Isabelle/HOL Exercises: A Bad Axiomatization

This exercise shows an example of how a bad axiomatization can destroy Isabelle's guarantee of soundness. We will add to Isabelle's existing theory the axioms of Naive Set Theory presented in Lecture 6.

First, we declare the existence of a new type `SET`, which represents the sets of our Naive Set Theory. We do this in Isabelle with the following declaration.

```
typedecl SET
```

Now we declare the membership predicate (written $\in$ in the lecture, but this clashes with Isabelle's existing set membership predicate), and the axiom. The `axiomatization` declaration first lists the new function symbols we are adding, and then lists the axioms.

Here, we state `comprehension` as a single axioms, but really it is an axiom schema, representing infinitely many axioms, one for every P.

```
axiomatization
  Member :: "SET ⇒ SET ⇒ bool"
where
  comprehension : "∃y. ∀x. Member x y ⟷ P x"
```

You can use the axiom comprehension in a proof by using the command:

```
apply (cut_tac P="λx. XXXX" in comprehension)
```

where `XXXX` is the predicate (with free variable x) that you want to instantiate the axiom with.

You can now prove the paradoxical statement shown in the lecture.

```
lemma member_iff_not_member :
    "∃y. Member y y ⟷ ¬ Member y y"
oops
```

Using the lemma, it is now possible to prove `False`, showing that we have definitely introduced an inconsistency. You will have to use the rule `excluded_middle` or one of the other axioms in order to complete the proof. It may be easier to try to work out how to prove this theorem on paper first, before attempting it on the computer.

(Note: Isabelle's built-in QuickCheck will attempt to show you a counterexample to this theorem, because it 'knows' that False is not provable. The QuickCheck facility doesn't take into account any axioms that have been added.)

```
theorem inconsistency : False
oops
```

After we have proved False, we can prove all sorts of nonsense:

```
theorem "1 = 0"
apply (rule FalseE)
apply (rule inconsistency)
done
```