1

Quiz

1. In ordered resolution, we require that $L \not\succ P$. Why do we not just require $P \succeq L$?

   Answer: For terms that contain variables, it is not possible to find an ordering that is both stable, well-founded, and also total. That is, there will always be terms s, t such that we neither have $s \succ t$ nor $s = t$ nor $t \succ s$. For example, $\mathtt{P}(x)$ and $\mathtt{P}(y)$ are incomparable in any stable well-founded order.

   Hence, if we limited resolution as described, we would for instance be unable to do *any* resolution step in the CNF with two clauses: $\mathtt{P}(x) \vee \mathtt{P}(y)$ and $\neg\mathtt{P}(a)$. Since this CNF is unsatisfiable, we see that it would clearly break refutation-completeness.

2. What is a Horn clause

   Answer: a clause with *at most* one positive literal.

3. Why is it more efficient to use resolution with Horn clauses than in general?

   Answer: We can make sure to present Horn clauses with the positive literal (if any) at the head. Then, to determine if a resolution step is possible between for instance $A \vee \neg B_1 \vee \cdots \vee \neg B_n$ and $C \vee \neg D_1 \vee \cdots \vee \neg D_m$, we only need to check if A unifies with some D_i, or C unifies with some B_i; that is, a linear number $(n + m)$ of unifications. This is much more efficient than general resolution, where the number of unifications would be quadratic (there would be $(n + 1) * (m + 1)$ potential predicate combinations to do the comparison on). (Note: if one of the Horn clauses has no positive literal, the number of checks is even smaller.)

4. Why do we need equational logic when we already have predicate logic?

   Answer: while we *could* define equality as a predicate, and describe properties like reflexivity, symmetry and transitivity using predicate formulas, the difficulty is that equality should also be *monotonic*: for example, if $x = y$ and $\mathtt{P}(x)$ then also $\mathtt{P}(y)$; and if $x = y$ then $f(x) = f(y)$.

   Technically we *could* encode this using predicate logic, by adding monotonicity requirements for every function symbol and every predicate symbol in the system. However, this causes a lot of overhead in proofs. It is far more efficient to avoid the encoding step and build equality into the system directly.

5. Give the positive superposition and negative superposition rule, and an example of how they might be applied (just one step; no need to give a full superposition proof).

   Answer:

   Positive superposition:

   $$\frac{s = t \vee V \qquad C[u] = q \vee W}{(C[t] = q \vee V \vee W)\sigma} \quad \begin{array}{l}\sigma = mgu(s, u) \text{ and} \\ u \text{ is not a variable}\end{array}$$

   Negative superposition:

$$\frac{s = t \vee V \qquad C[u] \neq q \vee W}{(C[t] \neq q \vee V \vee W)\sigma} \quad \begin{array}{l}\sigma = mgu(s,u) \text{ and} \\ u \text{ is not a variable}\end{array}$$

Example:

1 $FavDrink(Teacher(x)) = Tea \vee Name(x) \neq AR \vee Quarter(x) \neq Q2$
2 $Teacher(Course(Wed, 1030, y)) = Sebastian \vee y \neq Q1$
3 $Quarter(Course(x, y, z)) = z$

Positive superposition between 1 and 2 with substitution $[x := Course(Wed, 1030, y)]$ gives 4:

$$\begin{aligned} FavDrink(Sebastian) = Tea \quad &\vee \\ y \neq Q1 \quad &\vee \\ Name(Course(Wed, 1030, y)) \neq AR \quad &\vee \\ Quarter(Course(Wed, 1030, y)) \neq Q2 \end{aligned}$$

Let's rename 3 to avoid duplicate variables: $Quarter(Course(x', y', z')) = z'$.

Then negative superposition between 3 and 4 with substitution $[x' := Wed, y' := 1030, z' := y]$ gives:

$$\begin{aligned} FavDrink(Sebastian) = Tea \quad &\vee \\ y \neq Q1 \quad &\vee \\ Name(Course(Wed, 1030, y)) \neq AR \quad &\vee \\ y \neq Q2 \end{aligned}$$