COMP1002 Fall 2017 midterm exam study sheet

- *Propositional statement*: expression that has a truth value (true/false). It is a *tautology* if it is always true, *contradiction* if always false.
- *Logic connectives*: negation ("not") $\neg p$, conjunction ("and") $p \wedge q$, disjunction ("or") $p \vee q$, implication $p \to q$ (equivalent to $\neg p \vee q$), biconditional $p \leftrightarrow q$ (equivalent to $(p \to q) \wedge (q \to p)$). The order of precedence: $\neg$ strongest, $\wedge$ next, $\vee$ next, $\to$ and $\leftrightarrow$ the same, weakest.
- If $p \to q$ is an implication, then $\neg q \to \neg p$ is its *contrapositive*, $q \to p$ a *converse* and $\neg p \to \neg q$ an *inverse*. An implication is equivalent to its contrapositive, but not to converse/inverse or their negations. A negation of an implication $p \to q$ is $p \wedge \neg q$ (it is not an implication itself!)
- A *truth assignment* is a string of values of variables to the formula, usually a row with values of first several columns in the truth table (number of columns = number of variables). A truth assignment is *satisfying* the formula if the value of the formula on these variables is T, otherwise the truth assignment is *falsifying*. A formula is *satisfiable* if it has a satisfying assignment, otherwise it is *unsatisfiable* (a contradiction). A truth assignment can be encoded by a formula that is a $\wedge$ of variables and their negations, with negated variables in places that have F (false) in the assignment, and non-negated that have T (true). For example, $x = T, y = F, z = F$ is encoded as $(x \wedge \neg y \wedge \neg z)$.It is an encoding in a sense that this formula is true only on this truth assignment and nowhere else.
- A *truth table* has a line for each possible values of propositional variables (2^k lines if there are k variables), and a column for each variable and subformula, up to the whole statement.Its cells contain T and F depending whether the (sub)formula is true for the corresponding scenarios.
- Finding a method for checking if a formula has a satisfying assignment that is always significantly faster than using truth tables (that is, better than brute-force search) is a one of Clay Mathematics Institute millenium prize problems , known as "P vs. NP".
- Two formulas are *logically equivalent*, written $A \equiv B$, if they have the same truth value in all scenarios (truth assignments). $A \equiv B$ if and only if $A \leftrightarrow B$ is a tautology.
- There are several other important pairs of logically equivalent formulas, called *logical identities* or *logic laws*. We will talk more about them when we talk about Boolean algebras. The most famous example of logically equivalent formulas is $\neg(p \vee q) \equiv (\neg p \wedge \neg q)$ (with a dual version $\neg(p \wedge q) \equiv (\neg p \vee \neg q)$) where p and q can be arbitrary (propositional, here) formulas. These pairs of logically equivalent formulas are called *DeMorgan's law*. Here, remember that $FALSE \wedge p \equiv p \wedge \neg p \equiv FALSE$, $FALSE \vee p \equiv TRUE \wedge p \equiv p$ and $TRUE \vee p \equiv p \vee \neg p \equiv TRUE$.
- A set of logic connectives is called *complete* if it is possible to make a formula with any truth table out of these connectives. For example, $\neg, \wedge$ is a complete set of connectives, and so is the Sheffer's stroke $|$ (where $p|q \equiv \neg(p \wedge q)$), also called NAND for "not-and". But $\vee, \wedge$ is not a complete set of connectives since then it is impossible to express a truth table with 0 when all variables are 1.
- An *argument* consists of several formulas called *premises* and a final formula called a *conclusion*. If we call premises $A_1 \dots A_n$ and conclusion B, then an argument is *valid* iff premises imply the conclusion, that is, $A_1 \wedge \cdots \wedge A_n \to B$. We usually write them in the following format:

Today is either Thursday or Friday
On Thursdays I have to go to a lecture
Today is not Friday (alternatively, On Friday I have to go to the lecture)

$\therefore$ I have to go to a lecture today

- A valid form of argument is called *rule of inference*. The most prominent such rule is called *modus ponens*.

  $p \to q$
  p ————
  $\therefore q$

- We studied three methods for proving that a formula is a tautology: *truth tables*, *natural deduction* and *resolution* (where resolution proves that a formula is a tautology by proving that its negation is a contradiction).

- A *natural deduction proof* consists of a sequence of applications of modus ponens (and other rules of inference) until a desired conclusion is reached, or there is nothing new left to derive. Example: treasure hunt, where "desired conclusion" is a statement that the treasure is in a specific location.

- There are two main normal forms for the propositional formulas. One is called *Conjunctive normal form* (CNF, also known as Product-of-Sums) and is an $\wedge$ of $\vee$ of either variables or their negations (here, by $\wedge$ and $\vee$ we mean several formulas with $\wedge$ between each pair, as in $(\neg x \vee y \vee z) \wedge (\neg u \vee y) \wedge x$. A *literal* is a variable or its negation (x or $\neg x$, for example). A $\vee$ of (possibly more than 2) literals is called a *clause*, for example $(\neg u \vee z \vee x)$, so a CNF is true for some truth assignment whenever this assignment makes each of the clauses is true, that is, each clause has a literal that evaluates to true under this assignment. A *Disjunctive normal form* (DNF, Sum-of-Products) is like CNF except the roles of $\wedge$ and $\vee$ are reversed. A $\wedge$ of literals in a DNF is called a *term*. To construct canonical DNF and a CNF, start from a truth table and then for every satisfying truth assignment $\vee$ its encoding to a DNF, and for every falsifying truth assignment $\wedge$ the negation of its encoding to the CNF, and apply DeMorgan's law. This may result in a very large CNFs and DNFs, comparable to the size of the truth table itself ($2^{\text{number of variables}}$).

- A *resolution proof system* is used to find a contradiction in a formula (and, similarly, to prove that a formula is a tautology by finding a contradiction in its negation). Resolution starts with a formula in a CNF form, and applies the rule "from clause $(C \vee x)$ and clause $(D \vee \neg x)$ derive clause $(C \vee D)$ until a falsity F (equivalently, empty clause ()) is reached (so in the last step one of the clauses being *resolved* contains just one variable and another clause being resolved contains just that variable's negation.) Note that if a clause has opposing literals (e.g., from resolving $(x \vee y)$ with $(\neg x \vee \neg y)$ then it evaluates to true, and so is useless for deriving a contradiction. Resolution can be used to check the validity of an argument by running it on the $\wedge$ of all premises (converted, each, to a CNF) $\wedge$ together with the negation of the conclusion.

- *Pigeonhole principle* If n pigeons sit in $n-1$ holes, so that each pigeon sits in some hole, then some hole has at least two pigeons. There is no small resolution proof of the pigeonhole principle.

- *Boolean functions* are functions which take as argument boolean (ie, propositional) variables and return 1 or 0 (or, the convention here is 1 instead of T, and 0 instead of F). Each Boolean function on n variables can be fully described by its truth table. A size of a truth table of a function on n variables is 2^n. Even though we often can have a smaller description of a function, vast majority of Boolean functions cannot be described by anything much smaller. Every Boolean function can be described by a CNF or DNF, using the above construction.

Predicate logic:

- A *predicate* is like a propositional variable, but with *free variables*, and can be true or false depending on the values of these free variables. A *domain* of a predicate is a set from which the free variables can take their values (e.g., the domain of $Even(n)$ can be integers).

- *Quantifiers* For a predicate $P(x)$, a quantified statement "for all" ("every", "all") $\forall x P(x)$ is true iff $P(x)$ is true for every value of x from the domain (also called universe); here, $\forall$ is called a *universal quantifier*. A statement "exists" ("some", "a") $\exists x P(x)$ is true whenever $P(x)$ is true for at least one element x in the universe; $\exists$ is an existential quantifier. The word "any" means sometimes $\exists$ and sometimes $\forall$. A domain (universe) of a quantifier, sometimes written as $\exists x \in D$ and $\forall x \in D$ is the set of values from which the possible choices for x are made. If the domain of a quantifier is empty, then if the quantifier is universal then the formula is true, and if quantifier is existential, false. A *scope* of a quantifier is a part of the formula (akin to a piece of code) on which the variable under that quantifier can be used (after the quantifier symbol/inside the parentheses/until there is another quantifier over a variable with the same name). A variable is *bound* if it is under a some quantifier symbol, otherwise it is free.

- *First-order formula* A predicate is a first-order formula (possibly with free variables). A $\wedge, \vee, \neg$ of first-order formulas is a first-order formula. If a formula $A(x)$ has a free variable (that is, a variable x that occurs in some predicates but does not occur under quantifiers such as $\forall x$ or $\exists x$), then $\forall x\ A(x)$ and $\exists x\ A(x)$ are also first-order formulas. A first-order formula is in *prenex form* when all variables have different names and all quantifiers are in front of the formula.

- An *interpretation* is an assignment of specific values to domains and predicates. A *model* of a formula is an interpretation that makes this formula true. Example: in Tarski world interpretations, the domain is all possible pieces, and interpretation of Square assigns Square(x) to true iff x is a square piece, Blue(x) true to blue pieces, etc. A board which satisfies a given formula is a model of that formula.

- *Negating quantifiers.* Remember that $\neg \forall x P(x) \equiv \exists x \neg P(x)$ and $\neg \exists x P(x) \equiv \forall x \neg P(x)$. This is because $\forall$ is like a big $\wedge$ over all scenarios, and $\exists$ is an $\vee$.