CS228 Tutorial Solutions

Arpon Basu

February 10, 2023

Contents

1 Tutorial 1 **2**

2 Tutorial 2 **4**

3 Tutorial 3 **10**

4 Tutorial 4 **12**

5 Tutorial 5 **14**

6 Tutorial 6 **16**

6.1 Extra questions 18

1 Tutorial 1

Exercise 2.6

Let p_T be the propositional variable denoting if T is good, where $T \in \{A, B, C\}$. Then the puzzle can be encoded as

$$(p_A \Leftrightarrow (\neg p_A \wedge \neg p_B \wedge \neg p_C))$$

$$\bigwedge (p_B \Leftrightarrow ((p_A \wedge \neg p_B \wedge \neg p_C) \vee (\neg p_A \wedge p_B \wedge \neg p_C) \vee (\neg p_A \wedge \neg p_B \wedge p_C)))$$

Exercise 2.7

Consider the "let"-expression

$$\text{let } p_0 = (\text{let } p_1 = (\text{let } p_2 = (\ldots) \text{ in } p_2 \wedge p_2) \text{ in } p_1 \wedge p_1) \text{ in } p_0 \wedge p_0$$

This expression, in linear length, represents a formula of exponential size.

Exercise 3.10

We will not write $m(\cdot)$ in the top row for brevity.

(a)

p	q	$p \to q$	$\neg p$	$\neg p \vee q$
1	1	1	0	1
1	0	0	0	0
0	1	1	1	1
0	0	1	1	1

(b) DIY

(c)

p	q	$p \oplus q$	$\neg p$	$\neg q$	$\neg p \wedge q$	$p \wedge \neg q$	$(\neg p \wedge q) \vee (p \wedge \neg q)$
1	1	0	0	0	0	0	0
1	0	1	0	1	0	1	1
0	1	1	1	0	1	0	1
0	0	0	1	1	0	0	0

(d) DIY

Exercise 3.23

Consider the model m, where $m[p] = 1$ if and only if $p \in \Sigma$.
Assume for the sake of contradiction that $m \not\models \Sigma$, and thus let H be *a smallest* [1] formula such that $m \not\models H$. Note that if the root connective of H is $\neg$, then H

[1] note that every formula in propositional logic can be generated in finitely many steps from the base cases. Consequently, for every formula F, there is a minimum number of steps 'k' needed to generate F, and we call k the *size* of F. Once we have defined a finite size for every formula, we can talk about *a* smallest formula of some subset of formulae

has at least 2 connectives in it by the properties of m [2], and then by the rules of generating formulae, H must be of the form $\neg\neg F, F \wedge G, F \vee G, \neg(F \vee G)$ or $\neg(F \wedge G)$ for some formulae F, G. Now,

1. $H = \neg\neg F$: Since $H \in \Sigma$, $F \in \Sigma$. But since $m \not\models H$, $m \not\models F$. But F is a strictly smaller formula than H. Thus this case is not possible.

2. $H = F \wedge G$: Since $H \in \Sigma$, $F, G \in \Sigma$. But since $m \not\models H$, either $m \not\models F$ or $m \not\models G$. But F, G are strictly smaller than H. Thus this case is not possible.

3. $H = F \vee G$: Since $H \in \Sigma$, either $F \in \Sigma$ or $G \in \Sigma$. But since $m \not\models H$, $m \not\models F$ and $m \not\models G$. But F, G are strictly smaller than H. Thus this case is not possible.

4. $H = \neg(F \vee G)$: Since $H \in \Sigma$, $\neg F, \neg G \in \Sigma$. But since $m \not\models H$, $m \models (F \vee G)$, and thus $m \models F$ or $m \models G$, implying $m \not\models \neg F$ or $m \not\models \neg G$. But $\neg F, \neg G$ are strictly smaller than H. Thus this case is not possible.

5. $H = \neg(F \wedge G)$: Since $H \in \Sigma$, either $\neg F \in \Sigma$ or $\neg G \in \Sigma$. But since $m \not\models H$, $m \models (F \wedge G)$, and thus $m \models F$ and $m \models G$, implying $m \not\models \neg F$ and $m \not\models \neg G$. But $\neg F, \neg G$ are strictly smaller than H. Thus this case is not possible.

Consequently, we arrive at a contradiction. Thus $m \models \Sigma$, ie:- Σ is satisfiable.

Exercise 3.28

(a) F and $F[\neg p/p]$ are equisatisfiable: Suppose F is satisfiable. Let m be a model such that $m \models F$. Then note that $m[p \to 1 - m[p]] \models F[\neg p/p]$, ie:- if we flip the assignment of p in m, then we get a satisfying model for $F[\neg p/p]$, and consequently $F[\neg p/p]$ is satisfiable.
Now suppose $F[\neg p/p]$ is satisfiable: Then once again, for any satisfying model m' of $F[\neg p/p]$, $m'[p \to 1 - m'[p]] \models F$, and thus F is satisfiable.
Inductive "Rewriting" of the above solution:- We proceed by induction, with our induction hypothesis being that $m \models F \iff m' \models F[\neg p/p]$, where the assignment of p in m' is the opposite of that in m. If $F(p)$ is atomic, then we have two cases: Either $F(p) = p$, or $F(p) = q$ for some other propositional variable q. In the first case $F[\neg p/p] = \neg p$, and in the second case $F = F[\neg p/p] = q$, and in both cases $F, F[\neg p/p]$ are equisatisfiable.
Now assume F is not atomic: WLOG we can assume that its root connective is not $\neg$, because $F, F[\neg p/p]$ are equisatisfiable iff $\neg F, \neg F[\neg p/p]$ are. Then, let $\circ$ be the root binary connective of F. Note that if $F = F_1 \circ F_2$, then $F[\neg p/p] = F_1[\neg p/p] \circ F_2[\neg p/p]$. After this point, we have to perform casework on $\circ$ being $\wedge, \vee, \Rightarrow, \Leftrightarrow$ or $\oplus$. I'll do the $\wedge$ one, leaving

[2] If H is of the form $\neg F$ and F doesn't have any further connectives, then H is $\neg p$ for some propositional variable p and m satisfies it by definition

the others to you. If $m \models F$, then $m_i \models F_i$ where $m_i := m|_{\mathrm{Vars}(F_i)}$ for $i \in \{1, 2\}$. Consequently, $m_i' := m_i[1 - m_i[p]] \models F_i[\neg p/p]$ by the induction hypothesis. Further, since the m_i's are the restrictions of the same model m to the domain $\mathrm{Vars}(F_i)$, they can be "merged" safely back again, ie:- $m_i[1 - m_i[p]] \hookrightarrow m[1 - m[p]] \models F[\neg p/p]$, as desired.

(b) F and $F[(p \wedge q)/p]$ are not equisatisfiable: For $F = p \wedge \neg q$, F is satisfiable but $F[(p \wedge q)/p] = (p \wedge q) \wedge \neg q$ is not satisfiable.

(c) F and $F[(p \vee q)/p]$ are not equisatisfiable: For $F = \neg p \wedge q$, F is satisfiable but $F[(p \vee q)/p] = \neg(p \vee q) \wedge q \equiv (\neg p \wedge \neg q) \wedge q$ is not satisfiable.

2 Tutorial 2

Exercise 3.15

We make some quick observations about the $\neg, \oplus$ connectives. For any formulae F, G, H, we have:

1. $\neg(F \oplus G) \equiv \neg F \oplus G$: Indeed, if $m \not\models \neg(F \oplus G)$, then $m \models F \oplus G$, implying that m satisfies exactly one of the formulae among F, G. If $m \models G$, and thus $m \not\models F \implies m \models \neg F$, then $m \not\models \neg F \oplus G$ because m satisfies both $\neg F, G$. If $m \models F$, then $m \not\models G$, and $m \not\models \neg F$, and consequently, $m \not\models \neg F \oplus G$ since m doesn't satisfy either $\neg F, G$. Thus $m \not\models \neg(F \oplus G) \implies m \not\models \neg F \oplus G$.
   If $m \models \neg(F \oplus G)$, then $m \not\models F \oplus G$. Thus either m satisfies both F, G, in which case it satisfies exactly one formula in $\{\neg F, G\}$, and thus $m \models \neg F \oplus G$. Otherwise m doesn't satisfy F or G, and again $m \models \neg F \oplus G$. Consequently, $m \models \neg(F \oplus G) \implies m \models \neg F \oplus G$, and thus $\neg(F \oplus G) \equiv \neg F \oplus G$.

2. $F \oplus G \equiv G \oplus F$, ie:- $\oplus$ is a commutative connective

3. $(F \oplus G) \oplus H \equiv F \oplus (G \oplus H)$, ie:- $\oplus$ is an associative connective

Consequently, for any formula F consisting only of $\neg, \oplus$, we can push the $\neg$s inside by the first observation, and then we can flatten the parentheses out using the associativity of $\oplus$. Consequently, any formula consisting only of $\neg, \oplus$ is equivalent to $\bigoplus \ell_i$, where ℓ_i is either some propositional variable or the negation of a propositional variable. Also, note that $p \oplus p = \bot, p \oplus \neg p = \top, \top \oplus p = \neg p, \bot \oplus p = p$. Consequently, for any formula F built only from $\neg, \oplus$, we have that $F \equiv \top$, or $F \equiv \bot$, or $F \equiv \bigoplus \ell_i$, and furthermore, $\mathrm{Vars}(\ell_i) \neq \mathrm{Vars}(\ell_j)$ for $i \neq j$. In all of these cases observe that the number of satisfying assignments of F is either 0 or a power of 2. Consequently, F can't represent, for example, $p \vee q$, since $p \vee q$ has 3 satisfying assignments.

Exercise 4.3

Consider the propositional variables G, S, D, P denoting if the laws are good, if the laws have strict enforcement, if crime diminishes, and if our problem is a practical one, respectively.
We have to show that $\Sigma := \{(G \wedge S) \Rightarrow D, (S \Rightarrow D) \Rightarrow P, G\} \vdash P$. Then

1.	$\Sigma \vdash (G \wedge S) \Rightarrow D$	Assumption
2.	$\Sigma \vdash (S \Rightarrow D) \Rightarrow P$	Assumption
3.	$\Sigma \vdash G$	Assumption
4.	$\Sigma \cup \{S\} \vdash S$	Assumption
5.	$\Sigma \cup \{S\} \vdash G$	Monotonic on (3)
6.	$\Sigma \cup \{S\} \vdash (G \wedge S) \Rightarrow D$	Monotonic on (1)
7.	$\Sigma \cup \{S\} \vdash (G \wedge S)$	$\wedge$-intro in (5, 4)
8.	$\Sigma \cup \{S\} \vdash D$	$\Rightarrow$-elim in (6, 7)
9.	$\Sigma \vdash S \Rightarrow D$	$\Rightarrow$-intro in (8)
10.	$\Sigma \vdash P$	$\Rightarrow$-elim in (2, 9)

Exercise 4.4.1

Find it on page number 26 in here.

Exercise 5.4

In this question, we shall let Σ be the set of formulae given on the left-hand side of the derivation.

(1)

1.	$\Sigma \vdash p \Rightarrow q$	Assumption
2.	$\Sigma \vdash \neg p \vee q$	$\Rightarrow$-def on (1)
3.	$\Sigma \vdash p \vee q$	Assumption
4.	$\Sigma \vdash q \vee q$	Resolution on (2, 3)
5.	$\Sigma \cup \{q\} \vdash q$	Assumption
6.	$\Sigma \vdash q$	$\vee$-elim on (4, 5, 5)

(2)

Heuristic: Here we don't have any negation explicitly in our Σ, yet we have to derive $\neg F$. Your best bet here is to try to use ByContra somehow because it is one of the very few rules that introduce a negation in our formula.

1.	$\Sigma \vdash p \Rightarrow q$	Assumption
2.	$\Sigma \cup \{\neg r \wedge p\} \vdash p \Rightarrow q$	Monotonic on (1)
3.	$\Sigma \cup \{\neg r \wedge p\} \vdash \neg r \wedge p$	Assumption
4.	$\Sigma \cup \{\neg r \wedge p\} \vdash p \wedge \neg r$	$\wedge$-symm on (3)
5.	$\Sigma \cup \{\neg r \wedge p\} \vdash p$	$\wedge$-elim on (4)
6.	$\Sigma \cup \{\neg r \wedge p\} \vdash q$	$\Rightarrow$-elim on (2, 5)
7.	$\Sigma \cup \{\neg r \wedge p\} \vdash \neg r$	$\wedge$-elim on (3)
8.	$\Sigma \vdash q \Rightarrow r$	Assumption
9.	$\Sigma \vdash \neg q \vee r$	$\Rightarrow$-def on (8)
10.	$\Sigma \cup \{\neg r \wedge p\} \vdash \neg q \vee r$	Monotonic on (9)
11.	$\Sigma \cup \{\neg r \wedge p\} \vdash r$	UnitRes on (10, 6)
12.	$\Sigma \vdash \neg(\neg r \wedge p)$	ByContra on (11, 7)

(3)

1.	$\Sigma \vdash (q \vee (r \wedge s))$	Assumption
2.	$\Sigma \vdash q \Rightarrow t$	Assumption
3.	$\Sigma \cup \{q\} \vdash q \Rightarrow t$	Monotonic on (2)
4.	$\Sigma \cup \{q\} \vdash q$	Assumption
5.	$\Sigma \cup \{q\} \vdash t$	$\Rightarrow$-elim on (3, 4)
6.	$\Sigma \vdash t \Rightarrow s$	Assumption
7.	$\Sigma \cup \{q\} \vdash t \Rightarrow s$	Monotonic on (6)
8.	$\Sigma \cup \{q\} \vdash s$	$\Rightarrow$-elim on (7, 5)
9.	$\Sigma \cup \{r \wedge s\} \vdash r \wedge s$	Assumption
10.	$\Sigma \cup \{r \wedge s\} \vdash s \wedge r$	$\wedge$-symm on (9)
11.	$\Sigma \cup \{r \wedge s\} \vdash s$	$\wedge$-elim on (10)
12.	$\Sigma \vdash s$	$\vee$-elim on (1, 8, 11)

(4)

1.	$\Sigma \vdash p \vee q$	Assumption
2.	$\Sigma \vdash r \vee s$	Assumption
3.	$\Sigma \cup \{p\} \vdash r \vee s$	Monotonic on (2)
4.	$\Sigma \cup \{p, r\} \vdash p$	Assumption
5.	$\Sigma \cup \{p, r\} \vdash r$	Assumption
6.	$\Sigma \cup \{p, r\} \vdash p \wedge r$	$\wedge$-intro in (4, 5)
7.	$\Sigma \cup \{p, r\} \vdash (p \wedge r) \vee (q \vee s)$	$\vee$-intro in (6)
8.	$\Sigma \cup \{p, s\} \vdash s$	Assumption
9.	$\Sigma \cup \{p, s\} \vdash s \vee q$	$\vee$-intro in (8)
10.	$\Sigma \cup \{p, s\} \vdash q \vee s$	$\vee$-symm in (9)
11.	$\Sigma \cup \{p, s\} \vdash (q \vee s) \vee (p \wedge r)$	$\vee$-intro in (10)
12.	$\Sigma \cup \{p, s\} \vdash (p \wedge r) \vee (q \vee s)$	$\vee$-symm in (11)
13.	$\Sigma \cup \{p\} \vdash (p \wedge r) \vee (q \vee s)$	$\vee$-elim in (3, 7, 12)
14.	$\Sigma \cup \{q\} \vdash q$	Assumption
15.	$\Sigma \cup \{q\} \vdash (q \vee s)$	$\vee$-intro in (14)
16.	$\Sigma \cup \{q\} \vdash (q \vee s) \vee (p \wedge r)$	$\vee$-intro in (15)
17.	$\Sigma \cup \{q\} \vdash (p \wedge r) \vee (q \vee s)$	$\vee$-symm in (16)
18.	$\Sigma \vdash (p \wedge r) \vee (q \vee s)$	$\vee$-elim in (1, 13, 17)

(5)

Heuristic: To show $\Sigma \vdash F \Rightarrow G$ it is enough to show $\Sigma \cup \{F\} \vdash G$.

1.	$\{p\} \cup \{p \Rightarrow q\} \vdash p$	Assumption
2.	$\{p\} \cup \{p \Rightarrow q\} \vdash p \Rightarrow q$	Assumption
3.	$\{p\} \cup \{p \Rightarrow q\} \vdash q$	$\Rightarrow$-elim on (2, 1)
4.	$\{p\} \vdash (p \Rightarrow q) \Rightarrow q$	$\Rightarrow$-intro on (3)
5.	$\Sigma \vdash ((p \Rightarrow q) \Rightarrow q) \Rightarrow q$	Assumption
6.	$\Sigma \cup \{p\} \vdash ((p \Rightarrow q) \Rightarrow q) \Rightarrow q$	Monotonic on (5)
7.	$\Sigma \cup \{p\} \vdash (p \Rightarrow q) \Rightarrow q$	Monotonic on (4)
8.	$\Sigma \cup \{p\} \vdash q$	$\Rightarrow$-elim on (6, 7)
9.	$\Sigma \vdash p \Rightarrow q$	$\Rightarrow$-intro in (8)

(6)

Heuristic: The first clause becomes true when r is set to true, while the second clause becomes true when r is set to false. Thus we do casework on r, introducing $\neg r \vee r$ through the Tautology rule. The reason we don't split on p or q is that q is absent from the second clause while setting p to true doesn't lead to an automatic truth assignment of the clauses.

1.	$\emptyset \vdash \neg r \vee r$	Tautology
2.	$\{\neg r\} \vdash \neg r$	Assumption
3.	$\{\neg r\} \vdash \neg r \vee \neg p$	$\vee$-intro in (2)
4.	$\{\neg r\} \vdash r \Rightarrow \neg p$	$\Rightarrow$-def in (3)
5.	$\{\neg r\} \vdash (r \Rightarrow \neg p) \vee (p \Rightarrow (q \vee r))$	$\vee$-intro in (4)
6.	$\{\neg r\} \vdash (p \Rightarrow (q \vee r)) \vee (r \Rightarrow \neg p)$	$\vee$-symm in (5)
7.	$\{r\} \vdash r$	Assumption
8.	$\{r\} \vdash r \vee q$	$\vee$-intro in (7)
9.	$\{r\} \vdash q \vee r$	$\vee$-symm in (8)
10.	$\{r\} \vdash (q \vee r) \vee \neg p$	$\vee$-intro in (9)
11.	$\{r\} \vdash \neg p \vee (q \vee r)$	$\vee$-symm in (10)
12.	$\{r\} \vdash p \Rightarrow (q \vee r)$	$\Rightarrow$-def in (11)
13.	$\{r\} \vdash (p \Rightarrow (q \vee r)) \vee (r \Rightarrow \neg p)$	$\vee$-intro in (12)
14.	$\emptyset \vdash (p \Rightarrow (q \vee r)) \vee (r \Rightarrow \neg p)$	$\vee$-elim in (1, 6, 13)

(7)

1.	$\Sigma \vdash p$	Assumption
2.	$\Sigma \vdash p \vee \neg q$	$\vee$-intro on (1)
3.	$\Sigma \vdash \neg q \vee p$	$\vee$-symm on (2)
4.	$\Sigma \vdash q \Rightarrow p$	$\Rightarrow$-def on (3)

(8)

1.	$\Sigma \vdash (p \Rightarrow (q \Rightarrow r))$	Assumption
2.	$\Sigma \cup \{p \Rightarrow q\} \vdash p \Rightarrow q$	Assumption
3.	$\Sigma \cup \{p \Rightarrow q, p\} \vdash p \Rightarrow q$	Monotonic on (2)
4.	$\Sigma \cup \{p \Rightarrow q, p\} \vdash p$	Assumption
5.	$\Sigma \cup \{p \Rightarrow q, p\} \vdash q$	$\Rightarrow$-elim on (3, 4)
6.	$\Sigma \cup \{p \Rightarrow q, p\} \vdash (p \Rightarrow (q \Rightarrow r))$	Monotonic on (1)
7.	$\Sigma \cup \{p \Rightarrow q, p\} \vdash q \Rightarrow r$	$\Rightarrow$-elim on (4, 6)
8.	$\Sigma \cup \{p \Rightarrow q, p\} \vdash r$	$\Rightarrow$-elim on (5, 7)
9.	$\Sigma \cup \{p \Rightarrow q\} \vdash p \Rightarrow r$	$\Rightarrow$-intro on (8)
10.	$\Sigma \vdash ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))$	$\Rightarrow$-intro on (9)

(9)

A typical demonstration of the use of RevDoubleNeg.

1.	$\Sigma \vdash (\neg p \Rightarrow \neg q)$	Assumption
2.	$\Sigma \vdash \neg\neg p \lor \neg q$	$\Rightarrow$-def on (1)
3.	$\Sigma \vdash \neg q \lor \neg\neg p$	$\lor$-symm on (2)
4.	$\Sigma \cup \{\neg q\} \vdash \neg q$	Assumption
5.	$\Sigma \cup \{\neg q\} \vdash \neg q \lor p$	$\lor$-intro in (4)
6.	$\Sigma \cup \{\neg\neg p\} \vdash \neg\neg p$	Assumption
7.	$\Sigma \cup \{\neg\neg p\} \vdash p$	RevDoubleNeg on (6)
8.	$\Sigma \cup \{\neg\neg p\} \vdash p \lor \neg q$	$\lor$-intro in (7)
9.	$\Sigma \cup \{\neg\neg p\} \vdash \neg q \lor p$	$\lor$-symm in (8)
10.	$\Sigma \vdash \neg q \lor p$	$\lor$-elim on (3, 5, 9)
11.	$\Sigma \vdash q \Rightarrow p$	$\Rightarrow$-def on (10)

(10)

Heuristic: Note that our formula to be proved, $t \Rightarrow u$, is independent of r. This is usually a tell-tale sign of ByCases being involved, with the casework being done on the variable which isn't involved in the final formula.

1.	$\Sigma \vdash (r \lor s) \Rightarrow (u \lor \neg t)$	Assumption
2.	$\Sigma \cup \{r\} \vdash r$	Assumption
3.	$\Sigma \cup \{r\} \vdash r \lor s$	$\lor$-intro in (2)
4.	$\Sigma \cup \{r\} \vdash (r \lor s) \Rightarrow (u \lor \neg t)$	Monotonic on (1)
5.	$\Sigma \cup \{r\} \vdash u \lor \neg t$	$\Rightarrow$-elim on (4, 3)
6.	$\Sigma \cup \{r\} \vdash \neg t \lor u$	$\lor$-symm on (5)
7.	$\Sigma \cup \{r\} \vdash t \Rightarrow u$	$\Rightarrow$-def on (6)
8.	$\Sigma \cup \{\neg r\} \vdash \neg r$	Assumption
9.	$\Sigma \vdash r \lor (s \land \neg t)$	Assumption
10.	$\Sigma \cup \{r\} \vdash \neg\neg r$	DoubleNeg on (2)
11.	$\Sigma \cup \{r\} \vdash \neg\neg r \lor (s \land \neg t)$	$\lor$-intro in (10)
12.	$\Sigma \cup \{s \land \neg t\} \vdash s \land \neg t$	Assumption
13.	$\Sigma \cup \{s \land \neg t\} \vdash (s \land \neg t) \lor \neg\neg r$	$\lor$-intro in (12)
14.	$\Sigma \cup \{s \land \neg t\} \vdash \neg\neg r \lor (s \land \neg t)$	$\lor$-symm in (13)
15.	$\Sigma \vdash \neg\neg r \lor (s \land \neg t)$	$\lor$-elim in (9, 11, 14)
16.	$\Sigma \cup \{\neg r\} \vdash \neg\neg r \lor (s \land \neg t)$	Monotonic on (15)
17.	$\Sigma \cup \{\neg r\} \vdash s \land \neg t$	UnitRes on (16, 8)
18.	$\Sigma \cup \{\neg r\} \vdash \neg t \land s$	$\land$-symm on (17)
19.	$\Sigma \cup \{\neg r\} \vdash \neg t$	$\land$-elim on (18)
20.	$\Sigma \cup \{\neg r\} \vdash \neg t \lor u$	$\lor$-intro in (19)
21.	$\Sigma \cup \{\neg r\} \vdash t \Rightarrow u$	$\Rightarrow$-def in (20)
22.	$\Sigma \vdash t \Rightarrow u$	ByCases on (7, 21)

3 Tutorial 3

Exercise 6.13

$$\underbrace{p \oplus \ldots \oplus p}_{n} \oplus \underbrace{\neg p \oplus \ldots \oplus \neg p}_{k} = \begin{cases} \top, n \text{ odd}, k \text{ odd} \\ \bot, n \text{ even}, k \text{ even} \\ p, \quad n \text{ odd}, k \text{ even} \\ \neg p, n \text{ even}, k \text{ odd} \end{cases}$$

This can be formally established through a joint induction on n, k.

Exercise 6.16

(a) Let $m \models F \vee G(F)$. If $m \models F$, then $m \models F \vee G(\bot)$. Otherwise, if $m \not\models F$, then we have $(m \models F \Leftrightarrow m \models \bot)$, which implies, by Theorem 6.1, that $(m \models G(F) \Leftrightarrow m \models G(\bot))$. However, since $m \models F \vee G(F)$ yet $m \not\models F$, we have $m \models G(F)$, and consequently, $m \models G(\bot)$, further implying that $m \models F \vee G(\bot)$.
Thus $m \models F \vee G(F) \Rightarrow m \models F \vee G(\bot)$.
In the reverse direction, if $m' \models F \vee G(\bot)$, and if $m \models F$, we have $m' \models F \vee G(F)$. Otherwise $m' \not\models F, m' \models G(\bot)$. As above, we can then conclude $m' \models G(F)$, and thus $m' \models F \vee G(F)$, implying $m' \models F \vee G(\bot) \Rightarrow m' \models F \vee G(F)$.
Consequently, $F \vee G(F) \equiv F \vee G(\bot)$.

(b) Follows similarly as above.

(c) Follows similarly as above.

Exercise 7.12

The flaw with the argument is that it assumes that the Tseitin encoding preserves validity, which is not the case. Indeed, consider

$$F := (p_1 \wedge p_2) \vee (\neg p_1 \wedge p_2) \vee (p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge \neg p_2)$$

Then F is valid. However,

$$\text{Tseitin}(F) = (q_1 \vee q_2 \vee q_3 \vee q_4) \wedge (\neg q_1 \vee p_1) \wedge (\neg q_1 \vee p_2) \wedge (\neg q_2 \vee \neg p_1) \wedge (\neg q_2 \vee p_2)$$

$$\wedge (\neg q_3 \vee p_1) \wedge (\neg q_3 \vee \neg p_2) \wedge (\neg q_4 \vee \neg p_1) \wedge (\neg q_4 \vee \neg p_2)$$

is not valid, as is demonstrated by the model which assigns all the q_is to 0.

Exercise 7.20

We want to show that if Σ is unsatisfiable, then we can derive $\perp$ from Σ without involving valid clauses in the derivation. We shall assume that Σ is finite, and consequently, we can then induct on the number of propositional variables in Σ, ie:- $n := \left|\bigcup_{F\in\Sigma} \mathrm{Vars}(F)\right|$, to prove our assertion.
If $n = 1$, then Σ contains only one propositional variable, say p. Since Σ is unsatisfiable it must contain both $\{p\}$ and $\{\neg p\}$, and then we derive $\perp$ without involving $p \vee \neg p = \{p, \neg p\}$.
Let the above assertion be true for any Σ containing at most n variables in it, and consider any Σ with $\bigcup_{F\in\Sigma} \mathrm{Vars}(F) = \{p, p_1, \dots, p_n\}$. Partition Σ into 4 sets: $\Sigma_0, \Sigma_1, \Sigma_*, \Sigma_{\text{valid}}$, where

$$\Sigma_0 := \{F \in \Sigma : p \in F, \neg p \notin F\}$$

$$\Sigma_1 := \{F \in \Sigma : p \notin F, \neg p \in F\}$$

$$\Sigma_* := \{F \in \Sigma : p \notin F, \neg p \notin F\}$$

$$\Sigma_{\text{valid}} := \{F \in \Sigma : p \in F, \neg p \in F\}$$

We further define

$$\Sigma_0' := \{F \setminus \{p\} : F \in \Sigma_0\}$$

$$\Sigma_1' := \{F \setminus \{\neg p\} : F \in \Sigma_1\}$$

Note that since Σ is unsatisfiable, so is $\Sigma_0' \cup \Sigma_*$ [3]. Indeed, if $m \models \Sigma_0' \cup \Sigma_*$, then $m[p \to 0] \models \Sigma$. Similarly, one can see that $\Sigma_1' \cup \Sigma_*$ is unsatisfiable too. Further note that $\Sigma_i' \cup \Sigma_*, i \in \{0, 1\}$ have at most n variables, and thus have a resolution proof for $\perp$ *without involving valid clauses*, by the induction hypothesis. Now, if either of the proofs $\Sigma_i' \cup \Sigma_* \vdash \perp$ uses clauses only from Σ_*, then we're done by the induction hypothesis. Otherwise adjoin p or $\neg p$ to clauses in Σ_0' and Σ_1' respectively [4] to obtain $\Sigma_0 \cup \Sigma_* \vdash \{p\}$, $\Sigma_1 \cup \Sigma_* \vdash \{\neg p\}$, and then finally resolve $\{p\}, \{\neg p\}$ to get $\perp$. Note that we did not involve any clause from Σ_{valid} in this step, and we can be sure that no valid clause was invoked anywhere else in the proof by the induction hypothesis. Consequently, we have our desired proof of $\Sigma \vdash \perp$ without valid clauses.

Exercise 8.3

We use induction on n. For $n = 1$, as we observed in the earlier question, we can resolve Σ in at most $1 \leq 2^{1+1} - 1$ steps. For any Σ with n variables, as above, we can derive $\perp$ from $\Sigma_i' \cup \Sigma_*$ in at most $2^{(n-1)+1} - 1 = 2^n - 1$ steps. If any of these proofs use clauses only from Σ_*, then we're done in $2^n - 1 \leq 2^{n+1} - 1$ steps. Otherwise, adjoin $p, \neg p$ to these proofs, and resolve $p, \neg p$ in the final step to get a derivation of at most $2 \cdot (2^n - 1) + 1 = 2^{n+1} - 1$ steps, as desired.

[3] Note that $\mathcal{T}_1 \cup \mathcal{T}_2 \equiv \mathcal{T}_1 \bigwedge \mathcal{T}_2$ for any two CNFs $\mathcal{T}_1, \mathcal{T}_2$

[4] Note that adjoining p to a clause in Σ_0' doesn't make it valid: Indeed, if validity were to be introduced by the variable p, then one would have to adjoin both p and $\neg p$. Thus our proof $\Sigma_0 \cup \Sigma_* \vdash \perp$ remains free of valid clauses after the adjoining process.

Exercise 8.8

Note that if $\ell = p$, then $F|_\ell = F_1' \cup F_*$, and if $\ell = \neg p$, then $F|_\ell = F_0' \cup F_*$.

1. If $F|_\ell \vdash \perp$ then we either have $F_* \vdash \perp$, in which case we're done since $F_* \subseteq F \subseteq F \cup \{\ell\}$, or we have $F \vdash \{\overline{\ell}\}$, as in Exercise 7.20. Consequently, $F \wedge \ell \vdash \perp$, where we derive $\overline{\ell}$ from F and then resolve it using ℓ.

2. For convenience assume $\ell = p$ for some propositional variable p [5]. Consider the slimmest derivation $F|_\ell \vdash \perp$ with width w_1. Note that if this proof uses clauses only from F_*, then we don't proceed further. Otherwise, when we adjoin $\overline{\ell}$ to obtain the derivation $F \vdash \overline{\ell}$, the width becomes atmost $1 + w_1$. Having obtained $\overline{\ell}$, resolve it with every clause in F_0 to obtain $F|_{\overline{\ell}}$, and let the width of the derivation $F_0 \vdash F|_{\overline{\ell}}$ be w_2. Finally, derive $\perp$ from $F|_{\overline{\ell}}$ in width w_3 in the slimmest possible manner.
   Consequently, the width of the entire proof described above is at most $\max(1 + w_1, w_2, w_3)$. Now, $w_3 \leq k$ by the problem hypothesis. Also, $w_1 \leq k - 1 \Rightarrow 1 + w_1 \leq k$, once again, by the problem hypothesis. Finally, note that $w_2 = 1 + \text{width}(F_0) = 1 + \text{width}(F) - 1 \leq k$, and thus $\max(1 + w_1, w_2, w_3) \leq k$, as desired.

4 Tutorial 4

Exercise 10.8

DIY

Exercise 10.10

From the 2-SAT algorithm, we know that if our instance is unsatisfiable, then there exists an SCC of our implication graph which contains both $p, \neg p$ for some propositional variable p. Furthermore, from an algorithmic point of view, such an SCC can be detected during the run of the 2-SAT algorithm itself, ie:- when we encounter a component to whom we can't assign a value consistently.
Now, take that SCC, and find out all variables $\{p_i\}$ in it such that their negations are also present in that SCC. Now, for every such p_i, find out a shortest path from p_i to $\neg p_i$, and then find a shortest path from $\neg p_i$ to p_i. Then find a propositional variable (say p_0) from $\{p_i\}$ such that length($p_0 \to \neg p_0$)+length($\neg p_0 \to p_0$) is the minimum among all the variables in $\{p_i\}$.
We claim that the clauses represented by the edges of the $p_0 \to \neg p_0 \to p_0$ path form a minimal unsatisfiable core. Indeed, unsatisfiability follows since their implication graph contains $p_0, \neg p_0$ in the same SCC. For minimality note that once we delete any edge from the $p_0 \to \neg p_0 \to p_0$ path, $p_0, \neg p_0$ don't remain strongly connected anymore. Further, no other variable q is connected to its

[5]the proof goes through exactly the same way if $\ell = \neg p$

negation in the clipped $p_0 \to \neg p_0 \to p_0$ path since that would contradict the minimality of its length. Consequently, the clipped $p_0 \to \neg p_0 \to p_0$ path is satisfiable, demonstrating minimality.

Exercise 11.8

Refer to Figure 1 (Credits to Shantanu Nene for this image).

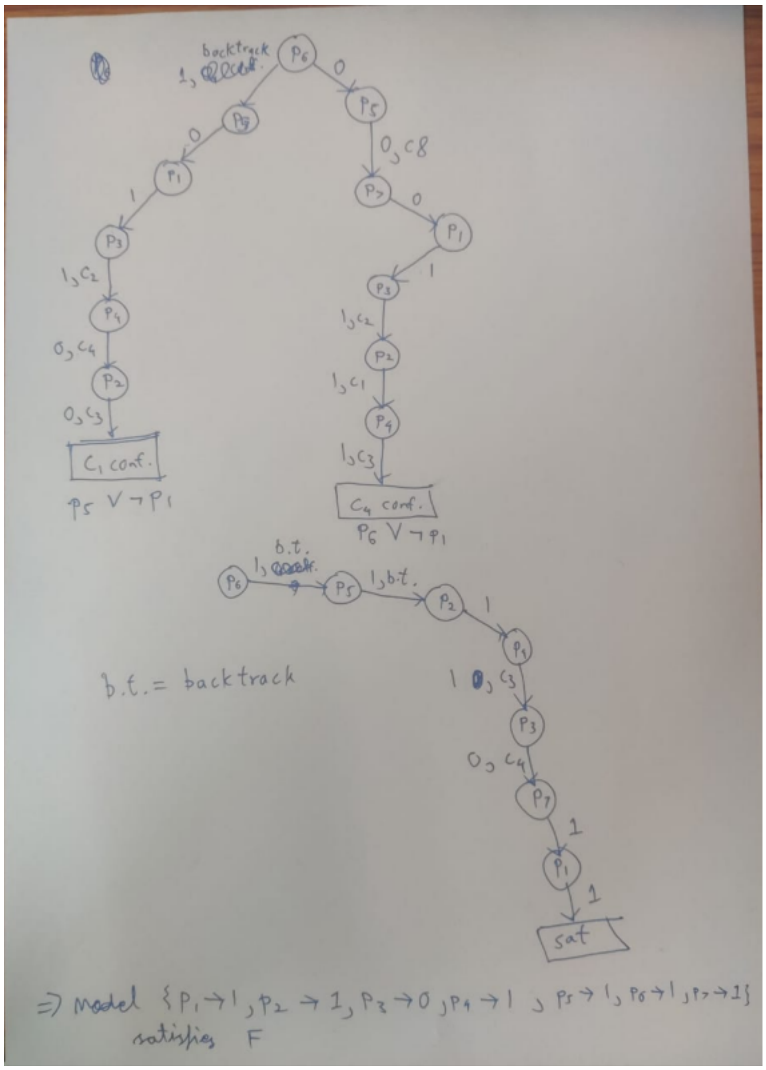

Figure 1: Exercise 11.8

Exercise 11.9

Given any formula F, note that no run of the CDCL algorithm repeats a (partial) model twice, because for any partial model m, if $m \models F$ then the algorithm terminates, and if $m \not\models F$, then we have a conflict after performing unit propagation from m, following which CDCL learns a clause which thereon prevents it from repeating the model m. Consequently, due to the finiteness of the number of partial models on $\text{Vars}(F)$, CDCL on F must terminate.

5 Tutorial 5

Throughout this tutorial we shall use the shorthand notation $[x], x \in \mathbb{N}$ to denote the set $\{1, 2, \ldots, x\}$.

Exercise 12.9

Let p_{ij} denote if we have a queen on the square (i, j), where $i, j \in [n]$. Then our SAT encoding is

$$\underbrace{\sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} = n}_{\text{We have } n \text{ queens}}$$

$$\bigwedge_{(i,j) \text{ and } (i',j') \text{ attack each other}} p_{ij} \Rightarrow \neg p_{i'j'}$$

Note that (i, j) and (i', j') attack each other if and only if at least one of the following conditions hold:-

1. $i = i'$
2. $j = j'$
3. $i + j = i' + j'$
4. $i - j = i' - j'$

Exercise 12.11

Suppose we have ℓ sets $\{S_i\}_{i \in [\ell]}$ such that $S_i \subseteq [n]$ and $|S_i| = k$ for every $i \in [\ell]$. Further, we also have $|S_i \cap S_j| = 1$ for all distinct $i, j \in [\ell]$.
Consider the propositional variables $\{p_{ij}\}_{i \in [n], j \in [\ell]}$, where p_{ij} is true if $i \in S_j$. Then the constraints of our problem dictate

$$F_\ell := \underbrace{\bigwedge_{j \in [\ell]} \left(\sum_{i \in [n]} p_{ij} = k \right)}_{\text{Every set has size } k} \wedge \underbrace{\bigwedge_{1 \le j_1 < j_2 \le \ell} \left(\sum_{i \in [n]} p_{ij_1} \wedge p_{ij_2} = 1 \right)}_{|S_{j_1} \cap S_{j_2}| = 1}$$

Finally, to encode the maximality of our set family, we write

$$G_\ell := F_\ell \wedge \neg F_{\ell+1}$$

$$F := \bigvee_{\ell=1}^{\binom{n}{k}} G_\ell$$

Note that we implicitly assume that $\mathrm{Vars}(G_{\ell_1}) \cap \mathrm{Vars}(G_{\ell_2}) = \mathrm{Vars}(F_{\ell_1}) \cap \mathrm{Vars}(F_{\ell_2}) = \emptyset$ if $\ell_1 \neq \ell_2$, ie:- we generate fresh variables every time we generate a different F_ℓ, G_ℓ. In particular, note that the variables of F_ℓ in G_ℓ and the variables of F_ℓ in $G_{\ell-1}$ are distinct.

Now note that exactly one clause in F is positive for a satisfying model of F, ie:- if ℓ_0 is the **maximum**[6] size of a pairwise 1-intersecting k-family, then $F_{\ell_0} \wedge \neg F_{\ell_0+1}$ is positive, and no other clause is.

Thus a satisfying assignment yields to us the maximum size of the desired family, as well as a possible family itself, which can directly be read off from the values of the propositional variables.

Exercise 13.7

Refer to Figure 2, Figure 3.
Credits to Amit Rajaraman for this code.

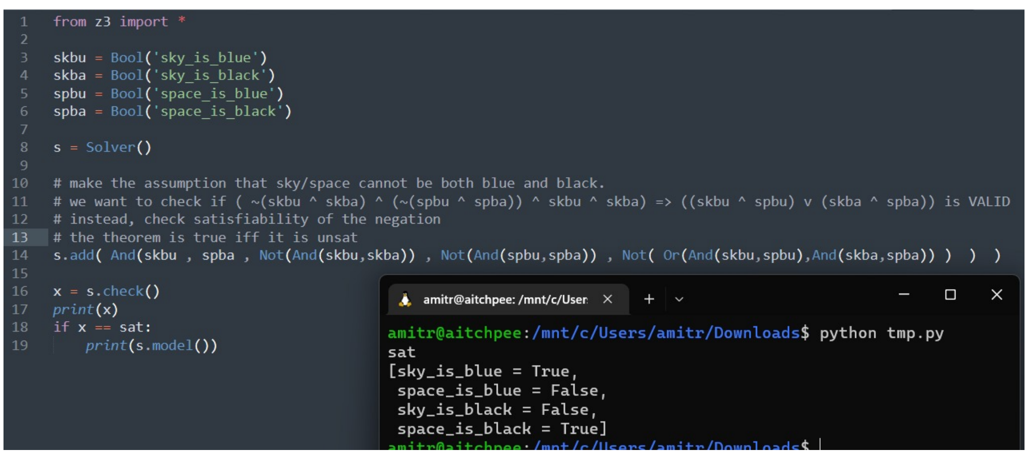

Figure 2: Exercise 13.7

Exercise 13.8

Draw a parse tree of the formula, and carry out a standard tree traversal of the parse tree, keeping a count of how many $\neg$ we have encountered so far.

[6] note that we were only asked to find a maximal family. We went ahead and found a maximum one, which is also obviously maximal

Figure 3: Exercise 13.7

Whenever we reach a leaf, if the number of $\neg$ seen so far is even, then that leaf occurs positively, otherwise not. A variable p occurs positively iff every leaf corresponding to p occurs positively.

6 Tutorial 6

Exercise 14.6

Let m be any model and let ν be any assignment on m.

(a) If $m, \nu \models \neg\forall x.P(x)$, then $m, \nu \not\models \forall x.P(x)$, implying there exists a $u \in D_m$ such that $m, \nu[x \to u] \not\models P(u)$, which further implies that $m, \nu[x \to u] \models \neg P(u)$. But then $m, \nu \models \exists x.\neg P(x)$ by the definition of semantic satisfaction of $\exists$.
Similarly, if $m, \nu \models \exists x.\neg P(x)$, then there exists a $u \in D_m$ such that $m, \nu[x \to u] \models \neg P(u)$. But then u witnesses $m, \nu \models \forall x.P(x)$ being violated, and thus we have $m, \nu \not\models \forall x.P(x)$.

(b) DIY

(c) If $m, \nu \models (\forall x.(P(x) \wedge Q(x)))$ then for any $u \in D_m$ we have $m, \nu[x \to u] \models P(u) \wedge Q(u)$ which implies $m, \nu[x \to u] \models P(u)$ and $m, \nu[x \to u] \models Q(u)$. But then by the semantic definition of satisfaction of $\forall$, we have $m, \nu \models \forall x.P(x)$ and $m, \nu \models \forall x.Q(x)$, implying $m, \nu \models (\forall x.P(x)) \wedge (\forall x.Q(x))$.
Conversely, if $m, \nu \models (\forall x.P(x)) \wedge (\forall x.Q(x))$, then $m, \nu \models \forall x.P(x)$ and $m, \nu \models \forall x.Q(x)$, which implies that for any $u_p, u_q \in D_m$ we have $m, \nu[x \to u_p] \models P(u_p)$, $m, \nu[x \to u_q] \models Q(u_q)$. In particular for any $u \in D_m$, we can choose $u_p = u_q = u$ to obtain that $m, \nu[x \to u] \models P(u) \wedge Q(u)$, which yields $m, \nu \models \forall x.(P(x) \wedge Q(x))$, as desired.

(d) DIY

Exercise 14.11

(a) $h(c)$

(b) $f(h(x), h(y), h(c))$

(c) $g(x, c)$

(d) $f(x)$

Exercise 14.12

Note that for the algorithm we shall treat constant functions as variables. Furthermore, instead of building the parenthesized formula directly, we shall first build the parse tree of the formula, and then generate the formula from it. Then note that the unparenthesized expression is just the DFS traversal ("infix" style) of our parse tree, from which we need to generate the tree, which is a classic well-known algorithm.

Exercise 15.8

Consider the signature $\mathbf{S} := (\{\}, \{< /2\})$, and consider the sentence

$$F := (\forall x. \neg(x < x)) \wedge (\forall x. \forall y. x < y \Rightarrow \neg(y < x)) \wedge (\forall x. \forall y. \forall z. (x < y) \wedge (y < z) \Rightarrow (x < z))$$

$$\bigwedge (\forall x. \exists y. x < y)$$

on any model of $\mathbf{S}$.
Effectively F ensures that $<$ is a total order on the domain of our model, and then the last clause says that our domain is unbounded, ie:- for every x we have a y greater than x. Now, if our model is finite (ie:- has a finite domain), then any total order will necessarily entail a maximum element, and thus $m \not\models F$ for any finite model m.
One example of a satisfying (infinite) model is $m_{\mathbb{R}} := (\mathbb{R}; \{\}, \{< /2\})$ with the predicate $<$ coinciding with the usual order on $\mathbb{R}$. Also, note that not every infinite model satisfies F: For example, $m_{[0,1]} := ([0, 1]; \{\}, \{< /2\})$, where $<$ is the usual order on $\mathbb{R}$, doesn't satisfy F since it has a maximum element, 1. Thus $m_{[0,1]} \models \neg F$.

Exercise 15.9

(a) $F := \forall x. \forall y. \forall z. ((x = y) \vee (y = z) \vee (z = x))$. Clearly if $m \models F$, then $|D_m| \leq 2$.

(b) If we are forbidden from using $=$, we prove that we can no longer restrict the satisfiability of our sentences to models of size at most 2. We go about this as follows: Take any signature $\mathbf{S}$, and consider any model m on it, such that $0 < |D_m| \leq 2$. Then we construct a model m' on $\mathbf{S}$ such that $|D_{m'}| \geq 3$, and for any sentence F, $m \models F \iff m' \models F$.

The construction of m' proceeds as follows: WLOG we assume that $D_m \subset D_{m'}$ [7]. We choose an arbitrary element $a \in D_m$, and we define a function $\mu : D_{m'} \mapsto D_m$ such that

$$\mu(x) := \begin{cases} x, \text{if } x \in D_m \\ a, \text{otherwise} \end{cases}$$

Then, for every function f/n in $\mathbf{S}$, we define

$$f_{m'}(x_1, x_2, \ldots, x_n) := f_m(\mu(x_1), \mu(x_2), \ldots, \mu(x_n))$$

and similarly, for every relation R/k in $\mathbf{S}$ we define

$$R_{m'} := \{(\mu(x_1), \mu(x_2), \ldots, \mu(x_k)) : (x_1, x_2, \ldots, x_k) \in R_m\}$$

We proceed by induction: If F doesn't contain any quantifier, then F contains only constants and 0-ary predicates. By definition, m and m' agree on them. Now, we induct on the maximum depth of the parse tree of F. Note that WLOG we can assume that we have only $\exists$ in our sentence: Indeed, wherever we see a $\forall$, we can replace it by $\neg\exists\neg$. Similarly, we can assume WLOG that $\wedge, \neg$ are the only propositional connectives which occur in our sentence. Then, if the root connective of F

(a) Is $\exists$, then $F = \exists x.F_0(x)$. Then $m \models F$, if and only if we have $u \in D_m$ such that $m, \nu[x \mapsto u] \models F_0(u) \iff m', \nu[x \mapsto u] \models F_0(u) \iff m' \models \exists x.F_0(x)$, as desired.

(b) Is $\neg$, ie:- $F = \neg F_0$. Then $m \models \neg F_0 \iff m \not\models F_0$. By the induction hypothesis, $m' \not\models F_0 \iff m' \models \neg F_0$, as desired.

(c) Is $\wedge$, ie:- $F = F_1 \wedge F_2$. Then $m \models F_1 \wedge F_2 \iff m \models F_1 \wedge m \models F_2 \iff m' \models F_1 \wedge m' \models F_2$, by the induction hypothesis. But then $m' \models F_1 \wedge F_2$, as desired.

Exercise 15.12

Similar in nature to Exercise 14.6.

6.1 Extra questions

Exercise 14.4

Consider FOL with only relations (ie:- no functions, and no constants in particular) and **no** quantifiers.

Then this restricted FOL is equivalent to PL, where the propositional variables in our PL are of the form $P(x_1, x_2, \ldots, x_n)$ or $x_i = x_j$ where x_is are FOL variables and n is the arity of P. The $\top, \bot$ of FOL carry over as it is to PL.

[7] note that domains by themselves are just sets with no additional structure

For example, the FOL formula $F := (P(x, y) \wedge P(x, z)) \implies (y = z)$ translates to $(\alpha \wedge \beta) \Rightarrow \gamma$, where α, β, γ are PL variables representing $P(x, y), P(x, z)$ and $y = z$ respectively.
Finally, note that the model $m := (\mathbb{N}; P_m/2)$ satisfies F, where $P_m(x, y)$ is defined to be "Is $x = y^2$?".