Solutins to Exercises for the Course

Logic

Week 7



Exercise 7.1 (First-Order Resolution)

Translate the following formulae into a (skolemized) clausal form. Show that the first two formulae are valid and the third formula is invalid by using the resolution calculus.

a) $\forall x\,(q \to p(x)) \to (q \to \forall y\, p(y))$

b) $\forall y\,(p(y) \wedge q(y)) \to \forall x\,(\forall x\, p(x) \wedge q(x))$

c) $\forall x\, \exists y\, p(x, y) \to \exists v\, \forall u\, p(u, v)$

Solution

Since we are going to show that the formulae are valid (or not) using resolution, we have to negate them before transforming them to clause form.

a)

- Formula: $\forall x\,(q \to p(x)) \to (q \to \forall y\, p(y))$
- Negation: $\neg(\forall x\,(q \to p(x)) \to (q \to \forall y\, p(y)))$
- Equivalent to: $\forall x\,(q \to p(x)) \wedge \neg(q \to \forall y\, p(y))$
- Equivalent to: $\forall x\,(q \to p(x)) \wedge q \wedge \neg\forall y\, p(y)$
- NNF: $\forall x\,(\neg q \vee p(x)) \wedge q \wedge \exists y\, \neg p(y)$
- Prenex form: $\exists y \forall x\,((\neg q \vee p(x)) \wedge q \wedge \neg p(y))$
- Skolemisation: $\forall x\,((\neg q \vee p(x)) \wedge q \wedge \neg p(c))$
- Clause form: $\{\{\neg q, p(x)\},\, \{q\},\, \{\neg p(c)\}\}$

Resolution refutation:

1. $\neg q, p(x)$
2. q
3. $\neg p(c)$
4. $p(x)$ — resolvent of 1 and 2
5. $\square$ — resolvent of 3 and 4 with $\sigma = [x \backslash c]$

b)

- Formula: $\forall y\,(p(y) \land q(y)) \to \forall x\,(\forall x\,p(x) \land q(x))$
- Negation: $\neg(\forall y\,(p(y) \land q(y)) \to \forall x\,(\forall x\,p(x) \land q(x)))$
- NNF: $\forall y\,(p(y) \land q(y)) \land \exists x\,(\exists x\,\neg p(x) \lor \neg q(x))$
- Renaming a bound variable: $\forall y\,(p(y) \land q(y)) \land \exists x\,(\exists z\,\neg p(z) \lor \neg q(x))$
- Prenex form: $\exists x \exists z \forall y\,((p(y) \land q(y)) \land (\neg p(z) \lor \neg q(x)))$
- Skolemisation: $\forall y\,((p(y) \land q(y)) \land (\neg p(d) \lor \neg q(c)))$
- Clause form: $\{\{p(y)\},\ \{q(y)\},\ \{\neg p(d), \neg q(c)\}\}$

Resolution refutation:

1. $p(y)$
2. $q(y)$
3. $\neg p(d), \neg q(c)$
4. $\neg q(c)$ — resolvent of 1 and 3 using $\sigma = [y \backslash d]$
5. $\square$ — resolvent of 2 and 4 using $\sigma = [y \backslash c]$

c)

- Formula: $\forall x\,\exists y\,p(x, y) \to \exists v\,\forall u\,p(u, v)$
- Negation: $\neg(\forall x\,\exists y\,p(x, y) \to \exists v\,\forall u\,p(u, v))$
- NNF: $\forall x\,\exists y\,p(x, y) \land \forall v\,\exists u\,\neg p(u, v))$
- Prenex form: $\forall x\,\exists y\,\forall v\,\exists u\,(p(x, y) \land \neg p(u, v))$
- Skolemisation: $\forall x\,\forall v\,(p(x, f(x)) \land \neg p(g(x, v), v))$
- Clause form (with variable renaming): $\{\{p(x, f(x))\},\ \{\neg p(g(x', v), v)\}\}$

To resolve these twi clauses, we would need to unify

$$p(x, f(x)) \qquad \text{and} \qquad p(g(x', v), v)$$

To unify the first critical pair, we apply $[x \backslash g(x', v)]$ so now we need to unify

$$p(g(x', v), f(g(x', v))) \qquad \text{and} \qquad p(g(x', v), v)$$

But then unification of the second arguments fails due to the occurs check. No resolution step is possible, so the empty clause cannot be derived, and since resolution is complete, we know that the clause set has to be satisfiable.

Exercise 7.2 (Clause Subsumption)

Given two propositional clauses C_1 and C_2, we say that C_1 *subsumes* C_2 if $C_1 \subseteq C_2$.

E.g. $\{p, \neg r\}$ subsumes $\{p, q, \neg r\}$. And the empty clause subsumes every other clause.

The intuition is that if an interpretation satisfies C_1 (i.e. makes one of the literals true) then it certainly also satisfies C_2 which has more literals to choose from. C_1 'says more' than C_2.

Syntactically, if we use the longer clause C_2 in a proof, we would think that it should be possible to use C_1 instead and get a simpler proof... and this is indeed the case!

> Prove that the resolution calculus remains complete if we allow to remove a clause $C_2 \in S$ from S if it is subsumed by a different clause $C_1 \in S$.

Hints:

- *If a node n in a semantic tree falsifies a clause C_2, and $C_1 \subseteq C_2$, can you conclude that n falsifies C_1?*
- *Show that if n is a failure node for a clause set S, then it falsifies some* non-subsumed clause*, i.e. a clause that is not subsumed by any other clause in S.*
- *Conclude that for an unsatisfiable set S not containing the empty clause, there is a resolution step between two* non-subsumed *clauses.*
- *Explain why a clause that is subsumed by another (different) clause in S will continue to be subsumed if we add resolvents to S or remove subsumed clauses.*

Solution

If C_2 is falsified by a node n in a semantic tree, then the path from the root to n contains the negation of every literal in C_2. So if C_1 subsumes C_2, i.e. $C_1 \subseteq C_2$, then n also falsifies C_1. **(*)**

If a node n falsifies some clause $C \in S$, then let C' be smallest (in terms of the number of literals) clause in S that is falsified by n. Now if C' is subsumed by any clause $D \in S$, then due to (*), D is also falsified by n. But since $D \subseteq C'$, D can have as most as many literals as C' and therefore $C' = D$. We conclude that every node that falsifies a clause in S also falsifies a *non-subsumed clause*, i.e. a clause that is not subsumed by any other clause in S. **(**)**

In the completeness proof presented in the lecture, we saw that if the clause set S is unsatisfiable, and does not contain the empty clause, then there are two sibling failure nodes n_1 and n_2, falsifying two clauses C_1 and C_2, which can be resolved.

But our observation **(**)** gives us the stronger statement that n_1 and n_2 falsify two non-subsumed clauses C_1' and C_2'.

We can prove as in the lecture that C_1' and C_2' admit a resolution step that leads to a smaller semantic tree, since they are falsified by sibling failure nodes. This means that any unsatisfiable set of clauses S has a resolution proof where we never need to resolve with a clause that is subsumed by another clause.

Does this mean that we can simply discard subsumed clauses? We have to be careful: maybe a subsumed clause will no longer be subsumed later on? However, we can see that this cannot happen. If a clause $C \in S$ is subsumed by a clause $D \in S$, then adding resolvents of (other) clauses to S is not going to change that fact. Also, removing D because it is subsumed by an even smaller clause E is unproblematic: If E subsumes D ($E \subseteq D$) and D subsumes C ($D \subseteq C$), then E also subsumes C ($E \subseteq C$). Therefore a clause that is subsumed at some point in a resolution derivation will always remain subsumed, and it is therefore safe to remove it from the clause set.