DISCRETE MATHEMATICS

Writing Assignment 1

Authors
Paul Beggs, Thomas Sebring

2nd Semester, 2024

1 Overview

Logical expressions, formed from atomic propositions (a basic, indivisible statement that expresses a single fact or condition and can be either true or false, but not both) and logical connectives ($\vee, \wedge,$ & $\neg$), represent statements that can be either true or false. Truth tables serve as a systematic method to explore all possible truth values of atomic propositions and their impact on the overall expression.

2 Logical Expressions Evaluated Through Truth Tables

2.1 Truth Table Analysis

Constructing and Analyzing Truth Tables

Constructing a truth table involves listing all possible combinations of truth values for the given atomic propositions and calculating the truth value of the compound expression under each combination. This method is crucial for understanding the behavior of logical expressions inferent scenarios.

Exploration of Satisfiability in Logical Expressions

Satisfiability refers to the condition in which a logical expression can be true for at least one combination of the truth values of its atomic propositions. Truth tables provide a clear framework to identify whether an expression is satisfiable by revealing if there exists at least one row where the expression evaluates to true.

For example, consider the logical expression $(\neg p \iff q) \wedge (p \wedge q)$. We can use truth tables to determine if this statement is satisfiable or not (refer to Table 1). As it turns out, this statement is *not* satisfiable, as there is no possible combination of truth values for which the entire statement is true.

p	q	$\neg p \iff q$	$p \wedge q$	$(\neg p \iff q) \wedge (p \wedge q)$
T	T	F	T	F
T	F	F	F	F
F	T	T	F	F
F	F	T	F	F

Table 1: Truth Table For $(\neg p \iff q) \wedge (p \wedge q)$

Now, consider the logical expression $(\neg(p \vee q) \Rightarrow r) \wedge (\neg r \wedge q)$. Using a truth table, it is clear that this expression is satisfiable, as there is *at least* one row that is true given the truth values for p, q, and r (see Table 2).

p	q	r	$\neg(p \vee q)$	$\neg(p \vee q) \Rightarrow r$	$\neg r \wedge q$	$(\neg(p \vee q) \Rightarrow r) \wedge (\neg r \wedge q)$
T	T	T	F	T	F	F
T	T	F	F	T	T	T
T	F	T	F	T	F	F
T	F	F	F	T	F	F
F	T	T	F	T	F	F
F	T	F	F	T	T	T
F	F	T	T	T	F	F
F	F	F	T	F	F	F

Table 2: Truth Table For $(\neg(p \vee q) \Rightarrow r) \wedge (\neg r \wedge q)$

Satisfiability with Truth Table Complexity

As you can see in Table 1 and Table 2, given more and more atomic propositions (p, q, r), the truth table becomes more complex with each addition. In fact, we can calculate the increase in rows using the formula 2^n (where n is the number of propositions that are present in the table).

Thus, given any set of atomic propositions $(p_1, p_2 \ldots p_n)$, we must examine the effect of each proposition on the whole expression to determine satisfiability. However, although we must check each proposition, determining satisfiability is not as cumbersome as it may seem. Recall that we only need to find *one* instance of truth among the propositions to verify a claim of satisfiability; conversely, if there are no instances of truth in the possible combinations, then the statement is false.

3 Conjunctive Normal Form (CNF)

3.1 CNF & Satisfiability

It turns out that one way to determine whether a particular expression is or is not satisfiable is to turn it into its CNF. An expression and the atomic propositions $p_1, p_2 \ldots, p_n$ are in CNF provided that it is a conjunction ("and" "$\wedge$") of dis-junctions ("or," "$\vee$"), where each atomic proposition can be listed as either itself or its negation ("$\neg$"). For example, each of the following is in CNF:

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

$$(p_1 \vee \neg p_3 \vee p_2) \wedge (p_1 \vee p_2) \wedge (p_4 \vee \neg p_5)$$

You may ask "Why would we want to put these expressions in CNF? Why go through the trouble of transforming expressions into an equivalent form of a conjunction of conjunctions?"

Transformationa statement into its CNF is particularly advantageous for computational processes. While it might not always enhance readability, CNF aligns with the binary logic that underpins computing systems, facilitating the use of efficient algorithms. The CNF structure, consisting of AND and OR operations, corresponds directly to the binary values that computers inherently process. This compatibility reduces the complexity of algorithmic decision-making, leading to a decrease in the computational resources required. As a result, CNF is invaluable for optimizing logic-based computations, which are integral to the vast array of technologies that define our modern environment.

Applications

1. Returning to the first expression in Section 2, we can use the rules of logical equivalences to turn,
$$(\neg p \iff q) \wedge (p \wedge q)$$
into its CNF for ease of determining its satisfiability.
   (a) To begin, we know that $\neg p \iff q$ can be expressed as $(\neg p \Rightarrow q) \wedge (q \Rightarrow \neg p)$, as these two statements adhere to the requirement of having the same truth value through the use of conditional statements. By using "and," we ensure that both conditions must be met simultaneously, which is the requirement imposed by the bi-conditional, $\neg p \iff q$.
   (b) Furthermore, by implication, $\neg p \Rightarrow q$ and $q \Rightarrow \neg p$ can be expressed as $\neg(\neg p) \vee q$ (which can be represented as $p \vee q$ using the rule of double negation) and $\neg q \vee \neg p$, respectively.
   (c) Integrating with the rest of the expression, we get $((p \vee q) \wedge (\neg q \vee \neg p)) \wedge (p) \wedge (q)$.
   (d) Thus, by putting the expression, $(\neg p \iff q) \wedge (p \wedge q)$, into a conjunction of disjunctions, we have shown that it is in its CNF by definition.

2. Similarly, we can do the same thing for the second expression in Section 2:

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

   (a) To put this expression into its CNF, we must transform the expression into a conjunction of dis-junctions. Thus, we must transform the implication $\neg(p \vee q) \Rightarrow r$ into a disjunctive.

   (b) Further, $\neg(p \vee q) \Rightarrow r$ can be put into form $\neg(\neg(p \vee q)) \vee r$ (because of the implication equivalence rule: $A \Rightarrow B \iff \neg A \vee B$).

   (c) Clearly, you can use double negation to remove the negation symbols, and you can use associativity to remove the parentheses to end up with $p \vee q \vee r$.

   (d) Integrating with the rest of the expression, we end up with $(p \vee q \vee r) \wedge (\neg r \wedge q)$, but can be further simplified to meet the requirements of CNF by changing $\neg r \wedge q$ into $(\neg r) \wedge (q)$, where $(p \vee q \vee r), (\neg r)$, and (q) are individual clauses connected by conjunctions.

Universal Application

Because of the imperative function of CNF, determining if this process is universally applicable to all logical expressions is inherently important.

CNF relies on fundamental properties of logical operations that are invariant to the specific content of the propositions involved, we can see that every step in the process is supported by an equivalence transformation that guarantees the logical equivalence of the initial and transformed expressions.

Simply put, this means we can use systematic conversion processes to convert *any* compound proposition into the desired form by employing any or all of the following:

1. **Eliminating Equivalence and Implications**:
   Replace bi-conditionals ($p \iff q$) and implications ($p \Rightarrow q$) with their equivalent expressions, $\wedge, \vee$, and $\neg$.

2. **Migrate Negations Inward**:
   Apply De Morgan's laws to move negations inward until they apply direction to atomic propositions only. In practice, this applies to expressions like $\neg(p \vee q)$ being expressed as $\neg p \vee \neg q$.

3. **Distribute $\wedge$ over $\vee$**:
   Use the distributive laws to ensure that the expression becomes a conjunction of disjunctions. This may involve expanding the expression significantly, especially when distributing a conjunction inside a disjunction.

4 Synthesis of Findings in Logical Analysis

Throughout this discussion, we have explored the practical utility of transforming logical expressions into *Conjunctive Normal Form* (CNF). While the readability of CNF may not always be straightforward, especially as the complexity of expressions grows, its true value lies in computational logic. CNF's alignment with the fundamental binary operations of digital computers streamlines the processing of logical expressions. This streamlined form is crucial for the efficient operation of algorithmic tools that underpin much of modern computing, from automated theorem proving to optimization in artificial intelligence and beyond.

By reducing logical expressions to a standard form, CNF facilitates a reduction in computational complexity and resource consumption. In summary, the adoption of CNF in computer science reflects a harmonization between the abstract world of logic and the binary reality of computer processing, enabling advanced computation and problem-solving in various domains that are foundational to technological advancement.