CS 228 TSC

Rishabh - Full Session (and memes)
Hari - Moral Support
Dion - Telling us what stuff to teach

Horn Formulae

- Each clause has at most one positive literal
- Easy to find satisfiability
- For example consider $F = (p \land (\neg p \lor \neg q \lor r) \land (\neg q \lor \neg r) \land (q \lor \neg r) \land q)$
- We rewrite the clauses as $(T \rightarrow p)$, $((p \land q) \rightarrow r)$,$((q \land r) \rightarrow \bot)$,$(r \rightarrow q)$,$(T \rightarrow q)$
- Now, mark p and q, which forces you to mark r
- Since both q and r are marked, $((q \land r) \rightarrow \bot)$ is UNSAT
- Hence, the entire formula is UNSAT

Create horn clauses of this : $(x \lor y \lor \neg z) \land (z)$

A Simple Horn Problem

Scenario: You are designing an expert system to diagnose hardware faults in a computer system. Each symptom has a cause, and there are some mutual exclusions. Use Boolean Horn formulae to describe the relationships between symptoms and possible causes.

- s1: System is overheating.
- s2: Memory is corrupt.
- s3: Disk is failing.
- s4: Power supply is unstable.
- s5: CPU is malfunctioning.

Constraints:

1. If the system is overheating, the CPU must be malfunctioning
2. If the disk is failing, the power supply must be unstable:
3. Memory corruption implies either the CPU is malfunctioning or the disk is failing
4. The system cannot overheat if the power supply is unstable
5. CPU malfunction and disk failure cannot both occur

DPLL

- An algorithm to find satisfiability of a formula in CNF
- Three actions - decisions, unit propagations, backtracking
- Unit Propagation - Lets say there's a clause (x, ¬y, ¬z) and y and z have already been assigned 1. For this clause to be satisfied, x must obviously be assigned 1
- Decisions - When there's nothing to unit propagate on, we must make "decisions" on variables, for example assigning an unassigned variable as true or false
- Backtracking - If we made a wrong decision we need to go back and reverse it. We come to know that we made a wrong decision because our formula under that partial assignment will become UNSAT

A Question on DPLL

In this question we will be exploring an algorithm that improves significantly upon DPLL, you will have to understand and prove some properties of the algorithm

Consider the formula made of the following Clauses :
({-p1,p2},{-p1,p3,p5},{-p2,p4},{-p3,-p4},{p1,p5,-p2},{p2,p3},{p2,-p3,p7},{p6,-p5})

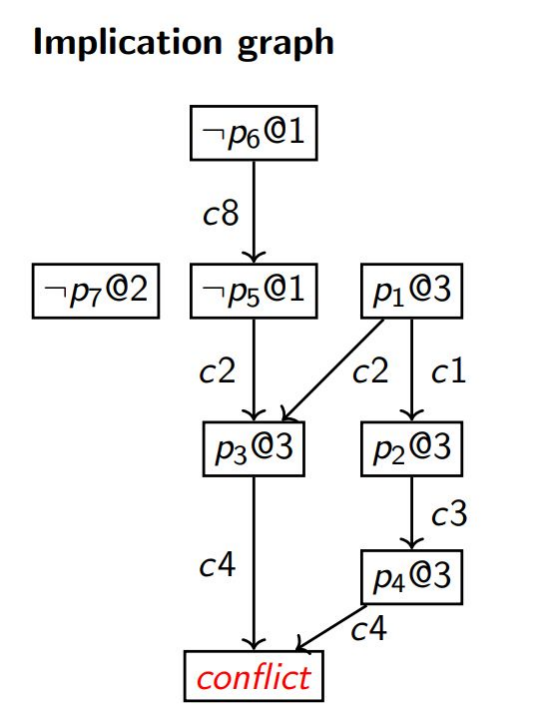
Q1] Run DPLL on the above formula

Call a partial model that has been formed by DPLL, a run of DPLL. So for example, assume that in our formula, assume the first decision is p6 is assigned 0, by UP, p5 becomes 0, lets say we decide p7 to be 0, then we decide p1 to be 1

Q1] Using only UP, state if the resultant formula is SAT or not?

Q2] What is the reason for this formula being SAT (or UNSAT ;))

Using the Clauses, we can make an implication graph



CDCL

- We traverse the graph backwards and find the set of decisions that caused the conflict
- We add conflict clause in the input clauses, backtrack to the second last conflicting decision, and proceed like DPLL

Questions :

1. Why does adding the conflict clause not change satisfiability? (Prove)
2. Prove that CDCL always terminates (HW, highly non-trivial)

First Order Logic

- Signature : Defines the symbols that are going to be used
- Structure : Gives the symbols meaning
- Assignment : Gives meaning to variables
- For example, define your signature = {P,Q,f} where P and Q are binary relations while f is a function
- Define $F = \forall x[\exists y\{P(x,z) \rightarrow \exists z[Q(x,z) \rightarrow P(y,f(z))]\} v Q(f(y),x)]$

Q1] Find and point out free variables

Q2] Write a semantically equivalent Prenix Normal Form Formula and an equivalent Skolem Form Formula

Q3] Give a structure+assignment over free variables that satisfies F and one that does not

FOL Over Words / First Order Languages

Given a sentence F over words there could be infinitely many words satisfying F.

First lets start with the basics:

1]Write a sentence whose language is the set of all words beginning with a

2]Write a sentence whose language is the set of all words with k^{th} letter being a

3]Write a sentence whose language is the set of all words ending with a

Bit more complex?

1]Write a sentence whose language is the set of all words with prefix being “aba”

FOL Over Words / First Order Languages

Bit more complex?

Write a sentence whose language is the set of all words with “cbabc” occurring somewhere

Some PTSD

Write a sentence whose language is the empty set (shouldn’t even accept $\in$;))

Thoda aur mushkil

Write a sentence whose language is the set of all words such that:

1] The word begins with an “ac” and

2] For every subsequent occurrence of “a”, a “b” follows

FOL Languages

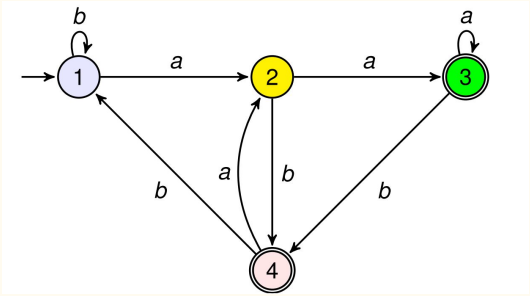
Given formulae, make DFA:

$[\exists x\{first(x) \Lambda (Q_a(x) \text{ v } Q_c(x))\}] \Lambda [\exists x\{second(x) \Lambda (Q_b(x))\}]$

$\Lambda [\forall y\{\neg first(y) second(y) \rightarrow (c_1(y) \text{ v } c_2(y))\}]$

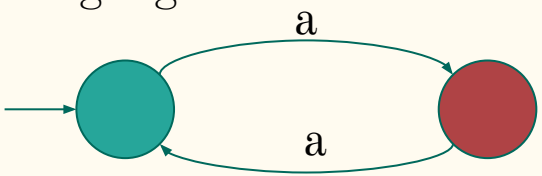
$\Lambda [c_1(y) \leftrightarrow \{Q_a(y) \Lambda \exists x(S(y,x) \Lambda Q_b(x))\}] \Lambda [c_2(y) \leftrightarrow \{Q_b(y) \Lambda (S(y,z) \rightarrow Q_a(z))\}]$

Given DFA make formula:



DFA

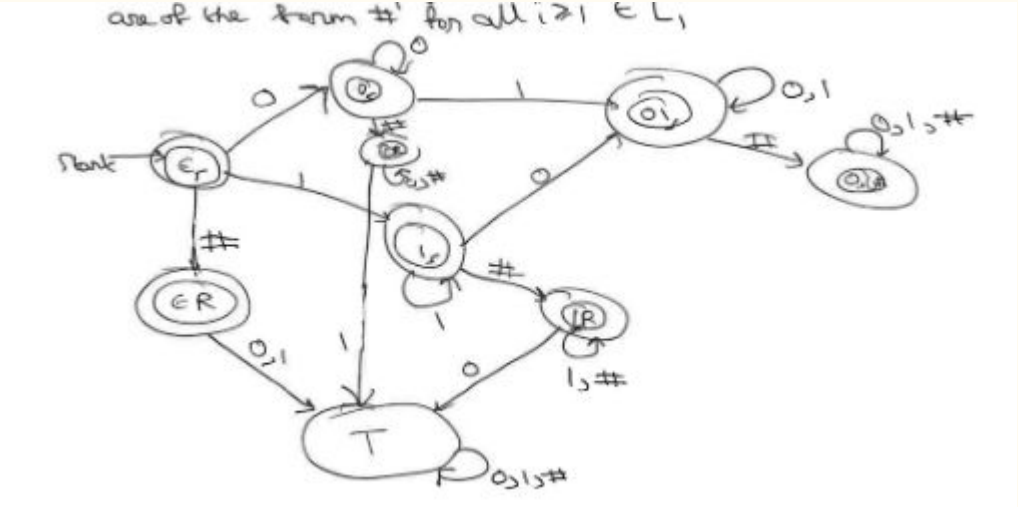
Make the sentence corresponding to the following DFA (difficulty literally impossible, assuming green is accepting state and alphabet is only ‘a’) What language does this DFA accept?



Draw a DFA for the following:

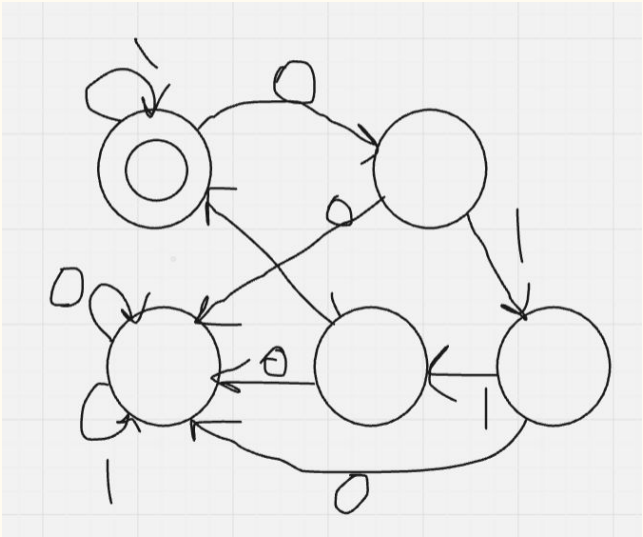
Let $\Sigma_n = \{0, 1, 2, 3, \ldots n\} \cup \{\#\}$, where $n \geq 1$. Define $L_n = \{w \in \Sigma^*_n \mid$ every non # symbol that appears after the first # in w also appears before the first # in w$\}$. For example, $012\#0\#21 \in L2$, but $01\#021 \not\in L2$

Hidden slide : Previous ka answer



Regular Languages

Describe the Language accepted by the following automata:

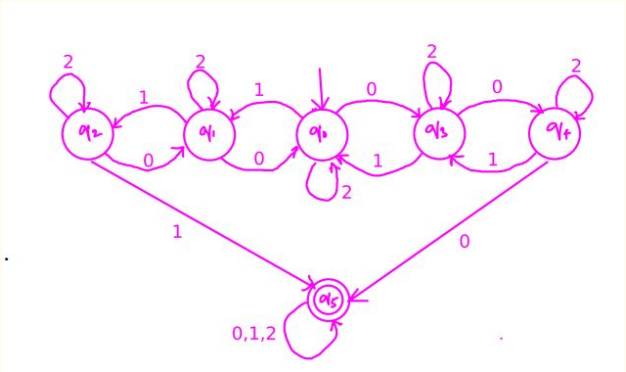


Regular Languages

Create a DFA that accepts the set of all words that have their last letter match with their first letter

Given two DFAs - D1 and D2, describe how you would build a DFA D3 whose language is the union of the languages of D1 and D2

What is the Language of the following automata?



Expressive Power and Invariants

We know ¬ and ∧ can express all possible formulae.

Can ∧ alone do this ?

The answer is no, but how to prove it??

We will show that we cannot represent ¬p

Is it trivially true? Because p ∧ p is the only thing we can do with ∧.

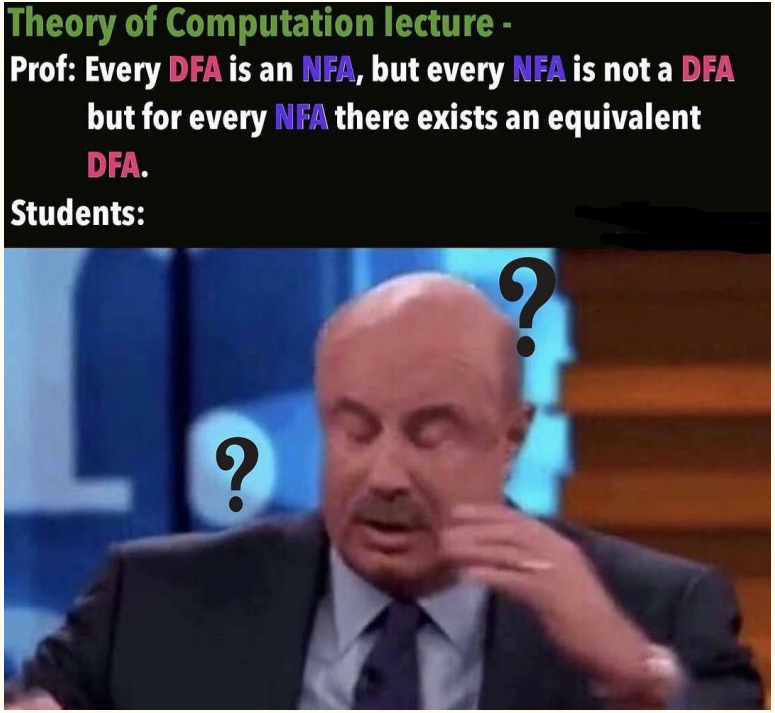
This proof is incorrect because we are only showing p∧p is not ¬p, what if there was some weird combination which worked.

So we use an invariant,

Claim : Given any formula F using only ∧, if we set all occurring variables to be true it must evaluate to true.

Proof - Induction

Random memes



Some questions

Let φ_1 and φ_2 be unknown propositional formulas over $x_1, x_2, \ldots x_n$. Consider the following implications labelled (1a), (1b) through (na), (nb). A solution to this set of simultaneous implications is a pair of specific propositional formulas (φ_1, φ_2) such that all implications are **valid**, i.e. evaluate to true for all assignments of variables.

$$\begin{array}{ll|ll} (1a) & (x_1 \wedge \varphi_1) \to \varphi_2 & (1b) & (x_1 \wedge \varphi_2) \to \varphi_1 \\ (2a) & (x_2 \wedge \varphi_1) \to \varphi_2 & (2b) & (x_2 \wedge \varphi_2) \to \varphi_1 \\ & \vdots & & \vdots \\ (na) & (x_n \wedge \varphi_1) \to \varphi_2 & (nb) & (x_n \wedge \varphi_2) \to \varphi_1 \end{array}$$

In each of the following questions, you must provide complete reasoning behind your answer. Answers without reasoning will fetch 0 marks.

1. *[5 marks]* Suppose we are told that φ_1 is $\bot$. How many semantically distinct formulas φ_2 exist such that implications (1a) through (nb) are valid?
2. *[2 marks]* Answer the above question, assuming now that φ_1 is $\top$.
3. *[5 marks]* If we do not assume anything about φ_1, how many semantically distinct pairs of formulas (φ_1, φ_2) exist such that all the implications are valid?
4. *[3 marks]* Does there exist a formula φ_1 such that there is exactly one formula φ_2 (modulo semantic equivalence) that can be paired with it to make (φ_1, φ_2) a solution of the above implications?

Some questions

Let n and k be integers so that $n > 0$, $k \geq 0$ and $k \leq n$. You are given n booleans x1 through xn and your goal is to come up with an efficient propositional encoding of the following constraint:

$x1 + x2 + ... + xn \leq k$ (1)

In the above equation, assume the booleans xi behaves as 0 when false and 1 when true. We can arrive at the encoding by adding $O(n \cdot k)$ auxiliary variables and only need $O(n \cdot k)$ clauses. We try to solve this problem iteratively.

- Let si,j denote that at least j variables among x1,...,xi are assigned 1 (true). One can deduce that si,j makes sense only when $i \geq j$. We now try to represent this constraint in propositional logic.

- Now given that you have represented si,j for all appropriate $1 \leq i \leq n$ and $0 \leq j \leq k$; come up with a propositional formula that represents the condition of the equation (1)

Some Questions

Tseitin encoding - given a single clause with n variables, how many extra variables to you need to reduce it to a clause with 3 variables?

What if you were only allowed logarithmically extra variables? How much can you break down a clause?

PL-definable languages

In this question we will view the set of assignments satisfying a set of propositional formulae as a language and examine some properties of such languages. Let P denote a countably infinite set of propositional variables p0, p1, p2,.... Let us call these variables positional variables. Let Σ be a countable set of formulae over these positional variables. Every assignment $\alpha : P \rightarrow \{0,1\}$ to the positional variable can be uniquely associated with an infinite bitstring w, where $wi = \alpha(pi)$. The language defined by Σ- denoted by $L(\Sigma)$- is the set of bitstrings w for which the corresponding assignment α, that has $\alpha(pi) = wi$ for each natural i, satisfies Σ, that is, for each formula $F \in \Sigma$, $\alpha \models F$. In this case, we say that $\alpha \models \Sigma$. Let us call the languages definable this way PL-definable languages.

PL-definable languages

Prove that they are closed under a countable (possibly infinite) union

Prove that they are closed under a finite intersection
(what comes to mind to prove this fact immediately, can we use this to prove for the first fact)

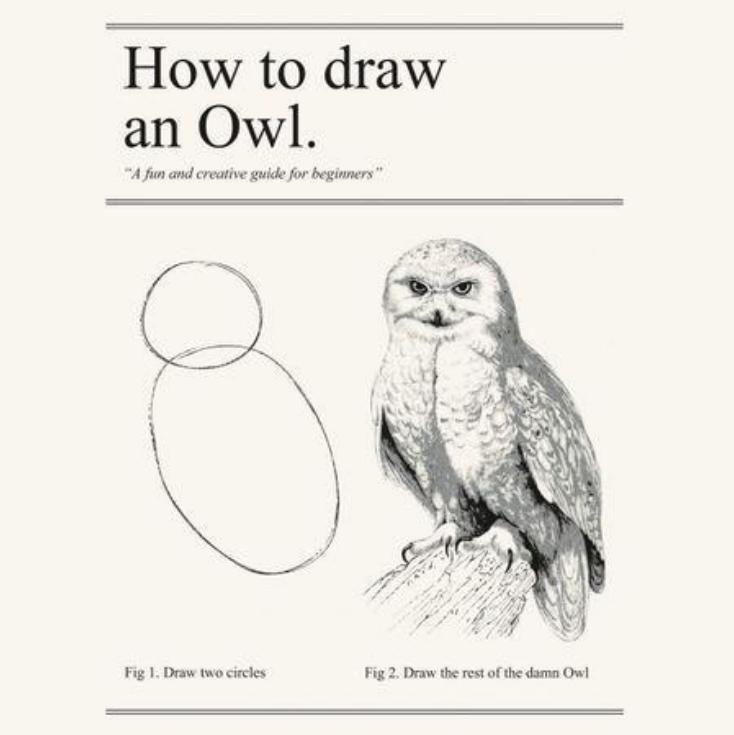
Doing DLDCA nonsense in a DFA

$\Sigma 3 = \{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1)\}$ Each letter of the alphabet is a 3-tuple of bits. Consider the language formed by words in this language that satisfy the following property:

If all the X-coordinates of the letters are considered as a binary number NX in reverse, and all the Y-coordinates of the letters are considered as a binary number NY in reverse, and all the Z-coordinates of the letters are considered as a binary number NZ in reverse, then NX + NY = NZ.

For example, the word (0,0,0)(0,1,1)(1,0,1) is in the language, since we have NX = 100 = 4, NY = 010 = 2 and NZ = 110 = 6. However, the word (0,0,0)(0,1,1)(1,1,0) is not in the language, since NX = 1002 = 4, NY = 1102 = 6 but NZ = 0102 = 2. 1. Prove that this language is regular, and draw a DFA for it. You can think of words accepted by this DFA as denoting the ”solutions” to the equation NX + NY = NZ

TSC be like


How to draw
an Owl.
"A fun and creative guide for beginners"
Fig 1. Draw two circles
Fig 2. Draw the rest of the damn Owl