CS3243 Tutorial 8

Eric Han

Oct 25, 2022

Annoucements

1. Assignment 6 scores are now on Luminus.
2. Highly recommended to send in your teaching feedback - I appreciate it greatly!

Student Feedback on Teaching (SFT)

Feedback is *optional* but *highly encouraged*, access here: **https://es.nus.edu.sg/blue/**

- [**Tutorial Feedback**] Your feedback is important to me, and will be used to improve my teaching.
  - If I have helped your learning in any way, your positive feedback will be an encouragement to me.
  - If you find your learning can be enhanced by some action on my part, that feedback will be used to improve my teaching.
- [**Module Feedback**] Your feedback will be used to improve the module.
- Feedback is confidential to the university and anonymous to us.
- Avoid mixing the feedback; ie. project feedback to tutorial feedback.

Past student feedback had been used to improve teaching; ie. Telegram access to provide faster feedback. I would greatly appreciate your feedback, especially this is my first time teacing AI.

Previously from T07, Q3

Given the following logical statements, use truth-table enumeration to show that $KB \models \alpha$. In other words, write down all possible true/false assignments to the variables, the ones for which KB is true and the one for which α is true, and see whether one is a subset of the other.

a. $KB = (x_1 \lor x_2) \land (x_1 \implies x_3) \land \neg x_2, \quad \alpha = x_3 \lor x_2$
b. $KB = (x_1 \lor x_3) \land (x_1 \implies \neg x_2), \quad \alpha = \neg x_2$

Recap

What does $KB \models \alpha$ mean?

. . .

- When KB is true $\implies$ α is true
- Use resolution algorithm

Answers

In order to show that $KB \models \alpha$ we need to show that whenever KB is true, so is α.

Use resolution, without using truth tables (eg. part a)

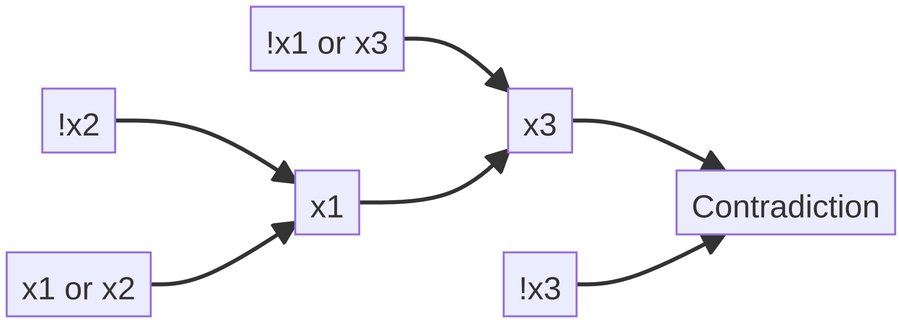
- $\neg\alpha = (\neg x_3) \land (\neg x_2)$
- $KB = (x_1 \lor x_2) \land (\neg x_1 \lor x_3) \land (\neg x_2)$

x_1	x_2	x_3	KB	α
$True$	$True$	$True$	$False$	$True$
$True$	$True$	$False$	$False$	$True$
$True$	$False$	$True$	$True$	$True$
$True$	$False$	$False$	$False$	$False$
$False$	$True$	$True$	$False$	$True$
$False$	$True$	$False$	$False$	$True$
$False$	$False$	$True$	$False$	$True$
$False$	$False$	$False$	$False$	$False$

Figure 1: T07 Q3a Answer - $KB \models \alpha$

x_1	x_2	x_3	KB	α
$True$	$True$	$True$	$False$	$False$
$True$	$True$	$False$	$False$	$False$
$True$	$False$	$True$	$True$	$True$
$True$	$False$	$False$	$True$	$True$
$False$	$True$	$True$	$True$	$False$
$False$	$True$	$False$	$False$	$False$
$False$	$False$	$True$	$True$	$True$
$False$	$False$	$False$	$False$	$True$

Figure 2: T07 Q3b Answer $KB \not\models \alpha$



Question 1

Assignment Question; Also solve this as bonus!

Hint

We didn't specify the order so take the fastest and most direct path.

Please dont go in circles!

Question 2

Given inference algorithm $\mathcal{A}$ and knowledge base KB with variables $x_1, \cdots, x_n$; proof that it is sound, $KB \vDash_{\mathcal{A}} \alpha \implies KB \vDash \alpha$.

Let $M(q)$ be the set of all truth assignments to variables for which logical formula q is true.

Recap

- What is a KB?
- What does entailment $KB \vDash \alpha$ mean?
- What is inference $KB \vDash_{\mathcal{A}} \alpha$?

Further understanding of $M(.)$

$KB = (a \vee b) \wedge (\neg a \vee c)$, which resolves to $b \vee c$. Entailment does not mean equivalence!

Table 1: Showcase of $M(KB) \subseteq M(b \vee c)$

a	b	c	$b \vee c$	KB
1	1	1	1	1
1	**1**	**0**	**1**	**0**
1	0	1	1	1
1	0	0	0	0
0	1	1	1	1
0	1	0	1	1
0	**0**	**1**	**1**	**0**
0	0	0	0	0

Answer

Assume wlog, KB clauses are in CNF and $T \in \{1, 0\}^n$ is any satisfying truth assignment for KB. $\mathcal{A}$ obtains α after a series of R resolution operations.

Induction case

Let KB^r to be the set of resolvents reached from KB after r steps. Let $p \vee x_i$ and $q \vee \neg x_i$ to be 2 clauses in KB^r. The resolution operation will yield $p \vee q$ which is appended to create KB^{r+1}.

Truth assignment T will satisfy both $(p \vee x_i) \wedge (q \vee \neg x_i)$:

- If x_i true, q must be true
- If x_i false, p must be true

Hence, $p \vee q$ must be true for any T; which means $M(KB) \subseteq M(p \vee q)$.

So, at R step, it will produce $M(KB) \subseteq M(\alpha)$ which completes the proof $KB \vDash \alpha$.

Question 3

k-CNF formula is one where each clause contains at most k literals. Show that every k-CNF formula can be converted to a 3-CNF formula.

Recap

- What is the resolution algorithm?
- How can we run it 'in the other direction' ?

Answer

All we need to do is to come up with an algorithm to convert k-CNF to 3-CNF. Specifically, we need to convert a clause from k terms 3-CNF:

1. Input clause $C = (\ell_1 \vee \cdots \vee \ell_q)$
2. If $q > 3$,
    1. Invent new variable y_i
    2. Add $(y_i \vee \ell_1 \vee \ell_2)$ to 3-CNF clauses
    3. Add $C' = (\neg y_i \vee \ell_3 \vee \cdots \vee \ell_q)$ to k-CNF clauses
3. If $q <= 3$, add $(\ell_1 \vee \cdots \vee \ell_q)$ to 3-CNF clauses

Algorithm Terminates

For every a set of k-CNF, we pick a clause C:

- Maximally reduce the k-CNF clauses by 1.
- Minimally reduce the k-CNF clauses by 0.
    - Reduce the clause C from length of q to C' of length $q - 1$.

It will reduce clause C eventually to a size of 3 and then reduce number of clauses by 1.

Algorithm Correctness, via induction

Assuming that $T \in \{1, 0\}^n$ is any solution to C. In every step with C,

- If $q > 3$,
    - If $(\ell_1 \vee \ell_2)$ is true and $(\ell_3 \vee \cdots \vee \ell_q)$ is false $\implies y_i$ is true
    - If $(\ell_1 \vee \ell_2)$ is false and $(\ell_3 \vee \cdots \vee \ell_q)$ is true $\implies y_i$ is false
    - Otherwise it does not matter.
- If $q <= 3$, T also satifies $(\ell_1 \vee \cdots \vee \ell_q)$.

At every step, we can find an assignment to y_i such that the solution is preserved.

Question 4

Show that the resolution procedure for CNF formulas described in class yields a polynomial time algorithm for deciding whether a 2-CNF formula is satisfiable.

Recap

- What is resolution?
- How does it decide satisfiablity?

Answer

Each 2-CNF can be

1. Rewritten to an implication, ie. $a \lor \neg b \equiv b \implies a$
2. Resolved with
    1. Literal to another Literal
    2. 2-CNF to another 2-CNF
        1. Transitivity, ie. $(a \lor \neg b \equiv b \implies a) \land (\neg c \lor b \equiv \neg c \implies b) \equiv \neg c \implies a$
        2. Useless clauses, ie. $b \lor \neg b$

Lemma 1.

If there exists a cycle that connects a node x with a node $\neg x$ then the CNF formula is not satisfiable.

Note in class, we consider a weaker version of the lemma 1 that reaches $\neg x$ from x without considering cycles then x is not a possible assignment; ie. x back to $\neg x$.

Key insight from Lemma 1 - each implication does not give a choice;

For example:

1. Lets consider both cases: $a \implies b; a \implies \neg b$
2. Contrapositive $\neg b \implies \neg a; b \implies \neg a$

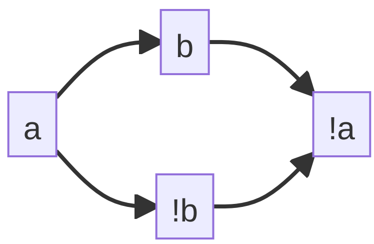


Figure 3: Impossible assignment!

When a, we do not need to consider both b and $\neg b$! This is the part where the combinatorial explosion is eliminated.

Algorithm Sketch

1. For example, $(a \lor \neg b) \land (c \lor d) \land (b \lor \neg c) \land (\neg a \lor d)$ yields
2. $[O(n^2)]$ $b \implies a; \neg c \implies d; c \implies b; a \implies d$ and
3. $[O(n^2)]$ the contrapositive $\neg a \implies \neg b; \neg d \implies c; \neg b \implies \neg c; \neg d \implies \neg a$

The implication graph is directional - to check that $\neg d$ is not a possible assignment, check that it can reach d in the graph; Choose $2n$ literals (+ve, -ve) to start from and each literal can take implications n^2. There are no further choice for every implication.

Bonus - Further Understanding - P=NP?

- 2-SAT is in P
- 3-SAT is NP-Complete
    - Written as $a \implies b \lor c$, the possiblity of $b \lor c$ causes combinatorial explosion.

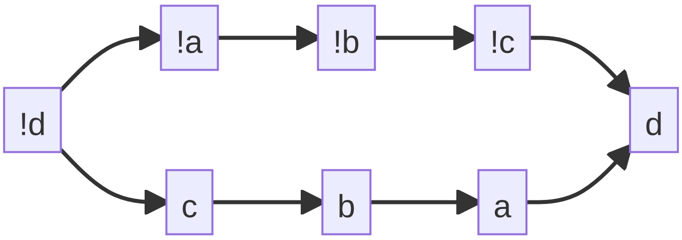


Figure 4: Implication Graph of the statement.

If you can prove that

- a P time algorithm to convert 3-SAT to 2-SAT.
- or no P time algorithm can exist.

Millennium Prize: https://www.claymath.org/millennium-problems/p-vs-np-problem