FOL with Equality: Equational Reasoning

In this exercise, we will study elementary equational reasoning for groups and orders, and learn how to combine this with reasoning via case distinction. The technical level is deliberately rather low since elementary fall-back techniques are necessary if more automated tactics fail.

1 More on Isabelle

1.1 Backward Proof Control Structures

Revising our first proof scripts, it becomes clear that proof-scripts contain considerable repetition. Thus, more automation can be achieved by introducing control structures in the ISAR-language. These are:

1. **M,M' sequential composition**: try tactic M; if it succeeds try tactic M'.
2. **M|M alternative**: try tactic M; if it fails try tactic M'.
3. **M? option**: try tactic M; if it fails report success.
4. **M+ repetition**: try tactic M and repeat as long as no failure occurs.
For example, instead of:

```
apply(rule X)
apply(erule Y)
```

we may write:

```
apply(rule X, rule Y)
```

Further, instead of:

```
apply(drule mp)
apply(assumption)
apply(assumption)
apply(erule disjE)
apply(drule mp)
apply(erule disjE)
```

we may write:

```
apply(drule mp,(assumption|erule disjE)+)+
```

### 1.2 FOL with Equality

In lecture, *first-order logic with equality* has been introduced as a logical system where the equality \( x = y \) has been defined as a predicate on terms which represents a congruence relation. This is covered in Isabelle/FOF by the following rules:

- **refl**: \( a=a \)
- **trans**: \[ x=z; x=y \implies y=z \]
- **sym**: \( y=x \implies x=y \)
- **subst**: \[ a=b; P(a) \implies P(b) \]

Note that the substitutivity rule in Isabelle does not distinguish between “formulas” and “terms” as described in the lecture.

### 1.3 New Tactics

We introduce two new tactical commands for case splitting reasoning and performing one rewrite step. Both can be understood as abbreviation of previously introduced commands and/or rules. These are:

1. **case_tac** "<form>", where `<form>` is a splitting formula. It is equivalent to: `insert excluded_middle[of "<form>"], erule disjE`
2. subst rule, where rule is a (conditional) equation performed left-to-right.

It is equivalent to: rule subst[OF sym[OF rule]]

Note that the subst chooses an arbitrary “position” where to perform a rewrite step; this lack of control may be sometimes undesirable. In such cases there may be no alternative to providing a more concrete substitution for meta variables, for example like rule_tac P = "λz. ?X * z = e" in subst[OF rule]. Here, the λ-expression denotes a function that generates a term (with a “hole” ?X). In general, giving too special substitutions is tedious and makes proof-scripts less robust; giving too general substitutions may result in a dead end of a proof.

By the way, sym[OF rule] is also equivalent to rule[symmetric].

1.4 New Declaration Elements

In an isar theory file, proofs can be mixed with other syntactic elements such as type declarations, constant declarations, definitions and axioms (here only used as exercise!). Consider:

```isar
typedec <T>
arity  <T> :: "term"
```

Here, the type <T> is declared; since Isabelle has a two-staged type system with “types of types” called type classes, the new type is declared to the class term introduced in the ifol theory.

A standard constant declaration is given by an example:

```isar
consts
  If :: "[o, i, i] ⇒ i"  ("( if ( _)/ then ( _)/ else ( _))"
axioms
  if_P : "P ⇒ ( if P then y else z) = y"
  if_notP : "¬P ⇒ (if P then y else z) = z"
```

The other possibility is by a special type of axioms, called definitions:

```isar
defs
  if_def : "( if P then y else z) = <E>"
```
where \(<E>\) is a closed expression not containing the constant If (we do not have the semantic means to give a useful definition for If at the moment).

Use analogies to declarations in the ifOL and FOL theories of the Isabelle distribution. You can find these theories nicely formatted on the Isabelle website: [http://isabelle.in.tum.de/library/FOL/index.html](http://isabelle.in.tum.de/library/FOL/index.html)

### 1.5 Proof-State Massage

The standard apply-command usually effects only the first subgoal. Thus, it may be desirable to rotate the list of subgoals in a proof state. The defer \(n\) or prefer \(n\) commands move a subgoal to the last or the first position.

For the choice of unifiers, the order of assumptions in a subgoal may be relevant. rotate_tac \(n\) rotates the assumptions of the first subgoal by \(n\) positions: from right to left if \(n\) is positive, and from left to right otherwise. The default value is one.

### 2 Exercises

#### 2.1 Exercise 15

Derive the symmetry and transitivity rules for \(=\)

\[
\begin{align*}
    & x = y \quad \text{sym} \quad y = x \\
    & x = y, \quad y = z \quad \text{trans} \quad x = z
\end{align*}
\]

using only applications of refl and subst.

#### 2.2 Exercise 16

Prove the following group properties from the lecture without using the tactic command subst.

\[
x^{-1} \ast x = e \quad \text{and} \quad x \ast e = x
\]

**Hint:** Declare a type \(i\) of sort term in Isabelle/FOL and the constants \(_{-1}, \_ \ast \_, \_\) and \(e\) over \(i\) in your theory! (use analogies to declarations in the theories FOL and IFOL).

**Hint:** Take the “axioms” of group theory, namely associativity, right identity and right inverse as named assumptions in a backward proof.
2.3 Exercise 17

Declare a predicate \( \_ \leq \_ \) of type \( i \Rightarrow i \Rightarrow o \) (similar to equality). Formalize that \( \_ \leq \_ \) is total or antisymmetric and use this as assumption at need in the proofs.

Prove that:

1. \( \neg x \leq y \Rightarrow y \leq x \)
2. \( \neg y \leq x \Rightarrow x \leq y \)
3. \( \neg y = x \Rightarrow (x \leq y) \vee (y \leq x) \)
4. \( y \leq x \Rightarrow x = y \lor (\neg x \leq y) \)

Hint: Use \( subst \) and \( case\_tac \) whenever possible.

Hint: Consider derived rules of classical logic like \( swap \), \( contrapos \) and \( contrapos2 \).

2.4 Exercise 18

Declare the constant \( if \) (presented syntactically in mix-fix notation) and define it via the axioms:

\[
\begin{align*}
if_\_P : & ~ " P \Rightarrow (if~P~then~y~else~z) = y" \\
if_notP : & ~ " \neg P \Rightarrow (if~P~then~y~else~z) = z"
\end{align*}
\]

Assume in the sequel that \( \_ \leq \_ \) is a partial order (i.e. reflexive, transitive, antisymmetric).

Declare and define the operation \( max \) based on \( \_ \leq \_ \) and \( if \).

Prove that \( max \) is

1. idempotent,
2. commutative
3. and left-idempotent (i.e. \( max(x,max(x,y)) = max(x,y) \))

Hint: Use \( subst \) and \( case\_tac \) whenever possible.