School of Computing and Information Systems
COMP30026 Models of Computation Tutorial Week 6

9–11 September 2020

Plan

There is quite a bit to do this week. Exercise 45 is non-trivial—a good test of your logical fitness. The optional Exercise 50 does not actually have a question; it is an example of using resolution to automate a non-trivial proof of a mathematical theorem (from group theory). The exercise is just to check the resolution proof and make sure you understand the details.

Some reading material on resolution theorem proving is available (see "Readings Online" on the LMS). Note, however, that Dowsing, Rayward-Smith and Walter use a different unification method. In the Week 5 Lecture (Part 1) we intr equations—a view that is arguably both simpler a *et al.*

The exercises

42. Consider these statements:

S_1: "No politician is honest."

S_2: "Some politicians are not honest."

(a) Using the predicate symbols P and H for being a politician and being honest, respectively, express the t $_1$ and F_2.

(b) Is $F_1 \Rightarrow F$

(c) Is $F_1 \Rightarrow F$

(d) Consider these statements:

S_3: "No Australia

S_4: "All honest poli

Using the predicate symbol A for "is Australian", express S_3 and S_4 in clausal form.

(e) Using resolution, show that S_1 is a logical consequence of S_3 and S_4.

(f) Prove or disprove the statement "S_2 is a logical consequence of S_3 and S_4."

43. Consider the following unsatisfiable set of clauses:

$$\{\{P(x)\}, \{\neg P(x), \neg Q(y)\}, \{Q(x), \neg R(y)\}, \{R(x), S(a)\}, \{R(b), \neg S(x)\}\}$$

What is the simplest refutation proof, if "simplest" means "the refutation tree has minimal depth"? What is the simplest refutation proof, if "simplest" means "the refutation tree has fewest nodes"?

44. Consider the following predicates:

- $E(x, y)$, which stands for "x envies y"
- $F(x, y)$, which stands for "x is more fortunate than y"

(a) Using 'a' for Adam, express, in first-order predicate logic, the sentence "Adam envies everyone more fortunate than him."

(b) Using 'e' for Eve, express, in first-order predicate logic, the sentence "Eve is no more fortunate than any who envy her."

(c) Formalise an argument for the conclusion that "Eve is no more fortunate than Adam." That is, express this statement in first-order predicate logic and show that it is a logical consequence of the other two.

45. For this question use the following predicates:

- $G(x)$ for "x is a green dragon"
- $R(x)$ for "x is a red dragon"
- $H(x)$ for "x is a happy dragon"
- $S(x)$ for "x is a dragon capable of spitting fire"
- $P(x, y)$ for "x is a parent of y"
- $C(x, y)$ for "x is a child of y"

(a) Express the following statements as formulas in first-order predicate logic:

i. x is a parent of y if and only if y is a child of x.

ii. A dragon is either green or red; not both.

iii. A dragon is green if and only if at least one of its parents is green.

iv. Green dragons ca

v. A dragon is happy if a

(b) Translate each of the fiv f

(c) Prove, using resolution, that all green dragons are happy.

46. Let p_i be the ith prime number and consider this conjecture: $p_1 p_2 \cdots p_k + 1$ is always prime, that is, when we add 1 to the product of the first k prime numbers, we get a new prime number. This statement is really a universally quantified statement; it says "for all k, 1 + the product of the first k primes is prime."

If the conjecture is wr his, by finding a counter-example. In genera versal" claims, or, equivalently, to prove

(a) Can we program our way out of proving "universal" clai e same ease?

(b) Does the conjecture hold?

A prime pair is a pair $(p, p+2)$ where both p and $p+2$ are prime.

(c) (Optional.) Use Haskell to find the first 50 prime pairs.

While we know that there are infinitely many primes, it is not known whether there are infinitely many prime pairs.

A prime triple is a triple $(p, p+2, p+4)$ with p, $p+2$, and $p+4$ all prime. Here is a Haskell definition of the list of all prime triples, assuming we have already defined `primes`, the list of all primes:

```
primeTriples :: [(Integer,Integer,Integer)]
primeTriples
  = triple primes
    where
      triple (x:y:z:xyzs)
        | x+2 == y && y+2 == z = (x,y,z) : triple (y:z:xyzs)
        | otherwise            =           triple (y:z:xyzs)
```

(d) What happens when you evaluate `primeTriples`?

(e) Prove that there exists one and only one prime triple.

47. (Drill.) Using the unification algorithm, determine whether $Q(f(g(x), y, f(y, z, z)), g(f(a, y, z)))$ and $Q(f(u, g(a), v), u)$ are unifiable. If they are, give a most general unifier. (As usual, we use letters from the end of the alphabet for variables, and letters from the beginning of the alphabet for constants.)

48. (Drill.) Determine whether $P(f(g(x), f(g(x), g(a))), x)$ and $P(f(u, f(v, v)), u)$ are unifiable. If they are, give a most general unifier.

49. (Drill.) Here is an example of a refutation proof where factoring will be needed. Let us try to capture Bertrand Russell's "barber paradox" as a formula in first order predicate logic. Let $B(x)$ mean "x is a barber" and let $S(u, v)$ mean "u shaves v". We want to express that barbers shave people who do not shave themselves, and also, no barber shaves someone who shaves themself. That is:

$$\forall x, y\Big(B(x) \Rightarrow (S(y, y) \oplus S(x, y))\Big) \tag{1}$$

Turn this formula into clausal form. Then use resolution (with factoring) to show that there are no barbers! That is, show that $\neg\exists v\ B(v)$ is a logical consequence of the formula (1).

50. (Optional, a bonus exercise for p owing more substantial resolution exa erstand each step. It is optional, but feel free t MS Discussion Forum if there are steps you don't understand.

Dowsing, Rayward-Smith and Walter (see **Readings Online** on the LMS) give the following example of a non-trivial proof using resolution. It is concerned with group theory. A *group* is a set endowed with a binary operation $\circ$. If we use $P(x, y, z)$ to mean $x \circ y = z$ then we can write the so-called *group axioms* as follows:

$\forall x \forall y \exists z (P(x, y, z))$ (closure)
$\forall x, y, z, u, v, w$)]) associativity)
$\exists x \forall y (P(x,$ (left identity and left inverse)

Notice that the associativity axiom says that if then $x \circ v = u \circ z$. In other words, $x \circ (y \circ z) = (x \circ y) \circ z$. The last axiom s ment x in the set, with the property that $x \circ y = y$ for al *neutral element* for $\circ$). Moreover, for each y there is a z such that $z \circ y$ yields that special element (in other words, each element has a *left inverse*). For an example of a group, consider the set $\mathbb{Z}$ of integers, endowed with the operation $+$.

We can translate the group axioms to clausal form. The first axiom (closure) becomes

$$\{P(x, y, f(x, y))\}$$

The second axiom (associativity) produces two clauses:

$$\{\neg P(x, y, u), \neg P(y, z, v), \neg P(x, v, w), P(u, z, w)\}$$
$$\{\neg P(x, y, u), \neg P(y, z, v), \neg P(u, z, w), P(x, v, w)\}$$

The last axiom (left identity and left inverse) also produces two clauses:

$$\{P(a, y, y)\}$$
$$\{P(g(y), y, a)\}$$

Suppose we want to prove that every element of a group also has a right inverse. That is, we want to prove

$$\exists x \forall y \exists z (P(y, z, x))$$

from the axioms. To do this we first negate our formula, obtaining:

$$\forall x \exists y \forall z (\neg P(y, z, x))$$

In clausal form this becomes $\{\neg P(h(x), z, x)\}$.

Below is the proof by resolution. It is a mechanical proof of a non-trivial theorem. When there is ambiguity, I have used underlining to show which atom takes part in the resolution step.

Make sure you understand each resolution step. Did the refutation make use of all the axioms? If you try to do the proof on your own without looking at the proof above, you will find that there are many blind alleys (most of them will end in failure due to the occur check). So you will most likely take a long time, and do lots of back-tracking. With a computer of course we find the refutation in a flash.

Notice how clauses have had their variables renamed to avoid name clashes. Try to track how the variables x' and z' from the original query get bound during this proof.