CS 228: Logic in CS Autumn 2024

Mid Semester Examination

Total Marks: 60 marks *19 September 2024*

Instructions.

- Write your ROLL NUMBER on your answer sheet
- Unless asked for explicitly, you may cite results/proofs covered in class without reproducing them.
- When asked to prove something, give a formal proof with suitable explanation. Similarly, if asked to disprove give a counter-example and argue why it is in-fact a counter-example.
- If you need to make any assumptions, state them clearly.
- **DO NOT COPY SOLUTIONS FROM OTHERS**

---

1. [**10 marks**] State TRUE or FALSE for each of the below. **1 mark for correct answer, -1 for incorrect answer, 0 marks if not attempted**. However, the negative marks won't overflow to other questions. So the minimum marks that you can score in this question is 0. **No justifications are expected**.

   Make sure that you write TRUE *or* FALSE *legibly. In case of any ambiguous answers, your answer will be treated as incorrect.*

   (a) Let φ be any FOL sentence and $\mathsf{free}(\varphi)$ and $\mathsf{bound}(\varphi)$ be the set of free and bound variables in φ, respectively. Then $\mathsf{free}(\varphi) \cap \mathsf{bound}(\varphi)$ is always empty.

   (b) In a proof system where $p \wedge \neg p$ is an axiom, the system is complete but not sound.

   (c) Let C_1 and C_2 be any two clauses. Then, C_1 and C_2 have a unique resolvent.

   (d) Tseitin encoding preserves the validity of the input CNF.

   (e) The FOL formula $\forall x\ (P(x) \vee Q(x))$ is true if $P(x)$ is true for all x.

   (f) Over the usual signature $\tau = (S, <, Q_a)$ for finite words discussed in class, consider the FOL formula
   $$\varphi = \forall x \exists y\, (x < y \wedge Q_a(x) \wedge Q_a(y))$$
   $L(\varphi)$ is the set of all words w such that for every position x in w holding an a, there is a later position y where a holds.

   (g) If a FOL formula F in Skolem Normal Form is valid, there is a finite witness of its validity. But if it is not valid but satisfiable, there may not be a finite witness.

   (h) Every DFA has at least one accepting state.

(i) Let A be a DFA accepting a language L, having only one final state. Consider the operation called "reversal of A" defined as follows, which gives a new automaton $\mathsf{rev}(A)$. To obtain $\mathsf{rev}(A)$ from A, (i) each transition (q, a, q') in A is reversed to (q', a, q), (ii) the initial state of A is made a final state, and (iii) the final state of A is made the initial state. Then $\mathsf{rev}(A)$ is a DFA which accepts the reverses of all words in $L(A)$.

(*The reverse of a word w is the word w' obtained by reading it back to front, for example, if $w = aab$, then its reverse is baa.*)

(j) Given a fixed natural number n, one can write a FOL formula that will evaluate to false under a structure whose universe has exactly n elements.

Solution

(a) TRUE. By definition, a sentence does not have any free variables. Thus, $\mathsf{free}(\varphi) = \emptyset \implies \mathsf{free}(\varphi) \cap \mathsf{bound}(\varphi) = \emptyset$

(b) TRUE. If $p \wedge \neg p$ is an axiom, we could derive any formula using bot elimination. So the system is complete. But it also allows us to derive statements which are false, hence not sound.

(c) FALSE. Consider $C_1 = p_1 \vee \neg p_2$ and $C_2 = \neg p_1 \vee p_2$.

(d) FALSE. Let $\varphi = p \vee \neg p$.Clearly φ is valid. After Tseitin encoding we get, $\varphi_t = (x \rightarrow (p \vee \neg p)) \wedge ((p \vee \neg p) \rightarrow x)$, which is not valid.

(e) TRUE. If $P(x)$ is true for all x, $P(x) \vee Q(x)$ is also true for all x. (The question doesn't ask about the other direction, which is not true)

(f) TRUE. Since $L(\varphi) = \emptyset$.

(g) TRUE. Due to semi-decidability of FOL

(h) FALSE. The DFA accepting the empty set $\emptyset$ has no accepting state.

(i) FALSE. The reversal of a DFA need not be a DFA.

(j) TRUE. See Problem Sheet 5 Qn 8,9.

Rubrics

For each question +1 mark is given if the answer is correct. -1 mark for each wrong answer and 0 marks for unattempted questions.

(f) was given as a BONUS. So, everyone receives 1 marks for that.

As mentioned in the question paper, the marks for this question is

$$\max(0, \text{marks you scored})$$

.

2. [**7 marks**] Define *Positive resolution* as a restriction of ordinary resolution as follows: derive a resolvent from clauses C_1 and C_2 only if C_1 is a positive clause, i.e., it consists only of positive literals. Prove or disprove : If F is an unsatisfiable CNF formula then one can derive the empty clause from F using only positive resolution.

   Question on propositional logic. Note that this question tweaks the standard resolution we did in class. The proof of standard resolution was discussed completely in class, and is also there on the slides. This question expects you to modify that proof.

   Solution

   Following the proof we did in the lecture, we prove the desired result by induction on the number of variables in F. The base case is that F has no variables. Then we must have $F = \{\}$, so there is a trivial positive resolution refutation of F.

   The induction step is as follows. Pick a variable P in F and consider the formulas $F_0 := F[\mathbf{false}/P]$ (notation replace P by true/false) and $F_1 := F[\mathbf{true}/P]$. Since F is unsatisfiable, F_0 and F_1 are both unsatisfiable. Since F_0 contains one fewer variable than F, by the induction hypothesis there is a positive resolution proof $C_0, C_1, \ldots, C_m = \bot$ that derives the empty clause from F_0. For each clause C_i in the above proof that comes from F_0, either C_i is already in F or $C_i \cup \{P\}$ is in F. By replacing the deleted copies of P from the latter type of clause we obtain a new (still positive) resolution proof $C'_0, C'_1, \ldots, C'_m$ from F, where either C'_m is empty or $C'_m = \{P\}$. In the first case there is nothing more to prove, so suppose $C'_m = \{P\}$, i.e., we have a positive resolution derivation of $\{P\}$ from F.

   Now we consider F_1. By induction, we also have a positive resolution proof $C_0, \ldots, C_m = \{\}$ from F_1. For each clause C_i in the above proof that comes from F_1, either C_i is already in F or $C_i \cup \{\neg P\}$ is in F. In the latter case we can obtain C_i from $C_i \cup \{\neg P\}$ and $\{P\}$ by a single positive resolution step. Thus we can construct a positive resolution proof from F.

   Rubrics

   - 7 marks only if the solution is completely correct.
   - 2 marks for every observation (with proof) that can significantly contribute to a correct proof (upto grader's discretion)

3. [**5 marks**] Define a clause of a CNF formula to be *syntactically valid* if it contains both a literal and its complement. Give an efficient algorithm to check the *semantic validity* of a CNF formula and give its time complexity. Argue for the correctness of your algorithm.

   A formula F is semantically valid *if F evaluates to true in all rows of its truth table.*

   Question on propositional logic. This question uses the idea behind validity of disjunctions.

Solution

Claim. *A CNF formula φ is* semantically valid *iff all its clauses are* syntactically valid.

Proof. ($\Longleftarrow$) Since $p \vee \neg p \equiv \top$, if all clauses are *syntactically valid*, for each clause C_i there is at least one literal, ℓ, such that ℓ and $\neg\ell$ are both present in C_i. Thus, for each C_i, $C_i \equiv \top$ and thus, $\varphi \equiv \top$.

($\Longrightarrow$) Assume that φ is *semantically valid* and there exists a clause C_k which is not *syntactically valid*. Let the variables in C_k be $p_1, \ldots, p_n$. For each p_i, either p_i is in C_k or $\neg p_i$ is in C_k. So, consider the assignment $\mathcal{A}$ such that $\mathcal{A}[p_i] = 0$ if p_i is present in C_k and $\mathcal{A}[p_i] = 1$ if $\neg p_i$ is present in C_k. Observe that $\mathcal{A} \not\models C_k \implies \mathcal{A} \not\models \varphi$, leading to a contradiction.

So all we need to do is to check whether all the clauses are *syntactically valid*. Algorithm 1 checks whether a given input clause C is syntactically valid. Algorithm 2 gives the final algorithm to check the semantic validity of a CNF formula φ.

Time Complexity. Since our algorithm just needs to read through the whole CNF formula at once, it takes $\mathcal{O}(n)$ time, where n is the length of the CNF.

Algorithm 1 Syntactic_VAL

```
Require: Clause $C$
1: for each literal $\ell$ in $C$ do
2:     if $\neg l$ is also in $C$ then
3:         return true
4:     end if
5: end for
6: return false
```

Algorithm 2 CNF_VAL

```
Require: CNF formula $\varphi$
1: for each clause $C_i$ in $\varphi$ do
2:     if Syntactic_VAL($C_i$) == false then
3:         return false
4:     end if
5: end for
6: return true
```

Rubrics

- 2 marks for correct algorithm, either written as pseudocode or in words.
- 2 marks for correctness of the algorithm. 1 mark each for each direction

- 1 mark for time complexity.

General Comments:

- For the algorithm you are expected to write when to conclude that the CNF is valid, and when not to.
- Some people have written the algorithm and correctness together. Marks are given if their solution is complete.
- A common mistake is to make a claim that "semantic validity" $\Leftrightarrow$ "syntactic validity" due to soundness and completeness of propositional logic. However, realise that these are different things. Marks were deducted for this.
- For time complexity, any valid polynomial complexity is accepted. However, just writing "polynomial" complexity is not given marks.
- No marks for DPLL or Resolution based algorithms as they have exponential worst-case complexities.

4. **[7 marks]** A *renamable Horn formula* is a CNF formula that can be turned into a Horn formula by negating (all occurrences of) some of its variables. For example,

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

can be turned into a Horn formula by negating p_1 and p_2.

Given a CNF-formula F, show how to derive a 2-CNF formula G such that G is satisfiable if and only if F is a renamable Horn formula. Show moreover that one can derive a renaming that turns F into a Horn formula from a satisfying assignment for G.

Question on propositional logic. In fact, the seed of this question came from a student who raised a question during the class when we did Horn. His question was about renaming variables to obtain a Horn formula. A related question was also done in the tut.

Solution

Let $p_1, \ldots, p_n$ be the propositional variables appearing in F. We define G over the same set of propositional variables, where the intuitive meaning of p_i in G is "negate the variable p_i in F". The definition of G is such that a satisfying assignment α of G gives a recipe to obtain a Horn formula from F—negate those variables p_i in F for which $\alpha(p_i) = 1$.

Now the set of clauses of G is just the set of all two-element subsets of clauses of F. For example, if p_1, p_2 both appear in some clause of F then G contains a clause $\{p_1, p_2\}$ meaning that either p_1 or p_2 must be negated in F. A simple case analysis (do it!!) shows that this definition ensures that no pair of literals in some clause of F are both positive after the renaming prescribed by G.

Rubrics

- 3 marks for "Show moreover that one can derive a renaming that turns F into a Horn formula from a satisfying assignment for G."
- 4 marks for showing how to construct a 2-CNF G with the required conditions holding.

Rest is based on errors you have done and how severe they are.

5. **[2 + 5 + 8 = 15 marks]** In this question we work with first-order logic without equality; that is, we do not use the atomic formula $x = y$ where x, y are terms. *So, you are bound to lose marks if you use the "=" predicate unless you properly define it using the signature given.*

   (a) Consider a signature τ containing only a binary relation symbol R. For each positive integer n show that there is a satisfiable τ-formula F_n such that every structure $\mathcal{A}$ that satisfies F_n has at least n elements.

   (b) Fix a signature τ and τ-structures $\mathcal{A}, \mathcal{B}$ and assignments α, β. Consider a relation $\sim$ on pairs $(\mathcal{A}, \alpha)$, $(\mathcal{B}, \beta)$ that satisfies the following two properties:

   (P1) If $(\mathcal{A}, \alpha) \sim (\mathcal{B}, \beta)$ then for every atomic formula F we have $\mathcal{A} \models_\alpha F$ iff $\mathcal{B} \models_\beta F$.

   (P2) If $(\mathcal{A}, \alpha) \sim (\mathcal{B}, \beta)$ then for each variable x we have (i) for each $a \in U^{\mathcal{A}}$ there exists $b \in U^{\mathcal{B}}$ such that $(\mathcal{A}, \alpha[x \mapsto a]) \sim (\mathcal{B}, \beta[x \mapsto b])$, and (ii) for all $b \in U^{\mathcal{B}}$ there exists $a \in U^{\mathcal{A}}$ such that $(\mathcal{A}, \alpha[x \mapsto a]) \sim (\mathcal{B}, \beta[x \mapsto b])$.

   If $(\mathcal{A}, \alpha) \sim (\mathcal{B}, \beta)$, then show that for any formula F built from atomic formulae using the connectives $\neg, \wedge, \exists$, we have $\mathcal{A} \models_\alpha F$ iff $\mathcal{B} \models_\beta F$.

   (c) Consider a signature τ containing only unary predicate symbols $P_1, \ldots, P_k$. Using (b), show that any satisfiable τ-formula has a structure with at most 2^k elements in its universe satisfying it.

The only question on First Order Logic. One of the non trivial questions in the lot.

Solution

(a) One solution is

$$F_n = \forall x\, \neg R(x, x) \wedge \exists x_1 \ldots \exists x_n \bigwedge_{1 \le i < j \le n} R(x_i, x_j)\,.$$

(b) The proof is by structural induction on F for all $(\mathcal{A}, \alpha)$ and $(\mathcal{B}, \beta)$.

The statement holds for atomic formulae F by definition of $\sim$. The induction steps for $\wedge$ and $\neg$ are straightforward. The induction step for the existential quantifier is as follows.

If $\mathcal{A} \models_\alpha \exists x G$, then $\mathcal{A} \models_{\alpha[x \mapsto a]} G$ for some $a \in U^{\mathcal{A}}$. By the assumption $(\mathcal{A}, \alpha) \sim (\mathcal{B}, \beta)$, there exists $b \in U^{\mathcal{B}}$ such that $(\mathcal{A}, \alpha[x \mapsto a]) \sim (\mathcal{B}, \beta[x \mapsto b])$. By the induction hypothesis, $\mathcal{B} \models_{\alpha[x\mapsto b]} G$. Thus, $\mathcal{B} \models_\beta \exists x G$. The converse implication, $\mathcal{B} \models_\beta \exists x G$ implies $\mathcal{A} \models_\alpha \exists x G$ follows by symmetry.

(c) We use (b) to solve this. Consider some $(\mathcal{A}, \alpha)$ and $(\mathcal{B}, \beta)$. Define an equivalence relation E on the universe of $\mathcal{A}$ by

$$E = \{(a, b) \in U^{\mathcal{A}} \times U^{\mathcal{A}} \mid a \in P^{\mathcal{A}} \text{ iff } b \in P^{\mathcal{B}} \text{ for all predicate symbols P}\}$$

We say that $(\mathcal{B}, \beta)$ is the quotient of $(\mathcal{A}, \alpha)$ by E if $U^{\mathcal{B}}$ is the set of equivalence classes of $U^{\mathcal{A}}$ wrt E, $P^{\mathcal{B}} = \{[a] \mid a \in P^{\mathcal{A}}\}$ for all predicate symbols P and $\beta(x) = [\alpha(x)]$. Then $U^{\mathcal{B}}$ has $\leq 2^k$ elements (think of all subsets of $P_1, \ldots, P_k$. A partition consists of elements which are in a subset of $P_1, \ldots, P_k$). For instance, if $U^{\mathcal{A}} = \{1, 2, \ldots, 10\}$ and if there are $k = 3$ predicates P_1, P_2, P_3 such that $P_1^{\mathcal{A}} = \{2, 3, 6, 7, 10\}, P_2^{\mathcal{A}} = \{2, 5, 6, 7, 9\}, P_3^{\mathcal{A}} = \{2, 3, 4, 5, 8\}$, then the equivalence classes of $U^{\mathcal{A}}$ are $\{[10], [9], [8], [7] = [6], [5] = [4], [3], [2], [1]\}$, and $E = \{(10, 10), (9, 9), (8, 8), (7, 7), (6, 7), (5, 5), (4, 5), (3, 3), (2, 2), (1, 1)\}$ qualifies as an equivalence relation.

Define a relation $(\mathcal{A}, \alpha) \sim (\mathcal{B}, \beta)$ iff $(\mathcal{B}, \beta)$ is a quotient of $(\mathcal{A}, \alpha)$ by the equivalence relation E. It is clear that $\sim$ satisfies condition (P1) from (b). To see that it also satisfies (P2), observe that if $(\mathcal{B}, \beta)$ is a quotient of $(\mathcal{A}, \alpha)$ by E then for any $a \in U^{\mathcal{A}}$, $(\mathcal{B}, \beta[x \mapsto [a]])$ is the quotient of $(\mathcal{A}, \beta[x \mapsto a])$ by E.

In (b), we showed that if $(\mathcal{A}, \alpha) \sim (\mathcal{B}, \beta)$, then $\mathcal{A} \models_\alpha F$ iff $\mathcal{B} \models_\beta F$. The result follows.

Rubrics

(a) 2 marks for correct formula F_n. No marks otherwise.

(b) 1 mark for Base case and Induction hypothesis. 1 mark each for the induction steps of $\wedge$ and $\neg$. 2 marks for existential quantifier.

(c) Partial Marks are given as is relevant.

General Comments:

- For (a) one common mistake is to define R in a specific way for each structure and then use it to create F_n. However, the question demands that your formula should not be satisfiable for any structure with less than n elements. So, no marks were awarded for this.
- For (b), it is observed that many don't do structural induction properly. This time the grading was liberal enough to accommodate small mistakes, and marks were awarded if the answers were reasonably correct.
- For (c), only 2 people made reasonable attempts. They are given partial marks according to merit.

6. [**5 marks**] Assume that a 2-CNF formula is input to DPLL. Prove that every clause that is learnt is either empty or a singleton.

Question of propositional logic. Directly tests your understanding of DPLL. Related questions on DPLL pertaining to Horn was done in the tut. This question asks about DPLL on the other well known polytime class of 2-CNF.

Solution with implication graph

Let the input formula be φ. The key property of 2-CNF is that for any clause $p \to q$ in $Res^*(\varphi)$, there is already a path from p to q in the implication graph of φ. This is easily seen by induction on resolution.

Recall that when we learn a clause C in DPLL when the partial assignment so far is α, then all variables in C are decision variables wrt α and all literals in C are made false by α.

Assume for a contradiction that C has two literals ℓ_1, ℓ_2. Then ℓ_1, ℓ_2 are both decision variables and the assignment α has the form $(\dots, \ell_1 \mapsto 0, \ell_2 \mapsto 0, \dots)$. But, as we noted above there is a path from $\neg\ell_1$ to ℓ_2 in the implication graph of φ. Then, after as assignment $\ell_1 \mapsto 0$, unit propagation would give $\ell_2 \mapsto 1$ which is a contradiction.

Solution directly using algorithm

Let the input formula be φ. By virtue of it being a 2-CNF we can express each of its clauses as $(\ell_i \vee \ell_j)$ where each literal $\ell = p$ or $\neg p$ for a variable p. Note that we learn the empty clause if $\perp \in F|_\phi$ where ϕ denotes the empty assignment. Let's assume this is not the case.

Recall that we learn a clause in DPLL when we get a conflict clause in resolved φ. So in order to get a false clause in φ, call it $(\ell_1 \vee \ell_2)$, we must've had either sequential falsification, *i.e.* WLoG ℓ_1 becoming false before ℓ_2, or we'd have simultaneous falsification, *i.e.* ℓ_1 and ℓ_2 simultaneously becoming false. The former case is not possible as if we had ℓ_1 assigned false, by virtue of unit propagation we'd automatically assign ℓ_2 to true and we'd thus remove ℓ_2 from $F|_\alpha$.

In the latter case we learn about exactly one literal, *i.e.* we learn a singleton clause since simultaneous falsification implies that ℓ_1 and ℓ_2 were both in resolved ϕ under α is itself a 2-CNF. This is because if some ℓ was present in a clause with a false literal it'd be true by unit propagation and if it were with a true literal the clause would be removed. Thus, both ℓ_1 and ℓ_2 were free variables in resolved ϕ, which both became simultaneously false after a decision variable ℓ was assigned. Thus we learn $\neg\ell$, a singelton clause. QED.

Rubrics

(a) 2 marks for considering all possible cases

(b) 3 marks for the correct proof

(c) Partial Marks are given for less rigorous proofs which miss some cases.

General Comments:

- A common mistake is to attempt to give proof just using the fact that the conflict clause has only 2 literals without showing why simultaneous falsification can't occur.
- Some students have not used any properties of 2-CNF in their proof. They have been awarded zero marks.
- No marks have been awarded for running the DPLL algorithm on specific cases.

7. [**4 marks**] Consider the language $L = \{w \in \{a,b\}^* \mid w$ has equal number of occurrences of patterns ab and $ba\}$. For example, $aab \notin L$ since it has one occurrence of ab and zero occurrences of ba while $aba \in L$ since it has one occurrence each of both ab and ba. Show that L is FO-definable and regular.

In order to show that L is FO-definable you need to exhibit a FOL formula φ and argue why $L(\varphi) = L$. Similarly to show that L is regular, you should draw a DFA and argue that the language accepted by the DFA is L.

Question from tut.

Solution

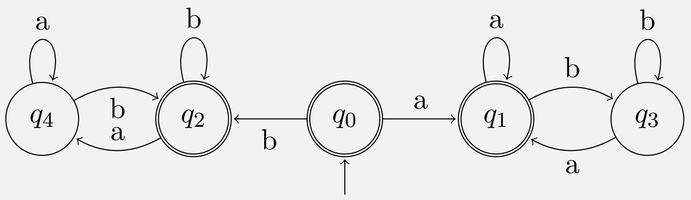
Observe that $L = a\Sigma^*a \cup b\Sigma^*b \cup \{\varepsilon, a, b\}$ where $\Sigma = \{a,b\}$. This can be shown by first observing that $\varepsilon \in L$ and then considering any word $w \in a\Sigma^*$ (starting with a) and showing that $w \in L \iff w$ ends with a (by considering two states based on last input symbol and noting that each transition increases the count of either ab or ba by 1 and hence $(\#ab)_w = (\#ba)_w \iff w$ starts and ends with same symbol. Prove similarly for words starting with b.

Taking first$(x) \equiv \neg\exists\ y[y < x]$ and last$(x) \equiv \neg\exists y\ [x < y]$, some of the common correct FOL sentences φ for which $L(\varphi) = L$ are:

- $\varphi_1 = \forall x\,(\text{first}(x) \implies Q_a(x) \land \text{last}(x) \implies Q_a(x))$
  $\varphi_2 = \forall x\,(\text{first}(x) \implies Q_b(x) \land \text{last}(x) \implies Q_b(x))$
  $\varphi = \varphi_1 \lor \varphi_2$
  $L(\varphi_1) = \{w : w \text{ starts and ends with a}\} \bigcup \{\varepsilon\}$
  $L(\varphi_1) = \{w : w \text{ starts and ends with b}\} \bigcup \{\varepsilon\}$
  $L(\varphi) = L(\varphi_1) \bigcup L(\varphi_2) = L$
- $\varphi_1 = \exists x \exists y\,(\text{first}(x) \land \text{last}(y) \land ((Q_a(x) \land Q_a(y)) \lor (Q_b(x) \land (Q_b(y))))$
  $\varphi_2 = \forall x(\neg(x = x))$
  $\varphi = \varphi_1 \lor \varphi_2$

$L(\varphi_1) = \{w : w \text{ starts and ends with the same alphabet and } |w| \geq 1\}$
$L(\varphi_2) = \{\varepsilon\}$
$L(\varphi) = L(\varphi_1) \bigcup L(\varphi_2) = L$

One of the DFAs (and the *minimal* one), whose corresponding language is L, is



$$\begin{tikzcd} q_4 \arrow[loop above, "a"] \arrow[r, bend left, "b"] & q_2 \arrow[loop above, "b"] \arrow[l, bend left, "a"] & q_0 \arrow[l, "b"] \arrow[r, "a"] & q_1 \arrow[loop above, "a"] \arrow[r, bend left, "b"] & q_3 \arrow[loop above, "b"] \arrow[l, bend left, "a"] \end{tikzcd}$$

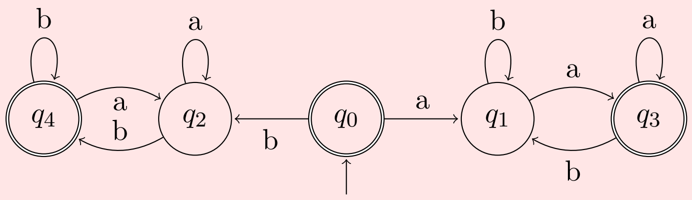
Denoting $L_i = \{w : \hat{\delta}(q_0, w) = q_i\}$ in this DFA, we can see that
$L_0 = \{\varepsilon\}$, $L_1 = \{a\} \bigcup a\Sigma^* a$, $L_3 = a\Sigma^* b$, $L_2 = \{b\} \bigcup b\Sigma^* b$, $L_4 = b\Sigma^* a$, and this can be proved by induction on $|w|$.

Rubrics

- 1 mark for correct FOL formula, and 1 mark for justification (argument for why $L = a\Sigma^* a \cup b\Sigma^* b \cup \{\epsilon, a, b\}$ or any other argument for why $L(\varphi) = L$)
- 1 mark for correct DFA, and 1 mark for justification (argument for why the language accepted by the DFA is exactly L - one justification is by describing the set of words $L_i = \{w : \hat{\delta}(q_0, w) = q_i\}$ corresponding to each state q_i.)

General Comments:

- A common mistake is not accounting for ε in L (by not making the start state of the DFA accepting or ε not in $L(\varphi)$). Marks have not been deducted if this is the **only** error.
- Marks have not been given for incorrect DFA's (those automatons which do not have both $\hat{\delta}(q_i, a)$ and $\hat{\delta}(q_i, b)$ transitions for each state), and those DFA's whose language accepted is not L or $L \backslash \{\varepsilon\}$, and similarly for incorrect FOL sentences.
- One common incorrect DFA, which has not been awarded any marks, is



$$\begin{tikzcd} q_4 \arrow[loop above, "b"] \arrow[r, bend left, "a"] & q_2 \arrow[loop above, "a"] \arrow[l, bend left, "b"] & q_0 \arrow[l, "b"] \arrow[r, "a"] & q_1 \arrow[loop above, "b"] \arrow[r, bend left, "a"] & q_3 \arrow[loop above, "a"] \arrow[l, bend left, "b"] \end{tikzcd}$$

This DFA does not accept a and b, but $a, b \in L$.

8. **[7 marks]** Consider the formula in 3-CNF,

$$\varphi = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee \neg x_2 \vee x_4)$$

Show that one can construct a graph G_φ from formula φ such that

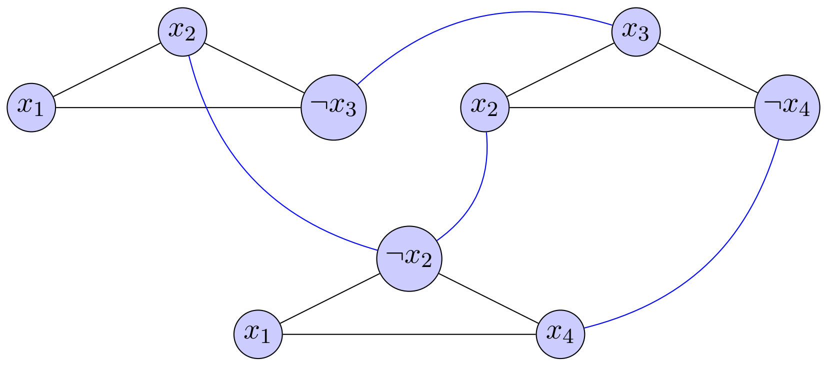


Figure 1: Construction of G_φ

- G_φ has an independent set of size k iff φ is satisfiable, where k is the number of literals assigned true in the satisfying assignment.
- The size of G_φ is polynomial in the size of φ.
- G_φ must be inspired from φ; that is, you must have an "encoding scheme" to construct a graph G_φ from any formula φ.

An independent set in a graph is a set of vertices such that no two of them are adjacent.

Give proper explanation to validate that your construction of G_φ is correct.

One of the non-trivial questions of the lot.

Solution

The graph G_φ is constructed from a 3-CNF formula φ as follows:

- For every clause $C_i = (l_1 \vee l_2 \vee l_3)$ in the formula φ, we introduce a triangle in the graph. The triangle has one vertex corresponding to each literal in the clause. That is, the vertices $v_{l_1}, v_{l_2}, v_{l_3}$ are introduced, and edges are drawn between them to form a triangle.
- Add edges between vertices corresponding to literals x_i and $\neg x_i$ across different clauses. This means that if a vertex in one triangle corresponds to x_i, and a vertex in another triangle corresponds to $\neg x_i$, then an edge is added between these two vertices.

Correctness of the construction.
($\implies$) Suppose φ is satisfiable, choose one satisfying assignment. From each clause, pick an arbitrary literal which is set to true. Pick vertices corresponding to the picked literals. The number of vertices picked is equal to the number of clauses. The picked vertices form an independent set because

- We pick one vertex from each triangle
- If vertex corresponding to literal x is picked then its neighbors correspond to $\neg x_i$, and hence will not be picked.

($\Longleftarrow$) Suppose G_φ has an independent set of size k. Fix k as the number of triangles. Then the independent set must have one vertex from each triangle. For each vertex in the independent set, set the corresponding literal true. This is possible because the independent set cannot have vertices corresponding to both x_i and $\neg x_i$. Set the remaining variables (if any) arbitrarily. This is a satisfying assignment φ for because for every clause at least one literal is set true.

Rubrics

Marks given only to correct, polynomial time construction of G_φ.