Homework2

By Giuliano Losa

March 5, 2011

Contents

1 Agatha was murdered in the Dreadbury Mansion. Whodunit?

theory *Agatha*
imports *Main*
begin

typedecl *Person*

consts
  Agatha :: *Person*
  Charles :: *Person*
  butler :: *Person*
  livesInMansion :: *Person* $\Rightarrow$ *bool*
  killed :: *Person* $\Rightarrow$ *Person* $\Rightarrow$ *bool*
  hates :: *Person* $\Rightarrow$ *Person* $\Rightarrow$ *bool*
  richer :: *Person* $\Rightarrow$ *Person* $\Rightarrow$ *bool*

definition $f1$ :: *bool*
  where $f1 \equiv \exists\ x\ .\ livesInMansion\ x \wedge killed\ x\ Agatha$
definition $f2$ :: *bool*
  where $f2 \equiv livesInMansion\ Agatha \wedge livesInMansion\ butler \wedge livesInMansion\ Charles$
    $\wedge\ (\forall\ x\ .\ x \neq Agatha \wedge x \neq butler \wedge x \neq Charles \longrightarrow \neg\ livesInMansion\ x)$
definition $f3$ :: *bool*
  where $f3 \equiv \forall\ x\ y\ .\ killed\ x\ y \longrightarrow (hates\ x\ y \wedge \neg\ richer\ x\ y)$
definition $f4$ :: *bool*
  where $f4 \equiv \forall\ x\ .\ hates\ Agatha\ x \longrightarrow \neg\ hates\ Charles\ x$
definition $f5$ :: *bool*
  where $f5 \equiv \forall\ x\ .\ x \neq butler \longrightarrow hates\ Agatha\ x$
definition $f6$ :: *bool*
  where $f6 \equiv \forall\ x\ .\ (\neg\ richer\ x\ Agatha) \longrightarrow hates\ butler\ x$
definition $f7$ :: *bool*
  where $f7 \equiv \forall\ x\ .\ hates\ Agatha\ x \longrightarrow hates\ butler\ x$
definition $f8$ :: *bool*

where *f8* ≡ ∀ *x* . ¬ (∀ *y* . *hates x y*)
definition *f9* :: *bool*
where *f9* ≡ *Agatha* ≠ *butler*

declare
f1-def [*simp*] *f2-def* [*simp*] *f3-def* [*simp*] *f4-def* [*simp*] *f5-def* [*simp*] *f6-def* [*simp*] *f7-def* [*simp*] *f8-def* [*simp*] *f9-def* [*simp*]
— here we declare the definition of the facts to be simplification rules. This will cause the simplifier to rewrite the facts (for example f1) with their definition.

lemma *Agatha-commited-suicide*:
assumes *f1* **and** *f2* **and** *f3* **and** *f4* **and** *f5* **and** *f6* **and** *f7* **and** *f8* **and** *f9*
shows *killed Agatha Agatha*
using ‹*f1*› ‹*f2*› ‹*f3*› ‹*f4*› ‹*f5*› ‹*f6*› ‹*f7*› ‹*f8*› ‹*f9*› **by** (*simp*, *metis*)
— simp rewrites the the facts according to their definition, then metis proves the goal

lemma *Agatha-commited-suicide2*:
— a more detailed proof
assumes *f1* **and** *f2* **and** *f3* **and** *f4* **and** *f5* **and** *f6* **and** *f7* **and** *f8* **and** *f9*
shows *killed Agatha Agatha*
proof −
have *not-charles*:¬ *killed Charles Agatha*
proof −
have ¬ *hates Charles Agatha*
proof −
from ‹*f5*› **and** ‹*f9*› **have** *hates Agatha Agatha* **by** *auto*
with ‹*f4*› **show** *?thesis* **by** *auto*
— "with" stands for "from '*hates Agatha Agatha*'"
qed
with ‹*f3*› **show** ¬ *killed Charles Agatha* **by** *auto*
— "with" stands for "from '¬ *hates Charles Agatha*'"
qed
have *not-butler*:¬ *killed butler Agatha* **using** ‹*f3*› ‹*f5*› ‹*f6*› ‹*f7*› ‹*f8*› **by** *force*
— The facts used were suggested by sledgehammer.
from *not-charles* **and** *not-butler* **and** ‹*f1*› **and** ‹*f2*› **show** *killed Agatha Agatha* **by** *force*
qed

end

2 Formalization and solution of problem 5 from exercises 2

theory *Exercises2Pb5*
imports *Main*
begin

In this example we make use of proof contexts. A proof contexts is a part of a proof delimited by curly brackets.

Inside a proof context on may fix some variables with the command "fix" (like in "fix x y"), one may assume some arbitrary facts with the "assume" command (like in "assume ¬ *killed Charles Agatha*". We may then prove several facts with the "have" command.

Upon closing the block, we will have proved that taking the fixed variables to be arbitrary and assuming what was assumed in the block, we can conclude that the fact proved in the last line of the block holds. See the examples in the file!.

Blocks are usefull to avoid considering what is the exact goal that isabelle wants you to prove. When starting a proof (with the "proof -" command), you may ignore what isabelle displays in the goals window and instead open a new proof block. In this new block you may prove what you think is to be proved. After closing the proof block, you may reconcile what you proved with what isabelle wanted you to prove by using the "auto" proof method. Again, see the examples in the file!

typedecl *Variable*
type-synonym *'a relation* = (*'a* × *'a*) *set*

datatype *relationalExpr* =
  Relation Variable
  |*Union relationalExpr relationalExpr*
  |*Comp relationalExpr relationalExpr*

primrec *semantics* :: *relationalExpr* ⇒ (*Variable* ⇒ *'a relation*) ⇒ *'a relation*
  — The semantics of a relational expression under an interpretation of its variables
  where
  semantics (*Relation rv*) *f* = *f rv*
  |*semantics* (*Union e1 e2*) *f* = (*semantics e1 f*) ∪ (*semantics e2 f*)
  |*semantics* (*Comp e1 e2*) *f* = (*semantics e1 f*) *O* (*semantics e2 f*)

theorem *monotonic*:
  — $f' = f(rv := r')$ says that *if* $x \neq rv$ *then* $f'\ x = f\ x$ *else* $f'\ x = r'$
  assumes *a1*:$f' = f(rv := r')$ **and** *a2*:$f\ rv \subseteq r'$
  shows *semantics E f* ⊆ *semantics E f'*
proof (*induct E*)
  — Each case of the induction is separated by the *next* keyword
  — There are three cases: one for *Relation rv*, one for *Union e1 e2*, and one for *Comp e1 e2*
  — We start with the based case
  fix *r*
  show *semantics* (*Relation r*) *f* ⊆ *semantics* (*Relation r*) *f'*
  proof (*cases rv* = *r*)
    assume *f1*:*rv* = *r*
    from *f1* **have** *f2*:*f r* ⊆ *r'* **using** *a2* **by** *auto*
    from *f2* **have** *f3*:*semantics* (*Relation r*) *f* ⊆ *r'* **by** *auto*
    from *a1* **and** *f1* **have** *f4*:*semantics* (*Relation r*) *f'* = *r'* **by** *auto*
    from *f3* **and** *f4* **show** *?thesis* **by** *auto*
  next
    assume *f1*:*rv* ≠ *r*
    have *f2*:*semantics* (*Relation r*) *f* = *f r* **by** *auto*
    from *f1* **and** *a1* **have** *f3*:*semantics* (*Relation r*) *f'* = *f r* **by** *auto*
    from *f2* **and** *f3* **show** *?thesis* **by** *auto*
  qed
next
    — Inductive step, case *Union e1 e2*
  fix *e1 e2*
  assume *ih1*:*semantics e1 f* ⊆ *semantics e1 f'*
    and *ih2*:*semantics e2 f* ⊆ *semantics e2 f'*
  have *f1*:*semantics* (*Union e1 e2*) *f* = *semantics e1 f* ∪ *semantics e2 f* **by** *auto*
  have *f2*:*semantics* (*Union e1 e2*) *f'* = *semantics e1 f'* ∪ *semantics e2 f'* **by** *auto*
  show *semantics* (*Union e1 e2*) *f* ⊆ *semantics* (*Union e1 e2*) *f'*
  proof −

— The opening curly bracket on the next line opens a new context where we may assume and prove whatever we like. When closing the context (with a matching curly bracket), we may use what we proved inside the context as an assumption.

{ fix *x y*
assume *f3*:$(x,\ y) \in$ *semantics* (*Union e1 e2*) *f*
from *f1* **and** *f3* **have** *f4*:$(x,\ y) \in$ *semantics e1 f* $\vee$ $(x,\ y) \in$ *semantics e2 f* **by** *auto*
from *f4* **and** *ih1* **and** *ih2* **have** *f5*:$(x,\ y) \in$ *semantics e1 f'* $\vee$ $(x,\ y) \in$ *semantics e2 f'* **by** *auto*
from *f2* **and** *f5* **have** $(x,\ y) \in$ *semantics* (*Union e1 e2*) *f'* **by** *auto*
}

On the line above we closed the context. We proved the statement $\bigwedge$ *x y* . $(x,\ y) \in$ *semantics* (*Union e1 e2*) *f* $\Longrightarrow$ $(x,\ y) \in$ *semantics* (*Union e1 e2*) *f'*

thus *?thesis* **by** *auto*
— "thus" stands for from $\bigwedge$ *x y* . $(x,\ y) \in$ *semantics* (*Union e1 e2*) *f* $\Longrightarrow$ $(x,\ y) \in$ *semantics* (*Union e1 e2*) *f'*. "?thesis" represents our goal
qed
next
— Inductive step, *Comp e1 e2*
fix *e1 e2*
assume *ih1*:*semantics e1 f* $\subseteq$ *semantics e1 f'* **and** *ih2*:*semantics e2 f* $\subseteq$ *semantics e2 f'*
have *f1*:*semantics* (*Comp e1 e2*) *f* = *semantics e1 f O semantics e2 f* **by** *auto*
have *f2*:*semantics* (*Comp e1 e2*) *f'* = *semantics e1 f' O semantics e2 f'* **by** *auto*
show *semantics* (*Comp e1 e2*) *f* $\subseteq$ *semantics* (*Comp e1 e2*) *f'*
proof −
— As above, we open a new context
{ fix *x y*
assume *f3*:$(x,\ y) \in$ *semantics* (*Comp e1 e2*) *f*
from *f1* **and** *f3* **obtain** *z* **where** *f4*:$(x,\ z) \in$ *semantics e1 f* $\wedge$ $(z,\ y) \in$ *semantics e2 f* **by** *auto*
from *f4* **and** *ih1* **and** *ih2* **have** *f5*:$(x,\ z) \in$ *semantics e1 f'* $\wedge$ $(z,\ y) \in$ *semantics e2 f'* **by** *auto*
from *f5* **have** $(x,\ y) \in$ *semantics* (*Comp e1 e2*) *f'* **by** *auto*
}

On the line above we close the context. We proved the statment $\bigwedge$ *x y* . $(x,\ y) \in$ *semantics* (*Comp e1 e2*) *f* $\Longrightarrow$ $(x,\ y) \in$ *semantics* (*Comp e1 e2*) *f'*

thus *?thesis* **by** *auto*
— As before, thus represents the fact we proved in the immediately preceding proof context
qed
qed

end

3 Homework 2, Problem 4. Formalization of Problem 3 from Homework 3.

theory *Homework2Pb3Sorry*
imports *Main*
begin

typedecl *Stmt*
type-synonym *'a relation* = (*'a* × *'a*) *set*

datatype *'a GuardedCmdExpr* =

Statement Stmt
|*SeqComp 'a GuardedCmdExpr 'a GuardedCmdExpr*
|*IfThenElse 'a set 'a GuardedCmdExpr 'a GuardedCmdExpr*

primrec *semantics* :: *'a GuardedCmdExpr* ⇒ (*Stmt* ⇒ *'a relation*) ⇒ *'a relation*
— *O* is relation composition
— *Id-on P* is the diagonal relation on set P. Its definition can be found by searching for *Id-on-def*.
— (− *P*) is the complement of P
where
semantics (*Statement s*) *f* = *f s*
|*semantics* (*SeqComp s1 s2*) *f* = *semantics s1 f O semantics s2 f*
|*semantics* (*IfThenElse P s1 s2*) *f* = (*Id-on P O semantics s1 f*) ∪ (*Id-on* (− *P*) *O semantics s2 f*)

This file is continued is the provided skeleton. Your task is to complete the skeleton. You are not allowed to use the "sorry" or "axioms" command. Your proof should look like the ones in this file. You may also use sledgehammer, using the command "sledgehammer [provers="e spass"]"

end