Knowledge Representation Lecture 3

Knowledge Representation (Vrije Universiteit Amsterdam)

KR 2021 – Lecture 3

PROPOSITIONAL LOGIC

Defining syntax of PL

- BNF helps contruct PL
- Parse the input (decide what the tool is) then you can use BNF to see what language to allow formulas
- Now you know what syntax is good

Semantics (reminder)

- Model: an assignment of truth values to assignment of propositional variables that make a formula/sentence/KB (one of them) true
- Entailment: calculate assignment that are consistent with knowledge base then check if statement is true
  - Something follows from
    - I have a set of sentences in KB and this implies something
    - Satisfiable makes it mathematically true

Semantics vs Calculus

- We want to show for PL a sentence….
  - Abbreviated with twoline S
    - Model is satisfiable
- DPLL does exactly same
  - Abbreviated with oneline S
  - It says this formula is satisfiable then it should be satisfiable but just because your algorithm says something does not make it true so then you apply DPLL and see if it the same in semantic world
  - Intuitively, it does the same but we have to prove mathematically that it does

DPLL Complexity

- The cost in terms of time
- The difference between TT and DPLL
  - N is the number of variables so it is faster to use DP, still high but better

KB

- Is a set of facts that we assume to be true

Definition: Properties of Inference

- Notion of soundness, completeness, termination and decidability
- If our algorithm DP (Oneline S) says statement S is satisfiable (yes answer) then it should be the same as semantic notion (Twoline S)
  - Inverse = complete
- We need to show that there is no loop, and thus it terminates = shows that we found a decidable calculus
- Decidability can also be shown without an algorithm but the easiest is with algorithm

Properties of DPLL based SAT solving

- You start with partial assignment, you check if it satisfies all the partial clauses if doesn't yet then it start assigning more and more values. If it at the end it finds out there is conflict with clauses it returns no (conflict between clauses & assignment; not satisfiable), whereas if says it satisfies all the clauses this returns yes.
    - If it yes = it means it found a model
- This is a trivial thing because what we do is in the algorithm, we construct the model

Proof(sketch) for DPLL based Entailment

- You know there is an assignment does DL really find it?
- Suppose you had an extra rule (besides pure, unit) it could be telling us something wrong. But of course this algorithm has been around a long time
- Backtracking- you try another value so due this you go through all options and then you can find a model that satisfies all clauses

Notion of Model

- Is not defined by humans it is mathematically defines by assignment of truth value to PL variables that make all constraints true
    - What does it mean that it is satisfiable? What does it mean that the algorithm calculates?
- Worst case you assign two values per variable youll never go back do it again because finite variable so you will finish in finite time

Proof of Entailment by Refutation

- Satisfiability: DP only says yes if only the formula is satisfiable
- Entailment: You have a set of KB, you want to know if something follows logically from it
- Refutation is a trick to check entailment
- You want to derive fact a from KB
    - You assume KB is true
    - But then you negate it and proof that it is impossible
        - Show there is no model → unsatisfiable
            - With CNF of a & KB
    - If this is possible then you know that a is entailed by KB

Semantics vs Calculus

- Only true if algorithm says, the algorithm is DP on negation
- Do the same stes for sound, complete and terminates
    - Alpha follows from algorithm of KB is and only iff it entails
    - Proof of refutation
        - You can even calculate if something follows from a KB if it is satisfiable

PROPOSITIONALISATION

- Sudoku is a hard thing to model because you need 800 axioms to explain the rule of the game which is tedious, it is more elegant to model this in a different way

Propositional logic is weak language

- First order: you can say something exists, quantify variable
- Problem with PL is that it is weak, you can use it in the first part of course because it is simple but in practise it is annoying
- Model can be nicer in more expressive logic then you can boil it down to PL
- You cannot talk about individuals (e.g., Mary is woman)
- Then you use FOP but it is expensive in computation, there is no guarantee of termination

Propositionalising FOL over finite domains

- If you have finite domains, it is sometimes convenient to model in FOL, then you would have to write it down for everyone that you are working with in the domain
- When you propositionalise (translate & reason) then you have large problem, whereas the clauses might be simple (pg19) because you have to apply formula in everywhere (model explicitly)
- This doesn't work with infinite domains

Problem solving with propositionalisation?

- Issue: if you now in your FOL has 10 variables so then 1618^2^^n
  - Explosion
  - Things don't get easier computation-wise

HARD SATISFIABILITY PROBLEMS

Structured vs Random Problems

- In research you can use structured problems that were randomly created that can be used for empirical studies
- Sudoku = structured problem
  - Sounds complex 81 boxes, 9 variable 700 something, 800 axioms, theoretically impossible to solve but it turns out relatively easily because of the structure of the problem
  - Unit clauses are all true, so you can resolve a lot of clauses by this rule
    - Making it a smaller problem
- Some hard problems that have been randomly produced
  - Very often of clauses length three

Random 3sat model

- You plot thousands of generated problem (runtime)
  - Depending on the ratio of clauses to variable, what is the computation time?
  - Colour red:
    - unsatisfiable
  - Colour blue:
    - satisfiable
  - You want high instances (high computation time)

  - Turns out you have a moment with ‘l’ clauses per ‘n’ variables, polarity is randomly distributed
    - How easy it to find an assignment for those?
  - The more you move to the right, the more clauses you get, meaning you have a chance of higher contradiction

Hardness of 3SAT

- This plot checks backtracks in DP so robust again RT
- This is what you see in many different experiments, around 3-5 you get the hardest instances
  - That is a sweet spot
- Identifies difficult problems and what is their properties? You have a good measure to know the problem is difficult and gives you intuition why it happens at this specific point

The magic ratio 4.6

- The chance if KB being satisfiable is 50-50 around 50% so it is the most uncertain then
- When you go more to right (more) clauses, you know there are contradictions thus satisfiability but you find out the mistakes quickly (you need no backtracking due to contradiction immediately)

Phase Transition

- You need to be careful with clauses (ratio)
- Move from unsatisfiable problems to satisfiable ones
- We are studying this point

INCOMPLETE SAT METHODS

Local Search for SAT

- Find solution quickly even worst case but not finding a solution in 1% is also possible because it is stochastic
- It is not intended to find ALL solution or prove satisfiability (no solution)
  - Then incomplete wont make sense
- It makes sense when you find one solution
- It simple makes a guess (assign values T/F for set of variables) then you test how you are doing, otherwise flip values that make things better (greedy search algorithm)
  - Repeat till satisfied
- If a,b,c are all true, which clauses are true?
  - Flip all one by one, how many clauses are now satisfied?
- Restart is key with stochastic algorithm

Decidability? Completeness?

- You want a language you want to be able to implement it (decidability)
- DP is sound & complete but takes long to finish, is that better?
- Decidability means that it is complete, guaranteed to find an answer
- Incompleteness can come from data or the fact that we don’t have time to wait
  - It makes sense to have incomplete answers depending on application, in sudoku we will find one because it is easy
- Undecidable does mean that you can still find a good answer

- NP-complete is bad, not so much of a problem when you are on the right of an exponential graph then you will not find a solution but sometimes a linear graph can take longer than exponential (red vs green line)
- Complexity ignores the constants