Logic in CS Autumn 2024

Problem Sheet 1

S. Krishna

1. There are three suspects for a murder: A, B, and C.

   - A says "I did not do it. The victim was an old acquaintance of B's. But C hated him."
   - B states "I did not do it. I did not even know the guy. Besides I was out of town all that week."
   - C says "I did not do it. I saw both A and B downtown with the victim that day; one of them must have done it."

   Assume that the two innocent men are telling the truth, but that the guilty man might not be. Who did it? Deduce the answer by encoding in propositional logic and finding a solution.

Solution

All three of them say, "I did not do it". So, this does not give any useful information to us. (*Why*?) Let us define the following propositions:

$$p_1 : \text{victim is acquaintance of B}$$

$$p_2 : \text{B was out of town}$$

Then, we can encode the statements of A, B, and C as $a = p_1$, $b = \neg p_1 \wedge p_2$, and $c = \neg p_2$ respectively. Since we know at least two people tell the truth, the formula, $(a \wedge b) \vee (b \wedge c) \vee (c \wedge a)$ has to be `true`.

$$a \wedge b = (p_1 \wedge (\neg p_1 \wedge p_2)) \equiv \bot$$
$$b \wedge c = ((\neg p_1 \wedge p_2) \wedge \neg p_2)) \equiv \bot$$

Hence, $c \wedge a$ is `true` $\Rightarrow$ c is `true` and a is `true` $\Rightarrow$ B is guilty

2. In an island, there are three tribes : the Knights, Knaves and Normals. The Knights always speak truth, the Knaves always lie, while the Normals lie sometimes and speak truth sometimes. On a visit to this island, I met two inhabitants A and B. A told me that B is a knight and B told me that A is a knave. Prove, using natural deduction, that one of them told the truth but is not a knight, or that one of them told a lie but is not a knave.

Solution

Note that if B is a knight, then A must be a knave, which makes her statement that B is a knight false, which contradicts the fact that B is a knight. Therefore B is not a knight. This means A's statement is false, which means A is also not a knight. So both are either normals or knaves, and A's statement is false. Let us take some cases now:

(a) If A is a knave, then B's statement is true, which means B must be a normal. This means one of them (B) told the truth but is not a knight.

(b) If B is a knave, then A must be a normal. This means one of them (A) told a lie but is not a knave.

(c) If both A and B are normals, then both their statements are false, which means one of them (either) told a lie but is not a knave.

Therefore, by cases, we can say that one of them told the truth but is not a knight, or that one of them told a lie but is not a knave.
This natural language proof can also be directly translated into a formal proof.

3. Let F, G and H be formulas and let $\mathbf{S}$ be a set of formulas. Which of the following statements are true? Justify your answer.

   (a) If F is unsatisfiable, then $\neg F$ is valid.
   (b) If $F \to G$ is satisfiable and F is satisfiable, then G is satisfiable.
   (c) $P_1 \to (P_2 \to (P_3 \to \ldots (P_n \to P_1) \ldots))$ is valid.
   (d) $\mathbf{S} \models F$ and $\mathbf{S} \models \neg F$ cannot both hold.
   (e) If $\mathbf{S} \models F \vee G$, $\mathbf{S} \cup \{F\} \models H$ and $\mathbf{S} \cup \{G\} \models H$, then $\mathbf{S} \models H$.

Solution

(a) True. Let $\mathcal{A}$ be an arbitrary assignment. Since F is unsatisfiable we have $\mathcal{A}[\![F]\!] = 0$ and thus $\mathcal{A}[\![\neg F]\!] = 1$.

(b) False. A counterexample is $P \to \texttt{false}$ for an atomic proposition P.

(c) True. Consider an assignment $\mathcal{A}$. If $\mathcal{A}[\![P_1]\!] = 0$ then the outermost implication is true. If $\mathcal{A}[\![P_1]\!] = 1$ then, arguing from the inside outwards, all the implication subformulas are true.

(d) False. If $\mathbf{S}$ is unsatisfiable then $\mathbf{S} \models F$ and $\mathbf{S} \models \neg F$ for any F.

(e) True. Let $\mathcal{A}$ be a model of $\mathbf{S}$. Since $\mathbf{S} \models F \vee G$, $\mathcal{A}$ is a model of F or a model of G. In the first case, since $\mathbf{S} \cup \{F\} \models H$, $\mathcal{A}$ is a model of H. Likewise in the second case, since $\mathbf{S} \cup \{G\} \models H$, $\mathcal{A}$ is a model of H. Since the two cases are exhaustive, $\mathcal{A}$ is a model of H. Thus every model of $\mathbf{S}$ is a model of H.

4. The Pigeon Hole Principle states that if there are $n+1$ pigeons sitting amongst n holes then there is atleast one hole with more than one pigeon sitting in it. For $i \in \{1, 2, .., n+1\}$ and $j \in \{1, 2, ..., n\}$, let the atomic proposition $P(i, j)$ indicate that the i-th pigeon is sitting in the j-th hole.

   Write out a propositional logic formula that states the Pigeon Hole Principle.

Solution

Let $P(i, j)$ represent the proposition that the i^{th} pigeon is sitting in the j^{th} hole, where $i \in \{1 \dots n+1\}$ and $j \in \{1 \dots n\}$.
The Pigeonhole principle states that, if there are $n+1$ pigeons and n holes, and every pigeon sits in exactly one hole, then there is a hole occupied by more than one pigeon. To convert this into a PL formula, let us convert each side of the implication into PL first.
Every pigeon sits in at least one hole can be expressed in PL as:

$$\bigwedge_{i=1}^{n+1} \bigvee_{j=1}^{n} P(i, j)$$

Here the inner disjunction refers to the i^{th} pigeon sitting in some hole, and the outer conjuction makes it so that every pigeon must sit in some hole. Call this condition F.
We also need no pigeon to sit in multiple holes. Say pigeon i sits in holes j and k with $j < k$. The formula $P(i, j) \wedge P(i, k)$ represents this scenario. There exists a pigeon sitting in multiple holes therefore becomes:

$$\bigvee_{i=1}^{n+1} \bigvee_{\substack{j,k=1 \\ j<k}}^{n} (P(i, j) \wedge P(i, k))$$

Here, the inner disjunction refers to the i^{th} pigeon sitting in multiple holes and the outer disjunction refers to there existing a pigeon sitting in multiple holes.
Negating this, we get the condition for no pigeon to sit in multiple holes:

$$\bigwedge_{i=1}^{n+1} \bigwedge_{\substack{j,k=1 \\ j<k}}^{n} (\neg P(i, j) \vee \neg P(i, k))$$

Call this condition G.
Now, say hole k is occupied by pigeons i and j with $i < j$. We then have $P(i, k) \wedge P(j, k)$.
There exists a hole occupied by more than one pigeon therefore becomes:

$$\bigvee_{k=1}^{n} \bigvee_{\substack{i,j=1 \\ i<j}}^{n+1} (P(i, k) \wedge P(j, k))$$

Here, the inner disjunction refers to the k^{th} hole being occupied by more than one pigeon and the outer disjunction refers to there existing a hole occupied by multiple pigeons. Call this condition H.
The Pigeonhole Principle therefore becomes:

$$F \wedge G \implies H$$

5. Prove formally $\vdash [(p \to q) \to q] \to [(q \to p) \to p]$

Solution

1.	$(p \to q) \to q$	assumption
2.	$q \to p$	assumption
3.	$\neg p$	assumption
4.	$\neg q$	MT 2, 3
5.	$\neg(p \to q)$	MT 1, 4
6.	p	assumption
7.	$\perp$	$\perp i$ 3, 6
8.	q	$\perp e$ 7
9.	$p \to q$	$\to i$ 6-8
10.	$\perp$	$\perp i$ 5, 9
11.	$\neg\neg p$	$\neg i$ 3, 10
12.	p	$\neg\neg e$ 11
13.	$(q \to p) \to p$	$\to i$ 1-12
14.	$((p \to q) \to q) \to ((q \to p) \to p)$	$\to i$ 1-13

6. Let $\mathcal{H}$ be a given set of premises. If $\mathcal{H} \vdash (A \to B)$ and $\mathcal{H} \vdash (C \vee A)$, then show that $\mathcal{H} \vdash (B \vee C)$ where A, B, C are wffs.

Solution

1.	$\mathcal{H}$	premises
2.	$A \to B$	as $\mathcal{H} \vdash (A \to B)$
3.	$C \vee A$	as $\mathcal{H} \vdash (C \vee A)$
4.	C	assumption
5.	$B \vee C$	$\vee i$ 3
6.	A	assumption
7.	B	MP 1, 5
8.	$B \vee C$	$\vee i$ 6
9.	$B \vee C$	$\vee e$ 2, 3-4, 5-7

7. Let $\mathcal{H}$ be a given set of premises. If $\mathcal{H} \vdash (A \to C)$ and $\mathcal{H} \vdash (B \to C)$, then show that $\mathcal{H} \vdash ((A \vee B) \to C)$. Here, A, B and C are wffs.

Solution

1.	$\mathcal{H}$	premises
2.	$A \to C$	as $\mathcal{H} \vdash (A \to C)$
3.	$B \to C$	as $\mathcal{H} \vdash (B \to C)$
4.	$A \lor B$	assumption
5.	A	assumption
6.	C	MP 1, 4
7.	B	assumption
8.	C	MP 2, 6
9.	C	$\lor e$ 3, 4-5, 6-7
10.	$(A \lor B) \to C$	$\to i$ 3-8

8. Show that a truth assignment α satisfies the wff

$$(\dots(x_1 \leftrightarrow x_2) \leftrightarrow \dots \leftrightarrow x_n)$$

iff $\alpha(x_i) = \texttt{false}$ for an even number of i's, $1 \leq i \leq n$.

Solution

Base Case. For $n = 1$, the wff will be (x_1), which is `true` iff $\alpha(x_1) = \texttt{true}$.
Induction Hypothesis. Assume α satisfies the wff $(\dots(x_1 \leftrightarrow x_2) \leftrightarrow \dots \leftrightarrow x_{n-1})$ iff $\alpha(x_i) = \texttt{false}$ for an even number of i's, $1 \leq i \leq n-1$.
Now consider $(\dots(x_1 \leftrightarrow x_2) \leftrightarrow \dots \leftrightarrow x_n)$. Here, $\alpha(x_n)$ can be `true` or `false`.

- If $\alpha(x_n) = \texttt{true}$, and the wff $(\dots(x_1 \leftrightarrow x_2) \leftrightarrow \dots \leftrightarrow x_n)$ is `true` $\Leftrightarrow$ $(\dots(x_1 \leftrightarrow x_2) \leftrightarrow \dots \leftrightarrow x_{n-1})$ is `true` $\Leftrightarrow \alpha(x_i) = \texttt{false}$ for an even number of i's, $1 \leq i \leq n-1 \Leftrightarrow \alpha(x_i) = \texttt{false}$ for an even number of i's, $1 \leq i \leq n$

- If $\alpha(x_n) = \texttt{false}$, and the wff $(\dots(x_1 \leftrightarrow x_2) \leftrightarrow \dots \leftrightarrow x_n)$ is `true` $\Leftrightarrow$ $(\dots(x_1 \leftrightarrow x_2) \leftrightarrow \dots \leftrightarrow x_{n-1})$ is `false` $\Leftrightarrow \alpha(x_i) = \texttt{false}$ for an odd number of i's, $1 \leq i \leq n-1 \Leftrightarrow \alpha(x_i) = \texttt{false}$ for an even number of i's, $1 \leq i \leq n$

9. Of the following three formulae, which tautologically imply which?

(a) $x \leftrightarrow y$

(b) $(\neg((x \leftrightarrow y) \to (\neg y \to x))))$

(c) $((\neg x \lor y) \land (x \lor \neg y))$

Solution

A wff p is said to *tautologically imply* a wff q if there is no truth assignment α which makes p `true` and q `false`. In this question, this can be seen by looking at the truth tables of given formulae:

x	y	$x \leftrightarrow y$	$(\neg((x \leftrightarrow y) \to (\neg y \to x)))$	$((\neg x \vee y) \wedge (x \vee \neg y))$
1	1	1	0	1
1	0	0	0	0
0	1	0	0	0
0	0	1	1	1

Clearly, from the truth tables we can conclude that:

- $x \leftrightarrow y$ tautologically implies $((\neg x \vee y) \wedge (x \vee \neg y))$
- $((\neg x \vee y) \wedge (x \vee \neg y))$ tautologically implies $x \leftrightarrow y$
- $(\neg((x \leftrightarrow y) \to (\neg y \to x))))$ tautologically implies $x \leftrightarrow y$
- $(\neg((x \leftrightarrow y) \to (\neg y \to x))))$ tautologically implies $((\neg x \vee y) \wedge (x \vee \neg y))$

10. Let $\mathcal{L}$ be a formulation of propositional logic in which the sole connectives are negation and disjunction. The rules of natural deduction corresponding to disjunction and negation (also includes double negation) are available. For any wffs A, B and C, let $\neg(A \vee B) \vee (B \vee C)$ be an axiom of $\mathcal{L}$. Show that any wff of $\mathcal{L}$ is a theorem of $\mathcal{L}$.

Solution

An axiom of a proof system is a formula that can always be taken as `true`. A theorem is a logical consequence of axioms, i.e a wff in a proof system is a theorem if it can be derived from axioms using the proof rules of the system. Let P be any wff of $\mathcal{L}$. We shall apply the axiom by choosing $A = \neg P$, $B = \bot$ and $C = P$.

1.	$\neg(\neg P \vee \bot) \vee (\bot \vee P)$	premise (axiom)
2.	$\neg(\neg P \vee \bot)$	assumption
3.	$\neg P$	assumption
4.	$\neg P \vee \bot$	$\vee i$ 3
5.	$\bot$	$\bot i$ 2, 4
6.	$\neg\neg P$	$\bot e$ 3-5
7.	P	$\neg\neg e$ 6
8.	$(\bot \vee P)$	assumption
9.	$\bot$	assumption
10.	P	$\bot e$
11.	P	assumption
12.	P	$\vee e$ 8, 9-10, 11
13.	P	$\vee e$ 1, 2-7, 8-12

Notice that we can also derive any wff P, if we can derive $\bot$.

11. Let $\mathcal{P}$ denote propositional logic. Suppose we add to $\mathcal{P}$ the axiom schema $(A \rightarrow B)$ for wffs A, B of $\mathcal{P}$. Comment on the consistency of the resulting logical system obtained. A logic system $\mathcal{P}$ is inconsistent if it is capable of producing $\bot$ using the rules of natural deduction.

Solution

The resulting logical system is inconsistent, since we can produce $\bot$ as follows. Let φ be any wff of $\mathcal{P}$.

1.	$(\varphi \vee \neg\varphi) \rightarrow \bot$	premise (axiom)
2.	$\varphi \vee \neg\varphi$	LEM
3.	$\bot$	MP 1, 2

Logic in CS Autumn 2024

Problem Sheet 2

S. Krishna

1. An adequate set of connectives is a set such that for every formula there is an equivalent formula with only connectives from that set. For example, $\{\neg, \vee\}$ is adequate for propositional logic since any occurrence of $\wedge$ and $\rightarrow$ can be removed using the equivalences

$$\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$$

$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$$

   (a) Show that $\{\neg, \wedge\}$, $\{\neg, \rightarrow\}$ and $\{\rightarrow, \bot\}$ are adequate sets of connectives. ($\bot$ treated as a nullary connective).

   (b) Show that if $C \subseteq \{\neg, \wedge, \vee, \rightarrow, \bot\}$ is adequate, then $\neg \in C$ or $\bot \in C$.

Solution

We assume the set of all connectives is $\{\top, \bot, \neg, \wedge, \vee, \rightarrow\}$.
To show that a subset of these connectives is adequate, we need to prove that for every formula constructed from the original set of connectives there is an equivalent formula constructed from the connectives in the subset. This can be proven via structural induction over the formulae.

(a) To show that $\{\neg, \wedge\}$ is adequate, we proceed by structural induction.

Base Case:

We have $\top \equiv \neg(p \wedge \neg p)$, $\bot \equiv p \wedge \neg p$, and $p \equiv p$

Inductive Steps:

Say φ and ψ are formulae constructed from the original set of connectives that have equivalent formulae that use only $\neg$ and $\wedge$. Let these equivalent formulae be φ' and ψ' respectively.

We shall show that every formula constructed from φ and ψ have equivalent formulae that use only $\neg$ and $\wedge$.

- $\neg\varphi \equiv \neg\varphi'$
- $\varphi \wedge \psi \equiv \varphi' \wedge \psi'$
- $\varphi \vee \psi \equiv \neg(\neg\varphi' \wedge \neg\psi')$
- $\varphi \rightarrow \psi \equiv \neg(\varphi' \wedge \neg\psi')$

Since φ' and ψ' contain only $\neg$ and $\wedge$, the formulae on the RHS also contain only $\neg$ and ψ', and therefore the induction step is completed. Therefore, every formula can be rewritten into an equivalent formula that uses only $\neg$ and $\wedge$.

To show that $\{\neg, \to\}$ and $\{\to, \bot\}$ are also adequate sets we follow a similiar method. The equivalences that we will use are:

i. For $\{\neg, \to\}$:
   - $\top \equiv p \to p$
   - $\bot \equiv \neg(p \to p)$
   - $\varphi \wedge \psi \equiv \neg(\varphi \to \neg\psi)$
   - $\varphi \vee \psi \equiv (\neg\varphi \to \psi)$

ii. For $\{\to, \bot\}$:
   - $\top \equiv \bot \to p$
   - $\neg\varphi \equiv (\varphi \to \bot)$
   - $\varphi \wedge \psi \equiv ([\varphi \to (\psi \to \bot)] \to \bot)$
   - $\varphi \vee \psi \equiv ([\varphi \to \bot] \to \psi)$

(b) We shall show that for any $C \subseteq \{\top, \bot, \neg, \wedge, \vee, \to\}$, if $\bot \notin C$ and $\neg \notin C$, then C cannot be adequate (This is equivalent to proving that if C is adequate then it contains either $\neg$ or $\bot$).

Before this, notice that if $C \subseteq C' \subseteq \{\top, \bot, \neg, \wedge, \vee, \to\}$ and if C is adequate, then clearly C' is adequate too. So we shall prove the above statement by showing that $\{\top, \wedge, \vee, \to\}$ is not adequate. We shall do this by showing that no formula made out of these connectives is equivalent to $\bot$.

Lemma. *For any formula made out of* $\{\top, \wedge, \vee, \to\}$*, setting all the propositional variables to* 1 *always results in the overall formula having a truth value of* 1.

Note that this immediately shows that no formula constructed only out of $\{\wedge, \vee, \to\}$ can be equivalent to $\bot$.

We shall prove this lemma via structural induction.

Base Case:

If the formula just consists of a single propositional variable p or is just $\top$, the result clearly follows.

Inductive Step:

Say φ and ψ are formulae constructed only with $\{\top, \wedge, \vee, \to\}$ and setting all the propositional variables to 1 results in the truth values of both φ and ψ being 1. The formulae we can construct from φ and ψ are $\varphi \wedge \psi$, $\varphi \vee \psi$ and $\varphi \to \psi$. If we set all propositional variables to 1, then by the inductive hypothesis, the truth value of φ and ψ also become 1, and it can be seen that the truth values of the new formulae are also 1.

Therefore, by structural induction, the lemma is proven and with it we have proved that $\{\top, \wedge, \vee, \to\}$ is inadequate.

2. The binary connective `nand`, $F \downarrow G$, is defined by the truth table corresponding to $\neg(F \wedge G)$. Show that `nand` is complete - that is, it can express all binary Boolean connectives.

Solution

The following equivalences (which can be verified via truth tables) imply that $\downarrow$ (`nand`) is complete, ie it can express all binary connectives.

(a) $p \wedge q \equiv (p \downarrow q) \downarrow (p \downarrow q)$

(b) $p \vee q \equiv (p \downarrow p) \downarrow (q \downarrow q)$

(c) $p \rightarrow q \equiv p \downarrow (q \downarrow q)$

$\downarrow$ can also express the unary and nullary connectives ($\neg$ and $\bot$) respectively:

(a) $\neg p \equiv p \downarrow p$

(b) $\top \equiv p \downarrow (p \downarrow p)$

(c) $\bot \equiv (p \downarrow (p \downarrow p)) \downarrow (p \downarrow (p \downarrow p))$

We can show, via structural induction, that a subset of connectives can express all connectives (not just binary ones) iff it is adequate.

Follow-up Question. Similar to `nand` is another binary connective, known as `nor`, $F \uparrow G$ with truth table, same as that of $\neg(F \vee G)$. Can you show that `nor` is also complete?

3. The binary connective `xor`, $F \oplus G$ is defined by the truth table corresponding to $(\neg F \wedge G) \vee (F \wedge \neg G)$. Show that `xor` is not complete - that is, it cannot express all binary Boolean connectives.

Solution

The truth table for $\oplus$ is as follows:

p	q	$p \oplus q$
0	0	0
0	1	1
1	0	1
1	1	0

We will show by structural induction that any formula φ constructed from two propositional variables (say p, q) will have an even number of 1s in its truth table. This means a formula like $p \wedge q$ that has an odd number of 1s in its truth table cannot be expressed via $\oplus$, ie $\oplus$ is not complete[a].

Base Case:

$\top$, $\bot$, p and q all have an even number of 1s in their truth tables.

Inductive Step:

Say φ and ψ are formulae formed with $\oplus$. By the inductive hypothesis, they have an even number of 1s in their truth tables. Let's say φ and ψ are both 0 in i places, are

both 1 in j places, φ is 0 and ψ is 1 in k places and φ is 1 while ψ is 0 in l places. By the inductive hypothesis, $j+k$ and $j+l$ are even. Therefore, their sum, $2j+k+l$ is even, which means $k+l$ is also even. By truth table, $\varphi \oplus \psi$ is 1 in $k+l$ places, and therefore its truth table also has an even number of 1s.
Therefore, any formula formed this way has an even number of 1s, and hence $\oplus$ is not complete.

Follow-up Question. As homework, you can show that all formulae whose truth tables contain an even number of 1s can be expressed with $\{\top, \bot, \oplus\}$.

[a]We will prove something stronger, as our proof will also allow $\top$, $\bot$ to be used - we prove $\{\oplus, \top, \bot\}$ is neither complete nor adequate

4. If a contradiction can be derived from a set of formulae, then the set of formulae is said to be inconsistent. Otherwise, the set of formulae is consistent. Let $\mathcal{F}$ be a set of formulae. Show that $\mathcal{F}$ is consistent iff it is satisfiable.

Solution.

First, we will show that satisfiablility implies consistency, ie if $\mathcal{F}$ is satisfiable, then it is consistent ($\mathcal{F} \nvdash \bot$).
Assume, this was not the case and there exists an assignment α such that $\alpha \models \mathcal{F}$ and $\mathcal{F} \vdash \bot$. Since our Formal Proof System is sound[a], we have $\mathcal{F} \models \bot$, and therefore, we have an assignment α such that $\alpha \models \bot$, which is not possible! Therefore, satisfiablility implies consistency.
Now, we will show the reverse, ie if $\mathcal{F}$ is consistent ($\mathcal{F} \nvdash \bot$) then it must be satisfiable. Since our Formal Proof System is complete[b], we have $\mathcal{F} \not\models \bot$, ie there exists an assignment α such that $\alpha \models \mathcal{F}$ and $\alpha \not\models \bot$. The latter is always true, but the former shows that $\mathcal{F}$ is satisfiable.

[a]The proof of this can be found here
[b]The proof of this can be found here

5. Suppose $\mathcal{F}$ is an inconsistent set of formulae. For each $G \in \mathcal{F}$, let $\mathcal{F}_G$ be the set obtained by removing G from $\mathcal{F}$.
   (a) Prove that for any $G \in \mathcal{F}$, $\mathcal{F}_G \vdash \neg G$, using the previous question.
   (b) Prove this using a formal proof.

Solution

We know that $\mathcal{F}$ is inconsistent, ie $\mathcal{F} \vdash \bot$

(a) By the previous result, $\mathcal{F}$ must be unsatisfiable, ie for all assignments α, $\alpha \not\models \mathcal{F}$. Now, $\mathcal{F}$ can be written as $\mathcal{F}_G \cup \{G\}$, ie $\forall \alpha, \alpha \not\models \mathcal{F}_G \cup \{G\}$, which can be rewritten as $\forall \alpha, \neg(\alpha \models \mathcal{F}_G) \vee \alpha \not\models G$. This can be further rewritten as $\forall \alpha, \alpha \models \mathcal{F}_G \implies \alpha \models \neg G$, which is the definition of $\mathcal{F}_G \models \neg G$. Now, since the Formal Proof System is complete, this also means that $\mathcal{F}_G \vdash \neg G$.

(b) i. $\mathcal{F}_G \cup \{G\} \vdash \bot$ (Premise)
ii. $\mathcal{F}_G \vdash \neg G$ ($\neg$ introduction on (i))

6. Consider a formula φ which is of the form $C_1 \wedge C_2 \wedge \dots C_n$ where each clause C_i is of the form $(\top \to \alpha)$ or $(\alpha_1 \wedge \dots \alpha_n \to \beta)$ or $(\gamma \to \bot)$ where $\alpha, \alpha_i, \beta, \gamma$ are literals. A logician wishes to apply `HornSAT` to this formula φ by renaming negative literals (if any) with fresh positive literals. Thus, if any $\alpha, \alpha_i, \beta, \gamma$ was of the form $\neg p$, the logician will replace that $\neg p$ with a fresh variable p'. The logician claims that he can check satisfiability of φ correctly by applying `HornSAT` on the new formula (call it φ') in the following way: φ is satisfiable iff `HornSAT` concludes that φ' is satisfiable, and φ is unsatisfiable iff `HornSAT` concludes that φ' is unsatisfiable. Do you agree with the logician?

Solution

Consider $\varphi = (p \to \neg p) \wedge (\neg p \to p)$. This formula is unsatisfiable. However, after renaming, the formula becomes $(p \to p') \wedge (p' \to p)$, which is satisfiable. Therefore, the logician is incorrect.

Note that the renaming process introduces new variables which do not retain the original relationship with the existing variables. In this example, p' is treated as an independent variable after renaming and does not inherently represent the negation of the existing variable p. This lack of representation can lead to incorrect conclusions about the satisfiability of the original formula, as the new formula φ does not fully capture the logical dependencies present in φ.

7. We have seen in class that `HornSAT` has a polynomial satisfiability, while general `SAT` is NP-complete. Here is a reduction called "Hornification" proposed by a student from `SAT` to `HornSAT`. Given a formula φ in CNF, "hornify" each non-horn clause as follows.

- If we have a clause $C_1 = p \vee q \vee r$, then all occurrences of p, q are renamed to $\neg p'$ and $\neg q'$ where p', q' are fresh variables (In general, you could have chosen to rename all but one positive literal to Hornify). Clearly, this renaming can be done in polynomial time.
- Additionally, to respect the relationship between the original variables and their renamed counterparts, add a new Horn clause $p' \wedge p \to \bot$ (and similarly for q') whenever you rename p as $\neg p'$. This ensures that the new variables p' and q' behave correctly with respect to their original negated forms.

Call the new formula (on an expanded set of variables) as φ'. Since φ' is in `HornSAT`, we can check its satisfiability in polynomial time. Can we conclude that "Hornification" makes `SAT` to be in P?

Solution

Notice that both the steps of "Hornification" process can be done in polynomial time. So, if we take a CNF formula, "hornify" it and check its satisfiability using `HornSAT`, the algorithm runs in polynomial time, but the algorithm may not be correct, i.e, "Hornification" does not necessarily preserve the satisfiability of the original formula, φ.

Let $\varphi = (p \vee q \vee r) \wedge (\neg p) \wedge (\neg q) \wedge (\neg r)$. Clearly, φ is UNSAT. Now after "Hornification" we get, $\varphi' = (\neg p' \vee \neg q' \vee r) \wedge (\neg p) \wedge (\neg q) \wedge (\neg r) \wedge (p' \wedge p \to \bot) \wedge (q' \wedge q \to \bot)$. But, $p = 0, p' = 0, q = 0, q' = 0, r = 0$ is a satisfying assignment to φ'. Thus `HornSAT`(φ) will return SAT.

Notice that the main difficulty is that we are not able to rightly capture the relationship between original and renamed variables even by introducing new Horn clauses. In fact, if we rename p by $\neg p'$, we need the equivalence $p \leftrightarrow \neg p'$ to hold. But, this cannot be expressed as a Horn clause. To see why, we can rewrite this as $p \leftrightarrow \neg p' \equiv (p \to \neg p') \wedge (\neg p' \to p) \equiv (\neg p \vee \neg p') \wedge (p \vee p')$. Observe that the last clause, $p \vee p'$, can't be written as Horn clause. This means that "Hornification" cannot accurately represent the negation relationship, leading to potentially incorrect satisfiability results.

8. Using resolution, show that $P_1 \wedge P_2 \wedge P_3$ is a consequence of

$$F := (\neg P_1 \vee P_2) \wedge (\neg P_2 \vee P_3) \wedge (P_1 \vee \neg P_3) \wedge (P_1 \vee P_2 \vee P_3)\,.$$

Solution

The negation of $P_1 \wedge P_2 \wedge P_3$ is equivalent to $\neg P_1 \vee \neg P_2 \vee \neg P_3$. The following is a resolution refutation of $F \wedge (\neg P_1 \vee \neg P_2 \vee \neg P_3)$ in which clauses are represented as sets of literals:

$$\dfrac{\dfrac{\dfrac{\{P_1, P_2, P_3\} \qquad \{P_1, \neg P_3\}}{\{P_1, P_2\}} \qquad \{\neg P_1, P_2\}}{\{P_2\}} \qquad \dfrac{\dfrac{\dfrac{\{\neg P_1, \neg P_2, \neg P_3\} \qquad \{\neg P_1, P_2\}}{\{\neg P_1, \neg P_3\}} \qquad \{P_1, \neg P_3\}}{\{\neg P_3\}} \qquad \{\neg P_2, P_3\}}{\{\neg P_2\}}}{\emptyset}$$

9. Show that the satisfiability of any 2-CNF formula can be checked in polynomial time.

Solution

We will show that we can run resolution in time polynomial in the size of the 2-CNF. Say the CNF has k clauses in n variables. Note that performing resolution on any two clauses of the CNF will again give a clause with at most two literals. Therefore, during the entire resolution process, we will only be dealing with clauses containing at most two literals.

Now the total possible number of such clauses is $\binom{2n}{2} + 2n$ (since there are $2n$ possible literals). The total number of possible resolutions that can be done is therefore $\binom{\binom{2n}{2}+2n}{2}$ which is $\Theta(n^4)$. Therefore, we will perform $O(n^4)$ resolutions, and since the remaining steps of performing the resolution and checking whether a clause has already been seen can be done in polynomial time, our algorithm will terminate in polynomial time. A more sophisticated analysis may reveal tighter bounds.

10. Call a set of formulae minimal unsatisfiable iff it is unsatisfiable, but every proper subset is satisfiable. Show that there exist minimal unsatisfiable sets of formulae of size n for each $n \geq 1$.

Solution

It can be easily shown that the set

$$\Sigma_n = \{p_1, \dots p_n, \bigvee_{i=1}^{n} \neg p_i\}$$

is an example of a minimal unsatisfiable set for $n \geq 1$. The proof is left to you as an exercise.

Logic in CS Autumn 2024

Problem Sheet 3

S. Krishna

1. For each of the following conditions, give an example of an unsatisfiable set of formulae, Γ that meets the condition.

   (a) Each member of the set is—by itself—satisfiable.

   (b) For any two members γ_1 and γ_2 of Γ, the set $\{\gamma_1, \gamma_2\}$ is satisfiable.

   (c) For any three members γ_1, γ_2, and γ_3 of Γ, the set $\{\gamma_1, \gamma_2, \gamma_3\}$ is satisfiable.

Solution

(a) $\Gamma = \{p, \neg p\}$

(b) $\Gamma = \{p_1, p_2, \neg p_1 \vee \neg p_2\}$

(c) $\Gamma = \{p_1, p_2, p_3, \neg p_1 \vee \neg p_2 \vee \neg p_3\}$

2. Let α be a wff whose only connective symbols are $\wedge$, $\vee$, and $\neg$. Let α^* be the result of interchanging $\wedge$ and $\vee$ and replacing each propositional variable by its negation. Show that α^* is tautologically equivalent to $\neg\alpha$. Observe that this result is a generalization and a stronger form of De Morgan's laws, which deal with the negation of conjunctions and disjunctions.

Solution

We will prove this via structural induction.
Base Case:
Consider the simplest wff, where α is a sentence symbol p. Then, $\alpha^* = \neg p$, and hence, $\alpha^* \equiv \neg\alpha$.

Inductive Step:
Say ϕ, ψ are formulae which such that $\phi* \equiv \neg\phi$ and $\psi* \equiv \neg\psi$. Consider the following cases:

- $\alpha = \neg\phi$: Then, $\alpha^* = \neg\phi^*$ and by the inductive hypothesis, $\phi^* \equiv \neg\phi$.Therefore, $\alpha^* = \neg(\neg\phi) = \phi$, and thus $\alpha^* \equiv \neg\alpha$.

- $\alpha = \phi \wedge \psi$: Then, by interchanging $\vee$ and $\wedge$, $\alpha^* = \phi^* \vee \psi^*$, and by the inductive hypothesis, $\phi^* \equiv \neg\phi$ and $\psi^* \equiv \neg\psi$. Thus, $\alpha^* = \neg\phi \vee \neg\psi$ and by De Morgan's law, $\alpha^* \equiv \neg\alpha$.

- $\alpha = \phi \vee \psi$: Then, by interchanging $\vee$ and $\wedge$, $\alpha^* = \phi^* \wedge \psi^*$, and by the inductive hypothesis, $\phi^* \equiv \neg\phi$ and $\psi^* \equiv \neg\psi$. Thus, $\alpha^* = \neg\phi \wedge \neg\psi$ and by De Morgan's law, $\alpha^* \equiv \neg\alpha$.

3. Let $\mathcal{F}$ and $\mathcal{G}$ be two sets of formulae. We say $\mathcal{F} \equiv \mathcal{G}$ iff for any assignment α, $\alpha \models \mathcal{F}$ iff $\alpha \models \mathcal{G}$ ($\alpha \models \mathcal{F}$ iff $\alpha \models F_i$ for every $F_i \in \mathcal{F}$). Prove or disprove: For any $\mathcal{F}$ and $\mathcal{G}$, $\mathcal{F} \equiv \mathcal{G}$ iff

   (a) For each $G \in \mathcal{G}$, there exists $F \in \mathcal{F}$ such that $G \models F$, and

   (b) For each $F \in \mathcal{F}$, there exists $G \in \mathcal{G}$ such that $F \models G$,

Solution

Consider $\mathcal{F} = \{p\}$ and $\mathcal{G} = \{p, \top\}$, where p is an atomic proposition. Observe that $\mathcal{F} \equiv \mathcal{G}$. *(Why?)* Now, we shall show that (a) is false, hence giving a counterexample.

Take $G = \top$. Clearly, $\top \not\models p$ and hence there does not exist a $F \in \mathcal{F}$ such that $G \models F$

4. A set of sentences $\mathcal{F}$ is said to be closed under conjunction if for any F and G in $\mathcal{F}$, $F \land G$ is also in $\mathcal{F}$. Suppose $\mathcal{F}$ is closed under conjunction and is inconsistent ($\mathcal{F} \vdash \bot$). Prove that for any $G \in \mathcal{F}$, there exists $F \in \mathcal{F}$ such that $\{F\} \vdash \neg G$.

Solution

We are given that $\mathcal{F}$ is inconsistent ($\mathcal{F} \vdash \bot$). Let $\mathcal{F}' \subseteq \mathcal{F}$ be the finite set of formulae in $\mathcal{F}$ used in a proof deducing $\bot$. Thus, $\mathcal{F}'$ is a finite subset of $\mathcal{F}$ such that $\mathcal{F}' \vdash \bot$.

Consider the formula $F = \bigwedge_{F' \in \mathcal{F}'} F'$. Since $\mathcal{F}'$ is a finite subset of $\mathcal{F}$, F is also an element of $\mathcal{F}$ since $\mathcal{F}$ is closed under conjunction. We shall show that $\{F\} \vdash \bot$.

$$\begin{aligned}
\mathcal{F}' \vdash \bot &\implies \mathcal{F}' \models \bot \qquad [\text{ Soundness of Formal Proof System }] \\
&\implies \nexists\, \alpha \text{ such that } (\forall F' \in \mathcal{F}',\ \alpha \models F') \\
&\implies \nexists\, \alpha \text{ such that } \alpha \models F \qquad [\because F = \bigwedge_{F' \in \mathcal{F}'} F'] \\
&\implies \{F\} \models \bot \implies \{F\} \vdash \bot \quad [\text{ Completeness of Formal Proof System }]
\end{aligned}$$

Hence, we showed that there exists a formula $F \in \mathcal{F}$ such that $\{F\} \vdash \bot$ and hence for any formula $G \in \mathcal{F}$, $\{F\} \vdash \neg G$ since $\bot \vdash \neg G$.

5. Suppose $\models (F \to G)$ and F is satisfiable and G is not valid. Show that there exists a formula H such that the atomic propositions in H are in both F and G and $\models F \to H$ and $\models H \to G$.

Solution

We denote the list of propositional variables that occur only in F as $\vec{p}$, the list of variables common to F and G as $\vec{q}$, and those only in G as $\vec{r}$. Let n_p, n_q, n_r be the number of variables in $\vec{p}, \vec{q}, \vec{r}$, respectively.
Define H as follows:

$$H = \bigvee_{\vec{\alpha} \in \{\bot, \top\}^{n_p}} F[\, \vec{p} = \vec{\alpha} \,].$$

where, $F[\, \vec{p} = \vec{\alpha} \,]$ denotes the formula obtained by replacing all occurrences of p_i by α_i, and then simplifying the resultant formula. By simplification, we mean the obvious ones like $\alpha \wedge \top = \alpha$, $\alpha \vee \top = \top$, $\alpha \wedge \neg\top = \bot$, $\alpha \vee \neg\top = \alpha$ for all sub-formulas α of F. Similarly for $\bot$.

Notice that H contains variables only in $\vec{q}$.
Now we need to show that (a) $\models F \rightarrow H$ and (b) $\models H \rightarrow G$. In the following discussion, α_p (resp. α_q and α_r) denote an assignment of variables in $\vec{p}$ (resp. $\vec{q}$ and $\vec{r}$). For this we will state the following lemma first.

Lemma. $(-, \alpha_q, -) \models H \iff \exists\, \alpha_p$ such that $(\alpha_p, \alpha_q, -) \models F$.
Here, $(-, \alpha_q, -)$ represents an assignment of all variables such that the variables in $\vec{q}$ are assigned α_q, while variables in $\vec{p}$ and $\vec{r}$ can take any truth values. Thus, the lemma follows from the definition of H and the fact that $[\![H]\!]$ does not depend on assignment of variables in $\vec{p}$.

(a) $\models F \rightarrow H$:

$$\begin{aligned} (\alpha_p, \alpha_q, -) \models F &\implies \exists\, \alpha_p \text{ such that } (\alpha_p, \alpha_q, -) \models F \\ &\implies (-, \alpha_q, -) \models H \quad [\text{ By } (\Longleftarrow) \text{ of Lemma }] \end{aligned}$$

(b) $\models H \rightarrow G$:

$$\begin{aligned} (-, \alpha_q, -) \models H &\implies \exists\, \alpha_p \text{ such that } (\alpha_p, \alpha_q, -) \models F \quad [\text{ By } (\Longrightarrow) \text{ of Lemma }] \\ &\implies \exists\, \alpha_p \left(\forall\, \alpha_r,\ (\alpha_p, \alpha_q, \alpha_r) \models F\right) \quad [\, [\![F]\!] \text{ does not depend on } \alpha_r \,] \\ &\implies \exists\, \alpha_p \left(\forall\, \alpha_r,\ (\alpha_p, \alpha_q, \alpha_r) \models G\right) \quad [\text{ Since } \models F \rightarrow G \,] \\ &\implies \forall \alpha_r,\ (-, \alpha_q, \alpha_r) \models G \quad [\, [\![G]\!] \text{ does not depend on } \alpha_p \,] \end{aligned}$$

This result is popularly known as *Craig's Interpolation Theorem*, as applied to propositional logic. The formula H is known as an *interpolant* of F and G.

6. Consider the parity function, $\mathsf{PARITY} : \{0,1\}^n \rightarrow \{0,1\}$, where PARITY evaluates to 1 iff an odd number of inputs is 1. In all of the CNFs below, we assume that each clause contains any variable at most once, i.e. no clause contains expressions of the form $p \wedge \neg p$ or $p \vee \neg p$. Furthermore, all clauses are assumed to be distinct.

(a) Prove that any CNF representation of PARITY must have n literals (from distinct variables) in every clause.

(b) Prove that any CNF representation of PARITY must have at least 2^{n-1} clauses.

Solution

Let PARITY $:= \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} \ell_{ij}$ be the CNF representation of PARITY, where n_i is the number of literals in the $i-$th clause, and m is the number of clauses. We want to prove that $n_i = n$ for every i, and $m \geq 2^n - 1$.

(a) Suppose $n_i < n$ for some i. Negate the entire formula to convert the CNF into a DNF. Now, choose an assignment of literals in the $i-$th cube (of the DNF) such that the cube, and hence the whole DNF formula, evaluates to 1. Now, consider a variable v, which doesn't appear in the $i-$th cube, and flip its value. The LHS then becomes 0. However, the $i-$th clause stays 1, leading to a contradiction.

(b) Once again, consider the DNF. Note that a conjunction of n literals is satisfied by a unique assignment of variables, and thus, the clauses in the DNF actually encode the assignments that satisfy the formula. Since PARITY is satisfied by 2^{n-1} clauses, we're done.

7. Using resolution, or otherwise, show that there is a polynomial-time algorithm to decide satisfiability of those CNF formulas F in which each propositional variable occurs at most twice. Justify your answer. Note that this question is not the same as 2-SAT.

Solution

If a variable occurs only positively or only negatively in F, then we can delete the clauses containing that variable without affecting the satisfiability of F. Thus, we can assume that all variables have one negative occurrence and one positive occurrence.

Suppose F has clauses C_1, C_2 such that $p \in C_1$ and $\neg p \in C_2$. Let $R = (C_1 \backslash \{p\}) \cup (C_2 \backslash \{\neg p\})$ be a resolvent of C_1, C_2. We claim that F and $F' = (F \backslash \{C_1, C_2\}) \cup \{R\}$ are equisatisfiable. It follows immediately from resolution that a valuation that satisfies F also satisfies F'. Thus, it suffices to show that an assignment α that satisfies F' can be extended to an assignment that satisfies F. Since such an assignment satisfies R, it satisfies either $C_1 \backslash \{p\}$ or $C_2 \backslash \{\neg p\}$. In the first case, the assignment $\alpha[p \leftarrow 0]$ satisfies F, and in the second case, the assignment $\alpha[p \leftarrow 1]$ satisfies F.

In summary, we can eliminate each variable from F without affecting satisfiability and without increasing the size of the overall formula. It follows that satisfiability can be decided in polynomial time.

8. Say that a set Σ_1 of wffs is equivalent to a set Σ_2 of wffs iff for any wff α, we have $\Sigma_1 \models \alpha$ iff $\Sigma_2 \models \alpha$. A set Σ is independent iff no member of Σ is tautologically implied by the remaining members in Σ. Show that a finite set of wffs has an independent equivalent subset by describing an algorithm to compute this independent equivalent subset. Prove that your algorithm returns a subset that is independent and equivalent.

Algorithm 1 Algorithm to compute an independent equivalent subset

```
1: $\Sigma' \leftarrow \Sigma$
2: while there exists $\sigma \in \Sigma'$ such that $\Sigma' \setminus \{\sigma\} \models \sigma$ do
3:     $\Sigma' \leftarrow \Sigma' \setminus \{\sigma\}$
4: end while
5: return $\Sigma'$
```

Solution

The key idea is to keep removing the formulae in Σ, which are tautologically implied by others, till there aren't any. This is given in Algorithm 1.

Notice that since Σ is finite, the algorithm terminates after finite number of iterations. Also, the algorithms terminates only when there does not exist a $\sigma \in \Sigma'$ such that $\Sigma \setminus \{\sigma\} \models \sigma$, i.e, the set Σ' returned at the end will be independent. Clearly $\Sigma' \subseteq \Sigma$. We are only left to show that Σ' is equivalent to Σ.

Let the algorithm terminate after n iterations. Let $\Sigma_0, \Sigma_1, \ldots, \Sigma_n$ be the sets representing Σ' at the end of each iteration. Clearly, $\Sigma_0 = \Sigma$, and Σ_n will be returned at the end. We will inductively show that at each iteration, i, Σ_i is equivalent to Σ.

Base case. $i = 0$. $\Sigma_i = \Sigma_0$ is clearly equivalent to Σ.
Induction Step. Assume that Σ_{i-1} is equivalent to Σ. We will now show that Σ_i is also equivalent to Σ. In our algorithm, we should have removed an element from Σ_{i-1} to obtain Σ_i. Let this element be σ_i. We have removed σ_i from Σ_{i-1} because, $\Sigma_i \models \sigma_i$. Now, let α be any wff. If $\Sigma_i \models \alpha$, then since $\Sigma_i \subseteq \Sigma$, $\Sigma \models \alpha$ (*Why?*).
Now, observe that since, $\Sigma_{i-1} \models \sigma_i$, the set of all assignments which make all formulae in Σ_{i-1} `true` is same as that which make all formulae in $\Sigma_i = \Sigma_{i-1} \setminus \{\sigma_i\}$ `true`. Hence, since Σ is equivalent to Σ_{i-1} it follows that $\Sigma \models \alpha \implies \Sigma_{i-1} \models \alpha \implies \Sigma_i \models \alpha$.
Thus, Σ_i is equivalent to Σ. Thus by induction Σ_n is equivalent to Σ, as we needed, and thus our algorithm returns an independent equivalent subset of Σ.

Follow-up Question. Is the output from our algorithm unique?

Logic in CS Autumn 2024

Problem Sheet 4

S. Krishna

The DPLL algorithm, used in this tutorial, consists of the following steps, which are done before each decision step.

1. Unit propogation (UP): If a clause contains only a single unassigned literal (unit clause), the variable is assigned the necessary value to make this literal true.
2. Pure literal elimination (UP): If a propositional variable occurs with only one polarity in the formula, it is called pure. The variable is assigned the necessary value to make all the clauses containing this literal true.

1. If DPLL always chooses to assign 0 to a decision variable before assigning 1 (if needed) to a variable, then show that DPLL will never need to backtrack when given a Horn formula encoded in CNF as input.

Solution

In HornSAT algorithm, we mark the occurences of a variable v on the RHS of a Horn clause C (written as an implication, $C = (p_1 \wedge \cdots \wedge p_n \to v))$ true, when all the variables on the LHS is marked true. This is equivalent to a unit propogation step in the DPLL algorithm, when the unit clause is the positive literal v.

Since, the HornSAT algorithm just tells to mark occurrences of variables true, as mentioned above, until the step cannot be done anymore, and mark all the other variables false to get a satisfying assignment. Else, you would get a clause in which all the variables in the LHS are marked true and the RHS is $\perp$, making the formula unsatisfiable.

Hence, by the correctness of HornSAT algorithm, we can conclude that on applying only unit propogation steps on a Horn formula H until no more can be done, H is unsatisfiable $\iff$ The partial assignment reduced one of the clauses to $\perp$. Therefore, after the initial unit propogation steps, assigning 0 to any decision variable will reduce it to an equisatisfiable Horn formula (since assigning all the variables false will give a satisfying assignment, by HornSAT).

2. Let's try to solve the HornSAT problem using DPLL. We can convert each Horn clause into a CNF clause by simply rewriting $(a \to b)$ as $(\neg a \vee b)$, which preserves the semantics of the formula. The resultant formula is in CNF.

   Suppose our version of DPLL always assigns 1 to a decision variable before assigning 0 (if needed). How many backtrackings are needed if we run this version of DPLL on the (CNFised version of) the following Horn formulae, assuming DPLL always chooses the unassigned variable with the smallest subscript when choosing a decision variable?

(a)

$$\left(\left(\bigwedge_{i=0}^{n-1} x_i\right) \to x_n\right) \wedge \bigwedge_{i=0}^{n-1}(x_n \to x_i) \wedge \left(\left(\bigwedge_{i=0}^{n} x_i\right) \to \bot\right)$$

(b)

$$\bigwedge_{i=0}^{n-1}((x_i \to x_{n+i}) \wedge (x_{n+i} \to x_i) \wedge (x_i \wedge x_{n+i} \to \bot))$$

Solution

In part (a), no unit clauses or pure literals are obtained until all of x_0 through x_{n-1} are assigned the value 1, one at a time. Once x_{n-1} is assigned 1, UP leads to a conflict—x_n must be assigned both 1 and 0 by UP. This causes a backtrack, which ends up setting x_{n-1} to 0. Once this happens, UP assigns the value 0 to x_n, resulting in the partial assignment $x_{n-1} = x_n = 0$, which satisfies all clauses. Therefore, there is exactly one backtrack.

In part (b), every time a variable x_i for $0 \le i \le n-1$ is assigned 1, UP causes x_{n+i} to be in conflict (must be assigned both 0 and 1). This induces a backtrack that sets x_i to 0, followed by UP setting x_{n+i} to 0. The DPLL algorithm will then choose x_{i+1} as the next decision variable, assign it 1, and the above process repeats. So, in this case, DPLL will incur n backtracks, one for each of $x_0, \ldots, x_{n-1}$.

3. Let P, Q, R be propositional variables. Convert the formula

$$\neg(P \vee (\neg Q \wedge R)) \to (\neg P \wedge (Q \vee \neg R))$$

to an equisatisfiable Conjunctive Normal Form (CNF) formula using Tseitin encoding.

Solution

Consider the parse of the given formula in Figure 1. We introduce 6 fresh variables $t_1, \ldots, t_6$ to replace for the sub-formulae of the given formula.
Then, the final formula after Tseitin encoding will be:

$$\begin{aligned}(t_1 \leftrightarrow (\neg Q \wedge R)) \wedge (t_2 \leftrightarrow P \vee t_1) \wedge (t_3 \leftrightarrow \neg t_2) \\ \wedge (t_4 \leftrightarrow (Q \vee \neg R)) \wedge (t_5 \leftrightarrow (\neg P \wedge t_4) \\ \wedge (t_6 \leftrightarrow (t_3 \to t_5)) \wedge (t_6)\end{aligned}$$

Homework: Convert the above formula to a CNF using common rules of semantic equivalences.

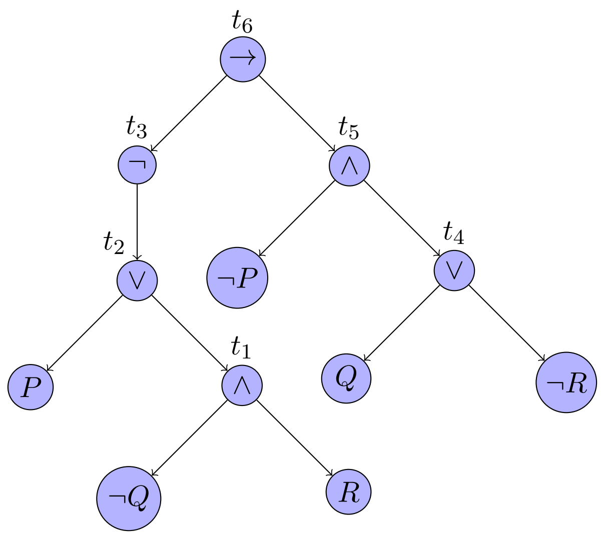


Figure 1: Tseitin Encoding

4. Consider the following CNF formula:

$$\begin{aligned}(\neg p_1 \vee p_2 \vee p_3) &\wedge (\neg p_1 \vee p_3 \vee p_9)\\ &\wedge (\neg p_2 \vee \neg p_3 \vee p_4)\\ &\wedge (\neg p_4 \vee p_5)\\ &\wedge (\neg p_4 \vee p_6 \vee \neg p_8)\\ &\wedge (\neg p_5 \vee \neg p_6)\\ &\wedge (p_7 \vee p_1 \vee \neg p_{10})\\ &\wedge (p_1 \vee p_8)\\ &\wedge (\neg p_7 \vee \neg p_8)\end{aligned}$$

(a) Check if this CNF formula is satisfiable using the DPLL method.

(b) Assume $p_9 = \bot$ and $p_{10} = \top$. Now, check if we can find a solution for the above formula using the DPLL method.

Solution

(a) Notice that there is no unit clause (clause with a single literal) in the given CNF formula. But p_9 and $\neg p_{10}$ are pure literals (literals which appear either positive or negative throughout). So we begin with Pure Literal Elimination (PLE) and proceed as follows:

Step No.	Variable Assigned	Value Assigned	Reason
1	p_9	1	PLE
2	p_{10}	0	PLE
3	p_7	0	PLE
4	p_1	0	Decision
5	p_8	1	UP on $(p_1 \lor p_8)$
6	p_2	0	PLE
7	p_4	0	PLE
8	p_5	0	PLE

(b) If $p_9 = \bot$ and $p_{10} = \top$, the formula after simplification becomes:

$$\begin{aligned}(\neg p_1 \lor p_2 \lor p_3) &\land (\neg p_1 \lor p_3)\\ &\land (\neg p_2 \lor \neg p_3 \lor p_4)\\ &\land (\neg p_4 \lor p_5)\\ &\land (\neg p_4 \lor p_6 \lor \neg p_8)\\ &\land (\neg p_5 \lor \neg p_6)\\ &\land (p_7 \lor p_1)\\ &\land (p_1 \lor p_8)\\ &\land (\neg p_7 \lor \neg p_8)\end{aligned}$$

Step No.	Variable Assigned	Value Assigned	Reason
1	p_1	1	Decision
2	p_7	0	PLE
3	p_8	0	PLE
4	p_3	1	UP on $(\neg p_1 \lor p_3)$
5	p_2	0	PLE
6	p_4	0	PLE
7	p_5	0	PLE

5. Discuss the best-case and worst-case time complexities of the DPLL algorithm. Provide a representative example for each scenario.

Solution

The best case time complexity of DPLL is $\mathcal{O}(n)$, because the algorithm needs to read through the formula atleast once to check for a pure literal. An example of a formula on which DPLL terminates within $\mathcal{O}(n)$ time is $p_1 \land p_2 \land \cdots \land p_n$.

The worst case time complexity of $\mathcal{O}(2^n)$ and it is attained by any unsatisfiable formula which does not result in a conflict during initial steps of Unit propagation and Pure literal elimination (*Why?*).

Logic in CS Autumn 2024

Problem Sheet 5

S. Krishna

1. Consider the formula $\varphi = \forall x \exists y R(x,y) \wedge \exists y \forall x \neg R(x,y)$. Show that φ is satisfiable over a structure whose universe is infinite and countable.

Solution

Let $\mathcal{N}$ be the structure having as universe the set $\mathbb{N}$ of natural numbers and which interprets $R(x,y)$ as the successor relation, i.e, $R(x,y) = \{(n, n+1) \mid n \in \mathbb{N}\}$. Observe that $\mathcal{N} \models \varphi$, because every $x \in \mathbb{N}$ has a successor in $\mathbb{N}$, making $\forall x \exists y R(x,y)$ true. But $1 \in \mathbb{N}$ is not the predecessor of any number in $\mathbb{N}$[a], making $\exists y \forall x \neg R(x,y)$ true.

Follow-up Question. What property should R satisfy so that φ is not satisfiable over any structure with infinite and countable universe?

[a]Assuming the set of natural numbers start with 1 and not 0

2. Let τ be a signature consisting of a binary relation P and a unary relation F. Let $\mathcal{F}$ be a structure consisting of a universe of people, $P(x,y)$ is interpreted on $\mathcal{F}$ as "x is a parent of y" and $F(x)$ is interpreted as "x is female". Given the τ-structure $\mathcal{F}$,

   (a) Define a formula $\varphi_B(x,y)$ which says x is a brother of y
   (b) Define a formula $\varphi_A(x,y)$ which says x is an aunt of y
   (c) Define a formula $\varphi_C(x,y)$ which says x and y are cousins
   (d) Define a formula $\varphi_O(x)$ which says x is an only child
   (e) Give an example of a family relationship that cannot be defined by a formula

Solution

(a) Assuming a brother is a distinct non-female who shares a parent in common with you, we have:

$$\varphi_B(x,y) = \neg(x = y) \wedge \neg F(x) \wedge \exists z[P(z,x) \wedge P(z,y)]$$

(b) Assuming an aunt is a female who shares a parent in common with a parent of yours, we have:

$$\varphi_A(x,y) = F(x) \wedge \exists z[P(z,y) \wedge \exists w(P(w,z) \wedge P(w,x))]$$

(c) Assuming cousins are distinct people who have distinct parents who have a parent in common, we can write:

$$\varphi_C(x,y) = \exists a \exists b \exists c[\neg(a=b) \wedge P(a,x) \wedge P(b,y) \wedge P(c,a) \wedge P(c,b)] \wedge \neg(x=y)$$

(d) Assuming an only child is a person whose parents have no other children, we have:

$$\varphi_O(x) = \forall y \forall z[P(z,x) \wedge P(z,y) \implies x=y]$$

(e) Since the only relationships modeled here are parent-child relationships and whether a person is female, a relationship such as marriage, i.e., $\varphi_M(x,y)$, that says x is married to y cannot be defined.

3. Consider the signature τ that has the binary functions $+, \times$. Let $\mathcal{N}$ be the structure over τ having as universe the set $\mathbb{N}$ of natural numbers and which interprets $+, \times$ in the usual way. Construct FO formulae $\mathsf{Zero}(x), \mathsf{One}(x), \mathsf{Even}(x), \mathsf{Odd}(x)$ and $\mathsf{Prime}(x)$ using τ such that

- For any $a \in \mathbb{N}$, $\mathcal{N} \models \mathsf{Zero}(a)$ iff a is zero.
- For any $a \in \mathbb{N}$, $\mathcal{N} \models \mathsf{One}(a)$ iff a is one.
- For any $a \in \mathbb{N}$, $\mathcal{N} \models \mathsf{Even}(a)$ iff a is even.
- For any $a \in \mathbb{N}$, $\mathcal{N} \models \mathsf{Odd}(a)$ iff a is odd.
- For any $a \in \mathbb{N}$, $\mathcal{N} \models \mathsf{Prime}(a)$ iff a is prime.

Goldbach's conjecture says that every even integer greater than 2 is the sum of two primes. Whether or not this is true is an open question in number theory. State Goldbach's conjecture as a FO-sentence over τ.

Solution

- $\mathsf{Zero}(a) = \forall x \ (a+x) = x$
- $\mathsf{One}(a) = \forall x \ (a \times x) = a$
- $\mathsf{Even}(a) = \exists x \ (x+x) = a$
- $\mathsf{Odd}(a) = \neg\mathsf{Even}(a)$
- $\mathsf{Prime}(a) = \neg(\exists x \exists y \ (\neg\mathsf{One}(x) \wedge \neg\mathsf{One}(y) \wedge (x \times y) = a)) \wedge \neg\mathsf{One}(a)$

We can also define a FO formula $\mathsf{Two}(a)$ such that for any $a \in \mathbb{N}$, $\mathcal{N} \models \mathsf{Two}(a)$ iff a is two, as $\mathsf{Two}(a) = \exists x \ (\mathsf{One}(x) \wedge (x+x) = a)$.
Using the above formulae, we can state Goldbach's conjecture as:

$$\mathsf{Goldbach} := \forall x \ (\neg\mathsf{Zero}(x) \wedge \neg\mathsf{Two}(x) \wedge \mathsf{Even}(x) \to \exists y \exists z \ \mathsf{Prime}(y) \wedge \mathsf{Prime}(z) \wedge (y+z) = x)$$

4. A group is a structure $(G, +, 0)$ where G is a set, $0 \in G$ is a special element called the identity and $+ : G \times G \to G$ is a binary operation such that

   (a) The operation $+$ is associative

   (b) The constant 0 is a right-identity for the operation $+$

   (c) Every element in G has a right inverse: for each $x \in G$, we can find $y \in G$ such that $x + y = 0$

   (d) For any three elements $x, y, z \in G$, if $x + z = y + z$, then $x = y$

   Using a signature $\tau = (c, \mathsf{op})$ where c is a constant and op is a binary function symbol write (a)-(d) in FO.

Solution

We write the following formulae for each one of the above specifications:

(a) $\varphi_a = \forall x \forall y \forall z\ [\mathsf{op}(\mathsf{op}(x, y), z) = \mathsf{op}(x, \mathsf{op}(y, z))]$

(b) $\varphi_b = \forall x\ (\mathsf{op}(x, c) = x)$

(c) $\varphi_c = \forall x \exists y\ (\mathsf{op}(x, y) = c)$

(d) $\varphi_d = \forall x \forall y \forall z\ [(\mathsf{op}(x, z) = \mathsf{op}(y, z)) \to x = y]$

5. Let τ be a signature consisting of the binary function symbol $+$ and a constant 0. We denote by $x + y$ the function $+(x, y)$. Consider the following sentences:

$$\varphi_1 := \forall x \forall y \forall z\ [(x + (y + z)) = ((x + y) + z)]$$

$$\varphi_2 := \forall x\ [(x + 0) = x \wedge (0 + x) = x]$$

$$\varphi_3 := \forall x\ [\exists y\ (x + y = 0) \wedge \exists z (z + x) = 0]$$

   Let ψ be the conjunction of the three sentences.

   (a) Show that ψ is satisfiable by exhibiting a τ-structure.

   (b) Show that ψ is not valid.

   (c) Let α be the sentence $\forall x \forall y\ ((x + y) = (y + x))$. Does α follow as a consequence of ψ? That is, is it the case that $\psi \to \alpha$?

   (d) Show that ψ is not equivalent to any of $\varphi_1 \wedge \varphi_2$, $\varphi_2 \wedge \varphi_3$ and $\varphi_1 \wedge \varphi_3$.

Solution

(a) A structure $\mathcal{A}$ that satisfies ψ can be one with $u(\mathcal{A}) = \mathbb{Z}$, $0_{\mathcal{A}} = 0$ and $+_{\mathcal{A}} = +_{\mathbb{Z}}$.

(b) A structure that does not satisfy ψ can be one where $u(\mathcal{A}) = \mathbb{Z}$, $0_{\mathcal{A}} = 0$ and $+_{\mathcal{A}}(x, y) = 2x + 3y$. You can in fact verify that none of φ_1, φ_2, and φ_3 are satisfied.

(c) An example of a structure $\mathcal{A}$ that does not satisfy $\psi \Rightarrow \alpha$ is one where $u(\mathcal{A}) = \{M \in \mathbb{R}^{n\times n} : |M| \neq 0\}$ (the set of invertible real-valued $n \times n$ matrices), $0_{\mathcal{A}} = I_n$, and $+_{\mathcal{A}}(A, B) = AB$. It can be verified that this structure satisfies ψ but not α (i.e., it is not commutative).

(d) i. A structure $\mathcal{A}$ satisfying $\varphi_1 \wedge \varphi_2$ but not ψ is one where $u(\mathcal{A}) = \mathbb{Z}$, $0_{\mathcal{A}} = 1$, and $+_{\mathcal{A}}(x, y) = xy$.

ii. A structure $\mathcal{A}$ satisfying $\varphi_2 \wedge \varphi_3$ but not ψ is one where $u(\mathcal{A}) = \mathbb{N}$, $0_{\mathcal{A}} = 0$, and $+_{\mathcal{A}}(x, y) = |x - y|$.

iii. A structure $\mathcal{A}$ satisfying $\varphi_1 \wedge \varphi_3$ but not ψ is one where $u(\mathcal{A}) = \mathbb{Z}$, $0_{\mathcal{A}} = 1$, and $+_{\mathcal{A}}(x, y) = x + y$.

6. Explain the difference between the first order prefixes $\exists x\forall y\exists z$ and $\forall x\exists y\forall z$.

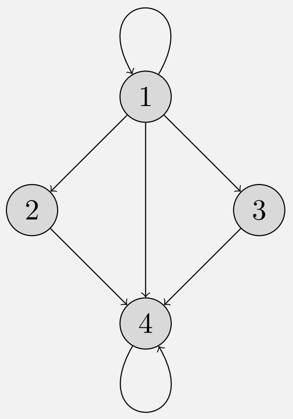
Solution

The first one states that there exists some x in the universe such that for every y in the universe there is some z in the universe (which may depend on y) such that the statement holds.
The second one states that for every x in the universe, there is some y in the universe (which may depend on x) such that for every z in the universe the statement holds.
The difference between the two is illustrated with an example in the following question.

7. Show that the sentences $\forall x\exists y\forall z\ (E(x, y) \wedge E(x, z) \wedge E(y, z))$ and $\exists x\forall y\exists z\ (E(x, y) \wedge E(x, z) \wedge E(y, z))$ are not equivalent by exhibiting a graph which satisfies one but not both of the sentences.

Solution

The first sentence is actually satisfied only by the complete graph K_n, where $E_{\mathcal{A}} = u(\mathcal{A}) \times u(\mathcal{A})$. To see this, assume there is some $(a, b) \notin E_{\mathcal{A}}$. If $\forall x\exists y\forall z\,[E(x, y) \wedge E(x, z) \wedge E(y, z)]$, then we can choose $x = a$, and then, for any y that we choose, choosing $z = b$ will cause $E(x, y) \wedge E(x, z) \wedge E(y, z)$ to not be satisfied, since $E(a, b)$ is not satisfied. This structure will also clearly satisfy the second sentence.
For an example of a structure that satisfies the second sentence but not the first, consider the following graph:



$$u(\mathcal{A}) = \{1,2,3,4\}, \quad E_{\mathcal{A}} = \{(1,1),(1,2),(1,3),(1,4),(2,4),(3,4),(4,4)\}$$

8. For each $n \in \mathbb{N}$, $\exists^{\geq n}$ denotes a counting quantifier. Intuitively, $\exists^{\geq n}$ means that "there exist atleast n such that". FO with counting quantifiers is the logic obtained by adding these quantifiers (for each $n \in \mathbb{N}$) to the fixed symbols of FO. The syntax and semantics are as follows:

   Syntax : For any formula φ of FO with counting quantifiers, $\exists^{\geq n}x\ \varphi$ is also a formula.

   Semantics : $\mathcal{A} \models \exists^{\geq n}x\ \varphi$ iff $\mathcal{A} \models \varphi(a_i)$ for each of n distinct elements $a_1, a_2, \ldots, a_n$ from the universe $u(\mathcal{A})$.

   (a) Using counting quantifiers, define a sentence φ_{45} such that $\mathcal{A} \models \varphi_{45}$ iff $|u(\mathcal{A})| = 45$.

   (b) Define a FO sentence φ (not using counting quantifiers) that is equivalent to the sentence $\exists^{\geq n}x\ (x = x)$.

Solution

(a)
$$\exists^{\geq k}x(x = x) \wedge \neg\exists^{\geq k+1}x(x = x)$$
is an FOL sentence with counting quantifiers that is true iff $|u(\mathcal{A})| = k$.

(b)
$$\exists x_1 \cdots \exists x_n \bigwedge_{1 \leq i < j \leq n} \neg(x_i = x_j)$$
is an FOL sentence equivalent to
$$\exists^{\geq n}x(x = x)$$

5

9. Write an FO formula that will evaluate to true only over a structure that has atleast n elements and atmost m elements.

Solution

Using the counting quantifiers we discussed earlier, such a sentence would be $\exists^{\geq n} x(x = x) \wedge \neg \exists^{\geq m+1} x(x = x)$. Removing the counting quantifiers, we get the sentence:

$$\left(\exists x_1 \cdots \exists x_n \bigwedge_{1 \leq i < j \leq n} \neg(x_i = x_j)\right) \wedge \neg \left(\exists x_1 \cdots \exists x_{m+1} \bigwedge_{1 \leq i < j \leq m+1} \neg(x_i = x_j)\right)$$

One can show that this sentence is equivalent to:

$$\left(\exists x_1 \cdots \exists x_n \bigwedge_{1 \leq i < j \leq n} \neg(x_i = x_j)\right) \wedge \left(\forall x_1 \cdots \forall x_{m+1} \bigvee_{1 \leq i < j \leq m+1} (x_i = x_j)\right)$$

Logic in CS Autumn 2024

Problem Sheet 6

S. Krishna

1. Translate the following formula to rectified form, then to prenex form, and finally to Skolem form:
$$\forall z\, \exists y\, (Q(x, g(y), z) \vee \neg\forall x\, P(x)) \wedge \neg\forall z\, \exists x\, \neg R(f(x, z), z)\,.$$

Solution

Recall that a formula is rectified if no variable appears both bound and free and if each bound variable has exactly one binding occurrence. An equivalent rectified form of the given formula is obtained by renaming some of the bound variables as follows:

$$\forall z\, \exists y\, (Q(x, g(y), z) \vee \neg\forall u\, P(u)) \wedge \neg\forall w\, \exists v\, \neg R(f(v, w), w)\,.$$

To obtain an equivalent prenex form, first push the negations inward:

$$\forall z\, \exists y\, (Q(x, g(y), z) \vee \exists u\, \neg P(u)) \wedge \exists w\, \forall v\, R(f(v, w), w)\,.$$

Then bring the quantifiers to the front (using the equivalences from Lecture 13):

$$\forall z\, \exists y\, \exists u\, \exists w\, \forall v\, ((Q(x, g(y), z) \vee \neg P(u)) \wedge R(f(v, w), w))\,.$$

Finally we obtain an equisatisfiable Skolem form with new unary function symbols f_1, f_2, f_3:

$$\forall z\, \forall v\, ((Q(x, g(f_1(z)), z) \vee \neg P(f_2(z))) \wedge R(f(v, f_3(z)), f_3(z)))$$

2. Are the following claims correct? Justify your answers.

   (a) For any formula F and term t, if F is valid then $F[t/x]$ is valid.

   (b) $\exists x\, (P(x) \to \forall y\, P(y))$ is valid.

   (c) For any formula F and constant symbol c, if $F[c/x]$ is valid and c does not appear in F then $\forall x\, F$ is valid.

Solution

(a) The claim is incorrect. Take the formula F to be $P(x) \to \forall y\, P(x)$. Then F is valid but $F[y/x]$ is the formula $P(y) \to \forall y\, P(y)$, which is not valid.

(b) The claim is correct. Let $\mathcal{A}$ be an arbitrary structure. We consider two cases. The first case is that $\mathcal{A} \models \forall y\, P(y)$. Then $\mathcal{A} \models P(x) \to \forall y\, P(y)$ and hence $\mathcal{A} \models$

$\exists x\,(P(x) \to \forall y\,P(y))$. The second case is that $\mathcal{A} \not\models \forall y\,P(y)$. By definition this means $\mathcal{A}_{[y\mapsto a]} \not\models P(y)$ for some $a \in U_{\mathcal{A}}$. It follows that $a \notin P_{\mathcal{A}}$ and so $\mathcal{A}_{[x\mapsto a]} \not\models P(x)$. This means $\mathcal{A}_{[x\mapsto a]} \models P(x) \to \forall y\,P(y)$ and hence $\mathcal{A} \models \exists x\,(P(x) \to \forall y\,P(y))$.

(c) The claim is correct. We have to show that $\mathcal{A} \models_{\alpha[x\mapsto a]} F$ for any structure $\mathcal{A}$ and all $a \in U^{\mathcal{A}}$.

Assume $F[c/x]$ is valid. Thus, all $\mathcal{A}$ and α are such that $\mathcal{A} \models_\alpha F[c/x]$.

Given such a $\mathcal{A}$ and α, let α' be such that $\alpha'(c) = \alpha(x)$ and let α' agree with α on all others. Then we have

$$\begin{aligned}
\mathcal{A} \models_\alpha F \text{ iff } & \mathcal{A} \models_{\alpha'} F && \text{(since } c \text{ is not mentioned in } F) \\
\text{iff } & \mathcal{A} \models_{\alpha[x\mapsto\alpha'(c)]} F && \text{(identical assignments)} \\
\text{iff } & \mathcal{A} \models_\alpha F[c/x] && \text{(Translation Lemma)}
\end{aligned}$$

Since $F[c/x]$ is valid, all assignments assigning any value of $U^{\mathcal{A}}$ to c satisfies $F[c/x]$. Indeed, if we now assign all those possible values to x in F by $\alpha[x \mapsto \alpha'(c)]$ for all possible $\alpha'(c) \in U^{\mathcal{A}}$, then $\mathcal{A} \models_{\alpha[x\mapsto\alpha'(c)]} F$ for all possible values $\alpha'(c) \in U^{\mathcal{A}}$ and any $\mathcal{A}$. Hence, $\mathcal{A} \models_{\alpha[x\mapsto a]} F$ for all $a \in U^{\mathcal{A}}$ and any $\mathcal{A}$. Hence, $\mathcal{A} \models_\alpha \forall x F$ for all $\mathcal{A}, \alpha$. Hence $\forall x F$ is valid.

3. Which of the following languages are regular? Which are FO definable?

(a) The set of words over $\{a, b\}$ which has equal number of occurrences of ab and ba. For example, aba is in the language, while $abab$ is not.

(b) The set of words over $\{a, b, \#\}$ with a single occurrence of #, and every symbol before the # is an a, and all symbols after the # are b's.

(c) The set of strings over $\{a, b\}$ which does not contain any occurrence of ba.

(d) The set of strings over $\{0, 1\}$ such that the second symbol from both ends is 0.

(e) Let $\Sigma = \{\begin{pmatrix} a \\ b \end{pmatrix} \mid a, b \in \{0, 1\}\}$. A string over Σ gives two rows of 0's and 1's. Treat each row as a binary number. The set of words

$$\{w \in \Sigma^* \mid \text{ the top row is larger than the bottom row } \}$$

Solution

(a) Observe that since our alphabet is $\{a, b\}$, this language is equivalent to the set of languages that end and start with the same alphabet. The FO formula for this would be:

$$\forall x\, (\text{first}(x) \implies Q_a(x) \wedge \text{last}(x) \implies Q_a(x)) \vee$$
$$\forall x\, (\text{first}(x) \implies Q_b(x) \wedge \text{last}(x) \implies Q_b(x))$$

where $\text{first}(x) \equiv \neg\exists\, y[y < x]$ and $\text{last}(x) \equiv \neg\exists y\ [x < y]$.

(b) The required FO formula is:

$$\exists x[Q_{\#}(x) \wedge \forall y\, (Q_{\#}(y) \implies y = x) \wedge \forall y\, (y < x \implies Q_a(y)) \wedge$$
$$\forall y\, (x < y \implies Q_b(y))]$$

(c) The required FO formula is:

$$\neg\exists x \exists y\, [S(x, y) \wedge Q_b(x) \wedge Q_a(y)]$$

(d) The required FO formula is:

$$\exists x \exists y \exists z \exists w\ [\text{first}(x) \wedge S(x, y) \wedge Q_0(y) \wedge \text{last}(z) \wedge S(w, z) \wedge Q_0(w)]$$

(e) A necessary and sufficient condition for the top row to be larger than the bottom row is that at the first point of difference (which should exist), the top row should have bit 1 and the bottom row should have bit 0. In FOL, this can be expressed as:

$$\exists x \left[Q_{\begin{pmatrix}1\\0\end{pmatrix}}(x) \wedge \forall y \left(y < x \implies \left[Q_{\begin{pmatrix}0\\0\end{pmatrix}}(y) \vee Q_{\begin{pmatrix}1\\1\end{pmatrix}}(y) \right] \right) \right]$$

You can construct the DFAs for each of these languages, to prove that they are regular, as an exercise.

4. Consider the following FO formulae. In each case, answer the following questions:

- What is $L(\varphi)$?
- What is $\overline{L(\varphi)}$?
- Is $L(\varphi)$ regular?
- Is $\overline{L(\varphi)}$ regular?

(a) $\forall x (x \neq x)$

(b) $\exists x \exists y [x < y \wedge Q_b(x) \wedge Q_a(y) \wedge \forall z[(x < z < y) \to Q_a(z)]]$

(c) $\exists x[Q_a(x) \wedge \exists y[S(x,y) \wedge \forall z[z \leq y]]]$

(d) $\exists x \forall y[x \leq y \wedge Q_a(x)] \wedge \exists x \forall y[y \leq x \wedge Q_b(x)] \wedge$
$\forall x \forall y[Q_a(x) \wedge S(x,y) \to Q_b(y)] \wedge \forall x \forall y[Q_b(x) \wedge S(x,y) \to Q_a(y)]$

Solution

In this question, let $\Sigma = \{a, b\}$. Firstly, note that the for each sub-question, $L(\varphi)$ and $\overline{L(\varphi)}$ will be both regular as all FO definable languages are regular, and FO definable languages are closed under complementation. By inspection, we observe that the answers to (a) and (b) for the languages defined by the FO formulas are:

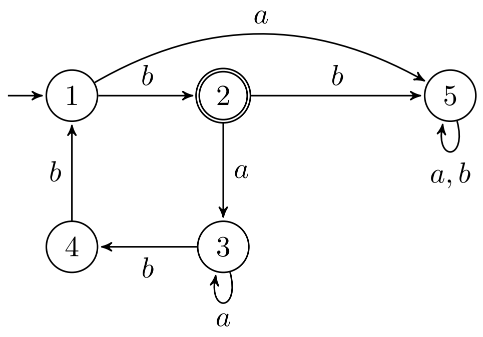
(a) $L(\varphi) = \{\epsilon\}$. Thus, $\overline{L(\varphi)} = \Sigma^* \setminus \{\epsilon\} = \Sigma^+$.

(b) $L(\varphi) = \Sigma^* ba \Sigma^*$. Also, $\overline{L(\varphi)} = a^* b^*$.

(c) $L(\varphi) = \Sigma^* a \Sigma$ and $\overline{L(\varphi)} = \{\epsilon\} \cup \Sigma \cup \Sigma^* b \Sigma$.

(d) $L(\varphi) = a(ba)^* b$ and $\overline{L(\varphi)} = \{\epsilon\} \cup b\Sigma^* \cup \Sigma^* a \cup \Sigma^* aa \Sigma^* \cup \Sigma^* bb \Sigma^*$

5. Consider the following automaton. What is the language L accepted? Can you write an FO formula φ such that $L = L(\varphi)$?



Solution

By observation, $L = b(a^+b^3)^*$ [a] and we also have $L = L(\varphi)$ where φ is

$$\exists x\, [\text{first}(x) \wedge Q_b(x)] \;\wedge\; \forall x[\neg\text{first}(x) \implies ([Q_a(x) \wedge \exists y(x < y \wedge Q_b(y))] \vee [C_1(x) \vee C_2(x) \vee C_3(x)])]$$

where $C_i(x)$ expresses that x is the i-th b in the a^+b^3 element, $i = 1, 2, 3$.

$$C_1(x) \equiv \exists p \exists q \exists r\, [Q_a(p) \wedge S(p,x) \wedge Q_b(x) \wedge S(x,q) \wedge Q_b(q) \wedge S(q,r) \wedge Q_b(r) \wedge \forall s(S(r,s) \implies Q_a(s))]$$

$$C_2(x) \equiv \exists p \exists q \exists r \, [Q_a(p) \land S(p,q) \land Q_b(q) \land S(q,x) \land Q_b(x) \land S(x,r) \land Q_b(r) \land \forall s (S(r,s) \implies Q_a(s))]$$

$$C_3(x) \equiv \exists p \exists q \exists r \, [Q_a(p) \land S(p,q) \land Q_b(q) \land S(q,r) \land Q_b(r) \land S(r,x) \land Q_b(x) \land \forall s (S(x,s) \implies Q_a(s))]$$

[a] $a^+ = a^* \setminus \{\epsilon\}$