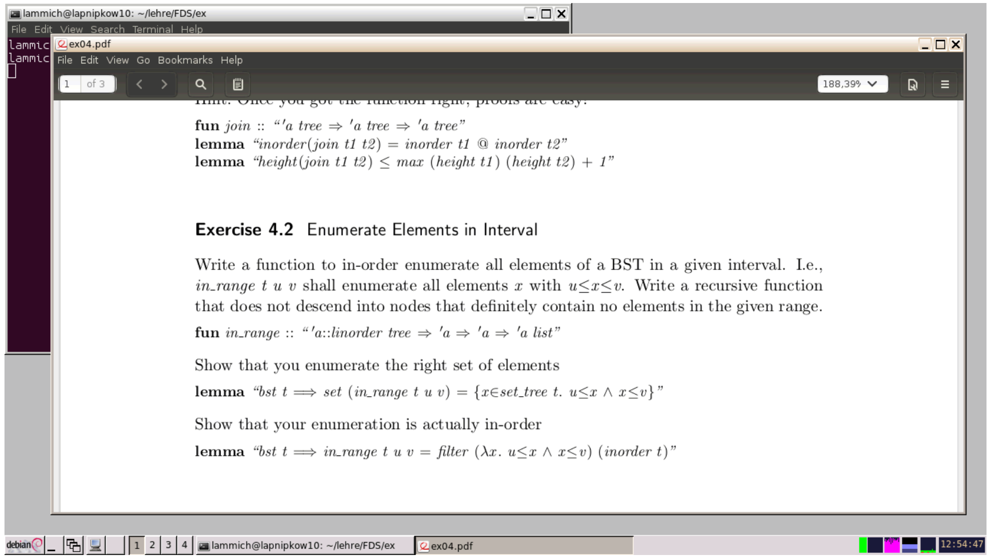
Script **generated by TTT**

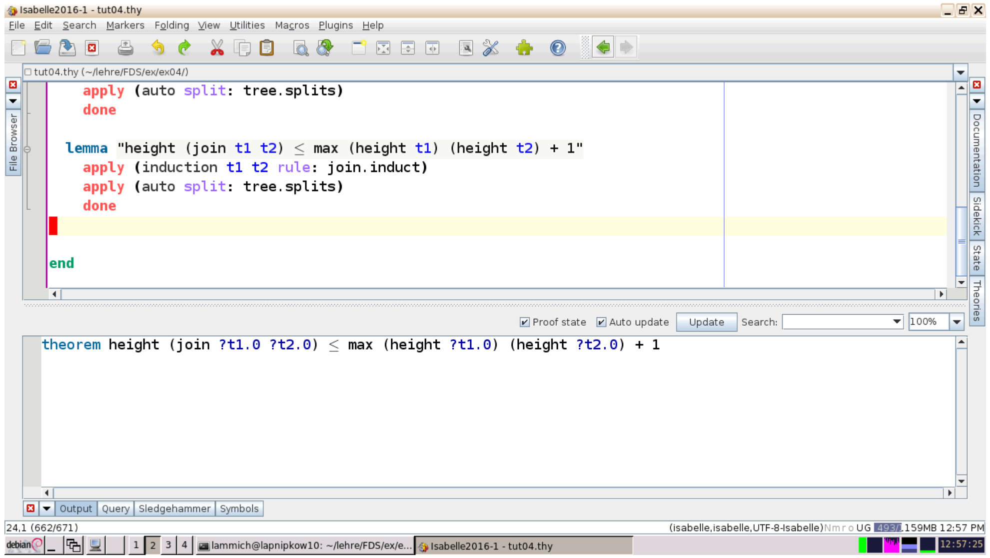
Title: Lammich: FDS Tutorial (19.05.2017)

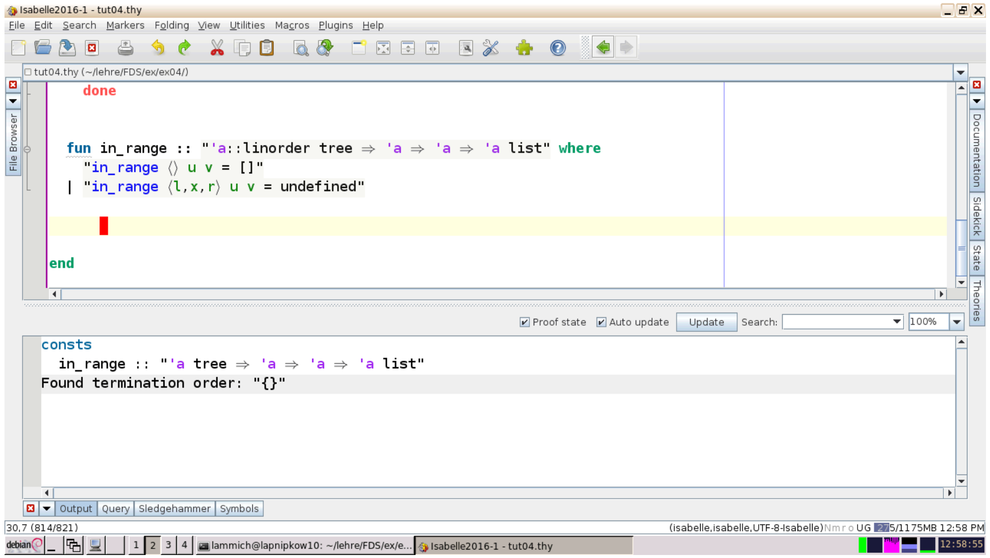
Date: Fri May 19 12:54:47 CEST 2017

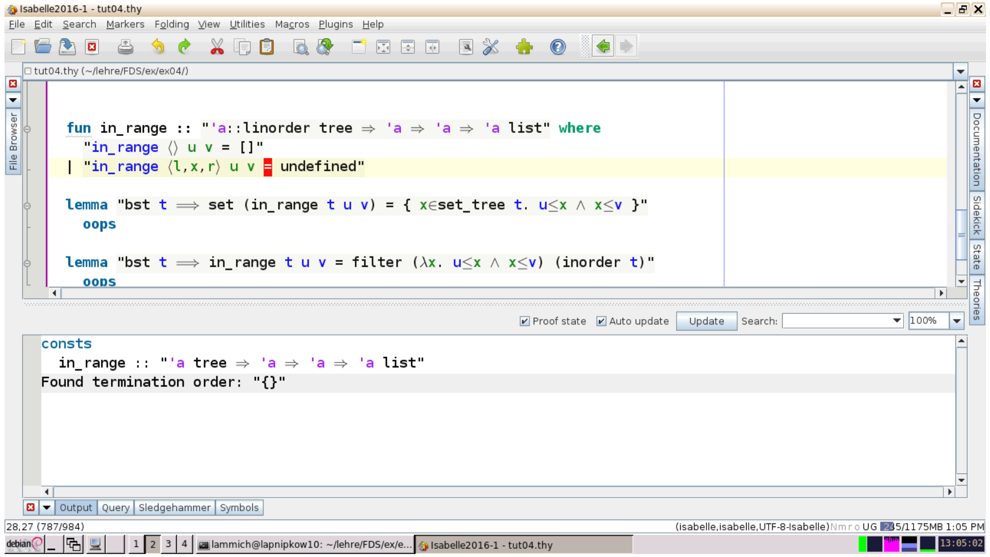
Duration: 63:50 min

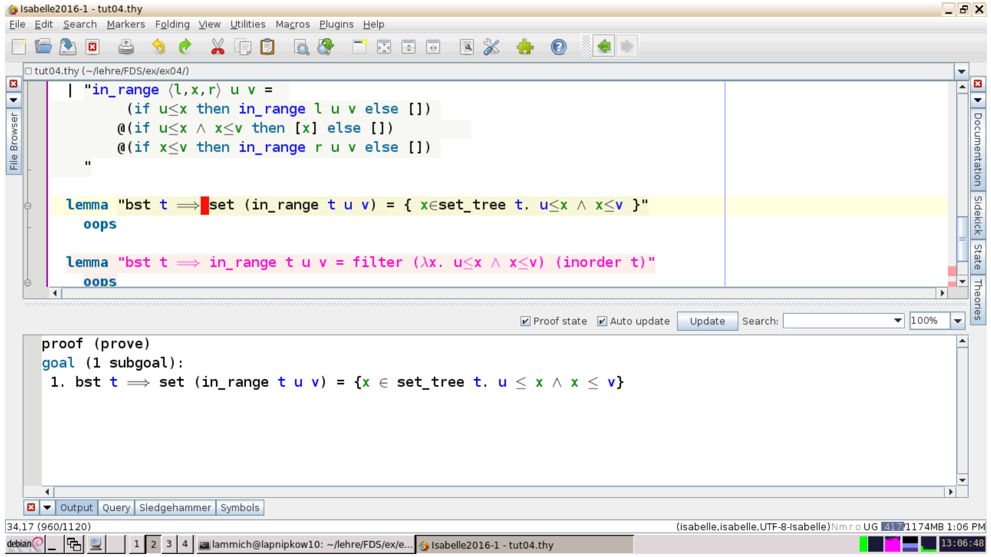
Pages: 55

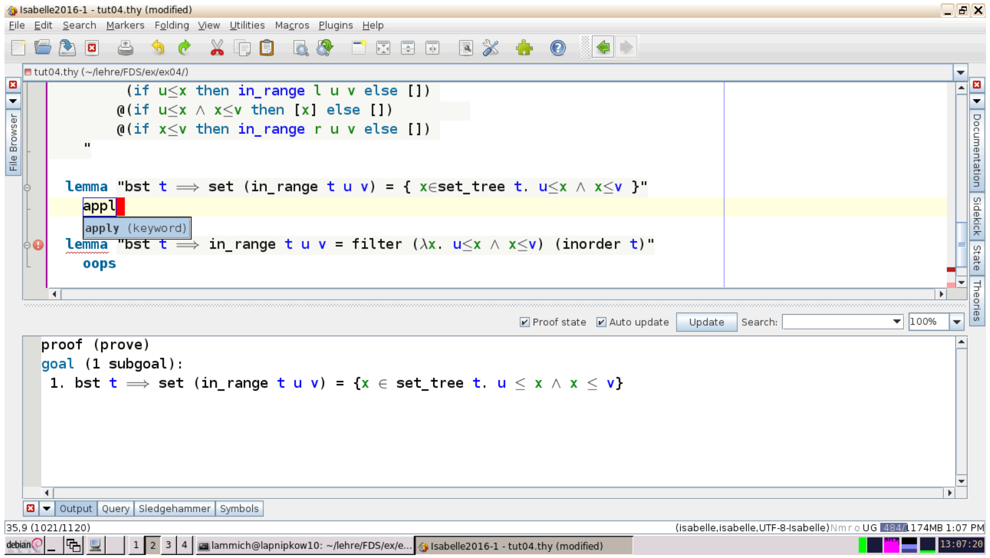


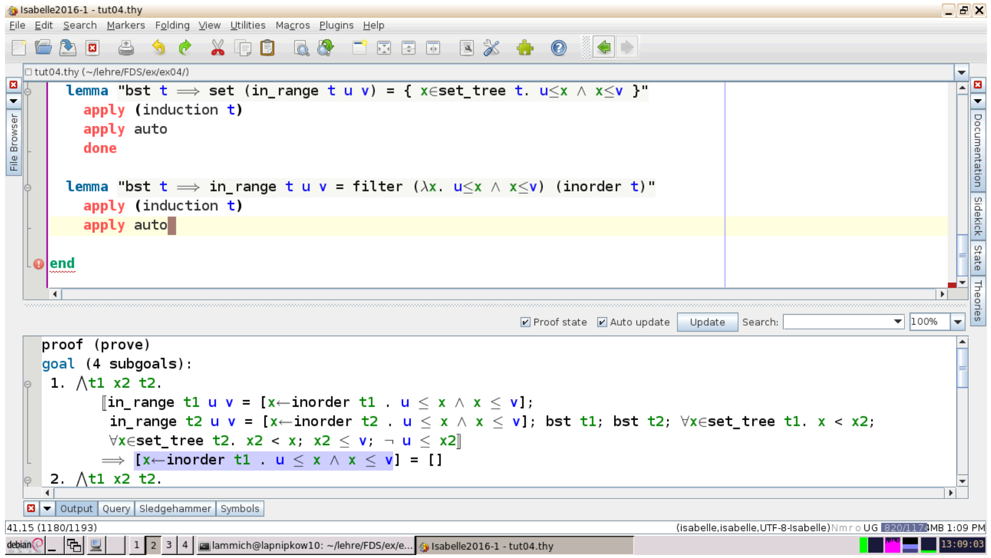


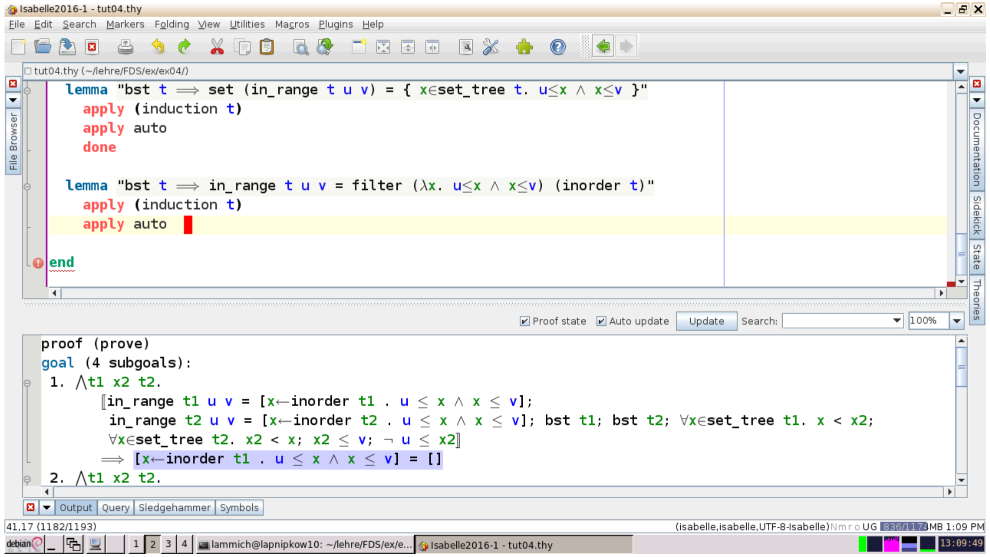


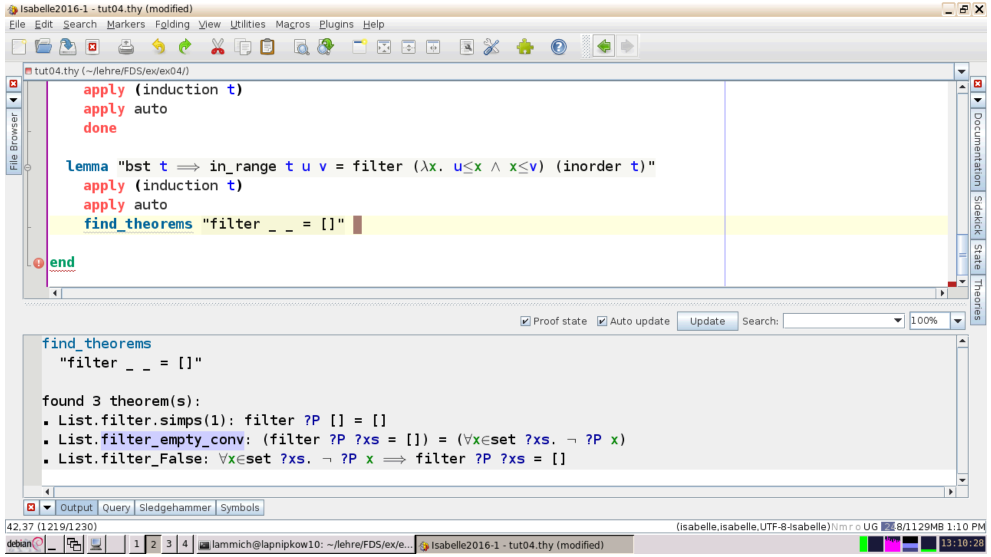

Isabelle2016-1 - tut04.thy
fun in_range :: "'a::linorder tree ⇒ 'a ⇒ 'a ⇒ 'a list" where
"in_range ⟨⟩ u v = []"
| "in_range ⟨l,x,r⟩ u v = undefined"
lemma "bst t ⟹ set (in_range t u v) = { x∈set_tree t. u≤x ∧ x≤v }"
oops
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t)"
oops
consts
in_range :: "'a tree ⇒ 'a ⇒ 'a ⇒ 'a list"
Found termination order: "{}"

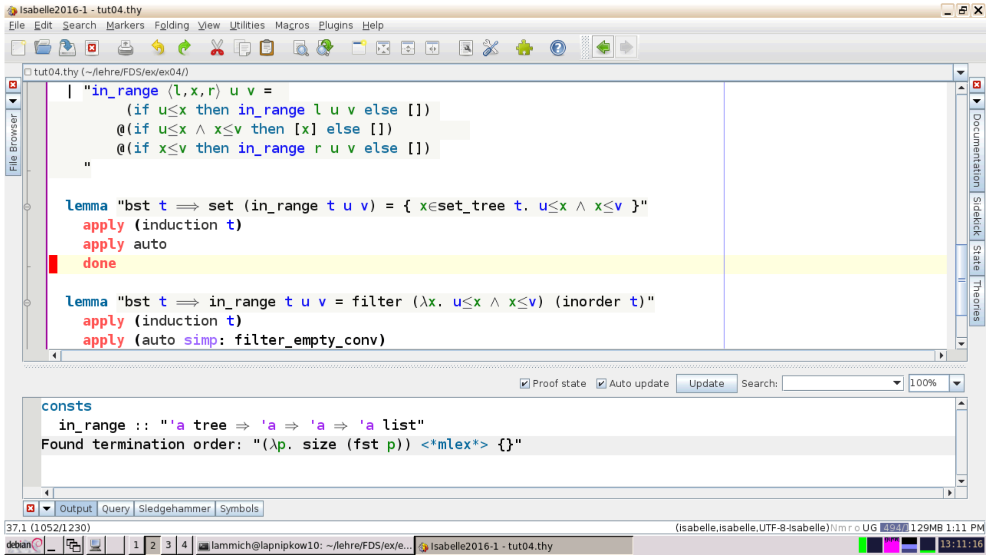

Isabelle2016-1 - tut04.thy
| "in_range ⟨l,x,r⟩ u v =
(if u≤x then in_range l u v else [])
@(if u≤x ∧ x≤v then [x] else [])
@(if x≤v then in_range r u v else [])
"
lemma "bst t ⟹ set (in_range t u v) = { x∈set_tree t. u≤x ∧ x≤v }"
oops
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t)"
oops
proof (prove)
goal (1 subgoal):
1. bst t ⟹ set (in_range t u v) = {x ∈ set_tree t. u ≤ x ∧ x ≤ v}

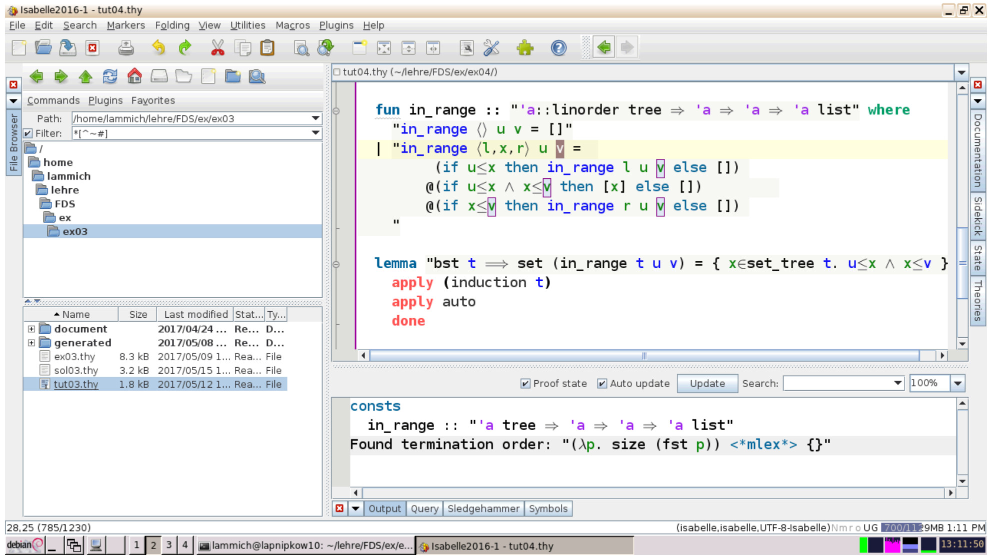

Isabelle2016-1 - tut04.thy (modified)
(if u≤x then in_range l u v else [])
@(if u≤x ∧ x≤v then [x] else [])
@(if x≤v then in_range r u v else [])
"
lemma "bst t ⟹ set (in_range t u v) = { x∈set_tree t. u≤x ∧ x≤v }"
appl
apply (keyword)
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t)"
oops
proof (prove)
goal (1 subgoal):
1. bst t ⟹ set (in_range t u v) = {x ∈ set_tree t. u ≤ x ∧ x ≤ v}

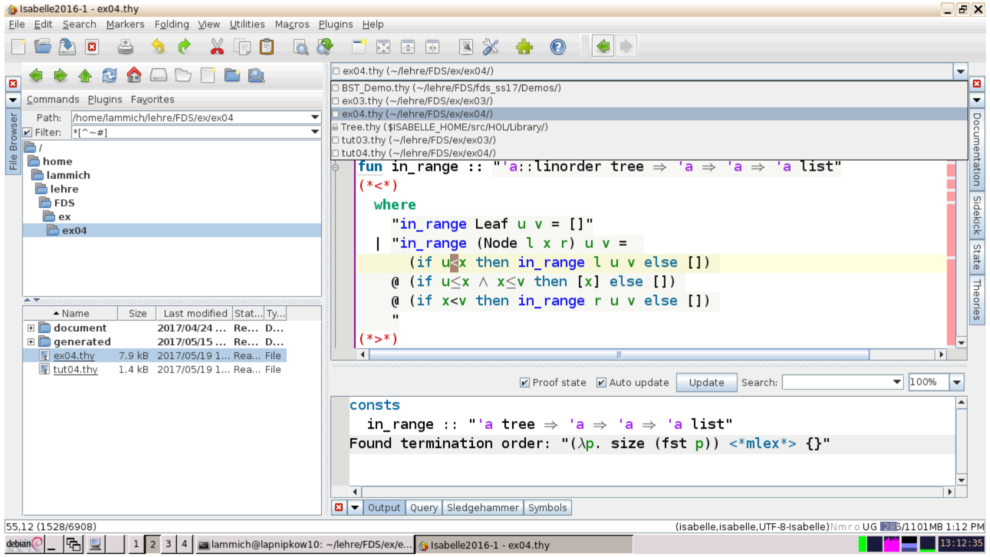

Isabelle2016-1 - tut04.thy
lemma "bst t ⟹ set (in_range t u v) = { x∈set_tree t. u≤x ∧ x≤v }"
apply (induction t)
apply auto
done
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t)"
apply (induction t)
apply auto
end
proof (prove)
goal (4 subgoals):
1. ⋀t1 x2 t2.
⟦in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1; bst t2; ∀x∈set_tree t1. x < x2;
∀x∈set_tree t2. x2 < x; x2 ≤ v; ¬ u ≤ x2⟧
⟹ [x←inorder t1 . u ≤ x ∧ x ≤ v] = []
2. ⋀t1 x2 t2.

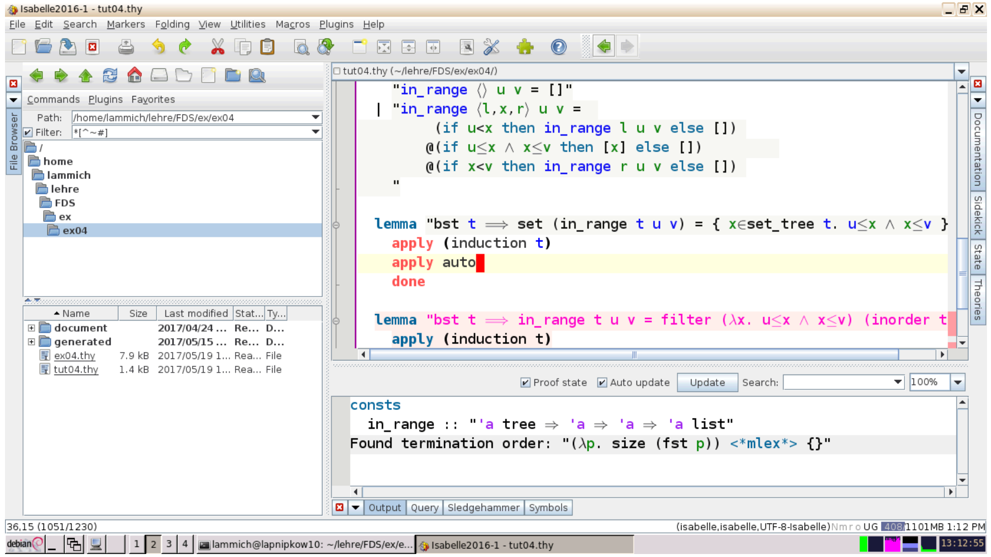

Isabelle2016-1 - tut04.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut04.thy (~/lehre/FDS/ex/ex04/)
lemma "bst t ⟹ set (in_range t u v) = { x∈set_tree t. u≤x ∧ x≤v }"
apply (induction t)
apply auto
done
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t)"
apply (induction t)
apply auto
end
Proof state Auto update Update Search: 100%
proof (prove)
goal (4 subgoals):
1. ⋀t1 x2 t2.
⟦in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1; bst t2; ∀x∈set_tree t1. x < x2;
∀x∈set_tree t2. x2 < x; x2 ≤ v; ¬ u ≤ x2⟧
⟹ [x←inorder t1 . u ≤ x ∧ x ≤ v] = []
2. ⋀t1 x2 t2.
Output Query Sledgehammer Symbols
41,17 (1182/1193)
13:09:49

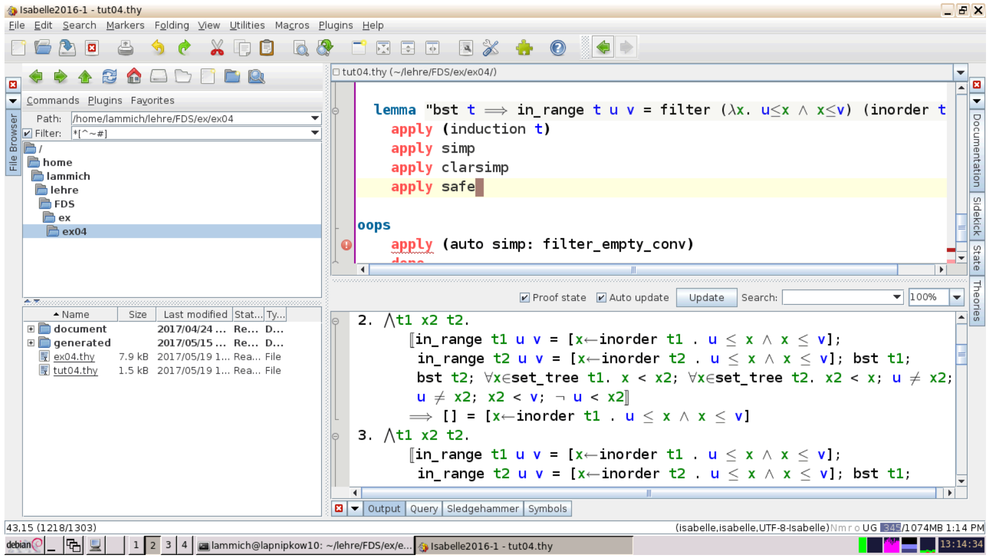

Isabelle2016-1 - tut04.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut04.thy (~/lehre/FDS/ex/ex04/)
apply (induction t)
apply auto
done
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t)"
apply (induction t)
apply auto
find_theorems "filter _ _ = []"
end
Proof state Auto update Update Search: 100%
find_theorems
"filter _ _ = []"
found 3 theorem(s):
• List.filter.simps(1): filter ?P [] = []
• List.filter_empty_conv: (filter ?P ?xs = []) = (∀x∈set ?xs. ¬ ?P x)
• List.filter_False: ∀x∈set ?xs. ¬ ?P x ⟹ filter ?P ?xs = []
Output Query Sledgehammer Symbols
42,37 (1219/1230)
13:10:28

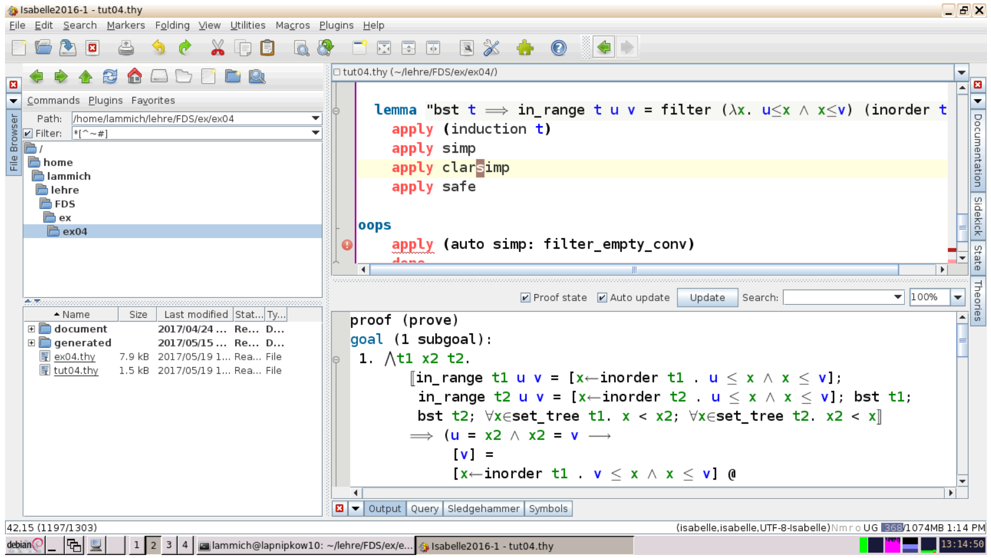

Isabelle2016-1 - tut04.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
tut04.thy (~/lehre/FDS/ex/ex04/)
| "in_range ⟨l,x,r⟩ u v =
(if u≤x then in_range l u v else [])
@(if u≤x ∧ x≤v then [x] else [])
@(if x≤v then in_range r u v else [])
"
lemma "bst t ⟹ set (in_range t u v) = { x∈set_tree t. u≤x ∧ x≤v }"
apply (induction t)
apply auto
done
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t)"
apply (induction t)
apply (auto simp: filter_empty_conv)
Proof state Auto update Update Search: 100%
consts
in_range :: "'a tree ⇒ 'a ⇒ 'a ⇒ 'a list"
Found termination order: "(λp. size (fst p)) <*mlex*> {}"
Output Query Sledgehammer Symbols
37,1 (1052/1230)
13:11:16

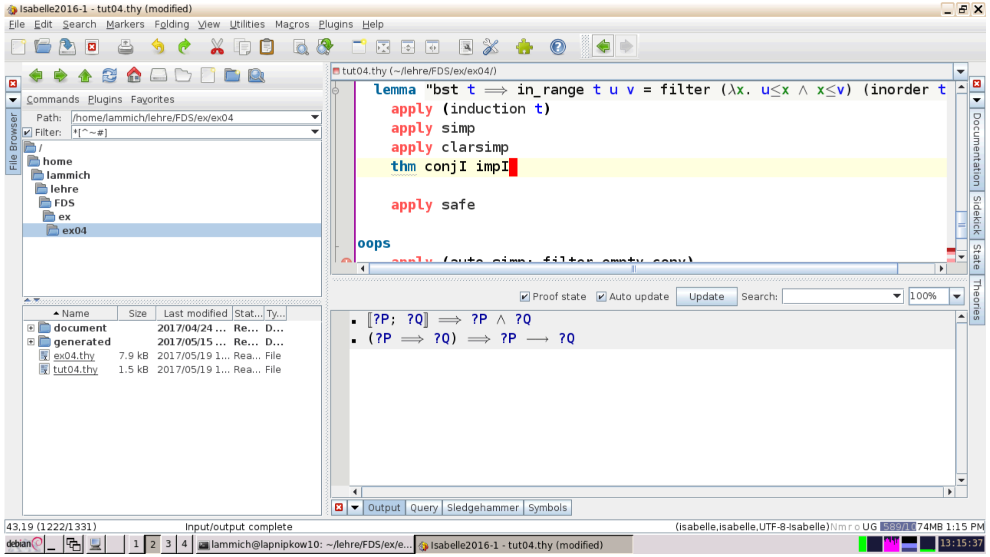

Isabelle2016-1 - tut04.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
Commands Plugins Favorites
Path: /home/lammich/lehre/FDS/ex/ex03
home
lammich
lehre
FDS
ex
ex03
Name Size Last modified
document
generated
ex03.thy 8.3 kB
sol03.thy 3.2 kB
tut03.thy 1.8 kB
tut04.thy (~/lehre/FDS/ex/ex04/)
fun in_range :: "'a::linorder tree ⇒ 'a ⇒ 'a ⇒ 'a list" where
"in_range ⟨⟩ u v = []"
| "in_range ⟨l,x,r⟩ u v =
(if u≤x then in_range l u v else [])
@(if u≤x ∧ x≤v then [x] else [])
@(if x≤v then in_range r u v else [])
"
lemma "bst t ⟹ set (in_range t u v) = { x∈set_tree t. u≤x ∧ x≤v }"
apply (induction t)
apply auto
done
Proof state Auto update Update Search: 100%
consts
in_range :: "'a tree ⇒ 'a ⇒ 'a ⇒ 'a list"
Found termination order: "(λp. size (fst p)) <*mlex*> {}"
Output Query Sledgehammer Symbols
28,25 (785/1230)
13:11:50

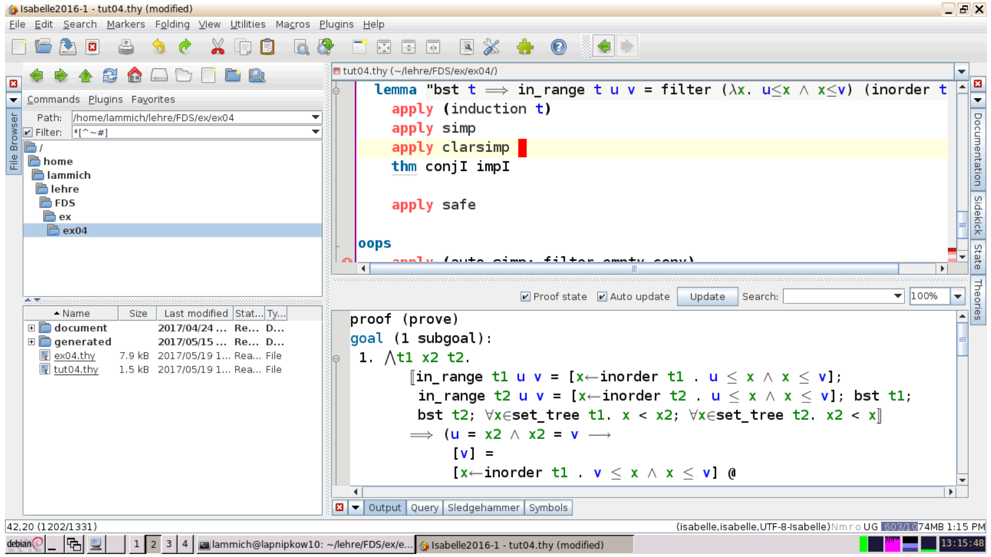

Isabelle2016-1 - ex04.thy
fun in_range :: "'a::linorder tree ⇒ 'a ⇒ 'a ⇒ 'a list"
(*<*)
where
"in_range Leaf u v = []"
| "in_range (Node l x r) u v =
(if u<x then in_range l u v else [])
@ (if u≤x ∧ x≤v then [x] else [])
@ (if x<v then in_range r u v else [])
"
(*>*)
consts
in_range :: "'a tree ⇒ 'a ⇒ 'a ⇒ 'a list"
Found termination order: "(λp. size (fst p)) <*mlex*> {}"
Output Query Sledgehammer Symbols

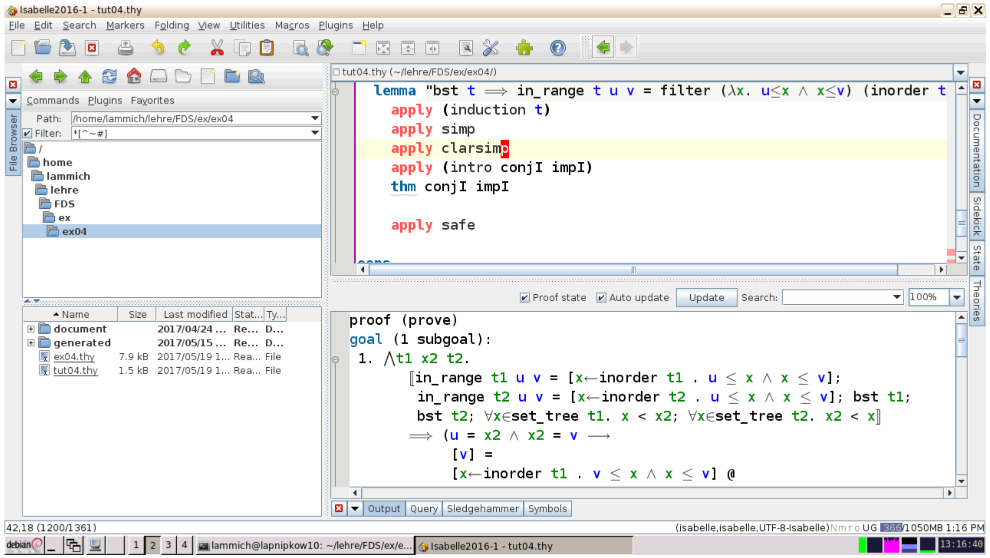

Isabelle2016-1 - tut04.thy
"in_range ⟨⟩ u v = []"
| "in_range ⟨l,x,r⟩ u v =
(if u<x then in_range l u v else [])
@(if u≤x ∧ x≤v then [x] else [])
@(if x<v then in_range r u v else [])
"
lemma "bst t ⟹ set (in_range t u v) = { x∈set_tree t. u≤x ∧ x≤v }"
apply (induction t)
apply auto
done
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t"
apply (induction t)
consts
in_range :: "'a tree ⇒ 'a ⇒ 'a ⇒ 'a list"
Found termination order: "(λp. size (fst p)) <*mlex*> {}"
Output Query Sledgehammer Symbols

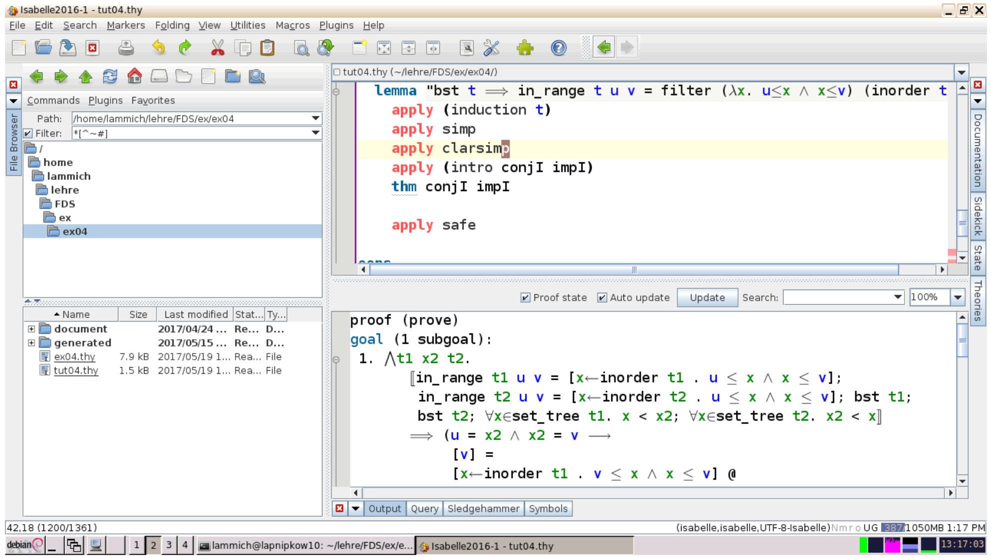

Isabelle2016-1 - tut04.thy
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
apply safe
oops
apply (auto simp: filter_empty_conv)
2. ⋀t1 x2 t2.
⟦in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < x2; ∀x∈set_tree t2. x2 < x; u ≠ x2;
u ≠ x2; x2 < v; ¬ u < x2⟧
⟹ [] = [x←inorder t1 . u ≤ x ∧ x ≤ v]
3. ⋀t1 x2 t2.
⟦in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1;
Output Query Sledgehammer Symbols

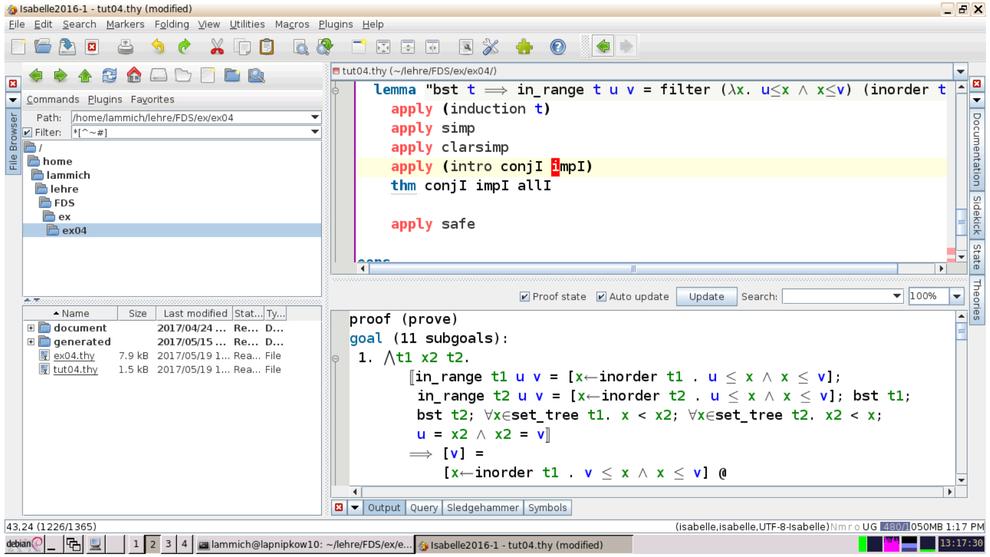

Isabelle2016-1 - tut04.thy
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
apply safe
oops
apply (auto simp: filter_empty_conv)
proof (prove)
goal (1 subgoal):
1. ⋀t1 x2 t2.
⟦in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < x2; ∀x∈set_tree t2. x2 < x⟧
⟹ (u = x2 ∧ x2 = v ⟶
[v] =
[x←inorder t1 . v ≤ x ∧ x ≤ v] @
Output Query Sledgehammer Symbols

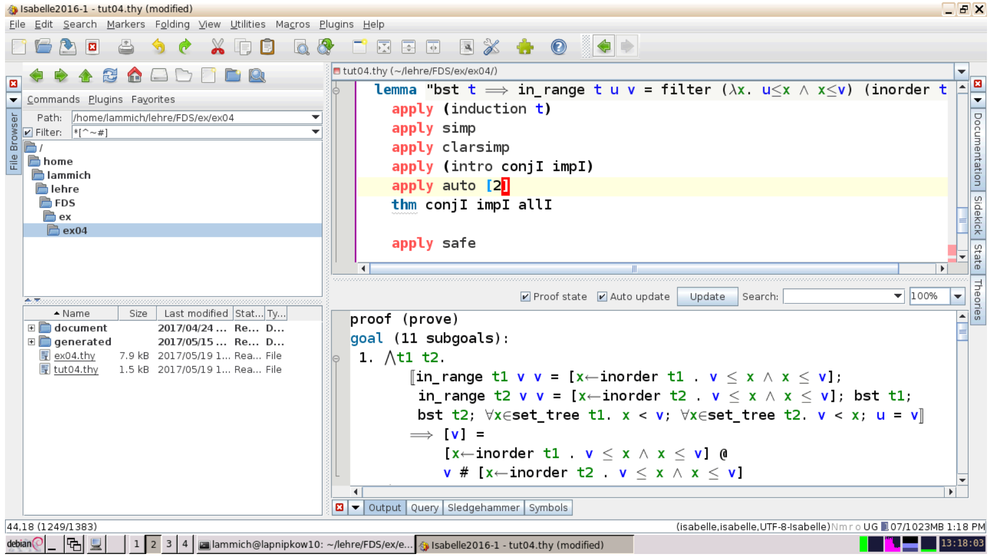

Isabelle2016-1 - tut04.thy (modified)
tut04.thy (~/lehre/FDS/ex/ex04/)
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
thm conjI impI
apply safe
oops
• ⟦?P; ?Q⟧ ⟹ ?P ∧ ?Q
• (?P ⟹ ?Q) ⟹ ?P ⟶ ?Q
Input/output complete

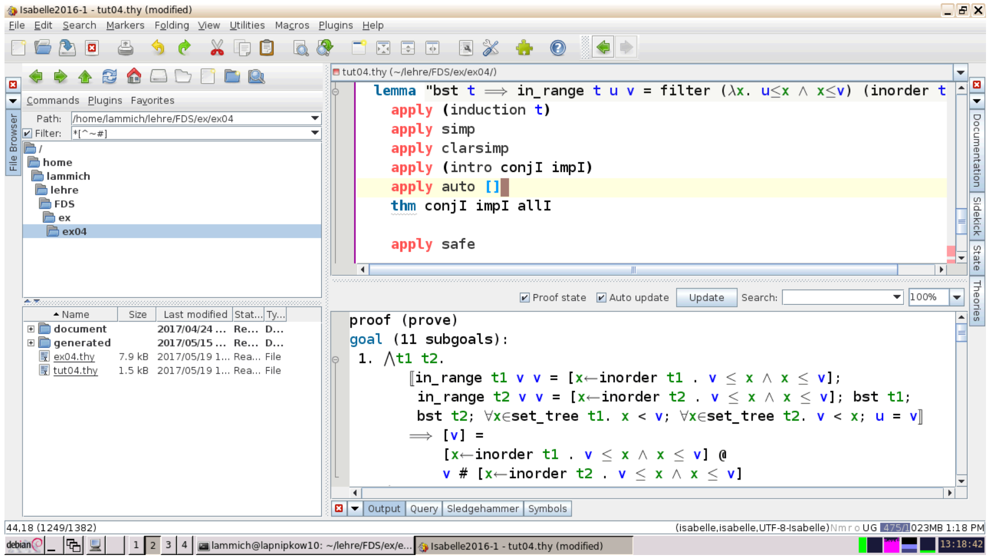

Isabelle2016-1 - tut04.thy (modified)
tut04.thy (~/lehre/FDS/ex/ex04/)
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
thm conjI impI
apply safe
oops
proof (prove)
goal (1 subgoal):
1. ⋀t1 x2 t2.
⟦in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < x2; ∀x∈set_tree t2. x2 < x⟧
⟹ (u = x2 ∧ x2 = v ⟶
[v] =
[x←inorder t1 . v ≤ x ∧ x ≤ v] @

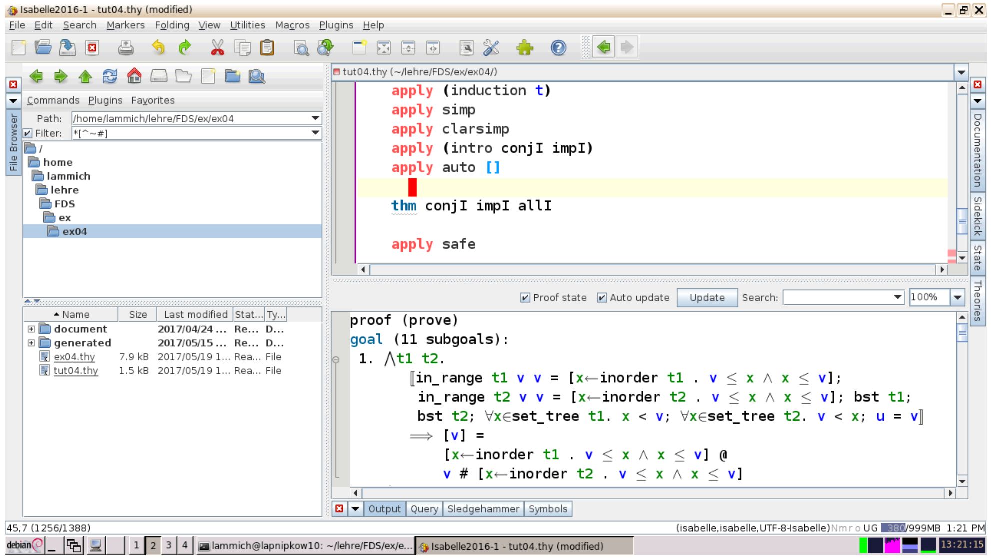

Isabelle2016-1 - tut04.thy
tut04.thy (~/lehre/FDS/ex/ex04/)
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
apply (intro conjI impI)
thm conjI impI
apply safe
proof (prove)
goal (1 subgoal):
1. ⋀t1 x2 t2.
⟦in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < x2; ∀x∈set_tree t2. x2 < x⟧
⟹ (u = x2 ∧ x2 = v ⟶
[v] =
[x←inorder t1 . v ≤ x ∧ x ≤ v] @

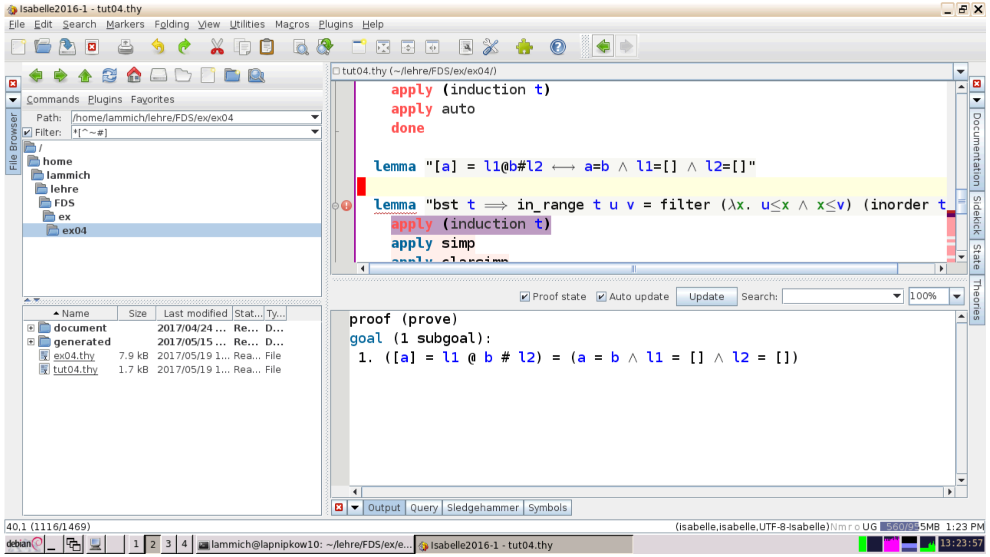

Isabelle2016-1 - tut04.thy
tut04.thy (~/lehre/FDS/ex/ex04/)
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
apply (intro conjI impI)
thm conjI impI
apply safe
proof (prove)
goal (1 subgoal):
1. ⋀t1 x2 t2.
⟦in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < x2; ∀x∈set_tree t2. x2 < x⟧
⟹ (u = x2 ∧ x2 = v ⟶
[v] =
[x←inorder t1 . v ≤ x ∧ x ≤ v] @

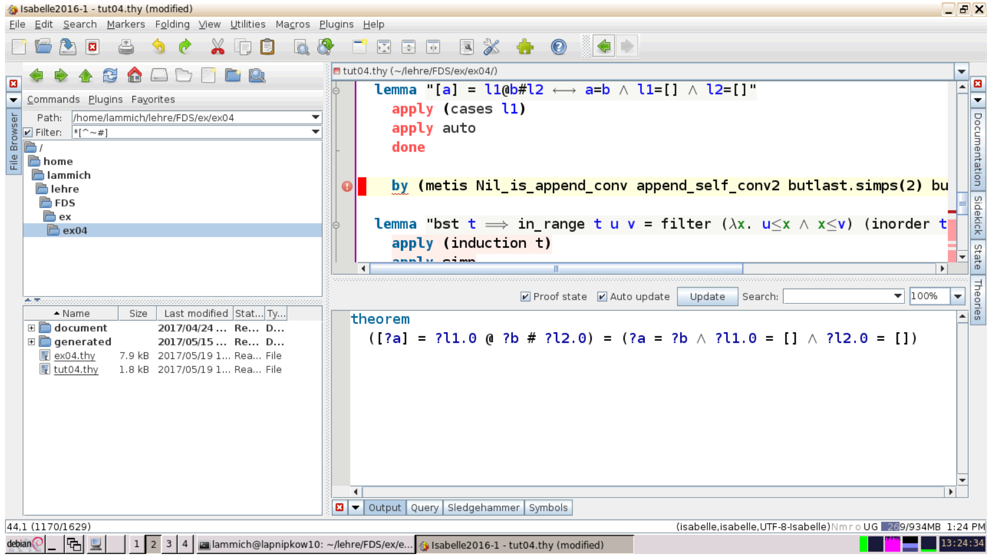

Isabelle2016-1 - tut04.thy (modified)
tut04.thy (~/lehre/FDS/ex/ex04/)
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
apply (intro conjI impI)
thm conjI impI allI
apply safe
proof (prove)
goal (11 subgoals):
1. ⋀t1 x2 t2.
[in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < x2; ∀x∈set_tree t2. x2 < x;
u = x2 ∧ x2 = v]
⟹ [v] =
[x←inorder t1 . v ≤ x ∧ x ≤ v] @

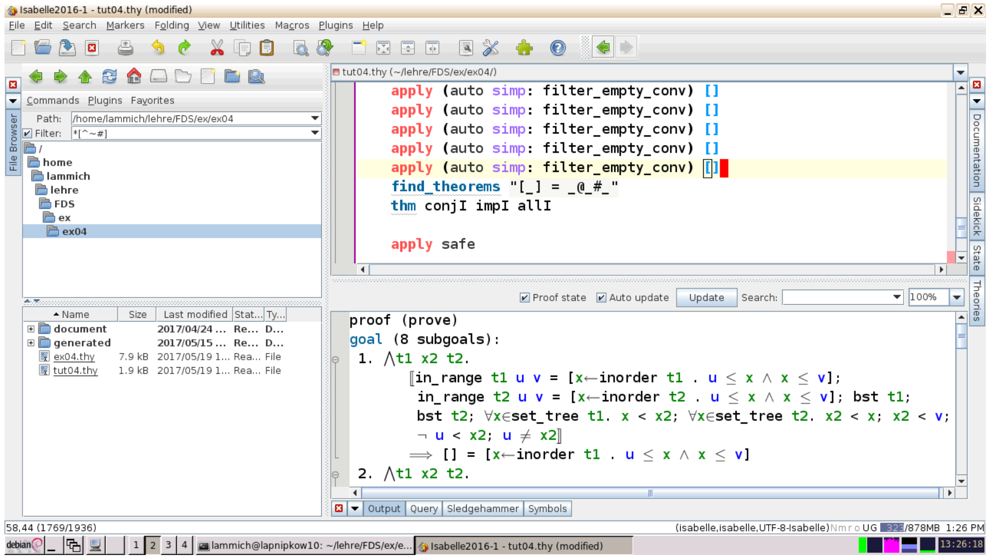

Isabelle2016-1 - tut04.thy (modified)
tut04.thy (~/lehre/FDS/ex/ex04/)
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
apply (intro conjI impI)
apply auto [2]
thm conjI impI allI
apply safe
proof (prove)
goal (11 subgoals):
1. ⋀t1 t2.
[in_range t1 v v = [x←inorder t1 . v ≤ x ∧ x ≤ v];
in_range t2 v v = [x←inorder t2 . v ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < v; ∀x∈set_tree t2. v < x; u = v]
⟹ [v] =
[x←inorder t1 . v ≤ x ∧ x ≤ v] @
v # [x←inorder t2 . v ≤ x ∧ x ≤ v]

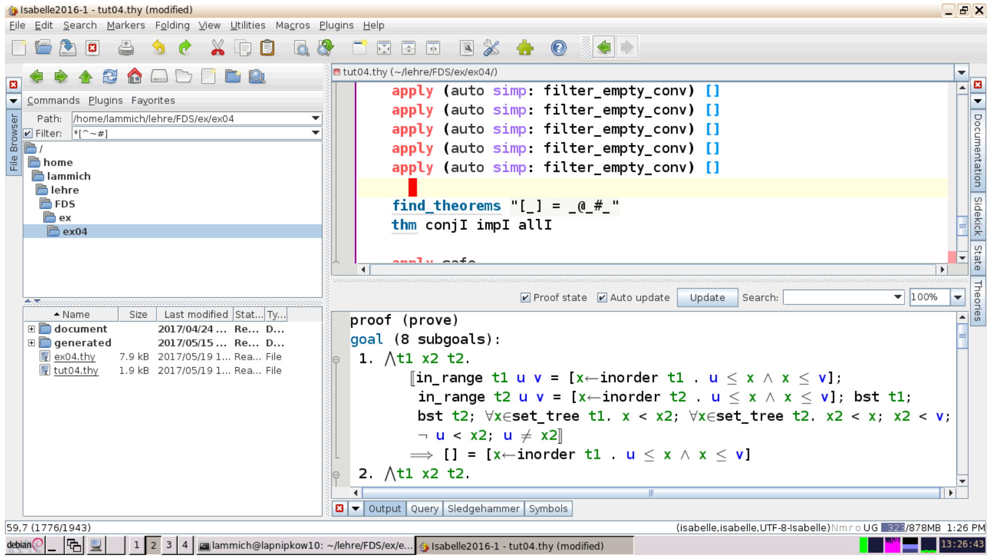

Isabelle2016-1 - tut04.thy (modified)
tut04.thy (~/lehre/FDS/ex/ex04/)
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
apply (intro conjI impI)
apply auto []
thm conjI impI allI
apply safe
proof (prove)
goal (11 subgoals):
1. ⋀t1 t2.
[in_range t1 v v = [x←inorder t1 . v ≤ x ∧ x ≤ v];
in_range t2 v v = [x←inorder t2 . v ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < v; ∀x∈set_tree t2. v < x; u = v]
⟹ [v] =
[x←inorder t1 . v ≤ x ∧ x ≤ v] @
v # [x←inorder t2 . v ≤ x ∧ x ≤ v]

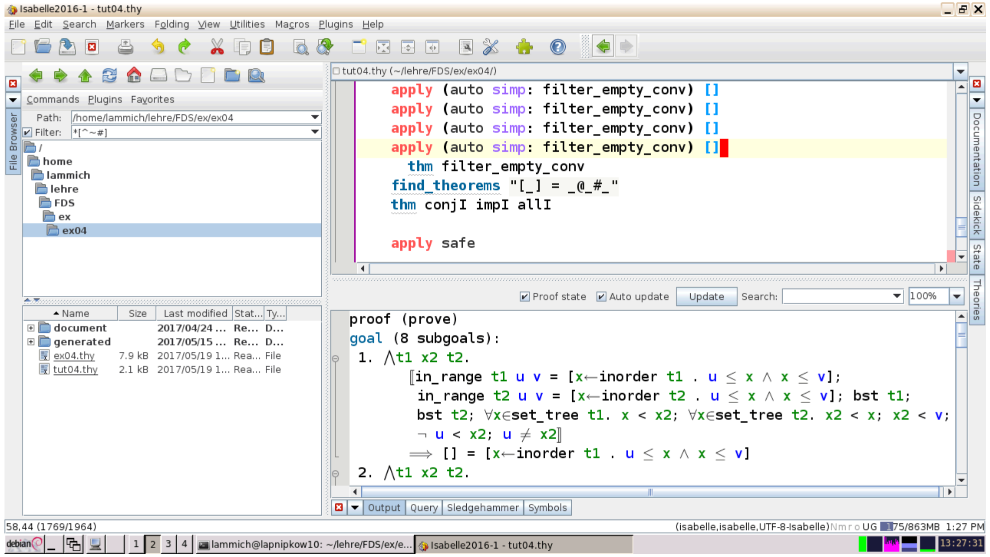
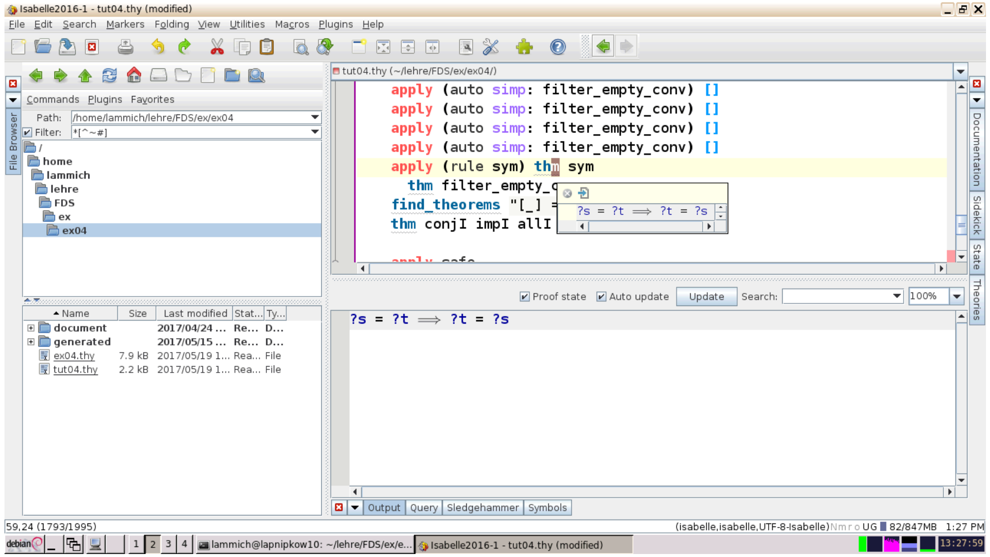
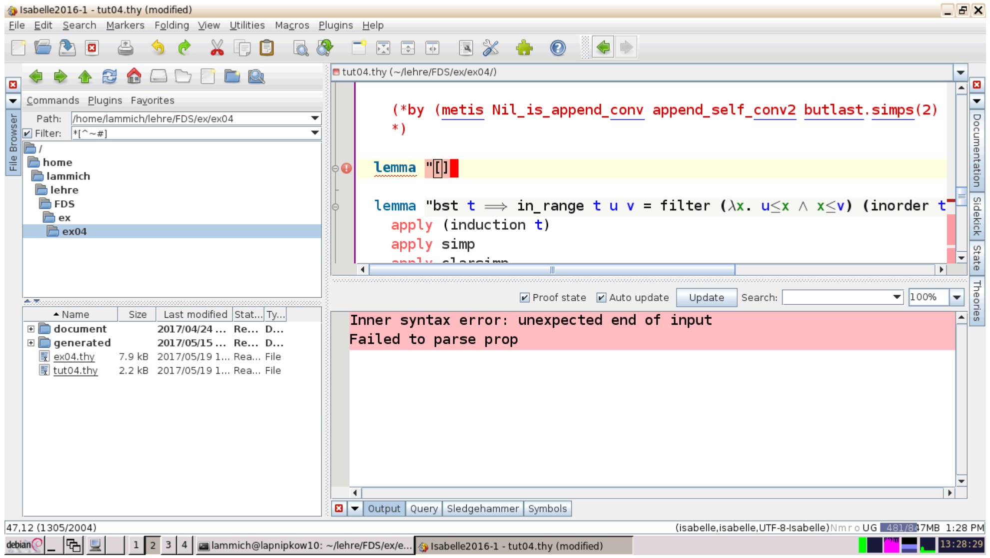
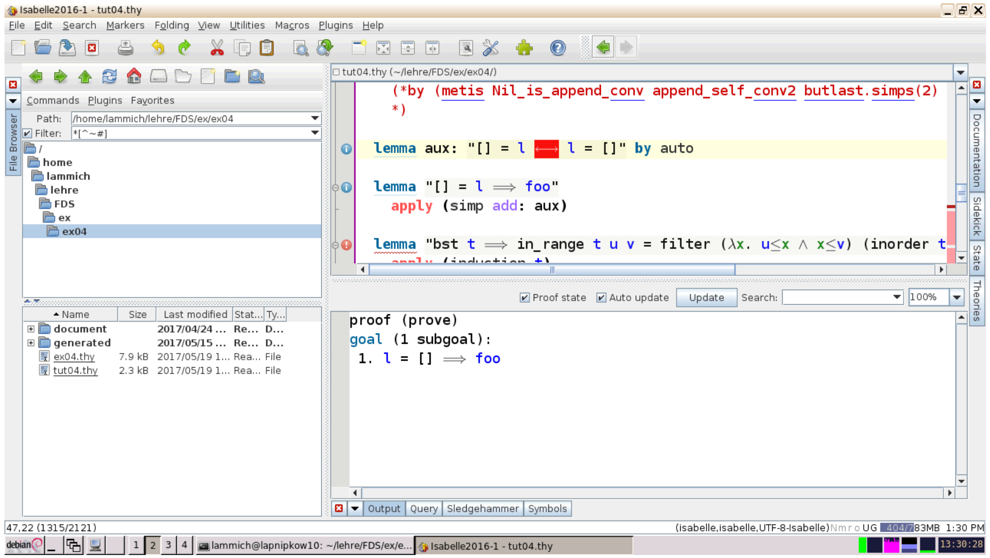

Isabelle2016-1 - tut04.thy (modified)
tut04.thy (~/lehre/FDS/ex/ex04/)
apply (induction t)
apply simp
apply clarsimp
apply (intro conjI impI)
apply auto []
thm conjI impI allI
apply safe
proof (prove)
goal (11 subgoals):
1. ⋀t1 t2.
[in_range t1 v v = [x←inorder t1 . v ≤ x ∧ x ≤ v];
in_range t2 v v = [x←inorder t2 . v ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < v; ∀x∈set_tree t2. v < x; u = v]
⟹ [v] =
[x←inorder t1 . v ≤ x ∧ x ≤ v] @
v # [x←inorder t2 . v ≤ x ∧ x ≤ v]

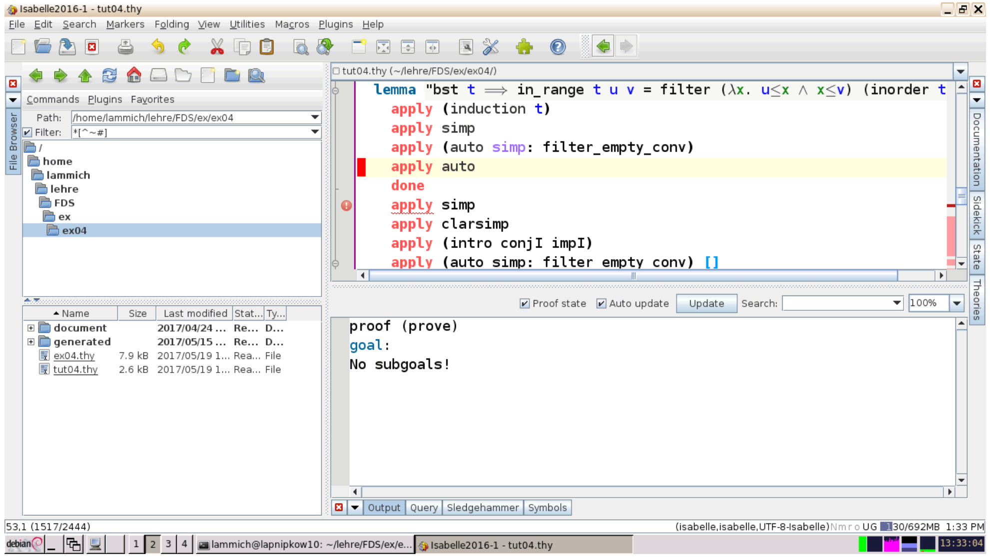

Isabelle2016-1 - tut04.thy
apply (induction t)
apply auto
done
lemma "[a] = l1@b#l2 ⟷ a=b ∧ l1=[] ∧ l2=[]"
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
proof (prove)
goal (1 subgoal):
1. ([a] = l1 @ b # l2) = (a = b ∧ l1 = [] ∧ l2 = [])

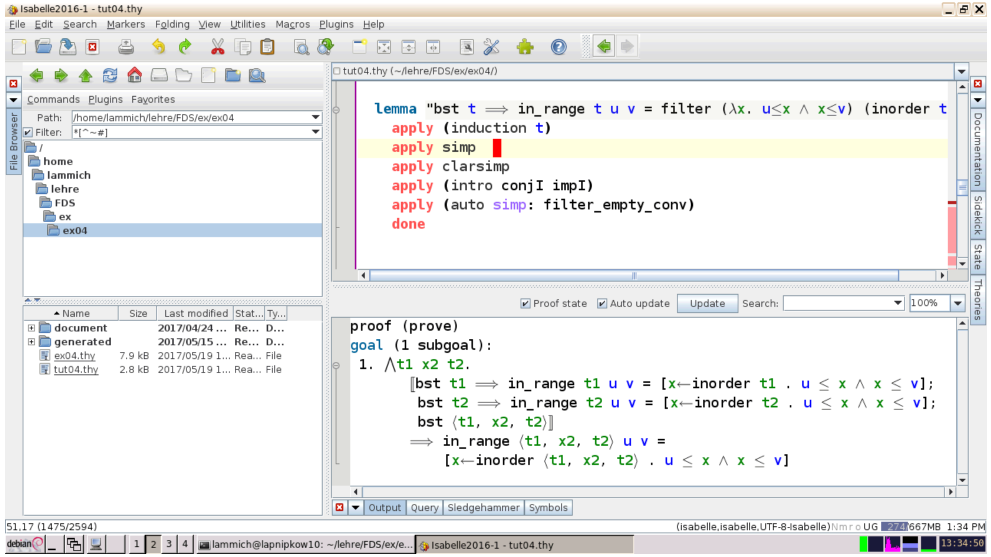

Isabelle2016-1 - tut04.thy (modified)
lemma "[a] = l1@b#l2 ⟷ a=b ∧ l1=[] ∧ l2=[]"
apply (cases l1)
apply auto
done
by (metis Nil_is_append_conv append_self_conv2 butlast.simps(2) bu
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
theorem
([?a] = ?l1.0 @ ?b # ?l2.0) = (?a = ?b ∧ ?l1.0 = [] ∧ ?l2.0 = [])

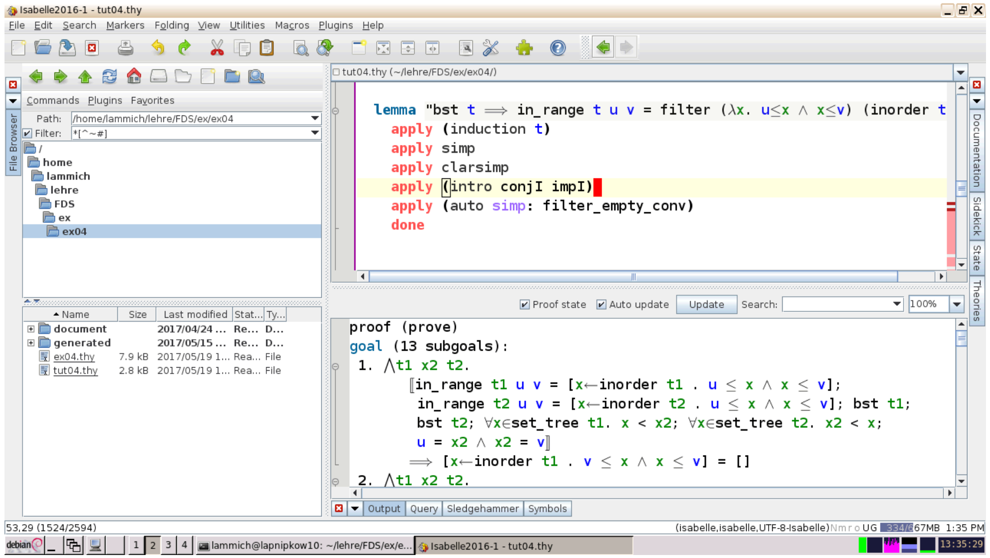

Isabelle2016-1 - tut04.thy (modified)
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
find_theorems "[_] = _@_#_"
thm conjI impI allI
apply safe
proof (prove)
goal (8 subgoals):
1. ⋀t1 x2 t2.
⟦in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < x2; ∀x∈set_tree t2. x2 < x; x2 < v;
¬ u < x2; u ≠ x2⟧
⟹ [] = [x←inorder t1 . u ≤ x ∧ x ≤ v]
2. ⋀t1 x2 t2.

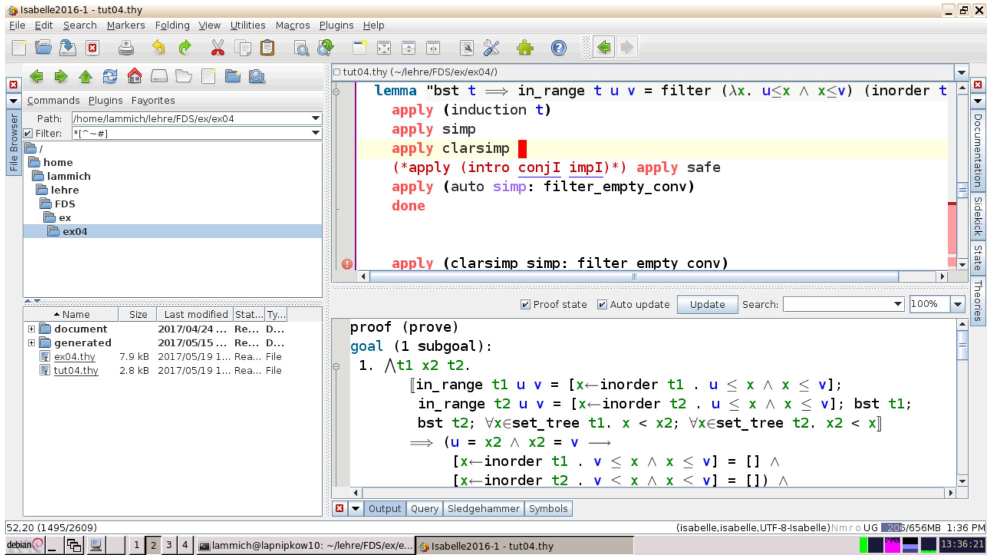

Isabelle2016-1 - tut04.thy (modified)
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
find_theorems "[_] = _@_#_"
thm conjI impI allI
proof (prove)
goal (8 subgoals):
1. ⋀t1 x2 t2.
⟦in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < x2; ∀x∈set_tree t2. x2 < x; x2 < v;
¬ u < x2; u ≠ x2⟧
⟹ [] = [x←inorder t1 . u ≤ x ∧ x ≤ v]
2. ⋀t1 x2 t2.

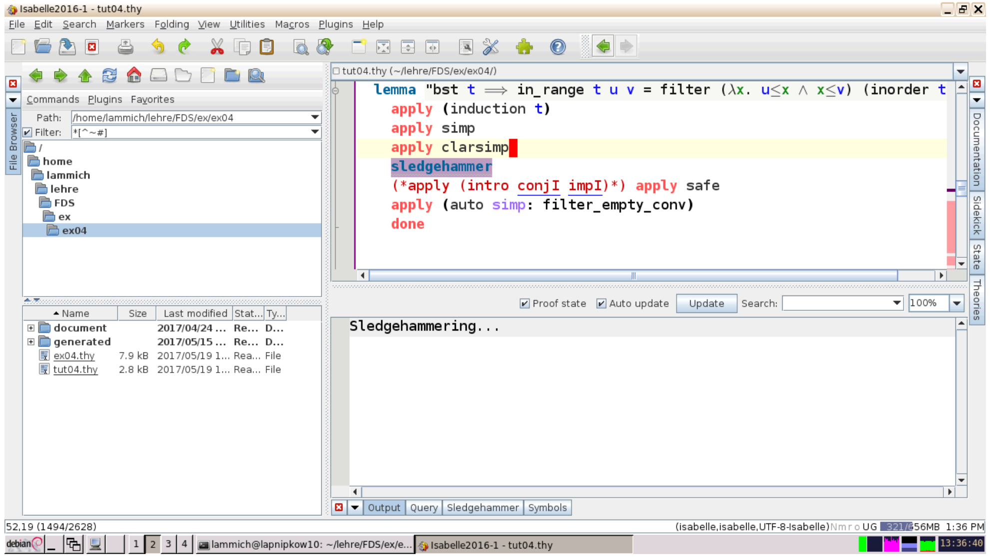

Isabelle2016-1 - tut04.thy
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
thm filter_empty_conv
find_theorems "[_] = _@_#_"
thm conjI impI allI
apply safe
proof (prove)
goal (8 subgoals):
1. ⋀t1 x2 t2.
[in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < x2; ∀x∈set_tree t2. x2 < x; x2 < v;
¬ u < x2; u ≠ x2]
⟹ [] = [x←inorder t1 . u ≤ x ∧ x ≤ v]
2. ⋀t1 x2 t2.

Isabelle2016-1 - tut04.thy (modified)
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
apply (auto simp: filter_empty_conv) []
apply (rule sym) thm sym
thm filter_empty_c
find_theorems "[_]
thm conjI impI allI
?s = ?t ⟹ ?t = ?s
?s = ?t ⟹ ?t = ?s

Isabelle2016-1 - tut04.thy (modified)
(*by (metis Nil_is_append_conv append_self_conv2 butlast.simps(2)
*)
lemma "[]
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
Inner syntax error: unexpected end of input
Failed to parse prop

Isabelle2016-1 - tut04.thy
(*by (metis Nil_is_append_conv append_self_conv2 butlast.simps(2)
*)
lemma aux: "[] = l ___ l = []" by auto
lemma "[] = l ⟹ foo"
apply (simp add: aux)
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
proof (prove)
goal (1 subgoal):
1. l = [] ⟹ foo

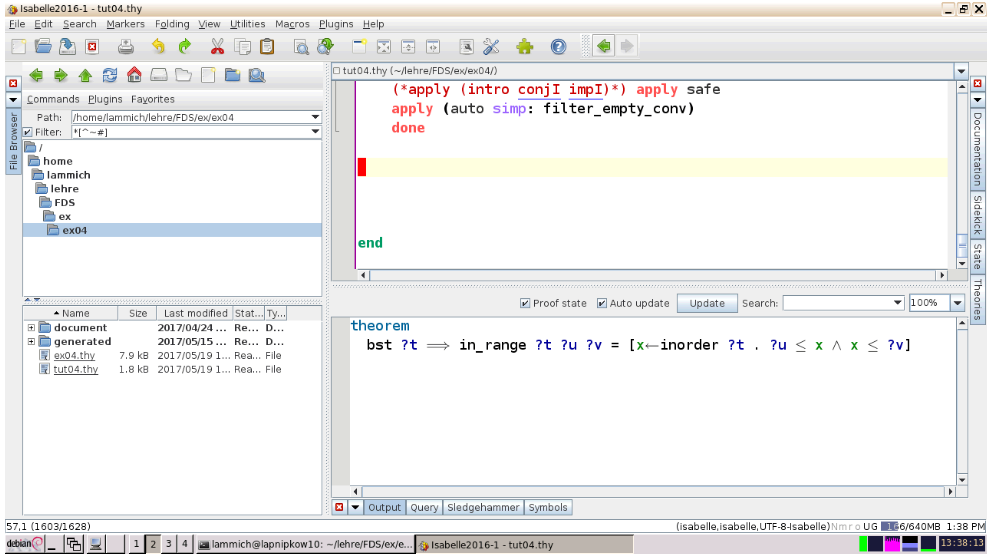

Isabelle2016-1 - tut04.thy
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply (auto simp: filter_empty_conv)
apply auto
done
apply simp
apply clarsimp
apply (intro conjI impI)
apply (auto simp: filter_empty_conv) []
proof (prove)
goal:
No subgoals!

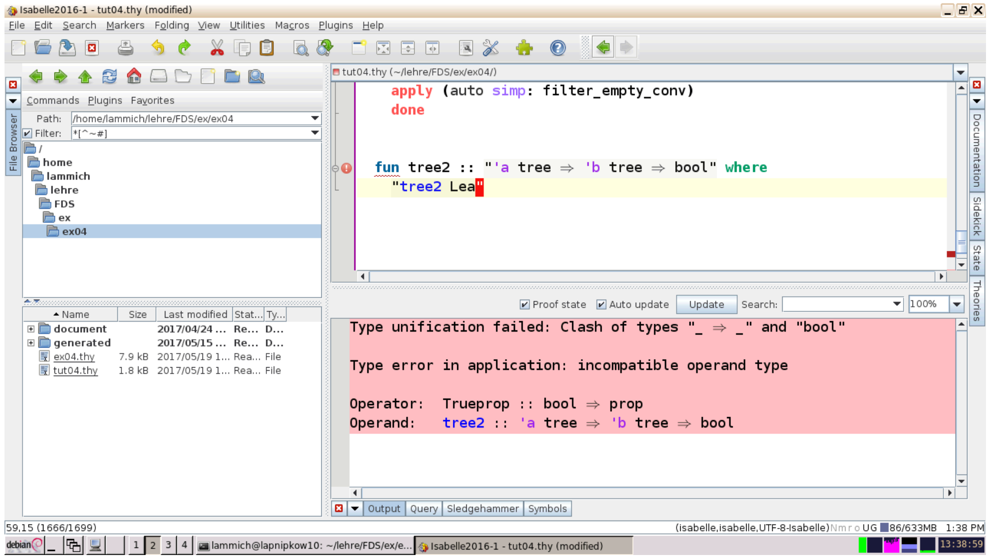

Isabelle2016-1 - tut04.thy
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
apply (intro conjI impI)
apply (auto simp: filter_empty_conv)
done
proof (prove)
goal (1 subgoal):
1. ⋀t1 x2 t2.
[bst t1 ⟹ in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
bst t2 ⟹ in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v];
bst ⟨t1, x2, t2⟩]
⟹ in_range ⟨t1, x2, t2⟩ u v =
[x←inorder ⟨t1, x2, t2⟩ . u ≤ x ∧ x ≤ v]

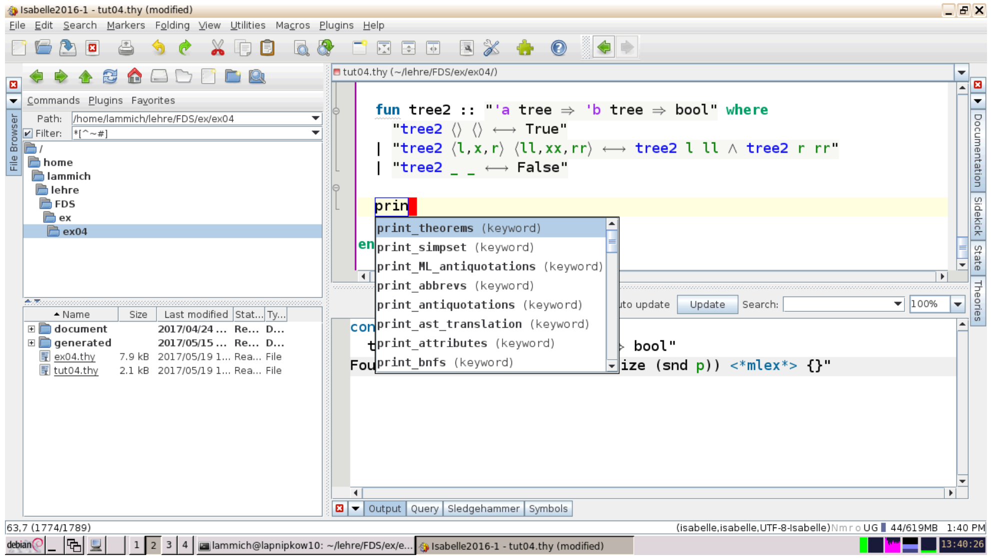

Isabelle2016-1 - tut04.thy
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
apply (intro conjI impI)
apply (auto simp: filter_empty_conv)
done
proof (prove)
goal (13 subgoals):
1. ⋀t1 x2 t2.
[in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < x2; ∀x∈set_tree t2. x2 < x;
u = x2 ∧ x2 = v]
⟹ [x←inorder t1 . v ≤ x ∧ x ≤ v] = []
2. ⋀t1 x2 t2.

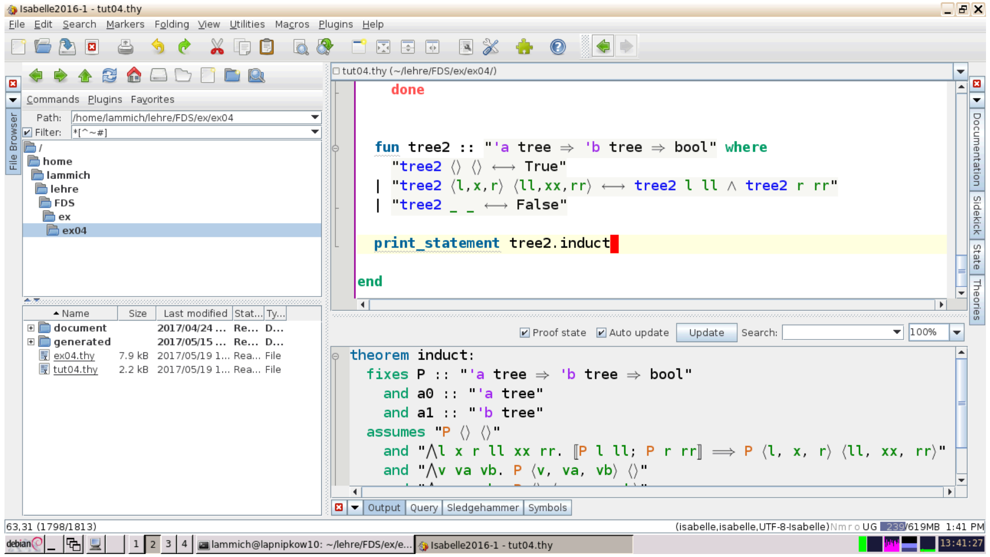

Isabelle2016-1 - tut04.thy
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
(*apply (intro conjI impI)*) apply safe
apply (auto simp: filter_empty_conv)
done
apply (clarsimp simp: filter_empty_conv)
proof (prove)
goal (1 subgoal):
1. ⋀t1 x2 t2.
[in_range t1 u v = [x←inorder t1 . u ≤ x ∧ x ≤ v];
in_range t2 u v = [x←inorder t2 . u ≤ x ∧ x ≤ v]; bst t1;
bst t2; ∀x∈set_tree t1. x < x2; ∀x∈set_tree t2. x2 < x]
⟹ (u = x2 ∧ x2 = v ⟶
[x←inorder t1 . v ≤ x ∧ x ≤ v] = [] ∧
[x←inorder t2 . v ≤ x ∧ x ≤ v] = []) ∧

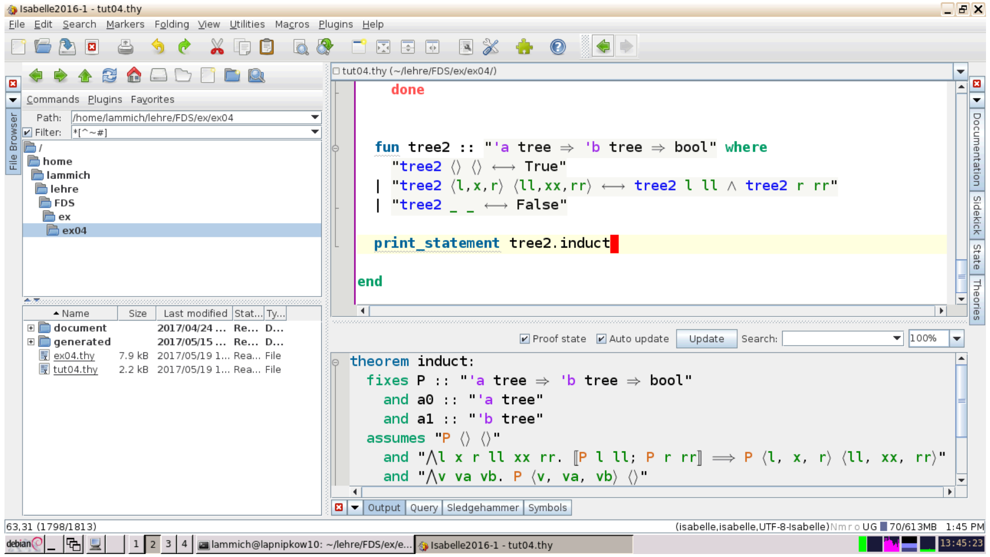

Isabelle2016-1 - tut04.thy
lemma "bst t ⟹ in_range t u v = filter (λx. u≤x ∧ x≤v) (inorder t
apply (induction t)
apply simp
apply clarsimp
sledgehammer
(*apply (intro conjI impI)*) apply safe
apply (auto simp: filter_empty_conv)
done
Sledgehammering...
Output Query Sledgehammer Symbols

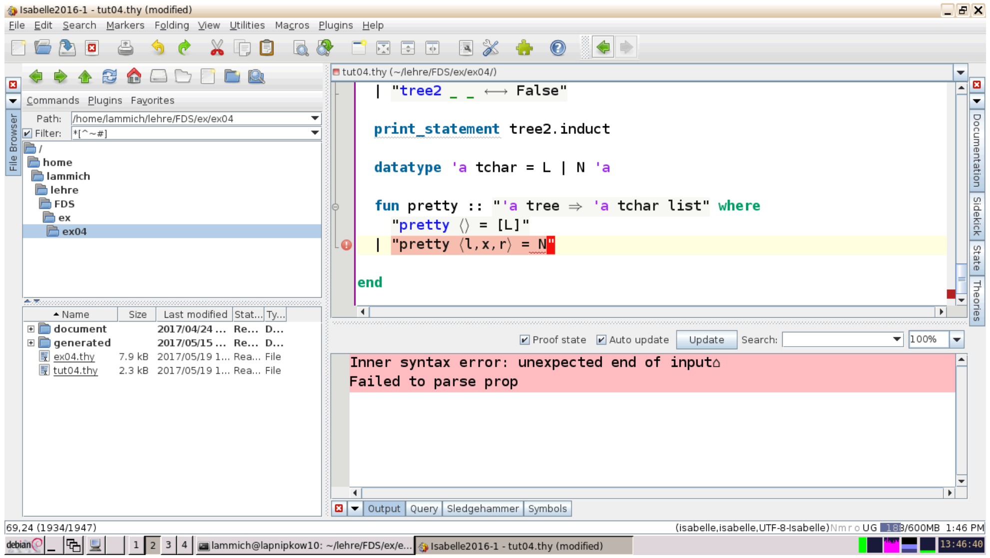

Isabelle2016-1 - tut04.thy
(*apply (intro conjI impI)*) apply safe
apply (auto simp: filter_empty_conv)
done
end
theorem
bst ?t ⟹ in_range ?t ?u ?v = [x←inorder ?t . ?u ≤ x ∧ x ≤ ?v]
Output Query Sledgehammer Symbols

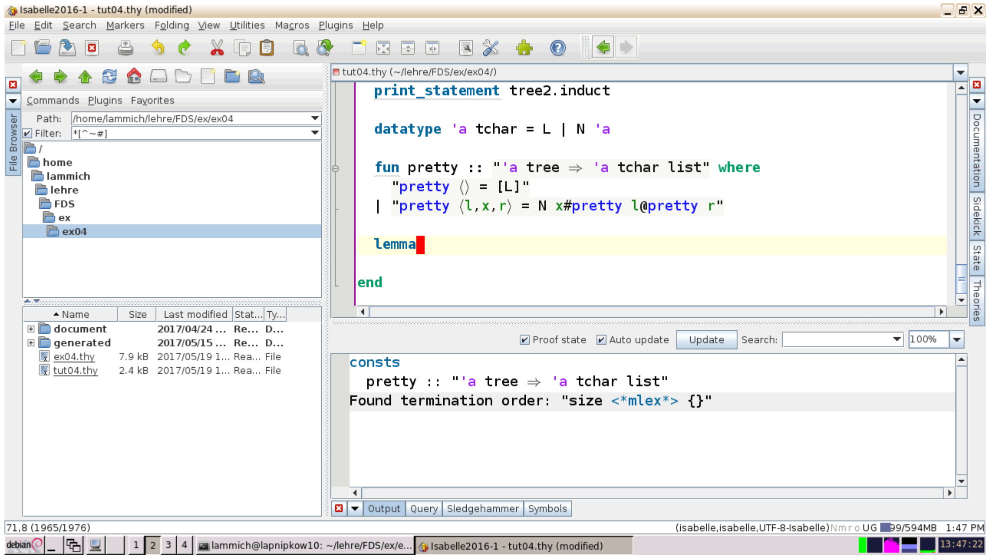

Isabelle2016-1 - tut04.thy (modified)
apply (auto simp: filter_empty_conv)
done
fun tree2 :: "'a tree ⇒ 'b tree ⇒ bool" where
"tree2 Lea
Type unification failed: Clash of types "_ ⇒ _" and "bool"
Type error in application: incompatible operand type
Operator: Trueprop :: bool ⇒ prop
Operand: tree2 :: 'a tree ⇒ 'b tree ⇒ bool
Output Query Sledgehammer Symbols

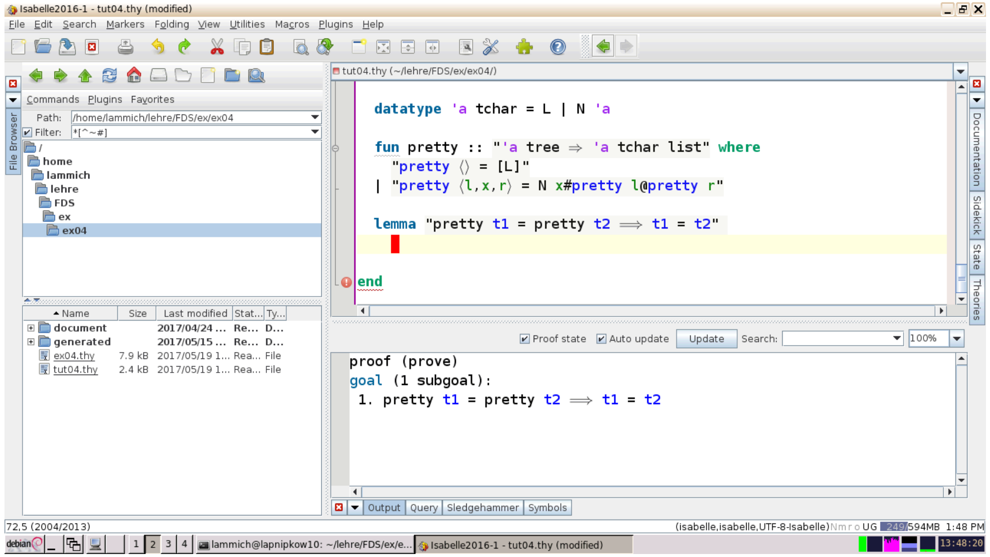

Isabelle2016-1 - tut04.thy (modified)
fun tree2 :: "'a tree ⇒ 'b tree ⇒ bool" where
"tree2 ⟨⟩ ⟨⟩ ⟷ True"
| "tree2 ⟨l,x,r⟩ ⟨ll,xx,rr⟩ ⟷ tree2 l ll ∧ tree2 r rr"
| "tree2 _ _ ⟷ False"
prin
print_theorems (keyword)
print_simpset (keyword)
print_ML_antiquotations (keyword)
print_abbrevs (keyword)
print_antiquotations (keyword)
print_ast_translation (keyword)
print_attributes (keyword)
print_bnfs (keyword)
Output Query Sledgehammer Symbols

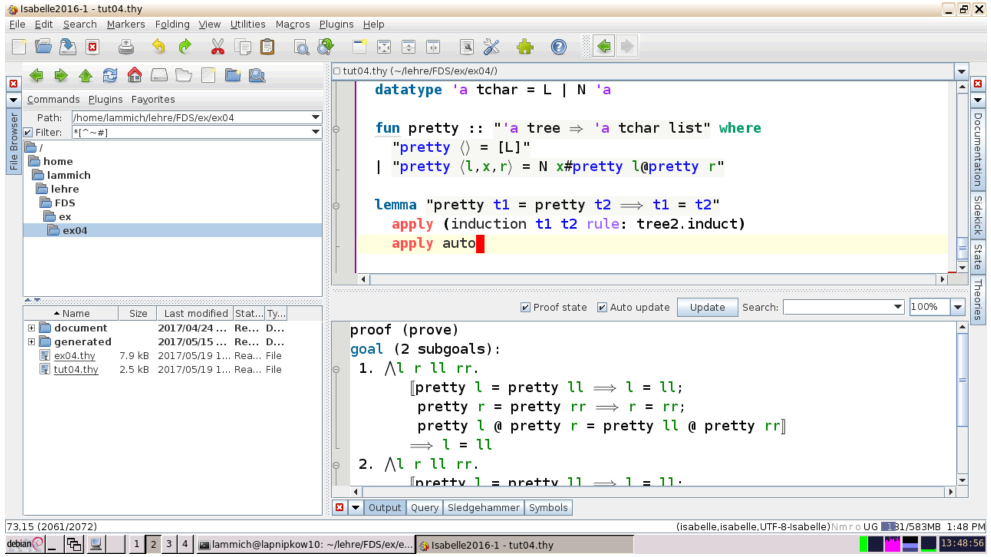

Isabelle2016-1 - tut04.thy
done
fun tree2 :: "'a tree ⇒ 'b tree ⇒ bool" where
"tree2 ⟨⟩ ⟨⟩ ⟷ True"
| "tree2 ⟨l,x,r⟩ ⟨ll,xx,rr⟩ ⟷ tree2 l ll ∧ tree2 r rr"
| "tree2 _ _ ⟷ False"
print_statement tree2.induct
end
theorem induct:
fixes P :: "'a tree ⇒ 'b tree ⇒ bool"
and a0 :: "'a tree"
and a1 :: "'b tree"
assumes "P ⟨⟩ ⟨⟩"
and "⋀l x r ll xx rr. ⟦P l ll; P r rr⟧ ⟹ P ⟨l, x, r⟩ ⟨ll, xx, rr⟩"
and "⋀v va vb. P ⟨v, va, vb⟩ ⟨⟩"

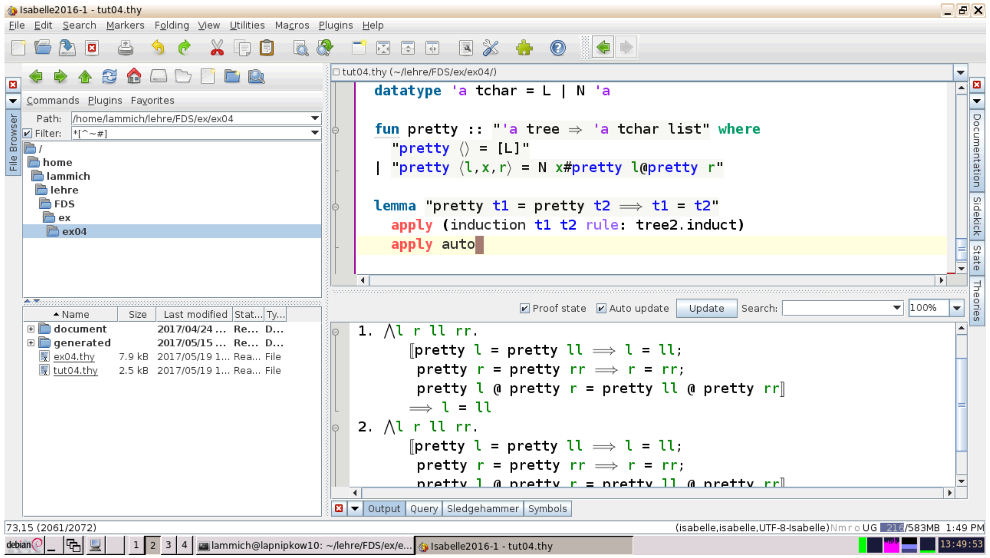

Isabelle2016-1 - tut04.thy
done
fun tree2 :: "'a tree ⇒ 'b tree ⇒ bool" where
"tree2 ⟨⟩ ⟨⟩ ⟷ True"
| "tree2 ⟨l,x,r⟩ ⟨ll,xx,rr⟩ ⟷ tree2 l ll ∧ tree2 r rr"
| "tree2 _ _ ⟷ False"
print_statement tree2.induct
end
theorem induct:
fixes P :: "'a tree ⇒ 'b tree ⇒ bool"
and a0 :: "'a tree"
and a1 :: "'b tree"
assumes "P ⟨⟩ ⟨⟩"
and "⋀l x r ll xx rr. ⟦P l ll; P r rr⟧ ⟹ P ⟨l, x, r⟩ ⟨ll, xx, rr⟩"
and "⋀v va vb. P ⟨v, va, vb⟩ ⟨⟩"

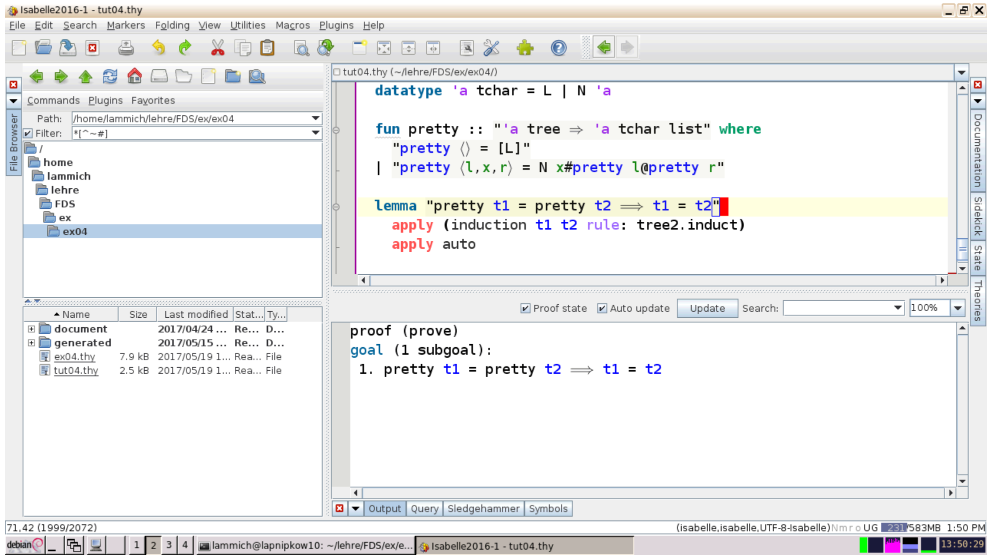

Isabelle2016-1 - tut04.thy (modified)
| "tree2 _ _ ⟷ False"
print_statement tree2.induct
datatype 'a tchar = L | N 'a
fun pretty :: "'a tree ⇒ 'a tchar list" where
"pretty ⟨⟩ = [L]"
| "pretty ⟨l,x,r⟩ = N"
end
Inner syntax error: unexpected end of input⌂
Failed to parse prop

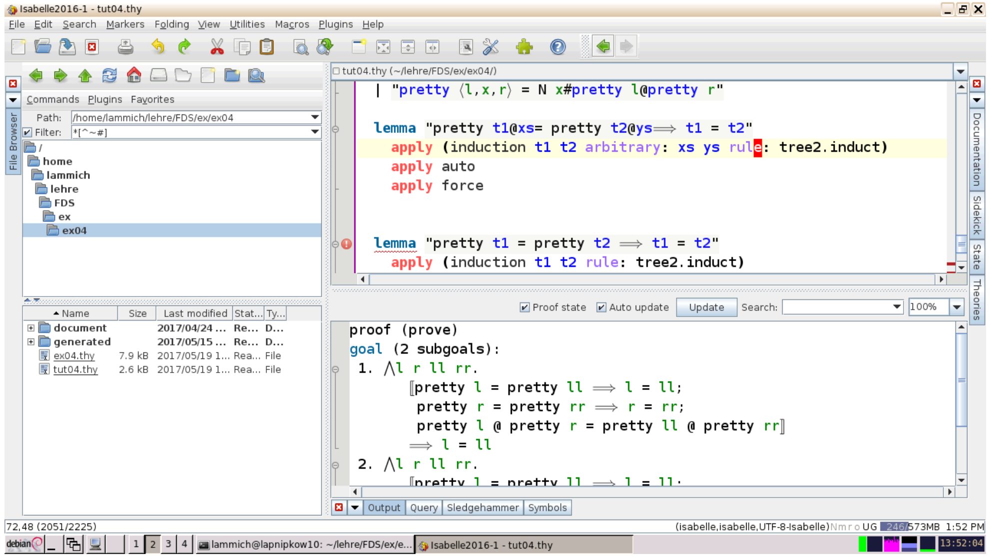
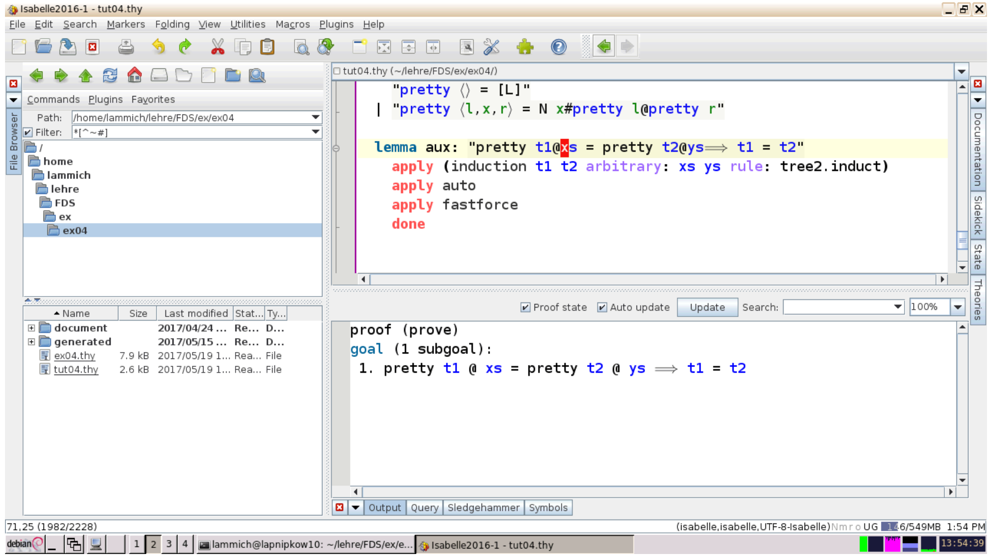
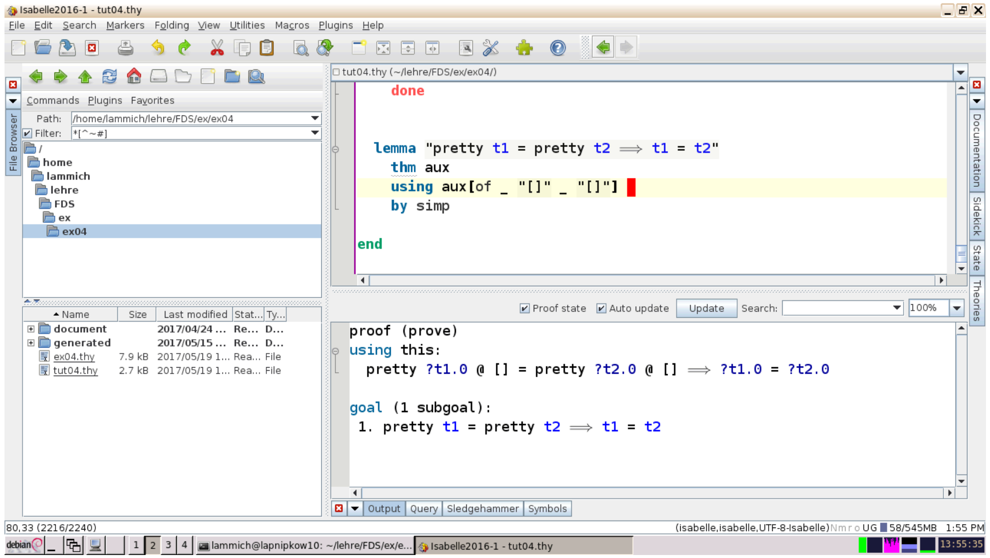
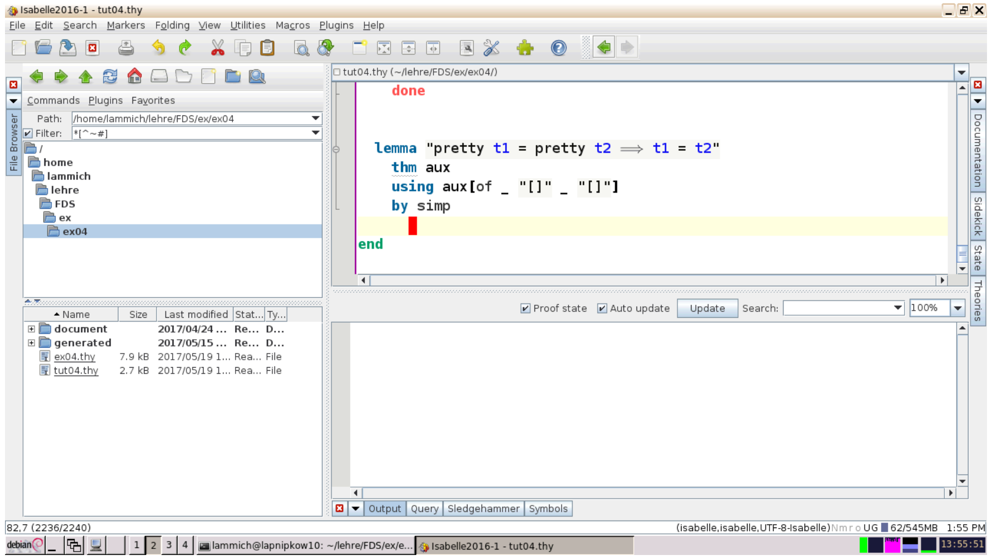

Isabelle2016-1 - tut04.thy (modified)
print_statement tree2.induct
datatype 'a tchar = L | N 'a
fun pretty :: "'a tree ⇒ 'a tchar list" where
"pretty ⟨⟩ = [L]"
| "pretty ⟨l,x,r⟩ = N x#pretty l@pretty r"
lemma
end
consts
pretty :: "'a tree ⇒ 'a tchar list"
Found termination order: "size <*mlex*> {}"


Isabelle2016-1 - tut04.thy (modified)
datatype 'a tchar = L | N 'a
fun pretty :: "'a tree ⇒ 'a tchar list" where
"pretty ⟨⟩ = [L]"
| "pretty ⟨l,x,r⟩ = N x#pretty l@pretty r"
lemma "pretty t1 = pretty t2 ⟹ t1 = t2"
end
proof (prove)
goal (1 subgoal):
1. pretty t1 = pretty t2 ⟹ t1 = t2


Isabelle2016-1 - tut04.thy
datatype 'a tchar = L | N 'a
fun pretty :: "'a tree ⇒ 'a tchar list" where
"pretty ⟨⟩ = [L]"
| "pretty ⟨l,x,r⟩ = N x#pretty l@pretty r"
lemma "pretty t1 = pretty t2 ⟹ t1 = t2"
apply (induction t1 t2 rule: tree2.induct)
apply auto
proof (prove)
goal (2 subgoals):
1. ⋀l r ll rr.
⟦pretty l = pretty ll ⟹ l = ll;
pretty r = pretty rr ⟹ r = rr;
pretty l @ pretty r = pretty ll @ pretty rr⟧
⟹ l = ll
2. ⋀l r ll rr.


Isabelle2016-1 - tut04.thy
datatype 'a tchar = L | N 'a
fun pretty :: "'a tree ⇒ 'a tchar list" where
"pretty ⟨⟩ = [L]"
| "pretty ⟨l,x,r⟩ = N x#pretty l@pretty r"
lemma "pretty t1 = pretty t2 ⟹ t1 = t2"
apply (induction t1 t2 rule: tree2.induct)
apply auto
1. ⋀l r ll rr.
⟦pretty l = pretty ll ⟹ l = ll;
pretty r = pretty rr ⟹ r = rr;
pretty l @ pretty r = pretty ll @ pretty rr⟧
⟹ l = ll
2. ⋀l r ll rr.
⟦pretty l = pretty ll ⟹ l = ll;
pretty r = pretty rr ⟹ r = rr;


Isabelle2016-1 - tut04.thy
datatype 'a tchar = L | N 'a
fun pretty :: "'a tree ⇒ 'a tchar list" where
"pretty ⟨⟩ = [L]"
| "pretty ⟨l,x,r⟩ = N x#pretty l@pretty r"
lemma "pretty t1 = pretty t2 ⟹ t1 = t2"
apply (induction t1 t2 rule: tree2.induct)
apply auto
proof (prove)
goal (1 subgoal):
1. pretty t1 = pretty t2 ⟹ t1 = t2


Isabelle2016-1 - tut04.thy
tut04.thy (~/lehre/FDS/ex/ex04/)
| "pretty ⟨l,x,r⟩ = N x#pretty l@pretty r"
lemma "pretty t1@xs= pretty t2@ys⟹ t1 = t2"
apply (induction t1 t2 arbitrary: xs ys rule: tree2.induct)
apply auto
apply force
lemma "pretty t1 = pretty t2 ⟹ t1 = t2"
apply (induction t1 t2 rule: tree2.induct)
proof (prove)
goal (2 subgoals):
1. ⋀l r ll rr.
⟦pretty l = pretty ll ⟹ l = ll;
pretty r = pretty rr ⟹ r = rr;
pretty l @ pretty r = pretty ll @ pretty rr⟧
⟹ l = ll
2. ⋀l r ll rr.
Output Query Sledgehammer Symbols

Isabelle2016-1 - tut04.thy
tut04.thy (~/lehre/FDS/ex/ex04/)
"pretty ⟨⟩ = [L]"
| "pretty ⟨l,x,r⟩ = N x#pretty l@pretty r"
lemma aux: "pretty t1@xs = pretty t2@ys⟹ t1 = t2"
apply (induction t1 t2 arbitrary: xs ys rule: tree2.induct)
apply auto
apply fastforce
done
proof (prove)
goal (1 subgoal):
1. pretty t1 @ xs = pretty t2 @ ys ⟹ t1 = t2
Output Query Sledgehammer Symbols

Isabelle2016-1 - tut04.thy
tut04.thy (~/lehre/FDS/ex/ex04/)
done
lemma "pretty t1 = pretty t2 ⟹ t1 = t2"
thm aux
using aux[of _ "[]" _ "[]"]
by simp
end
proof (prove)
using this:
pretty ?t1.0 @ [] = pretty ?t2.0 @ [] ⟹ ?t1.0 = ?t2.0
goal (1 subgoal):
1. pretty t1 = pretty t2 ⟹ t1 = t2
Output Query Sledgehammer Symbols

Isabelle2016-1 - tut04.thy
tut04.thy (~/lehre/FDS/ex/ex04/)
done
lemma "pretty t1 = pretty t2 ⟹ t1 = t2"
thm aux
using aux[of _ "[]" _ "[]"]
by simp
end
Output Query Sledgehammer Symbols

print_statement *bin_tree2.induct*

Binary trees can be uniquely pretty-printed by emitting a symbol L for a leaf, and a symbol N for a node. Each N is followed by the pretty-prints of the left and right tree. No additional brackets are required!

datatype *'a tchar = L | N 'a*

fun *pretty* :: *"'a tree ⇒ 'a tchar list"*

Show that pretty-printing is actually unique, i.e., no two different trees are pretty-printed the same way. Hint: Auxiliary lemma. Simultaneous induction over both trees.

lemma *pretty_unique*: *"pretty t = pretty t' ⟹ t=t'"*

Homework 4 Delete Minimum

Submission until Friday, May 26, 11:59am.

Define a function to return and delete the minimum element from a non-empty BST. You may omit the equation for the empty tree.

Show that your function preserves the search tree property. Hint: An auxiliary lemma of the form *t≠Leaf ⟹ del_min t = (x,t') ⟹ ...* relating *set_tree t* and *set_tree t'* may be helpful.

lemma *"⟦t≠Leaf; bst t⟧ ⟹ bst (snd (del_min t))"*

Show that your function returns the first element of the inorder traversal, and a tree whose inorder traversal corresponds to the tail of the original inorder traversal. Hint: You may need auxiliary lemmas. Some subgoals may require more *force*ful methods than *auto*.

lemma *"t≠Leaf ⟹ del_min t = (x,t')*
⟹ case inorder t of (y#ys) ⇒ x=y ∧ inorder t' = ys"

Define a function *del_min2* that uses so called continuations: The second argument is a function which is applied to the result. (i.e. the minimum element and the tree of remaining elements)

Do not use *del_min* for the definition, but define *del_min2* recursively, passing down (modified) continuations.

lemma *"⟦t≠Leaf; bst t⟧ ⟹ bst (snd (del_min t))"*

Show that your function returns the first element of the inorder traversal, and a tree whose inorder traversal corresponds to the tail of the original inorder traversal. Hint: You may need auxiliary lemmas. Some subgoals may require more *force*ful methods than *auto*.

lemma *"t≠Leaf ⟹ del_min t = (x,t')*
⟹ case inorder t of (y#ys) ⇒ x=y ∧ inorder t' = ys"

Define a function *del_min2* that uses so called continuations: The second argument is a function which is applied to the result. (i.e. the minimum element and the tree of remaining elements)

Do not use *del_min* for the definition, but define *del_min2* recursively, passing down (modified) continuations.

lemma *"t≠Leaf ⟹ del_min t = (x,t')*
⟹ case inorder t of (y#ys) ⇒ x=y ∧ inorder t' = ys"

Define a function *del_min2* that uses so called continuations: The second argument is a function which is applied to the result. (i.e. the minimum element and the tree of remaining elements)

Do not use *del_min* for the definition, but define *del_min2* recursively, passing down (modified) continuations.