Notes for Discrete Mathematics (2. Ed) by Rosen: Chapter 1: Logic and Proofs

Andreas Hansen

May 2025

Chapter 1.2: Applications of Propositional Logic (pg. 17-23)

Translating English Sentences (pg. 17–18)

Example 1: Translating from English to a Logical Expression

Statement:

"You can access the Internet from campus only if you are a computer science major or you are not a freshman."

Strategy: Rather than representing the entire sentence as a single propositional variable (e.g. p), which would not help in understanding or reasoning, we break it into smaller propositions and connect them using logical operators.

Let:

- a: You can access the Internet from campus
- c: You are a computer science major
- f: You are a freshman

Translation:

$$a \to (c \lor \neg f)$$

Example 2: Translating from English to a Logical Expression

Statement:

"You cannot ride the roller coaster if you are under 4 feet tall unless you are older than 16 years old"

Strategy: Same as previous Example 1.

Let:

- q: You can ride the roller coaster
- r: You are under 4 feet tall
- s: You are older than 16 years old

Translation:

$$(r \land \neg s) \to \neg q$$

Explanation:
The sentence contains an exception with "unless", which in logic means "if not".
So "*unless you are older than 16*" is logically equivalent to "*if you are not older than 16*", or $\neg s$.
Thus, the phrase becomes: "*If you are under 4 feet tall and not older than 16, then you cannot ride the roller coaster.*" That is:

$$(r \land \neg s) \to \neg q$$

System Specifications (pg. 18-19)

Example 1: Express a specification using logical connectives.

Specification:

"The automated reply cannot be sent when the file system is full"

Let:

- p: The automated reply can be sent
- q: The file system is full

Then we can express the negation of p as $\neg p$:

"It is not the case that the automated reply can be sent"
or simply: *"The automated reply cannot be sent"*

Translation (as conditional statement):

$$q \to \neg p$$

Remark 1

System specifications should be **consistent**, that is, they should not contain conflicting requirements that could be used to derive a contradiction.
When specifications are not consistent, there would be no way to develop a system that satisfies all specifications.

Example 2: Determine Consistency of System Specifications

Consider the following specifications:

1. *The diagnostic message is stored in the buffer or it is retransmitted.*
2. *The diagnostic message is not stored in the buffer.*
3. *If the diagnostic message is stored in the buffer, then it is retransmitted.*

Let:

- p: The diagnostic message is stored in the buffer
- q: The diagnostic message is retransmitted

Translations:

1. $p \lor q$
2. $\neg p$
3. $p \to q$

Truth Assignment and Evaluation:
Assume: $p = \mathrm{F},\ q = \mathrm{T}$

- (1) $p \lor q = \mathrm{F} \lor \mathrm{T} = \mathrm{T}$
- (2) $\neg p = \neg \mathrm{F} = \mathrm{T}$
- (3) $p \to q = \mathrm{F} \to \mathrm{T} = \mathrm{T}$ (a conditional with false hypothesis is always true)

Conclusion:
Under the assignment $p = \mathrm{F},\ q = \mathrm{T}$, all three specifications evaluate to true.
Therefore, the system specifications are consistent.

Example 3: Determine Consistency of System Specifications (2)

Consider the following specifications:

1. *The diagnostic message is stored in the buffer or it is retransmitted.*
2. *The diagnostic message is not stored in the buffer.*
3. *If the diagnostic message is stored in the buffer, then it is retransmitted.*
4. *The diagnostic message is not retransmitted.*

Let:

- p: The diagnostic message is stored in the buffer
- q: The diagnostic message is retransmitted

Translations:

1. $p \vee q$
2. $\neg p$
3. $p \to q$
4. $\neg q$

Truth Assignment and Evaluation:
Assume: $p = \mathrm{F},\ q = \mathrm{T}$

- (1) $p \vee q = \mathrm{F} \vee \mathrm{T} = \mathrm{T}$
- (2) $\neg p = \neg \mathrm{F} = \mathrm{T}$
- (3) $p \to q = \mathrm{F} \to \mathrm{T} = \mathrm{T}$ (a conditional with a false hypothesis is always true)
- (4) $\neg q = \neg \mathrm{T} = \mathrm{F} \leftarrow$ **Contradiction!**

Conclusion:
Under the truth assignment $p = \mathrm{F},\ q = \mathrm{T}$, the first three specifications are satisfied. However, the fourth specification ($\neg q$) is false.
Therefore, the system specifications are inconsistent.

Boolean Searches (pg. 19-20)

Definition: Boolean searching refers to a search technique that uses tools called operators and modifiers to limit, widen, and refine your search results

Remark 1

In Boolean searches, the connective AND is used to match records that contain both of two search terms, the connective OR is used to match one or both of two search terms, and the connective NOT (sometimes written as AND NOT) is used to exclude a particular search term

Remark 2

Web Page Searching: Most Web search engines support Boolean searching techniques, which is useful for finding Web pages about particular subjects

Example 1: Boolean Searching Using Logical Operators

Statement:

You want to search for Web pages about universities in New Mexico.

Strategy: We can use Boolean logic operators like AND, OR, and NOT to construct search expressions that include only the pages we want. We'll analyze what happens with different Boolean formulations.
Let:

- N: The page contains the word "NEW"
- M: The page contains the word "MEXICO"
- U: The page contains the word "UNIVERSITIES"

Boolean Search:

$$N \wedge M \wedge U \quad \text{(Logic operators)}$$

NEW AND MEXICO AND UNIVERSITIES

This query returns all pages that contain the words "NEW", "MEXICO", and "UNIVERSITIES".
Observation: This will include:

- Relevant pages about universities in New Mexico (desired)
- Irrelevant pages such as ones about "new universities in Mexico" (not desired)

Improved Search Strategy: To reduce irrelevant results, use quotation marks to group the phrase "NEW MEXICO" as a single unit:

$$\texttt{"NEW MEXICO"} \wedge U \quad \text{(Logic operators)}$$

"NEW MEXICO" AND UNIVERSITIES.

This ensures the search engine treats "NEW MEXICO" as a proper phrase and reduces matches like "new universities in Mexico".
Conclusion: Boolean search logic mimics propositional logic. Using grouping (e.g., quotation marks) and appropriate connectives improves search precision.

Example 2: Boolean Searching with OR and AND Precedence

Statement:

You want to search for Web pages about universities in either New Mexico or Arizona.

Strategy: We use Boolean logic to capture the intent: pages that mention "universities" and either: - both "NEW" and "MEXICO", or - "ARIZONA".

Let:

- N: The page contains the word "NEW"
- M: The page contains the word "MEXICO"
- A: The page contains the word "ARIZONA"
- U: The page contains the word "UNIVERSITIES"

Boolean Search:

$$((N \wedge M) \vee A) \wedge U$$

`(NEW AND MEXICO OR ARIZONA) AND UNIVERSITIES`

Note on Precedence: In Boolean logic (and in most search engines), the AND operator has higher precedence than OR. So this is interpreted as:

((NEW AND MEXICO) OR ARIZONA) AND UNIVERSITIES

Observation: This query matches:

- Pages about universities in New Mexico (contain NEW, MEXICO, and UNIVERSITIES)
- Pages about universities in Arizona (contain ARIZONA and UNIVERSITIES)
- But also possibly irrelevant pages such as:
  - Pages with NEW and MEXICO but not universities
  - Pages about Arizona universities not specifically comparing them with New Mexico

Improved Search Strategy: Use grouping with parentheses or quotation marks to better control matching, especially when using search engines like Google:

`"NEW MEXICO" OR ARIZONA AND UNIVERSITIES`

This still depends on how the search engine parses precedence. Some engines may interpret it as:

$$\texttt{"NEW MEXICO"} \vee (\texttt{ARIZONA} \wedge \texttt{UNIVERSITIES})$$

which may not be what you want.

Conclusion: Operator precedence matters in Boolean search. To refine results, group expressions using parentheses or quotes, and be aware that not all engines use the same precedence rules.

Example 3: Boolean Searching Using NOT (Exclusion)

Statement:

You want to find Web pages about universities in Mexico, but exclude those about New Mexico.

Strategy: A naive search for `MEXICO AND UNIVERSITIES` will match both:

- Pages about universities in Mexico (desired)
- Pages about universities in New Mexico (undesired)

To avoid including pages that mention "NEW", we use the **NOT** operator to exclude them.

Let:

- M: The page contains the word "MEXICO"
- U: The page contains the word "UNIVERSITIES"
- N: The page contains the word "NEW"

Boolean Search:

$$(M \land U) \land \neg N$$

```
(MEXICO AND UNIVERSITIES) NOT NEW
```

Observation: This search will include:

- Pages that mention both "MEXICO" and "UNIVERSITIES"
- But **exclude** any pages that also contain the word "NEW" — removing false positives related to "New Mexico"

Search Engine Note: Many engines like Google do not use the word `NOT`, but instead use the minus sign (`-`) to indicate exclusion.

Google-style equivalent:

```
MEXICO UNIVERSITIES -NEW
```

Conclusion: The NOT operator (or `-` in practice) is useful in Boolean searching for filtering out results that include unwanted keywords. When searching for content that overlaps with similar terms, this exclusion strategy improves result relevance.

Logic Puzzles (pg. 20-22)

Example 1: King's Daughter Treasure Logic Puzzle (1/2)

Puzzle:

- As a reward for saving his daughter from pirates, the King has given you the opportunity to win a treasure hidden inside one of three trunks.
- The two trunks that do not hold the treasure are empty.
- To win, you must select the correct trunk.
- Trunks 1 and 2 are each inscribed with the message "*This trunk is empty,*" and Trunk 3 is inscribed with the message "*The treasure is in Trunk 2.*"
- The Queen, who never lies, tells you that only one of these inscriptions is true, while the other two are wrong.
- Which trunk should you select to win?

Strategy: Let p_i be the proposition that the treasure is in Trunk i, for $i = 1, 2, 3$. To translate into propositional logic the Queen's statement that exactly one of the inscriptions is true, we observe that the inscriptions on Trunk 1, Trunk 2, and Trunk 3, are:

$$\begin{array}{rl} \neg p_1 & \text{(Not in Trunk 1)} \\ \neg p_2 & \text{(Not in Trunk 2)} \\ p_2 & \text{(In Trunk 2)} \end{array}$$

Then we have to construct the 3 different scenarios:

- Inscription in Trunk 1 is correct:

$$\neg p_1 \land \neg(\neg p_2) \land \neg p_2$$

- Inscription in Trunk 2 is correct:

$$\neg(\neg p_1) \land \neg p_2 \land \neg p_2$$

- Inscription in Trunk 3 is correct:

$$\neg(\neg p_1) \land \neg(\neg p_2) \land p_2$$

Only one scenario can be correct, so we have to combine them with the disjunction operator $\lor$ and we get:

$$(\neg p_1 \land \neg(\neg p_2) \land \neg p_2) \lor (\neg(\neg p_1) \land \neg p_2 \land \neg p_2) \lor (\neg(\neg p_1) \land \neg(\neg p_2) \land p_2)$$

Reducing and simplifying (1/2):

$$\neg(\neg p_i) = p_i$$

$$(\neg p_1 \land p_2 \land \neg p_2) \lor (p_1 \land \neg p_2 \land \neg p_2) \lor (p_1 \land p_2 \land p_2)$$

$$(p_1 \land \neg p_2) \lor (p_1 \land p_2)$$

Now using distributive law: $p_1 \land (\neg p_2 \lor p_2)$

Example 1: King's Daughter Treasure Logic Puzzle (2/2)

Reducing and simplying (2/2):

$$(p_1 \wedge \neg p_2) \vee (p_1 \wedge p_2) = p_1 \wedge (\neg p_2 \vee p_2)$$

But we know that it has to be true that either $\neg p_2$ or p_2 and we can denote this as

$$\neg p_2 \vee p_2 = \mathbf{T}$$

This gives

$$p_1 \wedge (\neg p_2 \vee p_2) = p_1 \wedge \mathbf{T}$$

But this is equivalent to

$$p_1 \wedge \mathbf{T} = p_1$$

Conclusion:
So the treasure is in Trunk 1 (that is, p_1 is true), and p_2 and p_3 are false; and the inscription on Trunk 2 is the only true one.

Example 2: Raymond Smullyan´s Knights and Knaves

Puzzle:

- In [Sm78] Smullyan posed many puzzles about an island that has two kinds of inhabitants, knights, who always tell the truth, and their opposites, knaves, who always lie.
- You encounter two people A and B.
- What are A and B if A says "*B is a knight*" and B says "T*he two of us are opposite types"?*

Strategy: Let p and q be the statements that A is a knight and B is a knight, respectively, so that

$$\begin{array}{ll} \neg p & \text{(A is a knave)} \\ \neg q & \text{(B is a knave)} \end{array}$$

Then we have to construct the 2 different scenarios:

- **A is a knight and tells the truth:**
  Then p is True, and B is also a knight (q is True) and tells the truth.
  But that means the statement B says also has to be true and this cannot be, since they are the same type. So this scenario is impossible and both A and B can´t be knights
- **A is a knave and lies:**
  Then A lies about B being a knight. This means that B also must be a knave. This also works when looking at B´s statement, because B also lies that the are different types.

Conclusion:
We can conclude that both A and B are knaves.

Example 3: Muddy children puzzle

Puzzle:

- A father tells his two children, a boy and a girl, to play in their backyard without getting dirty. However, while playing, both children get mud on their foreheads
- When the children stop playing, the father says "A*t least one of you has a muddy forehead,*" and then asks the children to answer "*Yes*" or "*No*" to the question: *"Do you know whether you have a muddy forehead?"*
- The father asks this question twice.
- What will the children answer each time this question is asked, assuming that a child can see whether his or her sibling has a muddy forehead, but cannot see his or her own forehead?
- Assume that both children are honest and that the children answer each question simultaneously.

Strategy: Let s be the statement that the son has a muddy forehead and let d be the statement that the daughter has a muddy forehead.

When the father says that at least one of the two children has a muddy forehead, he is stating that the disjunction is true

$$s \vee d$$

Then we have to construct the 2 different times the father asks:

- **Round 1:**
  We assum that each children tells the true and at the same time, so they would both answer "*No*", since the son only knows d is true and daughter only knows s is true.
- **Round 2:**
  Each child now knows that they themselves must have mud on their forehead, since they both answered "*No*" on the 1. question and the father stated that at least one of them has a muddy forehead.

  They are both going to answer "*Yes*" this time.

Conclusion:
First time the father asks, both children answer "*No*".
Second time the father asks, both children answer "*Yes*".

Logic Circuits (pg. 22-23)

Definition: Logic circuit (digital circuit)

Receives input signals $p_1, p_2, \ldots, p_n$, each a bit [either 0 (off) or 1 (on)], and produces output signals $s_1, s_2, \ldots, s_n$, each a bit.

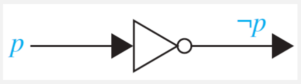
Remark 1

In this section we will restrict our attention to logic circuits with a single output signal; in general, digital circuits may have multiple outputs.

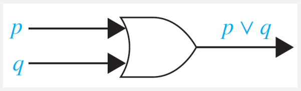
Definition: Basic Logic Gates

Complicated digital circuits can be constructed from three basic circuits, called gates:

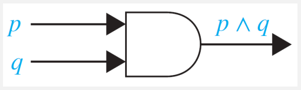
1. The **inverter**, or **NOT gate**, takes an input bit p and produces as output $\neg p$.



2. The **OR gate** takes two input signals p and q, each a bit, and produces as output the signal $p \vee q$.

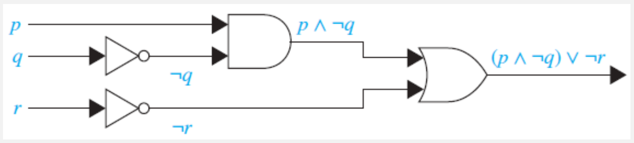


3. The **AND gate** takes two input signals p and q, each a bit, and produces as output the signal $p \wedge q$.



Example 1: Output for the combinatorial circuit

Determine the output for the combinatorial circuit



Strategy: We display the output of each logic gate in the circuit

- We see that the **AND gate** takes input of p and $\neg q$, the output of the inverter with input q, and produces $p \wedge \neg q$
- Next, we note that the **OR gate** takes input $p \wedge \neg q$ and $\neg r$ (the output of the inverter with input r), and produces the final output $(p \wedge \neg q) \vee \neg r$.

Example 2: Combinatorial circuit when want a certain output

Build a digital circuit that produces the output

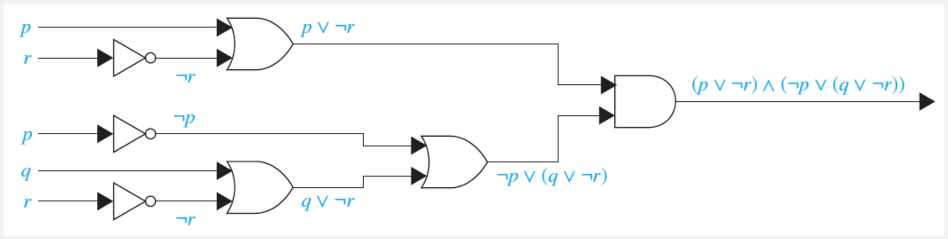
$$(p \vee \neg r) \wedge (\neg p \vee (q \vee \neg r))$$

when given input bits p, q, and r.

Strategy: We start with the output and "work backwards"

- **Output**: $(p \vee \neg r) \wedge (\neg p \vee (q \vee \neg r))$ is a conjunction (**AND gate**) between $(p \vee \neg r)$ and $(\neg p \vee (q \vee \neg r))$
- $\mathbf{p} \vee \neg \mathbf{r}$: This is a disjunction (**OR gate**) between p and $\neg r$
- $\neg \mathbf{r}$: This is a negation (**Inverter/NOT gate**) of r
- $\neg \mathbf{p} \vee (\mathbf{q} \vee \neg \mathbf{r})$: This is a disjunction (**OR gate**) between $\neg p$ and $q \vee \neg r$
- $\neg \mathbf{p}$: This is a negation (**Inverter/NOT gate**) of p
- $\mathbf{q} \vee \neg \mathbf{r}$: This is a disjunction between q and $\neg r$

Conclusion and figure:


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