\Sigma$ with $\Sigma \cup \{ (\neg A) \}$ throughout) is: if $\Sigma \cup \{ (\neg A) \}$ is usatisfiable, then $\Sigma \cup \{ (\neg A) \}$ is inconsistent.
  – By Lemma 10.1.6, we can re-write this as: if $\Sigma \cup \{ (\neg A) \}$ is usatisfiable, then $\Sigma \vdash A$.
  – Rewriting this using Lemma 10.1.5, we get: if $\Sigma \vDash A$, then $\Sigma \vdash A$, the exact statement of Theorem 10.2.1.

• So to prove the Theorem, we will prove that every consistent set is satisfiable.

  • Let $\Sigma$ be an arbitrary consistent set.
  • WLOG, assume that all the formulæ in $\Sigma$ are constructed using only $\{ \neg, \land, \lor \rightarrow \}$ (i.e. no $\leftrightarrow$ connectives, which can be replaced using $\rightarrow$ and $\land$ whenever needed). Once we finish the proof, it will be clear how to include the $\leftrightarrow$ connective if you want.
  • Suppose that $\Sigma$ is countable (i.e. $\Sigma$ is finite or we can write $\Sigma = \{ C_0, C_1, C_2, \ldots \}$), and assume that we can write a sequence of all the well-formed formulæ:

  $$A_0, A_1, \ldots, A_i, \ldots$$

• Now define

$$\Sigma_0 = \Sigma$$
$$\Sigma_{i+1} = \begin{cases} 
\Sigma_i \cup \{ A_i \} & \text{if } \Sigma_i \cup \{ A_i \} \text{ is consistent} \\
\Sigma_i & \text{otherwise}
\end{cases}, i \geq 0$$

• Observe that each $\Sigma_i$ is consistent by its construction. (Exercise: Prove it, by induction on $i \geq 0$.)
Remark: The assumptions about countability are too strong. \( \Sigma \) may not be countable. We could fix this using transfinite induction, which is beyond the scope of this course.

Let

\[ M = \bigcup_{i=0}^{\infty} \Sigma_i. \]

We use \( M \) to denote a “monster”.

Observe that \( M \) is consistent, by its construction. (Exercise: Prove it. If not, then some \( \Sigma_i \) is inconsistent, contradicting an earlier observation.)

The idea behind the construction of \( M \) is that \( M \) should be the largest possible set that both

- is consistent, and
- contains \( \Sigma \).

Define a truth valuation \( t \) via \( p^t = 1 \) if and only if \( p \in M \). This is a truth valuation, because every atomic formula either lies in \( M \) or lies outside of \( M \).

I claim that \( M^t = 1 \), i.e. that \( M \) is satisfiable.

This is enough since \( M \supseteq \Sigma \) by construction.

Let \( C \) be an arbitrary Propositional formula.

Define \( R(C) \) to be the property that \( C^t = 1 \) if and only if \( C \in M \). This is a truth valuation, because every atomic formula either lies in \( M \) or lies outside of \( M \).

We will prove that \( R(C) \) holds for every well-formed Propositional formula \( C \). The proof is by structural induction on \( C \).

We make some useful observations before giving the body of the proof.

Observation #1: For any formula \( A \), \( M \) contains \( A \) or \( (\neg A) \) and not both (since \( M \) is consistent by construction).

Observation #2: For any formula \( A \), if \( M \vdash A \), then \( A \in M \).

- By Observation #1, either \( A \in M \), or \( (\neg A) \in M \), and not both.
- Towards a contradiction, suppose that \( (\neg A) \in M \).
- Then \( M \vdash (\neg A) \).
- Since \( M \vdash A \) and \( M \vdash (\neg A) \), therefore \( M \vdash B \) for any Propositional formula \( B \).
- This shows that \( M \) is inconsistent.
- This contradiction completes the proof of this observation.

Observation #3: If \( B \in M \), then \( (A \rightarrow B) \in M \), for any \( A \) and \( (A \rightarrow B) \in M \), then \( B \in M \). Proof:

- \( A \in M \), so \( M \vdash A \).
- \( (A \rightarrow B) \in M \), so \( M \vdash (A \rightarrow B) \).
- By \( (\rightarrow -) \), \( M \vdash B \).
- By Observation #2, \( B \in M \).

Observation #4: If \( B \in M \), then \( (A \rightarrow B) \in M \), for any \( A \). Proof:

- Let \( A \) be arbitrary.
- Let \( B \in M \).
- By \( (\in) \), \( M \vdash B \).
- By \( (+) \), \( M \cup \{A\} \vdash B \).
- By → +, $M \vdash (A \rightarrow B)$.
- By Observation #2, $(A \rightarrow B) \in M$.

- Observation #5: If $A \notin M$, then $(A \rightarrow B) \in M$, for any $B$. Proof:
  - Let $B$ be arbitrary.
  - Let $A \notin M$.
  - By Observation #1, $(\neg A) \in M$.
  - By $(\in)$, $M \vdash (\neg A)$.
  - By $(\in)$, $M \cup \{A\} \vdash (\neg A)$.
  - By $(\in)$, $M \cup \{A\} \vdash A$.
  - Then $M \cup \{A\}$ is inconsistent.
  - Therefore $M \cup \{A\} \vdash B$.
  - Hence by $(\rightarrow +)$, $M \vdash (A \rightarrow B)$.
  - By Observation #2, $(A \rightarrow B) \in M$.

- Base ($C = p$ for some Propositional variable $p$):
  - By the construction of $t$, we then have $C^t = p^t$ which equals 1 if and only if $C \in M$.

- Induction We have the following sub-cases depending on the construction of $C$.
  - $C$ is $(\neg A)$, for some $A$:
    * The induction hypothesis is that $A^t = 1$ if and only if $A \in M$.
    * For the forward direction, assume that $C^t = 1$, i.e. $(\neg A)^t = 1$.
      - Then, by $\neg$-properties, we have that $A^t = 0$.
      - By the induction hypothesis, we have that $A \notin M$.
      - By Observation #1, we have $(\neg A) \in M$.
    * For the backward direction, assume that $C \in M$, i.e. $(\neg A) \in M$.
      - By Observation #1, we have that $A \notin M$.
      - By the induction hypothesis, we have that $A^t = 0$.
      - Then, by $\neg$-properties, we have that $(\neg A)^t = 1$.
  - $C$ is $(A \land B)$, for some $A, B$:
    * The induction hypothesis is
      - $A^t = 1$ if and only if $A \in M$, and
      - $B^t = 1$ if and only if $B \in M$.
    * For the forward direction, assume that $C^t = 1$, i.e. $(A \land B)^t = 1$.
      - By $\land$-properties, we have $A^t = 1$ and $B^t = 1$.
      - By the induction hypothesis, we have $A \in M$ and $B \in M$.
      - By $\land +$, we have $M \vdash (A \land B)$.
      - By Observation #2, we have $(A \land B) \in M$.
    * For the backward direction, assume that $C \in M$, i.e. $(A \land B) \in M$.
      - By two applications of $\land -$ , we have $M \vdash A$ and $M \vdash B$.
      - By two applications of Observation #2, we have $A \in M$ and $B \in M$.
      - By the induction hypothesis, we have $A^t = 1$ and $B^t = 1$.
      - By $\land$-properties, we have $(A \land B)^t = 1$.
  - $C$ is $(A \lor B)$, for some $A, B$: 

37
The induction hypothesis is
- \( A^t = 1 \) if and only if \( A \in M \), and
- \( B^t = 1 \) if and only if \( B \in M \).

For the forward direction, assume that \( C^t = 1 \), i.e. \( (A \lor B)^t = 1 \).
- By \( \lor \)-properties, we have \( A^t = 1 \) or \( B^t = 1 \), or both.
- If \( A^t = 1 \), then by the induction hypothesis, we have \( A \in M \).
- By \( \lor + \), we have \( M \vdash (A \lor B) \).
- By Observation #2, we have \( (A \lor B) \in M \).
- If \( B^t = 1 \), then the proof that \( (A \lor B) \in M \) is similar.

For the backward direction, assume that \( C \in M \), i.e. \( (A \lor B) \in M \).
- We are finished if we can prove that \((A \lor B)^t = 1\).
- Towards a contradiction, suppose that \( A^t = 0 \) and \( B^t = 0 \).
- By the induction hypothesis, we have \( A \notin M \) and \( B \notin M \).
- By Observation #1, we have \( (\neg A) \in M \) and \( (\neg B) \in M \).
- By \( \land + \), \( M \vdash ((\neg A) \land (\neg B)) \).
- By Observation #2, \( ((\neg A) \land (\neg B)) \in M \).
- By Theorem 2.6.9 [6], \( M \vdash (\neg(A \lor B)) \).
- This contradicts the fact that \( M \) is consistent.
- Therefore we have \( (A \lor B)^t = 1 \), as desired.

\( C \) is \((A \rightarrow B)\), for some \( A, B \):
- The induction hypothesis is
  - \( A^t = 1 \) if and only if \( A \in M \), and
  - \( B^t = 1 \) if and only if \( B \in M \).
- For the forward direction, assume that \( C^t = 1 \), i.e. \((A \rightarrow B)^t = 1\).
  - Thus \( B^t = 1 \) or \( A^t = 0 \).
  - If \( B^t = 1 \), then by induction \( B \in M \). By Observation #4, \( (A \rightarrow B) \in M \).
  - If \( A^t = 0 \), then by induction \( A \notin M \). By Observation #5, \( (A \rightarrow B) \in M \).
- For the backward direction, assume that \( C \in M \), i.e. \((A \rightarrow B) \in M \).
  - If \( A \in M \), then by induction \( A^t = 1 \). By Observation #3, \( B \in M \). By induction, \( B^t = 1 \). By the properties of \( \rightarrow \), we have \((A \rightarrow B)^t = 1\).
  - If \( A \notin M \), then by induction \( A^t = 0 \). By the properties of \( \rightarrow \), we have \((A \rightarrow B)^t = 1\).

\section{Lecture 11}

Outline

1. Soundness Revisited
2. Introduction to Predicate Logic
11.1 Soundness Revisited

Here is a different but equivalent statement of the Soundness Theorem.

**Theorem (Soundness of Propositional FD) 11.1.1.** If $\Sigma$ is satisfiable, then $\Sigma$ is consistent.

**Theorem 11.1.2.** Theorems 9.1.1 and 11.1.1 are equivalent.

**Proof.**
- For the forward direction, assume that $\Sigma$ satisfiable implies $\Sigma$ consistent.
  - The contrapositive, replacing $\Sigma$ by $\Sigma \cup \{(\neg A)\}$ throughout, is that $\Sigma \cup \{(\neg A)\}$ is inconsistent implies that $\Sigma \cup \{(\neg A)\}$ is unsatisfiable.
  - Via Lemma 10.1.5, we can rewrite this as $\Sigma \cup \{(\neg A)\}$ is inconsistent implies that $\Sigma \models A$.
  - Via Lemma 10.1.6, we can rewrite this as $\Sigma \vdash A$ implies that $\Sigma \not\models A$.
- For the backward direction, assume that $\Sigma \vdash A$ implies $\Sigma \models A$.
  - Via Lemma 10.1.6, we can rewrite this as $\Sigma \cup \{(\neg A)\}$ is inconsistent implies $\Sigma \models A$.
  - Via Lemma 10.1.5, we can rewrite this as $\Sigma \cup \{(\neg A)\}$ is inconsistent implies $\Sigma \cup \{(\neg A)\}$ is unsatisfiable.
  - The contrapositive (replacing $\Sigma \cup \{(\neg A)\}$ by $\Sigma$ throughout) is $\Sigma$ is satisfiable implies $\Sigma$ is consistent.

11.2 Introduction to Predicate Logic

**Remarks:**

1. Propositional Logic is simple and nice, but as a consequence, it is a bit limiting.
2. E.g. consider this argument:
   - **Premises**
     - All humans are mortal.
     - I am a human.
   - **Conclusion** Therefore I am mortal.
3. This argument “makes sense”.
4. However it can not be adequately expressed, much less proved, in Propositional Logic.
5. One reason: The first premise needs a $\forall$ quantifier to express it correctly.
6. Adding quantifiers is one major enhancement that we make when we move from Propositional Logic to Predicate Logic.
7. In the early going we will keep things informal; we will formalize later.
8. Predicate logic extends Propositional logic. Everything we already know about Propositional logic still works in the new setting.
9. Two key ingredients in defining Predicate logic are the **quantifiers** $\forall$ and $\exists$, which you met in Math 135. Quantifiers have the same meanings here - we will formalize
their use so that we can be rigorous.

Examples:

1. “For all integers adding 0 returns the same integer”, or “For every integer \( x, x + 0 = x \)” could translate to Predicate logic as

\[
(\forall x (x + 0 = x))
\]

Question from the Class: Where did “\( x \) is an integer” go in this translation?

Answer:
(a) We can work in the universe of integers, \( \mathbb{Z} \). In this universe, this translation tells the whole story.
(b) If the universe also contains non-integers (e.g. \( \mathbb{Q}, \mathbb{R} \) or \( \mathbb{C} \)), then we need to enhance our translation to express that \( x \) is an integer. Define

\[
I(x) = \begin{cases} 
1 & \text{if } x \text{ is an integer} \\
0 & \text{otherwise} 
\end{cases}
\]

Think of \( I(x) \) as the assertion “\( x \) is an integer”. With this notation, we can re-translate as

\[
(\forall x (I(x) \rightarrow (x + 0 = x))).
\]

Note, the following incorrect suggestion was made in the past:

\[
(\forall x (I(x) \land (x + 0 = x))).
\]

Make certain that you understand why this is an incorrect translation of the given English statement.

2. The Predicate formula

\[
(\forall x (x \cdot 1 = x))
\]

or, if the universe could contain non-integers,

\[
(\forall x (I(x) \rightarrow (x \cdot 1 = x)))
\]

could translate to English as “For every integer \( x, x \cdot 1 = x \)” or “for every integer, multiplying by 1 returns the same integer”.

For these examples, a natural semantic choice (where all symbols have their usual meanings) is

- Domain (a.k.a. “Universe”) \( \mathbb{Z} \)
- **Remark:** In Predicate Logic, individual, function and predicate symbols have **no intrinsic meanings**. These symbols get their meanings in some semantic context; in different semantic contexts, a symbol can get **different meanings**.
- Constant symbols \{0, 1\}
• Function symbols \( \{+^{(2)}, \cdot^{(2)}\} \)
• Predicate symbols \( \{I^{(1)}, =^{(2)}\} \) (we could easily add <, ≤, etc.)

**Question from the Class:** Why does “=” belong to the predicate symbols and not to the function symbols?

**Answer:** Fundamentally,

1. A function takes a tuple of domain elements and returns a new domain element as its output.
2. A predicate takes a tuple of domain elements and returns 0 or 1 as its output. A predicate stands for a statement, which is either 0 or 1 in some semantic context. We use “=” in the sense of comparison, not in the sense of assignment. As a comparison, “=” is clearly a predicate symbol and not a function symbol.

Clicker Questions:

1. CQ 7

### 12 Lecture 12

**Outline**

1. Informal Definitions
2. Ingredients of Predicate Logic
3. Syntax of Predicate Logic
4. Translations from English into Predicate Logic
5. Translations from Predicate Logic into English

#### 12.1 Informal Definitions

**Definition 12.1.1.** Let \( X \) be a non-empty set. Let \( k \geq 1 \). A k-ary predicate (a.k.a. relation) on \( X \) is any set of \( k \)-tuples of elements of \( X \).

**Examples:**

1. Let \( X = \{1, 2, 3, 4, 5\} \). Then
   (a) \( \{1, 2, 3\} \) is an example of a unary predicate (relation) on \( X \). We could equivalently express this predicate (relation) as
   \[ \{x \in X \mid x \leq 3\} \.
   
   (b) \( \{(1,1), (2,2), (3,3), (4,4), (5,5)\} \) is an example of a binary predicate (relation) on \( X \). We could equivalently express this predicate (relation) as
   \[ \{(x,y) \in X \times X \mid x = y\} \]
   in other words the predicate (relation) of **equality**.
(c) \{(1, 2, 3), (2, 2, 5), (3, 1, 2)\} is an example of a 3-ary predicate (relation) on \(X\).

2. Let \(X = \mathbb{Z}\). Then
\[
\{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid x < y\}
\]
is an example of a binary predicate (relation) on \(\mathbb{Z}\).

3. Let \(X = \mathbb{R}\). Then the parabola
\[
\{(x, y) \in \mathbb{R} \times \mathbb{R} \mid y = x^2\}
\]
is an example of a binary predicate (relation) on \(\mathbb{R}\). This is an example of a predicate (relation) which is also a function (think of the vertical line test).

4. Let \(X = \mathbb{R}\). Then the unit circle
\[
\{(x, y) \in \mathbb{R} \times \mathbb{R} \mid x^2 + y^2 = 1\}
\]
is an example of a binary predicate (relation) on \(\mathbb{R}\). This is an example of a predicate (relation) which fails to be a function (think of the vertical line test).

5. \(D = \mathbb{R} \times \mathbb{Z}\) is a good domain for the floor function. Note that the floor function is not an example of a binary predicate (relation), because the sets from which the two co-ordinates are taken to make the pairs are not the same.

Remarks:

1. With binary functions (e.g. \(+, \cdot\)) and predicates (e.g. \(=, >\)), we are used to writing the function or predicate (relation) symbol between its arguments. For example, we would write \(3 + 6\) and \(3 < 6\).

2. For consistency with the standard notation for any possible arity, we could always write the function/predicate first, and its arguments after, for example

<table>
<thead>
<tr>
<th>Symbol Type</th>
<th>Arithmetic</th>
<th>DrRacket</th>
<th>Predicate logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>(x + y)</td>
<td>(+\ x\ y)</td>
<td>(\text{sum}(x, y))</td>
</tr>
<tr>
<td>function</td>
<td>(x \cdot y)</td>
<td>(*\ x\ y)</td>
<td>(\text{product}(x, y))</td>
</tr>
<tr>
<td>predicate</td>
<td>(x = y)</td>
<td>(=\ x\ y)</td>
<td>(\text{Equals}(x, y))</td>
</tr>
<tr>
<td>predicate</td>
<td>(x &gt; y)</td>
<td>(&gt;\ x\ y)</td>
<td>(\text{Greater}(x, y))</td>
</tr>
</tbody>
</table>

3. Every function (of arity \(k\)) is a predicate (of arity \(k + 1\)); not every predicate is a function. Example 3 above is a binary predicate which comes from a unary function; Example 4 above is a binary predicate that does not come from any unary function.

12.2 Ingredients of Predicate Logic

The following ingredients make up the language in which we write our formulae of Predicate logic.

1. Constant symbols. Usually \(c, d, c_1, c_2, \ldots, d_1, d_2\) ...
2. Variable symbols. Usually \(x, y, z, \ldots, x_1, x_2, \ldots, y_1, y_2\) ...
   We use \(x, y, z\) for bound variables and \(u, v, w\) for free variables (this distinction to be explained soon).
3. Function symbols. Usually \( f, g, h, \ldots, f_1, f_2, \ldots, g_1, g_2, \ldots \)
4. Predicate symbols. \( P, Q, \ldots, P_1, P_2, \ldots, Q_1, Q_2, \ldots \)
5. Propositional Connectives: \( \neg, \land, \lor, \rightarrow, \text{and} \leftrightarrow \)
6. Quantifiers: \( \forall \) and \( \exists \)
7. Punctuation: ‘(’, ‘)’, and ‘,’

See pp 74-76 in the text.

### 12.3 Syntax of Predicate Logic

**Motivation:** What are the well-formed (i.e. syntactically correct) formulæ of Predicate logic?

#### 12.3.1 Terms

**Remarks:**

1. A term is a placeholder for a domain element.
2. Semantically, a term will evaluate to a domain element.
3. Examples of atomic terms: \( u, v, 0, 1 \).
4. Examples of non-atomic terms: \( (u + v), f(u), ((u + v) + v), ((u + v) + (u + v)) \).

**Inductive Definition of Terms:** Let

- \( X \) be the set of all strings that can be written using the individual symbols, free variable symbols, the function symbols and punctuation marks,
- \( A \) be the set of individual symbols plus variable symbols
- \( F \) contain one function per function symbol in the language:
  - For a function symbol \( f \), of arity \( k \), \( F \) contains a function (call it \( f \)) of arity \( k \) on the set of terms already created,
  - I.e. for \( f^{(k)} \), and for any terms \( t_1, \ldots, t_k \), \( f(t_1, \ldots, t_k) \) is a term.

Then the set of **terms** is \( I(X, A, F) \). See Definition 3.2.1 in the text.

#### 12.3.2 Atomic Formulae

**Remarks:**

1. Atomic formulæ play the role formerly played by Propositional variables (i.e. the simplest objects that become 0 or 1 in some semantic context).
2. Examples of atomic formulæ:
   - (a) \( S(u) \),
   - (b) \( E(u, v) \),
   - (c) \( (u > 0) \),
   - (d) \( F(f(u), 0) \).

**Definition of Atomic Formulae:** The set of **atomic formulæ** is defined as the union of
1. \( F(t_1, \ldots, t_k) \), for every predicate symbol \( P \) of arity \( k \), and terms \( t_1, \ldots, t_k \), and
2. \( \approx(t_1, t_2) \), for terms \( t_1 \) and \( t_2 \) (shorthand: \( t_1 \approx t_2 \)).

See Definition 3.2.2 in the text.

### 12.3.3 General Formulae

**Inductive Definition of Formulae:** Let

- \( X \) be the set of all strings that can be written using the individual symbols, variable symbols, the function symbols, the predicate symbols, the Propositional connectives and punctuation marks.
- \( A \) be the set of atomic formulæ.
- Let \( F \) contain one function per Propositional connective, plus one function per quantifier:
  - the Propositional connectives behave exactly as in the Propositional case, and
  - if \( A(u) \) is a formula, and if \( x \) does not occur in \( A(u) \), then \( \forall x A(x) \) and \( \exists x A(x) \) are formulæ (in both cases \( A(x) \) is called the **scope** of the quantifier - See Definition 3.2.8 in the text).

Then the set of formulæ is \( I(X, A, F) \). See Definition 3.2.3 in the text.

**Remarks:**

1. Precedence rule for quantifiers: bind the first syntactically correct formula to the immediate right of the quantifier.
2. As in the text, we do not add parentheses when introducing quantifiers, unless required by the above precedence rule.
3. The variable \( x \) in a formula of the form \( \forall x A(x) \) or \( \exists x A(x) \) is called **bound**.
4. A variable \( u \) is **free** if it is not bound.
5. A formula is a placeholder for a statement.
6. Semantically, a formula will evaluate to 0 or 1.
7. A formula is either
   (a) **simple** if it is of the form \( F(t_1, \ldots, t_n) \) for some \( n \)-ary predicate symbol \( P \) and some terms \( t_1, \ldots, t_n \) (e.g. \( I(u) \) asserting “\( u \) is an integer”), or
   (b) constructed from simpler formulæ using connectives and/or quantifiers (e.g. \( \forall x (I(x) \rightarrow (x + 0 = x)) \)).
8. Examples of non-simple formulæ:
   (a) \( \forall x F(f(x), 0) \).
   (b) \( S(u) \land E(u, v) \).

### 12.3.4 Parse Trees

**Remarks:**
1. Parse trees in the Predicate Logic case are similar to those in the Propositional logic case.
2. See p79 in the text.

Examples:

1. \[ \forall x F(f(x), 0). \]
   \[ \forall x F(f(x), 0) \]
   \[ \mid \]
   \[ F(f(u), 0) \]

2. \[ \forall x ((F(x) \rightarrow Q(x)) \land S(x, v)). \]
   \[ \forall x ((F(x) \rightarrow Q(x)) \land S(x, v)) \]
   \[ \mid \]
   \[ ((F(u) \rightarrow Q(u)) \land S(u, v)) \]
   \[ (F(u) \rightarrow Q(u)) \]
   \[ \mid \]
   \[ S(u, v) \]
   \[ F(u) \]
   \[ \mid \]
   \[ Q(u) \]

See the example on p79 of the text.

12.3.5 Variables

Remarks:

1. Since all variables are (atomic) terms, therefore a variable is a placeholder for a domain element.
2. Recall
   (a) \( x \) is bound in \( (\forall x (x + 0 = x)) \), and
   (b) \( u \) is free in \( (u + 0 = u) \), and
3. A formula with a \textbf{free variable} asserts something about the domain element for which the free variable stands in. For example, \( u \) is free in each of the following:
   (a) \( \text{Even}(u) \) could assert “\( u \) is even”.
   (b) \( \text{Greater}(u, 5) \) could assert “\( u > 5 \)”.
   (c) \( (\exists y \ \text{Greater}(u, y)) \) could assert “There exists \( y \) such that \( u > y \)”.
4. Often, but not always, some choices of domain element for a \textbf{free variable} \( u \) satisfy the formula, while other choices do not.
5. Many of our examples of formulæ in this course will be \textbf{closed formulæ} or \textbf{sentences} (i.e. formulæ with no free variables).

Examples:
1. Let $A$ be $(u > 0)$.
   (a) $A$ is an atomic formula. It is constructed using the predicate symbol $>(2)$, applied to the atomic terms $u$ and $0$.
   (b) The variable $u$ in $A$ is free.
   (c) A typical valuation, $v$, might choose
      i. domain: $D = \mathbb{Z}$
      ii. individual symbols: 0 has its usual meaning in $\mathbb{Z}$
      iii. functions: $\emptyset$ (we can omit this line safely)
      iv. predicates: $>(2)$ has its usual meaning in $\mathbb{Z}$
   (d) Note that this choice of $v$ is not enough to interpret $A$, because $A$ has a free variable, $u$. An assignment is needed to complete the interpretation of $A$.

2. Let $B$ be $(\forall x (x > 0))$.
   (a) The $x$ in $B$ is bound.
   (b) Since $B$ is a sentence, therefore the interpretation $v$ suffices to interpret $B$.
   (c) Indeed $B$ is 0 under $v$, since $-3 \in \mathbb{Z}$, and $(-3 > 0)$ is 0.

3. Let $C$ be $(\exists x (x > 0))$.
   (a) The $x$ in $C$ is bound.
   (b) Since $C$ is a sentence, therefore the interpretation $v$ suffices to interpret $C$.
   (c) Indeed $C$ is 1 under $v$, since $7 \in \mathbb{Z}$, and $(7 > 0)$ is 1.

4. If there are no free variables, then $v$ suffices to interpret a formula.

5. In the Predicate formula $(\forall x (\exists y F(x, y, z)))$, the $x$ and $y$ variables are bound and the $z$ variable is free. This formula is not a sentence, since it contains a free variable.

6. In the Predicate formula $(\forall x (\exists y F(x, y, a)))$ (where $a$ is an individual symbol), the $x$ and $y$ variables are bound. This formula is a sentence, since it contains no free variables.

13 Lecture 13

Outline

1. Translations from English into Predicate Logic
2. Translations from Predicate Logic into English
3. Semantics of Predicate Logic
   (a) Domains
   (b) Interpretations
   (c) Assignments
   (d) Valuations
   (e) Terms
   (f) Formulae
   (g) Satisfiability and Validity
   (h) Examples
13.1 Translations from English into Predicate Logic

Problems: Translate the following English sentences into Predicate Logic. Use the following notation.

- $S(u)$: $u$ is a student.
- $F(u)$: $u$ is an professor.
- $C(u)$: $u$ is a course.
- $M(u)$: $u$ belongs to Math.
- $U(u)$: $u$ belongs to Computer Science.
- $E(u, v)$: $u$ is enrolled in $v$.
- $T(u, v)$: $u$ teaches $v$.

1. $w$ is a course.
   Solution: $C(w)$

2. There exists a student $y$ who is enrolled in a course $u$.
   Solution: $(\exists y (S(y) \land (C(u) \land E(y, u))))$  

3. Every Computer Science course is a Math course.
   Solution: $(\forall x ((C(x) \land U(x)) \rightarrow (C(x) \land M(x))))$

4. Not every Math course belongs to Computer Science.
   Solution: $\neg(\forall x ((C(x) \land M(x)) \rightarrow U(x)))$
   Or (logically equivalent)
   $$(\exists x (C(x) \land (M(x) \land \neg U(x))))$$

5. There is a student who is enrolled in a course.
   Solution: $(\exists x (\exists y (S(x) \land (C(y) \land E(x, y)))))$

6. There is a student who is enrolled in some Computer Science course.
   Solution: $$(\exists x (\exists y (S(x) \land (C(y) \land (U(y) \land E(x, y))))))$$
   Or (logically equivalent)
   $$(\exists y (\exists x (S(x) \land (C(y) \land (U(y) \land E(x, y))))))$$
7. Some professor does not teach any course.
   
   **Solution:**
   \[
   \left( \exists x \left( F(x) \land \left( \forall y \left( C(y) \rightarrow \neg T(x, y) \right) \right) \right) \right)
   \]
   Or (logically equivalent)
   \[
   \left( \exists x \left( \forall y \left( F(x) \land \left( C(y) \rightarrow \neg T(x, y) \right) \right) \right) \right)
   \]
   
   **Remarks:**
   (a) Make certain you understand why we need \( \rightarrow \) and not \( \land \) here!

8. A professor cannot be a student.
   
   **Solution:**
   \[
   \left( \forall x \left( F(x) \rightarrow \neg S(x) \right) \right)
   \]
   Or (logically equivalent)
   \[
   \left( \neg \exists x \left( F(x) \land S(x) \right) \right)
   \]

### 13.2 Translations from Predicate Logic into English

**Problems:** Translate the following Predicate Logic sentences into English. Use the same notation from the last batch of examples.

1.
   \[
   \left( \exists x \left( S(x) \land \left( \forall y \left( C(y) \rightarrow \neg E(x, y) \right) \right) \right) \right)
   \]
   
   **Solution:** Some student is not enrolled in any course.

2.
   \[
   \left( \exists x \left( \left( C(x) \land U(x) \right) \land \left( \forall y \left( \left( C(y) \land U(y) \right) \rightarrow \left( y = x \right) \right) \right) \right) \right)
   \]
   
   **Solution:** There is exactly one Computer Science course.

3.
   \[
   \left( F(u) \land \left( \forall y \left( \left( C(y) \land M(y) \right) \rightarrow T(u, y) \right) \right) \right)
   \]
   
   **Solution:** Professor \( u \) teaches every Math course.

4.
   \[
   \left( \left( F(u) \land C(v) \right) \land T(u, v) \right)
   \]
   
   **Solution:** Professor \( u \) teaches course \( v \).

### 13.3 Semantics of Predicate Logic

**Motivation:** How do we assign 0 or 1 to each well-formed (i.e. syntactically correct) formula of Predicate logic?
13.3.1 Domains

Definition 13.3.1. A domain $D$ is a non-empty set.

13.3.2 Interpretations

Definition 13.3.2. An interpretation selects a domain $D$, then maps
- each individual symbol to a domain element,
- each function symbol of arity $k$ to a total function of arity $k$ on $D$,
- each relation symbol of arity $k$ to a relation of arity $k$ on $D$.

The text does not define any notation for an interpretation. The text only defines notation for a valuation, later.

13.3.3 Assignments

Definition 13.3.3. An assignment uses a selected a domain $D$ from an interpretation, then maps
- each (free) variable symbol to a domain element.

The text does not define any notation for an assignment. The text only defines notation for a valuation, later.

13.3.4 Valuations

Definition 13.3.4. A valuation $v$ is an interpretation and an assignment put together.

Notation:

<table>
<thead>
<tr>
<th>Given</th>
<th>Notation</th>
<th>Means</th>
</tr>
</thead>
<tbody>
<tr>
<td>individual symbol $a$</td>
<td>$a^v$</td>
<td>domain element to which $v$ maps $a$</td>
</tr>
<tr>
<td>free variable symbol $u$</td>
<td>$u^v$</td>
<td>domain element to which $v$ maps $u$</td>
</tr>
<tr>
<td>$k$-ary function symbol $f$</td>
<td>$f^v$</td>
<td>total function $D^k \to D$ to which $v$ maps $f$</td>
</tr>
<tr>
<td>$k$-ary relation symbol $F$</td>
<td>$F^v$</td>
<td>relation on $D^k$ to which $v$ maps $F$</td>
</tr>
</tbody>
</table>

13.3.5 Terms

Definition: Fix a valuation $v$. For each term $t$, the value of $t$ under $v$, denoted $t^v$, is as follows.

- If $t$ is an identity $a$, the value $t^v$ is $a^v$.
- If $t$ is a free variable $u$, the value $t^v$ is $u^v$.
- If $t$ is $f(t_1, \ldots, t_n)$, the value $t^v$ is $f^v(t_1^v, \ldots, t_n^v)$.

(See p88 in the text.)

Remarks:
1. Recall that a term is a placeholder for a domain element.

2. Hence when evaluated under some valuation, a term evaluates to a domain element:

**Proposition 13.3.5.** Let \( t \) be any well-formed term of Predicate logic. Let \( v \) be any valuation, with domain \( D \). Then \( t^v \in D \).

**Proof.** The proof is by structural induction on \( t \) and is left as an exercise. ■

**Lemma 13.3.6.** Let \( t \) be any well-formed term of Predicate Logic. Let \( v_1, v_2 \) be any valuations, over the same domain \( D \), and such that

- \( a^{v_1} = a^{v_2} \), for every individual symbol \( a \) in \( t \),
- \( u^{v_1} = u^{v_2} \), for every free variable symbol \( u \) in \( t \),
- \( f^{v_1} = f^{v_2} \), for every function symbol \( f \) in \( t \).

Then

\[ t^{v_1} = t^{v_2}. \]

**Proof.** See A06, Q3. ■

### 13.3.6 Formulae

#### A Notation Needed for Quantified Formulae

Let

- \( D \) be a domain,
- \( \alpha \in D \),
- \( u \) be a free variable, and
- \( v \) be a valuation.

Then define a new valuation

\[ v(u/\alpha) \]

which is the same as \( v \), except that \( u^{v(u/\alpha)} = \alpha \). In other words, \( v(u/\alpha) \) is \( v \), with the evaluation of \( u \) "overridden" to \( \alpha \). (See p88 in the text)

We write \( A^v = 1 \) to indicate that a valuation \( v \) satisfies a formula \( A \).

We write \( A^v = 0 \) to indicate that a valuation \( v \) does not satisfy a formula \( A \).
Form of $A$  
Condition for $A^v = 1$

$F(t_1, \ldots, t_k)$  
$\langle t_1^v, \ldots, t_k^v \rangle \in F^v$

$\neg B$  
$B^v = 0$

$(B \land C)$  
both $B^v = 1$ and $C^v = 1$

$(B \lor C)$  
either $B^v = 0$ or $C^v = 1$ (or both)

$(B \rightarrow C)$  
either $B^v = 0$ or $C^v = 1$ (or both)

$(B \leftrightarrow C)$  
$B^v = C^v$

$\forall x B$  
for every $\alpha \in \text{dom}(v)$, $A(u)^v(u/\alpha) = 1$

$\exists x B$  
there is some $\alpha \in \text{dom}(v)$, such that $A(u)^v(u/\alpha) = 1$

(See p89 in the text.)

Lemma 13.3.7. Let $A$ be any well-formed formula of Predicate Logic. Let $v_1, v_2$ be any valuations, over the same domain $D$, and such that

- $a^{v_1} = a^{v_2}$, for every individual symbol $a$ in $A$,
- $u^{v_1} = u^{v_2}$, for every free variable symbol $u$ in $A$,
- $f^{v_1} = f^{v_2}$, for every function symbol $f$ in $A$, and
- $F^{v_1} = F^{v_2}$, for every relation symbol $F$ in $A$.

Then

$A^{v_1} = A^{v_2}$.

Proof. Exercise. See 13.3.6 for one ingredient. ■

14 Lecture 14

Outline

1. Semantics of Predicate Logic
   (a) Satisfiability and Validity
   (b) Examples
   (c) Tautological Consequence

14.1 Semantics of Predicate Logic

14.1.1 Satisfiability and Validity

Validity and satisfiability of formulas have definitions analogous to the ones for propositional logic.

Definition: A formula $A$ is

- **valid** if every valuation satisfies $A$; that is, if $A^v = 1$ for every $v$, and
- **satisfiable** if some valuation satisfies $A$; that is, if $A^v = 1$ for some $v$, and
• unsatisfiable if no valuation satisfies $A$; that is, if $A^v = 0$ for every $v$.

(The term “tautology” is not used in predicate logic.)

14.1.2 Examples

1. Let $A$ be the Predicate Formula $\exists x (F(x) \rightarrow G(x))$. Let $D = \{a, b\}$.
   (a) Problem: Create $v_1$ over $D$ such that $A^{v_1} = 1$.
      Solution: Define $v$ to select
      
      $F^v = \{a\}$, and
      $G^v = \{a, b\}$.

      Then we have $(F(u) \rightarrow G(u))^{v_1(u/a)} = 1$, so that, by $\exists$-satisfaction rules, $\exists x (F(x) \rightarrow G(x))^v = 1$.
   (b) Problem: Create $v_2$ over $D$ such that $A^{v_2} = 0$.
      Solution: Define $v_2$ to select
      
      $F^{v_2} = \{a, b\}$, and
      $G^{v_2} = \emptyset$.

      Then we have
      
      $(F(u) \rightarrow G(u))^{v_2(u/a)} = 0$, and
      $(F(u) \rightarrow G(u))^{v_2(u/b)} = 0$,

      so that, by $\exists$-satisfaction rules, $\exists x (F(x) \rightarrow G(x))^{v_2} = 0$.

2. Let $A$ be the Predicate formula $\forall x (F(x) \rightarrow G(x))$. Let $D = \{a, b\}$.
   (a) Problem: Create $v_1$ over $D$ such that $A^{v_1} = 1$.
      Solution: Define $v_1$ to select
      
      $F^{v_1} = \{a\}$, and
      $G^{v_1} = \{a, b\}$.

      Then we have
      
      $(F(u) \rightarrow G(u))^{v_1(u/a)} = 1$, and
      $(F(u) \rightarrow G(u))^{v_1(u/b)} = 1$,

      so that, by $\forall$-satisfaction rules, $\forall x (F(x) \rightarrow G(x))^{v_1} = 1$. 52
(b) **Problem:** Create \( v_2 \) over \( D \) such that \( A^{v_2} = 0 \).

**Solution:** Define \( v_2 \) to select
\[
F^{v_2} = \{a, b\}, \quad \text{and} \quad G^{v_2} = \emptyset.
\]
Then we have
\[
(F(u) \rightarrow G(u))^{v_2(u/a)} = 0,
\]
so that, by \( \forall \)-satisfaction rules, \( \forall x (F(x) \rightarrow G(x)))^{v_2} = 1 \).

### 14.1.3 Tautological Consequence

**Definition 14.1.1.** let \( \Sigma \) be a set of well-formed formulæ of Predicate logic. Let \( v \) be a valuation. We say that \( v \) **satisfies** \( \Sigma \) (Notation: \( \Sigma^v = 1 \)), if and only if \( A^v = 1 \), for every formula \( A \in \Sigma \). Otherwise we say that \( v \) **does not satisfy** \( \Sigma \) (Notation: \( \Sigma^v = 0 \)).

**Definition 14.1.2.** Suppose \( \Sigma \) is a set of well-formed Predicate formulæ and \( A \) is a well-formed Predicate formula. We say that \( \Sigma \) (logically) **implies** \( A \), (Notation: \( \Sigma \vDash A \)), if and only if, for every valuation \( v \), we have
\[
\Sigma^v = 1 \text{ implies } A^v.
\]

**Remarks:**

1. As in Propositional logic, \( \emptyset \vdash A \) implies that \( A \) is valid.

**Examples:**

1. **Example:** Show that
\[
\emptyset \vdash \left( (\forall x (A \rightarrow B)) \rightarrow (\forall x A) \rightarrow (\forall x B) \right),
\]
i.e. prove that the formula \( (\forall x (A \rightarrow B)) \rightarrow (\forall x A) \rightarrow (\forall x B) \) is valid, for any well-formed Predicate formulæ \( A \) and \( B \).

Proof by contradiction. Suppose there is a valuation \( v \) such that
\[
((\forall x (A \rightarrow B)) \rightarrow (\forall x A) \rightarrow (\forall x B))^{v} = 0.
\]
Then by the \( \rightarrow \)-satisfaction rule, we must have \((\forall x (A \rightarrow B))^v = 1\) and \((\forall x A) \rightarrow (\forall x B))^v = 0\).

The second fact (again by the \( \rightarrow \)-satisfaction rule) gives \((\forall x A)^v = 1\) and \((\forall x B)^v = 0\).

Using the definition of \( \emptyset \vdash \) for formulas with \( \forall \), we have

for every \( a \in \text{dom}(v)\), \( A(u) \rightarrow B(u))^{v(u/a)} = 1 \) and \( A(u)^{v(u/a)} = 1 \).

Thus also \( B(u)^{v(u/a)} = 1 \) for every \( a \in \text{dom}(v)\).

Thus \((\forall x B)^v = 1\), a contradiction.
2. **Example.** Show that \( \{ (\forall x (\neg C)) \} \models (\neg (\exists x C)) \).

Suppose that \((\forall x (\neg C))^v = 1\). By \(\forall\)-satisfaction rules, this means for every \(d \in \text{dom}(v)\), \((\neg C(u))^v(\omega/d) = 1\).

By \(\neg\)-satisfaction rules, this is equivalent to for every \(d \in \text{dom}(v)\), \((C(u))^v(\omega/d) = 0\) in other words there does not exist \(d \in \text{dom}(v)\) such that \((C(u))^v(\omega/d) = 1\).

This last is the definition of \((\neg (\exists x C))^v = 1\), as required.

**Remarks:**
(a) The tautological consequence in the other direction could be proved similarly.
(b) Hence these two formulæ are **logically equivalent**.

3. **Example:** Show that, in general, \( \{ ((\forall x A) \rightarrow (\forall x B)) \} \not\models (\forall x (A \rightarrow B)) \).

(That is, exhibit a choice of \(A\) and \(B\) such that entailment does not hold, and prove that your choice is correct.)

**Key idea:** \((A \rightarrow C)\) yields true whenever \(A\) is false.

Let \(A\) be \(F(x)\). Let \(v\) have domain \(\{a, b\}\) and \(P^v = \{a\}\). Then \(((\forall x A) \rightarrow (\forall x B))^v = 1\) for any \(B\). (Why?)

- The reason why \(((\forall x A) \rightarrow (\forall x B))^v = 1\) for any \(B\):
  - Then we have \((A(u))^v(\omega/b) = 0\).
  - This shows that \((\forall x A)^v = 0\).
  - Then we have \(((\forall x A) \rightarrow (\forall x B))^v = 1\) for any \(B\), by \(\rightarrow\)-properties.

To obtain \((\forall x (A \rightarrow B))^v = 1\), we can use \((\neg F(x))\) for \(B\). (Why?)

- The reason why, to obtain \((\forall x (A \rightarrow B))^v = 0\), we can use \(B = (\neg F(u))\):
  - Let \(B = (\neg F(u))\).
  - Then \((A(u))^v(\omega/a) = 1\) and \((B(u))^v(\omega/a) = 0\).
  - This shows that \((\forall x (A \rightarrow B))^v = 0\).

Thus \(\{ ((\forall x A) \rightarrow (\forall x B)) \} \not\models (\forall x (A \rightarrow B))\), as required. (Why?)

- The reason why \(\{ ((\forall x A) \rightarrow (\forall x B)) \} \not\models (\forall x (A \rightarrow B))\):
  - Just apply Definition 14.1.2, using the previous two facts.

4. Prove the following Proposition.

**Proposition 14.1.3.** Let \(A\) be any well-formed Predicate formula **without** a free variable \(u\). Let \(v\) be any valuation. Then \(A^v = (\forall x A)^v\).

**Proof.** Let \(D\) be the domain of \(v\). Since \(u\) is not free in \(A\), therefore \(w^v = w^{v(\omega/a)}\), for every \(a \in D\) and for every \(w\) that occurs free in \(A\).

Then by the Lemma 13.3.7, we have that \(A^v = A^{v(\omega/a)}\), for any \(a \in D\), which establishes the desired result.

### 15 Lecture 15

Outline
1. Definability
   (a) Definability of a Relation
   (b) Definability of a Set of Structures
2. More Examples of Tautological Consequence

15.1 Definability

Remarks:
1. The two notions of definability described here give a flavour for the kinds of mathematical properties that can be described in Predicate logic.
2. This is the beginning of the subject of Model Theory. We will not have time to push further into this subject in CS 245.

Definition 15.1.1. A structure is a choice of
- a domain
- interpretations of all individual symbols
- interpretations of all function symbols
- interpretations of all relation symbols

This is also called an interpretation in the text (although the text defines no notation for an interpretation).

Remarks:
1. Another name for a structure is a model.
2. Hence a structure is denoted by $\mathcal{M}$.
3. A structure is enough of a semantic context to interpret any formula without free variables.
4. If $A$ is a formula without free variables, then we write $A^\mathcal{M}$ for the interpretation of $A$ under $\mathcal{M}$.
5. Similarly we write $a^\mathcal{M}$, $f^\mathcal{M}$, $F^\mathcal{M}$ for the interpretations of an individual symbol, a function symbol and a relation symbol.

15.1.1 Definability of a Relation

Definition 15.1.2. Let $F$ be a $k$-ary relation symbol. Let $\mathcal{M}$ be a structure. Then $F^\mathcal{M}$ is definable in $\mathcal{M}$ if there exists a formula $A(u_1, \ldots, u_k)$ such that

$$F^\mathcal{M} = \{(a_1, \ldots, a_k) \mid A(u_1, \ldots, u_k)^\mathcal{M}(u_1/a_1) \cdots (u_k/a_k) = 1\}.$$

Examples:
1. Let $\mathcal{L}$ be $\{+\}$, for a binary function symbol $\cdot$. Let $\mathcal{D}$ be $\mathbb{N}$. Let $\cdot$ have its usual meaning. Then
   (a) $\{0\}$ is definable, via $(u + u \approx u)$, and
(b) \( \{(u, w) \in \mathbb{N} \times \mathbb{N} \mid u \leq w\} \) is definable, via \( \exists z (u + z \approx w) \).

2. Let \( \mathcal{L} \) be \{+, \cdot\}, for binary function symbols +, \cdot. Let \( D \) be \( \mathbb{N} \). Let +, \cdot have their usual meanings. Then \( \{(u, w) \in \mathbb{N} \times \mathbb{N} \mid u \leq w\} \) is definable, via \( \exists z (u \cdot z = w) \).

3. Let \( \mathcal{L} \) be \{0, s\}, for a unary function symbol \( s \). Let \( D \) be \( \mathbb{N} \). Let 0 have its usual meaning. Let \( s \) be the \textit{successor function}, i.e. \( s(n) = n + 1 \). Then \( \{u \in \mathbb{N} \mid u \geq 3\} \) is definable, via \( \exists z (u \approx s(s(s(z))) \).

4. Let \( \mathcal{L} \) be \{\(<\)\}, for a binary relation symbol \(<\). Let \( D \) be \( \mathbb{R} \). Let \(<\) have its usual meaning. Then any singleton set is undefinable. One can prove that every definable set in this structure is either empty or infinite.

Remarks:

1. Which relations are definable is completely dependent on the choice of structure.
2. Definable relations are somehow “nicer” than undefinable relations.

**Theorem 15.1.3.** For a fixed arity, \( k \),

1. the complement of a definable relation is definable,
2. the union of two definable relations is definable, and
3. the intersection of two definable relations is definable.

**Proof.** 1. negation,
2. or, and
3. and.

### 15.1.2 Definability of a Set of Structures

**Definition 15.1.4.** A set, \( S \), of structures, is defined by a sentence, \( A \), if

\[
S = \{ \mathcal{M} \mid A^\mathcal{M} = 1 \}.
\]

**Examples:**

1. \( \forall x \forall y (F(x, y) \lor F(y, x)) \) defines the set of structures in which \( F \) is a \textit{total order} (every pair of elements is comparable).
2. (a) \( \forall x F(x, x) \) defines the set of structures in which \( F \) is \textit{reflexive}.
   (b) \( \forall x \forall y (F(x, y) \rightarrow F(y, x)) \) defines the set of structures in which \( F \) is \textit{reflexive}.
   (c) \( \forall x \forall y \forall z ((F(x, y) \land F(y, x)) \rightarrow F(x, z)) \) defines the set of structures in which \( F \) is \textit{transitive}.

   Therefore assembling all three of these sentences defines the set of structures in which \( F \) is an equivalence relation.
3. (a) \( \forall x \exists y (x \neq y) \) defines the set of structures which contain at least two elements.
   (b) \( \forall x \forall y \forall z ((x \approx y \lor x \approx z) \lor y \approx z) \) defines the set of structures which contain at most two elements.
So assembling these two sentences defines the set of structures which have exactly two elements.

### 15.2 More Examples of Tautological Consequence

1. **Problem:** Prove that \( \{∀xF(x)\} \vdash ∃xF(x) \).

   **Solution:**
   
   • Let \( v \) be any valuation such that \( ∀xF(x)^v = 1 \).
   • Let \( D \) be the domain of \( v \).
   • Let \( d ∈ D \) be arbitrary.
   • By the \( ∀ \)-satisfaction rule, we have
     
     \[ F(u)^v(u/d) = 1. \]
   • Then by the \( ∃ \)-satisfaction rule, we have
     
     \[ ∃xF(x)^v = 1. \]
   • Since \( v \) was arbitrary, this completes the proof.

2. **Problem:** Prove that \( \{∃xF(x)\} \not\vdash ∀xF(x) \).

   **Solution:**
   
   • Let \( v \) be the valuation defined by
     
     - domain \( D = \{0, 1\} \)
     - \( F^v = \{0\} \)
   • We have that
     
     \[ F(u)^v(u/0) = 1. \]
   • Then by the \( ∃ \)-satisfaction rule, we have
     
     \[ ∃xF(x)^v = 1. \]
   • However we also have that
     
     \[ F(u)^v(u/1) = 0. \]
   • Therefore by the \( ∀ \)-satisfaction rule, we have
     
     \[ ∀xF(x)^v = 0. \]

### 16 Lecture 16

**Outline**

1. Formal Deduction for Predicate Logic
   (a) Inference Rules for Quantified Formulae
   (b) Examples
16.1 Formal Deduction for Predicate Logic

1. In this lecture we extend Formal Deduction for Propositional Logic to create the Formal Deduction proof system for Predicate Logic.
2. As in the Propositional case, proofs in Predicate Natural Deduction are 100% syntactic, 0% semantic.
3. All the inference rules for Formal Deduction for Propositional Logic work as before.
4. We will see during the next lecture that this new proof system is both sound and complete.

16.1.1 Inference Rules for Quantified Formulae

1. \((\forall-)\): 
   If \(\Sigma \vdash \forall x A(x)\), then \(\Sigma \vdash A(t)\), for any term \(t\).
2. \((\forall+)\): 
   If \(\Sigma \vdash A(u), u\) not occurring in \(\Sigma\), then \(\Sigma \vdash \forall x A(x)\).
3. \((\exists-)\): 
   If \(\Sigma, A(u) \vdash B, u\) not occurring in \(\Sigma\) or \(B\), then \(\Sigma, \exists x A(x) \vdash B\).
4. \((\exists+)\): 
   If \(\Sigma \vdash A(t)\), then \(\Sigma \vdash \exists x A(x)\), where \(A(x)\) results by replacing some (not necessarily all) occurrences of \(t\) in \(A(t)\) by \(x\).
5. \((\equiv-)\): 
   If \(\Sigma \vdash A(t_1)\), \(\Sigma \vdash t_1 \equiv t_2\), then \(\Sigma \vdash A(t_2)\), where \(A(t_2)\) results by replacing some (not necessarily all) occurrences of \(t_1\) in \(A(t_1)\) by \(t_2\).
6. \((\equiv+)\): 
   \(\emptyset \vdash u \equiv u\).

Remarks:

1. To carefully define the \(\forall+\) rule, we need to prove our formula \(A\) holds for an arbitrary domain element \(u\) with no additional assumptions about \(u\). If such an argument involves a formula in which a variable \(u\) is free, then it may impose additional undesired conditions on \(u\). For example, if the argument involves \(Bird(u)\), then the conclusion \(A\) may hold only for domain elements which are birds, not necessarily to all domain elements.
2. Consider the example below:

\[
\Sigma = \{\forall x F(x), \exists z G(u, z)\}.
\]
If $G$ is the $<$ predicate, then the formula $\exists z G(u, z)$ asserts that “$u$ is not the maximal element of the domain”. If the desired conclusion $A(u)$ is connected with $u$ being maximal in the domain, then the presence of the earlier formula might tempt us to assume too much about $u$.

### 16.1.2 Examples

1. Prove that $\forall x F(x) \vdash \exists x F(x)$.

**Solution:**

(1) $\forall x F(x) \vdash \forall x F(x)$ (by ($\in$))

(2) $\forall x F(x) \vdash F(t)$ (by ($\forall -$), (1))

(3) $\forall x F(x) \vdash \exists x F(x)$ (by ($\exists +$), (2))

2. Prove that $\exists x F(x) \not\vdash \forall x F(x)$.

**Solution:** We will need the (contrapositive of) soundness of Predicate Formal Deduction to prove this fact, next time.

3. Prove

$$\emptyset \vdash \forall x (F(x) \rightarrow F(x))$$

**Solution:**

(1) $F(u) \vdash F(u)$ (by ($\in$))

(2) $\emptyset \vdash (F(u) \rightarrow F(u))$ (by ($\rightarrow +$), (1))

(3) $\emptyset \vdash \forall x(F(x) \rightarrow F(x))$ (by ($\forall +$), (2))

4. Prove

$$\{\forall x(F(x) \rightarrow G(x)), \forall xF(x)\} \vdash \forall xG(x)$$

**Solution:**

(1) $\forall x(F(x) \rightarrow G(x)), \forall xF(x) \vdash \forall x(F(x) \rightarrow G(x))$ (by ($\in$))

(2) $\forall x(F(x) \rightarrow G(x)), \forall xF(x) \vdash (F(u) \rightarrow G(u))$ (by ($\forall -$), (1))

(3) $\forall x(F(x) \rightarrow G(x)), \forall xF(x) \vdash \forall xF(x)$ (by ($\in$))

(4) $\forall x(F(x) \rightarrow G(x)), \forall xF(x) \vdash F(u)$ (by ($\forall -$), (3))

(5) $\forall x(F(x) \rightarrow G(x)), \forall xF(x) \vdash G(u)$ (by ($\rightarrow -$), (2), (4))

(6) $\forall x(F(x) \rightarrow G(x)), \forall xF(x) \vdash \forall xG(x)$ (by ($\forall +$), (5))

5. Modify the proof in part 4 to show

$$\{\forall x(F(x) \rightarrow G(x)), \forall xF(x)\} \vdash \exists xG(x)$$

**Solution:** Exercise.

6. Prove

$$\{ (\forall xF(x) \rightarrow \exists xH(x)), \forall x(F(x) \land G(x))\} \vdash \exists xH(x)$$

**Solution:**

7. Prove

$$\{ \exists x(F(x) \lor G(x))\} \vdash (\exists xF(x) \lor \exists xG(x))$$
Solution:
(1) $F(u) \vdash F(u)$ (by $(\in)$)
(2) $F(u) \vdash \exists x F(x)$ (by $(\exists +), (1))$
(3) $F(u) \vdash (\exists x F(x) \lor \exists x G(x))$ (by $(\lor +), (2))$
(4) $G(u) \vdash G(u)$ (by $(\in)$)
(5) $G(u) \vdash \exists x G(x)$ (by $(\exists +), (4))$
(6) $G(u) \vdash (\exists x F(x) \lor \exists x G(x))$ (by $(\lor +), (5))$
(7) $(F(u) \land G(u)) \vdash (\exists x F(x) \lor \exists x G(x))$ (by $(\lor -), (3), (6))$
(8) $\exists x(F(x) \land G(x)) \vdash (\exists x F(x) \lor \exists x G(x))$

8. Prove
\[
\{\exists x(P(x) \land Q(x))\} \vdash (\exists x P(x) \land \exists x Q(x))
\]

Solution: Exercise.

9. Prove
\[
\{\exists x F(x), \forall x(F(x) \rightarrow G(x))\} \vdash \exists x G(x)
\]

Solution:
(1) $\forall x(F(x) \rightarrow G(x)), F(u) \vdash F(u)$ (by $(\in)$)
(2) $\forall x(F(x) \rightarrow G(x)), F(u) \vdash \forall x(F(x) \rightarrow G(x))$ (by $(\in)$)
(3) $\forall x(F(x) \rightarrow G(x)), F(u) \vdash (F(u) \rightarrow G(u))$ (by $(\forall -), (2))$
(4) $\forall x(F(x) \rightarrow G(x)), F(u) \vdash G(u)$ (by $(\rightarrow -), (1), (3))$
(5) $\forall x(F(x) \rightarrow G(x)), F(u) \vdash \exists x G(x)$ (by $(\exists +), (4))$
(6) $\forall x(F(x) \rightarrow G(x)), \exists x F(x) \vdash \exists x G(x)$ (by $(\exists -), (5))$

10. Prove
\[
\{\exists y \forall x F(x, y)\} \vdash \forall x \exists y F(x, y)
\]

Solution:
(1) $\forall x F(x, w) \vdash \forall x F(x, w)$ (by $(\in)$)
(2) $\forall x F(x, w) \vdash F(u, w)$ (by $(\forall -), (1))$
(3) $\forall x F(x, w) \vdash \exists y F(u, y)$ (by $(\exists +), (2))$
(4) $\exists y \forall x F(x, y) \vdash \exists y F(u, y)$ (by $(\exists -), (3))$
(5) $\exists y \forall x F(x, y) \vdash \forall x \exists y F(x, y)$ (by $(\forall +), (4))$

11. Prove
\[
\{(\forall x F(x) \land \forall x G(x))\} \vdash \forall x(F(x) \lor G(x))
\]

Solution:

12. Prove
\[
\emptyset \vdash \forall x((F(x) \rightarrow G(x)) \land (G(x) \rightarrow F(x))).
\]

You may use derived rules.

Solution:

17 Lecture 17

Outline
1. Soundness and Completeness of Formal Deduction for Predicate Logic
   (a) Soundness
   (b) Rule $\forall^-$ is Sound
   (c) Rule $\exists^+$ is Sound
   (d) Rule $\forall^+$ is Sound
   (e) Rule $\exists^-$ is Sound
   (f) Rule $\approx^-$ is Sound
   (g) Rule $\approx^+$ is Sound
   (h) Completeness
   (i) Examples

17.1 Soundness and Completeness of Formal Deduction for Predicate Logic

17.1.1 Soundness

Theorem 17.1.1. Formal Deduction for Predicate Logic is sound, i.e. whenever $\Sigma \vdash A$, it follows that $\Sigma \vDash A$.

Proof (Outline). Because Propositional Formal Deduction is sound, therefore it suffices to prove the soundness of $\forall^-, \exists^+, \forall^+, \exists^-, \approx^-$ and $\approx^+$.

17.1.2 Rule $\forall^-$ is Sound

Theorem 17.1.2. The $\forall^-$ inference rule is sound.

Proof. Recall the $\forall^-$ inference rule:
If $\Sigma \vdash \forall x A(x)$,
then $\Sigma \vdash A(t)$, for any term $t$.

- Suppose that $\Sigma \vDash \forall x A(x)$.
- We are finished if we can prove that $\Sigma \vDash A(t)$, for any term $t$.
- Let $t$ be any Predicate term.
- Let $v$ be any valuation such that $\Sigma^v = 1$.
- Let $D$ be the domain of $v$.
- Then by our hypothesis $\forall x A(x)^v = 1$.
  - By Proposition 13.3.5, $t^v \in D$.
  - By $\forall$-satisfaction, $A(u)^v(u/t^v) = 1$.
  - In other words, $A(t)^v = 1$.

17.1.3 Rule $\exists^+$ is Sound

Theorem 17.1.3. The $\exists^+$ inference rule is sound.
Proof. Recall the \( \exists^+ \) inference rule:

If \( \Sigma \vdash A(t) \),
then \( \Sigma \vdash \exists x A(x) \), where \( A(x) \) results by replacing some (not necessarily all) occurrences of \( t \) in \( A(t) \) by \( x \).

- Suppose that \( \Sigma \models A(t) \), for some Predicate term \( t \).
- We are finished if we can prove that \( \Sigma \models \exists x A(x) \), where \( A(x) \) results by replacing some (not necessarily all) occurrences of \( t \) in \( A(t) \) by \( x \).
- Let \( v \) be any valuation such that \( \Sigma^v = 1 \).
- Let \( D \) be the domain of \( v \).
- Suppose that \( A(u) \) results by replacing some (not necessarily all) occurrences of \( t \) in \( A(t) \) by \( u \).
- Then by our hypothesis \( A(u)^{v(u/t^v)} = 1 \).
- By Proposition 13.3.5, \( t^v \in D \).
- By \( \exists \)-satisfaction, \( \exists x A(x)^v = 1 \).

\[ \blacksquare \]

17.1.4 Rule \( \forall^+ \) Is Sound

Theorem 17.1.4. The \( \forall^+ \) inference rule is sound.

Proof. Recall the \( \forall^+ \) inference rule:

If \( \Sigma \vdash A(u) \), \( u \) not occurring in \( \Sigma \),
then \( \Sigma \vdash \forall x A(x) \).

- Suppose that \( \Sigma \vdash A(u) \), \( u \) not occurring in \( \Sigma \).
- We are finished if we can prove \( \Sigma \vdash \forall x A(x) \).
- Let \( v \) be any valuation such that \( \Sigma^v = 1 \).
- Let \( D \) be the domain of \( v \).
- Let \( d \in D \) be arbitrary.
- I claim that Lemma 13.3.7 implies that \( \Sigma^{v(u/d)} = \Sigma^v \).
- Let \( B \in \Sigma \) be arbitrary.
- It suffices to prove that \( B^{v(u/d)} = B^v \).
- To apply Lemma 13.3.7 for formula \( B \), we need to verify that \( w^{v(u/d)} = w^v \), for every free variable \( w \) in \( B \).
- Because \( u \) does not occur in \( \Sigma \), therefore \( w \neq u \) (i.e. \( w \) and \( u \) are different free variable symbols).
- Hence by the definition of \( v(u/d) \), it is clear that \( w^{v(u/d)} = w^v \).
- Therefore Lemma 13.3.7 applies as stated.
- To summarize, \( \Sigma \vdash A(u) \) and \( \Sigma^{v(u/d)} = \Sigma^v = 1 \).
- Hence \( A(u)^{v(u/d)} = 1 \).
- Since \( d \in D \) was arbitrary, therefore by \( \forall \)-satisfaction, \( \forall x A(x)^v = 1 \).

\[ \blacksquare \]
17.1.5 Rule $\exists -$ Is Sound

Theorem 17.1.5. The $\exists -$ inference rule is sound.

Proof. Recall the definition of the $\exists -$ inference rule:
If $\Sigma, A(u) \vdash B, u$ not occurring in $\Sigma$ or $B$,
then $\Sigma, \exists x A(x) \vdash B$.

- Suppose that $\Sigma, A(u) \models B, u$ not occurring in $\Sigma$ or $B$.
- We are finished if we can prove that $\Sigma, \exists x A(x) \models B$.
- Let $v$ be an arbitrary valuation such that $\Sigma^v = 1$ and $\exists x A(x)^v = 1$.
- Let $D$ be the domain of $v$.
- There is some $d \in D$ such that $A(u)^v(u/d) = 1$.
- Since $u$ does not occur in $\Sigma$, therefore (similarly to the $\forall +$ case) $\Sigma^v(u/d) = \Sigma^v = 1$.
- Then since $\Sigma^v(u/d) = 1$ and $A(u)^v(u/d) = 1$, we have that $B^v(u/d) = 1$.
- Since $u$ does not occur in $B$, therefore (similarly to the $\forall +$ case) $B^v = B^v(u/d) = 1$.

17.1.6 Rule $\approx -$ Is Sound

Theorem 17.1.6. The $\approx -$ inference rule is sound.

Proof. Recall the definition of the $\approx -$ inference rule:
If $\Sigma \vdash A(t_1)$,
$\Sigma \vdash t_1 \approx t_2$,
then $\Sigma \vdash A(t_2)$, where $A(t_2)$ results by replacing some
(not necessarily all) occurrences of $t_1$ in $A(t_1)$ by $t_2$.

- Suppose that $\Sigma \vdash A(t_1)$, and $\Sigma \vdash t_1 \approx t_2$.
- We are finished if we can prove $\Sigma \models A(t_2)$, where $A(t_2)$ results by replacing some (not necessarily all) occurrences of $t_1$ in $A(t_1)$ by $t_2$.
- Let $v$ be any valuation such that $\Sigma^v = 1$.
- Then by our hypotheses, $A(t_1)^v = 1$ and $(t_1 \approx t_2)^v = 1$, in other words $t_1^v = t_2^v$.
- Then because $A(t_2)$ results by replacing some (not necessarily all) occurrences of $t_1$ in $A(t_1)$ by $t_2$, it follows that $A(t_2)^v = 1$.

17.1.7 Rule $\approx +$ Is Sound

Theorem 17.1.7. The $\approx +$ inference rule is sound.

Proof. Recall the definition of the $\approx +$ inference rule:
$\emptyset \vdash u \approx u$.

- We must prove that $\emptyset \models u \approx u$, i.e. that $(u \approx u)^v = 1$, for every valuation $v$. 

63
• The above can be re-written as \( u^v = u^w \), which clearly always holds.

\[ \]

17.1.8 Completeness

• The main ingredient in the proof of completeness is the same as in the Propositional case, namely that every consistent set of formulæ of Predicate logic is satisfiable.
• We had to wave our hands to prove Completeness in the simpler Propositional case. In the Predicate case, things become so complicated that we cannot prove completeness even with hand-waving.
• We will state without proof that whenever \( \Sigma \vdash A \), we also have that \( \Sigma \models A \).

17.1.9 Examples

1. **Problem:** Prove that \( \{ \exists x F(x) \} \nvdash \forall x F(x) \).
   **Solution:**
   • We showed earlier that \( \{ \exists x F(x) \} \nvdash \forall x F(x) \).
   • By the contrapositive of soundness, we have \( \{ \exists x F(x) \} \nvdash \forall x F(x) \).

18 Lecture 18

Outline

1. Mid-Term Exam Review

18.1 Mid-Term Exam Review

1. **Question 12** Let \( B \) be a Predicate formula. Define the Predicate formula \( A \) below.

\[ A : (\exists x B) \rightarrow (\forall x B) \]

(a) Give a Predicate formula \( B \) such that the formula \( A \) is valid. Prove your answer.
   **Solution:** Choose \( B \) to be any valid predicate formula. For example, let \( B = (F(x) \lor \neg F(x)) \) where \( F \) is a unary predicate symbol.
   Since \( B \) is valid, so \( (\exists x B) \) and \( (\forall B) \) must be valid as well. Thus, \( A \) must be valid.

(b) Give a Predicate formula \( B \) such that the formula \( A \) is satisfiable but not valid. Prove your answer.
   **Solution:** Let \( B = F(x) \) where \( F \) is a unary predicate symbol.
   The formula is satisfiable since it is true under the valuation \( v_1: D = \{1, 2\}, F^{v_1} = \{1, 2\} \).
   Form \( F(u) \).
Since $F(u)^{v_1(u/1)} = 1$, therefore $(\exists x F(x))^{v_1} = 1$.

Since $F(u)^{v_1(u/1)} = 1$ and $F(u)^{v_1(u/2)} = 1$, therefore $(\forall x F(x))^{v_1} = 1$.

The formula is not valid since it is not true under the valuation $v_2$: $D = \{1, 2\}$, $F^{v_2} = \{1\}$.

Form $F(u)$.

Since $F(u)^{v_1(u/1)} = 1$, therefore $(\exists x F(x))^{v_1} = 1$.

Since $F(u)^{v_1(u/2)} = 0$, therefore $(\forall x F(x))^{v_1} = 0$.

2. Question 9

(a) Let $p_1, \ldots, p_n$ be Propositional variables, for some $n \geq 1$. Let $\Sigma$ be a set of well-formed Propositional formulae, such that $p_i \in \Sigma$, for all $1 \leq i \leq n$.

Let $A$ be a Propositional formula such that every Propositional variable in $A$ is among $\{p_1, \ldots, p_n\}$. Prove that either $\Sigma \vdash A$ or $\Sigma \not\vdash (\neg A)$.

Solution:

Case 1: If $\Sigma \vdash A$, then there is nothing to prove. ✓

Case 2: Assume that $\Sigma \not\vdash A$. ✓ We will prove that $\Sigma \vdash (\neg A)$.

Since $\Sigma \not\vdash A$, by the contrapositive of completeness, we have $\Sigma \not\vdash A$. This means that there exists a truth valuation $t_1$ such that $\Sigma^{t_1} = 1$ and $A^{t_1} = 0$. ✓

Since $p_i \in \Sigma$, for all $1 \leq i \leq n$, therefore $p_i^{t_1} = 1$, for all $1 \leq i \leq n$. Note that every Propositional variable in $A$ is among $\{p_1, \ldots, p_n\}$. Also, note that for $t_1$, $p_i^{t_1} = 1$, for all $1 \leq i \leq n$ and $A^{t_1} = 0$. This shows that any truth valuation $t_2$, such that $p_i^{t_2} = 1$, for all $1 \leq i \leq n$, also satisfies $A^{t_2} = 0$. ✓

Now, we will prove that $\Sigma \vdash (\neg A)$. Let $t_3$ be any truth valuation such that $\Sigma^{t_3} = 1$. Now, by the above observation, we have that $A^{t_3} = 0$. By the $\neg$-satisfaction rule, we have that $(\neg A)^{t_3} = 1$. This proves the claim that $\Sigma \vdash (\neg A)$. ✓

Then, by the completeness of Formal Deduction for Propositional logic, we have that $\Sigma \vdash (\neg A)$. ✓

19 Lecture 19

Outline

1. Applications of Soundness / Completeness of Predicate Formal Deduction, Likely Needed for A08
2. More Mid-Term Exam Review

19.1 Applications of Soundness / Completeness of Predicate Formal Deduction, Likely Needed for A08

Remarks:
1. All the facts we proved as parts of the Soundness and Completeness Theorems for Propositional Formal Deduction still apply for Predicate Formal Deduction. In particular,

(a) Completeness is equivalent to “every consistent set is satisfiable”. The contrapositive of this fact can be used to disprove consistency, which in my experience can be more difficult to disprove directly from the definition(s).

(b) Soundness is equivalent to “every satisfiable set is consistent”. This fact can be used to prove consistency, which in my experience is more difficult to prove directly from the definition(s).

Examples:

1. For each given set $\Sigma$, determine with proof whether $\Sigma$ is consistent or inconsistent.

   (a) $\Sigma = \{F(u,w), G(f(w),g(u))\}$

   **Solution:**
   - I claim that $\Sigma$ is consistent.
   - By soundness, it suffices to prove that $\Sigma$ is satisfiable.
   - Let $v$ be the valuation defined by
     - domain $D = \{1, 2\}$
     - $f^v(1) = 2; f^v(2) = 1$
     - $g^v(1) = 2; g^v(2) = 2$
     - $F^v = \{(1, 2), (2, 1)\}$
     - $G^v = \{(1, 1), (2, 2)\}$
     - $u^v = 2$
     - $w^v = 1$
   - Then we have that $F(u,w)^v = 1$, and $G(f(w),g(u))^v = 1$.
   - Hence $\Sigma$ is satisfiable.
   - By soundness, $\Sigma$ is consistent.

   (b) $\Sigma = \{(F(u) \rightarrow G(u)), (\neg G(u)) \land F(u)\}$

   **Solution 1:**
   - I claim that $\Sigma$ is inconsistent.
   - By the contrapositive of completeness, it suffices to prove that $\Sigma$ is unsatisfiable.
   - Let $v$ be any valuation.
   - We have four subcases, depending on $F(u)^v, G(u)^v$.
     - $F(u)^v = 0, G(u)^v = 0$ Then $((\neg G(u)) \land F(u))^v = 0$, so $\Sigma^v = 0$.
     - $F(u)^v = 0, G(u)^v = 1$ Then again $((\neg G(u)) \land F(u))^v = 0$, so $\Sigma^v = 0$.
     - $F(u)^v = 1, G(u)^v = 0$ Then $(F(u) \rightarrow G(u))^v = 0$, so $\Sigma^v = 0$.
     - $F(u)^v = 1, G(u)^v = 1$ Then again $((\neg G(u)) \land F(u))^v = 0$, so $\Sigma^v = 0$.
   - In all cases, $\Sigma^v = 0$.
   - Since $v$ was arbitrary, therefore $\Sigma$ is unsatisfiable.
   - By the contrapositive of completeness, $\Sigma$ is consistent.

   **Solution 2:**
and lines (3) and (5) witness that \( \Sigma \) is inconsistent.

19.2 More Mid-Term Exam Review

1. Question 9
   
   (a) (part b) Let \( p_1, \ldots, p_n \) be Propositional variables, for some \( n \geq 1 \). Prove or disprove the following statement.
   
   For every Propositional formula \( A \), if none of \( p_1, \ldots, p_n \) occurs in \( A \), then \( \{p_1, \ldots, p_n\} \not \models A \).
   
   **Solution:** We provide a counterexample to disprove the statement.
   
   • Let \( A \) be \( (q_1 \lor (\neg q_1)) \).
   
   • Then since \( (q_1 \lor (\neg q_1)) \) is a tautology, therefore \( \emptyset \models (q_1 \lor (\neg q_1)) \).
   
   • By the completeness of Formal Deduction for Propositional logic, \( \emptyset \vdash (q_1 \lor (\neg q_1)) \).
   
   • By \((+, \{p_1, \ldots, p_n\} \vdash (q_1 \lor (\neg q_1)) \).
   
   • Now since none of \( p_1, \ldots, p_n \) occurs in \( A \), this completes the explanation of the counterexample.

2. Question 8
   
   Let \( \Sigma \) be a set of well-formed Propositional formulæ. Let \( A \) be a well-formed Propositional formula. Prove that \( \Sigma \models A \) if and only if \( \Sigma \cup \{\neg A\} \) is unsatisfiable.
   
   **Remark:** I suggested this question for inclusion on the mid-term exam. My original version also included the instruction, translated into our language for this term:
   
   You must use the definition of semantic entailment to write your proof. Do not use any other technique such as truth tables, valuation trees, logical equivalence, Formal Deduction, soundness, or completeness.

   **Solution:** For the forward direction, assume that \( \Sigma \models A \). Let \( t \) be any truth valuation.
   
   ✓ We have these cases for \( \Sigma^t \).
   
   • If \( \Sigma^t = 1 \), then because \( \Sigma \models A \), it follows that \( A^t = 1 \). Hence \( (\neg A)^t = 0 \). Therefore \( \Sigma \cup \{\neg A\}^t = 0 \). ✓
   
   • If \( \Sigma^t = 0 \), then \( \Sigma \cup \{\neg A\}^t = 0 \). ✓
   
   In either case, \( \Sigma \cup \{\neg A\}^t = 0 \). Since \( t \) was arbitrary, this shows that \( \Sigma \cup \{\neg A\} \) is unsatisfiable.
   
   For the backward direction, assume that \( \Sigma \cup \{\neg A\} \) is unsatisfiable. Let \( t \) be a truth valuation such that \( \Sigma^t = 1 \). ✓ We have these cases for \( A^t \).
   
   • If \( A^t = 0 \), then \( (\neg A)^t = 1 \). But then \( \Sigma \cup \{\neg A\}^t = 1 \). This contradicts the fact that \( \Sigma \cup \{\neg A\} \) is unsatisfiable, and so this case cannot occur. ✓
   
   • The only remaining possibility, namely that \( A^t = 1 \), must occur. ✓
   
   This proves that \( \Sigma \models A \).

3. Question 3
   
   Let \( A_i \) and \( B_i \) be Propositional formulæ where \( i = 1, \ldots, n \). Let \( t \) be a
truth valuation such that
\[(A_i \rightarrow B_i)^t = 1 \quad (i = 1, ..., n), \quad (1)\]
\[(A_1 \lor ... \lor A_n)^t = 1, \quad (2)\]
\[(B_i \land B_j)^t = 0, \text{ for } i \neq j, (i, j = 1, ..., n). \quad (3)\]

Prove that \((B_i \rightarrow A_i)^t = 1\), for all \((i = 1, ..., n)\).

Solution:
- By formula 2, \(A_i\) is true for at least one \(i = 1, ..., n\). Let \(k\) be the smallest choice such that \(A_k^t = 1\). ✓
- Therefore, \((B_k \rightarrow A_k)^t = 1\). ✓
- By formula 1, \((A_i \rightarrow B_i)\) is true for every \(i = 1, ..., n\), hence \(B_k^t = 1\). ✓
- By formula 3, since \(B_k^t = 1\), therefore \(B_j^t = 0\) for all \(j \neq k\). ✓
- Hence for all \(j \neq k\), \((B_j \rightarrow A_j)^t = 1\). ✓

20 Lecture 20 - Peano Arithmetic I

Outline

1. Properties of Equality
   (a) Reflexivity
   (b) Symmetry
   (c) Transitivity

2. Introduction to Peano Arithmetic

20.1 Properties of Equality

20.1.1 Reflexivity

We have
\[(1) \emptyset \vdash u \approx u \quad \text{(by } \approx+\text{)}\]
\[(2) \emptyset \vdash \forall x(x \approx x) \quad \text{(by } \forall+\text{, (1)})\]

20.1.2 Symmetry

We have
\[(1) \emptyset \vdash u \approx u \quad \text{(by } \approx+\text{)}\]
\[(2) u \approx v \vdash u \approx v \quad \text{(by } \in\text{)}\]
\[(3) u \approx v \vdash u \approx u \quad \text{(by } +, (1)\text{)}\]
\[(4) u \approx v \vdash v \approx u \quad \text{(by } \approx-, (2), (3)\text{)}\]
\[(5) \emptyset \vdash (u \approx v \rightarrow v \approx u) \quad \text{(by } \rightarrow +\text{, (4)}\text{)}\]
\[(6) \emptyset \vdash \forall y(u \approx y \rightarrow y \approx u) \quad \text{(by } \forall+\text{, (5)}\text{)}\]
\[(7) \emptyset \vdash \forall x \forall y(x \approx y \rightarrow y \approx x) \quad \text{(by } \forall+\text{, (6)}\text{)}\]
Explanation of the application of \((\approx-\approx)\) on line (4):

- \(\Sigma\) is \(\{u \approx v\}\).
- \(t_1\) is \(u\)
- \(t_2\) is \(v\)
- \(A(t_1)\) is \(u \approx u\)
- \(A(t_2)\) is \(v \approx u\)

Remarks:

1. I am being extra verbose here, as this is our first opportunity to use the \((\approx-\approx)\) rule. You do not need to include this type of explanation in your own proofs.
2. Strictly following our syntax rules would require us to write \(\neg(t_1 \approx t_2)\) instead of \((t_1 \not\not\approx t_2)\). But A06 permitted the \(\not\not\approx\) notation, so this is also fine.

20.1.3 Transitivity

We have

\[
\begin{align*}
(1) & \quad (u \approx v \land v \approx w) \vdash (u \approx v \land v \approx w) & \text{(by (\(\in\)))} \\
(2) & \quad (u \approx v \land v \approx w) \vdash u \approx v & \text{(by (\(\land-\))),(1)} \\
(3) & \quad \emptyset \vdash \forall x \forall y(x \approx y \to y \approx x) & \text{(by symmetry of \(\approx\))} \\
(4) & \quad \emptyset \vdash \forall y(u \approx y \to y \approx u) & \text{(by (\(\forall-\)), (3))} \\
(5) & \quad \emptyset \vdash (u \approx v \to v \approx u) & \text{(by (\(\forall-\)), (4))} \\
(6) & \quad (u \approx v \land v \approx w) \vdash (u \approx v \to v \approx u) & \text{(by (+), (5))} \\
(7) & \quad (u \approx v \land v \approx w) \vdash v \approx u & \text{(by (\(\to-\)), (2), (6))} \\
(8) & \quad (u \approx v \land v \approx w) \vdash v \approx w & \text{(by (\(\land-\)), (1))} \\
(9) & \quad (u \approx v \land v \approx w) \vdash u \approx w & \text{(by (\(\approx-\)), (7), (8))} \\
(10) & \quad \emptyset \vdash ((u \approx v \land v \approx w) \to u \approx w) & \text{(by (\(\to+\)), (9))} \\
(11) & \quad \emptyset \vdash \forall z((u \approx v \land v \approx z) \to u \approx z) & \text{(by (\(\forall+\)), (10))} \\
(12) & \quad \emptyset \vdash \forall y \forall z((u \approx y \land y \approx z) \to u \approx z) & \text{(by (\(\forall+\)), (11))} \\
(13) & \quad \emptyset \vdash \forall x \forall y \forall z((x \approx y \land y \approx z) \to x \approx z) & \text{(by (\(\forall+\)), (12))} \\
\end{align*}
\]

Explanation of the application of \((\approx-\approx)\) on line (9):

- \(\Sigma\) is \(\{(u \approx v \land v \approx w)\}\).
- \(t_1\) is \(v\)
- \(t_2\) is \(u\)
- \(A(t_1)\) is \(v \approx w\)
- \(A(t_2)\) is \(u \approx w\)

Remarks:

1. Since \(\approx\) is reflexive, symmetric and transitive, therefore \(\approx\) is an equivalence relation.
20.2 Introduction to Peano Arithmetic

1. Fix the domain as \( \mathbb{N} \), the natural numbers.
2. Interpret the individual symbol 0 as zero and the unary function symbol \( s \) as successor \( (s(n) = n + 1) \).
3. Thus each number in \( \mathbb{N} \) has a term: 0, \( s(0) \), \( s(s(0)) \), \( s(s(s(0))) \), ....

Zero and successor satisfy the following axioms.

**PA1:** \( \forall x \ (\neg(s(x) \approx 0)) \).
“Zero is not a successor.”

**PA2:** \( \forall x \forall y \ (s(x) \approx s(y) \rightarrow x \approx y) \).
“Nothing has two predecessors.”

(“PA” stands for Peano Axioms, named for Giuseppe Peano.) Further axioms characterize + (addition) and \( \times \) (multiplication).

**PA3:** \( \forall x \ (x + 0 \approx x) \).
Adding zero to any number yields the same number.

**PA4:** \( \forall x \forall y \ (x + s(y) \approx s(x + y)) \).
Adding a successor yields the successor of adding the number.

**PA5:** \( \forall x \ (x \times 0 \approx 0) \).
Multiplying by zero yields zero.

**PA6:** \( \forall x \forall y \ (x \times s(y) \approx (x \times y) + x) \).
Multiplication by a successor.

The six axioms above define + and \( \times \) for any particular numbers. They do not, however, allow us to reason adequately about all numbers. For that, we use an additional axiom: **induction**.

**PA7:** For each formula \( A(u) \) and variable \( x \),

\[
\left( (A(0) \land \forall x \ (A(x) \rightarrow A(s(x)))) \rightarrow \forall x \ A(x) \right)
\]

is an axiom.

The formula \( A(u) \) represents the “property” to be proved, for a free \( u \).

To prove \( \forall x \ A(x) \), we can

1. prove the base case \( A(0) \), and
2. prove the inductive case \( \forall x \ (A(x) \rightarrow A(s(x))) \).

The axiom **PA7** is just the translation of the Principle of Mathematical Induction into Predicate Logic.
These axioms imply all of the familiar properties of the natural numbers. For example,

**Theorem 20.2.1.**

\[ \emptyset \vdash \forall x \, (\neg(s(x) \approx x)) \]

**Proof.** The formula \( A(u) \) is \((\neg(s(u) \approx u))\).

**Base Case:** The base case is \( A(0) \), which in this proof is \((\neg(s(0) \approx 0))\). We have

1. \( \emptyset \vdash \forall x \, (\neg(s(x) \approx 0)) \) (by PA1)
2. \( \emptyset \vdash (\neg(s(0) \approx 0)) \) (by (\forall -), (1))

**Inductive Case:** The inductive case is

\[ \forall x \,(A(x) \rightarrow A(s(x))) \],

which in this proof is

\[ \forall x \, ((\neg(s(x) \approx x)) \rightarrow (\neg(s(s(x)) \approx s(x)))) \].

We have

1. \( \neg(s(u) \approx u), s(u) \approx s(u) \vdash s(s(u)) \approx s(u) \) (by (∈))
2. \( \emptyset \vdash \forall x \, \forall y \, (s(x) \approx s(y) \rightarrow x \approx y) \) (by (∀ 2), (2))
3. \( \emptyset \vdash \forall y \, (s(s(y)) \approx s(y) \rightarrow s(u) \approx y) \) (by (→), (2))
4. \( \emptyset \vdash (s(s(u)) \approx s(u) \rightarrow s(u) \approx u) \) (by (→), (3))
5. \( \neg(s(u) \approx u), s(u) \approx s(u) \vdash (s(s(u)) \approx s(u) \rightarrow s(u) \approx u) \) (by (∈))
6. \( \neg(s(u) \approx u), s(u) \approx s(u) \vdash (s(s(u)) \approx s(u) \rightarrow s(s(u)) \approx s(u)) \) (by (→), (4))
7. \( \neg(s(u) \approx u), s(u) \approx s(u) \vdash (s(s(u)) \approx s(u) \rightarrow s(u) \approx s(s(u))) \) (by (→), (8))
8. \( \emptyset \vdash \forall x \, ((\neg(s(x) \approx x)) \rightarrow (\neg(s(s(x)) \approx s(x)))) \) (by (∀ 1), (9))

**Putting It All Together:** The PA7 axiom is

\[ ((A(0) \land \forall x \,(A(x) \rightarrow A(s(x))))) \rightarrow \forall x \, A(x)) \],

which in this proof is:

\[ ((\neg(s(0) \approx 0)) \land \forall x \, ((\neg(s(x) \approx x)) \rightarrow (\neg(s(s(x)) \approx s(x)))) \rightarrow \forall x \, (\neg(s(x) \approx x))) \].

We have

1. \( \emptyset \vdash (\neg(s(0) \approx 0)) \land \forall x \, ((\neg(s(x) \approx x)) \rightarrow (\neg(s(s(x)) \approx s(x)))) \) (by PA7)
2. \( \emptyset \vdash (\neg(s(0) \approx 0)) \) (by Base Case)
3. \( \emptyset \vdash \forall x \, ((\neg(s(x) \approx x)) \rightarrow (\neg(s(s(x)) \approx s(x)))) \) (by Induction Case)
4. \( \emptyset \vdash ((\neg(s(0) \approx 0)) \land \forall x \, ((\neg(s(x) \approx x)) \rightarrow (\neg(s(s(x)) \approx s(x)))) \) (by (∧), (2), (3))
5. \( \emptyset \vdash \forall x \, (\neg(s(x) \approx x)) \) (by (→), (1), (4))
21 Lecture 21 - Peano Arithmetic II

Outline

1. Commutativity of +
2. Soundness and Completeness of Peano Arithmetic

21.1 Commutativity of +

Theorem: Addition in Peano Arithmetic is commutative; that is,

\[ \emptyset \vdash_{PA} \forall x \forall y (x + y \approx y + x) . \]

(Notation “\( \emptyset \vdash_{PA} \)” or “\( PA \vdash \)” means “provable [in FD] using the \( \approx \) and \( PA \) axioms.”)

How can we find such a proof?

We must use induction (\[ PA7 \]). The key first step: choose a good formula \( A \) for the induction property.

Recall Axiom \[ PA7: \]

\[ \left( (A(0) \land \forall x (A(x) \rightarrow A(s(x)))) \rightarrow \forall x A(x) \right) \]

Choosing \( A \) to be \( \forall y (x + y \approx y + x) \) yields the instance

\[ (\forall y (0 + y \approx y + 0) \land \forall x \forall y (x + y \approx y + x) \rightarrow \forall y (s(x) + y \approx y + s(x))) \rightarrow \forall x \forall y (x + y \approx y + x). \]

Thus we must prove the base case

\[ \forall y (0 + y \approx y + 0) \]

and the inductive case

\[ \forall x \forall y (x + y \approx y + x) \rightarrow \forall y (s(x) + y \approx y + s(x)) . \]

Then one application of (\( \land + \)) one application of (\( \rightarrow - \)) yield the desired formula

\[ \emptyset \vdash_{PA} \forall x \forall y (x + y \approx y + x) . \]

How should we prove \( \forall y (0 + y \approx y + 0) \)?

We must use induction, in order to prove the base case of the main result.

To control the complication, let’s make it a lemma:
Lemma 21.1.1. Peano Arithmetic has a proof of $\emptyset \vdash_{PA} (\forall y (0 + y \approx y + 0))$.

Plan, to prove lemma: induction ($\text{PA}_7$) with $0 + y \approx y + 0$ for $P$.

**Basis:** Prove $0 + 0 \approx 0 + 0$. Immediate from $(\approx +)$.

**Inductive step:** Prove $\forall y (0 + y \approx y + 0) \rightarrow (0 + s(y) \approx s(y) + 0)$.

Pick $u$ free. Then assume $0 + u \approx u + 0$. Then

\[
\begin{align*}
0 + s(u) &\approx s(0 + u) & \text{PA}_4 \\
&\approx s(u + 0) & \text{Assumption} \\
&\approx s(u) & \text{PA}_3 + (\forall -) \\
&\approx s(u) + 0 & \text{PA}_3 + (\forall -) .
\end{align*}
\]

Applying $(\rightarrow +)$ and generalization $(\forall +)$ yield the required formula.

Using $\text{PA}_7$, $(\land +)$ and $(\rightarrow -)$ completes the proof of the lemma.

Now for the main proof of the Lemma, We have

\begin{enumerate}
    
    \item $\emptyset \vdash ((0 + 0 = 0 + 0) \land \forall y ((0 + y = y + 0) \rightarrow (0 + s(y) = s(y) + 0))))$ (by $\text{PA}_7$)
    
    \item $\emptyset \vdash (0 + 0 = 0 + 0)$ (by Base Case)
    
    \item $\emptyset \vdash \forall y ((0 + y = y + 0) \rightarrow (0 + s(y) = s(y) + 0)))$ (by Induction Case)
    
    \item $\emptyset \vdash ((0 + 0 = 0 + 0) \land \forall y ((0 + y = y + 0) \rightarrow (0 + s(y) = s(y) + 0))))$ (by $(\land +)$, (2), (3))
    
    \item $\emptyset \vdash \forall y (0 + y = y + 0)$ (by $(\rightarrow -)$, (1), (4))
\end{enumerate}

This completes the proof of the Lemma for the base case.

The next Lemma provides the content for the induction step of the induction case.

Lemma 21.1.2. For each free variable $u$,

\[
\{ (\forall y (u + y \approx y + u)) \} \vdash_{PA} \forall y (s(u) + y \approx y + s(u)) .
\]

Plan of proof: induction on variable $y$ with formula $s(u) + y \approx y + s(u)$ for $A$.

**Basis:** prove $s(u) + 0 \approx 0 + s(u)$. The target follows from $(\forall y (0 + y \approx y + 0))$ plus $(\forall -)$ plus the symmetry of $\approx$.

**Ind. step:** $\left( \forall z ((s(u) + z \approx z + s(u)) \rightarrow (s(u) + s(z) \approx s(z) + s(u))) \right)$.

Assuming $s(u) + w \approx w + s(u)$ yields

\[
\begin{align*}
s(u) + s(w) &\approx s(s(u) + w) & \text{PA}_4 \\
&\approx s(w + s(u)) & \text{Assumption} \\
&\approx s(s(w + u)) & \text{PA}_4 .
\end{align*}
\]
The premise of the lemma implies \( u + s(w) \approx s(w) + u \); thus
\[
\begin{align*}
    s(w) + s(u) & \approx s(s(w) + u) & \text{PA4} \\
    & \approx s(u + s(w)) & \text{premise} \\
    & \approx s(s(u + w)) & \text{PA4}.
\end{align*}
\]

The premise also implies \( u + w \approx w + u \). Then
\[
s(u) + s(w) \approx s(w) + s(u),
\]
as required.

Using \( \text{PA7}, (\land +) \) and \( (\rightarrow -) \) completes the proof of the lemma.

Now for the main proof, taking \( A = (\forall y (u + y \approx y + u)) \), We have
\[
\begin{align*}
    (1) & \quad \emptyset \vdash (\forall y (0 + y \approx y + 0) \land \forall x ((\forall y (x + y \approx y + x)) \rightarrow ((\forall y (s(x) + y \approx y + s(x))))) \\
    & \quad \rightarrow \forall x (\forall y (x + y \approx y + x)) & \text{(by PA7)} \\
    (2) & \quad \emptyset \vdash \forall y (0 + y \approx y + 0) & \text{(by Base)} \\
    (3) & \quad \emptyset \vdash \forall x ((\forall y (x + y \approx y + x)) \rightarrow ((\forall y (s(x) + y \approx y + s(x))))) & \text{(by Induction)} \\
    (4) & \quad \emptyset \vdash (\forall y (0 + y \approx y + 0) \land \forall x ((\forall y (x + y \approx y + x)) \rightarrow ((\forall y (s(x) + y \approx y + s(x))))) & \text{(by (\land +))} \\
    (5) & \quad \emptyset \vdash \forall x (\forall y (x + y \approx y + x)) & \text{(by (\rightarrow -))}
\end{align*}
\]

The other familiar properties of addition and multiplication have similar proofs. One can continue: divisibility, primeness, etc.

21.2 Soundness and Completeness of Peano Arithmetic

PA is sound:
- We already proved that Formal Deduction for Predicate Logic is sound.
- PA augments Predicate FD by adding the 7 Peano Axioms, in the semantic context of the arithmetic of \( \mathbb{N} \).
- In the semantic context of the arithmetic of \( \mathbb{N} \), each Peano axiom obviously holds.
- Hence any result proved in PA must hold in the semantic context of the arithmetic of \( \mathbb{N} \).

PA is not complete:
- Gödel’s Incompleteness Theorem tells us this.
- However, constructing an example that witnesses this is well beyond the scope of CS 245.

22 Lecture 22 - Decidability/Undecidability I

Outline
22.1 Intro to Computability

- Intuitively, something is computable if it can be calculated by a systematic procedure (aka an algorithm).
- We say that a function is computable if there is an algorithm that a computer could execute to compute it.

Remarks:

1. Decidable problems and computable functions are closely connected.
2. For more details about decidable problems and computable functions, take CS 360/365.

One might think that, given enough resources and a sufficiently sophisticated program, a computer could solve any problem.

However, solving some problems cannot be automated.

22.2 Decision Problems

Definition 22.2.1. A decision problem is a problem which calls for an answer of either yes or no, given some input.

- These examples are decidable:
  - Given a formula of Propositional logic, is it satisfiable?
    * A: Yes, the algorithm should be obvious.
  - A Variation: Given a formula of Propositional logic, is it valid?
    * A: Yes, the algorithm should be obvious.
  - Given a positive integer, is it prime?
    * A: Yes, an algorithm was given in Math 135.
- By the end of the section, we will show that this example is undecidable:
  - Given a program P, and an input I, will program P terminate when run with input I? (the Halting Problem)

Definition 22.2.2. A decision problem is decidable if there is an algorithm that, given an input to the problem,

- outputs yes (or 1) if the input has answer yes, and
- outputs no (or 0) if the input has answer no.

A decision problem is undecidable if it is not decidable.

Remarks:
1. The algorithm must always complete after **finitely many steps**. If there is even one case in which the candidate algorithm will not finish in finitely many steps, then it is not actually an algorithm.

2. To prove that a decision problem is decidable, **write down an algorithm to decide it**.

3. A decision problem may be undecidable, and yet have particular choices of input for which the correct yes/no answer can be determined. The existence of special choice of input for which the question can be decided does not contradict the general result of undecidability. To say that a decision problem is undecidable is to say that no algorithm exists to give the correct yes/no answer **for every input**.

### 22.3 Decision Problems - Decidable

- Many nice decision problems can be stated in terms of the question of membership in some set.
- Lots of non-trivial examples exist for subsets $S \subseteq \mathbb{N}$, where $\mathbb{N}$ denotes the Natural numbers.

**Definition 22.3.1.** Let $S \subseteq \mathbb{N}$ be any subset. The $S$-**membership problem** asks for an arbitrary $x \in \mathbb{N}$, is $x \in S$?

**Definition 22.3.2.** A set $S \subseteq \mathbb{N}$ is called **decidable** (resp undecidable) if the $S$-membership problem is decidable (resp undecidable).

**Fact:** Some $S$s are decidable; others are undecidable.

**Examples:**

1. Suppose that $S \subseteq \mathbb{N}$ is decidable. Prove that the **complement**, $S^c = \mathbb{N}\setminus S = \{n \in \mathbb{N} \mid n \notin S\}$, is decidable.

   **Proof:** The following algorithm decides membership in $S^c$:
   
   (a) Let $x \in \mathbb{N}$ be given.
   
   (b) Use the decider for $S$ to determine whether $x \in S$.
   
   (c) If $x \in S$ then return “no”.
   
   (d) If $x \notin S$ then return “yes”.

2. Suppose that $S \subseteq \mathbb{N}$ is finite. Does it follow that $S$ is decidable?
   
   – **A:** Yes. The required algorithm is obvious.

3. Suppose that $S \subseteq \mathbb{N}$ is infinite. Does it follow that $S$ is undecidable?
   
   – **A:** No. For a counterexample, let $S$ be $\{y \in \mathbb{N} \mid y \equiv 0 \text{ mod } 2\}$, the subset of even numbers. The algorithm to decide membership in $S$ is obvious.

4. For examples of $S$s which are undecidable, take CS 360/365.

**A few more examples are on A10, and this week’s tutorial.**
22.4 Decision Problems - Undecidable

Our Strategy For Proving Undecidability: For now, we take for granted that the Halting Problem is undecidable. (We will prove this fact during Lecture 23.) To use this fact to show another decision problem, $D$, is undecidable, we follow this outline (for CS 245 purposes):

1. Towards a contradiction, suppose that $D$ is decidable.
2. Let $(P I)$ be any candidate for the Halting Problem.
3. Use the decider for $D$ to determine whether $(P I)$ halts.
4. This provides a decider for the Halting Problem.
5. Since the Halting Problem is undecidable, therefore we have a contradiction.
6. Therefore $D$ must be undecidable.

This week’s tutorial notes will show some examples of this technique.

Examples:

1. The HaltOnEveryInput decision problem asks
   Given any program $Q$, does $Q$ halt for every input?
   Prove that the HaltOnEveryInput problem is undecidable.
   **Solution:**
   (a) Towards a contradiction, suppose that the HaltOnEveryInput problem is decidable. Let $D$ be a decider for the HaltOnEveryInput problem.
   (b) Let $(P I)$ be any candidate for the Halting Problem.
   (c) Construct a new program, $Q$ to carry out the following algorithm:
      i. Given any input, $x$,
      ii. Ignore the input, $x$, and run $P$ on input $I$.
      iii. If $P$ halts on input $I$, then halt and accept $x$.
      iv. $P$ may run forever on input $I$.
      Then by construction,
      i. $Q$ halts and accepts every input $x$ if $P$ halts on input $I$, and
      ii. $Q$ runs forever on every input $x$ if $P$ runs forever on input $I$.
   (d) Now feed $Q$ into $D$, and return the same answer for whether $P$ halts on input $I$. This provides a decider for the Halting Problem.
   (e) Since the Halting Problem is undecidable, therefore this is a contradiction.
   (f) Therefore the HaltOnEveryInput decision problem is also undecidable.

23 Lecture 23 - Decidability/Undecidability II

Outline

1. The Undecidability of the Halting Problem
2. Reductions
3. More Examples
23.1 The Undecidability of the Halting Problem

From now on, assume that all of our programs consume and produce natural numbers.

In this subsection, we present an argument that the Halting Problem is undecidable (Theorem \[23.1.1\]).

**Why We Care:** We use the undecidability of the Halting Problem to prove that other decision problems are undecidable.

**Theorem 23.1.1.** *The Halting Problem is undecidable.*

**Proof.**

- Towards a contradiction, suppose that the Halting Problem is decidable.
- In other words, assume that the definition \(\text{define (halts? P I) ...}\) satisfies this contract:
  
  ;; Contract: halts?: SchemeProgram Input → boolean
  ;;
  ;; If the evaluation of \((P I)\) halts, then \((halts? P I)\)
  ;; halts with value true, and
  ;; If the evaluation of \((P I)\) does not halt, then
  ;; \((halts? P I)\) halts with the value false.
  ;;

  ;; Example: \((halts? loop loop)\) returns false,
  ;; given definition \((define (loop x) (loop loop))\)
  (define (halts? P I)

  )

- Define the DrRacket program, \(C\):

  (define (diagHalts? P)
      (cond [(halts? P P) (loop loop)]
           [else #f]))

- Here, I wave my hands over the fact that any program and any input can be encoded into a natural number, so that using the identifier for a program as input to a program makes sense.
- We could explain how this could be done systematically, if we had enough time.
- Now consider the key question of whether \(C\) halts when processing input \(C\).
- If \(C\) halts when processing input \(C\), then
  - \((halts? C C)\) halts and returns \#t.
  - By the first clause in the cond statement, \(C\) runs forever when processing input \(C\).
  - This is a contradiction.
- If \(C\) runs forever when processing input \(C\), then
  - \((halts? C C)\) halts and returns \#f.
By the second clause in the `cond` statement, `C` halts when processing input `C`.

- This is a contradiction.
- All possibilities lead to contradiction.
- Therefore we are done.

Remarks:

1. This is an informal definition of a Turing reduction which suffices for our purposes in CS 245. For the formal definition of a many-to-one reduction, take CS 360.

2. The desired contradiction occurs somewhere on the diagonal of this table:

<table>
<thead>
<tr>
<th>programs</th>
<th>inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 1 2 ⋯</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>2</td>
<td>*</td>
</tr>
<tr>
<td>⋮</td>
<td></td>
</tr>
</tbody>
</table>

### 23.2 Reductions

A reduction is a technique for using the undecidability of the Halting Problem to argue that a new decision problem is undecidable.

**Example:** Consider the decision problem

Given any program `Q`, does `Q` run forever when processing a particular input `n = 0`?

Prove that this decision problem is undecidable.

**Solution:** The key idea to construct `Q` is to start by running `P` with input `I`, **ignoring the input to `Q`**. This causes `Q` to behave the same way regardless of its input, in other words for all of its inputs.

**Remarks:**

1. A reduction shows how, if we can decide one decision problem (e.g. the given decision problem), it follows that we can decide another decision problem (e.g. the Halting Problem). In the presence of a reduction, if the “another problem” is already known to be undecidable, it follows that the “one problem” must also be undecidable.

2. The 5-step template for writing a proof of undecidability will suffice for A10, and for the final exam.

For a contradiction, suppose that the given decision problem is decidable. Let `(P I)` be any candidate for the Halting Problem. Construct a new program, `Q`, which will

- ignore its input and run `P` with input `I`, and
• if \( P \) halts when run with input \( I \), then halt and return 0.

By construction, \( Q \) halts for all inputs (including 0) if and only if \( P \) halts when run with input \( I \).

Now feed \( Q \) in to the decider for the given decision problem, and return the opposite answer for whether \( (P, I) \) halts. This provides a decider for the Halting Problem. Since the Halting Problem is undecidable, therefore we have a contradiction. This completes the proof.

• The Significance of a Reduction
  – The first bullet says “If there is a reduction from \( A \) to \( B \), and \( B \) is decidable, then \( A \) is also decidable”.
  – The second bullet says “If there is a reduction from \( A \) to \( B \), and \( A \) is undecidable, then \( B \) is also undecidable”.
  – Note that the second bullet is just the contrapositive of the first.

23.3 More Examples

• Provability in Predicate Logic is Undecidable
  – Remark: This is an example of a reduction.
  – If you want to learn more about reductions, then take CS 360.
  – The instruction “Base the formula on Step” refers to the generalized technique of translating from Scheme into Predicate Logic, which we did not present this term.

• Another Undecidable Problem
  – Post’s Correspondence Problem is the question: “Given an instance of PCP, does a solution exist?”
  – Post’s Correspondence Problem is undecidable.
  – For another application of Post’s Correspondence Problem (to prove that a decision problem about Context-Free Languages is undecidable), take CS 360.

24 Lecture 24 - Program Verification

Outline

1. Program Verification