Roll No.: ______________________

CS 208 : Automata Theory and Logic Spring 2024

Quiz 1

Time: 21:00 - 23:30 *Max Marks:* 60

Instructions:

- ***Please write your roll number on all pages in the space provided at the top.***
- *Be brief, complete, and stick to what has been asked.*
- ***You must write your answer for every question only in the space allocated for answering the question. Answers written outside the allocated space risk not being graded.***
- ***You can use an extra answer book for rough calculations.***
- ***You must submit this question+answer book in its entirey along with any extra answer book for rough calculations (if you used one).***
- *Untidy presentation of answers, and random ramblings will be penalized by negative marks.*
- *Unless asked for explicitly, you may cite results/proofs covered in class without reproducing them.*
- ***If you need to make any assumptions, state them clearly.***
- ***Do not copy solutions from others. All detected cases of copying will be reported to DADAC with names and roll nos. of all involved. The stakes are high if you get reported to DADAC, so you are strongly advised not to risk this.***

1. **We love NNFs!** [10]

   Let P, Q, R be propositional variables.

   (a) Convert the formula $\big((P \rightarrow (Q \rightarrow R)) \rightarrow \neg(P \rightarrow (R \rightarrow Q))\big)$ to a semantically equivalent formula in Disjunctive Normal Form (DNF). Do not include cubes that contain both a literal and its negation in your DNF formula. [5]
   You must show all intermediate steps. Answers without steps will fetch no marks.

   Solution:

   $$\begin{array}{lll} & \big((P \rightarrow (Q \rightarrow R)) \rightarrow \neg(P \rightarrow (R \rightarrow Q))\big) & \\ \Leftrightarrow & \neg(P \rightarrow (Q \rightarrow R)) \vee \neg(P \rightarrow (R \rightarrow Q)) & \text{... Semantics of } \rightarrow \\ \Leftrightarrow & \neg(\neg P \vee (\neg Q \vee R)) \vee \neg(\neg P \vee (\neg R \vee Q)) & \text{... Semantics of } \rightarrow \\ \Leftrightarrow & (P \wedge \neg(\neg Q \vee R)) \vee (P \wedge \neg(\neg R \vee Q)) & \text{... DeMorgan's Law and } \neg\neg \text{ elim} \\ \Leftrightarrow & (P \wedge (Q \wedge \neg R)) \vee (P \wedge (R \wedge \neg Q)) & \text{... DeMorgan's Law and } \neg\neg \text{ elim} \\ \Leftrightarrow & (P \wedge Q \wedge \neg R) \vee (P \wedge R \wedge \neg Q) & \text{... Simplify (using associativity)} \end{array}$$

   (b) Convert the formula $\big(\neg(P \vee (\neg Q \wedge R)) \rightarrow (\neg P \wedge (Q \vee \neg R))\big)$ to an equisatisfiable Conjunctive Normal Form (CNF) formula using Tseitin encoding. Do not include clauses that contain both a literal and its negation in your CNF formula. [5]
   You must NOT simplify the given formula or check its satisfiability before applying Tseitin encoding. You must show all intermediate steps. Answers

without steps or obtained after simplifying the given formula or after checking its satisfiability will fetch no marks. Answers that give an equisatisfiable formula without using Tseitin encoding will also fetch no marks.

Solution: We first introduce a fresh variable t_i for each sub-formula that is neither a variable nor its negation.

$$
\begin{array}{lllr}
(t_1 & \leftrightarrow & (\neg Q \land R)) & \land \\
(t_2 & \leftrightarrow & (P \lor t_1)) & \land \\
(t_3 & \leftrightarrow & \neg t_2) & \land \\
(t_4 & \leftrightarrow & (Q \lor \neg R)) & \land \\
(t_5 & \leftrightarrow & (\neg P \land t_4)) & \land \\
(t_6 & \leftrightarrow & (t_3 \rightarrow t_5)) & \land \\
t_6 & & &
\end{array}
$$

Next, we expand each bi-implication into a conjunction of two implications.

$$
\begin{array}{lllclllc}
(t_1 & \rightarrow & (\neg Q \land R)) & \land & (t_1 & \leftarrow & (\neg Q \land R)) & \land \\
(t_2 & \rightarrow & (P \lor t_1)) & \land & (t_2 & \leftarrow & (P \lor t_1)) & \land \\
(t_3 & \rightarrow & \neg t_2) & \land & (t_3 & \leftarrow & \neg t_2) & \land \\
(t_4 & \rightarrow & (Q \lor \neg R)) & \land & (t_4 & \leftarrow & (Q \lor \neg R)) & \land \\
(t_5 & \rightarrow & (\neg P \land t_4)) & \land & (t_5 & \leftarrow & (\neg P \land t_4)) & \land \\
(t_6 & \rightarrow & (t_3 \rightarrow t_5)) & \land & (t_6 & \leftarrow & (t_3 \rightarrow t_5)) & \land \\
t_6 & & & & & & &
\end{array}
$$

Next, we use the semantics of $\rightarrow$ (or $\leftarrow$) to get

$$
\begin{array}{ccccc}
(\neg t_1 \lor (\neg Q \land R)) & \land & (t_1 \lor \neg(\neg Q \land R)) & \land \\
(\neg t_2 \lor (P \lor t_1)) & \land & (t_2 \lor \neg(P \lor t_1)) & \land \\
(\neg t_3 \lor \neg t_2) & \land & (t_3 \lor \neg\neg t_2) & \land \\
(\neg t_4 \lor (Q \lor \neg R)) & \land & (t_4 \lor \neg(Q \lor \neg R)) & \land \\
(\neg t_5 \lor (\neg P \land t_4)) & \land & (t_5 \lor \neg(\neg P \land t_4)) & \land \\
(\neg t_6 \lor (\neg t_3 \lor t_5)) & \land & (t_6 \lor \neg(\neg t_3 \lor t_5)) & \land \\
t_6 & & & &
\end{array}
$$

Next, we use DeMorgan's laws and $\neg\neg$ elimination to get

$$
\begin{array}{ccccc}
(\neg t_1 \lor (\neg Q \land R)) & \land & (t_1 \lor (Q \lor \neg R)) & \land \\
(\neg t_2 \lor (P \lor t_1)) & \land & (t_2 \lor (\neg P \land \neg t_1)) & \land \\
(\neg t_3 \lor \neg t_2) & \land & (t_3 \lor t_2) & \land \\
(\neg t_4 \lor (Q \lor \neg R)) & \land & (t_4 \lor (\neg Q \land R)) & \land \\
(\neg t_5 \lor (\neg P \land t_4)) & \land & (t_5 \lor (P \lor \neg t_4)) & \land \\
(\neg t_6 \lor (\neg t_3 \lor t_5)) & \land & (t_6 \lor (t_3 \land \neg t_5)) & \land \\
t_6 & & & &
\end{array}
$$

Finally, using distributivity of $\land$ over $\lor$ and vice versa, we get the desired Tseitin encoding

$$
\begin{array}{cc}
(\neg t_1 \lor \neg Q) \ \land \ (\neg t_1 \lor R) \ \land \ (t_1 \lor Q \lor \neg R) & \land \\
(\neg t_2 \lor P \lor t_1) \ \land \ (t_2 \lor \neg P) \ \land (t_2 \lor \neg t_1) & \land \\
(\neg t_3 \lor \neg t_2) \ \land \ (t_3 \lor t_2) & \land \\
(\neg t_4 \lor Q \lor \neg R) \ \land \ (t_4 \lor \neg Q) \ \land \ (t_4 \lor R) & \land \\
(\neg t_5 \lor \neg P) \ \land \ (\neg t_5 \lor t_4) \ \land \ (t_5 \lor P \lor \neg t_4) & \land \\
(\neg t_6 \lor \neg t_3 \lor t_5)) \ \land \ (t_6 \lor t_3) \ \land \ (t_6 \lor \neg t_5) & \land \\
t_6 &
\end{array}
$$

2. **How expressive are you?** [15]

We know that propositional logic formulas are constructed using symbols in the set $\{\top, \bot, \neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ in addition to variables, parentheses and commas (if needed). It turns out that such a large set of symbols may not be needed. For example, the set $S' = \{\wedge, \neg\}$ suffices to construct a formula that is semantically equivalent to any propositional logic formula. Indeed, from DeMorgan's laws we know that $p_1 \vee p_2$ is semantically equivalent to $\neg(\neg p_1 \wedge \neg p_2)$ for every propositional variable (or sub-formula) p_1 and p_2.

Let S be a set of symbols that are used in addition to variables, parentheses and commas (if needed) to construct formulas. We say that S is ***propositionally expressive*** if for every propositional logic formula φ over variables $x_1, \ldots x_n$, there is a semantically equivalent formula over $x_1, \ldots x_n$ that uses only symbols from S in addition to variables, parentheses and commas (if needed). From what we have studied in class, it should be easy for you to see that $\{\wedge, \neg\}$ is propositionally expressive.

For purposes of this question, we define new ternary logic connectives α, β, γ and δ with the following semantics.

- $\alpha(p, q, r)$ evaluates to 1 (true) iff either both p and q are 1 (true) or p is 0 (false) and r is 1 (true). For example, $\alpha(1, 1, 0) = \alpha(0, 0, 1) = 1$ but $\alpha(1, 0, 1) = \alpha(0, 1, 0) = 0$. You can think of $\alpha(p, q, r)$ as intuitively implementing "if p then q else r".
- $\beta(p, q, r)$ evaluates to 1 (true) iff p, q and r all have the same truth value. Thus, $\beta(1, 1, 1) = \beta(0, 0, 0) = 1$, but $\beta(1, 0, 1) = \beta(1, 1, 0) = 0$. You can think of $\beta(p, q, r)$ as intuitively implementing "all of p, q, r are in consensus".
- $\gamma(p, q, r)$ evaluates to 1 (true) iff exactly one of p, q and r has the value 1 (true). Thus, $\gamma(1, 0, 0) = 1$ and $\gamma(1, 1, 0) = \gamma(0, 0, 0) = 0$.
- $\delta(p, q, r)$ evaluates to the same truth value as the majority of p, q and r. Thus, $\delta(1, 1, 0) = 1$ and $\delta(0, 1, 0) = 0$.

Given below are three sets S of symbols used to construct formulas. In each case you must indicate whether S is propositionally expressive, with justification.

You may use the fact that $\{\wedge, \neg\}$ is known to be propositionally expressive. Hence, for "Yes" answers, you need to show that $\neg p$ and $p \wedge q$ can be equivalently expressed using the symbols in S, for any propositional variables (or sub-formulas) p and q. For "No" answers, you must show that there is at least one formula that can be written using $\{\wedge, \neg\}$ but a semantically equivalent formula cannot be written using S.

(a) $S = \{\top, \bot, \alpha\}$ [4]

Solution: To show that S is propositionally expressive, one has to find a semantically equivalent formula constructed using S for every formula constructed using $\{\neg, \wedge\}$. But ***if we can find formulas constructed using S that are equivalent to $\neg p$ and $p \wedge q$***, we can find a formula equivalent to any formula constructed using $\{\neg, \wedge\}$! Why? Simple structural induction : take any formula F constructed using $\{\neg, \wedge\}$. F is either a variable or $\neg G$ or $G_1 \wedge G_2$, where G, G_1, G_2 are formulas constructed using $\{\neg, \wedge\}$.

- Base case: p, $\neg p$ and $p \wedge q$ have semantically equivalent formulas constructed using S.
- Assume: Every sub-formula of F has a semantically equivalent formula constructed using S.

- Induction: Suppose F is $\neg G$ (resp. $G_1 \wedge G_2$). Take the formula for $\neg p$ (resp. $q \wedge r$) constructed using S, and replace p (resp. q, r) with G (resp. G_1, G_2). We have a formula for F constructed using S!

Using the above argument (**this is not necessary to be shown as part of your solution**), we can easily conclude that $S = \{\top, \bot, \alpha\}$ is propositionally expressive. Indeed, $\neg p \Leftrightarrow \alpha(p, \bot, \top)$ and $(p \wedge q) \Leftrightarrow \alpha(p, q, \bot)$.

(b) $S = \{\top, \gamma\}$ 4

Solution: $\neg p \Leftrightarrow \gamma(p, p, \top)$ and $(\neg p \wedge q) \Leftrightarrow \gamma(p, p, q)$. Hence $(p \wedge q) \Leftrightarrow \gamma(\neg p, \neg p, q)$. Using the equivalence for $\neg p$ already obtained above, we get

$$(p \wedge q) \Leftrightarrow \gamma(\gamma(p, p, \top), \gamma(p, p, \top), q)$$

(c) $S = \{\beta, \delta\}$ 7

Solution: S is not propositionally expressive. There is a very simple formula that cannot be expressed using S : $\varphi(p) = p \wedge \neg p = \bot$

Consider a formula B constructed using $S = \{\beta, \delta\}$ that has only one variable p. Notice that every internal node of the parse tree of B has 3 children and every leaf node of the parse tree must be p. Furthermore, every leaf node has a parent labeled either β or δ. Given that the parse tree of a formula constructed using S necessarily has finite height, there must be some leaf node with both its siblings also as leaf nodes. For such a leaf node, if its parent is β, then we have $\beta(p, p, p) \equiv \top$. On the other hand, if its parent is δ, we have $\delta(p, p, p) \equiv p$. In order to understand what propositional formula the parse tree represents, We can prune such a leaf node along with its parent and replace it with p or $\top$ accordingly. Notice that this introduces $\top$ as a symbol at a leaf of the modified parse tree, but this doesn't change the semantics of the formula represented by the parse tree.

After the above pruning step, again consider the leaf nodes whose siblings are also leaf nodes. Unlike in the previous case, a leaf node can now be either p or $\top$. However, its parent must still be labeled by either β or γ. Furthermore, we know that $\beta(p, p, \top) \equiv \beta(p, \top, p) \equiv \beta(\top, p, p) \equiv \beta(p, \top, \top) \equiv \beta(\top, p, \top) \equiv \beta(\top, \top, p) \equiv \delta(p, p, \top) \equiv \delta(p, \top, p) \equiv \delta(\top, p, p) \equiv p$ and $\delta(p, \top, \top) \equiv \delta(\top, p, \top) \equiv \delta(\top, \top, p) \equiv \delta(\top, \top, \top) \equiv \top$. So if we again prune the leaf, its parent and siblings, we will either replace it with p or $\top$. We can now inductively argue that by continuing this process, the entire parse tree will finally be replaced by p or $\top$. Therefore, the parse tree cannot represent a formula that is semantically equivalent to $\varphi(p) = p \wedge \neg p = \bot$.

Notice that by the same argument, we can't construct a formula semantically equivalent to $\neg p$ either using S.

However, if we expand S to be $\{\bot, \beta, \gamma\}$, we can easily express $\neg p$ as $\beta(p, p, \bot)$ and $p \wedge q$ as $\beta(p, q, \beta(p, p, p))$, where $\beta(p, p, p)$ is equivalent to $\top$. In fact, we don't need γ to be in S at all if $\bot$ and β are in S!

3. **Shipping with SAT** [10]

A shipping company has n cargo containers that must be transported via ships. Let $C = \{c_1, \ldots, c_n\}$ denote the containers. The company has m ships; let $S = \{s_1, \ldots s_m\}$ denote these ships. It turns out that not every ship can transport every container. Let $A_i \subseteq C$ be the set of containers that ***are allowed*** to be transported on ship i. Furthermore, each ship s_i has a ***maximum limit*** of l_i containers that it can transport. All l_i's are assumed to be non-negative integers.

The shipping company wants to find a set X of at most k $(0 < k \leq m)$ ships that can be used to transport all n containers, while loading each ship only with containers that are allowed on the ship, and without overloading each ship beyond the maximum number of containers it can transport.

As an example, consider $n = 4$, $m = 5$ and $l_1 = l_2 = l_4 = 1, l_3 = l_5 = 2$. Furthermore, suppose $A_1 = \{c_1, c_2\}$, $A_2 = \{c_2, c_3, c_4\}$, $A_3 = \{c_1, c_2, c_4\}$, $A_4 = \{c_2\}$ and $A_5 = \{c_2, c_4\}$. In this example, it is impossible to transport all 4 containers on only 2 ships. However, it is possible to transport all of them on 3 ships. For example, s_1 can be used to transport c_1, s_2 can be used to transport c_3 and s_5 can be used to transport c_2 and c_4. Hence $X = \{s_1, s_2, s_5\}$ is one possible solution the shipping company seeks.

We wish to use a satisfiability checker for NNF formulas to help the shipping company. Specifically, you must construct a propositional NNF formula φ such that

- Given $n, m, k,\ l_1, l_2, \ldots l_m$ where $0 \leq l_j \leq n$ for each $j \in \{1, \ldots m\}$, and the sets $A_1, A_2, \ldots A_m$, the formula φ can be constructed in time polynomial in m, n and k.
- There is a bijection between satisfying assignments of φ and distinct choices X of at most k ships that can transport all n containers, while respecting each ship's constraints. Note that this means φ must be unsatisfiable if it is impossible to transport all containers in at most k ships.

To construct the above formula, we will use propositional variables x_i for each $i \in \{1, \ldots m\}$, such that that x_i is true iff ship i is included in the set X of chosen ships.

You are free to use auxiliary propositional variables as you consider necessary. However, you must indicate the interpretation (what does the variable represent) for each such auxiliary variables. You are also free to use the cardinality constraints of the form $\sum_{p=u}^{v} b_p \leq w$ for propositional variables $b_u, \ldots b_v$ where $u \leq v$ and $0 \leq w \leq (v - u + 1)$. We have already discussed in Tutorial 1 how such a cardinality constraint can be encoded as a NNF formula in time polynomial in $(v - u)$ and w, possibly with the use of auxiliary propositional variables. Hence, if you are using such cardinality constraints, you don't need to explicitly write the NNF formula corresponding to $\sum_{p=u}^{v} b_p \leq w$, but can simply use $\left(\sum_{p=u}^{v} b_p \leq w\right)$ as a proxy for the NNF formula.

Solution: Let the propositional variable x_i $(1 \leq i \leq m)$ encode "ship $s_i \in X$", as required by the question. Furthermore, let propositional variable $t_{i,j}$ $(1 \leq i \leq m, 1 \leq j \leq n)$ encode "container c_j is loaded in ship s_i".

The overall NNF formula is obtained as $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$, where

- φ_1 is $\left(\sum_{i=1}^{m} x_i \leq k\right)$. This encodes that at most k ships are chosen to be in the solution set X.
- φ_2 is $\bigwedge_{j=1}^{n} \bigwedge_{i\,:\,c_j \notin A_i} \neg t_{ij}$. This encodes that for each container c_j and for each ship s_i such that c_j is not allowed on s_i, the variable t_{ij} must be false.

- φ_3 is $\bigwedge_{j=1}^{n} \left(\bigvee_{i \,:\, c_j \in A_i} \left(x_i \wedge t_{ij} \wedge \bigwedge_{k \,:\, c_j \in A_k, k \neq i} \neg t_{kj} \right) \right)$. This encodes that for each container c_j, there is exactly one ship s_i such that $s_i \in X$ and $c_j \in A_i$ for which t_{ij} is true. All other ships s_k such that $c_j \in A_k$ must have t_{kj} false. In other words, every container must be in exactly one ship in X that is allowed to transport the container.
- φ_4 is $\bigwedge_{i=1}^{m} \left(x_i \to \left(\sum_{j=1}^{n} t_{ij} \leq l_i \right) \right)$. This encodes that for each ship s_i, if it is chosen to be in X, the total count of containers loaded on s_i can be no more than l_i.

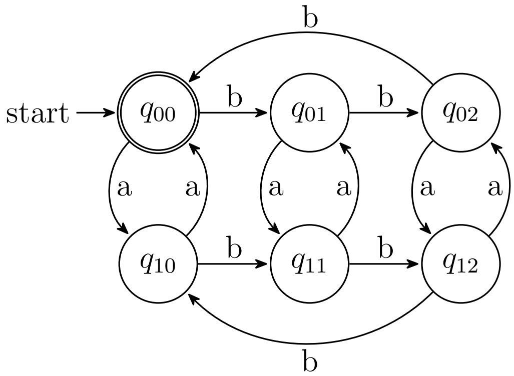
There are some variations of the above formulation that also serve the purpose of the question. Think about if a satisfying assignment of $\varphi_1 \wedge \varphi_3 \wedge \varphi_4$ would also serve the purpose of the shipping company. What would be the interpretation of t_{ij} in this case?

4. **I think I saw you in Tuesday's lecture** $\boxed{10}$

Draw a Deterministic Finite Automaton (DFA) for each of the following languages.

(a) $\mathcal{L} := \{w \in \{a,b\}^* : 2 \text{ divides } n_a(w) \text{ and } 3 \text{ divides } n_b(w)\}$. Here $n_a(w)$ stands for the number of a's in w, and $n_b(w)$ stands for the number of b's in w. For example, $n_a(abbaab) = n_b(abbaab) = 3$. Therefore, $ababbaa \in \mathcal{L}$ but $aabababa \notin \mathcal{L}$. $\boxed{5}$

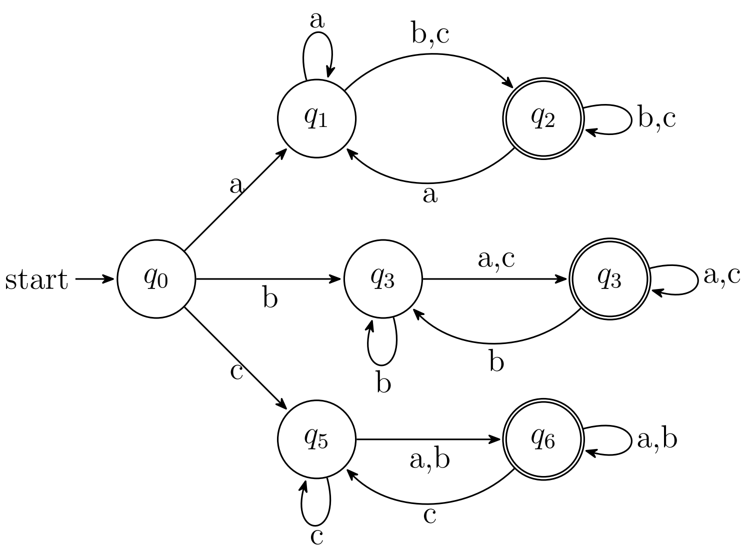
Solution:



State q_{ij} represents "Word w seen so far has $n_a(w) \mod 2 = i$ and $n_b(w) \mod 3 = j$.

(b) $\mathcal{L} := \{w \in \{a,b,c\}^* :$ first and last letters of w are different$\}$. For example, $abbacb \in \mathcal{L}$ but $cbbabac \notin \mathcal{L}$. $\boxed{5}$

Solution:



State	Interpretation
q_0	Haven't seen any letter
q_1	Start and end letters a
q_2	Start letter a, end letter b or c
q_3	Start and end letters b
q_4	Start letter b, end letter a or c
q_5	Start and end letters c
q_5	Start letter c, end letter a or b

5. **To CNF or to DNF** Define the *length* of a CNF (or DNF) formula as the total number of all literals over all clauses (or all cubes, respectively) in the formula. For example, consider the CNF formula $\phi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee x_3 \vee x_4)$. This formula has length $2 + 2 + 3 = 7$. Siimilarly, the length of the DNF formula $\psi = (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3 \wedge \neg x_4) \vee (\neg x_2 \wedge x_3 \wedge x_4)$ is $2 + 3 + 3 = 8$. [15]

   Show that there is a family of formulas $\mathcal{F} = \{\varphi_n \mid n \in \mathbb{N}\}$ such that

   - Every φ_n is a DNF formula over $\mathcal{O}(n)$ propositional variables.
   - Every φ_n has length in $\mathcal{O}(n)$.
   - For every φ_n, there exists **no semantically equivalent** CNF formula ψ_n (over the same variables as φ_n) such that the length of ψ_n grows polynomially with n.

   In order to answer this question, you must (a) clearly write the DNF formula φ_n, (b) show that its length is in $\mathcal{O}(n)$, and (c) prove that there exists **no semantically equivalent** CNF formula of length polynomial in n.

   In order to score marks in this question, all three parts must be answered correctly.

   Note: A formula has polynomial length if and only if *length* $\in \mathcal{O}(f(n))$ where f is some polynomial in n. You **MUST** prove why φ_n can't be equivalently represented by any polynomial length CNF formula.

Solution:

Note: This is not the only possible solution. There are alternative solutions as well.

Consider the family $\mathcal{F} = \{\varphi_n \mid n \in \mathbb{N}\}$, where

$$\varphi_n = (X_1 \wedge Y_1) \vee (X_2 \wedge Y_2) \vee \cdots \vee (X_n \wedge Y_n)$$

Clearly φ_n is in DNF and has $2n$ variables. Furthermore, its length is $2n$. We now show below that there exists no semantically equivalent CNF formula ψ_n such that the length of ψ_n grows polynomially in n.

Claim 1: Each clause of the CNF formula ψ_n must contain either X_i or Y_i as literals, for every $i \in \{1, \ldots n\}$.

Suppose, if possible, there is a clause C_j that does contains neither X_i nor Y_i as literal. Consider the assignment where X_i and Y_i have value 1, and all literals in C_j have value 0. Under this assignment, φ_n evaluates to 1 (recall φ_n has a cube $X_i \wedge Y_i$), but ψ_n evaluates to 0 as C_j evaluates to 0. Therefore, φ_n is not semantically equivalent to ψ_n – a contradiction! Hence, every clause C_j must have either X_i or Y_i as literal, for every $i \in \{1, \ldots n\}$.

Given Claim 1, the clauses in ψ_n can be divided into two categories: (a) those that have both X_i and Y_i as literals for some $i \in \{1, \ldots n\}$, and (b) those that have either X_i or Y_i, but not both, as literals for every $i \in \{1, \ldots n\}$.

Let us focus on clauses of type (b).

Claim 2: For every tuple of literals in $\{X_1, Y_1\} \times \{X_2, Y_2\} \times \cdots \times \{X_n, Y_n\}$, there is a clause of type (b) in ψ_n that contains the literals in the tuple.

Suppose, if possible, there is a tuple of literals in the Cartesian product such that there is no clause of type (b) in ψ_n that contains the literals in this tuple. Now consider the assignment that sets all variables in this specific tuple to 1 and all other variables to 0. Thus, this assignment sets exactly one of $\{X_i, Y_i\}$ to 0 and exactly one of them to 1, for every $i \in \{1, \dots n\}$. Since by Claim 1, every clause in ψ_n has either X_i or Y_i as a literal, for every $i \in \{1, \dots n\}$, it follows that every clause in ψ_n, and hence ψ_n itself, evaluates to 1 under this assignment. However, clearly φ_n evaluates to 0 under this assignment, since for every $i \in \{1, \dots n\}$, at least one of X_i and Y_i is 0. Hence, φ_n is not semantically equivalent to ψ_n – a contradiction! It follows that for every tuple of literals in the Cartesian product, the corresponding literals must be present in a type (b) clause of ψ_n.

Since there are 2^n distinct tuples in $\{X_1, Y_1\} \times \cdots \times \{X_n, Y_n\}$, Claim 2 shows that there are at least 2^n clauses of type (b) in ψ_n. Hence the length of ψ_n is at least 2^n.