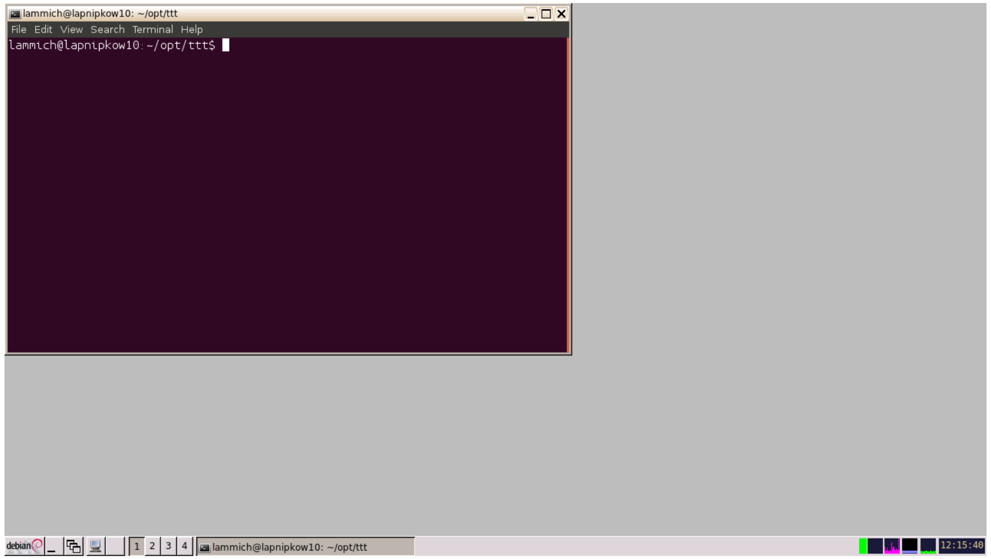
Script **generated by TTT**

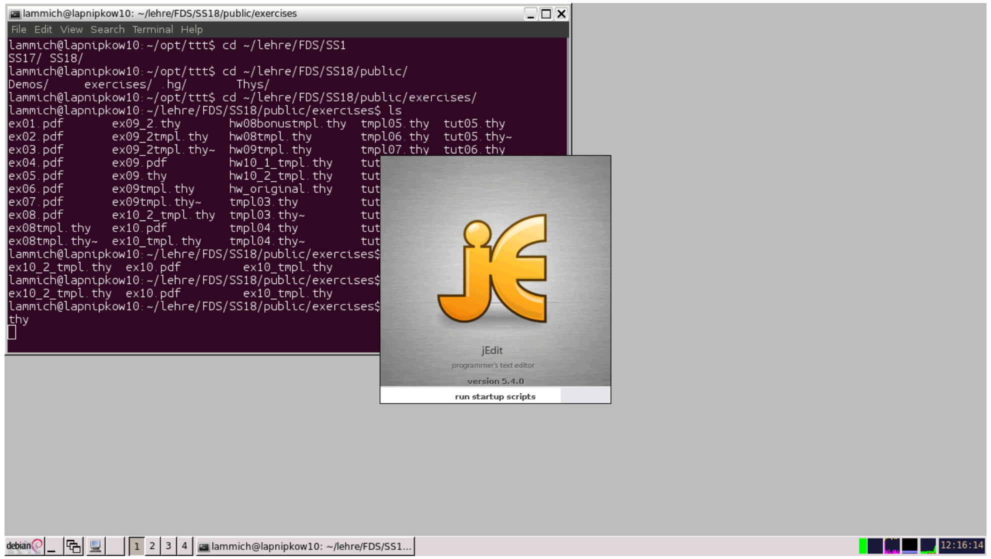
Title: Lammich: FDS Tutorial (15.06.2018)

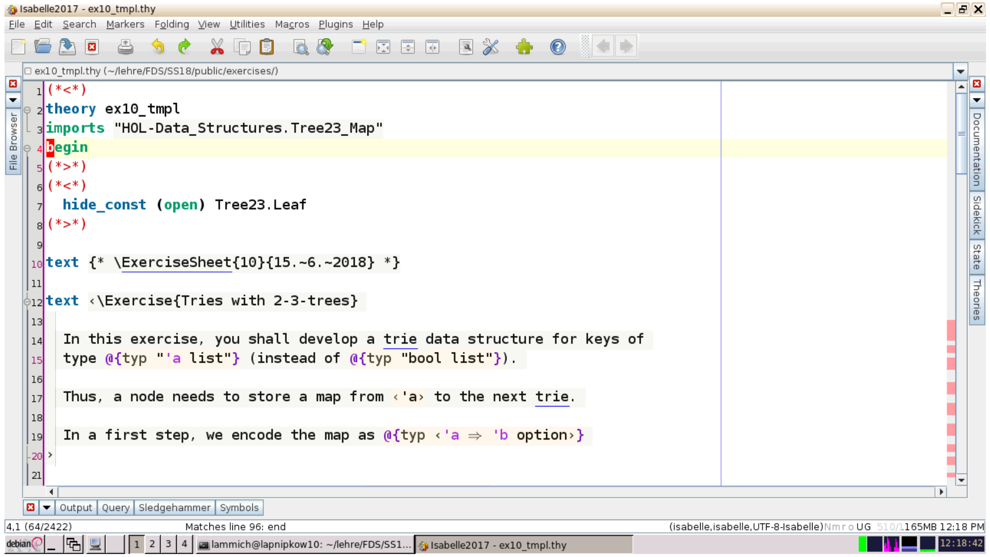
Date: Fri Jun 15 12:15:33 CEST 2018

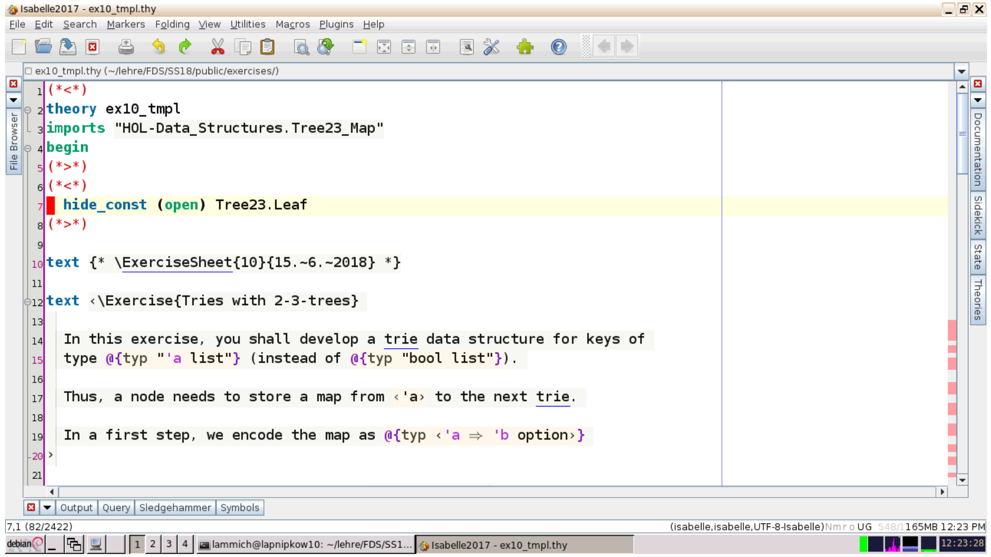
Duration: 100:15 min

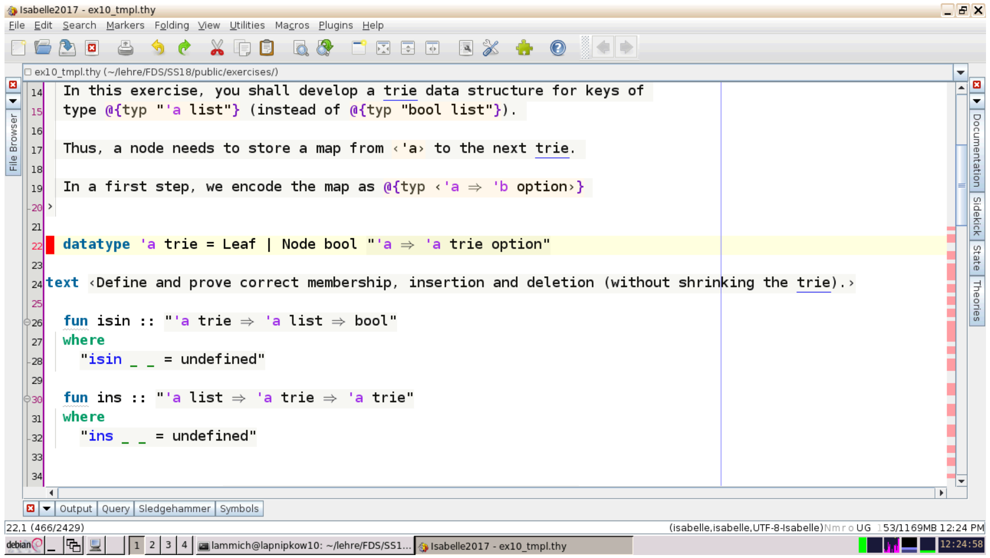
Pages: 90

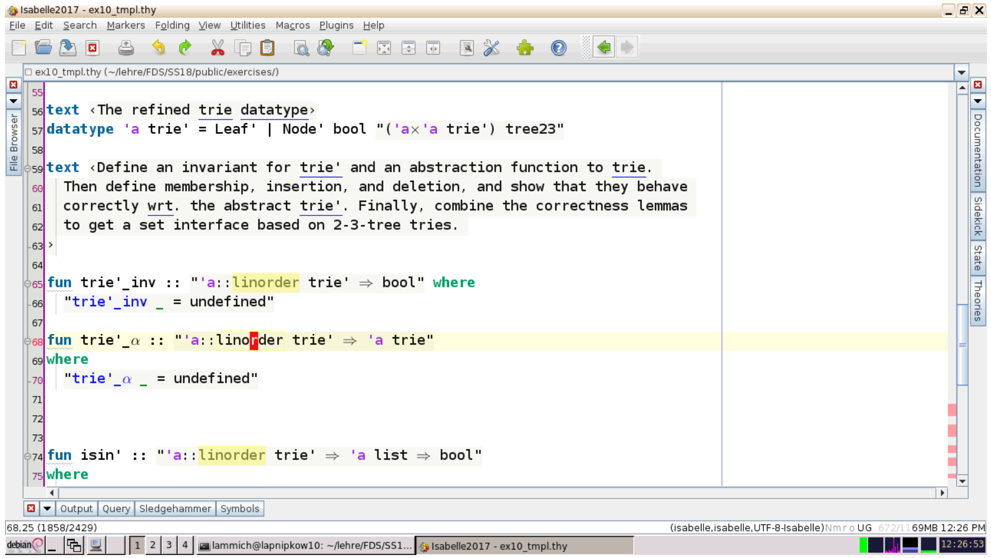


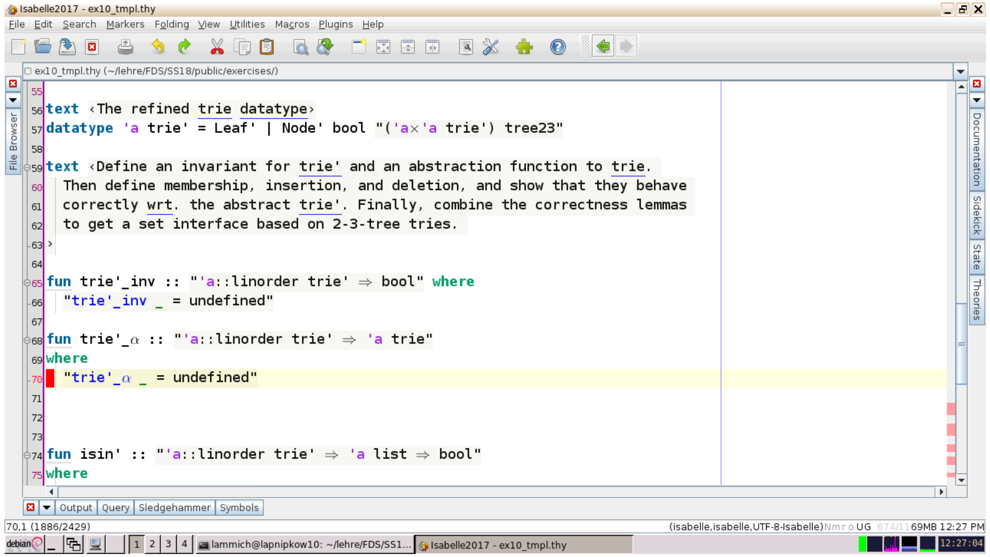
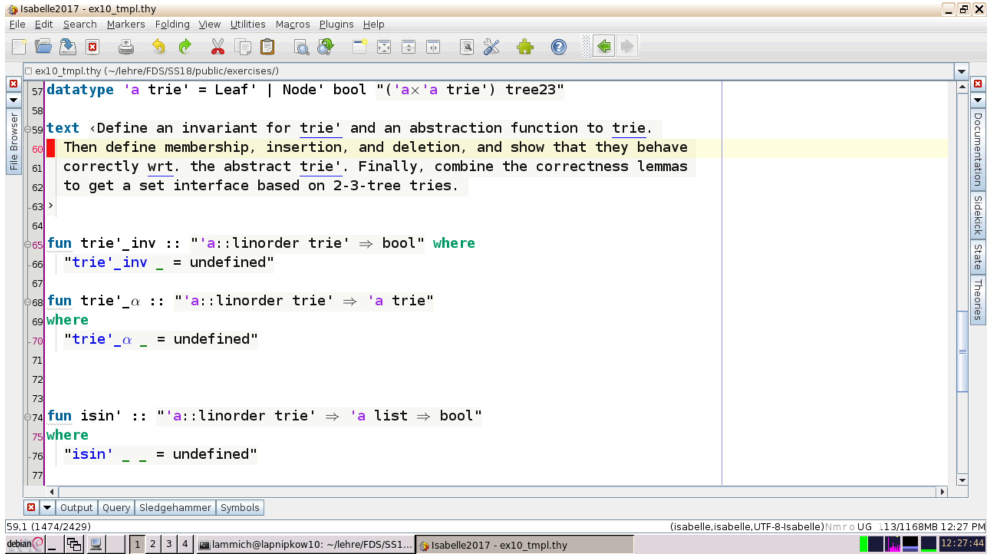
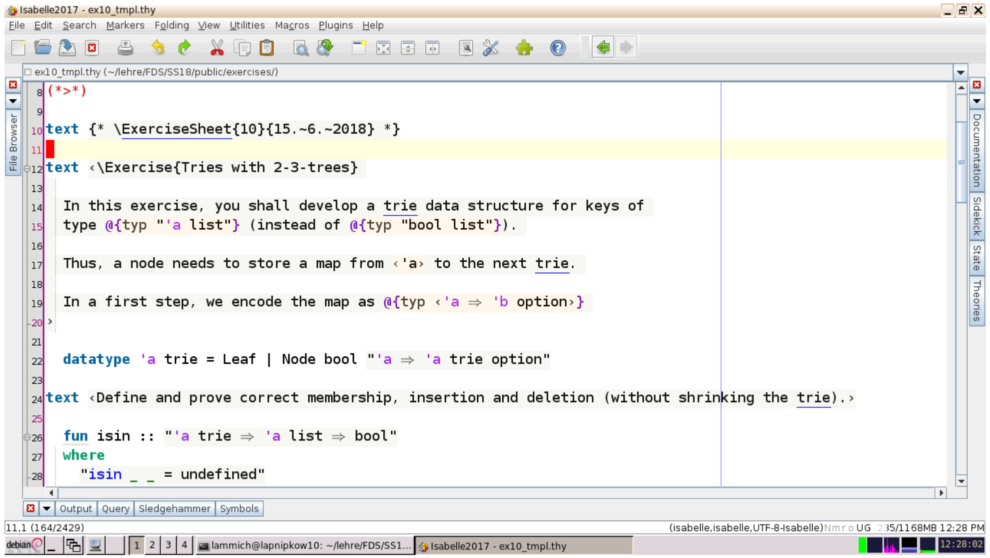
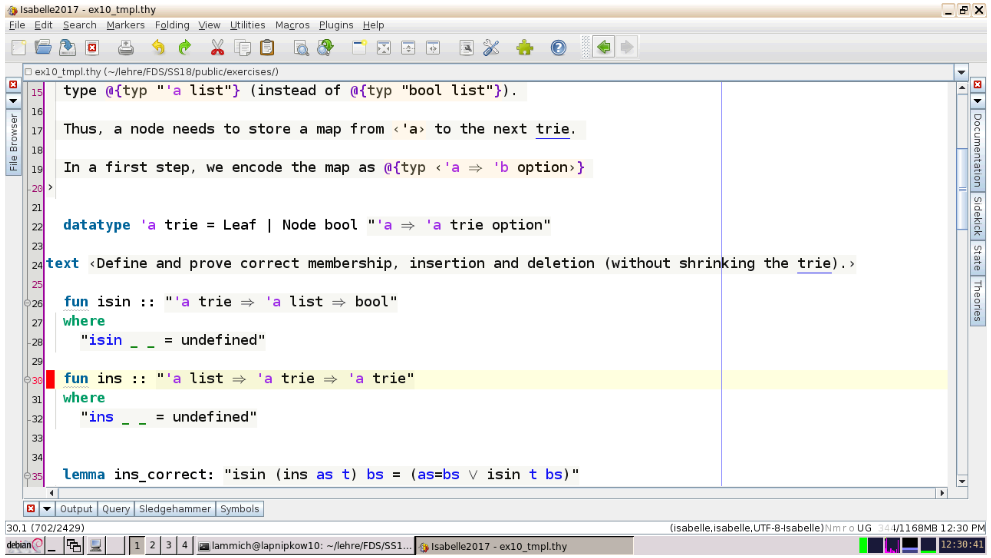



Isabelle2017 - ex10_tmpl.thy
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
(*<*)
theory ex10_tmpl
imports "HOL-Data_Structures.Tree23_Map"
begin
(*>*)
(*<*)
hide_const (open) Tree23.Leaf
(*>*)
text {* \ExerciseSheet{10}{15.~6.~2018} *}
text ‹\Exercise{Tries with 2-3-trees}
In this exercise, you shall develop a trie data structure for keys of
type @{typ "'a list"} (instead of @{typ "bool list"}).
Thus, a node needs to store a map from ‹'a› to the next trie.
In a first step, we encode the map as @{typ ‹'a ⇒ 'b option›}
›
Output Query Sledgehammer Symbols
4,1 (64/2422) Matches line 96: end (isabelle,isabelle,UTF-8-Isabelle) 165MB 12:18 PM


Isabelle2017 - ex10_tmpl.thy
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
(*<*)
theory ex10_tmpl
imports "HOL-Data_Structures.Tree23_Map"
begin
(*>*)
(*<*)
hide_const (open) Tree23.Leaf
(*>*)
text {* \ExerciseSheet{10}{15.~6.~2018} *}
text ‹\Exercise{Tries with 2-3-trees}
In this exercise, you shall develop a trie data structure for keys of
type @{typ "'a list"} (instead of @{typ "bool list"}).
Thus, a node needs to store a map from ‹'a› to the next trie.
In a first step, we encode the map as @{typ ‹'a ⇒ 'b option›}
›
Output Query Sledgehammer Symbols
7,1 (82/2422) (isabelle,isabelle,UTF-8-Isabelle) 165MB 12:23 PM


Isabelle2017 - ex10_tmpl.thy
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
In this exercise, you shall develop a trie data structure for keys of
type @{typ "'a list"} (instead of @{typ "bool list"}).
Thus, a node needs to store a map from ‹'a› to the next trie.
In a first step, we encode the map as @{typ ‹'a ⇒ 'b option›}
›
datatype 'a trie = Leaf | Node bool "'a ⇒ 'a trie option"
text ‹Define and prove correct membership, insertion and deletion (without shrinking the trie).›
fun isin :: "'a trie ⇒ 'a list ⇒ bool"
where
"isin _ _ = undefined"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins _ _ = undefined"
Output Query Sledgehammer Symbols
22,1 (466/2429) (isabelle,isabelle,UTF-8-Isabelle) 53/1169MB 12:24 PM


Isabelle2017 - ex10_tmpl.thy
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
text ‹The refined trie datatype›
datatype 'a trie' = Leaf' | Node' bool "('a×'a trie') tree23"
text ‹Define an invariant for trie' and an abstraction function to trie.
Then define membership, insertion, and deletion, and show that they behave
correctly wrt. the abstract trie'. Finally, combine the correctness lemmas
to get a set interface based on 2-3-tree tries.
›
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv _ = undefined"
fun trie'_α :: "'a::linorder trie' ⇒ 'a trie"
where
"trie'_α _ = undefined"
fun isin' :: "'a::linorder trie' ⇒ 'a list ⇒ bool"
where
Output Query Sledgehammer Symbols
68,25 (1858/2429) (isabelle,isabelle,UTF-8-Isabelle) 69MB 12:26 PM


Isabelle2017 - ex10_tmpl.thy
text ‹The refined trie datatype›
datatype 'a trie' = Leaf' | Node' bool "('a×'a trie') tree23"
text ‹Define an invariant for trie' and an abstraction function to trie.
Then define membership, insertion, and deletion, and show that they behave
correctly wrt. the abstract trie'. Finally, combine the correctness lemmas
to get a set interface based on 2-3-tree tries.
›
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv _ = undefined"
fun trie'_α :: "'a::linorder trie' ⇒ 'a trie"
where
"trie'_α _ = undefined"
fun isin' :: "'a::linorder trie' ⇒ 'a list ⇒ bool"
where
Output Query Sledgehammer Symbols
70,1 (1886/2429)

Isabelle2017 - ex10_tmpl.thy
datatype 'a trie' = Leaf' | Node' bool "('a×'a trie') tree23"
text ‹Define an invariant for trie' and an abstraction function to trie.
Then define membership, insertion, and deletion, and show that they behave
correctly wrt. the abstract trie'. Finally, combine the correctness lemmas
to get a set interface based on 2-3-tree tries.
›
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv _ = undefined"
fun trie'_α :: "'a::linorder trie' ⇒ 'a trie"
where
"trie'_α _ = undefined"
fun isin' :: "'a::linorder trie' ⇒ 'a list ⇒ bool"
where
"isin' _ _ = undefined"
Output Query Sledgehammer Symbols
59,1 (1474/2429)

Isabelle2017 - ex10_tmpl.thy
(*>*)
text {* \ExerciseSheet{10}{15.~6.~2018} *}
text ‹\Exercise{Tries with 2-3-trees}
In this exercise, you shall develop a trie data structure for keys of
type @{typ "'a list"} (instead of @{typ "bool list"}).
Thus, a node needs to store a map from ‹'a› to the next trie.
In a first step, we encode the map as @{typ ‹'a ⇒ 'b option›}
›
datatype 'a trie = Leaf | Node bool "'a ⇒ 'a trie option"
text ‹Define and prove correct membership, insertion and deletion (without shrinking the trie).›
fun isin :: "'a trie ⇒ 'a list ⇒ bool"
where
"isin _ _ = undefined"
Output Query Sledgehammer Symbols
11,1 (164/2429)

Isabelle2017 - ex10_tmpl.thy
type @{typ "'a list"} (instead of @{typ "bool list"}).
Thus, a node needs to store a map from ‹'a› to the next trie.
In a first step, we encode the map as @{typ ‹'a ⇒ 'b option›}
›
datatype 'a trie = Leaf | Node bool "'a ⇒ 'a trie option"
text ‹Define and prove correct membership, insertion and deletion (without shrinking the trie).›
fun isin :: "'a trie ⇒ 'a list ⇒ bool"
where
"isin _ _ = undefined"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins _ _ = undefined"
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
Output Query Sledgehammer Symbols
30,1 (702/2429)

Isabelle2017 - ex10_tmpl.thy

ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)

```
type @{typ "'a list"} (instead of @{typ "bool list"}).

Thus, a node needs to store a map from ‹'a› to the next trie.

In a first step, we encode the map as @{typ ‹'a ⇒ 'b option›}
›

datatype 'a trie = Leaf | Node bool "'a ⇒ 'a trie option"

text ‹Define and prove correct membership, insertion and deletion (without shrinking the trie).›

fun isin :: "'a trie ⇒ 'a list ⇒ bool"
where
  "isin _ _ = undefined"

fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
  "ins _ _ = undefined"

lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
```

Isabelle2017 - ex10_tmpl.thy (modified)

ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)

```
datatype 'a trie = Leaf | Node bool "'a ⇒ 'a trie option"

text ‹Define and prove correct membership, insertion and deletion (without shrinking the trie).›

fun isin :: "'a trie ⇒ 'a list ⇒ bool"
where
  "isin Leaf _ = False"
| "isin (Node b m) [] ⟷ b"
| "isin (Node b m) (k#ks) ⟷ (case m k of None ⇒ False | Some t ⇒ isin t ks)"



fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
  "ins _ _ = undefined"


lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
  oops
```

Isabelle2017 - ex10_tmpl.thy

ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)

```
| "isin (Node b m) [] ⟷ b"
| "isin (Node b m) (k#ks) ⟷ (case m k of None ⇒ False | Some t ⇒ isin t ks)"


term "m(k↦t)"
term "m(k:=Some t)"
term "fun_upd m k (Some t)"

fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
  "ins _ _ = undefined"


lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
```

Proof state · Auto update · Update · Search: · 100%

```
"m(k ↦ t)"
  :: "'a ⇒ 'b option"
```

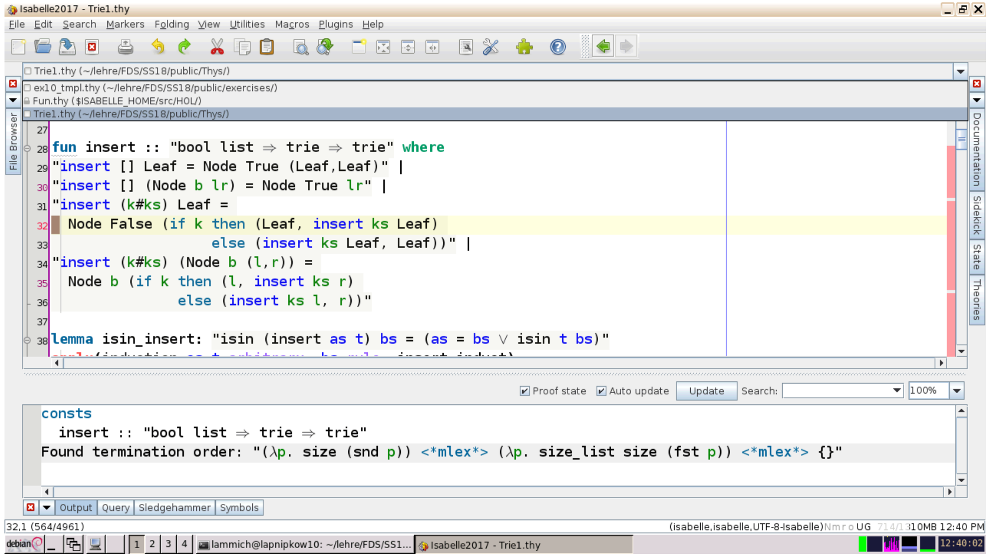
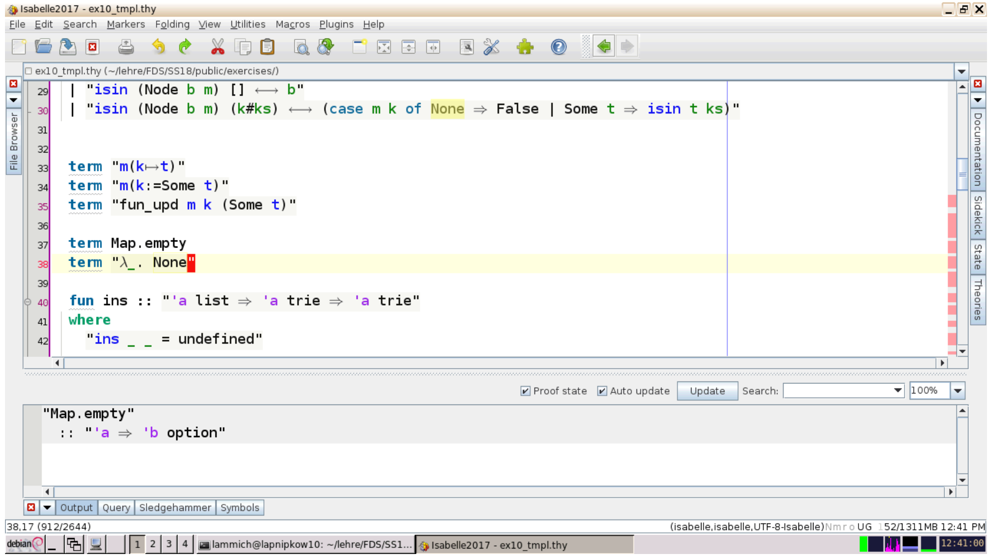
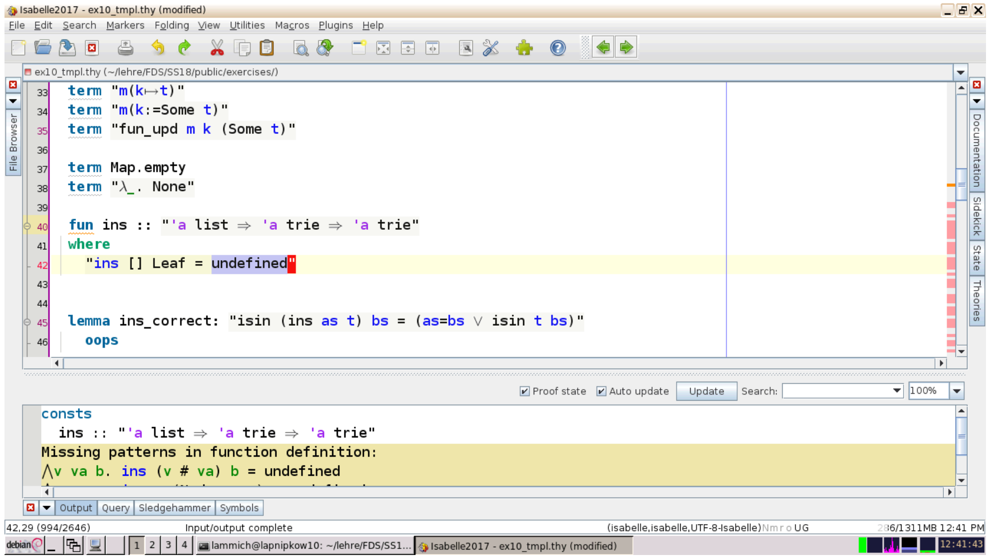
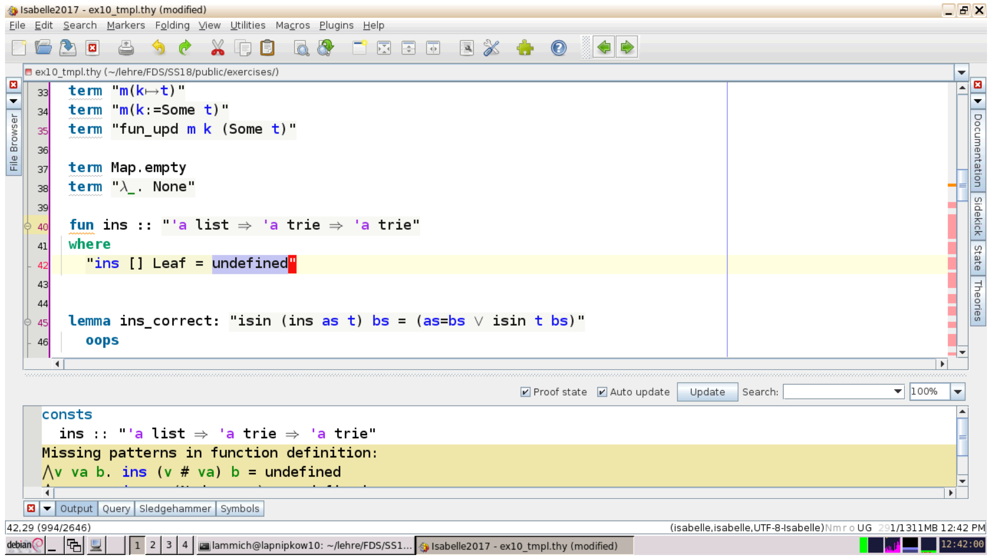
Isabelle2017 - Trie1.thy

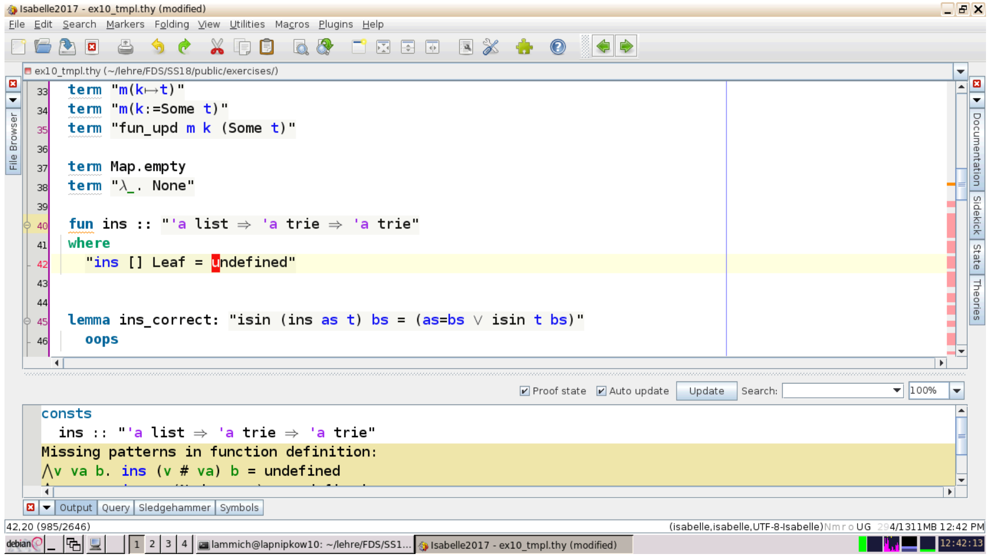
Trie1.thy (~/lehre/FDS/SS18/public/Thys/)

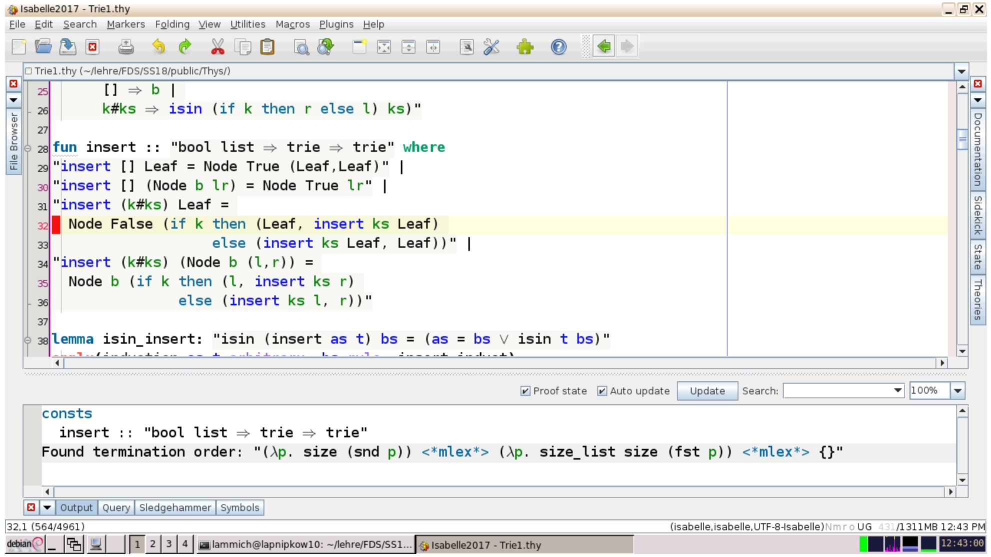
```
    [] ⇒ b |
    k#ks ⇒ isin (if k then r else l) ks)"

fun insert :: "bool list ⇒ trie ⇒ trie" where
"insert [] Leaf = Node True (Leaf,Leaf)" |
"insert [] (Node b lr) = Node True lr" |
"insert (k#ks) Leaf =
  Node False (if k then (Leaf, insert ks Leaf)
                   else (insert ks Leaf, Leaf))" |
"insert (k#ks) (Node b (l,r)) =
  Node b (if k then (l, insert ks r)
               else (insert ks l, r))"

lemma isin_insert: "isin (insert as t) bs = (as = bs ∨ isin t bs)"
```

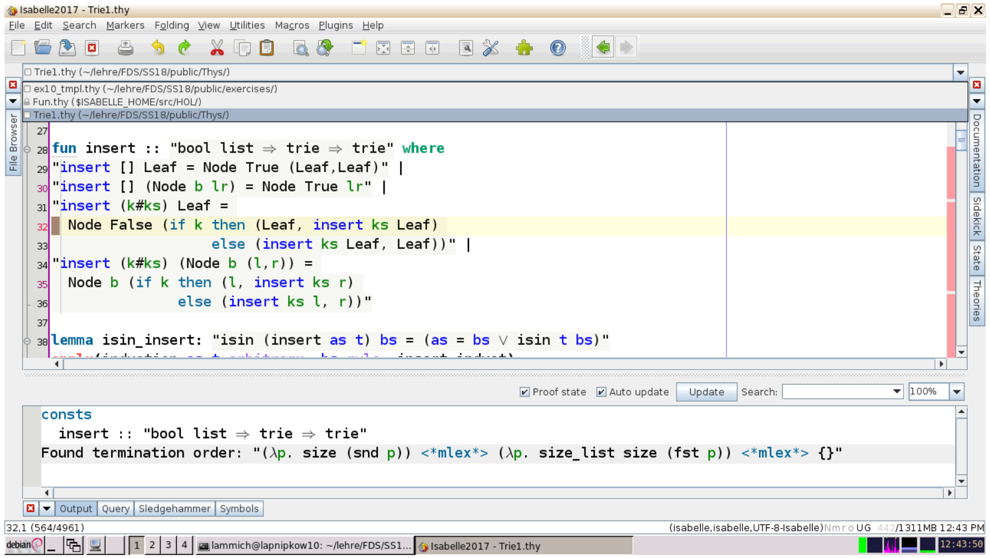
Proof state · Auto update · Update · Search: · 100%

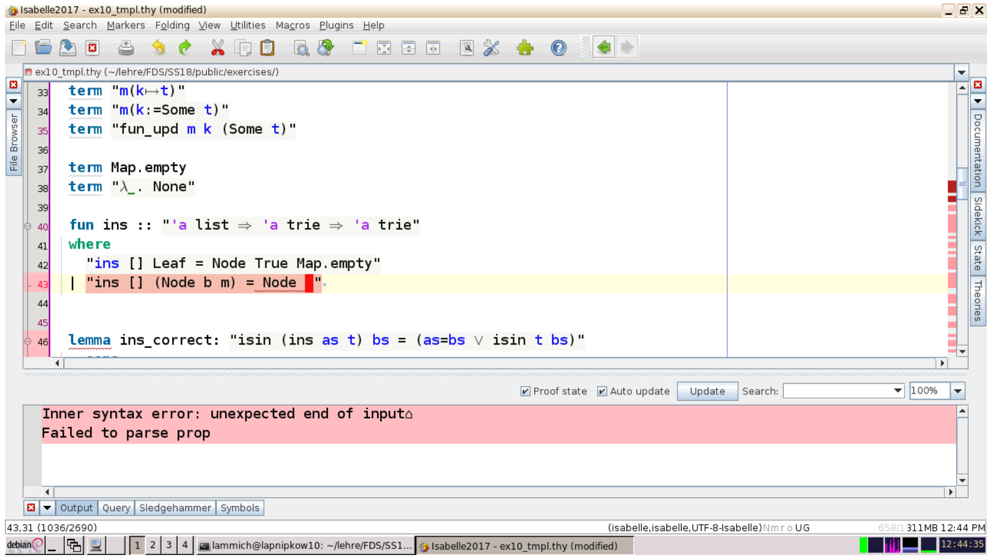
```
consts
  insert :: "bool list ⇒ trie ⇒ trie"
Found termination order: "(λp. size (snd p)) <*mlex*> (λp. size_list size (fst p)) <*mlex*> {}"
```

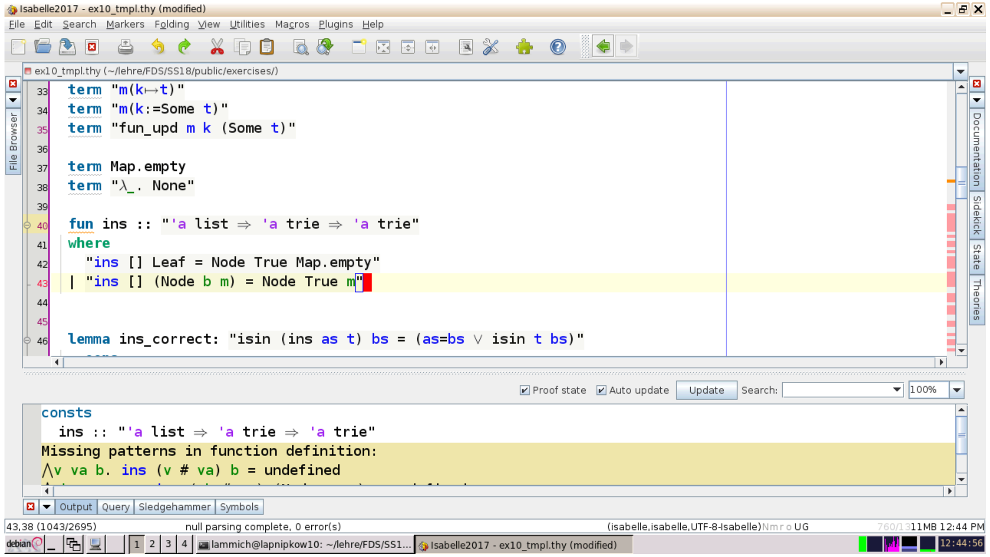

Isabelle2017 - Trie1.thy
fun insert :: "bool list ⇒ trie ⇒ trie" where
"insert [] Leaf = Node True (Leaf,Leaf)" |
"insert [] (Node b lr) = Node True lr" |
"insert (k#ks) Leaf =
Node False (if k then (Leaf, insert ks Leaf)
else (insert ks Leaf, Leaf))" |
"insert (k#ks) (Node b (l,r)) =
Node b (if k then (l, insert ks r)
else (insert ks l, r))"
lemma isin_insert: "isin (insert as t) bs = (as = bs ∨ isin t bs)"
consts
insert :: "bool list ⇒ trie ⇒ trie"
Found termination order: "(λp. size (snd p)) <*mlex*> (λp. size_list size (fst p)) <*mlex*> {}"

Isabelle2017 - ex10_tmpl.thy
| "isin (Node b m) [] ⟷ b"
| "isin (Node b m) (k#ks) ⟷ (case m k of None ⇒ False | Some t ⇒ isin t ks)"
term "m(k↦t)"
term "m(k:=Some t)"
term "fun_upd m k (Some t)"
term Map.empty
term "λ_. None"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins _ _ = undefined"
"Map.empty"
:: "'a ⇒ 'b option"

Isabelle2017 - ex10_tmpl.thy (modified)
term "m(k↦t)"
term "m(k:=Some t)"
term "fun_upd m k (Some t)"
term Map.empty
term "λ_. None"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins [] Leaf = undefined"
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
oops
consts
ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
Missing patterns in function definition:
⋀v va b. ins (v # va) b = undefined

Isabelle2017 - ex10_tmpl.thy (modified)
term "m(k↦t)"
term "m(k:=Some t)"
term "fun_upd m k (Some t)"
term Map.empty
term "λ_. None"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins [] Leaf = undefined"
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
oops
consts
ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
Missing patterns in function definition:
⋀v va b. ins (v # va) b = undefined

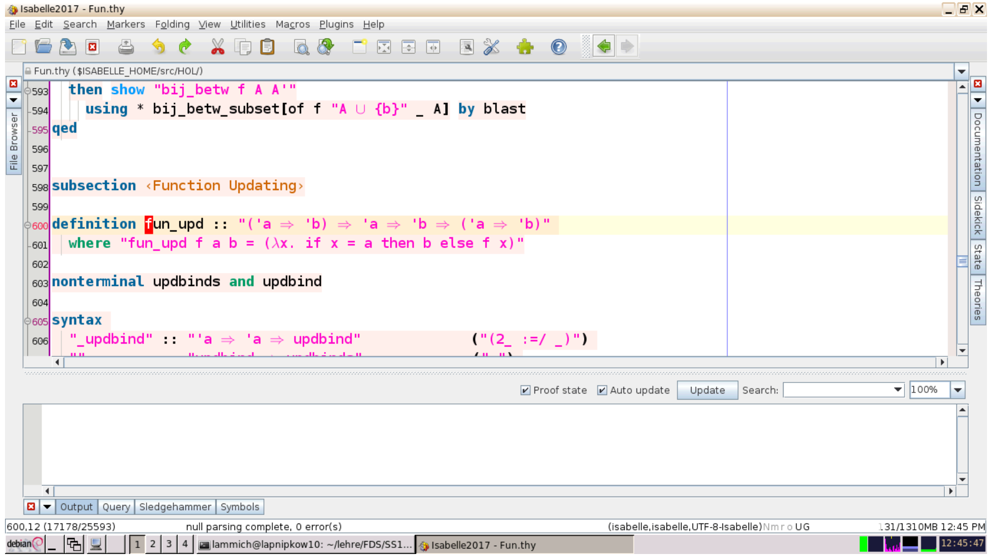

Isabelle2017 - ex10_tmpl.thy (modified)
term "m(k↦t)"
term "m(k:=Some t)"
term "fun_upd m k (Some t)"
term Map.empty
term "λ_. None"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins [] Leaf = undefined"
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
oops
consts
ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
Missing patterns in function definition:
⋀v va b. ins (v # va) b = undefined

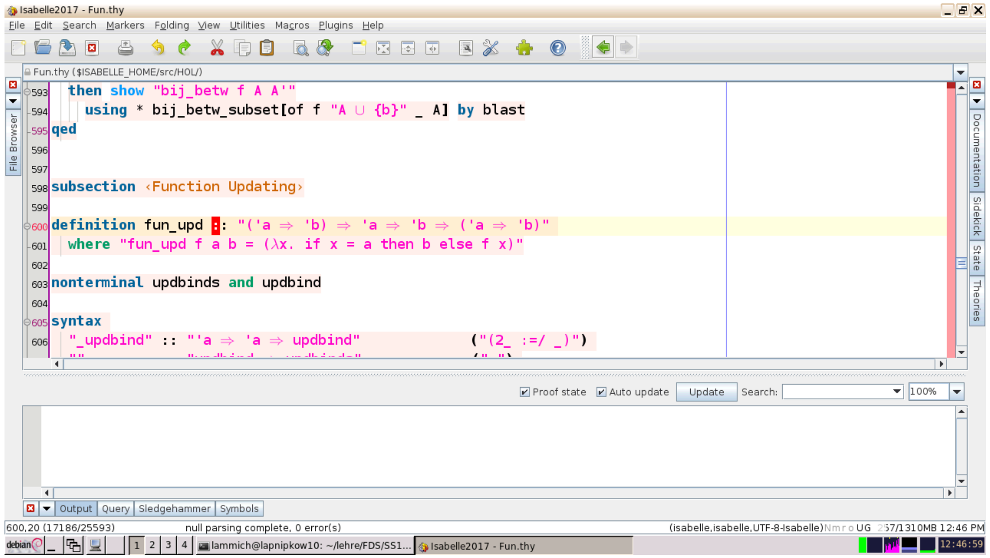

Isabelle2017 - Trie1.thy
[] ⇒ b |
k#ks ⇒ isin (if k then r else l) ks)"
fun insert :: "bool list ⇒ trie ⇒ trie" where
"insert [] Leaf = Node True (Leaf,Leaf)" |
"insert [] (Node b lr) = Node True lr" |
"insert (k#ks) Leaf =
Node False (if k then (Leaf, insert ks Leaf)
else (insert ks Leaf, Leaf))" |
"insert (k#ks) (Node b (l,r)) =
Node b (if k then (l, insert ks r)
else (insert ks l, r))"
lemma isin_insert: "isin (insert as t) bs = (as = bs ∨ isin t bs)"
consts
insert :: "bool list ⇒ trie ⇒ trie"
Found termination order: "(λp. size (snd p)) <*mlex*> (λp. size_list size (fst p)) <*mlex*> {}"

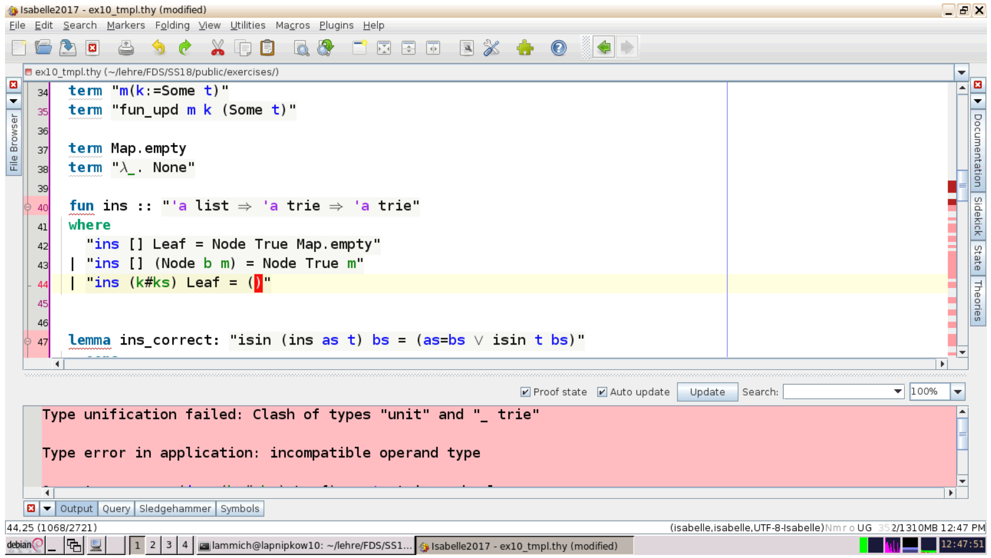

Isabelle2017 - Trie1.thy
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
Fun.thy ($ISABELLE_HOME/src/HOL/)
Trie1.thy (~/lehre/FDS/SS18/public/Thys/)
fun insert :: "bool list ⇒ trie ⇒ trie" where
"insert [] Leaf = Node True (Leaf,Leaf)" |
"insert [] (Node b lr) = Node True lr" |
"insert (k#ks) Leaf =
Node False (if k then (Leaf, insert ks Leaf)
else (insert ks Leaf, Leaf))" |
"insert (k#ks) (Node b (l,r)) =
Node b (if k then (l, insert ks r)
else (insert ks l, r))"
lemma isin_insert: "isin (insert as t) bs = (as = bs ∨ isin t bs)"
consts
insert :: "bool list ⇒ trie ⇒ trie"
Found termination order: "(λp. size (snd p)) <*mlex*> (λp. size_list size (fst p)) <*mlex*> {}"

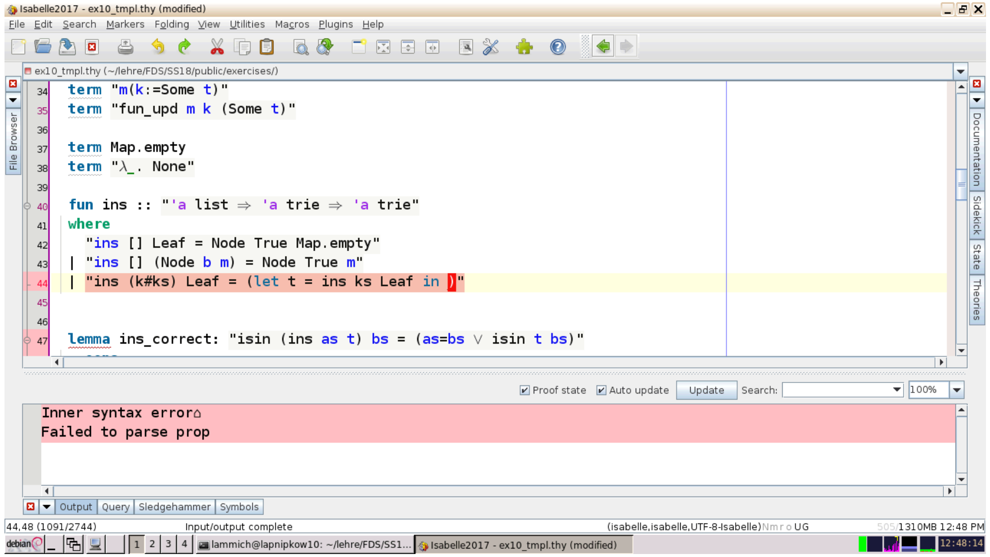
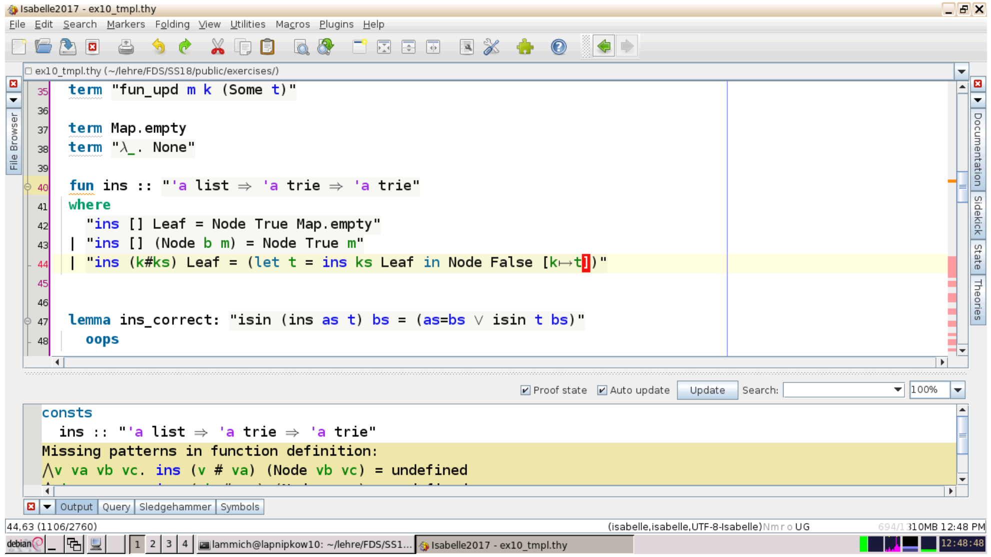
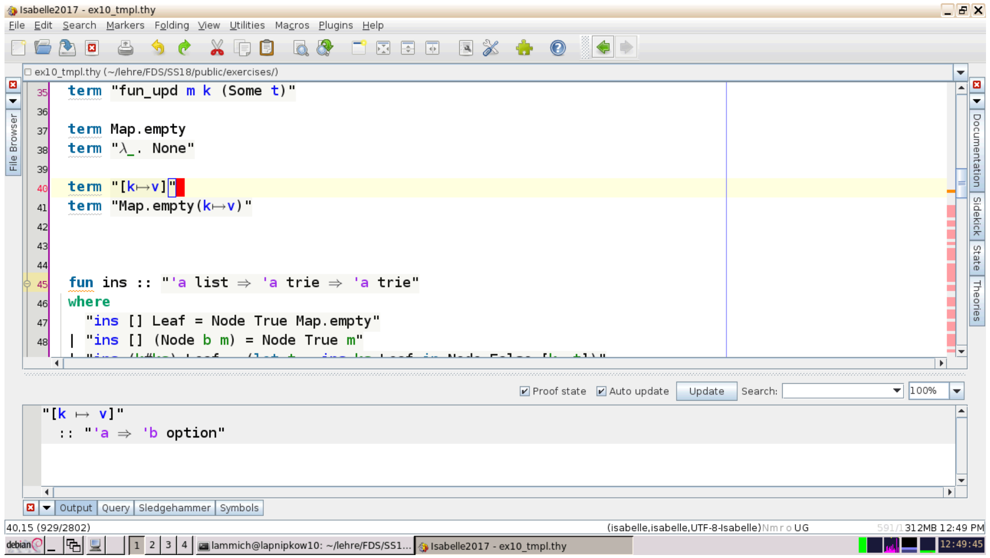
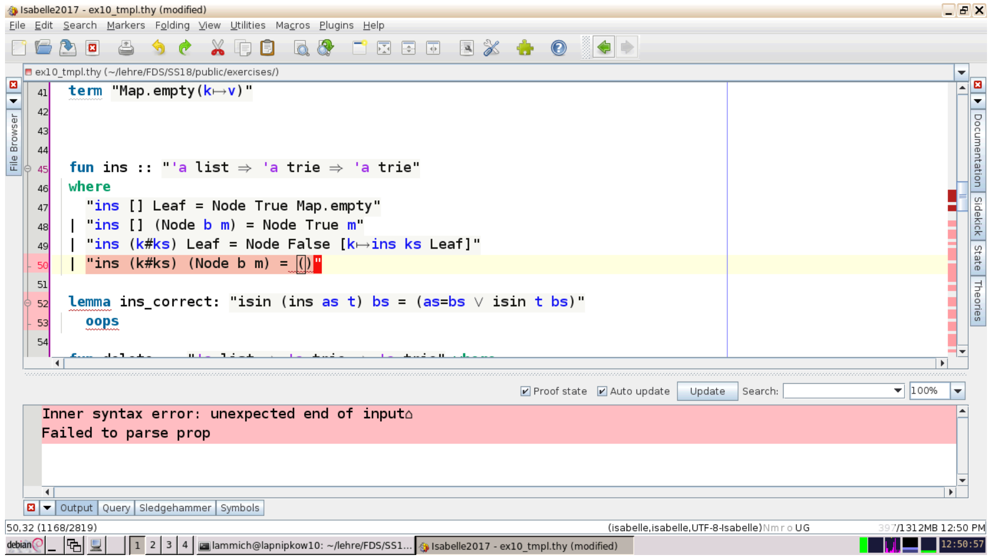

Isabelle2017 - ex10_tmpl.thy (modified)
term "m(k↦t)"
term "m(k:=Some t)"
term "fun_upd m k (Some t)"
term Map.empty
term "λ_. None"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins [] Leaf = Node True Map.empty"
| "ins [] (Node b m) = Node "
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
Inner syntax error: unexpected end of input
Failed to parse prop

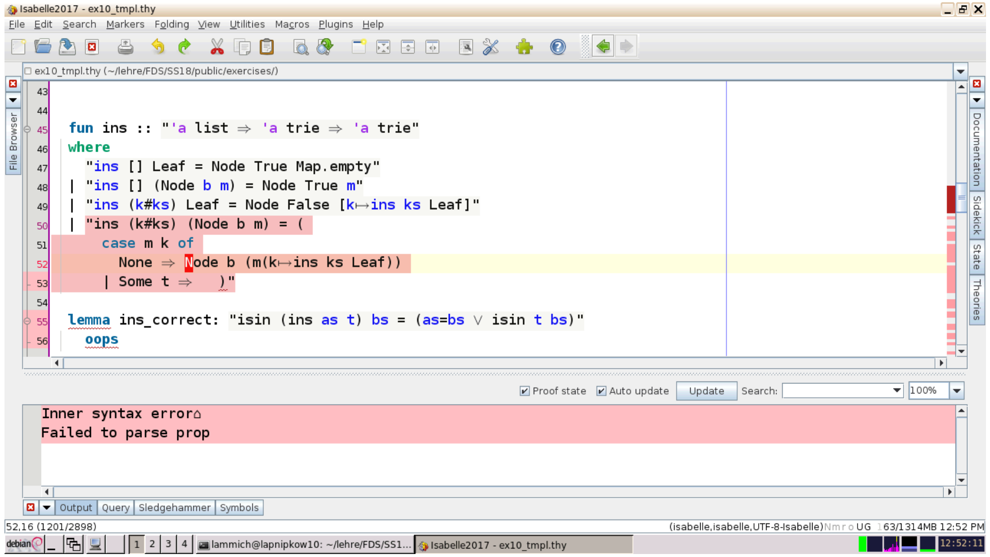
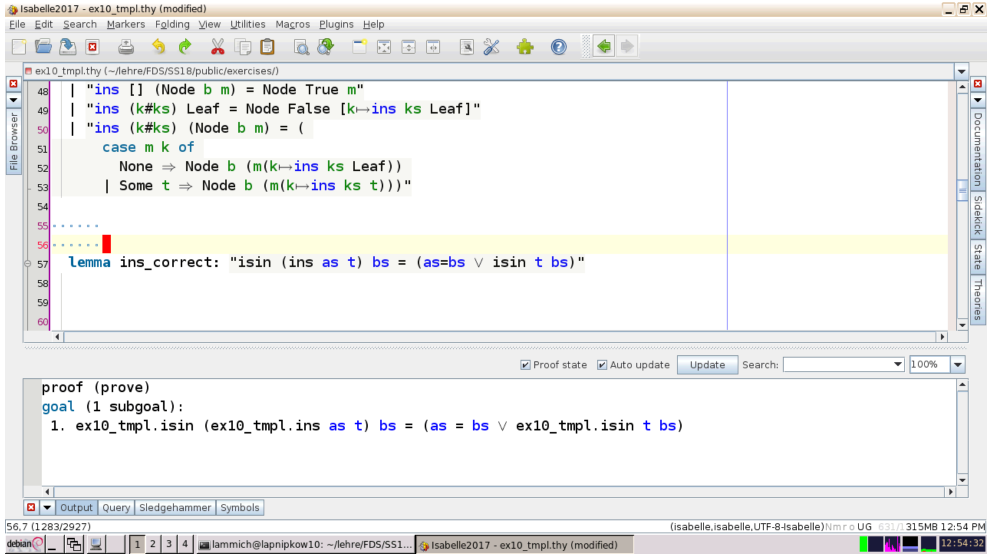
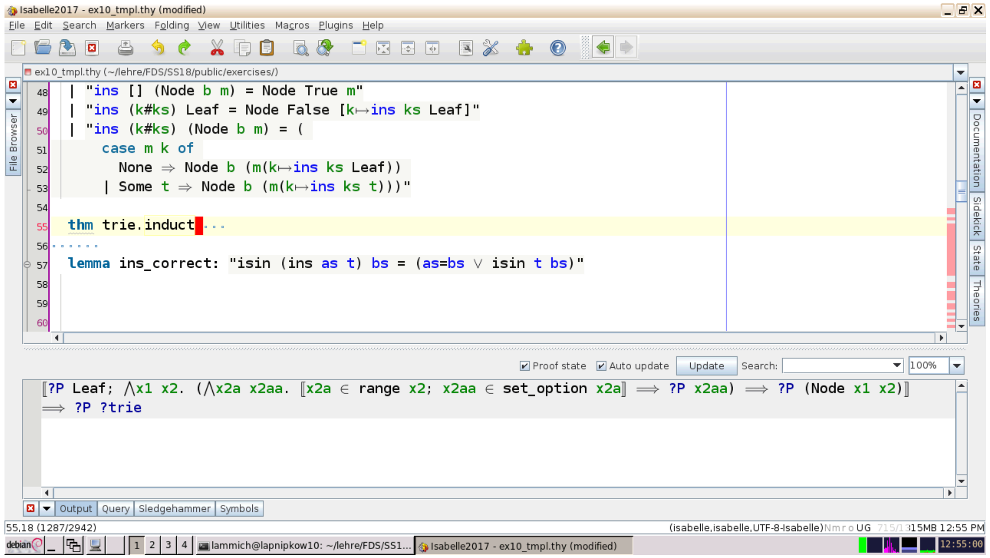
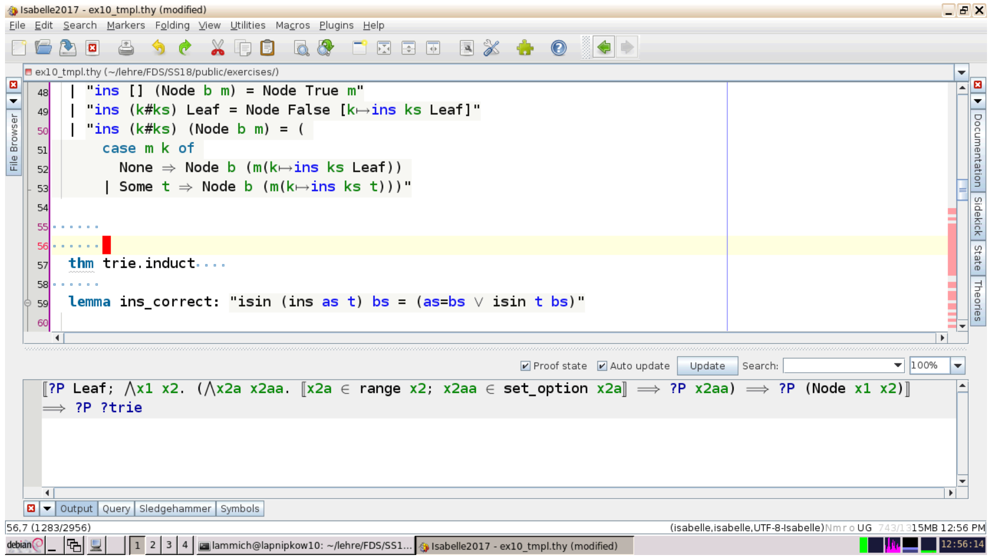

Isabelle2017 - ex10_tmpl.thy (modified)
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
term "m(k↦t)"
term "m(k:=Some t)"
term "fun_upd m k (Some t)"
term Map.empty
term "λ_. None"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins [] Leaf = Node True Map.empty"
| "ins [] (Node b m) = Node True m"
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
consts
ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
Missing patterns in function definition:
⋀v va b. ins (v # va) b = undefined
43,38 (1043/2695)
null parsing complete, 0 error(s)
12:44 PM

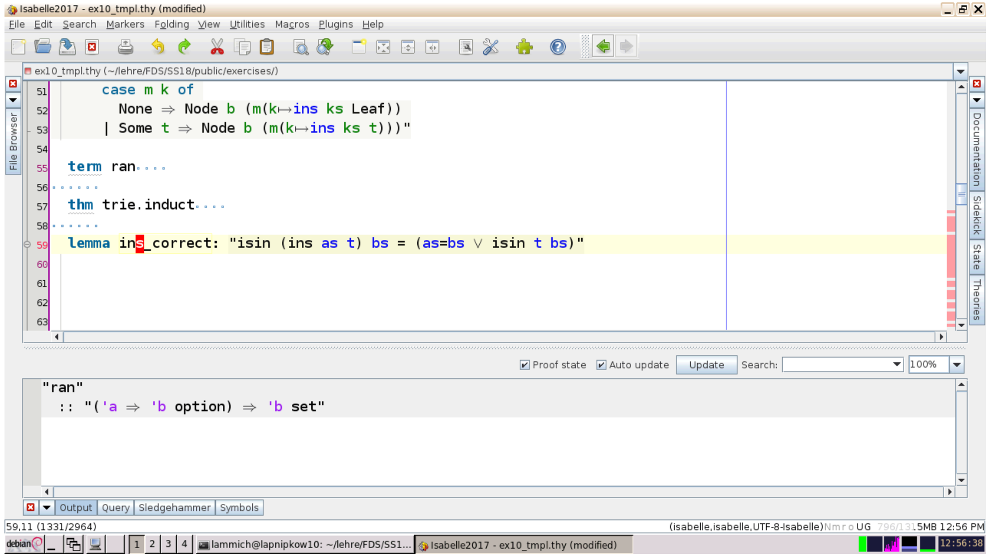

Isabelle2017 - Fun.thy
Fun.thy ($ISABELLE_HOME/src/HOL/)
then show "bij_betw f A A'"
using * bij_betw_subset[of f "A ∪ {b}" _ A] by blast
qed
subsection ‹Function Updating›
definition fun_upd :: "('a ⇒ 'b) ⇒ 'a ⇒ 'b ⇒ ('a ⇒ 'b)"
where "fun_upd f a b = (λx. if x = a then b else f x)"
nonterminal updbinds and updbind
syntax
"_updbind" :: "'a ⇒ 'a ⇒ updbind" ("(2_ :=/ _)")
600,12 (17178/25593)
null parsing complete, 0 error(s)
12:45 PM

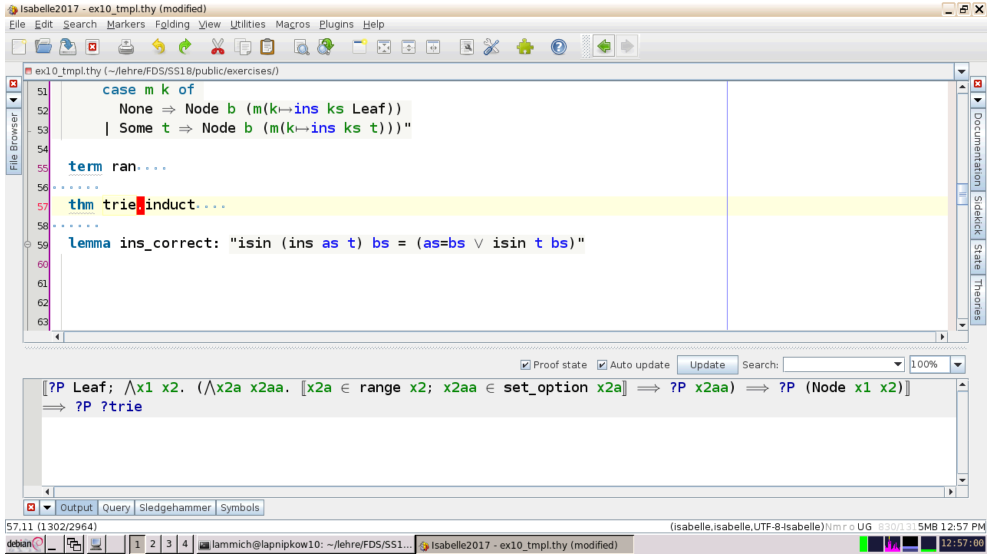

Isabelle2017 - Fun.thy
Fun.thy ($ISABELLE_HOME/src/HOL/)
then show "bij_betw f A A'"
using * bij_betw_subset[of f "A ∪ {b}" _ A] by blast
qed
subsection ‹Function Updating›
definition fun_upd :: "('a ⇒ 'b) ⇒ 'a ⇒ 'b ⇒ ('a ⇒ 'b)"
where "fun_upd f a b = (λx. if x = a then b else f x)"
nonterminal updbinds and updbind
syntax
"_updbind" :: "'a ⇒ 'a ⇒ updbind" ("(2_ :=/ _)")
600,20 (17186/25593)
null parsing complete, 0 error(s)
12:46 PM

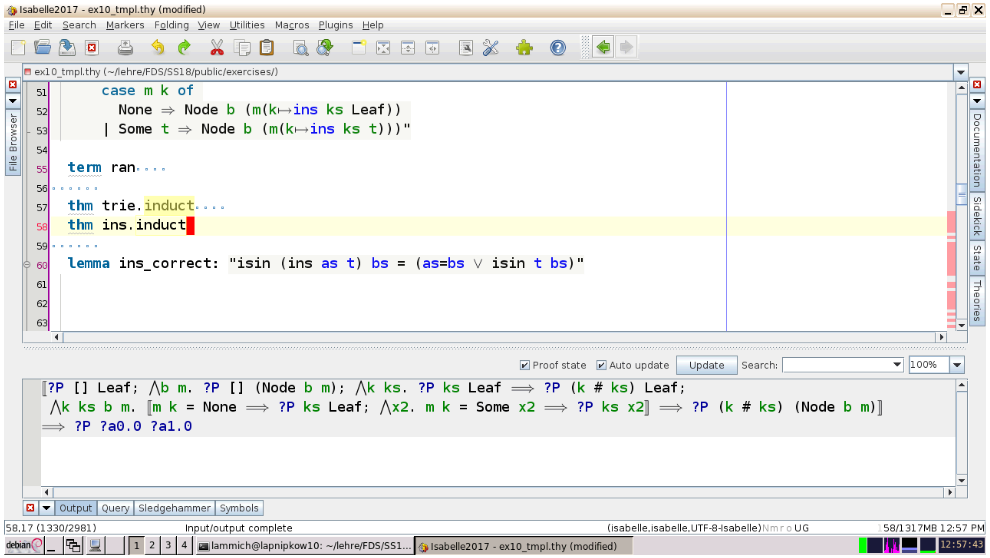

Isabelle2017 - ex10_tmpl.thy (modified)
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
term "m(k:=Some t)"
term "fun_upd m k (Some t)"
term Map.empty
term "λ_. None"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins [] Leaf = Node True Map.empty"
| "ins [] (Node b m) = Node True m"
| "ins (k#ks) Leaf = ()"
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
Type unification failed: Clash of types "unit" and "_ trie"
Type error in application: incompatible operand type
44,25 (1068/2721)
12:47 PM

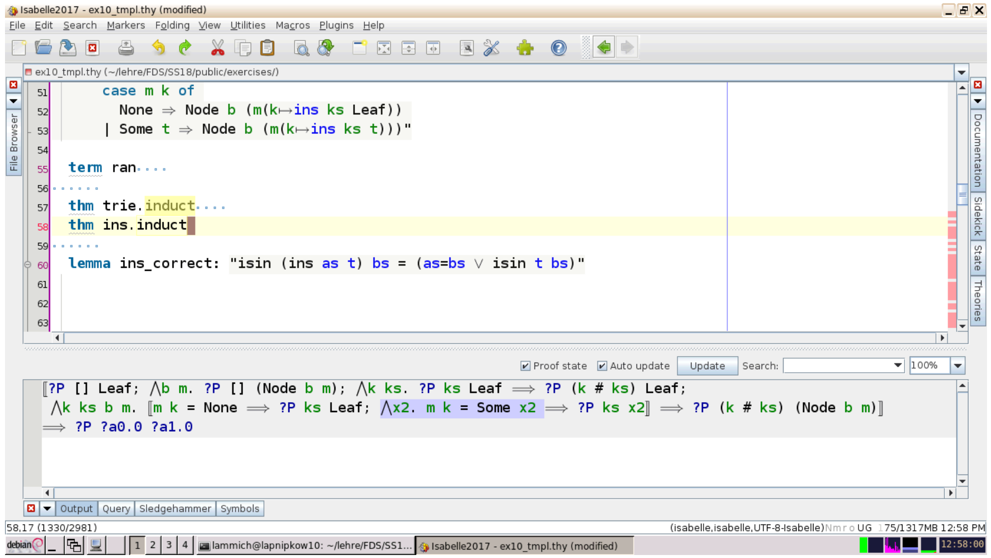

Isabelle2017 - ex10_tmpl.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
term "m(k:=Some t)"
term "fun_upd m k (Some t)"
term Map.empty
term "λ_. None"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins [] Leaf = Node True Map.empty"
| "ins [] (Node b m) = Node True m"
| "ins (k#ks) Leaf = (let t = ins ks Leaf in)"
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
Proof state Auto update Update Search: 100%
Inner syntax error
Failed to parse prop
Output Query Sledgehammer Symbols
44,48 (1091/2744) Input/output complete (isabelle,isabelle,UTF-8-Isabelle) Nmro UG 1310MB 12:48 PM

Isabelle2017 - ex10_tmpl.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
term "fun_upd m k (Some t)"
term Map.empty
term "λ_. None"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins [] Leaf = Node True Map.empty"
| "ins [] (Node b m) = Node True m"
| "ins (k#ks) Leaf = (let t = ins ks Leaf in Node False [k↦t])"
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
oops
Proof state Auto update Update Search: 100%
consts
ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
Missing patterns in function definition:
⋀v va vb vc. ins (v # va) (Node vb vc) = undefined
Output Query Sledgehammer Symbols
44,63 (1106/2760) (isabelle,isabelle,UTF-8-Isabelle) Nmro UG 310MB 12:48 PM

Isabelle2017 - ex10_tmpl.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
term "fun_upd m k (Some t)"
term Map.empty
term "λ_. None"
term "[k↦v]"
term "Map.empty(k↦v)"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins [] Leaf = Node True Map.empty"
| "ins [] (Node b m) = Node True m"
Proof state Auto update Update Search: 100%
"[k ↦ v]"
:: "'a ⇒ 'b option"
Output Query Sledgehammer Symbols
40,15 (929/2802) (isabelle,isabelle,UTF-8-Isabelle) Nmro UG 312MB 12:49 PM

Isabelle2017 - ex10_tmpl.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
term "Map.empty(k↦v)"
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins [] Leaf = Node True Map.empty"
| "ins [] (Node b m) = Node True m"
| "ins (k#ks) Leaf = Node False [k↦ins ks Leaf]"
| "ins (k#ks) (Node b m) = ()"
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
oops
Proof state Auto update Update Search: 100%
Inner syntax error: unexpected end of input
Failed to parse prop
Output Query Sledgehammer Symbols
50,32 (1168/2819) (isabelle,isabelle,UTF-8-Isabelle) Nmro UG 312MB 12:50 PM

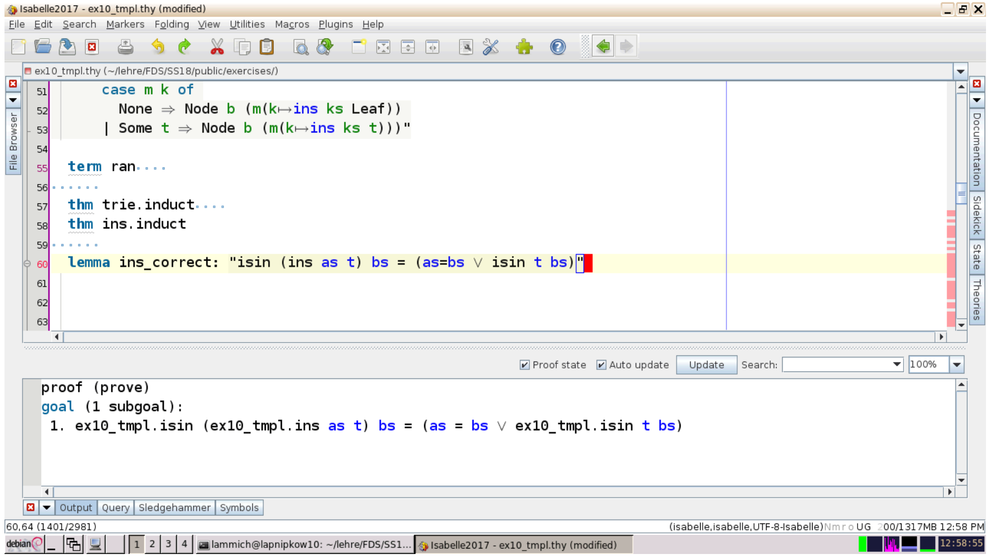

Isabelle2017 - ex10_tmpl.thy
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
fun ins :: "'a list ⇒ 'a trie ⇒ 'a trie"
where
"ins [] Leaf = Node True Map.empty"
| "ins [] (Node b m) = Node True m"
| "ins (k#ks) Leaf = Node False [k↦ins ks Leaf]"
| "ins (k#ks) (Node b m) = (
case m k of
None ⇒ Node b (m(k↦ins ks Leaf))
| Some t ⇒)"
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
oops
Inner syntax error
Failed to parse prop

Isabelle2017 - ex10_tmpl.thy (modified)
| "ins [] (Node b m) = Node True m"
| "ins (k#ks) Leaf = Node False [k↦ins ks Leaf]"
| "ins (k#ks) (Node b m) = (
case m k of
None ⇒ Node b (m(k↦ins ks Leaf))
| Some t ⇒ Node b (m(k↦ins ks t)))"
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
proof (prove)
goal (1 subgoal):
1. ex10_tmpl.isin (ex10_tmpl.ins as t) bs = (as = bs ∨ ex10_tmpl.isin t bs)

Isabelle2017 - ex10_tmpl.thy (modified)
| "ins [] (Node b m) = Node True m"
| "ins (k#ks) Leaf = Node False [k↦ins ks Leaf]"
| "ins (k#ks) (Node b m) = (
case m k of
None ⇒ Node b (m(k↦ins ks Leaf))
| Some t ⇒ Node b (m(k↦ins ks t)))"
thm trie.induct
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
⟦?P Leaf; ⋀x1 x2. (⋀x2a x2aa. ⟦x2a ∈ range x2; x2aa ∈ set_option x2a⟧ ⟹ ?P x2aa) ⟹ ?P (Node x1 x2)⟧
⟹ ?P ?trie

Isabelle2017 - ex10_tmpl.thy (modified)
| "ins [] (Node b m) = Node True m"
| "ins (k#ks) Leaf = Node False [k↦ins ks Leaf]"
| "ins (k#ks) (Node b m) = (
case m k of
None ⇒ Node b (m(k↦ins ks Leaf))
| Some t ⇒ Node b (m(k↦ins ks t)))"
thm trie.induct
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
⟦?P Leaf; ⋀x1 x2. (⋀x2a x2aa. ⟦x2a ∈ range x2; x2aa ∈ set_option x2a⟧ ⟹ ?P x2aa) ⟹ ?P (Node x1 x2)⟧
⟹ ?P ?trie

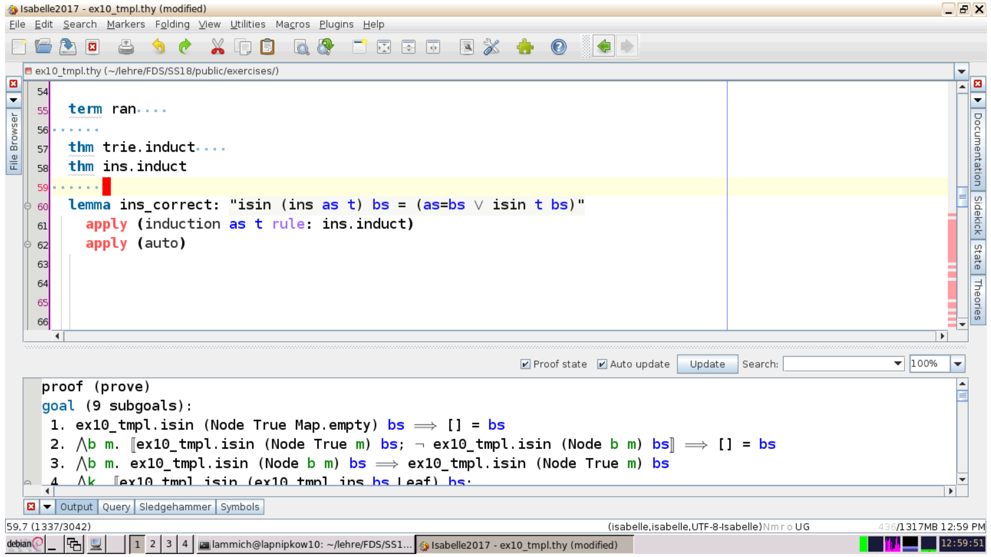

Isabelle2017 - ex10_tmpl.thy (modified)
case m k of
None ⇒ Node b (m(k↦ins ks Leaf))
| Some t ⇒ Node b (m(k↦ins ks t)))"
term ran
thm trie.induct
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
"ran"
:: "('a ⇒ 'b option) ⇒ 'b set"

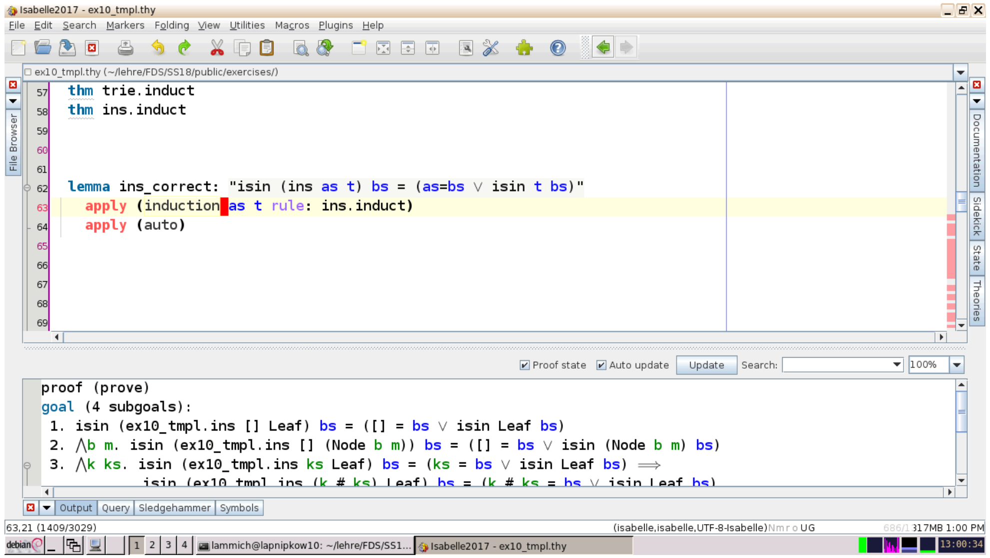

Isabelle2017 - ex10_tmpl.thy (modified)
case m k of
None ⇒ Node b (m(k↦ins ks Leaf))
| Some t ⇒ Node b (m(k↦ins ks t)))"
term ran
thm trie.induct
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
⟦?P Leaf; ⋀x1 x2. (⋀x2a x2aa. ⟦x2a ∈ range x2; x2aa ∈ set_option x2a⟧ ⟹ ?P x2aa) ⟹ ?P (Node x1 x2)⟧
⟹ ?P ?trie

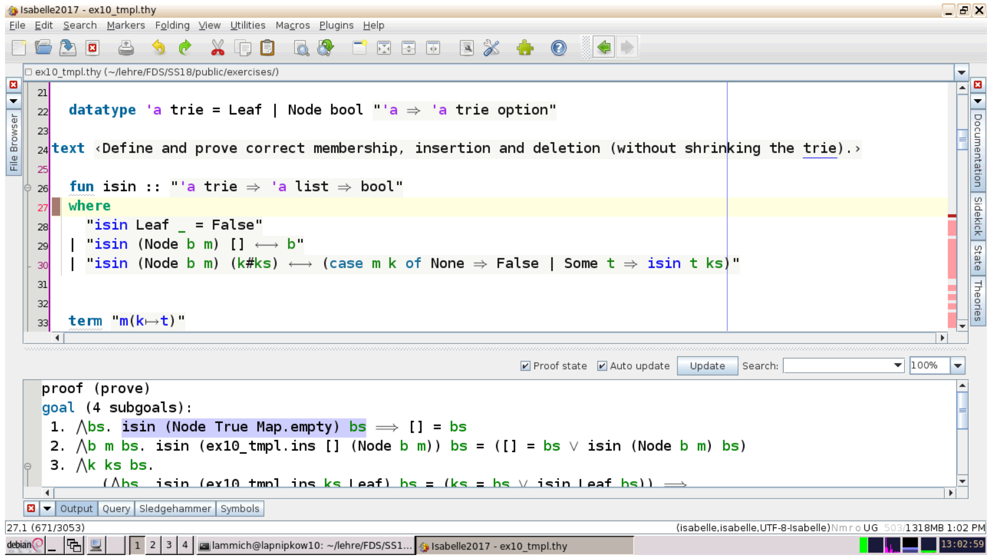

Isabelle2017 - ex10_tmpl.thy (modified)
case m k of
None ⇒ Node b (m(k↦ins ks Leaf))
| Some t ⇒ Node b (m(k↦ins ks t)))"
term ran
thm trie.induct
thm ins.induct
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
⟦?P [] Leaf; ⋀b m. ?P [] (Node b m); ⋀k ks. ?P ks Leaf ⟹ ?P (k # ks) Leaf;
⋀k ks b m. ⟦m k = None ⟹ ?P ks Leaf; ⋀x2. m k = Some x2 ⟹ ?P ks x2⟧ ⟹ ?P (k # ks) (Node b m)⟧
⟹ ?P ?a0.0 ?a1.0

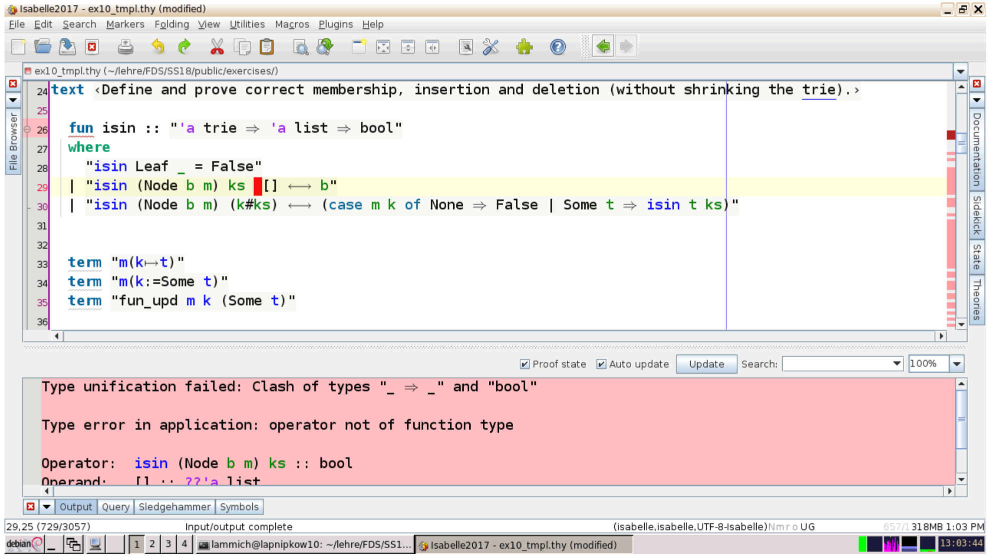

Isabelle2017 - ex10_tmpl.thy (modified)
case m k of
None ⇒ Node b (m(k↦ins ks Leaf))
| Some t ⇒ Node b (m(k↦ins ks t)))"
term ran
thm trie.induct
thm ins.induct
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
⟦?P [] Leaf; ⋀b m. ?P [] (Node b m); ⋀k ks. ?P ks Leaf ⟹ ?P (k # ks) Leaf;
⋀k ks b m. ⟦m k = None ⟹ ?P ks Leaf; ⋀x2. m k = Some x2 ⟹ ?P ks x2⟧ ⟹ ?P (k # ks) (Node b m)⟧
⟹ ?P ?a0.0 ?a1.0

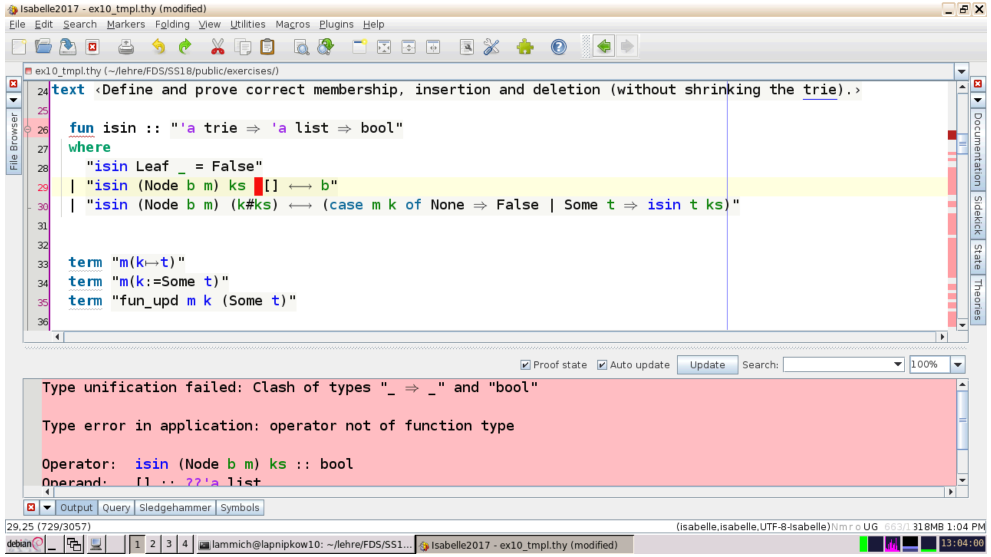

Isabelle2017 - ex10_tmpl.thy (modified)
case m k of
None ⇒ Node b (m(k↦ins ks Leaf))
| Some t ⇒ Node b (m(k↦ins ks t)))"
term ran
thm trie.induct
thm ins.induct
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
proof (prove)
goal (1 subgoal):
1. ex10_tmpl.isin (ex10_tmpl.ins as t) bs = (as = bs ∨ ex10_tmpl.isin t bs)

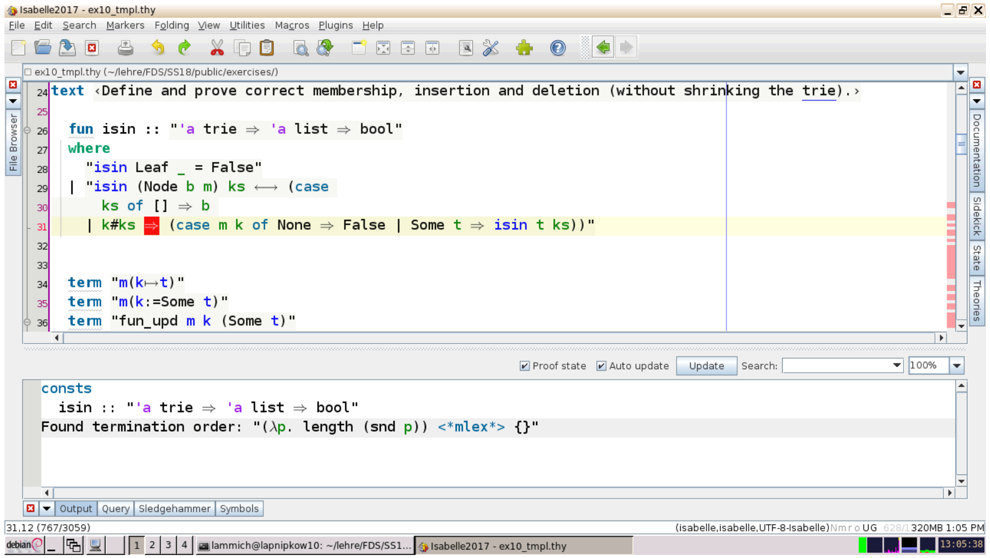

Isabelle2017 - ex10_tmpl.thy (modified)
term ran
thm trie.induct
thm ins.induct
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
apply (induction as t rule: ins.induct)
apply (auto)
proof (prove)
goal (9 subgoals):
1. ex10_tmpl.isin (Node True Map.empty) bs ⟹ [] = bs
2. ⋀b m. ⟦ex10_tmpl.isin (Node True m) bs; ¬ ex10_tmpl.isin (Node b m) bs⟧ ⟹ [] = bs
3. ⋀b m. ex10_tmpl.isin (Node b m) bs ⟹ ex10_tmpl.isin (Node True m) bs

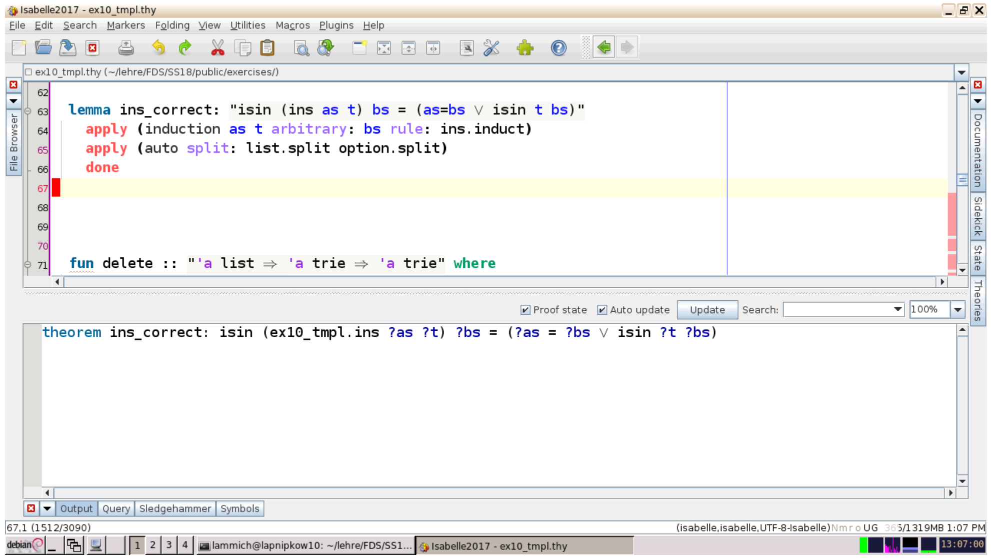

Isabelle2017 - ex10_tmpl.thy
thm trie.induct
thm ins.induct
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
apply (induction as t rule: ins.induct)
apply (auto)
proof (prove)
goal (4 subgoals):
1. isin (ex10_tmpl.ins [] Leaf) bs = ([] = bs ∨ isin Leaf bs)
2. ⋀b m. isin (ex10_tmpl.ins [] (Node b m)) bs = ([] = bs ∨ isin (Node b m) bs)
3. ⋀k ks. isin (ex10_tmpl.ins ks Leaf) bs = (ks = bs ∨ isin Leaf bs) ⟹

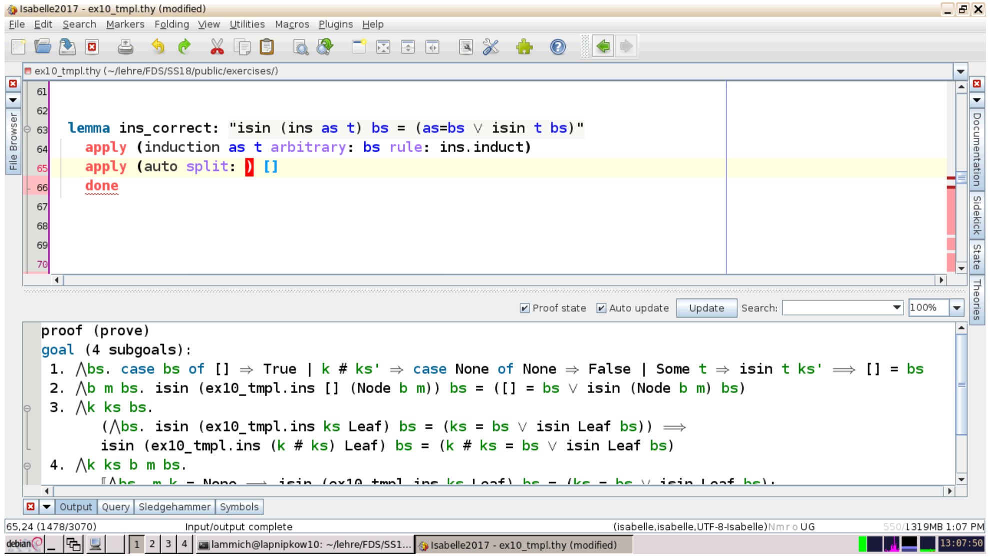

Isabelle2017 - ex10_tmpl.thy
datatype 'a trie = Leaf | Node bool "'a ⇒ 'a trie option"
text ‹Define and prove correct membership, insertion and deletion (without shrinking the trie).›
fun isin :: "'a trie ⇒ 'a list ⇒ bool"
where
"isin Leaf _ = False"
| "isin (Node b m) [] ⟷ b"
| "isin (Node b m) (k#ks) ⟷ (case m k of None ⇒ False | Some t ⇒ isin t ks)"
term "m(k↦t)"
proof (prove)
goal (4 subgoals):
1. ⋀bs. isin (Node True Map.empty) bs ⟹ [] = bs
2. ⋀b m bs. isin (ex10_tmpl.ins [] (Node b m)) bs = ([] = bs ∨ isin (Node b m) bs)
3. ⋀k ks bs.

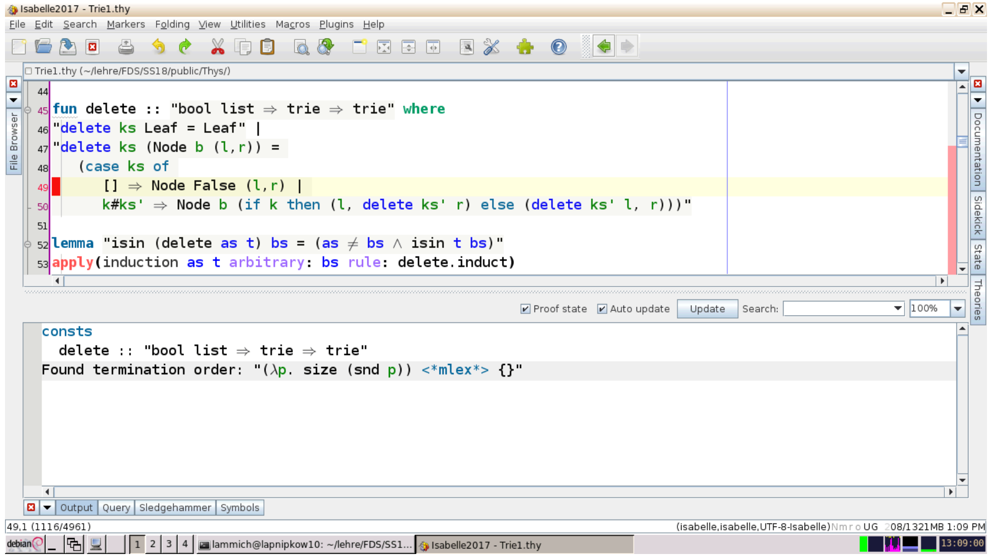

Isabelle2017 - ex10_tmpl.thy (modified)
text ‹Define and prove correct membership, insertion and deletion (without shrinking the trie).›
fun isin :: "'a trie ⇒ 'a list ⇒ bool"
where
"isin Leaf _ = False"
| "isin (Node b m) ks [] ⟷ b"
| "isin (Node b m) (k#ks) ⟷ (case m k of None ⇒ False | Some t ⇒ isin t ks)"
term "m(k↦t)"
term "m(k:=Some t)"
term "fun_upd m k (Some t)"
Type unification failed: Clash of types "_ ⇒ _" and "bool"
Type error in application: operator not of function type
Operator: isin (Node b m) ks :: bool

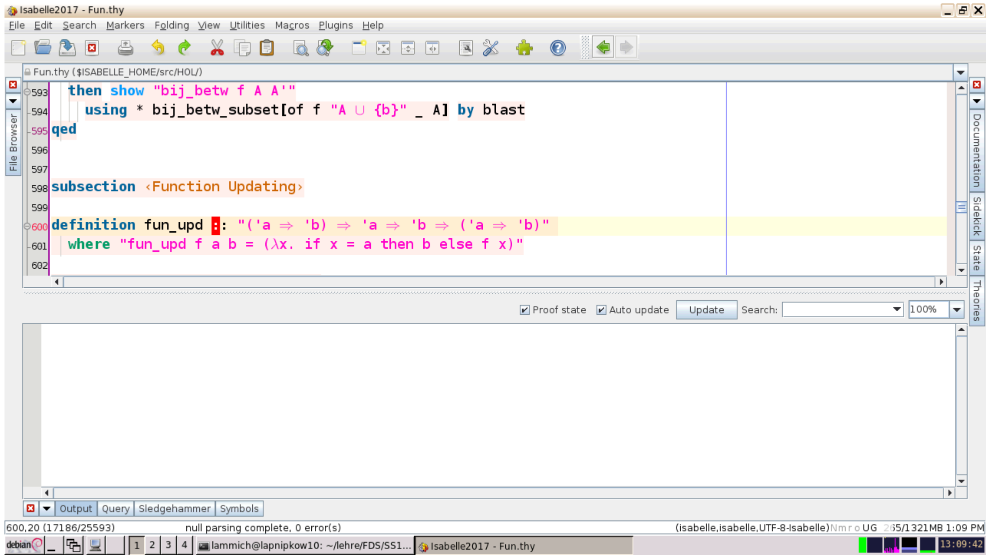

Isabelle2017 - ex10_tmpl.thy (modified)
text ‹Define and prove correct membership, insertion and deletion (without shrinking the trie).›
fun isin :: "'a trie ⇒ 'a list ⇒ bool"
where
"isin Leaf _ = False"
| "isin (Node b m) ks [] ⟷ b"
| "isin (Node b m) (k#ks) ⟷ (case m k of None ⇒ False | Some t ⇒ isin t ks)"
term "m(k↦t)"
term "m(k:=Some t)"
term "fun_upd m k (Some t)"
Type unification failed: Clash of types "_ ⇒ _" and "bool"
Type error in application: operator not of function type
Operator: isin (Node b m) ks :: bool

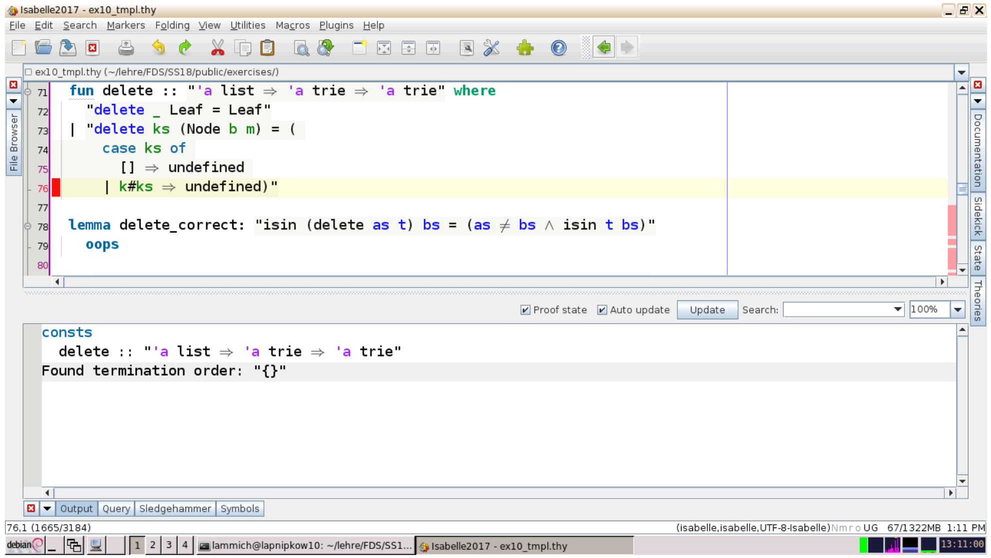

Isabelle2017 - ex10_tmpl.thy
text ‹Define and prove correct membership, insertion and deletion (without shrinking the trie).›
fun isin :: "'a trie ⇒ 'a list ⇒ bool"
where
"isin Leaf _ = False"
| "isin (Node b m) ks ⟷ (case
ks of [] ⇒ b
| k#ks ⇒ (case m k of None ⇒ False | Some t ⇒ isin t ks))"
term "m(k↦t)"
term "m(k:=Some t)"
term "fun_upd m k (Some t)"
consts
isin :: "'a trie ⇒ 'a list ⇒ bool"
Found termination order: "(λp. length (snd p)) <*mlex*> {}"

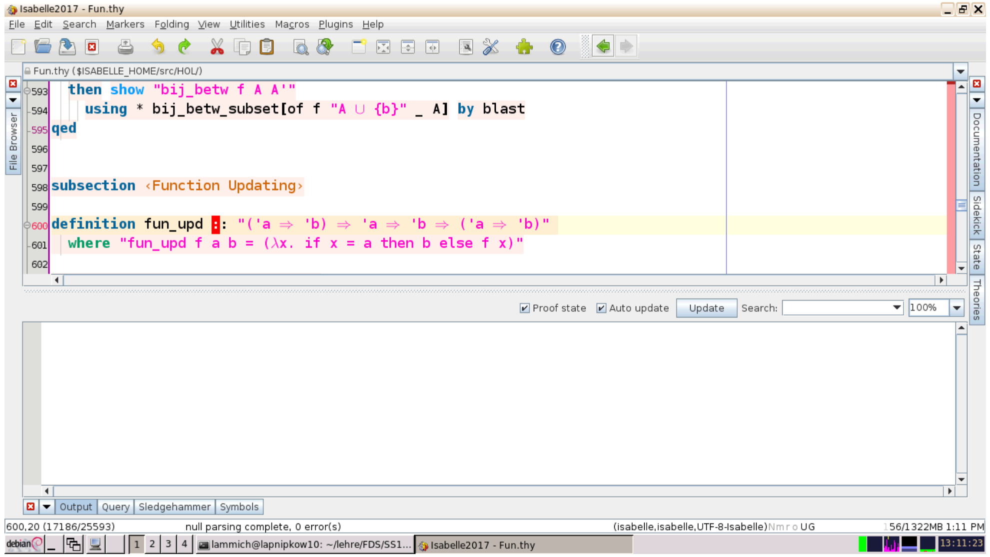
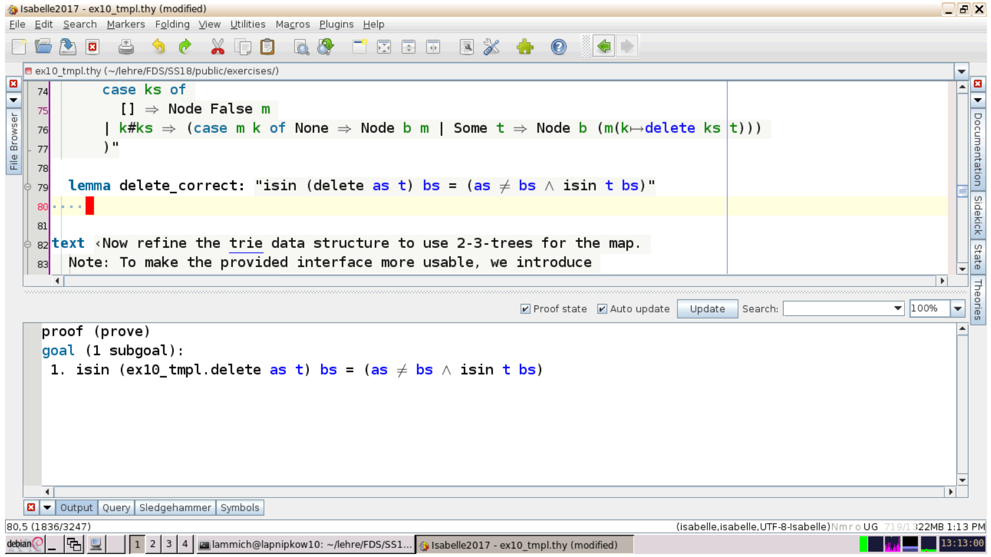
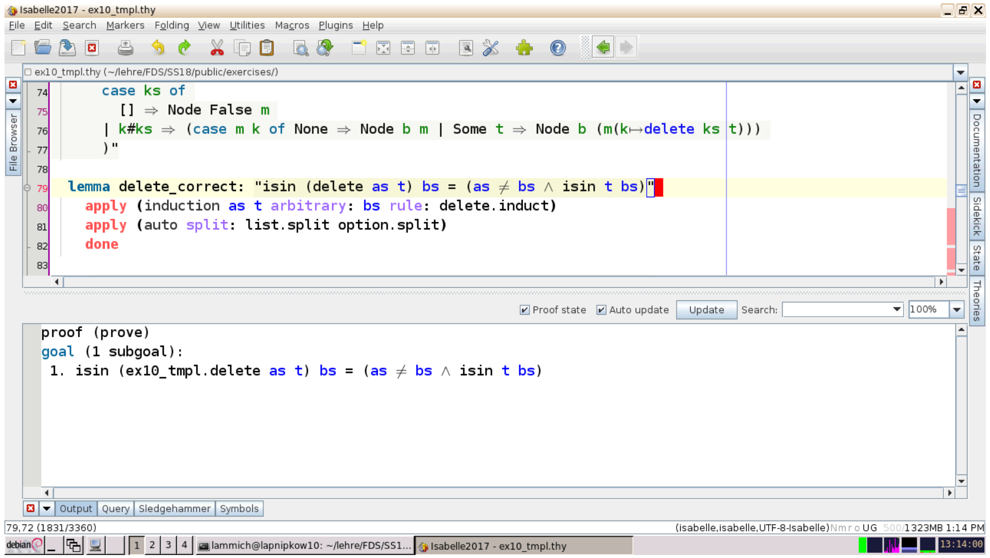
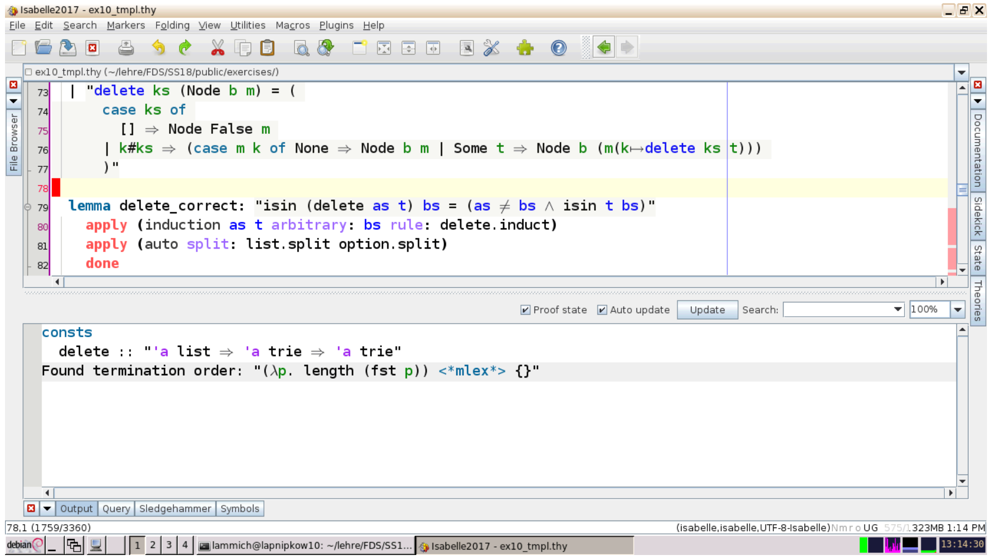

Isabelle2017 - ex10_tmpl.thy
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
apply (induction as arbitrary: bs rule: ins.induct)
apply (auto split: list.split option.split)
done
fun delete :: "'a list ⇒ 'a trie ⇒ 'a trie" where
theorem ins_correct: isin (ex10_tmpl.ins ?as ?t) ?bs = (?as = ?bs ∨ isin ?t ?bs)

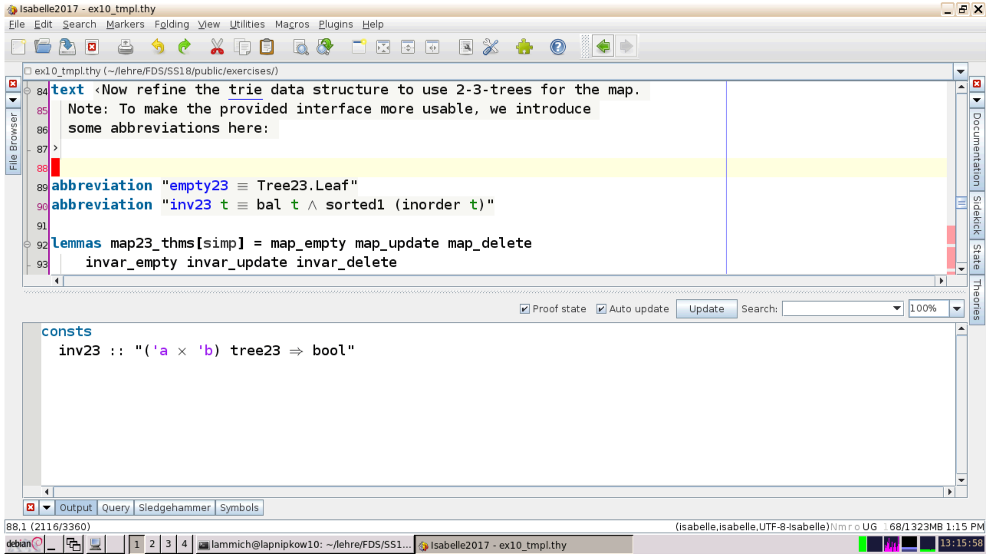

Isabelle2017 - ex10_tmpl.thy (modified)
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
lemma ins_correct: "isin (ins as t) bs = (as=bs ∨ isin t bs)"
apply (induction as t arbitrary: bs rule: ins.induct)
apply (auto split:) []
done
Proof state
Auto update
Update
Search:
100%
proof (prove)
goal (4 subgoals):
1. ⋀bs. case bs of [] ⇒ True | k # ks' ⇒ case None of None ⇒ False | Some t ⇒ isin t ks' ⟹ [] = bs
2. ⋀b m bs. isin (ex10_tmpl.ins [] (Node b m)) bs = ([] = bs ∨ isin (Node b m) bs)
3. ⋀k ks bs.
(⋀bs. isin (ex10_tmpl.ins ks Leaf) bs = (ks = bs ∨ isin Leaf bs)) ⟹
isin (ex10_tmpl.ins (k # ks) Leaf) bs = (k # ks = bs ∨ isin Leaf bs)
4. ⋀k ks b m bs.
Output
Query
Sledgehammer
Symbols
65.24 (1478/3070)
Input/output complete
(isabelle,isabelle,UTF-8-Isabelle)
1319MB 1:07 PM
lammich@lapnipkow10: ~/lehre/FDS/SS1...
Isabelle2017 - ex10_tmpl.thy (modified)
13:07:50

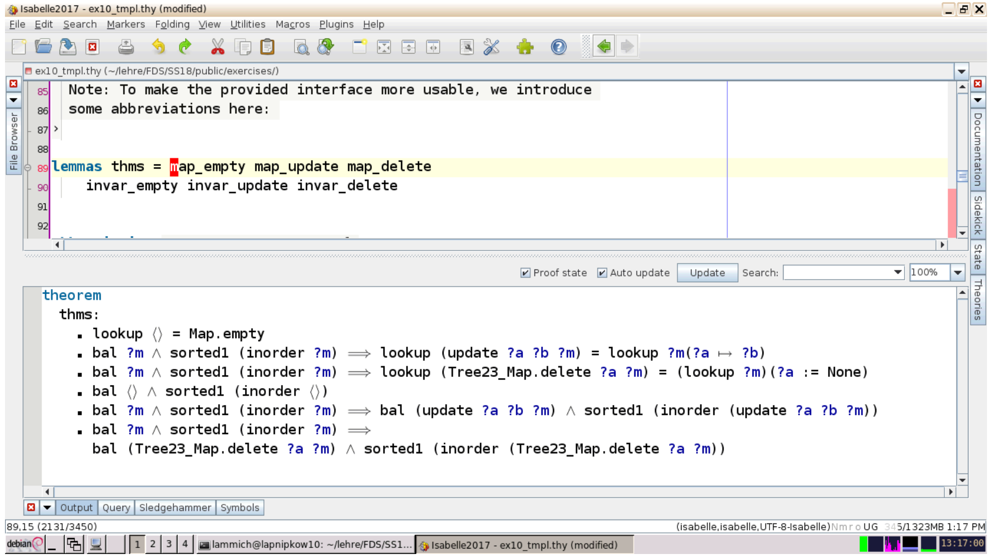

Isabelle2017 - Trie1.thy
Trie1.thy (~/lehre/FDS/SS18/public/Thys/)
fun delete :: "bool list ⇒ trie ⇒ trie" where
"delete ks Leaf = Leaf" |
"delete ks (Node b (l,r)) =
(case ks of
[] ⇒ Node False (l,r) |
k#ks' ⇒ Node b (if k then (l, delete ks' r) else (delete ks' l, r)))"
lemma "isin (delete as t) bs = (as ≠ bs ∧ isin t bs)"
apply(induction as t arbitrary: bs rule: delete.induct)
Proof state
Auto update
Update
Search:
100%
consts
delete :: "bool list ⇒ trie ⇒ trie"
Found termination order: "(λp. size (snd p)) <*mlex*> {}"
Output
Query
Sledgehammer
Symbols
49,1 (1116/4961)
(isabelle,isabelle,UTF-8-Isabelle)
08/1321MB 1:09 PM
lammich@lapnipkow10: ~/lehre/FDS/SS1...
Isabelle2017 - Trie1.thy
13:09:00

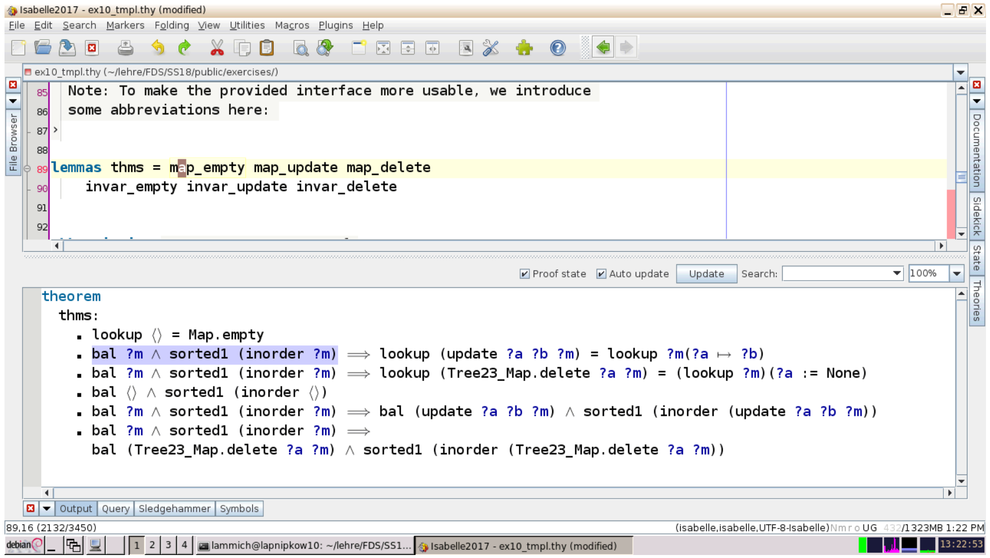

Isabelle2017 - Fun.thy
Fun.thy ($ISABELLE_HOME/src/HOL/)
then show "bij_betw f A A'"
using * bij_betw_subset[of f "A ∪ {b}" _ A] by blast
qed
subsection ‹Function Updating›
definition fun_upd :: "('a ⇒ 'b) ⇒ 'a ⇒ 'b ⇒ ('a ⇒ 'b)"
where "fun_upd f a b = (λx. if x = a then b else f x)"
Proof state
Auto update
Update
Search:
100%
Output
Query
Sledgehammer
Symbols
600,20 (17186/25593)
null parsing complete, 0 error(s)
(isabelle,isabelle,UTF-8-Isabelle)
1321MB 1:09 PM
lammich@lapnipkow10: ~/lehre/FDS/SS1...
Isabelle2017 - Fun.thy
13:09:42

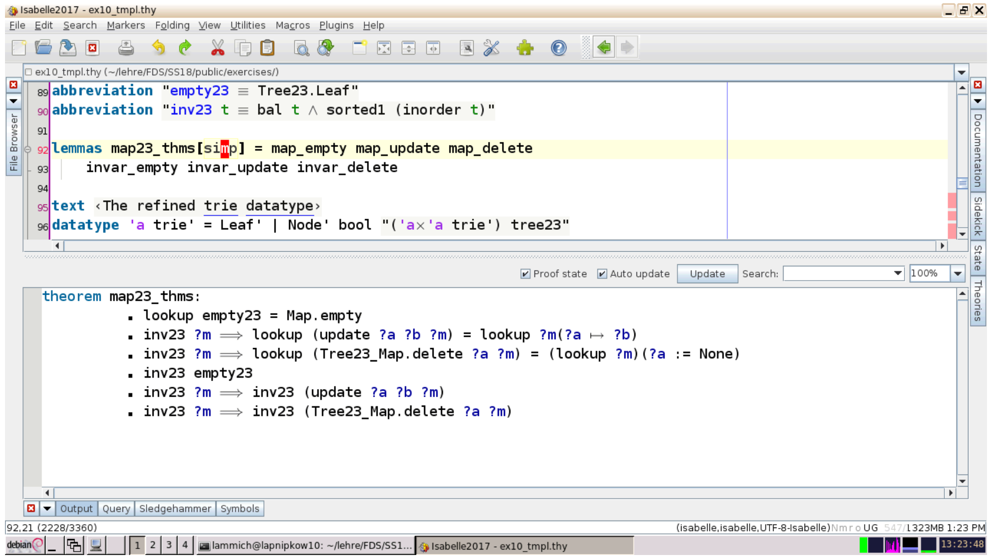

Isabelle2017 - ex10_tmpl.thy
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
fun delete :: "'a list ⇒ 'a trie ⇒ 'a trie" where
"delete _ Leaf = Leaf"
| "delete ks (Node b m) = (
case ks of
[] ⇒ undefined
| k#ks ⇒ undefined)"
lemma delete_correct: "isin (delete as t) bs = (as ≠ bs ∧ isin t bs)"
oops
Proof state
Auto update
Update
Search:
100%
consts
delete :: "'a list ⇒ 'a trie ⇒ 'a trie"
Found termination order: "{}"
Output
Query
Sledgehammer
Symbols
76,1 (1665/3184)
(isabelle,isabelle,UTF-8-Isabelle)
67/1322MB 1:11 PM
lammich@lapnipkow10: ~/lehre/FDS/SS1...
Isabelle2017 - ex10_tmpl.thy
13:11:00

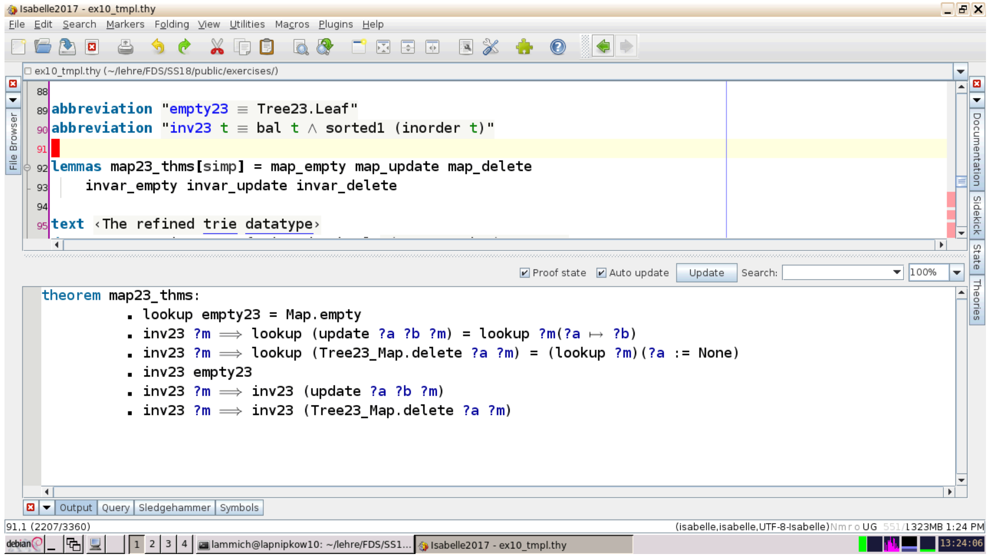

Isabelle2017 - Fun.thy
Fun.thy ($ISABELLE_HOME/src/HOL/)
then show "bij_betw f A A'"
using * bij_betw_subset[of f "A ∪ {b}" _ A] by blast
qed
subsection ‹Function Updating›
definition fun_upd :: "('a ⇒ 'b) ⇒ 'a ⇒ 'b ⇒ ('a ⇒ 'b)"
where "fun_upd f a b = (λx. if x = a then b else f x)"
Proof state
Auto update
Update
Search:
100%
Output
Query
Sledgehammer
Symbols
600,20 (17186/25593)
null parsing complete, 0 error(s)

Isabelle2017 - ex10_tmpl.thy (modified)
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
case ks of
[] ⇒ Node False m
| k#ks ⇒ (case m k of None ⇒ Node b m | Some t ⇒ Node b (m(k↦delete ks t)))
)"
lemma delete_correct: "isin (delete as t) bs = (as ≠ bs ∧ isin t bs)"
text ‹Now refine the trie data structure to use 2-3-trees for the map.
Note: To make the provided interface more usable, we introduce
proof (prove)
goal (1 subgoal):
1. isin (ex10_tmpl.delete as t) bs = (as ≠ bs ∧ isin t bs)
80,5 (1836/3247)

Isabelle2017 - ex10_tmpl.thy
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
case ks of
[] ⇒ Node False m
| k#ks ⇒ (case m k of None ⇒ Node b m | Some t ⇒ Node b (m(k↦delete ks t)))
)"
lemma delete_correct: "isin (delete as t) bs = (as ≠ bs ∧ isin t bs)"
apply (induction as t arbitrary: bs rule: delete.induct)
apply (auto split: list.split option.split)
done
proof (prove)
goal (1 subgoal):
1. isin (ex10_tmpl.delete as t) bs = (as ≠ bs ∧ isin t bs)
79,72 (1831/3360)

Isabelle2017 - ex10_tmpl.thy
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
| "delete ks (Node b m) = (
case ks of
[] ⇒ Node False m
| k#ks ⇒ (case m k of None ⇒ Node b m | Some t ⇒ Node b (m(k↦delete ks t)))
)"
lemma delete_correct: "isin (delete as t) bs = (as ≠ bs ∧ isin t bs)"
apply (induction as t arbitrary: bs rule: delete.induct)
apply (auto split: list.split option.split)
done
consts
delete :: "'a list ⇒ 'a trie ⇒ 'a trie"
Found termination order: "(λp. length (fst p)) <*mlex*> {}"
78,1 (1759/3360)

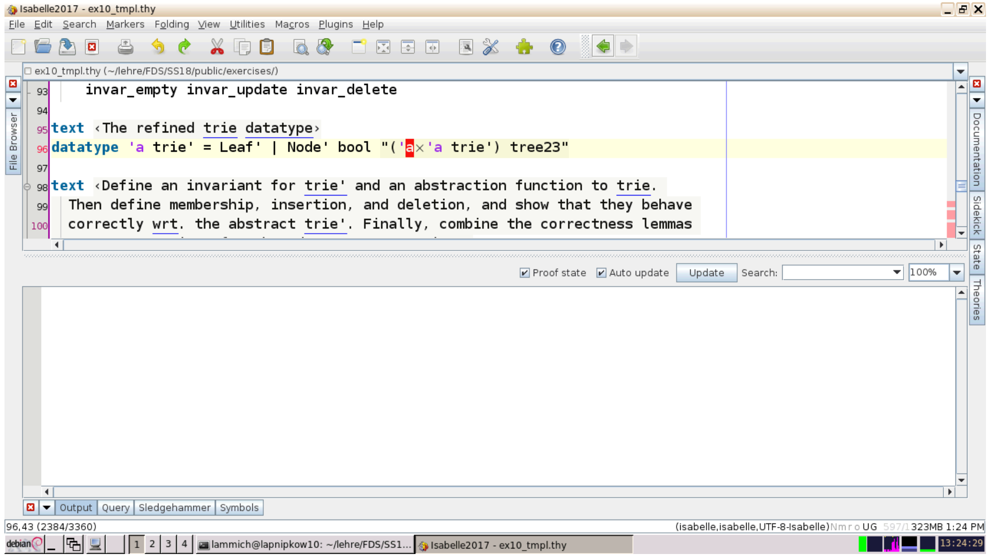

Isabelle2017 - ex10_tmpl.thy
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
text ‹Now refine the trie data structure to use 2-3-trees for the map.
Note: To make the provided interface more usable, we introduce
some abbreviations here:
›
abbreviation "empty23 ≡ Tree23.Leaf"
abbreviation "inv23 t ≡ bal t ∧ sorted1 (inorder t)"
lemmas map23_thms[simp] = map_empty map_update map_delete
invar_empty invar_update invar_delete
consts
inv23 :: "('a × 'b) tree23 ⇒ bool"
88.1 (2116/3360)

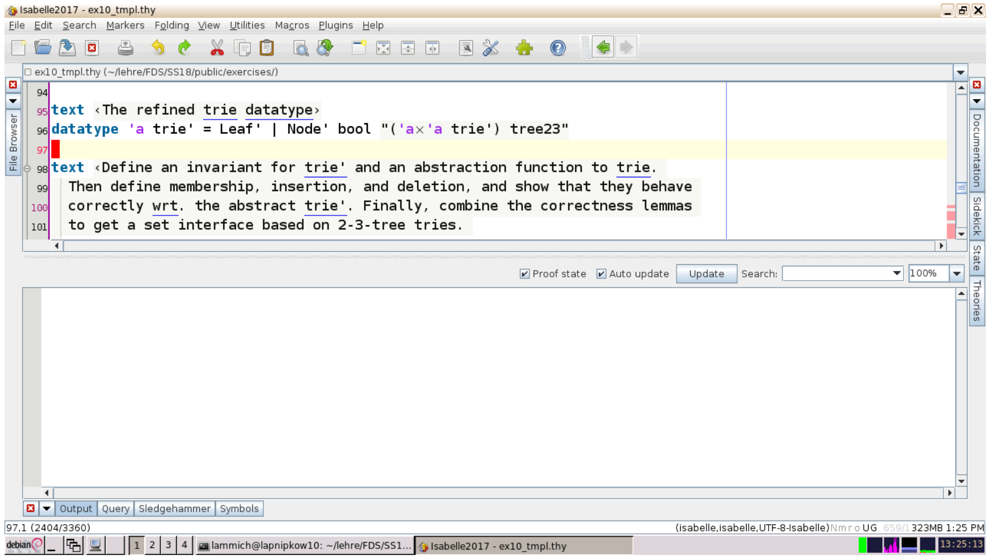

Isabelle2017 - ex10_tmpl.thy (modified)
Note: To make the provided interface more usable, we introduce
some abbreviations here:
›
lemmas thms = map_empty map_update map_delete
invar_empty invar_update invar_delete
theorem
thms:
lookup ⟨⟩ = Map.empty
bal ?m ∧ sorted1 (inorder ?m) ⟹ lookup (update ?a ?b ?m) = lookup ?m(?a ↦ ?b)
bal ?m ∧ sorted1 (inorder ?m) ⟹ lookup (Tree23_Map.delete ?a ?m) = (lookup ?m)(?a := None)
bal ⟨⟩ ∧ sorted1 (inorder ⟨⟩)
bal ?m ∧ sorted1 (inorder ?m) ⟹ bal (update ?a ?b ?m) ∧ sorted1 (inorder (update ?a ?b ?m))
bal ?m ∧ sorted1 (inorder ?m) ⟹
bal (Tree23_Map.delete ?a ?m) ∧ sorted1 (inorder (Tree23_Map.delete ?a ?m))
89.15 (2131/3450)

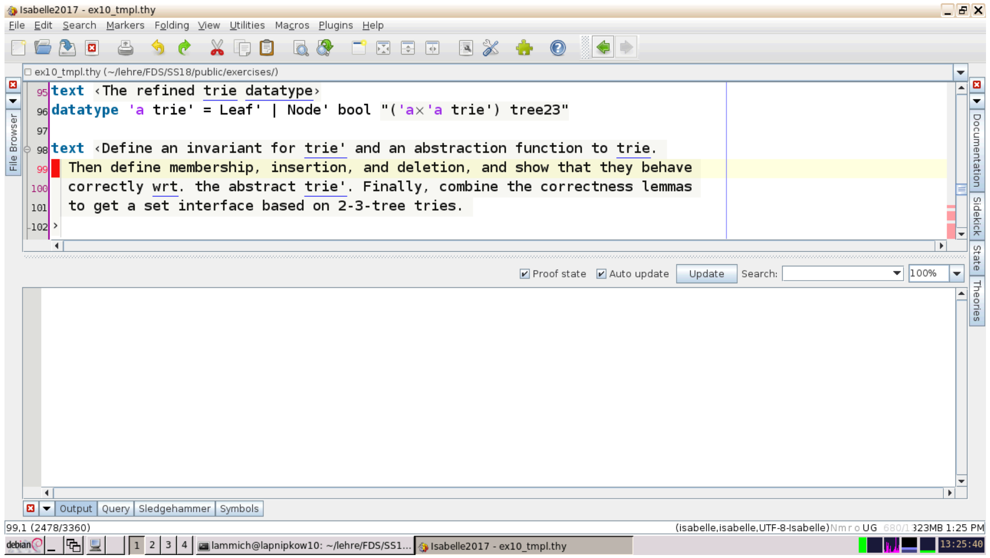

Isabelle2017 - ex10_tmpl.thy (modified)
Note: To make the provided interface more usable, we introduce
some abbreviations here:
›
lemmas thms = map_empty map_update map_delete
invar_empty invar_update invar_delete
theorem
thms:
lookup ⟨⟩ = Map.empty
bal ?m ∧ sorted1 (inorder ?m) ⟹ lookup (update ?a ?b ?m) = lookup ?m(?a ↦ ?b)
bal ?m ∧ sorted1 (inorder ?m) ⟹ lookup (Tree23_Map.delete ?a ?m) = (lookup ?m)(?a := None)
bal ⟨⟩ ∧ sorted1 (inorder ⟨⟩)
bal ?m ∧ sorted1 (inorder ?m) ⟹ bal (update ?a ?b ?m) ∧ sorted1 (inorder (update ?a ?b ?m))
bal ?m ∧ sorted1 (inorder ?m) ⟹
bal (Tree23_Map.delete ?a ?m) ∧ sorted1 (inorder (Tree23_Map.delete ?a ?m))
89.16 (2132/3450)

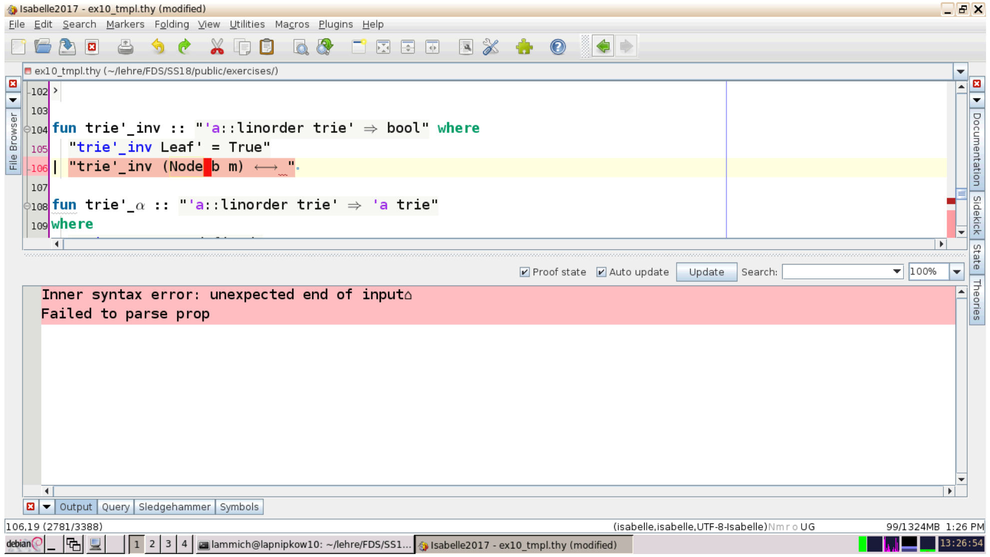
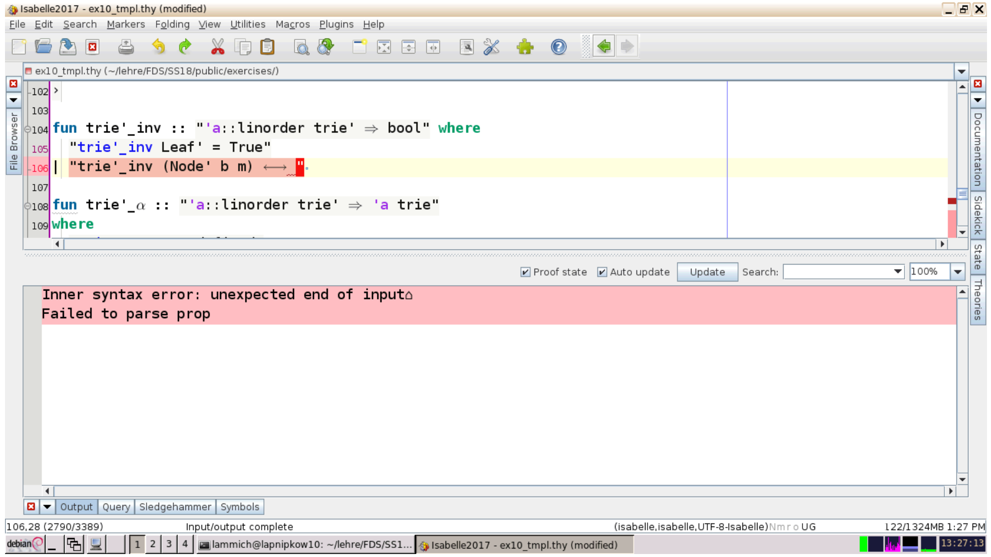
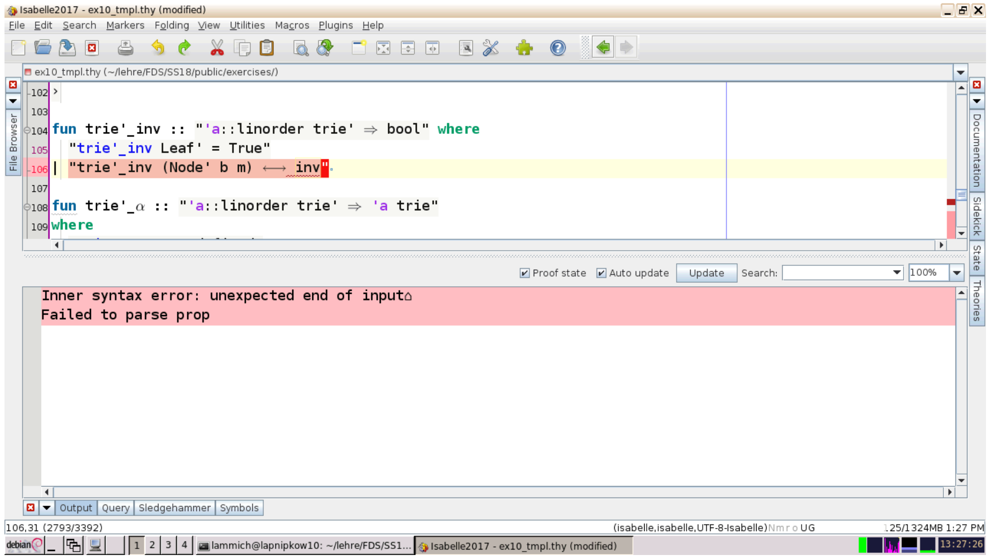
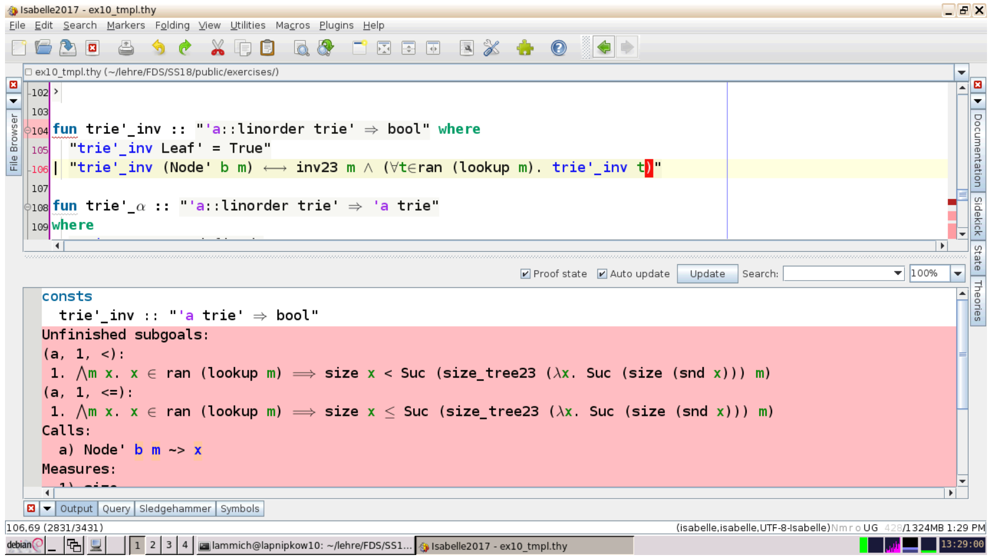

Isabelle2017 - ex10_tmpl.thy
abbreviation "empty23 ≡ Tree23.Leaf"
abbreviation "inv23 t ≡ bal t ∧ sorted1 (inorder t)"
lemmas map23_thms[simp] = map_empty map_update map_delete
invar_empty invar_update invar_delete
text ‹The refined trie datatype›
datatype 'a trie' = Leaf' | Node' bool "('a×'a trie') tree23"
theorem map23_thms:
lookup empty23 = Map.empty
inv23 ?m ⟹ lookup (update ?a ?b ?m) = lookup ?m(?a ↦ ?b)
inv23 ?m ⟹ lookup (Tree23_Map.delete ?a ?m) = (lookup ?m)(?a := None)
inv23 empty23
inv23 ?m ⟹ inv23 (update ?a ?b ?m)
inv23 ?m ⟹ inv23 (Tree23_Map.delete ?a ?m)
92.21 (2228/3360)

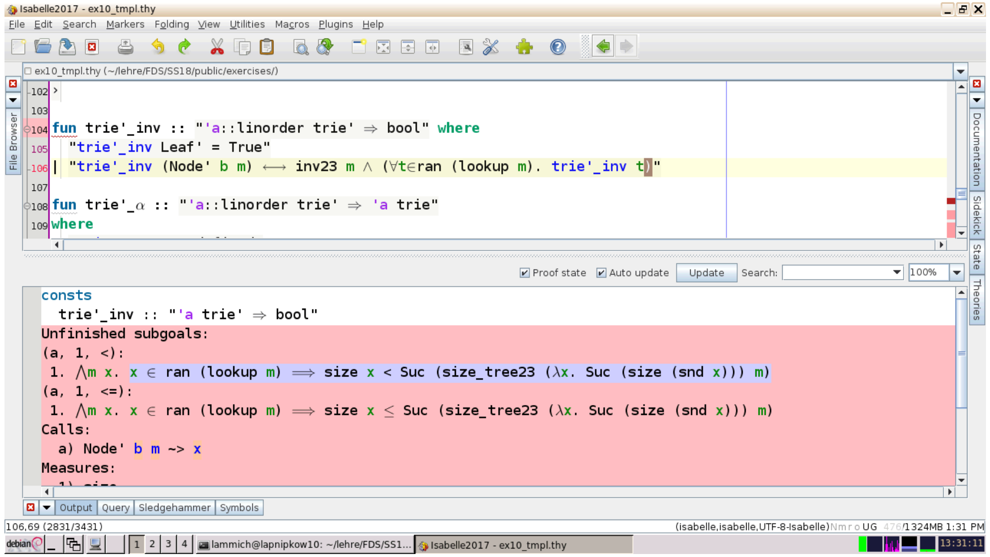
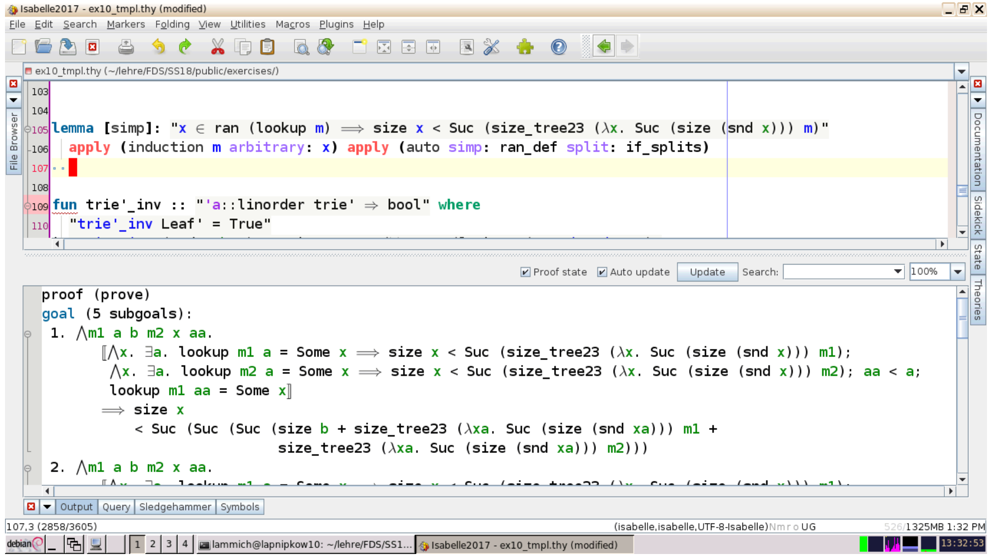
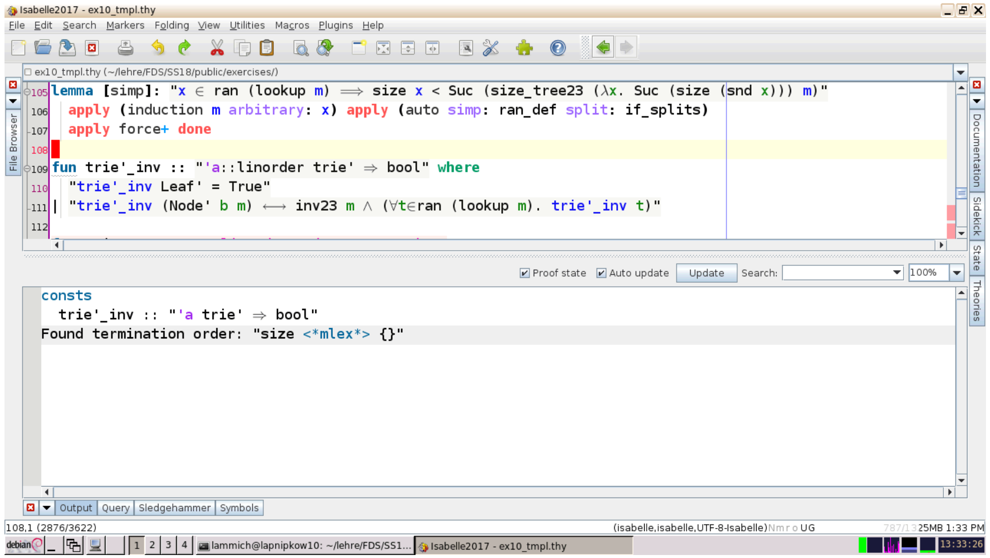
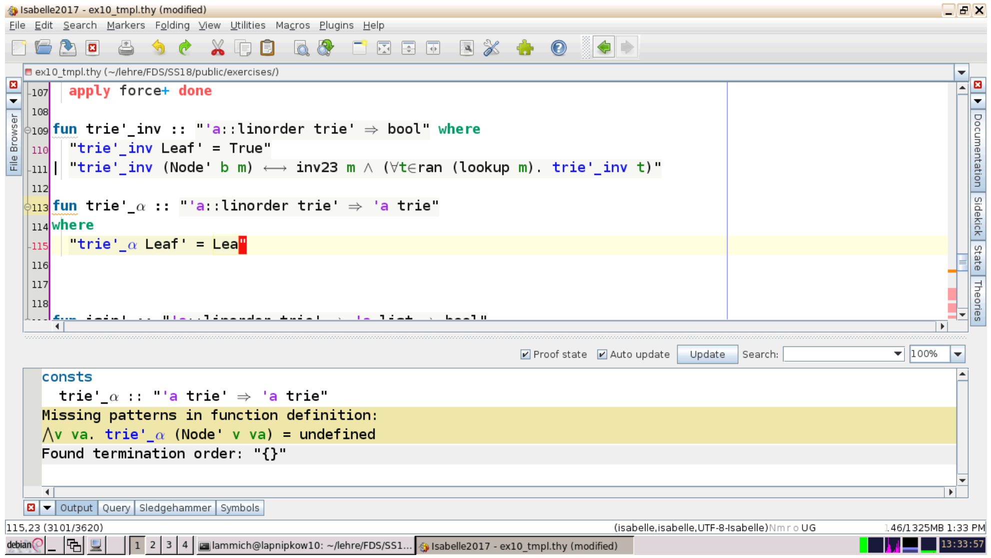

Isabelle2017 - ex10_tmpl.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
abbreviation "empty23 ≡ Tree23.Leaf"
abbreviation "inv23 t ≡ bal t ∧ sorted1 (inorder t)"
lemmas map23_thms[simp] = map_empty map_update map_delete
invar_empty invar_update invar_delete
text ‹The refined trie datatype›
Proof state
Auto update
Update
Search:
100%
theorem map23_thms:
lookup empty23 = Map.empty
inv23 ?m ⟹ lookup (update ?a ?b ?m) = lookup ?m(?a ↦ ?b)
inv23 ?m ⟹ lookup (Tree23_Map.delete ?a ?m) = (lookup ?m)(?a := None)
inv23 empty23
inv23 ?m ⟹ inv23 (update ?a ?b ?m)
inv23 ?m ⟹ inv23 (Tree23_Map.delete ?a ?m)
Output Query Sledgehammer Symbols
91,1 (2207/3360)
(isabelle,isabelle,UTF-8-Isabelle)
1:24 PM

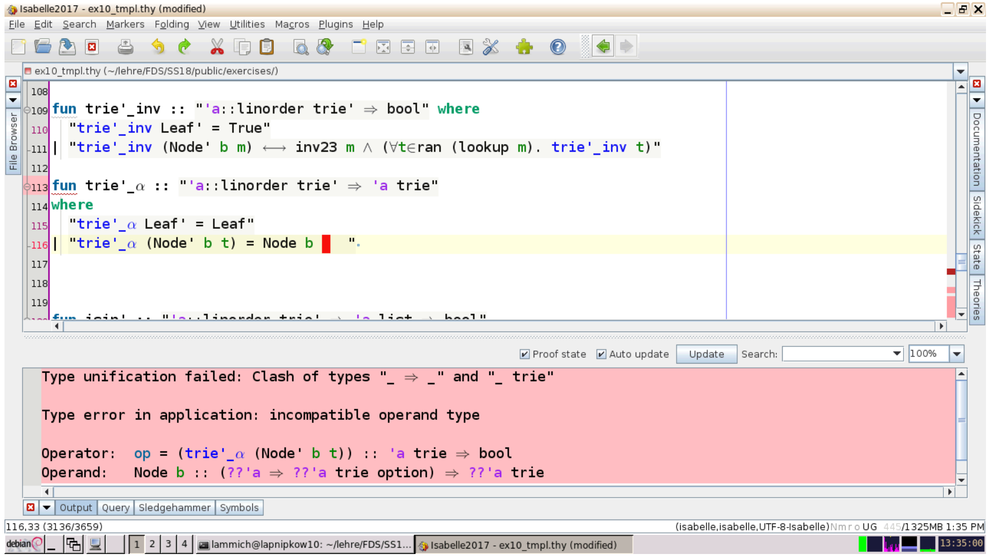

Isabelle2017 - ex10_tmpl.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
invar_empty invar_update invar_delete
text ‹The refined trie datatype›
datatype 'a trie' = Leaf' | Node' bool "('a × 'a trie') tree23"
text ‹Define an invariant for trie' and an abstraction function to trie.
Then define membership, insertion, and deletion, and show that they behave
correctly wrt. the abstract trie'. Finally, combine the correctness lemmas
Proof state
Auto update
Update
Search:
100%
Output Query Sledgehammer Symbols
96,43 (2384/3360)
(isabelle,isabelle,UTF-8-Isabelle)
1:24 PM

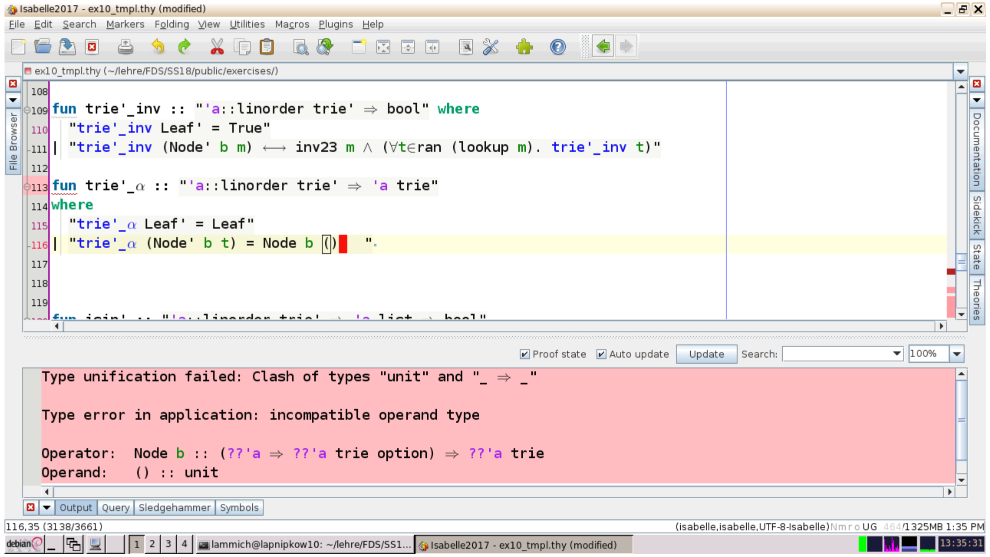

Isabelle2017 - ex10_tmpl.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
text ‹The refined trie datatype›
datatype 'a trie' = Leaf' | Node' bool "('a × 'a trie') tree23"
text ‹Define an invariant for trie' and an abstraction function to trie.
Then define membership, insertion, and deletion, and show that they behave
correctly wrt. the abstract trie'. Finally, combine the correctness lemmas
to get a set interface based on 2-3-tree tries.
Proof state
Auto update
Update
Search:
100%
Output Query Sledgehammer Symbols
97,1 (2404/3360)
(isabelle,isabelle,UTF-8-Isabelle)
1:25 PM

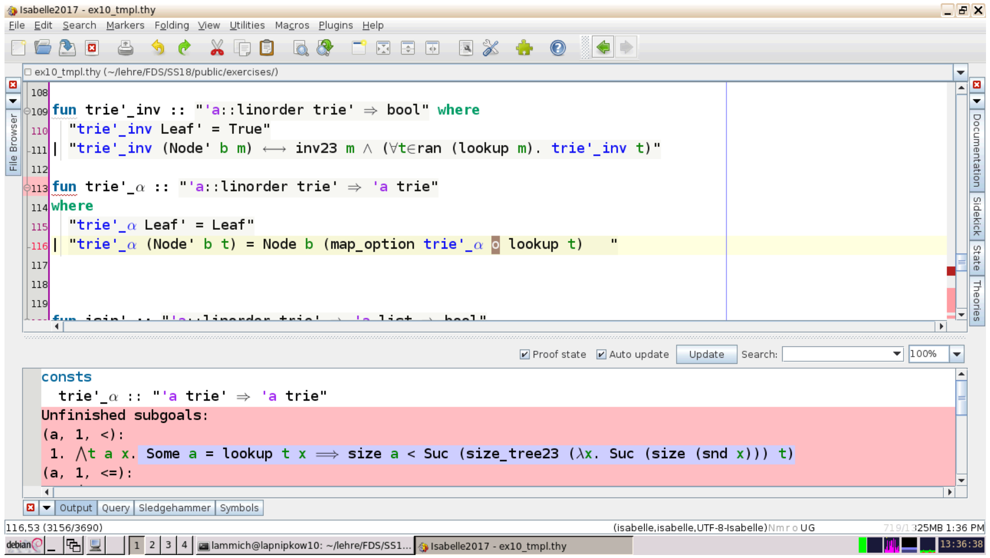

Isabelle2017 - ex10_tmpl.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
text ‹The refined trie datatype›
datatype 'a trie' = Leaf' | Node' bool "('a × 'a trie') tree23"
text ‹Define an invariant for trie' and an abstraction function to trie.
Then define membership, insertion, and deletion, and show that they behave
correctly wrt. the abstract trie'. Finally, combine the correctness lemmas
to get a set interface based on 2-3-tree tries.
›
Proof state
Auto update
Update
Search:
100%
Output Query Sledgehammer Symbols
99,1 (2478/3360)
(isabelle,isabelle,UTF-8-Isabelle)
1:25 PM

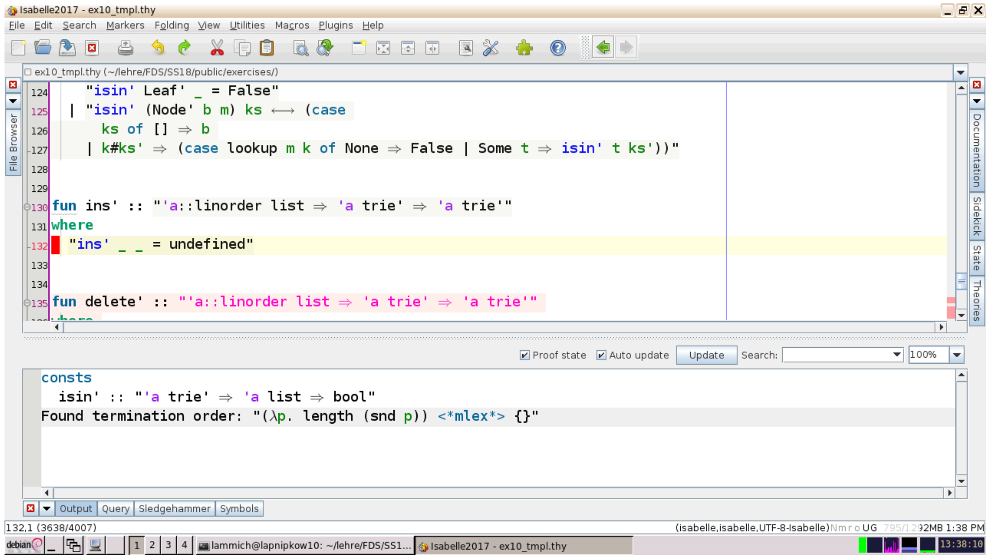

Isabelle2017 - ex10_tmpl.thy (modified)
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv Leaf' = True"
| "trie'_inv (Node b m) ⟷ "
fun trie'_α :: "'a::linorder trie' ⇒ 'a trie"
where
Inner syntax error: unexpected end of input
Failed to parse prop

Isabelle2017 - ex10_tmpl.thy (modified)
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv Leaf' = True"
| "trie'_inv (Node' b m) ⟷ "
fun trie'_α :: "'a::linorder trie' ⇒ 'a trie"
where
Inner syntax error: unexpected end of input
Failed to parse prop

Isabelle2017 - ex10_tmpl.thy (modified)
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv Leaf' = True"
| "trie'_inv (Node' b m) ⟷ inv"
fun trie'_α :: "'a::linorder trie' ⇒ 'a trie"
where
Inner syntax error: unexpected end of input
Failed to parse prop

Isabelle2017 - ex10_tmpl.thy
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv Leaf' = True"
| "trie'_inv (Node' b m) ⟷ inv23 m ∧ (∀t∈ran (lookup m). trie'_inv t)"
fun trie'_α :: "'a::linorder trie' ⇒ 'a trie"
where
consts
trie'_inv :: "'a trie' ⇒ bool"
Unfinished subgoals:
(a, 1, <):
1. ⋀m x. x ∈ ran (lookup m) ⟹ size x < Suc (size_tree23 (λx. Suc (size (snd x))) m)
(a, 1, <=):
1. ⋀m x. x ∈ ran (lookup m) ⟹ size x ≤ Suc (size_tree23 (λx. Suc (size (snd x))) m)
Calls:
a) Node' b m ~> x
Measures:

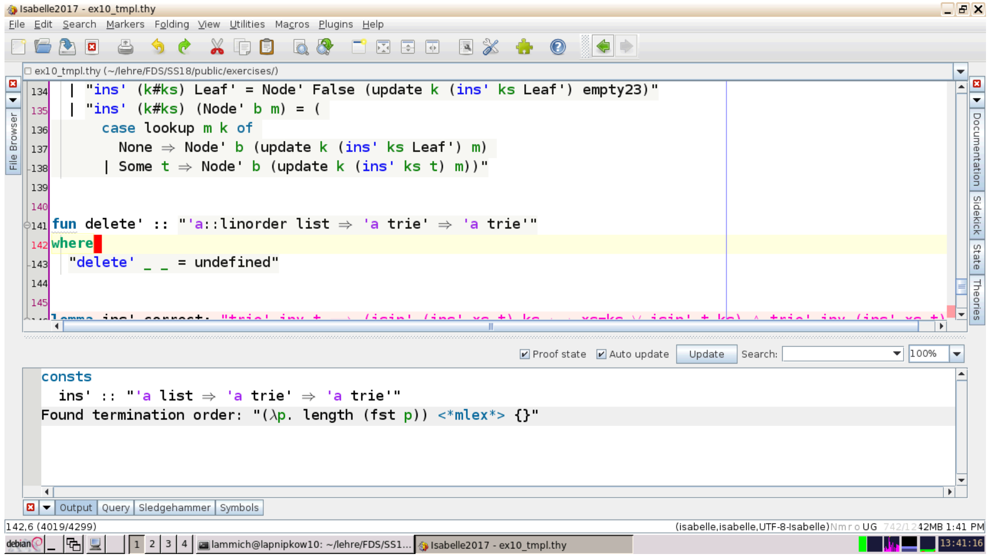

Isabelle2017 - ex10_tmpl.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv Leaf' = True"
| "trie'_inv (Node' b m) ⟷ inv23 m ∧ (∀t∈ran (lookup m). trie'_inv t)"
fun trie'_α :: "'a::linorder trie' ⇒ 'a trie"
where
Proof state
Auto update
Update
Search:
100%
consts
trie'_inv :: "'a trie' ⇒ bool"
Unfinished subgoals:
(a, 1, <):
1. ⋀m x. x ∈ ran (lookup m) ⟹ size x < Suc (size_tree23 (λx. Suc (size (snd x))) m)
(a, 1, <=):
1. ⋀m x. x ∈ ran (lookup m) ⟹ size x ≤ Suc (size_tree23 (λx. Suc (size (snd x))) m)
Calls:
a) Node' b m ~> x
Measures:
Output Query Sledgehammer Symbols
106,69 (2831/3431)
(isabelle,isabelle,UTF-8-Isabelle)
1:31 PM
lammich@lapnipkow10: ~/lehre/FDS/SS1...
Isabelle2017 - ex10_tmpl.thy
13:31:11

Isabelle2017 - ex10_tmpl.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
lemma [simp]: "x ∈ ran (lookup m) ⟹ size x < Suc (size_tree23 (λx. Suc (size (snd x))) m)"
apply (induction m arbitrary: x) apply (auto simp: ran_def split: if_splits)
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv Leaf' = True"
Proof state
Auto update
Update
Search:
100%
proof (prove)
goal (5 subgoals):
1. ⋀m1 a b m2 x aa.
⟦⋀x. ∃a. lookup m1 a = Some x ⟹ size x < Suc (size_tree23 (λx. Suc (size (snd x))) m1);
⋀x. ∃a. lookup m2 a = Some x ⟹ size x < Suc (size_tree23 (λx. Suc (size (snd x))) m2); aa < a;
lookup m1 aa = Some x⟧
⟹ size x
< Suc (Suc (Suc (size b + size_tree23 (λxa. Suc (size (snd xa))) m1 +
size_tree23 (λxa. Suc (size (snd xa))) m2)))
2. ⋀m1 a b m2 x aa.
Output Query Sledgehammer Symbols
107,3 (2858/3605)
(isabelle,isabelle,UTF-8-Isabelle)
1:32 PM
lammich@lapnipkow10: ~/lehre/FDS/SS1...
Isabelle2017 - ex10_tmpl.thy (modified)
13:32:53

Isabelle2017 - ex10_tmpl.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
lemma [simp]: "x ∈ ran (lookup m) ⟹ size x < Suc (size_tree23 (λx. Suc (size (snd x))) m)"
apply (induction m arbitrary: x) apply (auto simp: ran_def split: if_splits)
apply force+ done
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv Leaf' = True"
| "trie'_inv (Node' b m) ⟷ inv23 m ∧ (∀t∈ran (lookup m). trie'_inv t)"
Proof state
Auto update
Update
Search:
100%
consts
trie'_inv :: "'a trie' ⇒ bool"
Found termination order: "size <*mlex*> {}"
Output Query Sledgehammer Symbols
108,1 (2876/3622)
(isabelle,isabelle,UTF-8-Isabelle)
1:33 PM
lammich@lapnipkow10: ~/lehre/FDS/SS1...
Isabelle2017 - ex10_tmpl.thy
13:33:26

Isabelle2017 - ex10_tmpl.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex10_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
apply force+ done
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv Leaf' = True"
| "trie'_inv (Node' b m) ⟷ inv23 m ∧ (∀t∈ran (lookup m). trie'_inv t)"
fun trie'_α :: "'a::linorder trie' ⇒ 'a trie"
where
"trie'_α Leaf' = Lea"
Proof state
Auto update
Update
Search:
100%
consts
trie'_α :: "'a trie' ⇒ 'a trie"
Missing patterns in function definition:
⋀v va. trie'_α (Node' v va) = undefined
Found termination order: "{}"
Output Query Sledgehammer Symbols
115,23 (3101/3620)
(isabelle,isabelle,UTF-8-Isabelle)
1:33 PM
lammich@lapnipkow10: ~/lehre/FDS/SS1...
Isabelle2017 - ex10_tmpl.thy (modified)
13:33:57

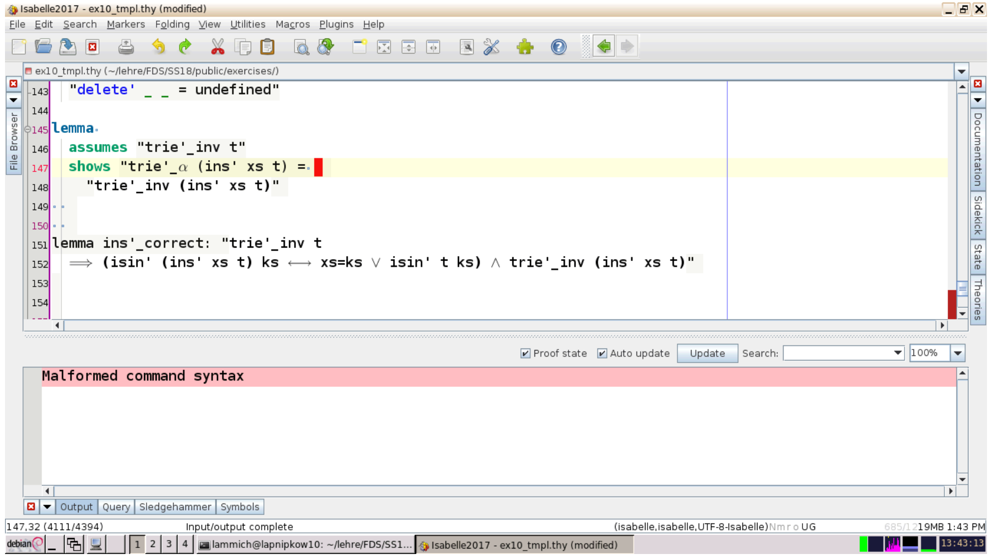

Isabelle2017 - ex10_tmpl.thy (modified)
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv Leaf' = True"
| "trie'_inv (Node' b m) ⟷ inv23 m ∧ (∀t∈ran (lookup m). trie'_inv t)"
fun trie'_α :: "'a::linorder trie' ⇒ 'a trie"
where
"trie'_α Leaf' = Leaf"
| "trie'_α (Node' b t) = Node b "
Type unification failed: Clash of types "_ ⇒ _" and "_ trie"
Type error in application: incompatible operand type
Operator: op = (trie'_α (Node' b t)) :: 'a trie ⇒ bool
Operand: Node b :: (??'a ⇒ ??'a trie option) ⇒ ??'a trie

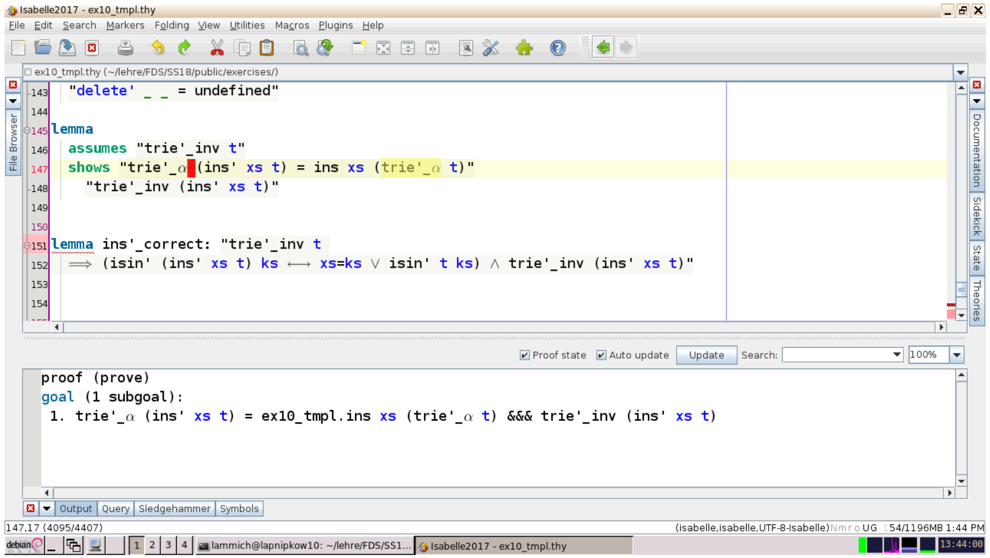

Isabelle2017 - ex10_tmpl.thy (modified)
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv Leaf' = True"
| "trie'_inv (Node' b m) ⟷ inv23 m ∧ (∀t∈ran (lookup m). trie'_inv t)"
fun trie'_α :: "'a::linorder trie' ⇒ 'a trie"
where
"trie'_α Leaf' = Leaf"
| "trie'_α (Node' b t) = Node b () "
Type unification failed: Clash of types "unit" and "_ ⇒ _"
Type error in application: incompatible operand type
Operator: Node b :: (??'a ⇒ ??'a trie option) ⇒ ??'a trie
Operand: () :: unit

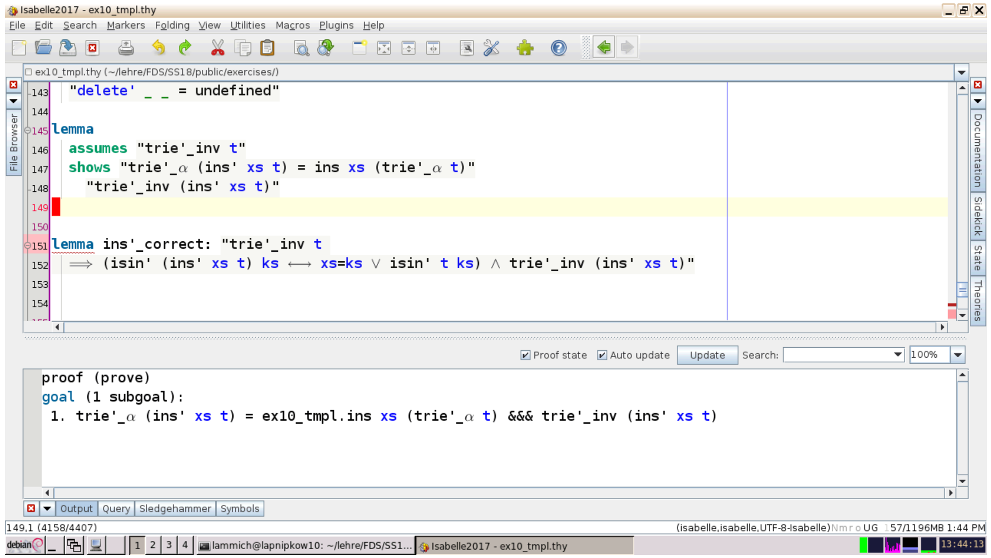

Isabelle2017 - ex10_tmpl.thy
fun trie'_inv :: "'a::linorder trie' ⇒ bool" where
"trie'_inv Leaf' = True"
| "trie'_inv (Node' b m) ⟷ inv23 m ∧ (∀t∈ran (lookup m). trie'_inv t)"
fun trie'_α :: "'a::linorder trie' ⇒ 'a trie"
where
"trie'_α Leaf' = Leaf"
| "trie'_α (Node' b t) = Node b (map_option trie'_α o lookup t) "
consts
trie'_α :: "'a trie' ⇒ 'a trie"
Unfinished subgoals:
(a, 1, <):
1. ⋀t a x. Some a = lookup t x ⟹ size a < Suc (size_tree23 (λx. Suc (size (snd x))) t)
(a, 1, <=):

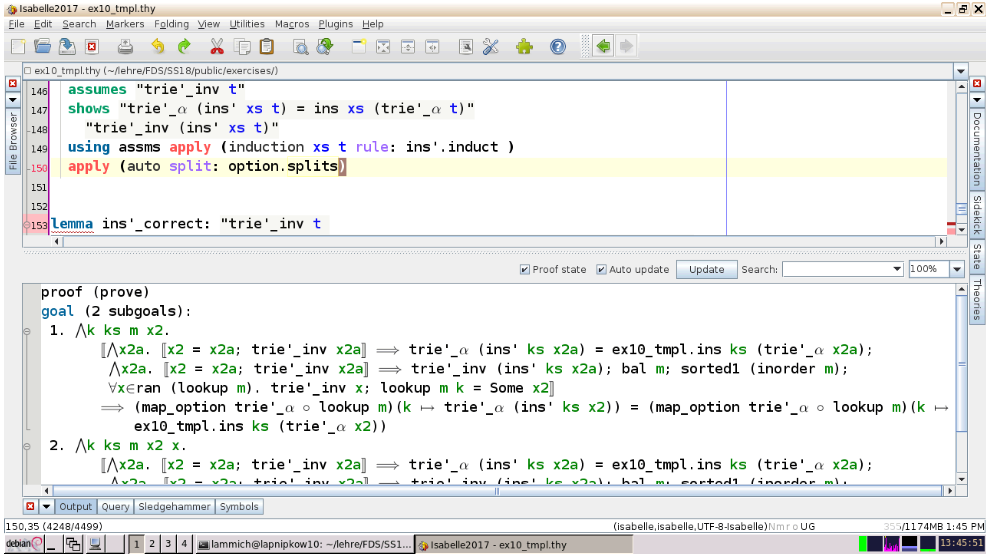

Isabelle2017 - ex10_tmpl.thy
"isin' Leaf' _ = False"
| "isin' (Node' b m) ks ⟷ (case
ks of [] ⇒ b
| k#ks' ⇒ (case lookup m k of None ⇒ False | Some t ⇒ isin' t ks'))"
fun ins' :: "'a::linorder list ⇒ 'a trie' ⇒ 'a trie'"
where
"ins' _ _ = undefined"
fun delete' :: "'a::linorder list ⇒ 'a trie' ⇒ 'a trie'"
consts
isin' :: "'a trie' ⇒ 'a list ⇒ bool"
Found termination order: "(λp. length (snd p)) <*mlex*> {}"

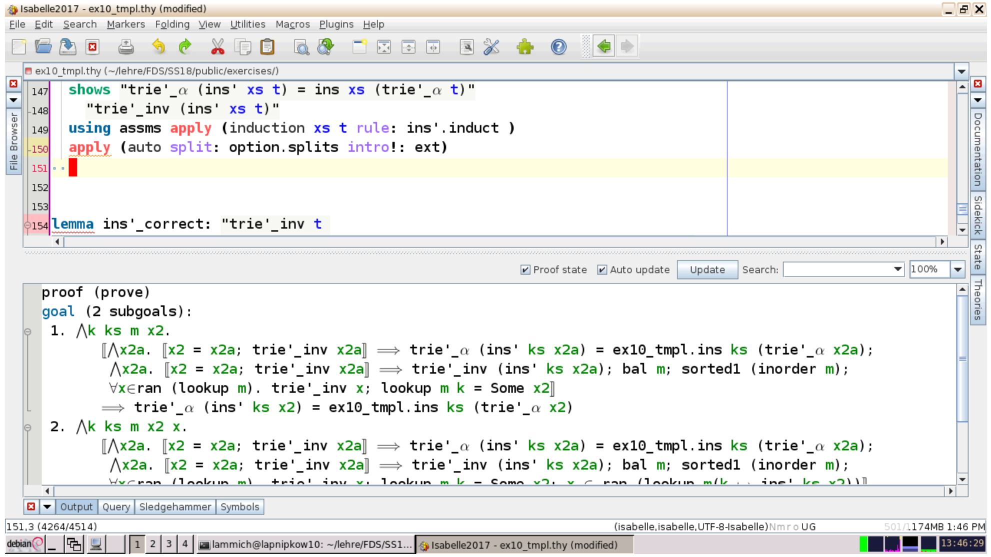

Isabelle2017 - ex10_tmpl.thy
| "ins' (k#ks) Leaf' = Node' False (update k (ins' ks Leaf') empty23)"
| "ins' (k#ks) (Node' b m) = (
case lookup m k of
None ⇒ Node' b (update k (ins' ks Leaf') m)
| Some t ⇒ Node' b (update k (ins' ks t) m))"
fun delete' :: "'a::linorder list ⇒ 'a trie' ⇒ 'a trie'"
where
"delete' _ _ = undefined"
consts
ins' :: "'a list ⇒ 'a trie' ⇒ 'a trie'"
Found termination order: "(λp. length (fst p)) <*mlex*> {}"

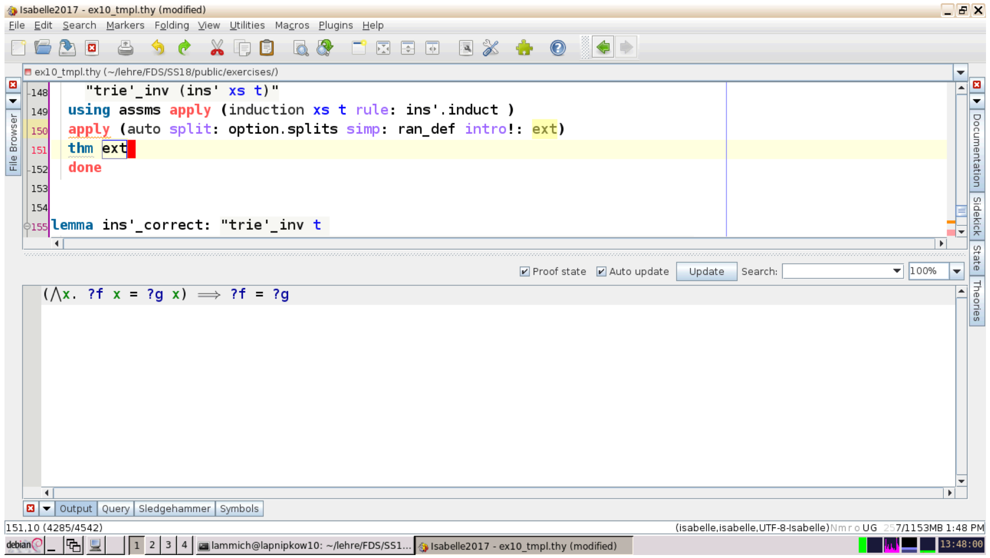

Isabelle2017 - ex10_tmpl.thy (modified)
"delete' _ _ = undefined"
lemma
assumes "trie'_inv t"
shows "trie'_α (ins' xs t) =
"trie'_inv (ins' xs t)"
lemma ins'_correct: "trie'_inv t
⟹ (isin' (ins' xs t) ks ⟷ xs=ks ∨ isin' t ks) ∧ trie'_inv (ins' xs t)"
Malformed command syntax

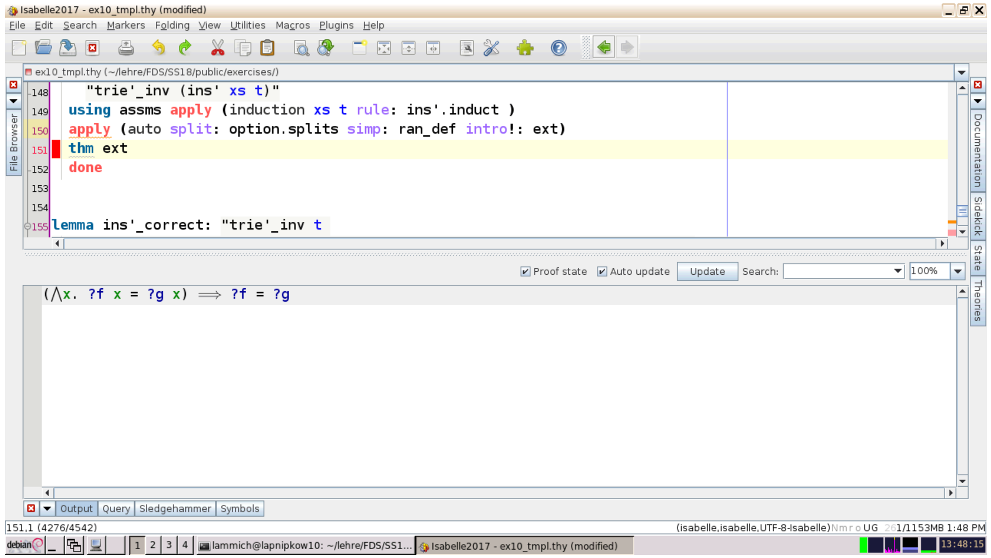

Isabelle2017 - ex10_tmpl.thy
"delete' _ _ = undefined"
lemma
assumes "trie'_inv t"
shows "trie'_α (ins' xs t) = ins xs (trie'_α t)"
"trie'_inv (ins' xs t)"
lemma ins'_correct: "trie'_inv t
⟹ (isin' (ins' xs t) ks ⟷ xs=ks ∨ isin' t ks) ∧ trie'_inv (ins' xs t)"
proof (prove)
goal (1 subgoal):
1. trie'_α (ins' xs t) = ex10_tmpl.ins xs (trie'_α t) &&& trie'_inv (ins' xs t)

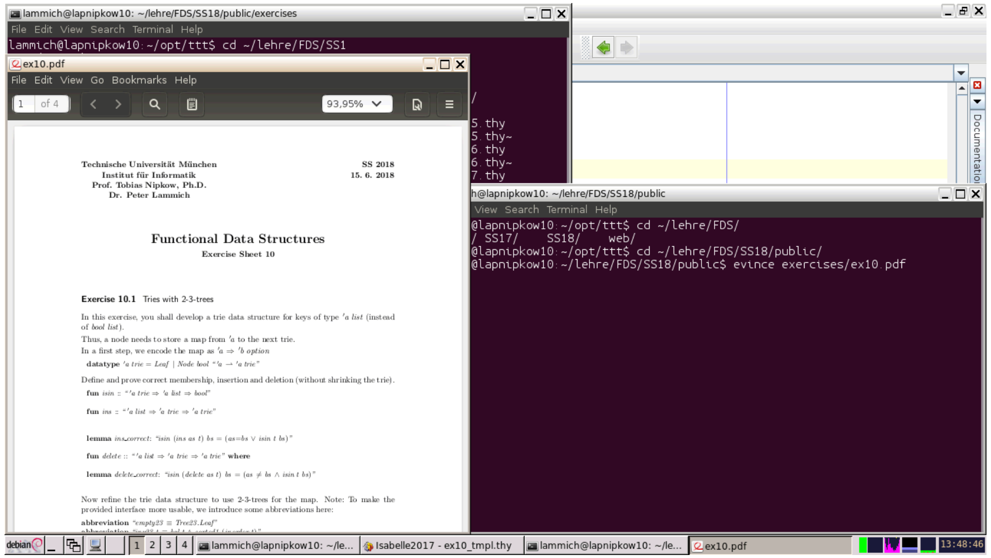

Isabelle2017 - ex10_tmpl.thy
"delete' _ _ = undefined"
lemma
assumes "trie'_inv t"
shows "trie'_α (ins' xs t) = ins xs (trie'_α t)"
"trie'_inv (ins' xs t)"
lemma ins'_correct: "trie'_inv t
⟹ (isin' (ins' xs t) ks ⟷ xs=ks ∨ isin' t ks) ∧ trie'_inv (ins' xs t)"
proof (prove)
goal (1 subgoal):
1. trie'_α (ins' xs t) = ex10_tmpl.ins xs (trie'_α t) &&& trie'_inv (ins' xs t)

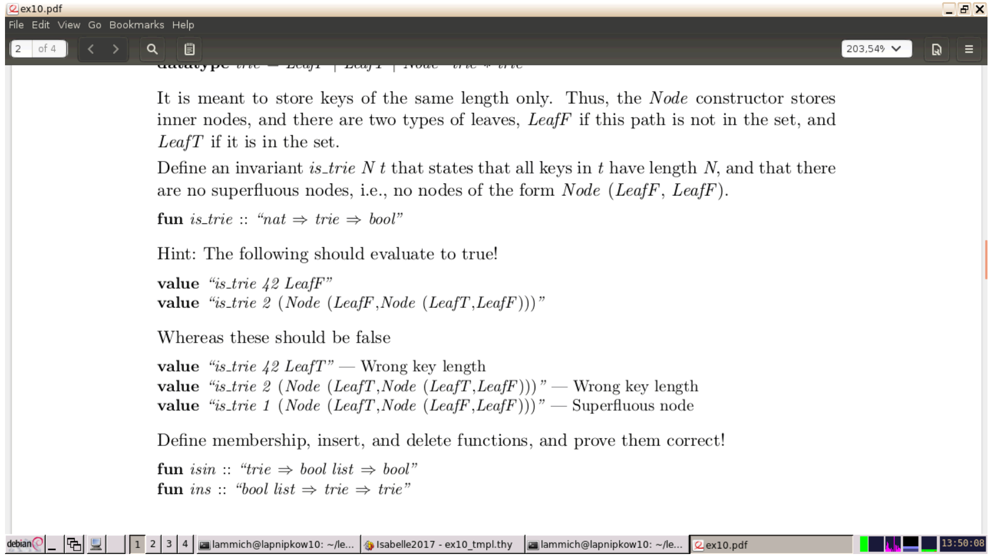

Isabelle2017 - ex10_tmpl.thy
assumes "trie'_α t"
shows "trie'_α (ins' xs t) = ins xs (trie'_α t)"
"trie'_inv (ins' xs t)"
using assms apply (induction xs t rule: ins'.induct)
apply (auto split: option.splits)
lemma ins'_correct: "trie'_inv t
proof (prove)
goal (2 subgoals):
1. ⋀k ks m x2.
⟦⋀x2a. ⟦x2 = x2a; trie'_inv x2a⟧ ⟹ trie'_α (ins' ks x2a) = ex10_tmpl.ins ks (trie'_α x2a);
⋀x2a. ⟦x2 = x2a; trie'_inv x2a⟧ ⟹ trie'_inv (ins' ks x2a); bal m; sorted1 (inorder m);
∀x∈ran (lookup m). trie'_inv x; lookup m k = Some x2⟧
⟹ (map_option trie'_α ∘ lookup m)(k ↦ trie'_α (ins' ks x2)) = (map_option trie'_α ∘ lookup m)(k ↦
ex10_tmpl.ins ks (trie'_α x2))
2. ⋀k ks m x2 x.
⟦⋀x2a. ⟦x2 = x2a; trie'_inv x2a⟧ ⟹ trie'_α (ins' ks x2a) = ex10_tmpl.ins ks (trie'_α x2a);

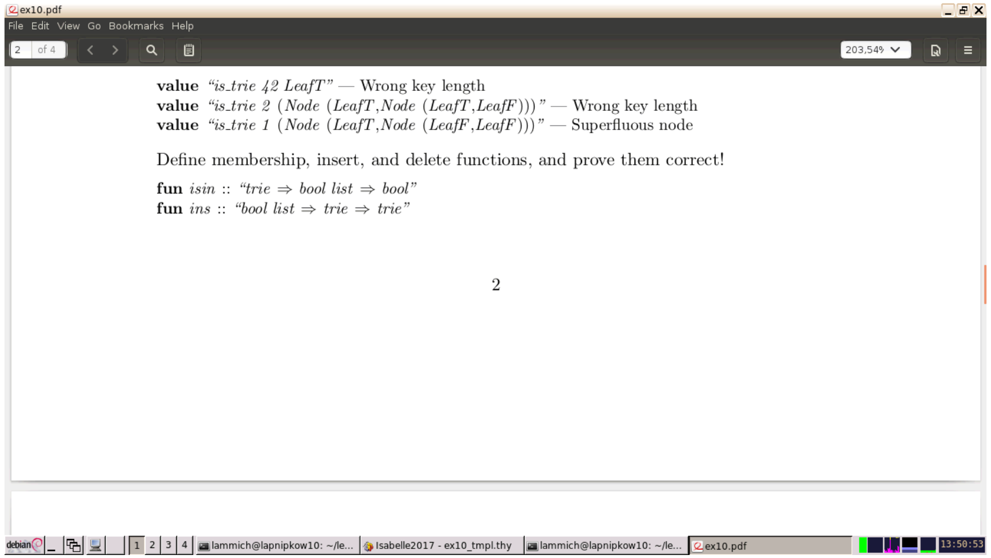

Isabelle2017 - ex10_tmpl.thy (modified)
shows "trie'_α (ins' xs t) = ins xs (trie'_α t)"
"trie'_inv (ins' xs t)"
using assms apply (induction xs t rule: ins'.induct)
apply (auto split: option.splits intro!: ext)
lemma ins'_correct: "trie'_inv t
proof (prove)
goal (2 subgoals):
1. ⋀k ks m x2.
⟦⋀x2a. ⟦x2 = x2a; trie'_inv x2a⟧ ⟹ trie'_α (ins' ks x2a) = ex10_tmpl.ins ks (trie'_α x2a);
⋀x2a. ⟦x2 = x2a; trie'_inv x2a⟧ ⟹ trie'_inv (ins' ks x2a); bal m; sorted1 (inorder m);
∀x∈ran (lookup m). trie'_inv x; lookup m k = Some x2⟧
⟹ trie'_α (ins' ks x2) = ex10_tmpl.ins ks (trie'_α x2)
2. ⋀k ks m x2 x.
⟦⋀x2a. ⟦x2 = x2a; trie'_inv x2a⟧ ⟹ trie'_α (ins' ks x2a) = ex10_tmpl.ins ks (trie'_α x2a);
⋀x2a. ⟦x2 = x2a; trie'_inv x2a⟧ ⟹ trie'_inv (ins' ks x2a); bal m; sorted1 (inorder m);

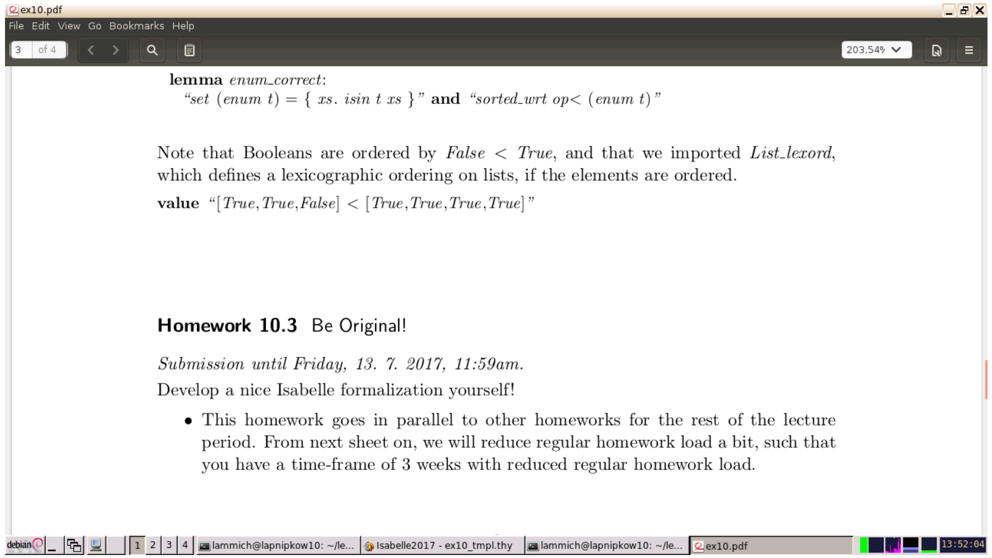

Isabelle2017 - ex10_tmpl.thy (modified)
"trie'_inv (ins' xs t)"
using assms apply (induction xs t rule: ins'.induct)
apply (auto split: option.splits simp: ran_def intro!: ext)
thm ext
done
lemma ins'_correct: "trie'_inv t
(⋀x. ?f x = ?g x) ⟹ ?f = ?g


Isabelle2017 - ex10_tmpl.thy (modified)
"trie'_inv (ins' xs t)"
using assms apply (induction xs t rule: ins'.induct)
apply (auto split: option.splits simp: ran_def intro!: ext)
thm ext
done
lemma ins'_correct: "trie'_inv t
(⋀x. ?f x = ?g x) ⟹ ?f = ?g


lammich@lapnipkow10: ~/lehre/FDS/SS18/public/exercises
lammich@lapnipkow10:~/opt/ttt$ cd ~/lehre/FDS/SS1
ex10.pdf
Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Dr. Peter Lammich
SS 2018
15. 6. 2018
Functional Data Structures
Exercise Sheet 10
Exercise 10.1 Tries with 2-3-trees
h@lapnipkow10: ~/lehre/FDS/SS18/public
@lapnipkow10:~/opt/ttt$ cd ~/lehre/FDS/
/ SS17/ SS18/ web/
@lapnipkow10:~/opt/ttt$ cd ~/lehre/FDS/SS18/public/
@lapnipkow10:~/lehre/FDS/SS18/public$ evince exercises/ex10.pdf
13:48:46


ex10.pdf
It is meant to store keys of the same length only. Thus, the Node constructor stores inner nodes, and there are two types of leaves, LeafF if this path is not in the set, and LeafT if it is in the set.
Define an invariant is_trie N t that states that all keys in t have length N, and that there are no superfluous nodes, i.e., no nodes of the form Node (LeafF, LeafF).
fun is_trie :: "nat ⇒ trie ⇒ bool"
Hint: The following should evaluate to true!
value "is_trie 42 LeafF"
value "is_trie 2 (Node (LeafF,Node (LeafT,LeafF)))"
Whereas these should be false
value "is_trie 42 LeafT" — Wrong key length
value "is_trie 2 (Node (LeafT,Node (LeafT,LeafF)))" — Wrong key length
value "is_trie 1 (Node (LeafT,Node (LeafF,LeafF)))" — Superfluous node
Define membership, insert, and delete functions, and prove them correct!
fun isin :: "trie ⇒ bool list ⇒ bool"
fun ins :: "bool list ⇒ trie ⇒ trie"
13:50:08


ex10.pdf
value "is_trie 42 LeafT" — Wrong key length
value "is_trie 2 (Node (LeafT,Node (LeafT,LeafF)))" — Wrong key length
value "is_trie 1 (Node (LeafT,Node (LeafF,LeafF)))" — Superfluous node
Define membership, insert, and delete functions, and prove them correct!
fun isin :: "trie ⇒ bool list ⇒ bool"
fun ins :: "bool list ⇒ trie ⇒ trie"
2
13:50:53


ex10.pdf
lemma enum_correct:
"set (enum t) = { xs. isin t xs }" and "sorted_wrt op< (enum t)"
Note that Booleans are ordered by False < True, and that we imported List_lexord, which defines a lexicographic ordering on lists, if the elements are ordered.
value "[True,True,False] < [True,True,True,True]"
Homework 10.3 Be Original!
Submission until Friday, 13. 7. 2017, 11:59am.
Develop a nice Isabelle formalization yourself!
This homework goes in parallel to other homeworks for the rest of the lecture period. From next sheet on, we will reduce regular homework load a bit, such that you have a time-frame of 3 weeks with reduced regular homework load.
13:52:04

Homework 10.3 Be Original!

Submission until Friday, 13. 7. 2017, 11:59am.

Develop a nice Isabelle formalization yourself!

- This homework goes in parallel to other homeworks for the rest of the lecture period. From next sheet on, we will reduce regular homework load a bit, such that you have a time-frame of 3 weeks with reduced regular homework load.

- This homework will yield 15 points (for minimal solutions). Additionally, up to 15 bonus points may be awarded for particularly nice/original/etc solutions.
- You may develop a formalization from all areas, not only functional data structures.
- Document your solution, such that it is clear what you have formalized and what your main theorems state!
- Set yourself a time frame and some intermediate/minimal goals. Your formalization needs not be universal and complete after 3 weeks.
- You are welcome to discuss the realizability of your project with the tutor!
- In case you should need inspiration to find a project: Sparse matrices, skew binary numbers, arbitrary precision arithmetic (on lists of bits), interval data structures (e.g. interval lists), spatial data structures (quad-trees, oct-trees), Fibonacci heaps, prefix tries/arrays and BWT, etc.

- Document your solution, such that it is clear what you have formalized and what your main theorems state!
- Set yourself a time frame and some intermediate/minimal goals. Your formalization needs not be universal and complete after 3 weeks.
- You are welcome to discuss the realizability of your project with the tutor!
- In case you should need inspiration to find a project: Sparse matrices, skew binary numbers, arbitrary precision arithmetic (on lists of bits), interval data structures (e.g. interval lists), spatial data structures (quad-trees, oct-trees), Fibonacci heaps, prefix tries/arrays and BWT, etc.