Logic Exercises

Technical University of Munich
Chair for Logic and Verification

Prof. Tobias Nipkow
Kevin Kappelmann

SS 2021 | Exercise Sheet 2 | 21.04.2021

Exercise 2.1. **[¡Viva La Resolutión!]**

1. We learnt that resolution is a decision procedure for the unsatisfiability problem of CNF formulas. Explain: what does it mean for an algorithm $\mathcal{A} : U \to \{0,1\}$ to be a "decision procedure" for a problem class $\mathcal{P} \subseteq U$?
2. Let S be a set of clauses and C be a clause. Does $S \models C$ imply $S \vdash_{\text{Res}} C$? Proof or counterexample!
3. Can you prove $S \models C$ by resolution?

Solution:

1. A decision procedure must be
   (a) sound: if $\mathcal{A}(p)$ answers 1 then $p \in \mathcal{P}$.
   (b) complete: if $p \in \mathcal{P}$ then $\mathcal{A}(p)$ answers 1.
   (c) terminating: $\mathcal{A}$ terminates on any input.
2. Counterexample: $S \coloneqq \emptyset, C \coloneqq \{\{A, \neg A\}\}$
3. Yes: $S \models C$ iff $S, \neg C \models \bot$ iff $S \cup \{\neg C\}$ is unsatisfiable iff $S, \neg C \vdash_{\text{Res}} \Box$

Exercise 2.2. **[Resolution of Horn-Clauses]**
Can the resolvent of two Horn-clauses be a non-Horn clause?

Solution:
No. Proof: Let C_1, C_2 be two Horn clauses. Both of them have at most one positive literal. Without loss of generality, let A_i be the positive literal occuring in C_1 that we resolve on. Hence, $\neg A_i$ must occur in C_2. The resolvent is $C' = (C_1 \setminus \{A_i\}) \cup (C_2 \setminus \{\neg A_i\})$. We count the positive literals: None in $(C_1 \setminus \{A_i\})$ and at most one in $(C_2 \setminus \{\neg A_i\})$. Hence, there is at most one positive literal in C', i.e. C' is horn.

Exercise 2.3. [The clause is trivial and left as an exercise]
We call a clause C *trivially true* if $A_i \in C$ and $\neg A_i \in C$ for some atom A_i. Show that the resolution algorithm remains complete if it does not consider trivially true clauses for resolution.

Solution:
First we prove a lemma: If S is unsatisfiable and contains a trivially true clause C, then $S' = S \setminus C$ is still unsatisfiable. Proof by contraposition. Assume $S \setminus C$ is satisfiable. Because C is trivially satisfiable, $(S \setminus C) \cup C = S$ is satisfiable.

Assume that S is unsatisfiable. We modify the completeness proof of resolution presented in the lecture. Recall that the proof proceeds by induction on the number of atoms in S. We strengthen the induction by mandating that S contains no trivially true clauses. The base case is trivial. If S is an unsatisfiable set of clauses containing $n+1$ atoms, we first use the previous lemma to remove all trivial clauses from S. Then we construct S_0 and S_1 by setting A_{n+1} to 0 and 1, respectively. Both S_0 and S_1 are unsatisfiable and contain no trivial clauses. By the inductive hypothesis, we obtain resolution proofs such that $S_0 \vdash_{\text{Res'}} \square$ and $S_1 \vdash_{\text{Res'}} \square$, where Res' is our resolution procedure that does not consider trivial clauses. Finally, constructing the resolution proof for S from these proofs (as done in the lecture) introduces no new trivial clauses: in both cases, we either add back A_{n+1} or $\neg A_{n+1}$ but not both.

Exercise 2.4. [Finite Axiomatisation]
Let S_0 and S be sets of formulas. S_0 is called an *axiom schema* for S if for all assignments $\mathcal{A}$, $\mathcal{A} \models S_0$ iff $\mathcal{A} \models S$.

A set S is called *finitely axiomatisable* iff there is a finite axiom schema for S.

1. Are all sets of formulas finitely axiomatisable? Proof or disprove!
2. Let $S = \{F_i \mid i \in \mathbb{N}\}$ be a set of formulas such that for all i, $F_{i+1} \models F_i$ and $F_i \not\models F_{i+1}$. Is S finitely axiomatisable?

Solution:

1. Counterexample: $S \coloneqq \{A_1, A_1 \wedge A_2, A_1 \wedge A_2 \wedge A_3, \ldots\}$. Assume there is a finite axiom schema S_0. S_0 can only contain finitely many atoms. Let $\mathcal{A}$ be an assignment that maps all A_i in S_0 to 1, but all other A_i to 0. Then $\mathcal{A} \models S_0$ but $\mathcal{A} \not\models S$.
2. The same counterexample as above works here.

Exercise 2.5. [What's Semantics Anyway?]
Discuss: Can you think of other ways to give a semantic interpretation of propositional formulas than the one introduced in the lecture? What makes for a good semantic interpretation? What makes for a good model of a set of axioms?

Homework 2.1. [by auto] (+)

Use the resolution procedure to decide if the following formulas are satisfiable. Show your work (by giving the corresponding DAG or linear derivation)!

1. $(A_1 \vee A_2 \vee \neg A_3) \wedge \neg A_1 \wedge (A_1 \vee A_2 \vee A_3) \wedge (A_1 \vee \neg A_2)$
2. $(\neg A_1 \vee A_2) \wedge (\neg A_2 \vee A_3) \wedge (A_1 \vee \neg A_3) \wedge (A_1 \vee A_2 \vee A_3)$

Solution:
Algorithmic

Homework 2.2. [Model Extraction] (+++)

In the lecture, you proved completeness of propositional resolution (if $F \nvdash_{\text{Res}} \square$ then F is satisfiable) in a way that does not directly give raise to a model of F. In practice, however, it is of course very useful to obtain such a model.

On slide 15 of the Resolution lecture slides, the professor gave an algorithm that iteratively adds new clauses to F until no new clause can be added; in other words, it computes the least fixed point of the resolution rule starting on F. We say that the resulting set of this process is *saturated under resolution*.

Give a constructive method that builds a model $\mathcal{M}$ for F from the saturated set of clauses created by the resolution process. Proof the correctness of your construction.

If you need a hint: you can find the construction without a proof here. Only slides 4, 11–14 and 16 are relevant.

Solution:
We use the construction from the hint. We prove its correctness by induction on the number of steps of the algorithm. We denote the nth considered clause by C_n and the maximal literal in a clause C by L_C. Our invariants are $I_{C_n} \models C_i$ and $I_{C_i} \subseteq I_{C_n}$ for all $i \leq n$.

Case 0: if L_{C_0} is negative, we set $I_{C_0} := \emptyset$. If $L_{C_0} = A_i$, we set $I_{C_0} := \{A_i\}$.

Case $n+1$: If $I_{C_n} \models C_{n+1}$, we set $I_{C_{n+1}} := I_{C_n}$ and are done. Assume $I_{C_n} \not\models C_{n+1}$.
If $L_{C_{n+1}} = A_i$, we set $I_{C_{n+1}} := I_{C_n} \cup \{A_i\}$. As A_i is maximal in C_{n+1}, $\neg A_i$ does not occur in C_i for any $i \leq n$. Hence $I_{C_{n+1}} \models C_i$ for all $i \leq n+1$.
Finally assume $L_{C_{n+1}} = \neg A_i$. By assumption, $I_{C_n} \not\models C_{n+1}$. Thus $A_i \in I_{C_n}$. Hence, there is $j \leq n$ such that $L_j = A_i$ and $I_{C_j} = I_{C_{j-1}} \uplus \{A\}$. Let R be the resolvant of C_j and C_{n+1} on A_i. Then R does not contain A_i. Hence, $R \prec C_j \prec C_{n+1}$ and since R is a resolvant, we must have $R = C_k$ for some $k < j$ (remember: our set is saturated under resolution). By the inductive hypothesis, we have $I_{C_{j-1}} \models R$. Thus there is $L \in R$ such that $I_{C_{j-1}} \models L$. As $L \prec A_i = L_{C_j} \preceq \cdots \preceq L_{C_n}$, the assignments $I_{C_{j-1}}, \ldots, I_{C_n}$ agree on L. As $R \subseteq C_j \cup C_{n+1}$, either $L \in C_{n+1}$ or $L \in C_j$.
In the former case, $I_{C_n} \models L$ by agreement of $I_{C_{j-1}}$ and I_{C_n} on L and thus $I_{C_n} \models C_{n+1}$, contradicting assumption $I_{C_n} \not\models C_{n+1}$. In the latter case, $I_{C_{j-1}} \models L$ and thus $I_{C_{j-1}} \models C_j$. Hence, by construction, $I_{C_j} = I_{C_{j-1}}$, contradicting the fact that $I_{C_j} = I_{C_{j-1}} \uplus \{A\}$.
(Note: all in all, we showed that the case $I_{C_n} \not\models C_{n+1}$ and $L_{C_{n+1}} = \neg A_i$ is impossible.)

Homework 2.3. **[by blast]** (+)

Check the following formulas for satisfiability using one of the algorithms seen in the lecture:

1. $(A \vee \neg B \vee \neg D \vee \neg E) \wedge (\neg B \vee C) \wedge B \wedge (\neg C \vee D) \wedge (\neg D \vee E)$
2. $\neg(((A \rightarrow B) \wedge (B \rightarrow A)) \rightarrow (A \leftrightarrow B))$
3. $(A \rightarrow E) \wedge (B \rightarrow \bot) \wedge (C \rightarrow B) \wedge (\top \rightarrow A) \wedge (A \wedge B \rightarrow C) \wedge (C \rightarrow D)$

Show your work! Remember to give a model for satisfiable formulas.

Solution:

Algorithmic by, for example, resolution, truth tables, or equivalences.

Homework 2.4. **[Kőnig's Lemma]** (++)

A finitely branching tree has the following structure:

- There is exactly one root node.
- Every node has a finite number of children.

We assign the root node the *level* 0 and the children of a node at level n the level $n+1$. Let T_n denote the set of all nodes at level n, and T the set of all nodes, i.e. $T = \bigcup_{n \in \mathbb{N}} T_n$. Let P_t for $t \in T$ be the set of parent nodes of a node, i.e. t is a child (or grand-child, ...) of all $t' \in P_t$. A path is a sequence of connected nodes, starting from the root node.

Prove the following lemma using the compactness theorem: Every countably infinite, finitely branching tree has an infinite path.

Hint: Use the following template for the proof.

1. Fix a set of tree nodes T. This set is (countably) infinite. You can assume that the sets T_n and the sets P_t are given.
2. For each node $t \in T$, let A_t be an atom. If an assignment $\mathcal{A}$ makes A_t true, the node t is part of the path.
3. Define a set of propositions S that together guarantee the existence of an infinite path. That set is composed of three subsets:
   (a) For each level $n \in \mathbb{N}$, a node $t \in T_n$ is part of the path.
   (b) If a node t is part of the path, so are all of its parent nodes $t' \in P_t$.
   (c) For each level $n \in \mathbb{N}$, there is at most one node of level n part of the path.
4. Show that any finite subset of $S' \subseteq S$ is satisfiable by constructing an assignment such that $\mathcal{A}_{S'} \models S'$. Consider the largest n for which a proposition from subset (a) is contained in S'.
5. Hence, S is satisfiable. Show that a model $\mathcal{A} \models S$ represents an infinite path in T.

Solution:

See here, Lemma 16.6.

Homework 2.5. **[Negative Resolution]** (++)
We call a clause C *negative* if it only contains negative literals. Show that resolution remains complete if it only resolves two clauses if one of them is negative.

Solution:
Again, we modify the completeness proof of resolution presented in the lecture. The base case is trivial. Assume S is an unsatisfiable set of clauses containing $n+1$ atoms. Then we construct S_0 and S_1 by setting A_{n+1} to 0 and 1, respectively. Both S_0 and S_1 are unsatisfiable. By the inductive hypothesis, we obtain resolution proofs such that $S_0 \vdash_{\text{Res'}} \square$ and $S_1 \vdash_{\text{Res'}} \square$, where Res' is our negative resolution procedure. Now add back $\neg A$ to all clauses resolved in the latter proof. If it is still a refutation, we are done. If we obtain $\{\neg A_{n+1}\}$, we can use it to resolve it against any clause containing A_{n+1} in S. Note that the resulting clauses are those of S_0. Thus, to conclude, we can append the proof of $S_0 \vdash_{\text{Res'}} \square$.

> If you use a trick in logic, whom can you be tricking other than yourself?
> — Ludwig Wittgenstein