Applications of Propositional Logic
1 & 2

CTPS 2018
Objectives

To understand application of simple propositional logic in

- Querying search engines.
- Analysis and synthesis of digital circuits.
- Querying databases.
Digital Logic

• *Digital circuits* are the physical components of a computing system.
• A digital circuit is an *electronic system* that enables a computer to perform arithmetic operations such as addition and multiplication among many other operations.
• These digital circuits are typically constructed by *combining logic gates* in various ways.
• A *logic gate* is an electronic device that implements a Boolean operator.
• It has *inputs* and produces a *single output* corresponding to the operator that it implements.
Think Binary!

- Let's examine a typical situation.
- You have some sort of device (Figure: A schematic representation of Logic Signal generator) that generates a logic signal.
- It could be a telephone that converts your voice signal into a sequence of zeros and ones.
- It could be the thermostat on the wall that generates a 1 when the temperature is too low, and a 0 when the temperature is above the set point temperature.
- The logic signal, takes on values of 0 (FALSE, OFF) or 1 (TRUE, ON).
  - The signal might really be a voltage, a switch closure, etc.
  - However, we want to think in terms of zeros and ones, not in terms of the values of the voltage.

Figure: A schematic representation of Logic Signal generator
Operations on Logic Signals

- Once we have the concept of a logic signal we can talk about operations that can be performed on logic signals.
- Begin by assuming we have two logic signals, A and B.
- Then assume that those two signals form an input set to some circuit that takes two logic signals as inputs, and has an output that is also a logic signal.
- The output, C, depends upon the inputs, A and B.
- There are many different ways that C could depend upon A and B.
- The output, C, is a function, - a logic function - of the inputs, A and B.
- This situation is represented as a logic circuit in:

Figure: A schematic representation of Logic Function.
Electrical circuits that implement AND and OR

- To discuss electrical circuits that implement AND and OR, let us consider for each case, a battery is connected to a light bulb and two switches.

- For the AND circuit (Figure: AND circuit), it is apparent that both switches must be closed in order to light up the bulb.

- If either one of the switches in the OR circuit (Figure: OR circuit) is closed, the light bulb will be illuminated.
Logical gates

- *Logic gates* are usually implemented using transistors as switches.

- In the context of digital logic, a battery can be thought of as a power source with a voltage level of 1.

- Since the switches either supply full power or no power to the output device, there are only two signal levels present in the circuit.

- These two signals are usually denoted using 0, which corresponds to no power or False and 1, which corresponds to full power or True.
Truth Table

• If we think of two signals, A and B, as representing a truth value of two different propositions, then A could be either TRUE (a logical 1) or FALSE (a logical 0). B can take on the same values.

• Now consider a situation in which the output, C, is TRUE only when both A is TRUE and B is TRUE.

• We can construct a truth table for this situation.

• In the truth table (Figure: Truth Table), we insert all of the possible combinations of inputs, A and B, and for every combination of A and B we list the output, C.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
Schematic notation for Boolean logic gates

- AND, OR, and NOT logic gates are usually drawn using symbolic forms (Figure: Symbolic forms for AND, OR and NOT).

- Simple propositions can be represented by one of these logic gates.
  - $P$ and $Q$
  - $P$ or $Q$
  - Not $P$

- Compound propositions can be represented as a combination of these simple logic gates.

- Digital circuits are therefore equivalent to compound Boolean propositions.
Control two pumps

• You need to control two pumps that supply two different concentrations of reactant to a chemical process.
• The strong reactant is used when pH is very far from the desired value, and the weak reactant when pH is close to desired.
• You need to ensure that only one of the two pumps runs at any time.
• Each pump controller (S and W) responds to standard logic signals, that is when the input to the pump controller is 1, the pump operates, and when that input is 0, the pump does not operate.
• You have a bunch of two-input AND gates, OR gates and Inverters, and you need to design a logic circuit to control the pumps.
• You can generate a signal that is 1 when Pump S is ON, and 0 when Pump W is ON.

✓ Can you design the circuit?
More logic gates...

- Now, let us try to understand the NOR function.
- The truth table and the symbolic form is given in Figure: NOR Gate.
- \( \text{NOR} = \text{NOT(OR)} \).

✓ Can you try the truth table for the NAND function and give the symbolic form for it?
Solution

NAND = NOT(AND)
Compound Boolean Proposition

• The compound boolean proposition “P and not (Q or R)” is represented using logic gates in Figure: P and not (Q or R).

  ![Logic Circuits Diagram](image)

  Figure: P and not (Q or R)

• The “OR” operator is enclosed in parenthesis, that part of the expression is performed first.
• In a logic circuit, this implies that the “OR” gate occurs first in the path of the signal.
• The resulting expression, (Q OR R) is then inverted and finally fed into the “AND” gate to produce the result.
Try it yourself

✓ Give the truth table for the circuit.
✓ What do you think the circuit performs?

✓ What does this circuit implement?
Solution

- This circuit encodes the concept that $P$ is only equal to $Q$ whenever (a) both $P$ and $Q$ are True or whenever (b) both $P$ and $Q$ are False.
- It has an output of True whenever $P = Q$, otherwise the output is False.
- It is a circuit that performs an equivalency check on two inputs.
Solution
Note...

• When designing an electronic circuit we would prefer to use as few gates as possible.
• Since each gate increases the cost of the system, minimizing the number of logic gates results in a less expensive system.
• Also, fewer gates often also means that the circuitry is faster since a digital signal does not need to travel through as many gates.
• Since digital circuits are expressions of Boolean propositions, it is natural to use Boolean algebra to design and analyze complex digital circuitry.
Try it yourself

✓ Verify whether the given pair of circuits implement the same Boolean function.

Figure : A

Figure : B

Figure : C
Try it yourself

✓ Draw a digital circuit for each of the following propositions.

a. (P OR Q) AND R
b. P OR (Q AND R)
c. NOT (P AND (Q OR R))
d. (NOT(A))B + A(NOT(B))
Simple gates using combinations

**NOT x**

\[ \overline{x} \]

**x AND y**

\[ x \land y \]

**x OR y**

\[ x \lor y \]
SIMPLIFY THE FOLLOWING CIRCUIT
Answer

\[ AB + BC(B + C) \]

Distributing terms

\[ AB + BBC + BCC \]

Applying identity \( AA = A \) to 2nd and 3rd terms

\[ AB + BC + BC \]

Applying identity \( A + A = A \) to 2nd and 3rd terms

\[ AB + BC \]

Factoring \( B \) out of terms

\[ B(A + C) \]
Truth Table for the following Logic

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

It can be simplified by:

\[ ABC + A\overline{B}(\overline{A} + \overline{C}) \]  
\[ ABC + A\overline{B} + ABC \]

DeMorgan's theorem

sum of products form

\[ BA = AB \text{ and } AA = A \]

\[ AC(B + \overline{B}) + \overline{AB} \]

\[ AC + A\overline{B} \]

\[ A(C + \overline{B}) \]

\[ B + \overline{B} = 1 \]
Truth Table for the following Logic

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ A B C + A B \overline{C} + A \overline{B} C \]

with the "brute force" implementation:

\[ \overline{C} + C = 1 \]

An important trick in logic simplification is to realize that once you have a variable in a logic expression, you can OR that variable in again as many times as you like without changing the logic. It is also important to realize that if you have a lot of ORs, you may have redundancy.

\[ A (B + C) \]

\[ A B (C + \overline{C}) + A C (\overline{B} + B) \]

\[ A B + A C = A (B + C) \]

\[ A B C + A B \overline{C} + A \overline{B} C + A B C = A B C + A B C \]
LN #12
2. Search Engine

- A *web search engine* is a software system to search for information on the World Wide Web.

- It is a remotely accessible program that lets you do *keyword searches* for information on the Internet.

- The search engine takes your *phrase / keyword* and returns search engine results pages with a list of sites it deems relevant or connected to your searched keyword.

*List 10 Internet search engines.*
Top 10 Internet search engines

- **Google** - Offering everything from image searches, map searches, news searches, etc. With impressive keyword relevancy and a continuously improving search algorithm, it's easy to see why Google is still the reigning champ.
- **Mahalo** - Mahalo is a unique 'human-powered' search engine that employs a group of editors to manually sift and organize thousands of pieces of content.
- **Yahoo** - While Yahoo has been suffering as of late, it's still a classic and a popular search engine.
- **Bing** - The Microsoft powered search engine prides itself on being a "decision engine" by offering search suggestions on the side column and providing extra search options.
- **Ask** - Clean layout and handy results grouping.
- **AOL Search** - AOL continues to be used, primarily by people who still use AOL.
- **Blekko** - Blekko's clean, minimalist layout is easy to navigate, and /tags allow for grouping searches.
- **DogPile** - the once alternative to Google is getting a comeback and is a great alternative to bigger search engines.
- **Duck Duck Go** - Doesn't track your search history and avoids spammy sites.
- **The Internet Archive** - This search engine lets users travel back in time to see how web pages looked in years gone by. A very fun search engine to play around with.
Querying

• Consider a situation where you live in a city that has a grocery store named “People.” You would like to use a search engine to find the phone number and the hours of operation for this store.
• You use the term “people”.
• When this text is written, the returned results may be:
  • People Magazine website,
  • Yahoo's People search page,
  • The website of “Peoples Bank” ,
  • A Wikipedia article titled “People” ,
  • A Wikipedia article devoted to a band named “Peoples” , and
  • A link to a web page devoted to the “People of Walmart”.

✓ The query did not produce useful results!
Search Query

- A *search query* is a phrase that describes the information that the user seeks to obtain.

- *Single-word query*
  - The word may be ambiguous or frequently used across many different contexts.
    - Sometimes effective, but often fail to produce the right type of information.
    - Often not specific enough to provide useful results.
  - You need to narrow the focus of your query.
  - You should construct more specific search queries.
  - The Boolean operators AND, OR, and NOT are one technique that is commonly used to narrow a search query.
Conjunction in Search Queries

- A search query returns web pages that contain all the terms or phrases in the query.
- By default, all search queries that use more than one word use conjunction.
- As an example, consider searching for information about a high school classmate named Hannah Garcia.
- You first decide to use the search query “Hannah”.
- The query returns a flood of web pages related to Hannah's of all kinds; none of whom are related to the “Hannah Garcia” that was your high school friend.
- You decide to narrow your search
  - by forming the two-word query “Hannah Garcia.”
- The search engine understands this query to be the conjunction of the two words.
- It understands the query to mean “Hannah AND Garcia.”
- The only pages that are returned by the search engine are those web pages that contain both the word “Hannah” and the word “Garcia.”
Conjunction in Search Queries contd---

• Let us consider that unfortunately, you still cannot find any web page related to your high school classmate Hannah Garcia.

• As you think about how to improve your search results, you remember that she married someone with the surname “Mason”.

• You realize that although she may still use the surname “Garcia” in certain business contexts, she may instead use the surname “Mason”.

• You therefore attempt to construct a search query
  • “Hannah Garcia Mason” .

• This search query would be overly narrow since only pages containing all three words would be returned.

• The search engine would interpret this query to mean “find all web pages that contain the words Hannah AND Garcia AND Mason.”

✓ Is this what you wanted the search engine to search?
Disjunction in Search Queries

- Continuing with the previous example of finding your high school classmate...

- You therefore attempt to construct a search query that expresses the possibility that her last name is either Garcia or Mason.

- You therefore construct the search query
  - “Hannah AND (Garcia OR Mason).”

- This query correctly expresses your desire to find web pages that contain only one of the surnames Garcia or Mason but not both.
Negation in Search Queries

- Perhaps the search query “Hannah AND (Garcia OR Mason)” returns many web pages related to a well-known soccer player.
- You are certain that your high school classmate is not a soccer player and therefore you attempt to construct a search query that prevents soccer-related pages from appearing.
- Search engines use the NOT operator to exclude pages that contain certain words.
- You can therefore construct the following search query:
  - “Hannah AND (Garcia OR Mason) AND NOT soccer”.
- This query will find all pages that contain either the words “Hannah” and “Garcia” or the words “Hannah” and “Mason,” while preventing any web page containing the word “soccer.”
Note…

- Negation is expressed in some search engines as a dash (−) immediately preceding the word that should be excluded.
- In this case, the search query “Hannah AND (Garcia OR Mason) AND NOT soccer” would be expressed as “Hannah AND (Garcia OR Mason) AND−soccer”.
- Also note that most search engines require the logical operators to be entered in all capital letters.
- If the word is typed in lowercase, the search engine will likely return all web pages related to the word “and,” for example, rather than interpreting AND as an operator!
3. Database Queries

- Databases are software systems designed to efficiently store enormous amounts of data such that pieces of data in the database can be very quickly located and retrieved.

- Most databases store information in tables such that:
  - Each row of the table contains a set of data that belongs to a single record.
  - Each column of the table defines a field and every cell of the table contains one field for one record of data.

- Database systems are an indispensable component of almost every software system and used in almost every profession.

**Figure: Information in a table**

**Can you give a list of large databases?**
A list of large databases

• No Fly List is a large database of individuals who are not allowed to board a commercial aircraft
• The National Do Not Call Registry is a database containing the names of individuals that telemarketers are not allowed to solicit.
• The National Sex Offender Registry is used by law enforcement to track the location of convicted sex offender.
• Every bank and financial institution maintains a database of clients and accounts.
• Wikipedia maintains a database of articles.
• Biologists maintain large databases of DNA sequences.
• Politicians maintain large databases of registered voters.
Eg. A professional baker

• If you were a professional baker, you would have lists to keep track of: a list of customers, a list of products sold, a list of prices, a list of orders...and so on.
Eg. A political lobbyist

- The lobbyist wants to concentrate all of their fund-raising efforts on a narrow group of likely donors and has been given access to a database of individuals who have made similar donations in the past.
- Each record should contain person's name, the amount donated, their state of residence, and their party affiliation.
- *Structure of the table is given as:*

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>Age</th>
<th>Amount</th>
<th>State</th>
<th>Party</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record #1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record #2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Department of CSE, Coimbatore
• The *Donor table* for the fund-raising is given in Figure: Donor table.

• Each record consists of six fields: First, Last, Age, Amount, State, and Party.

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>Age</th>
<th>Amount</th>
<th>State</th>
<th>Party</th>
</tr>
</thead>
<tbody>
<tr>
<td>William</td>
<td>Shell</td>
<td>61</td>
<td>1,300</td>
<td>WI</td>
<td>D</td>
</tr>
<tr>
<td>Lisa</td>
<td>Dough</td>
<td>19</td>
<td>125</td>
<td>WY</td>
<td>T</td>
</tr>
<tr>
<td>Helen</td>
<td>Lobby</td>
<td>35</td>
<td>1,200</td>
<td>CA</td>
<td>R</td>
</tr>
<tr>
<td>Reggie</td>
<td>Green</td>
<td>33</td>
<td>800</td>
<td>NY</td>
<td>R</td>
</tr>
<tr>
<td>Harvey</td>
<td>Levirage</td>
<td>41</td>
<td>500</td>
<td>NY</td>
<td>I</td>
</tr>
<tr>
<td>Robin</td>
<td>Round</td>
<td>32</td>
<td>200</td>
<td>NY</td>
<td>D</td>
</tr>
<tr>
<td>Jennifer</td>
<td>Dichali</td>
<td>38</td>
<td>650</td>
<td>WI</td>
<td>D</td>
</tr>
</tbody>
</table>

Figure: Donor table

• The political party is denoted as D for Democrats, R for Republicans, T for the Tea Party, and I for all independents.

• There are seven rows and hence we understand that the table contains seven records.
Querying the database

✓ Perhaps the lobbyist wants to solicit individuals who have made similar donations of at least $500.

• The list of potential donors can be generated by selecting a subset of the individual donors in the database.
• In this case, the lobbyist will generate this list
  • by scanning down the column labeled ‘Amount’ and
  • selecting those rows having a value of at least 500 in the ‘Amount’ column.

✓ Since there are only seven records in our naïve example, generating the list of potential donors is easily done, but most real-world databases contain many millions of records.
✓ The enormous size of data in most real-world databases requires sophisticated software to extract and analyze data of interest.
Structured Query Language

- Fortunately, databases are designed to efficiently locate records in response to well-formed queries.
- Although somewhat similar to a web search, a database query requires more precise structure and is generally written in a language known as SQL (Structured Query Language).
- Records in a database are located by issuing a select statement.
- The general form of a select statement is:

  ```
  SELECT field1, field2, ..., fieldN FROM table WHERE criteria
  ```

  - the italicized elements represents the details that control what data is retrieved.
To select the names of donors who have contributed \textit{at least} $500

• Since we are only interested in the names of the individuals, we would select only the First and Last fields.
• We are selecting data from the Donor table and the criteria that determines which elements to select is expressed as Amount $\geq 500$.
• The proper SQL query is given as:

\begin{verbatim}
SELECT First, Last FROM Donor WHERE Amount $\geq 500$
\end{verbatim}

• The criterion of this select statement is a \textit{simple logical predicate} that locates and retrieves only those records of interest to the lobbyist.
• The result of the select statement is
  • a list of five donors: William Shell, Helen Lobby, Reggie Green, Harvey Levirage, and Jennifer Dichali.
A compound logical predicate

• Consider, however, a scenario where the lobbyist wants to specifically target only those donors who are under 40 years of age and have contributed at least $500 in the past.

• This requires a criterion that is expressed as a compound logical predicate:

\[
\text{SELECT First, Last FROM Donor}
\]

\[
\text{WHERE Amount} \geq 500 \text{ AND Age} < 40
\]

• A list of the three donors: Helen Lobby, Reggie Green, and Jennifer Dichali.
Try it yourself

✓ Perhaps the lobbyist wants to specifically target only those donors who live in either the coastal states of New York or California regardless of their age or their past donations.

   The appropriate SQL query is expressed as:

   _____________________________________________________
   _____________________________________________________
   _____________________________________________________

✓ To locate every donor that is under 40 years of age, has donated at least $500 in the past, and lives in either New York or California.

   The appropriate SQL query is expressed as:

   _____________________________________________________
   _____________________________________________________
   _____________________________________________________
Solution

• The lobbyist wants to specifically target only those donors who live in either the coastal states of New York or California regardless of their age or their past donations.

• The appropriate SQL query is expressed as:

   ```sql
   SELECT First, Last FROM Donor WHERE State = 'NY' OR State = 'CA'
   ```

   • A list of four donors: Helen Lobby, Reggie Green, Harvey Levirage, and Robin Round.

• To locate every donor that is under 40 years of age, has donated at least $500 in the past, and lives in either New York or California.

• The appropriate SQL query is expressed as:

   ```sql
   SELECT First, Last FROM Donor WHERE Age < 40 AND Amount >= 500 AND (State = 'NY' OR State = 'CA')
   ```

   • Only two such donors: Helen Lobby and Reggie Green.
Write SQL query for the given database

Consider the following database table. The database is used by a local club to track its members. The Paid column indicates whether a person has paid dues for this year, the Gender column has the entry ‘U’ for those members who have signed up but not yet provided their gender information.

Local club table

<table>
<thead>
<tr>
<th>First</th>
<th>Last</th>
<th>e-Mail</th>
<th>Paid</th>
<th>Gender</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jason</td>
<td>Lin</td>
<td><a href="mailto:jlin@ctmail.com">jlin@ctmail.com</a></td>
<td>Y</td>
<td>M</td>
<td>32</td>
</tr>
<tr>
<td>Hannah</td>
<td>Flynn</td>
<td><a href="mailto:fly@ctmail.edu">fly@ctmail.edu</a></td>
<td>N</td>
<td>F</td>
<td>19</td>
</tr>
<tr>
<td>Jordyn</td>
<td>Ash</td>
<td><a href="mailto:ash@ct.com">ash@ct.com</a></td>
<td>Y</td>
<td>U</td>
<td>59</td>
</tr>
<tr>
<td>Reginald</td>
<td>Holt</td>
<td><a href="mailto:reholt@ct.edu">reholt@ct.edu</a></td>
<td>Y</td>
<td>M</td>
<td>31</td>
</tr>
<tr>
<td>Sophia</td>
<td>Grace</td>
<td><a href="mailto:sophi@ct.edu">sophi@ct.edu</a></td>
<td>Y</td>
<td>F</td>
<td>29</td>
</tr>
<tr>
<td>Ella</td>
<td>Peters</td>
<td><a href="mailto:ellap@ct2.edu">ellap@ct2.edu</a></td>
<td>N</td>
<td>F</td>
<td>23</td>
</tr>
<tr>
<td>Dakota</td>
<td>Flynn</td>
<td><a href="mailto:dfly@ctmail.edu">dfly@ctmail.edu</a></td>
<td>N</td>
<td>U</td>
<td>19</td>
</tr>
<tr>
<td>Aaron</td>
<td>McGregor</td>
<td><a href="mailto:amg@ctmail.edu">amg@ctmail.edu</a></td>
<td>Y</td>
<td>M</td>
<td>59</td>
</tr>
</tbody>
</table>
✓ For each of the following criteria, write an SQL query to generate the desired result.

a. The first and last name of all female members.
b. The e-mail addresses of those members who have paid their dues.
c. The e-mail addresses of those members under 21 who have not paid their dues.
d. The first and last name of all who are either male or have an unknown gender.
e. The e-mail address of all dues-paying members who have an unknown gender.
What has been described?

How to apply simple propositional logic to

- Query search engines.
- Design digital circuits.
- Query search databases.

Credits
- http://www.facstaff.bucknell.edu/mastascu/lessonshtml/Logic/Logic1.html
- Discrete Structures ,CMSC 2123; cs2.uco.edu/~trt/cs2123/1002.pdf
- Google images