Master FIIL : Preuve interactives et Applications

Année 2018-2019

Prof. Burkhart Wolff
wolff@lri.fr

TP 2 - Datatypes and Induction in Isabelle/HOL

Semaine du 25 septembre 2018

Exercice 1 (Simple Logical Backward - Proofs)
State the following properties as `lemma` and prove them :

1. $A \wedge B \wedge C \rightarrow B \wedge A$
2. $((A \rightarrow B) \rightarrow A) \rightarrow A$ (Pierce Law)
3. $(\forall x. A \rightarrow B(x)) = (A \rightarrow (\forall x. B(x)))$

Objective : try to solve these proofs with elementary Isabelle proof methods, i.e. `rule`, `rule_tac`, `erule`, `erule_tac` before applying more advanced automated procedures like `simp` and `auto`. Hint : search for basic logical rules from the `HOL` theory involving the logical connectives/quantifiers.

Exercice 2 (Simple Induction Proofs)

1. Define an own inductive datatype for lists (called `'a seq`) with the variants `Empty` and `Seq`
2. Define a function `revert` and prove the property : `revert(revert s) = s` by induction.
3. Define a function `conc` (concatenate) and prove the properties : `conc xs Empty = xs` and `conc (conc xs ys) zs = conc xs (conc ys zs)` (associativity) by induction.
4. Define an own inductive datatype for trees `'a tree` (with labelled Nodes and anonymous Leaf's).
5. Define an own inductive datatype for trees `'a tree` (with labelled Nodes and anonymous Leaf's).
6. Define `reflect` $'a\ tree \Rightarrow' a\ tree$, `replace` $nat\ list \Rightarrow' a\ tree \Rightarrow' a\ tree \Rightarrow' a\ tree$ suitably (the index list should be a Dewey-position in the term to be replaced) and prove the lemmas : `reflect(reflect t) = t` and
   `replace s t (replace s t t') = (replace s t t')`.
7. Define an abstract syntax tree for the IMP language involving `SKIP`, the assignment, the IF-THEN-ELSE and the WHILE loop.

Hints :

1. use the specification constructs `datatype` (for inductive datatype definitions) and `fun` for recursive function definitions.
2. apply it as suitably instantiated (substitution!) rule via the variant `rule_tac`
3. apply variant with the proof method : `induct`.

Exercice 3 (Modeling Exercise)

1. Define an abstract syntax tree for the IMP language involving `SKIP`, the assignment, the IF-THEN-ELSE and the WHILE loop.

Hints :

1. you may use the following type synonyms for the task :

```
type_synonym vname  = string
type_synonym val = int
type_synonym state  = "vname => val"
type_synonym aexp = "state => val"
type_synonym bexp = "state => bool"
```

2. You may define a concrete syntax via syntax-paraphrasings like
   `("IF _ THEN _ ELSE _"  [65, 60, 61] 60)`
   (compare with the `‘a list` definition from the HOL library.