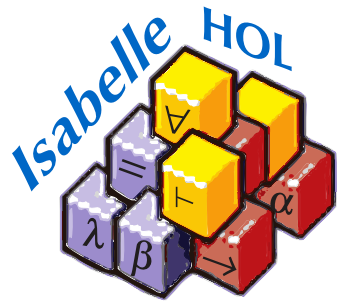
Truly Modular (Co)datatypes

for



Jasmin Blanchette
Lorenz Panny
Dmitriy Traytel

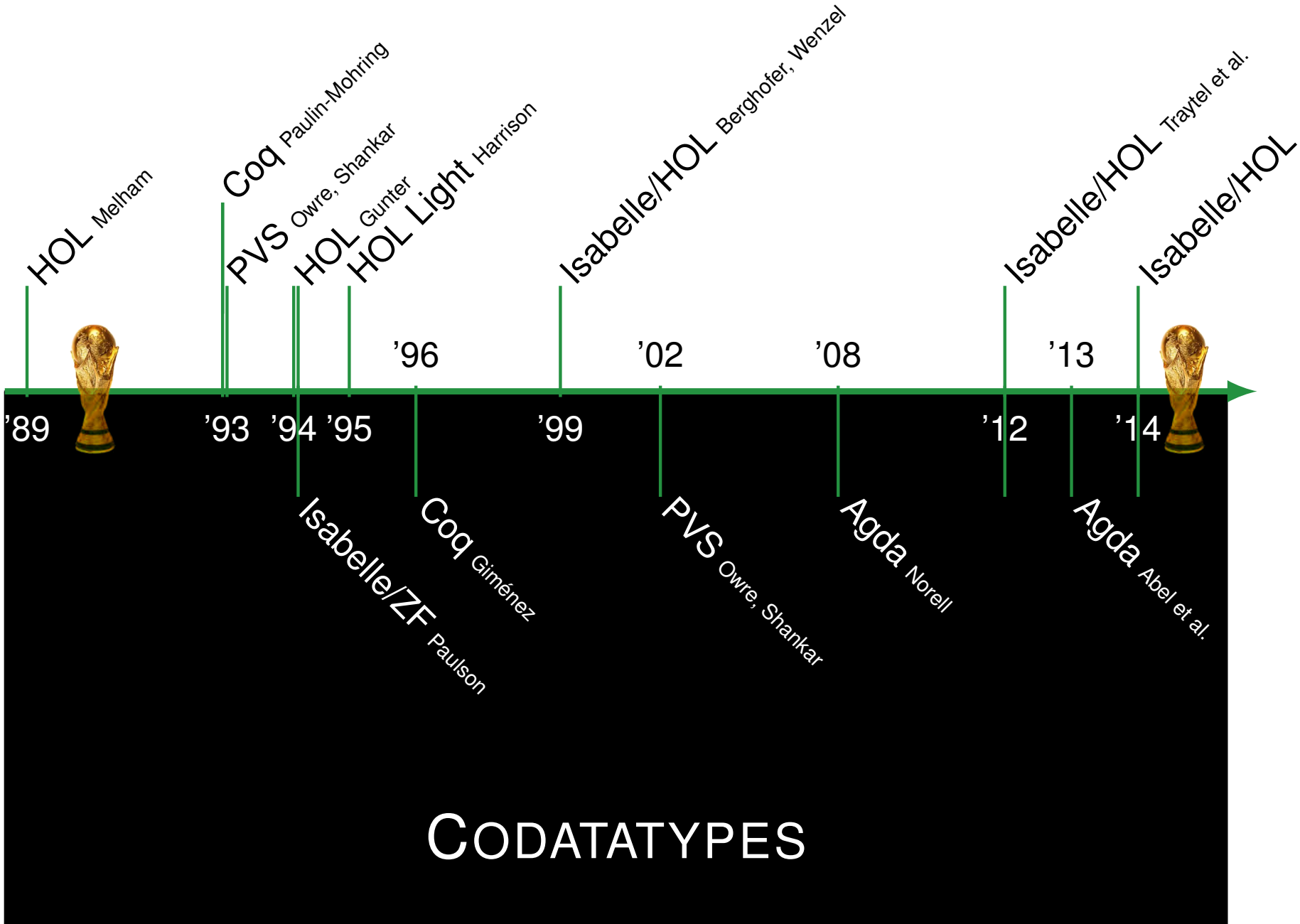
Johannes Hölzl
Andrei Popescu

Technische Universität München

Andreas Lochbihler

ETH zürich

DATATYPES



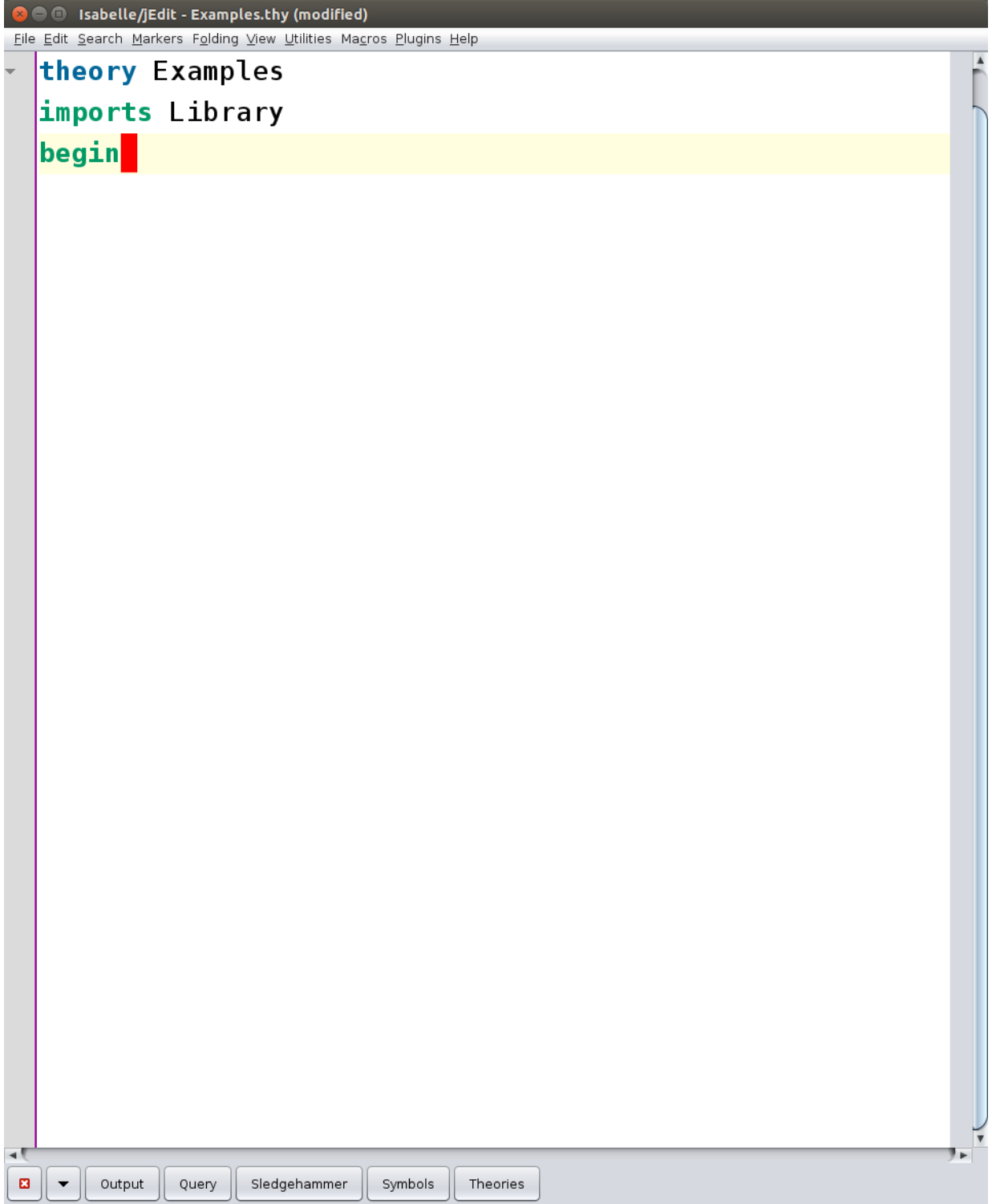
CODATATYPES

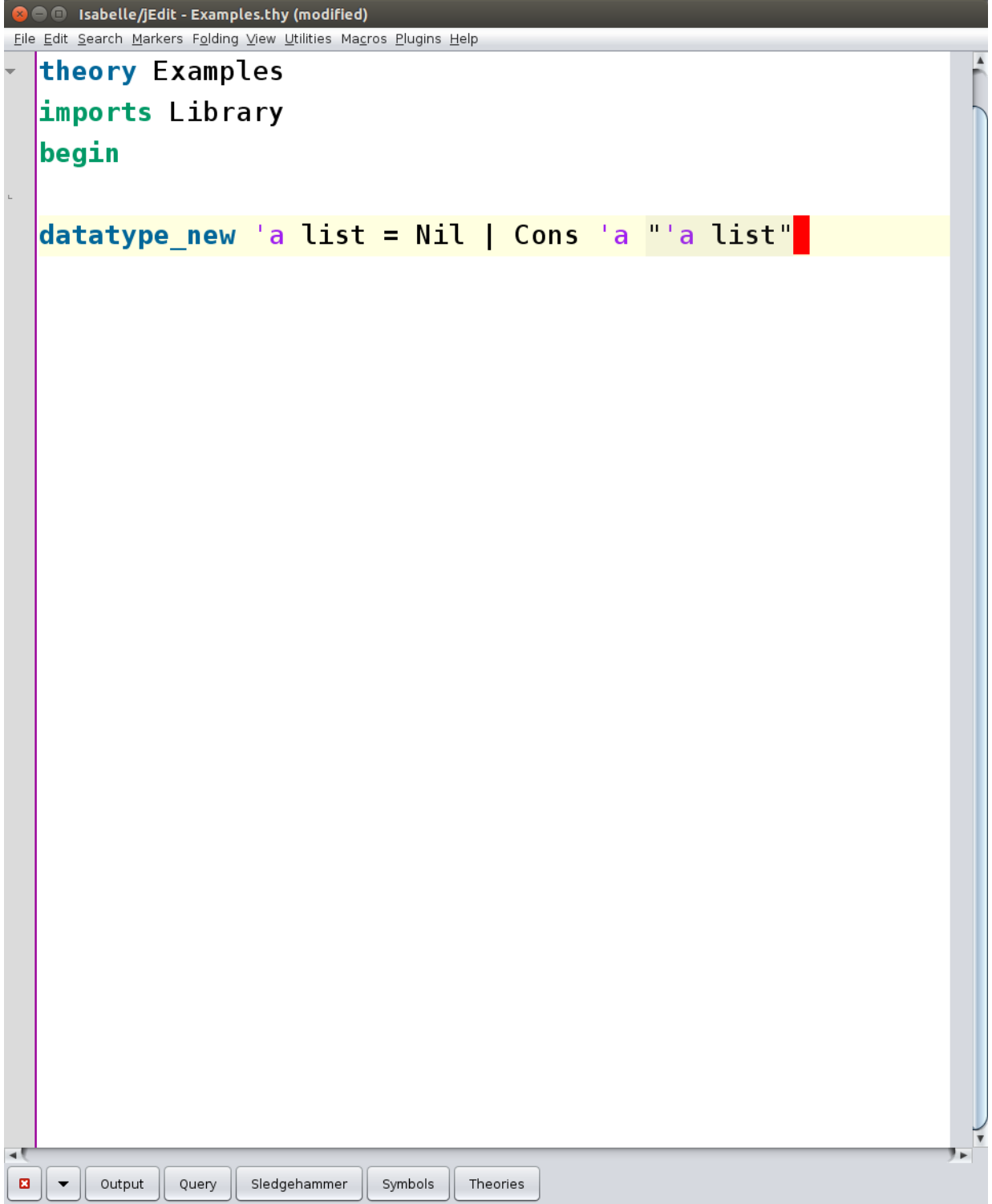
Disclaimer

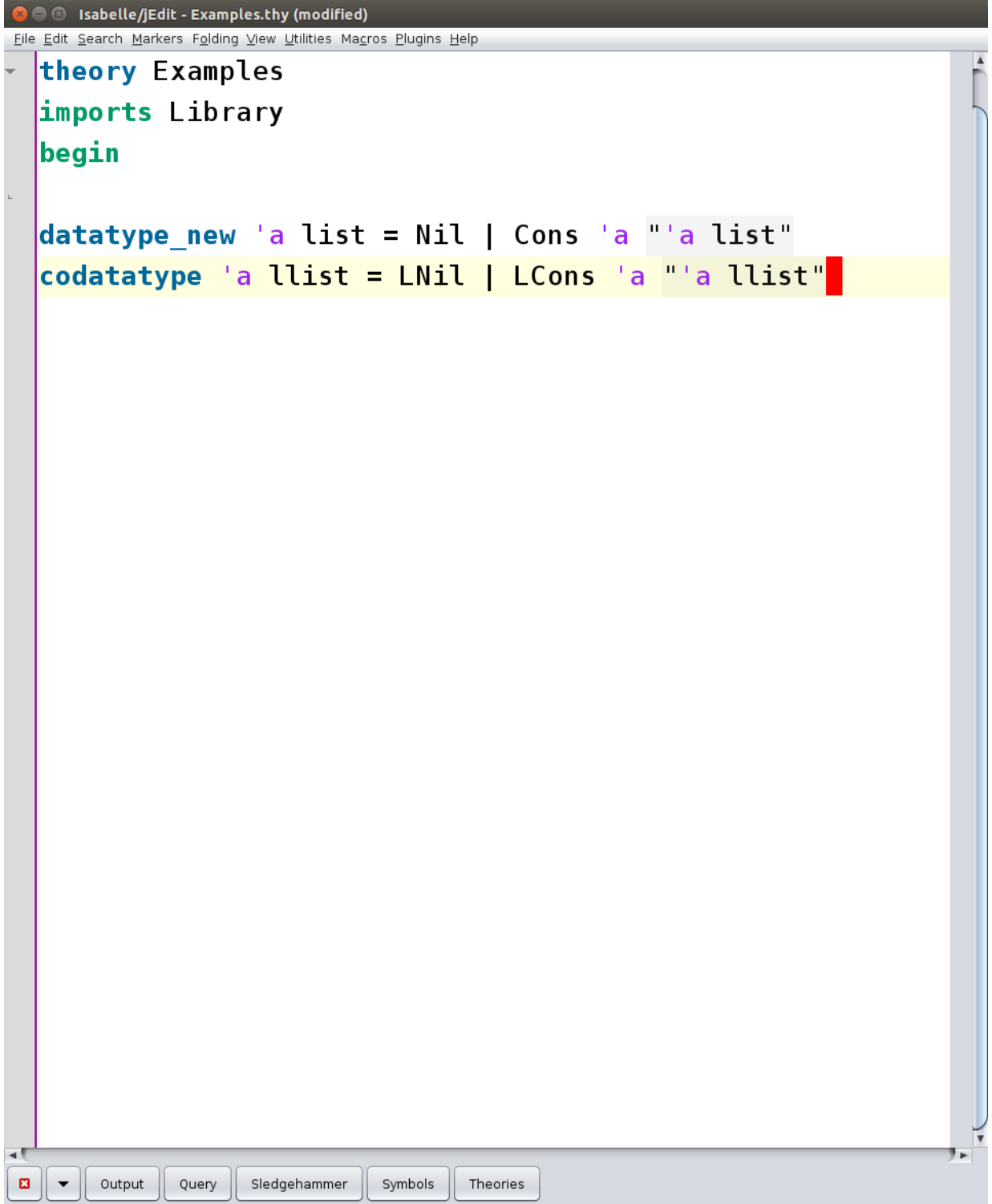
Isabelle/HOL was **not** harmed

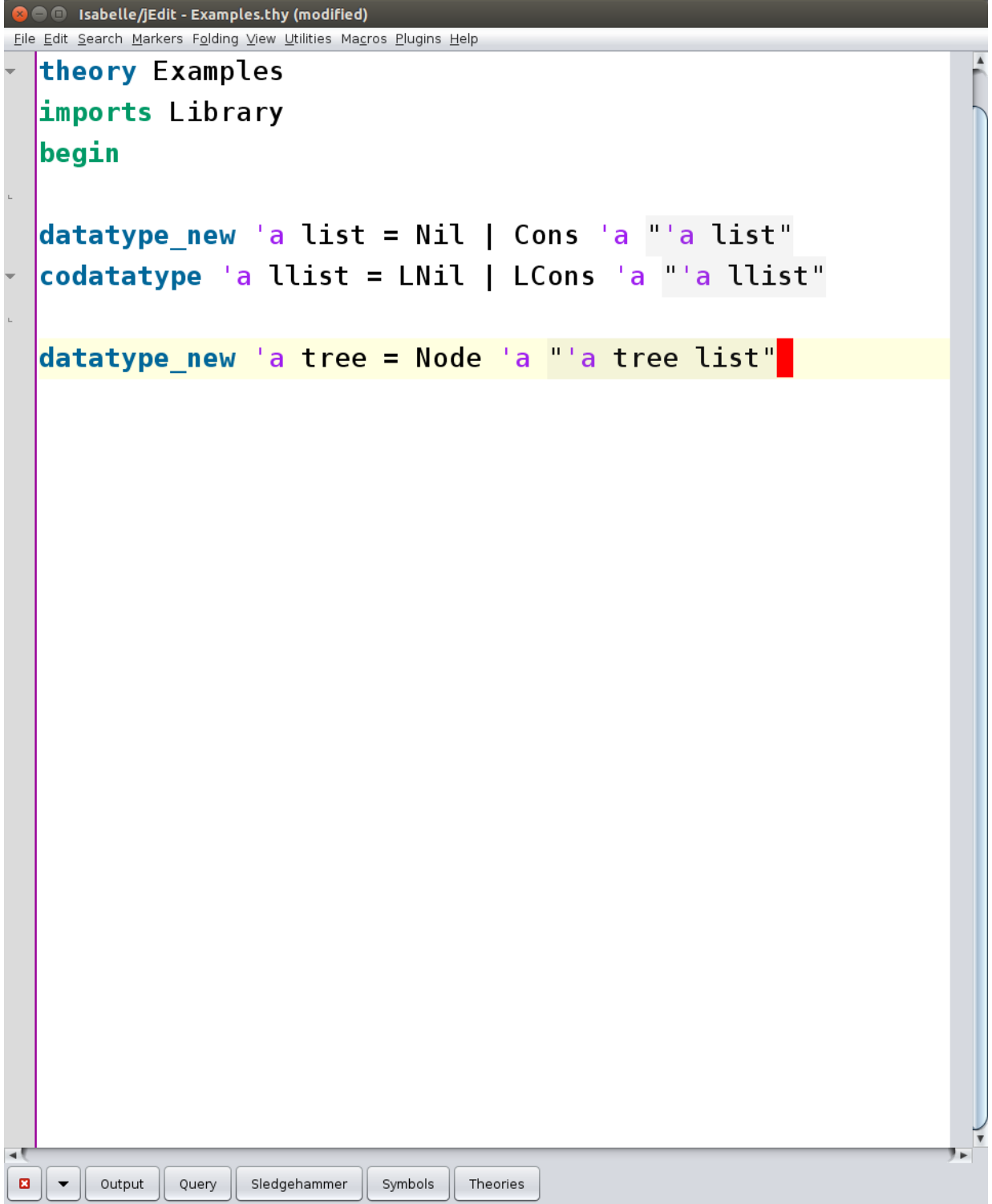
by the introduction of new axioms

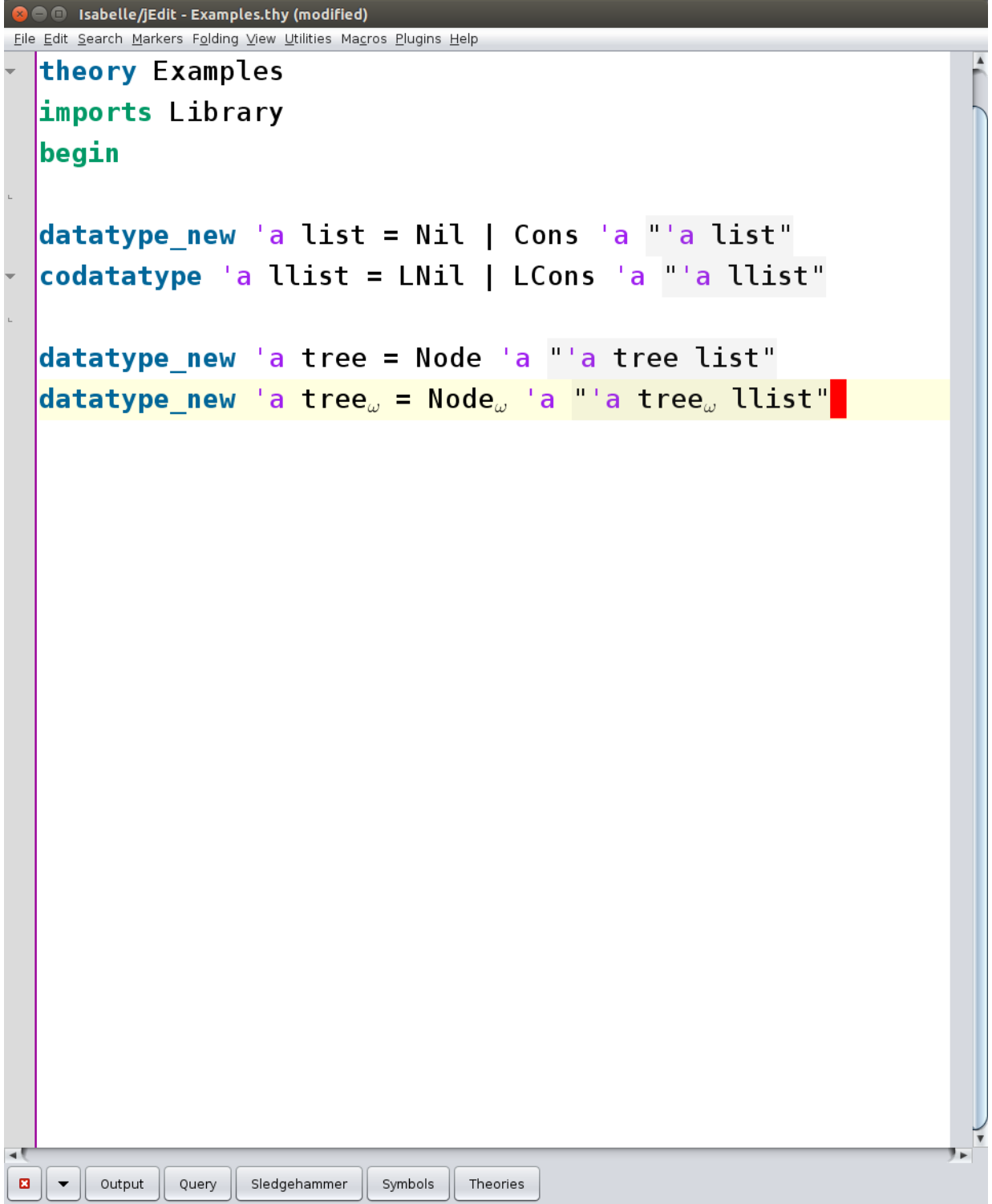
to achieve the following

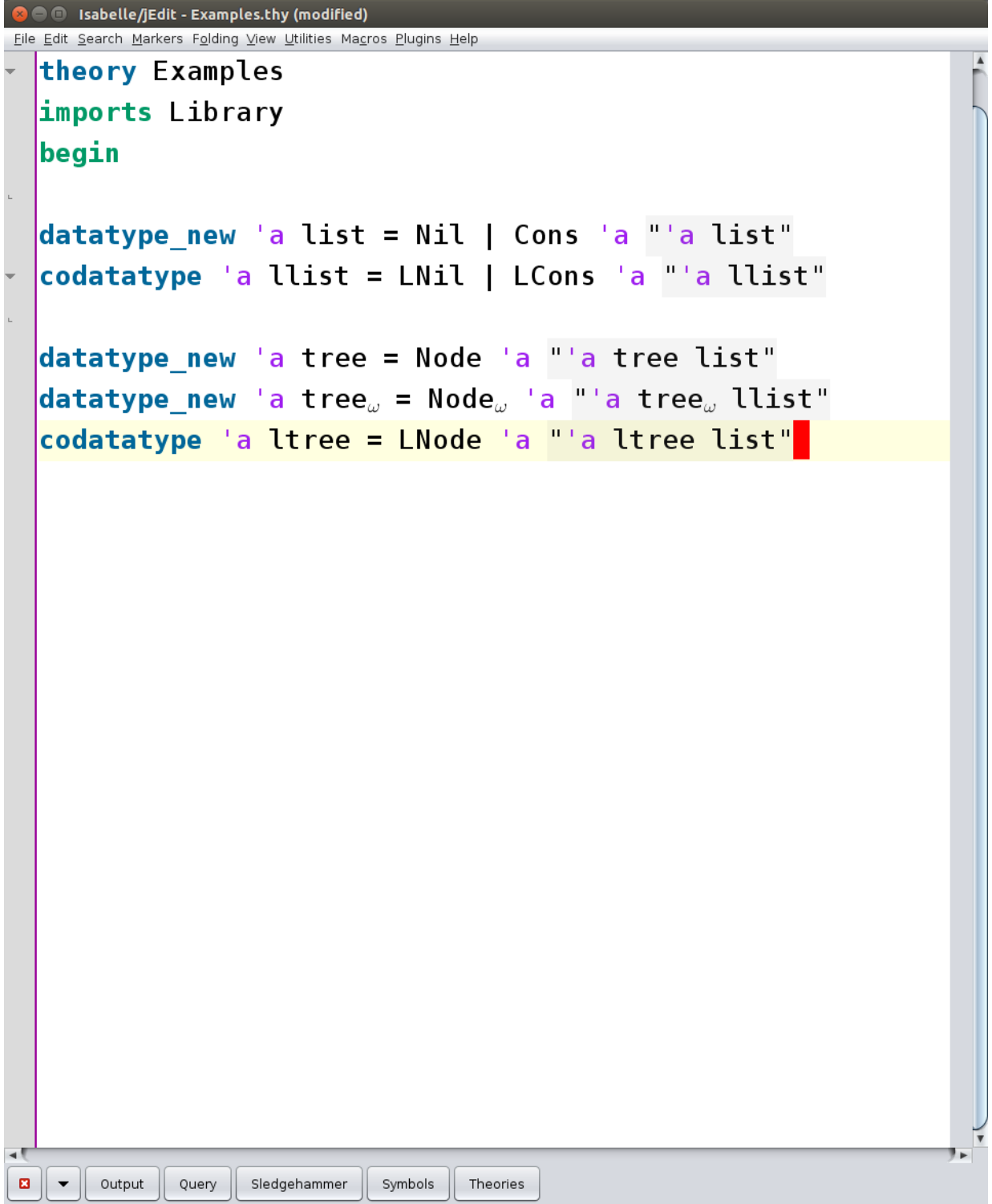

Isabelle/jEdit - Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Examples
imports Library
begin
Output
Query
Sledgehammer
Symbols
Theories

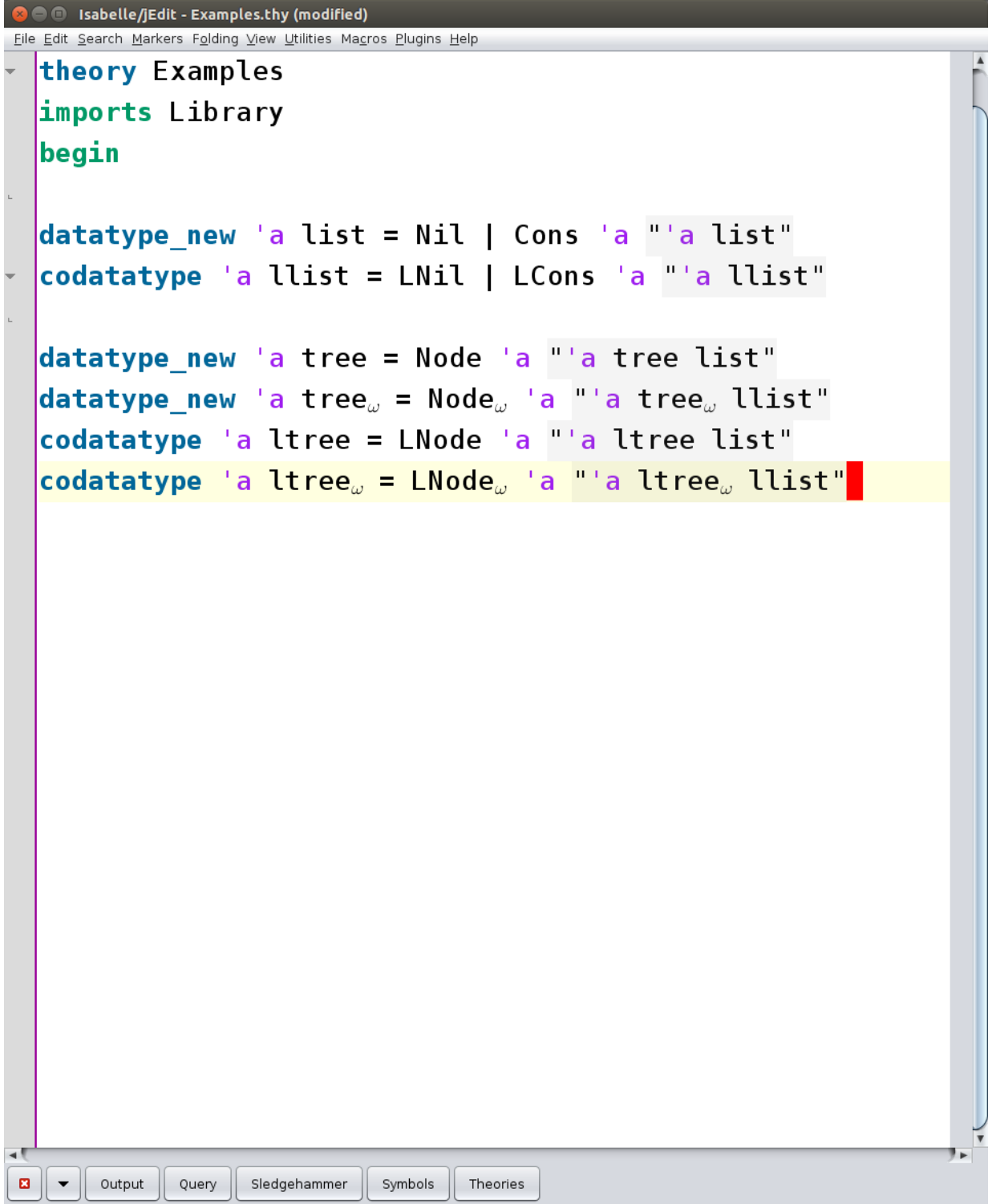

Isabelle/jEdit - Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Examples
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
Output
Query
Sledgehammer
Symbols
Theories

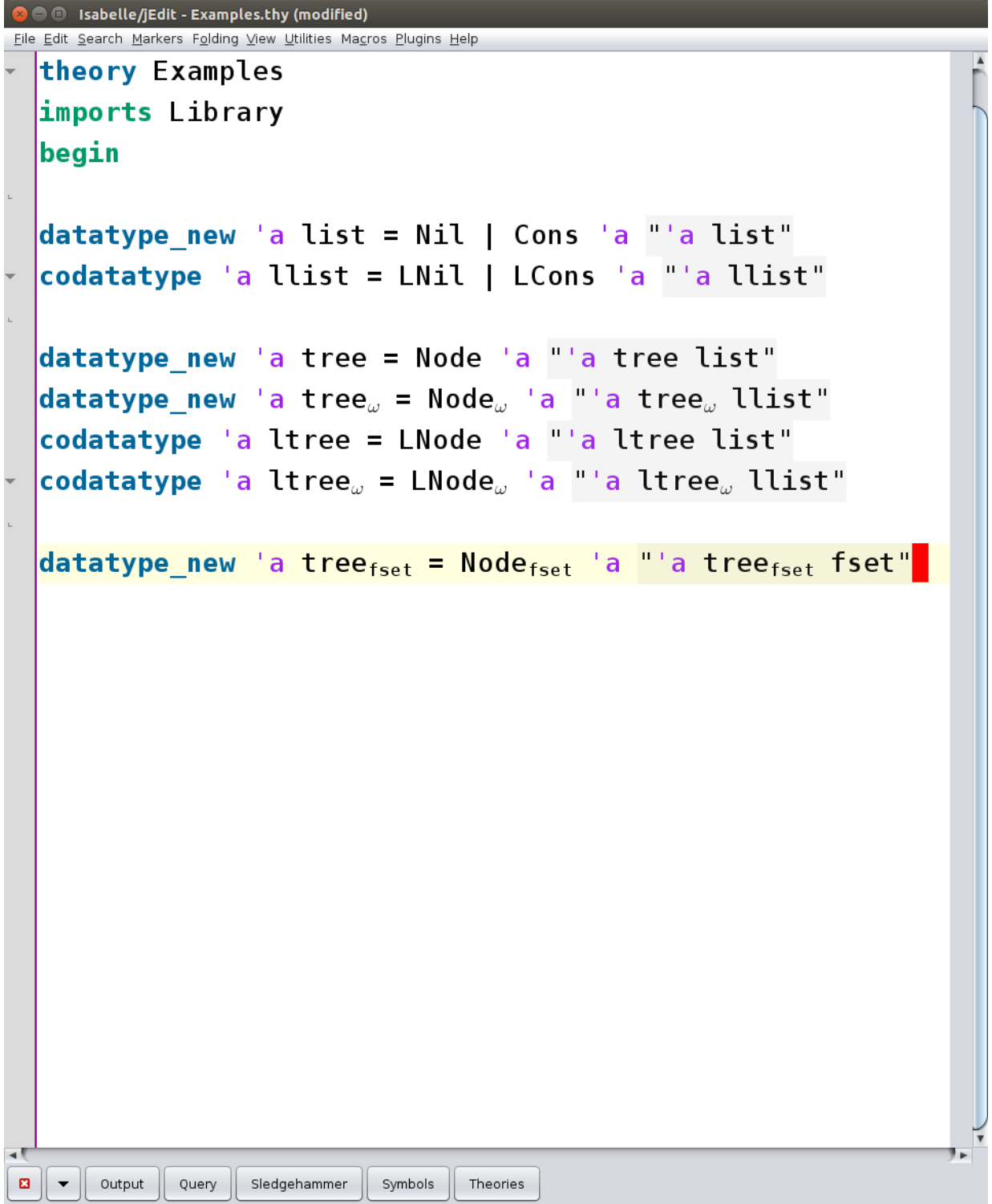

Isabelle/jEdit - Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Examples
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
codatatype 'a llist = LNil | LCons 'a "'a llist"
Output
Query
Sledgehammer
Symbols
Theories

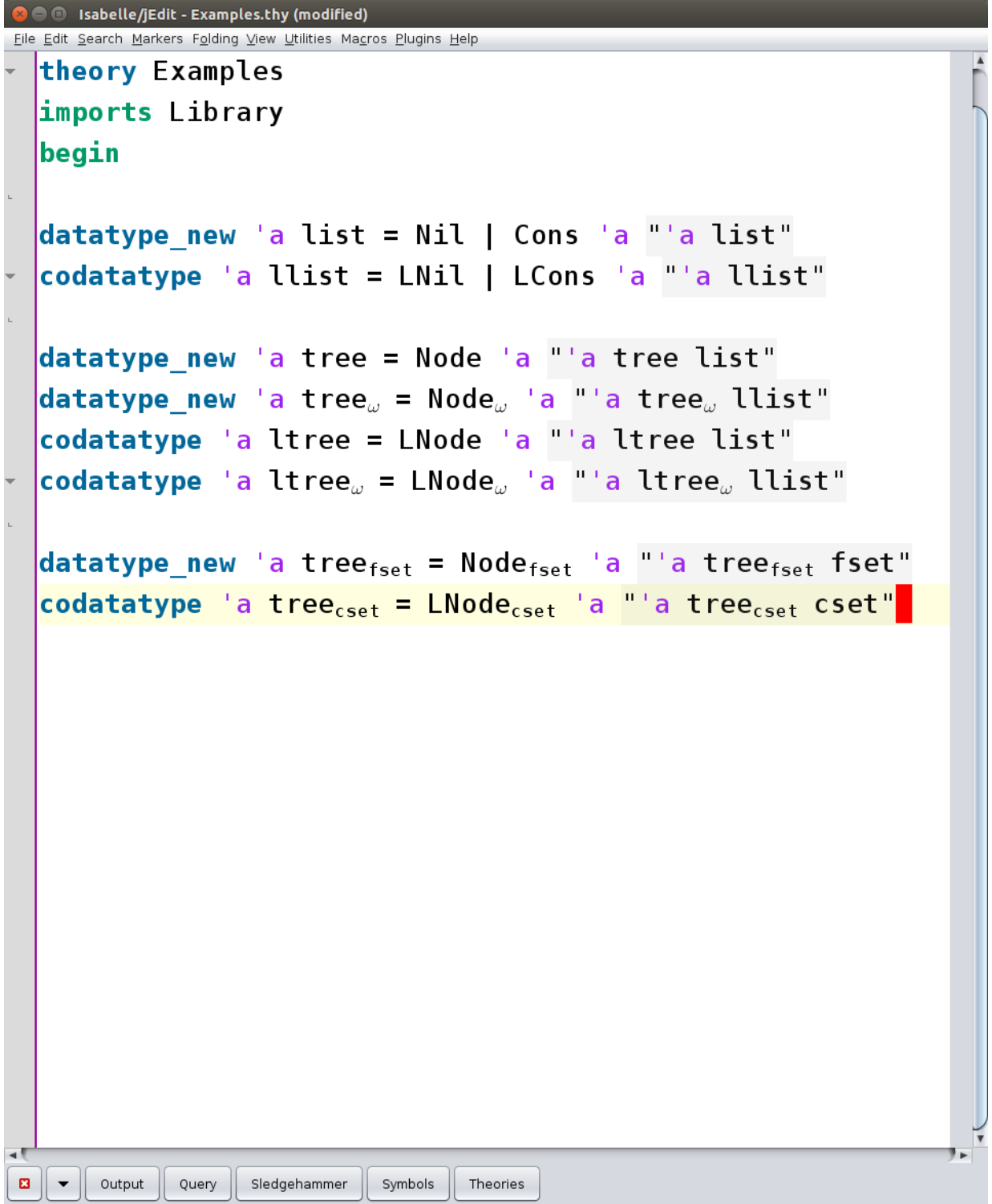

Isabelle/jEdit - Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Examples
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
codatatype 'a llist = LNil | LCons 'a "'a llist"
datatype_new 'a tree = Node 'a "'a tree list"
Output
Query
Sledgehammer
Symbols
Theories

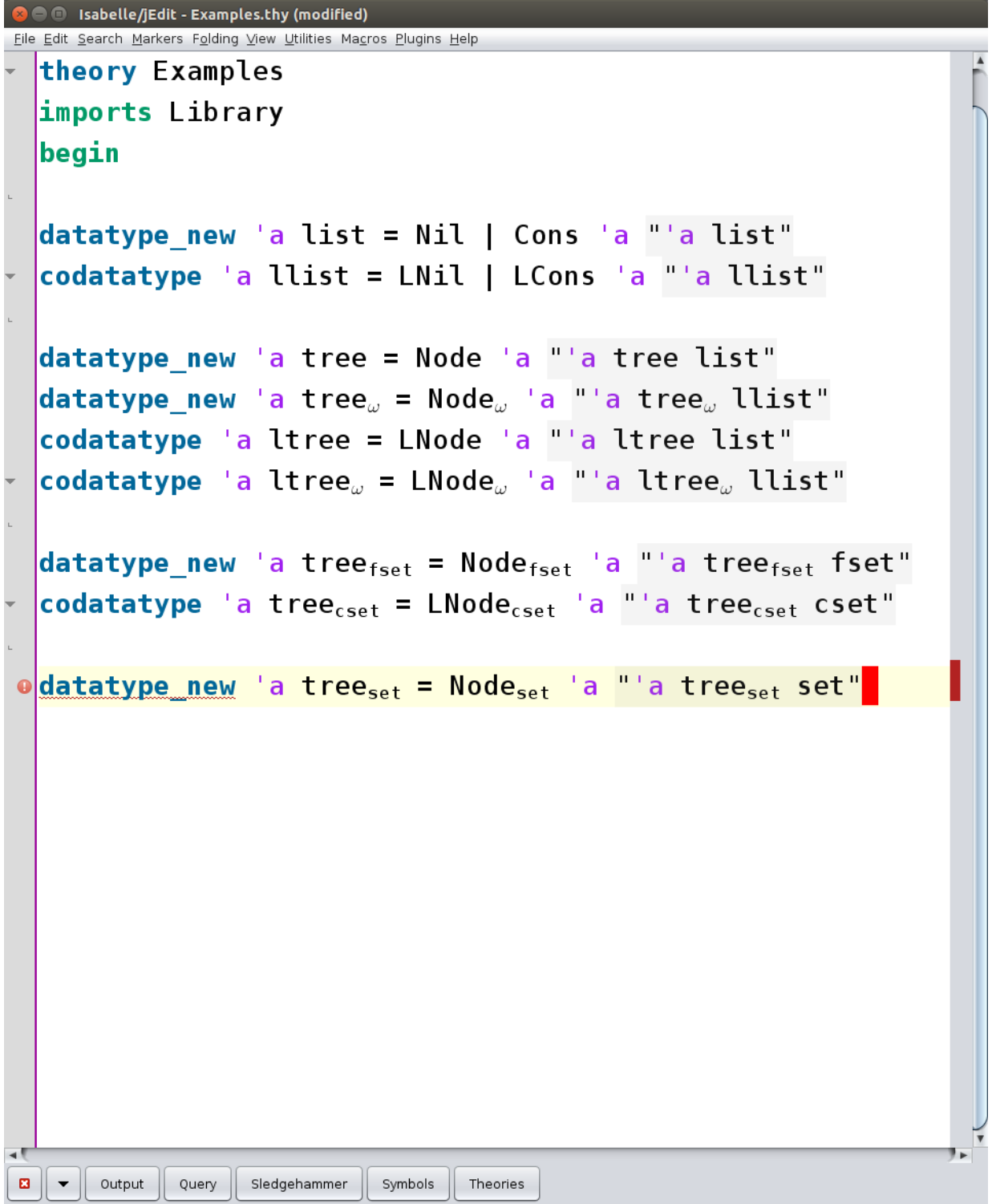

Isabelle/jEdit - Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Examples
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
codatatype 'a llist = LNil | LCons 'a "'a llist"
datatype_new 'a tree = Node 'a "'a tree list"
datatype_new 'a tree$_\omega$ = Node$_\omega$ 'a "'a tree$_\omega$ llist"
Output
Query
Sledgehammer
Symbols
Theories

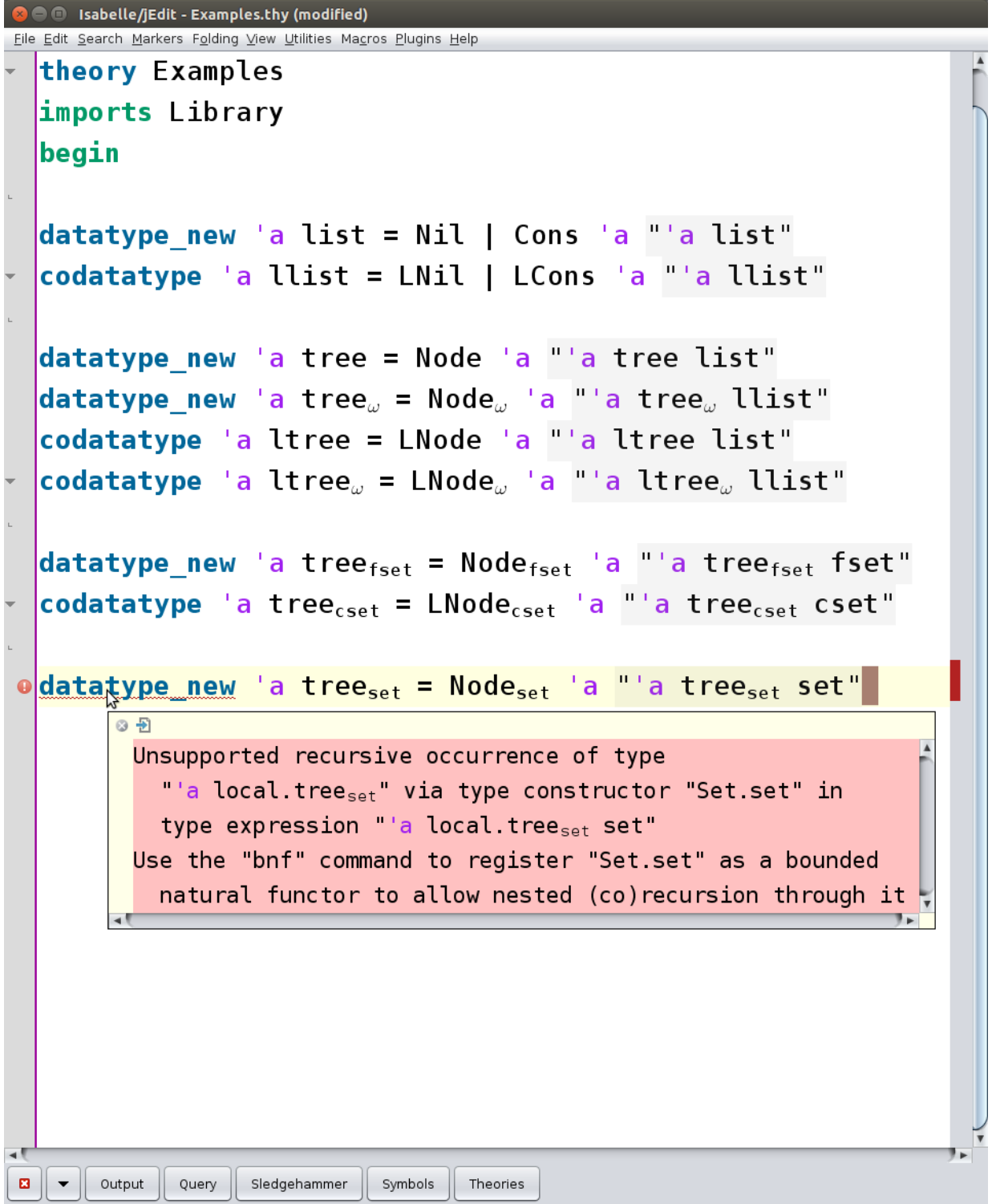

Isabelle/jEdit - Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Examples
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
codatatype 'a llist = LNil | LCons 'a "'a llist"
datatype_new 'a tree = Node 'a "'a tree list"
datatype_new 'a tree$_\omega$ = Node$_\omega$ 'a "'a tree$_\omega$ llist"
codatatype 'a ltree = LNode 'a "'a ltree list"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Examples
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
codatatype 'a llist = LNil | LCons 'a "'a llist"
datatype_new 'a tree = Node 'a "'a tree list"
datatype_new 'a tree$_\omega$ = Node$_\omega$ 'a "'a tree$_\omega$ llist"
codatatype 'a ltree = LNode 'a "'a ltree list"
codatatype 'a ltree$_\omega$ = LNode$_\omega$ 'a "'a ltree$_\omega$ llist"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Examples
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
codatatype 'a llist = LNil | LCons 'a "'a llist"
datatype_new 'a tree = Node 'a "'a tree list"
datatype_new 'a tree$_\omega$ = Node$_\omega$ 'a "'a tree$_\omega$ llist"
codatatype 'a ltree = LNode 'a "'a ltree list"
codatatype 'a ltree$_\omega$ = LNode$_\omega$ 'a "'a ltree$_\omega$ llist"
datatype_new 'a tree$_{fset}$ = Node$_{fset}$ 'a "'a tree$_{fset}$ fset"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Examples
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
codatatype 'a llist = LNil | LCons 'a "'a llist"
datatype_new 'a tree = Node 'a "'a tree list"
datatype_new 'a tree$_\omega$ = Node$_\omega$ 'a "'a tree$_\omega$ llist"
codatatype 'a ltree = LNode 'a "'a ltree list"
codatatype 'a ltree$_\omega$ = LNode$_\omega$ 'a "'a ltree$_\omega$ llist"
datatype_new 'a tree$_{fset}$ = Node$_{fset}$ 'a "'a tree$_{fset}$ fset"
codatatype 'a tree$_{cset}$ = LNode$_{cset}$ 'a "'a tree$_{cset}$ cset"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Examples
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
codatatype 'a llist = LNil | LCons 'a "'a llist"
datatype_new 'a tree = Node 'a "'a tree list"
datatype_new 'a tree$_\omega$ = Node$_\omega$ 'a "'a tree$_\omega$ llist"
codatatype 'a ltree = LNode 'a "'a ltree list"
codatatype 'a ltree$_\omega$ = LNode$_\omega$ 'a "'a ltree$_\omega$ llist"
datatype_new 'a tree$_{fset}$ = Node$_{fset}$ 'a "'a tree$_{fset}$ fset"
codatatype 'a tree$_{cset}$ = LNode$_{cset}$ 'a "'a tree$_{cset}$ cset"
datatype_new 'a tree$_{set}$ = Node$_{set}$ 'a "'a tree$_{set}$ set"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Examples.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Examples
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
codatatype 'a llist = LNil | LCons 'a "'a llist"
datatype_new 'a tree = Node 'a "'a tree list"
datatype_new 'a tree$_\omega$ = Node$_\omega$ 'a "'a tree$_\omega$ llist"
codatatype 'a ltree = LNode 'a "'a ltree list"
codatatype 'a ltree$_\omega$ = LNode$_\omega$ 'a "'a ltree$_\omega$ llist"
datatype_new 'a tree$_{fset}$ = Node$_{fset}$ 'a "'a tree$_{fset}$ fset"
codatatype 'a tree$_{cset}$ = LNode$_{cset}$ 'a "'a tree$_{cset}$ cset"
datatype_new 'a tree$_{set}$ = Node$_{set}$ 'a "'a tree$_{set}$ set"
Unsupported recursive occurrence of type
"'a local.tree$_{set}$" via type constructor "Set.set" in
type expression "'a local.tree$_{set}$ set"
Use the "bnf" command to register "Set.set" as a bounded
natural functor to allow nested (co)recursion through it
Output
Query
Sledgehammer
Symbols
Theories

Outline

Outline

BNF

=

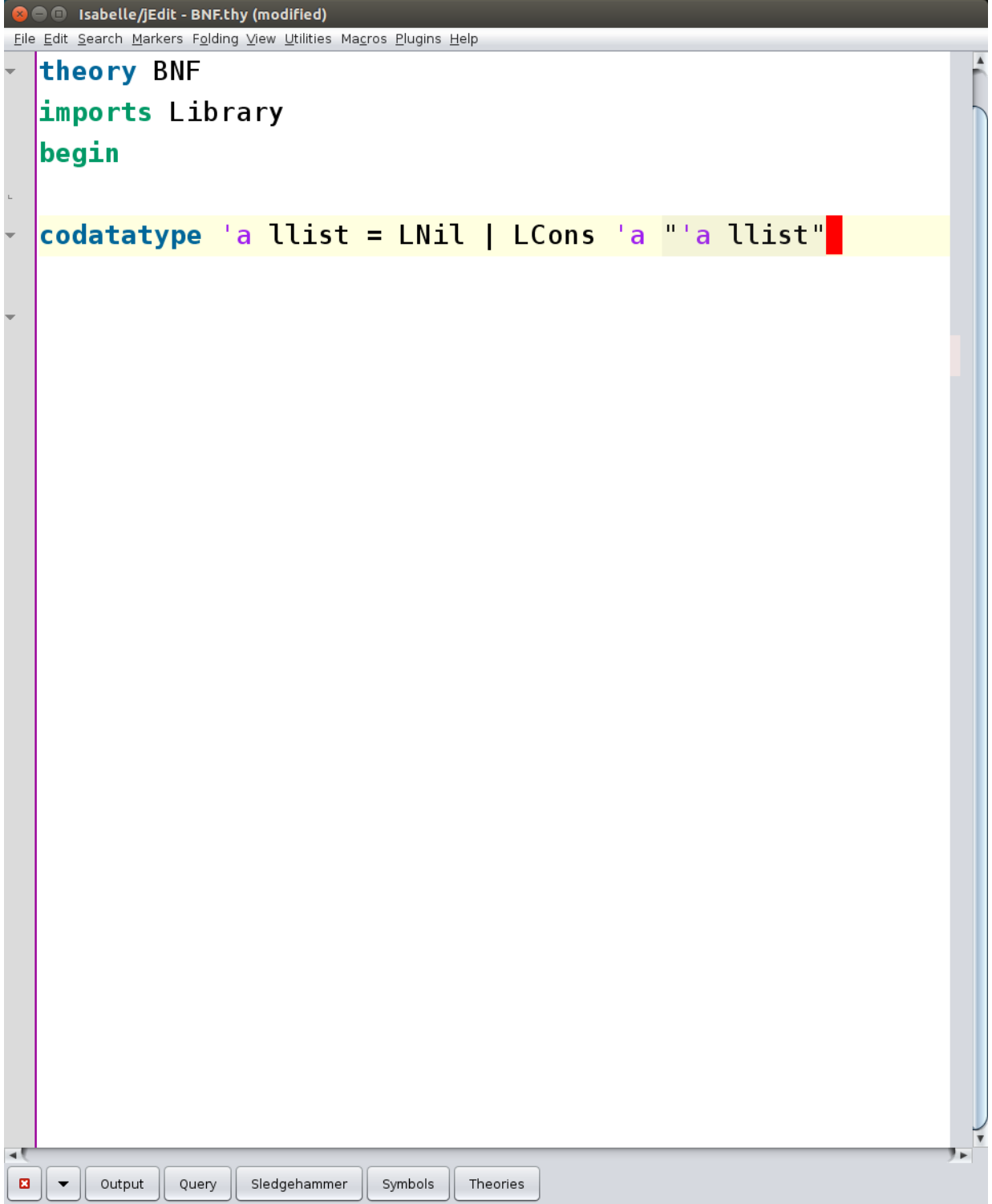
type

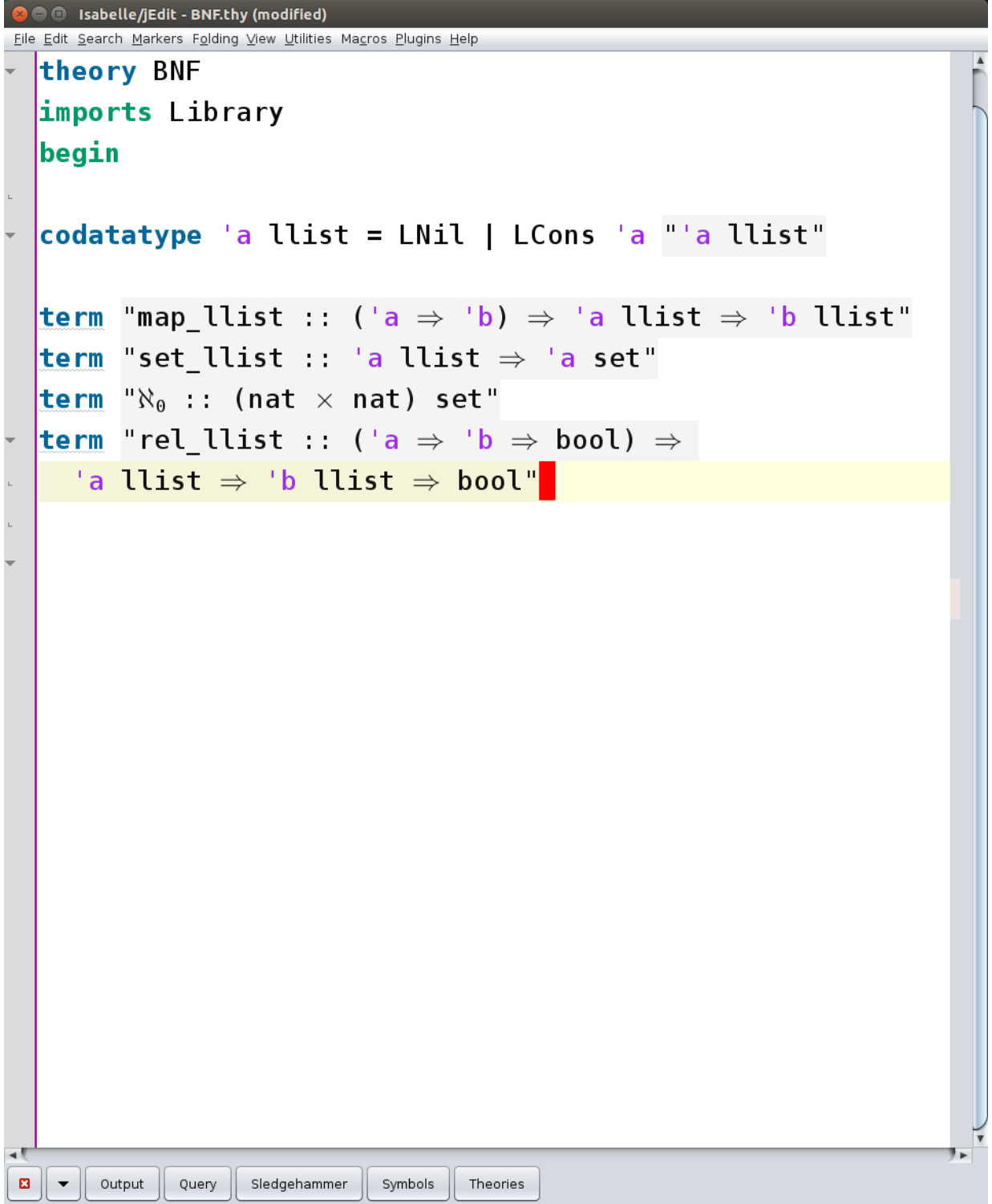
+

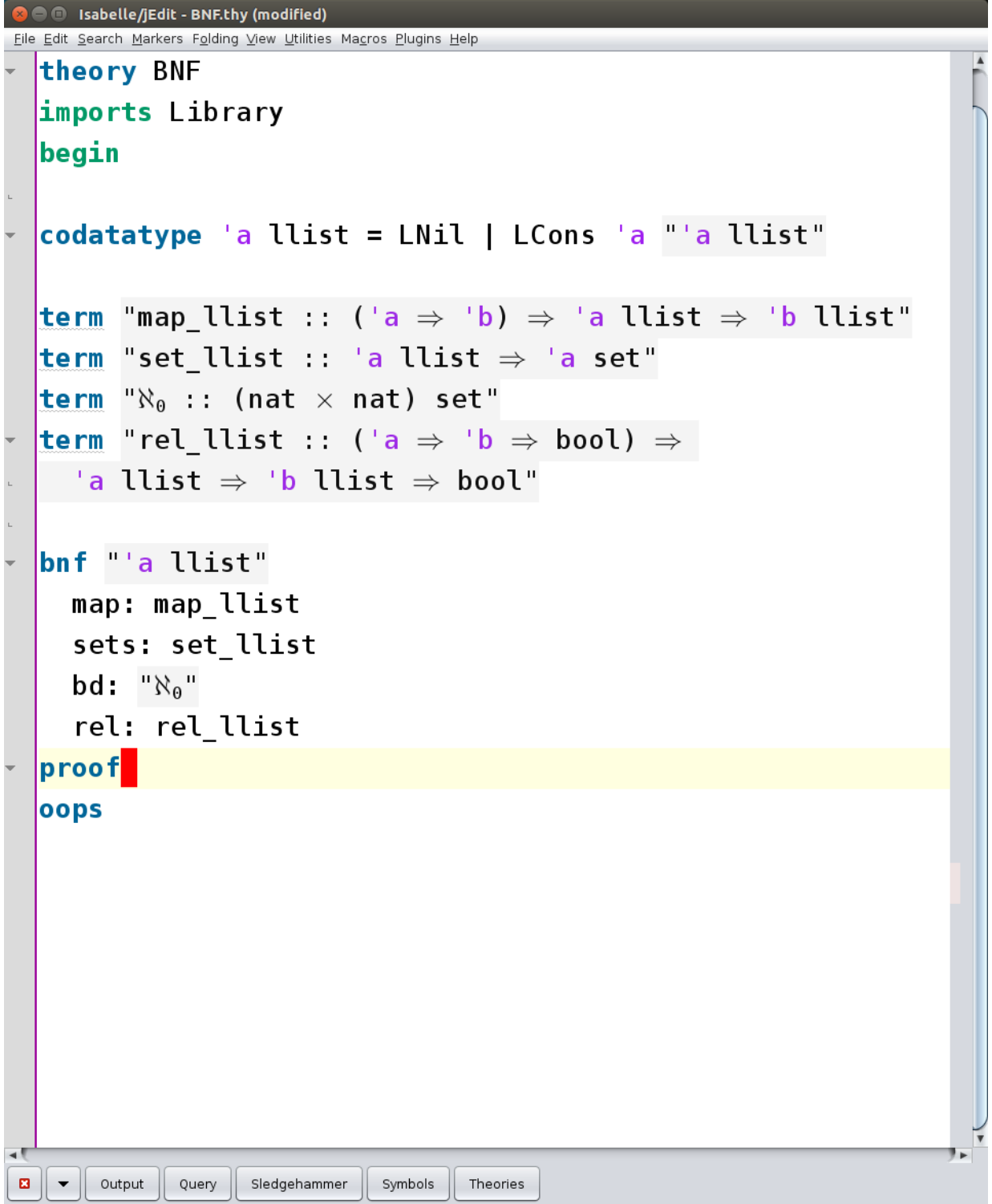
polymorphic constants
(map, set, bound, relator)

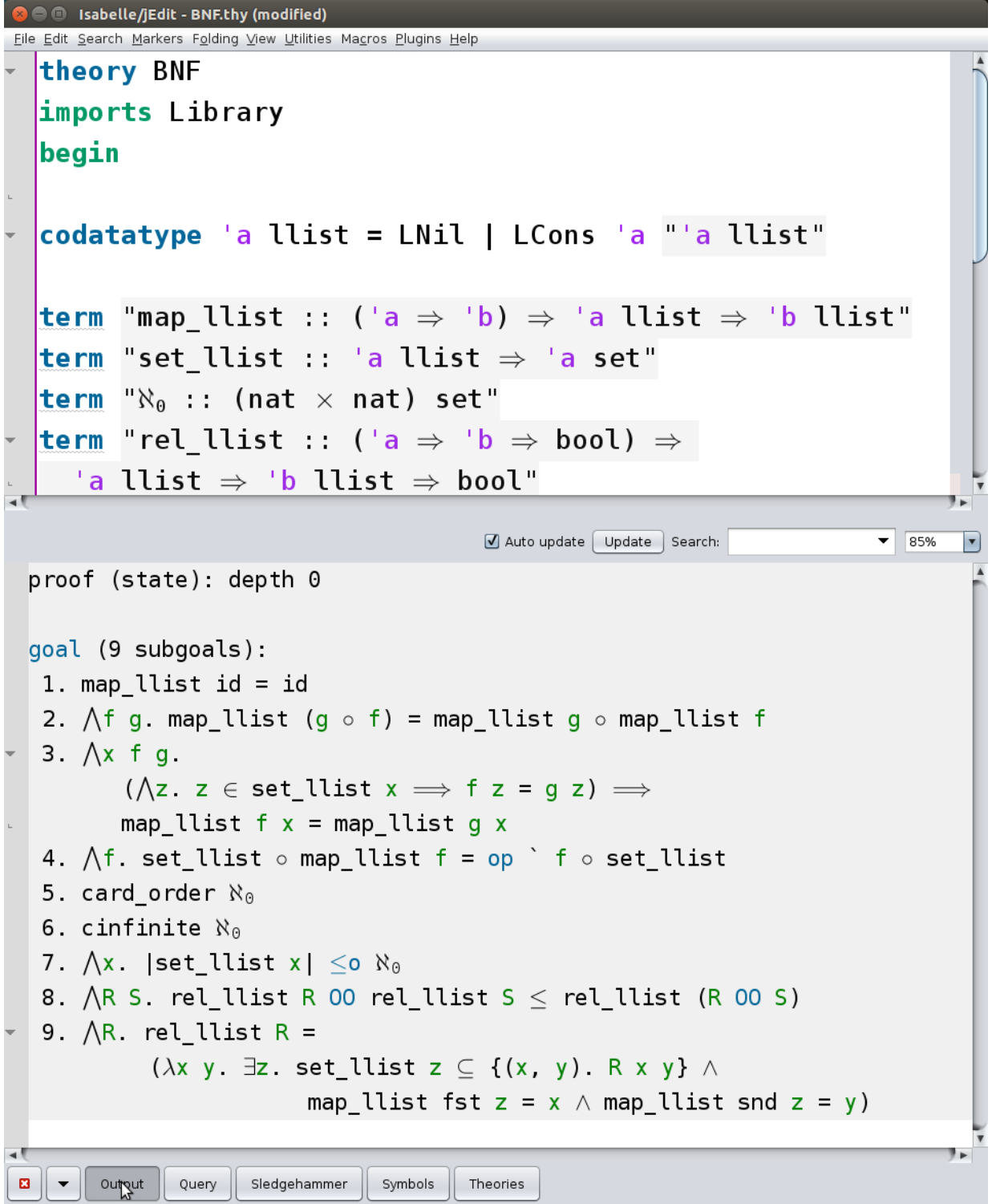
+

theorems


Isabelle/jEdit - BNF.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory BNF
imports Library
begin
codatatype 'a llist = LNil | LCons 'a "'a llist"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - BNF.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory BNF
imports Library
begin
codatatype 'a llist = LNil | LCons 'a "'a llist"
term "map_llist :: ('a ⇒ 'b) ⇒ 'a llist ⇒ 'b llist"
term "set_llist :: 'a llist ⇒ 'a set"
term "ℵ₀ :: (nat × nat) set"
term "rel_llist :: ('a ⇒ 'b ⇒ bool) ⇒
'a llist ⇒ 'b llist ⇒ bool"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - BNF.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory BNF
imports Library
begin
codatatype 'a llist = LNil | LCons 'a "'a llist"
term "map_llist :: ('a ⇒ 'b) ⇒ 'a llist ⇒ 'b llist"
term "set_llist :: 'a llist ⇒ 'a set"
term "ℵ₀ :: (nat × nat) set"
term "rel_llist :: ('a ⇒ 'b ⇒ bool) ⇒
'a llist ⇒ 'b llist ⇒ bool"
bnf "'a llist"
map: map_llist
sets: set_llist
bd: "ℵ₀"
rel: rel_llist
proof
oops
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - BNF.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory BNF
imports Library
begin
codatatype 'a llist = LNil | LCons 'a "'a llist"
term "map_llist :: ('a ⇒ 'b) ⇒ 'a llist ⇒ 'b llist"
term "set_llist :: 'a llist ⇒ 'a set"
term "ℵ₀ :: (nat × nat) set"
term "rel_llist :: ('a ⇒ 'b ⇒ bool) ⇒
'a llist ⇒ 'b llist ⇒ bool"
Auto update
Update
Search:
85%
proof (state): depth 0
goal (9 subgoals):
1. map_llist id = id
2. ⋀f g. map_llist (g ∘ f) = map_llist g ∘ map_llist f
3. ⋀x f g.
(⋀z. z ∈ set_llist x ⟹ f z = g z) ⟹
map_llist f x = map_llist g x
4. ⋀f. set_llist ∘ map_llist f = op ` f ∘ set_llist
5. card_order ℵ₀
6. cinfinite ℵ₀
7. ⋀x. |set_llist x| ≤o ℵ₀
8. ⋀R S. rel_llist R OO rel_llist S ≤ rel_llist (R OO S)
9. ⋀R. rel_llist R =
(λx y. ∃z. set_llist z ⊆ {(x, y). R x y} ∧
map_llist fst z = x ∧ map_llist snd z = y)
Output
Query
Sledgehammer
Symbols
Theories

BNF

=

a semantic criterion
for legal rhs of a
(co)datatype declaration

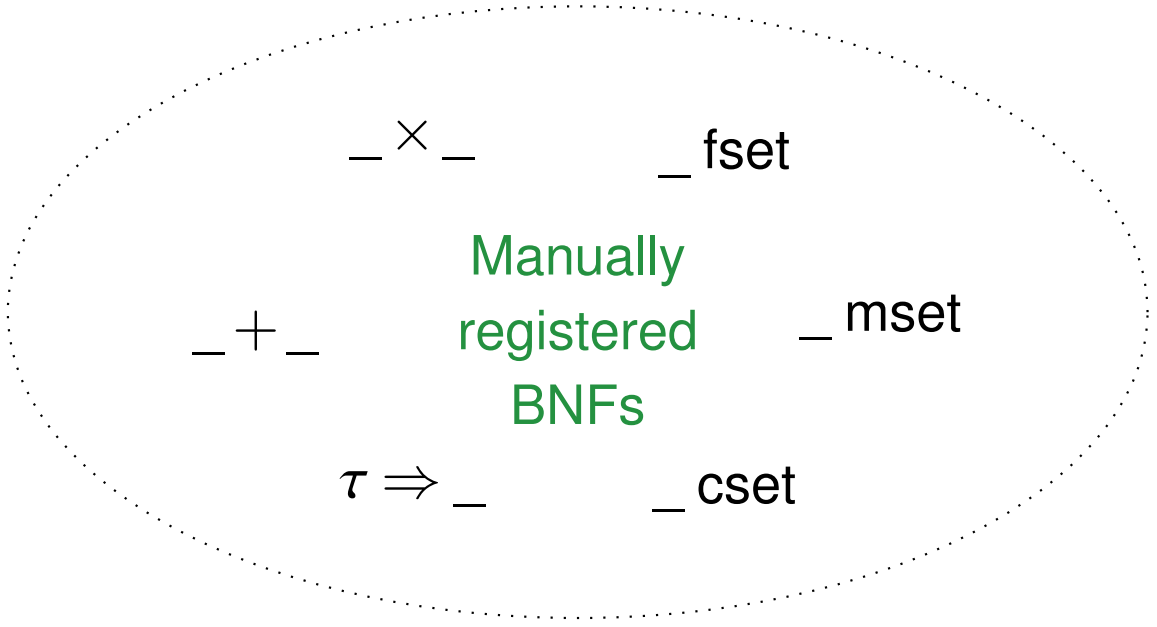
BNF

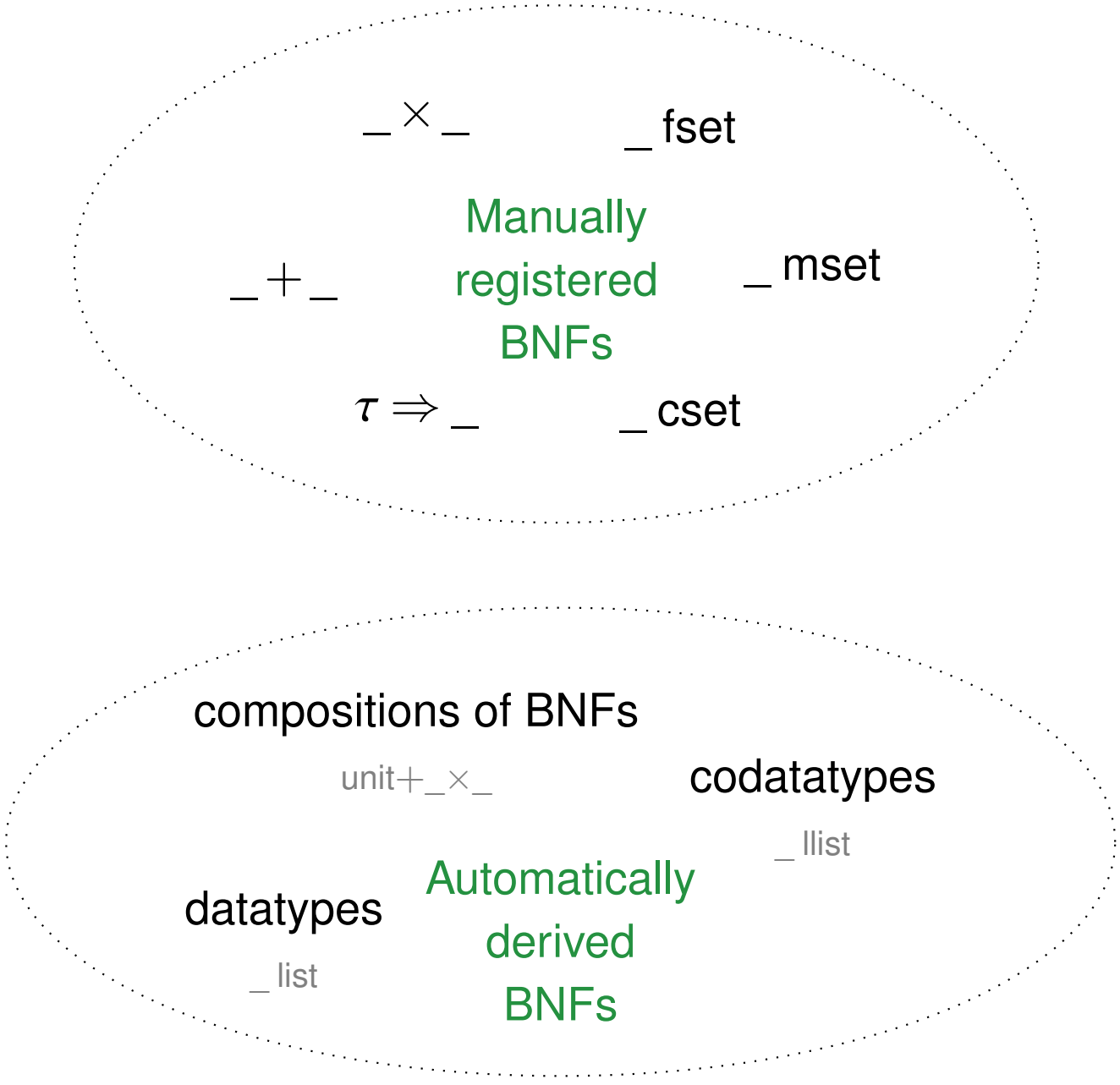
=

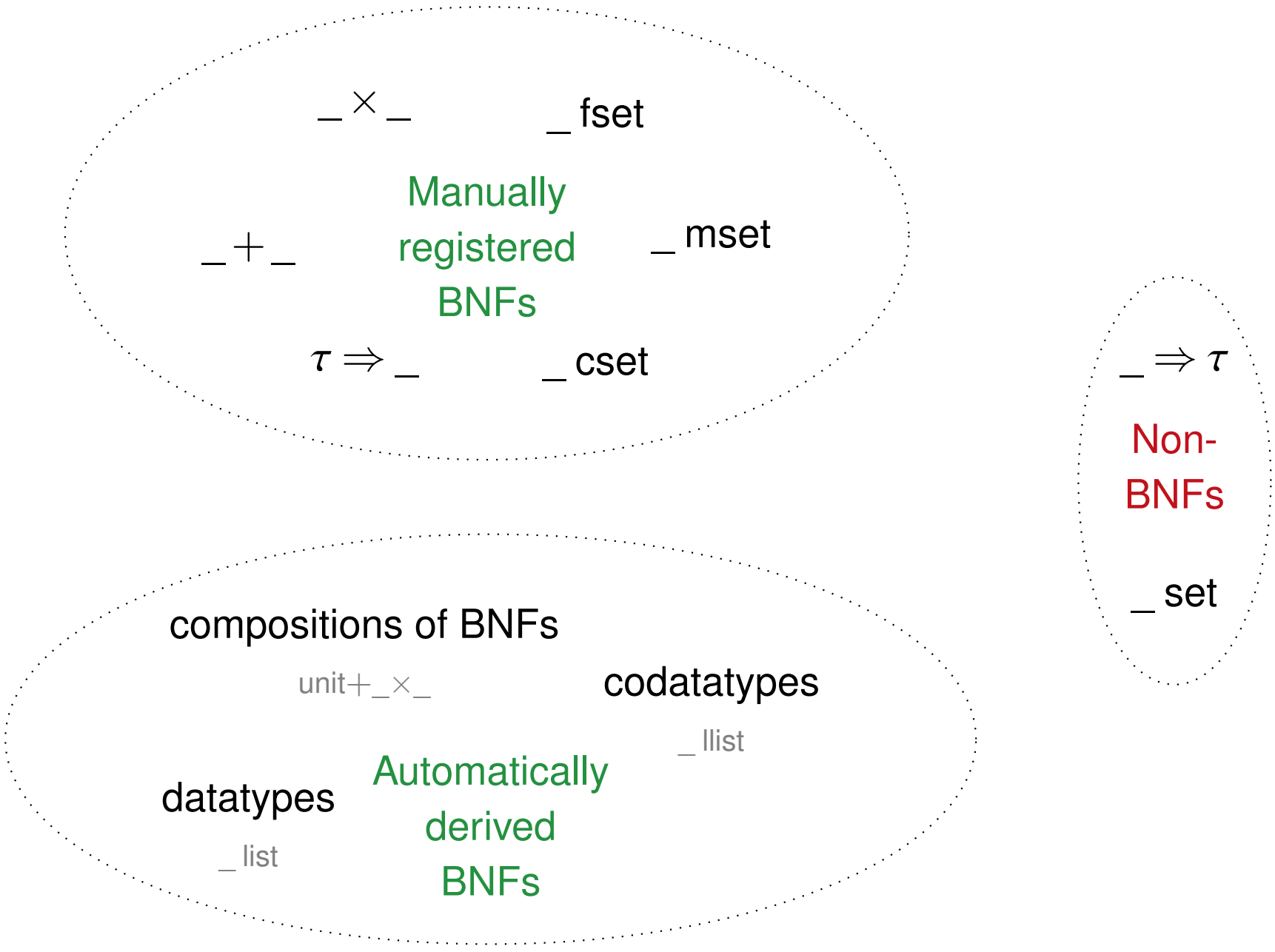
a semantic criterion

for legal rhs of a

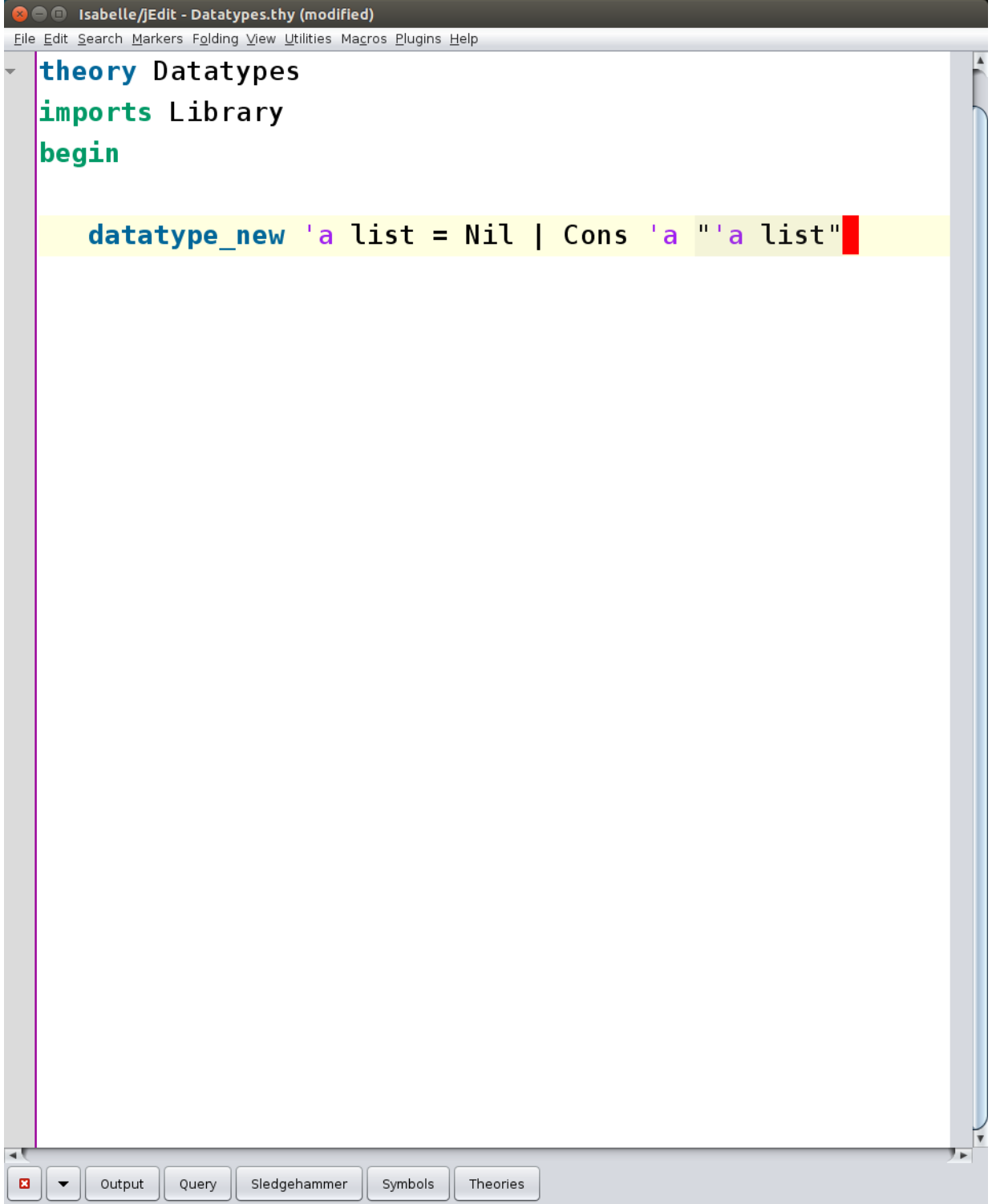
(co)datatype declaration


_ × _
_ fset
Manually
registered
BNFs
_ + _
_ mset
τ ⇒ _
_ cset

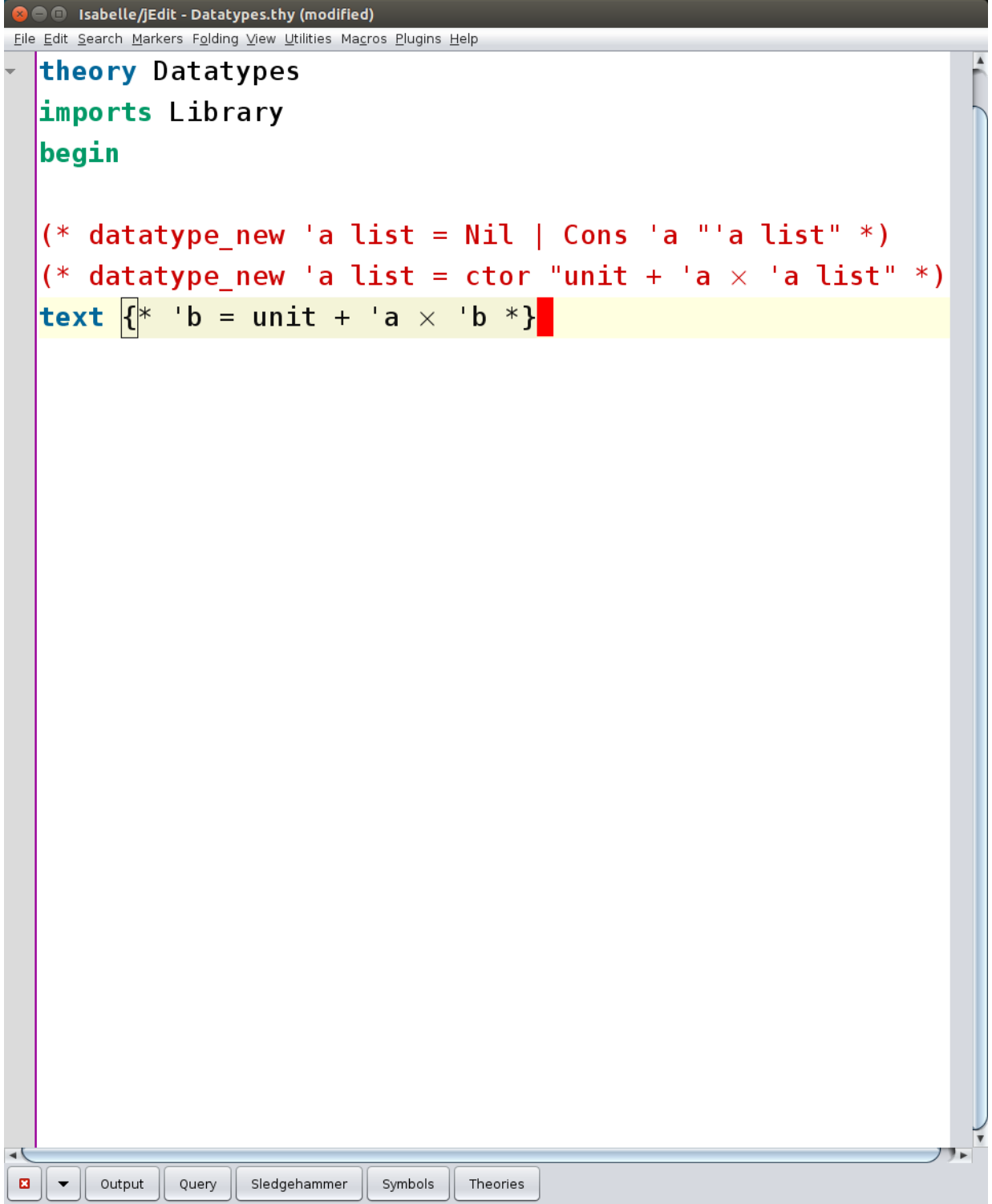

_ × _
_ fset
Manually registered BNFs
_ + _
_ mset
τ ⇒ _
_ cset
compositions of BNFs
unit+_×_
codatypes
_ llist
datatypes
_ list
Automatically derived BNFs

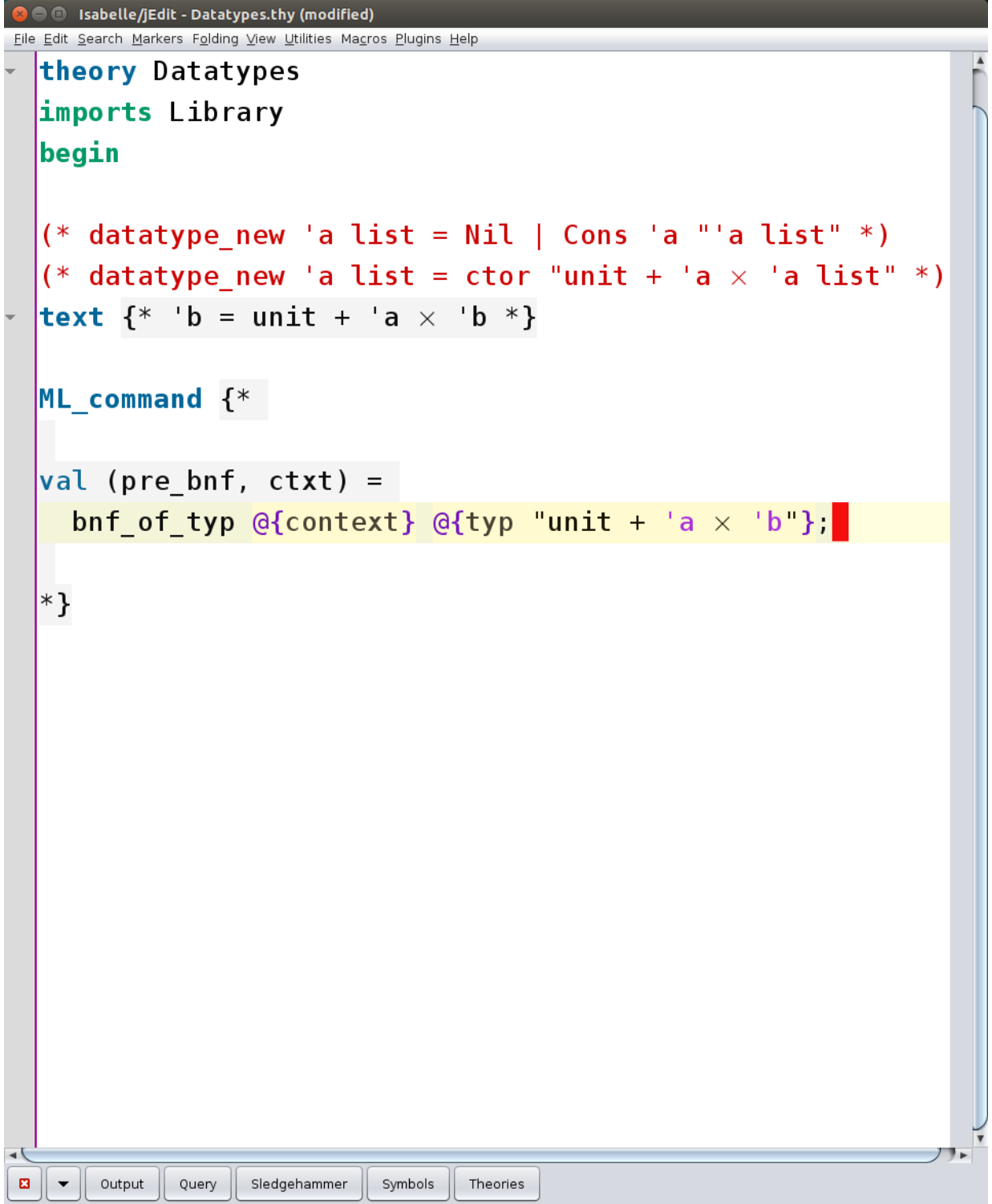

_ × _
_ fset
Manually registered BNFs
_ + _
_ mset
τ ⇒ _
_ cset
compositions of BNFs
unit+_×_
codatypes
_ llist
datatypes
_ list
Automatically derived BNFs
_ ⇒ τ
Non-BNFs
_ set

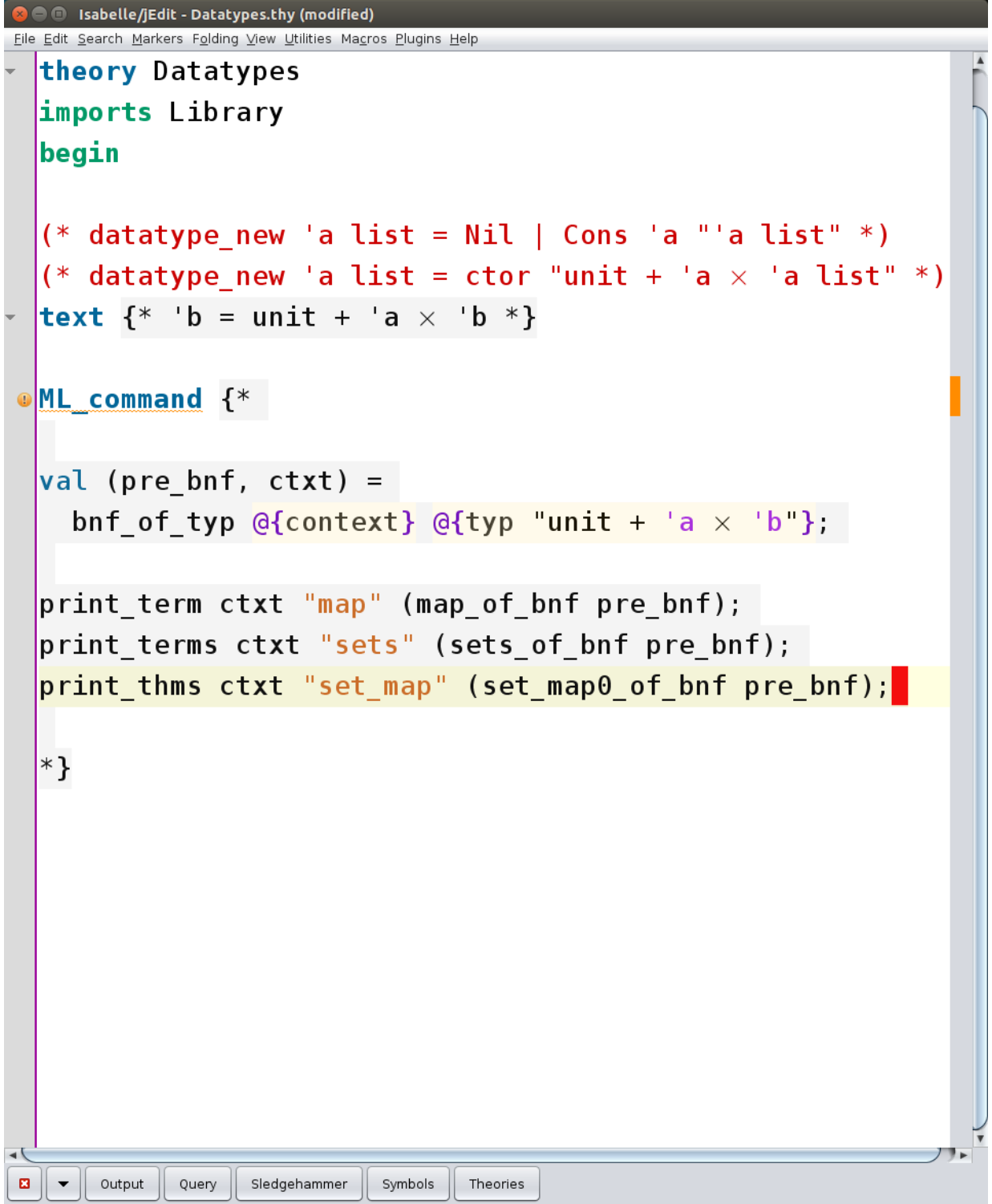
Outline

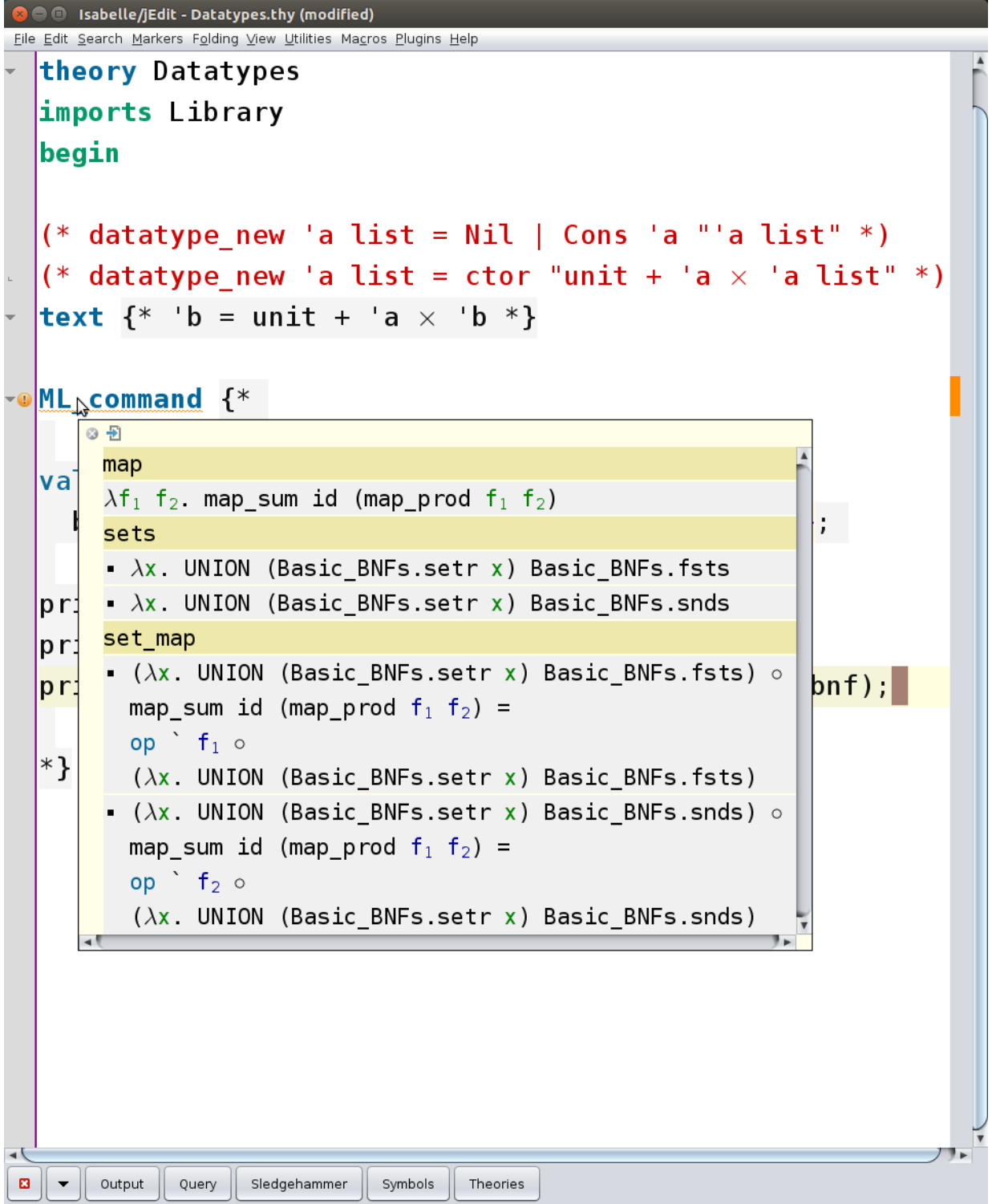

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
Output
Query
Sledgehammer
Symbols
Theories

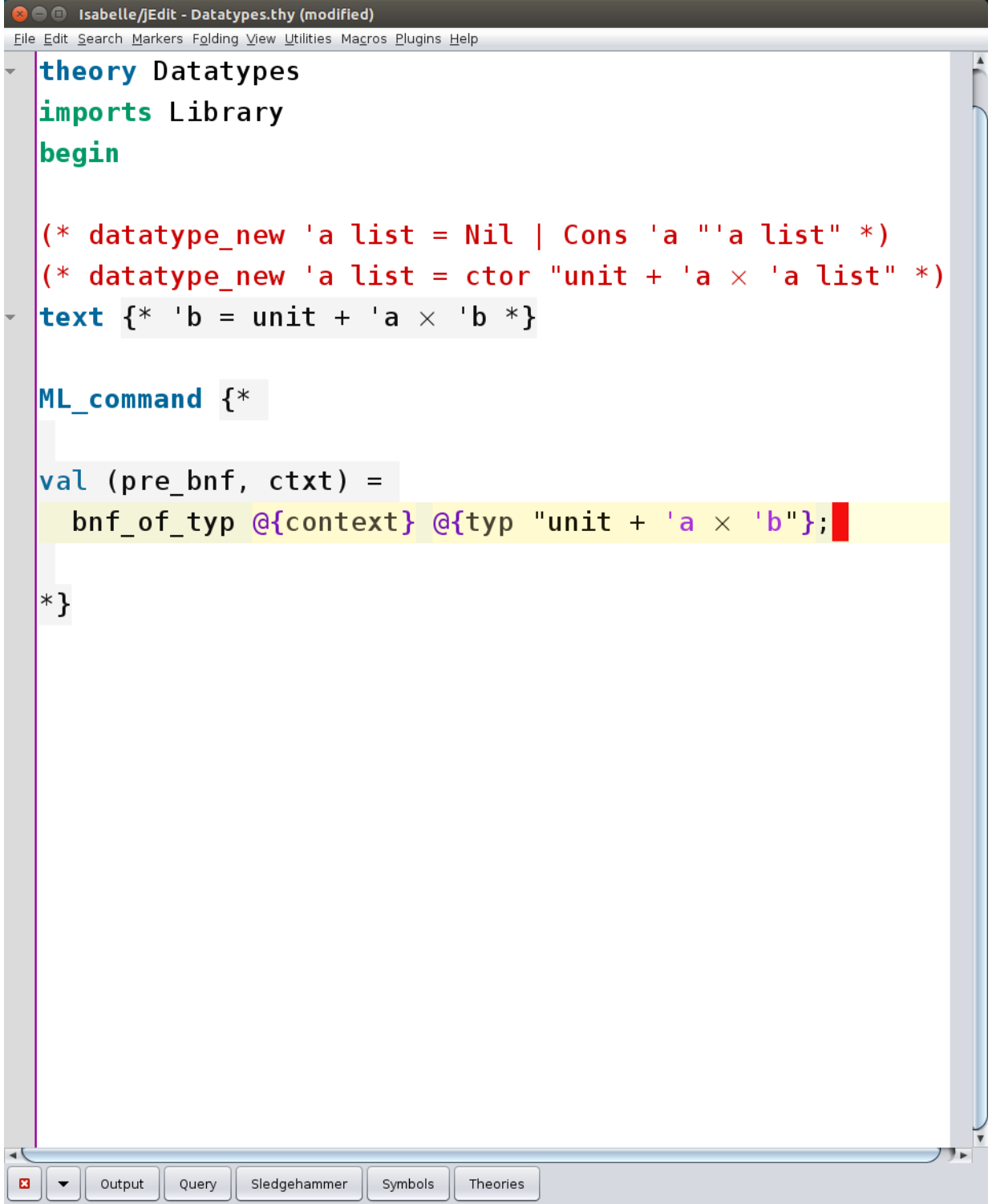

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
(* datatype_new 'a list = Nil | Cons 'a "'a list" *)
datatype_new 'a list = ctor "unit + 'a × 'a list"
Output
Query
Sledgehammer
Symbols
Theories

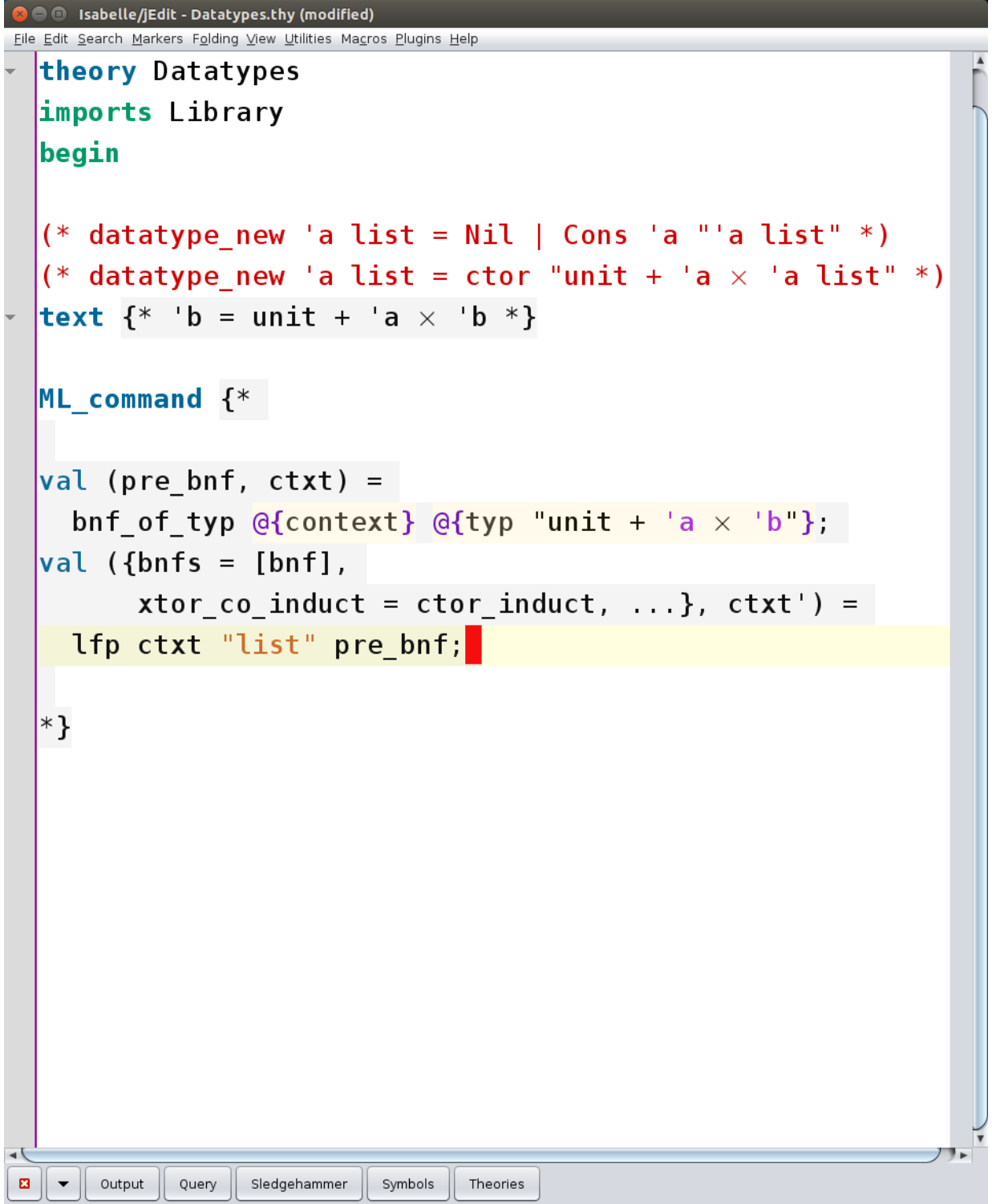

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
(* datatype_new 'a list = Nil | Cons 'a "'a list" *)
(* datatype_new 'a list = ctor "unit + 'a × 'a list" *)
text {* 'b = unit + 'a × 'b *}
Output
Query
Sledgehammer
Symbols
Theories

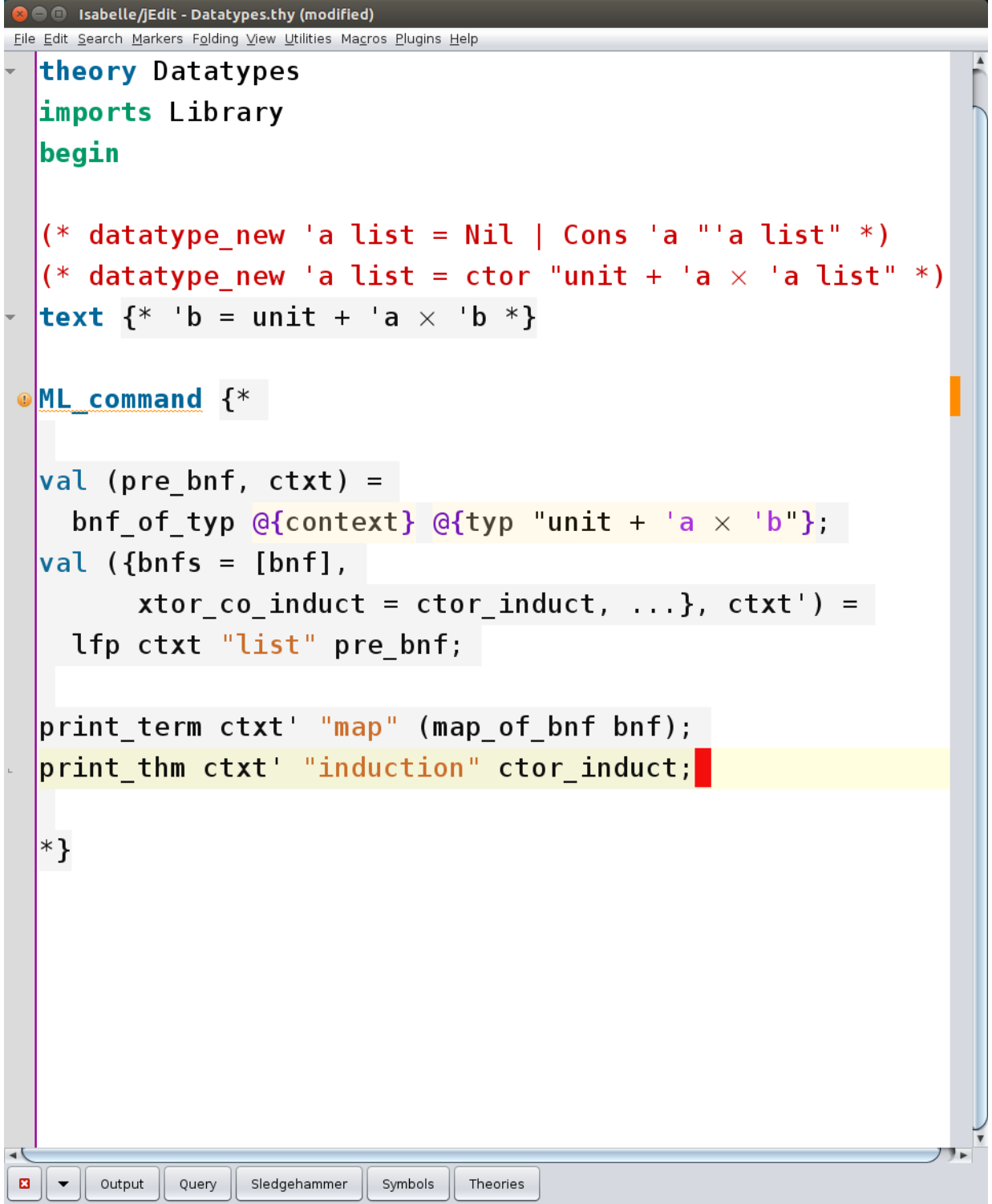

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
(* datatype_new 'a list = Nil | Cons 'a "'a list" *)
(* datatype_new 'a list = ctor "unit + 'a × 'a list" *)
text {* 'b = unit + 'a × 'b *}
ML_command {*
val (pre_bnf, ctxt) =
bnf_of_typ @{context} @{typ "unit + 'a × 'b"};
*}
Output
Query
Sledgehammer
Symbols
Theories

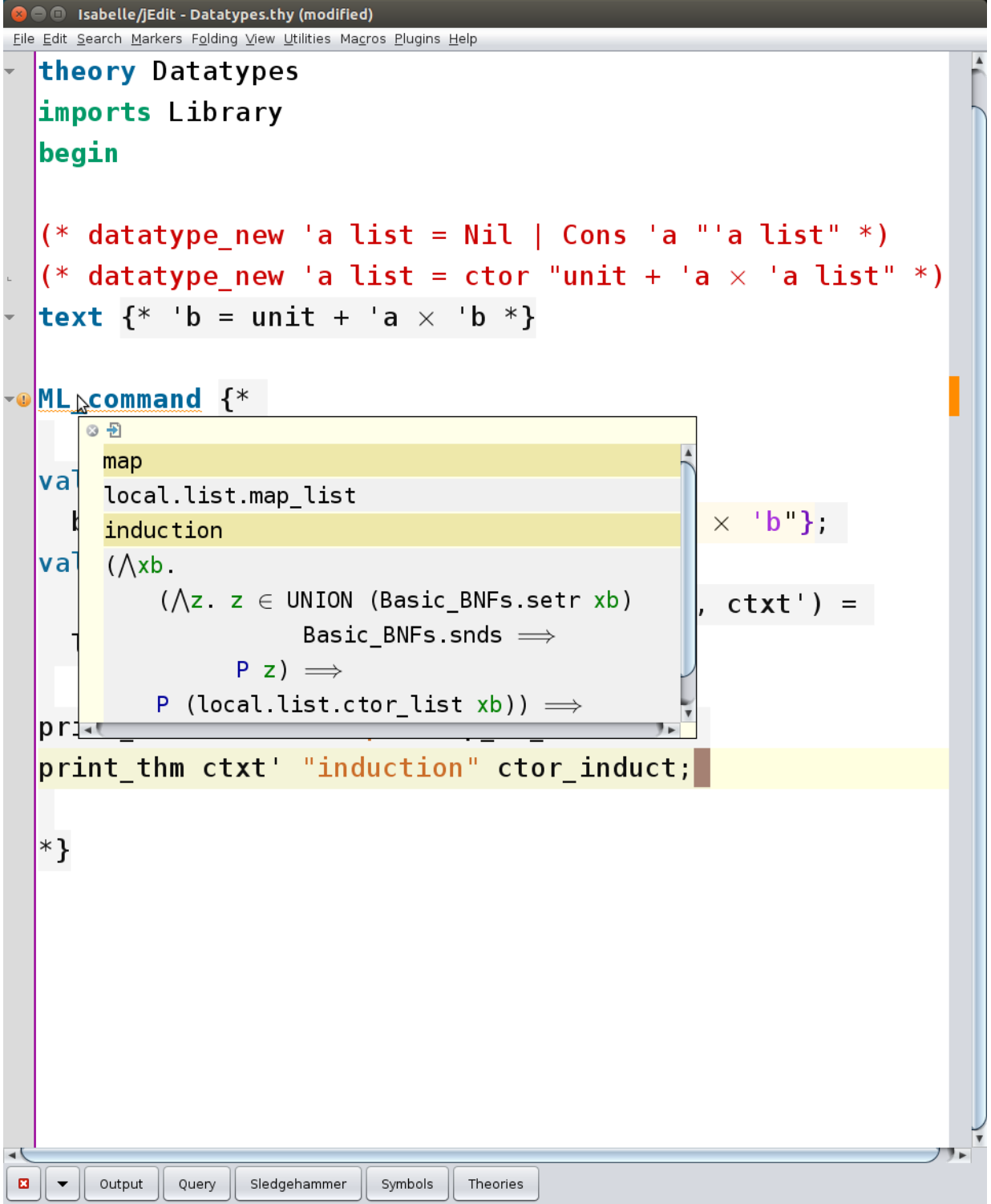

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
(* datatype_new 'a list = Nil | Cons 'a "'a list" *)
(* datatype_new 'a list = ctor "unit + 'a × 'a list" *)
text {* 'b = unit + 'a × 'b *}
ML_command {*
val (pre_bnf, ctxt) =
bnf_of_typ @{context} @{typ "unit + 'a × 'b"};
print_term ctxt "map" (map_of_bnf pre_bnf);
print_terms ctxt "sets" (sets_of_bnf pre_bnf);
print_thms ctxt "set_map" (set_map0_of_bnf pre_bnf);
*}
Output
Query
Sledgehammer
Symbols
Theories

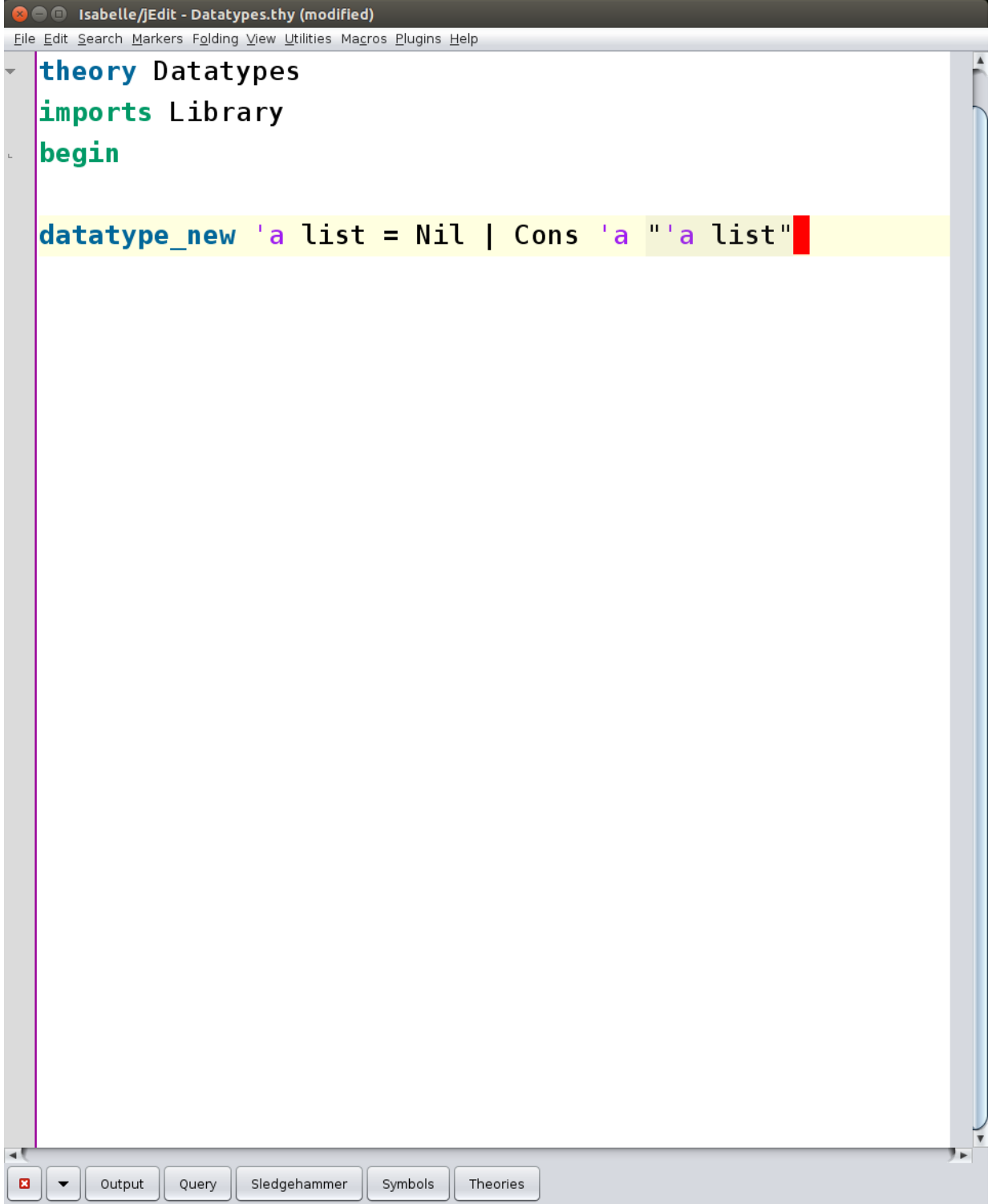

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
(* datatype_new 'a list = Nil | Cons 'a "'a list" *)
(* datatype_new 'a list = ctor "unit + 'a × 'a list" *)
text {* 'b = unit + 'a × 'b *}
ML_command {*
map
λf1 f2. map_sum id (map_prod f1 f2)
sets
▪ λx. UNION (Basic_BNFs.setr x) Basic_BNFs.fsts
▪ λx. UNION (Basic_BNFs.setr x) Basic_BNFs.snds
set_map
▪ (λx. UNION (Basic_BNFs.setr x) Basic_BNFs.fsts) ∘
map_sum id (map_prod f1 f2) =
op ` f1 ∘
(λx. UNION (Basic_BNFs.setr x) Basic_BNFs.fsts)
▪ (λx. UNION (Basic_BNFs.setr x) Basic_BNFs.snds) ∘
map_sum id (map_prod f1 f2) =
op ` f2 ∘
(λx. UNION (Basic_BNFs.setr x) Basic_BNFs.snds)
Output
Query
Sledgehammer
Symbols
Theories

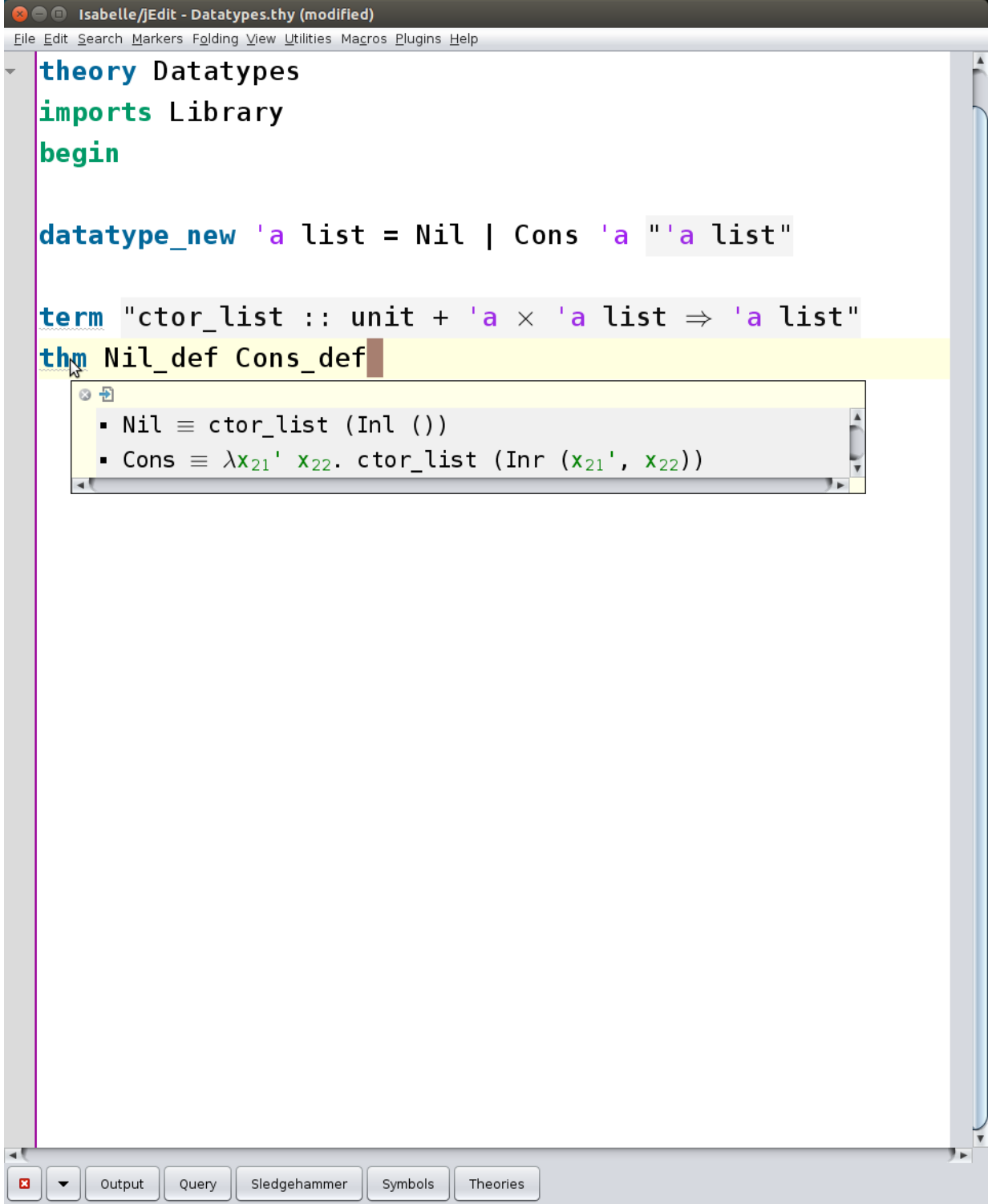

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
(* datatype_new 'a list = Nil | Cons 'a "'a list" *)
(* datatype_new 'a list = ctor "unit + 'a × 'a list" *)
text {* 'b = unit + 'a × 'b *}
ML_command {*
val (pre_bnf, ctxt) =
bnf_of_typ @{context} @{typ "unit + 'a × 'b"};
*}
Output
Query
Sledgehammer
Symbols
Theories

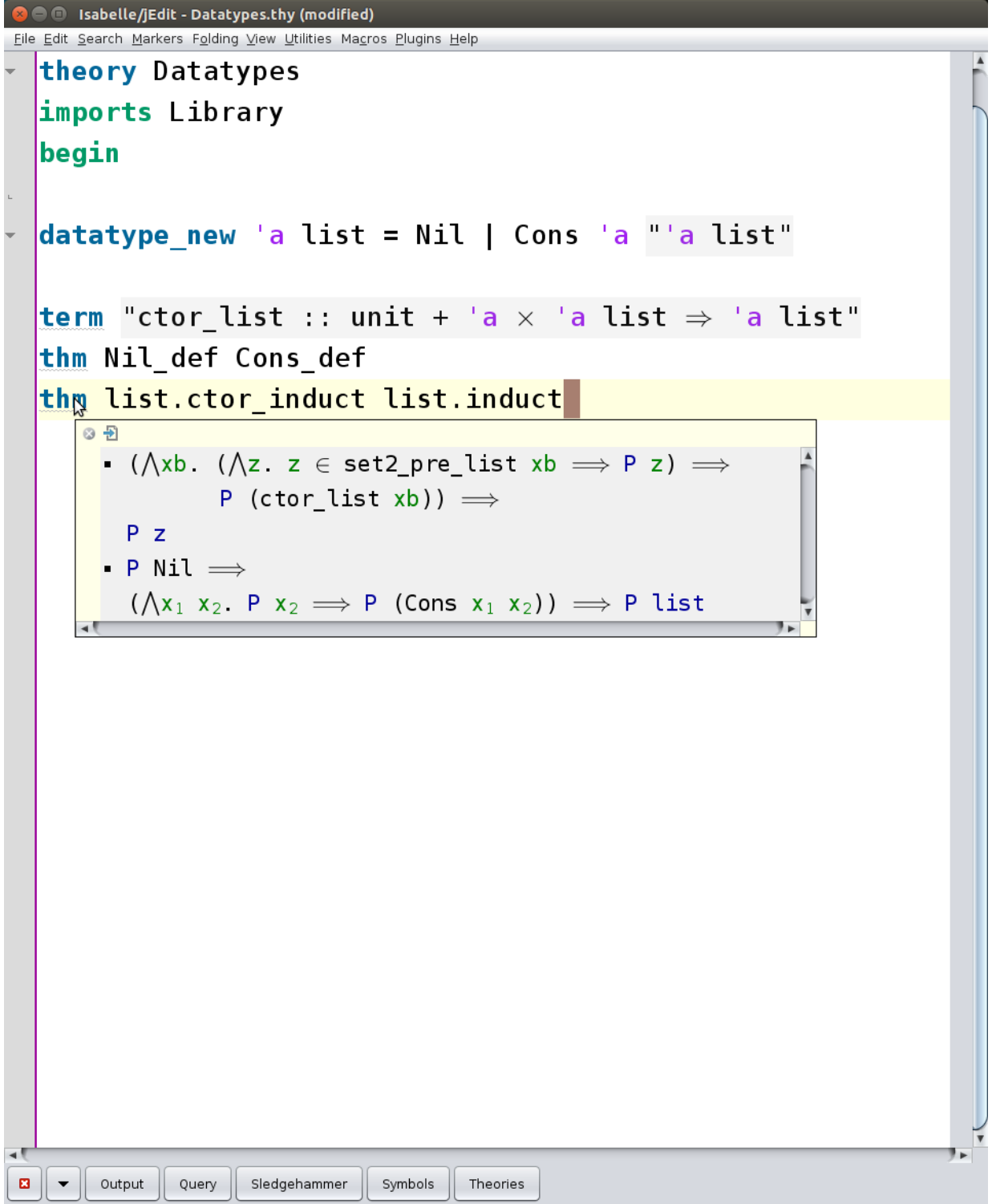

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
(* datatype_new 'a list = Nil | Cons 'a "'a list" *)
(* datatype_new 'a list = ctor "unit + 'a × 'a list" *)
text {* 'b = unit + 'a × 'b *}
ML_command {*
val (pre_bnf, ctxt) =
bnf_of_typ @{context} @{typ "unit + 'a × 'b"};
val ({bnfs = [bnf],
xtor_co_induct = ctor_induct, ...}, ctxt') =
lfp ctxt "list" pre_bnf;
*}
Output
Query
Sledgehammer
Symbols
Theories

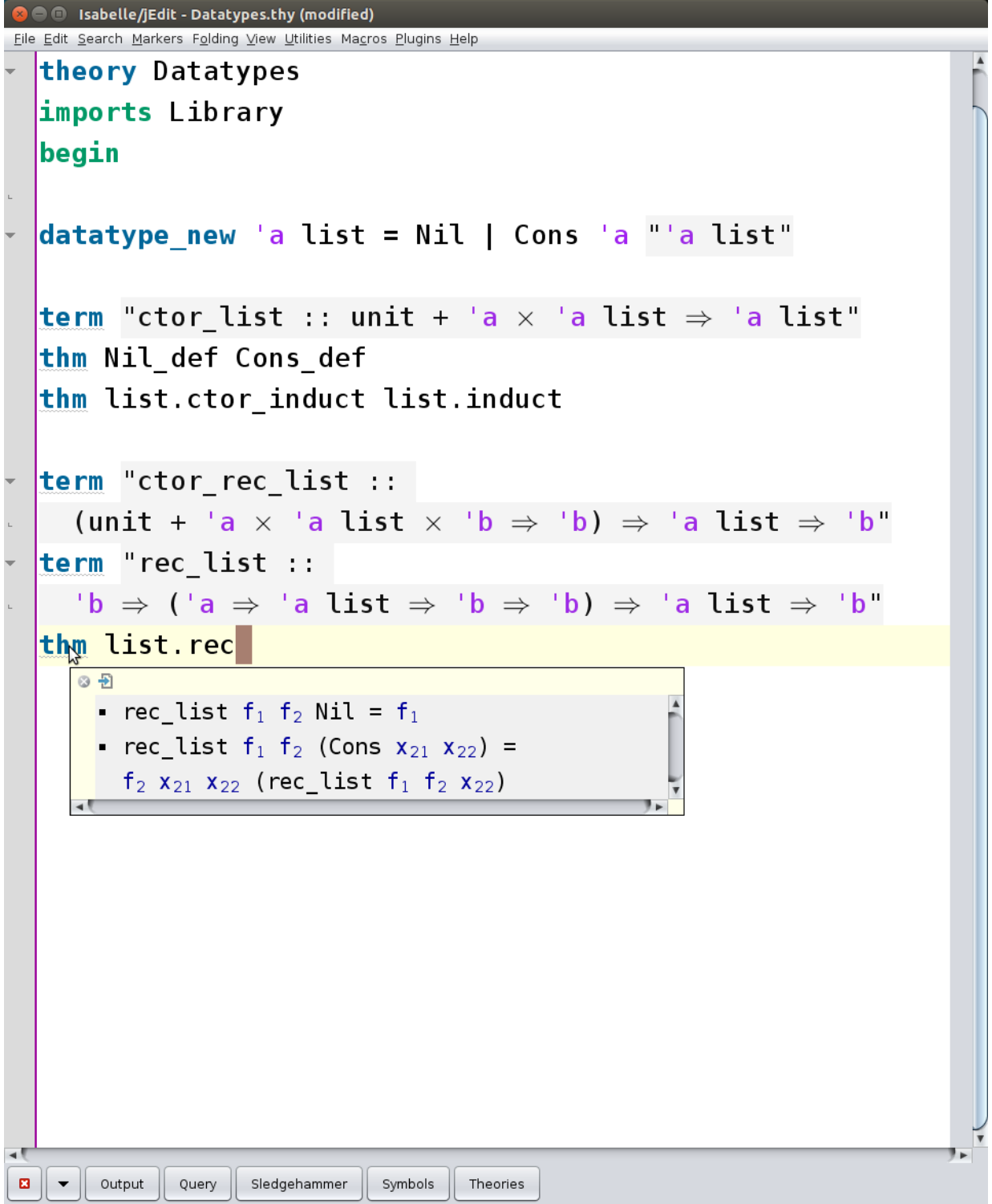

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
(* datatype_new 'a list = Nil | Cons 'a "'a list" *)
(* datatype_new 'a list = ctor "unit + 'a × 'a list" *)
text {* 'b = unit + 'a × 'b *}
ML_command {*
val (pre_bnf, ctxt) =
bnf_of_typ @{context} @{typ "unit + 'a × 'b"};
val ({bnfs = [bnf],
xtor_co_induct = ctor_induct, ...}, ctxt') =
lfp ctxt "list" pre_bnf;
print_term ctxt' "map" (map_of_bnf bnf);
print_thm ctxt' "induction" ctor_induct;
*}
Output
Query
Sledgehammer
Symbols
Theories

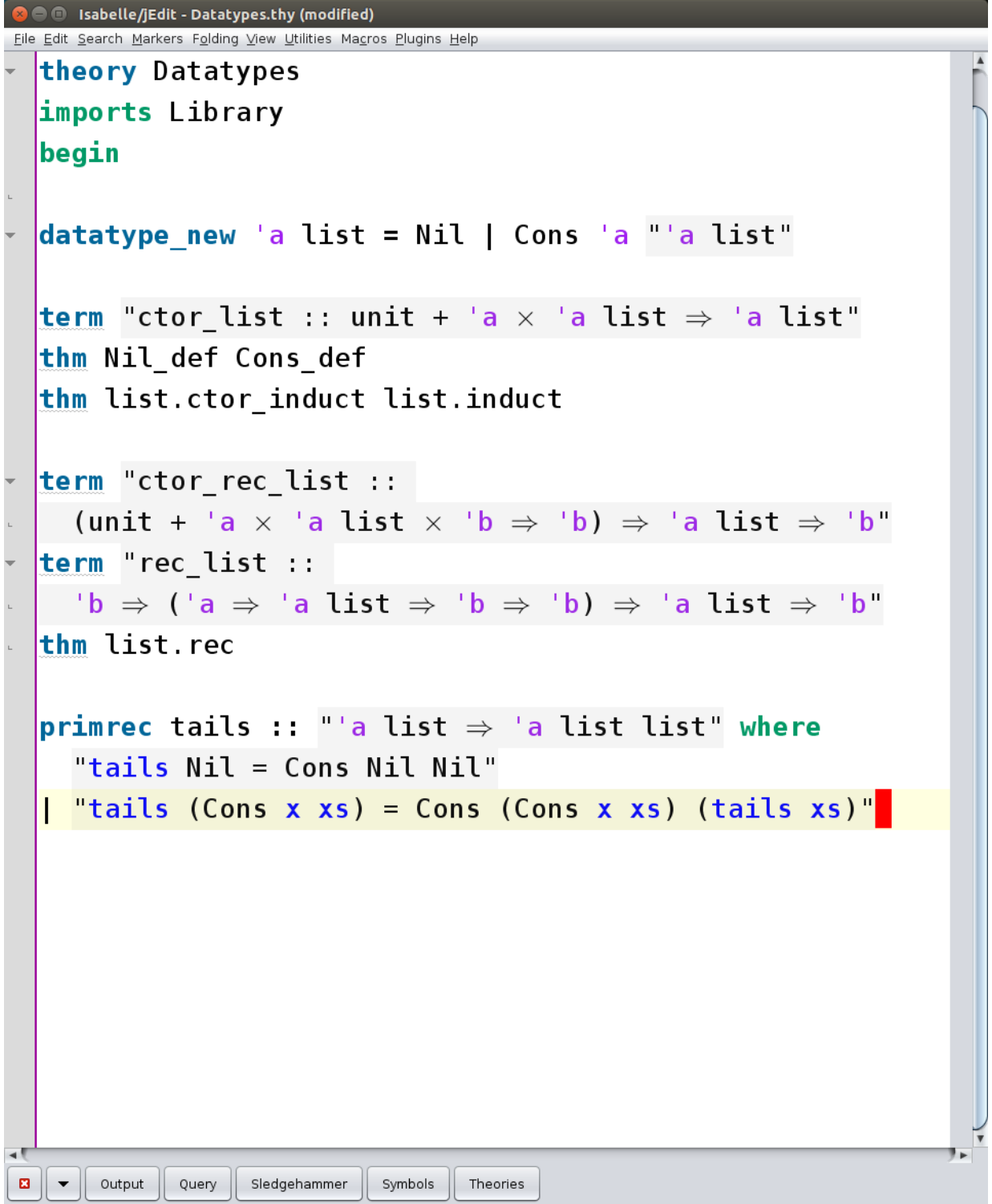

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
(* datatype_new 'a list = Nil | Cons 'a "'a list" *)
(* datatype_new 'a list = ctor "unit + 'a × 'a list" *)
text {* 'b = unit + 'a × 'b *}
ML_command {*
map
local.list.map_list
induction
(⋀xb.
(⋀z. z ∈ UNION (Basic_BNFs.setr xb)
Basic_BNFs.snds ⟹
P z) ⟹
P (local.list.ctor_list xb)) ⟹
× 'b"};
, ctxt') =
print_thm ctxt' "induction" ctor_induct;
*}
Output
Query
Sledgehammer
Symbols
Theories

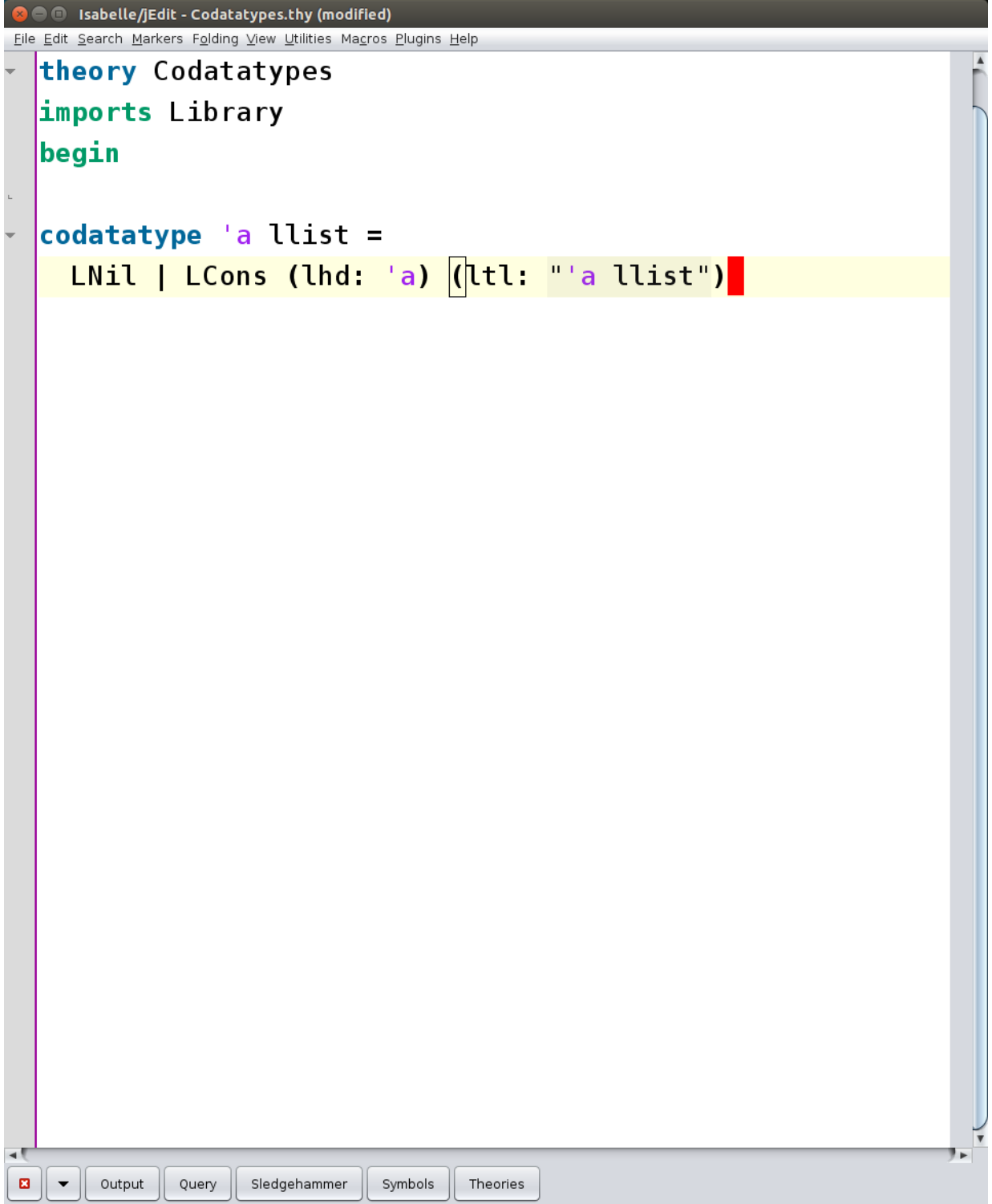

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
Output
Query
Sledgehammer
Symbols
Theories

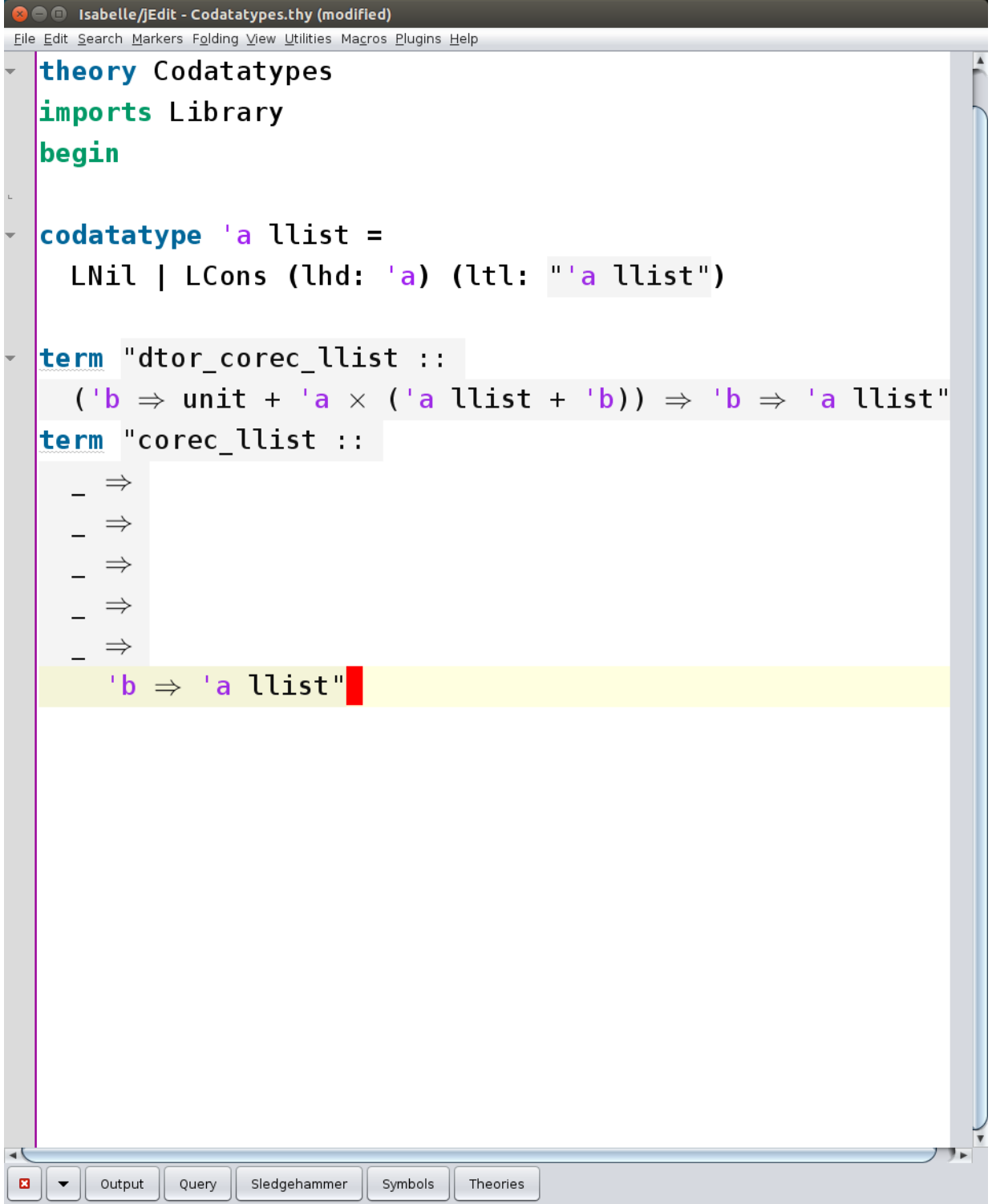

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
term "ctor_list :: unit + 'a × 'a list ⇒ 'a list"
thm Nil_def Cons_def
▪ Nil ≡ ctor_list (Inl ())
▪ Cons ≡ λx21' x22. ctor_list (Inr (x21', x22))
Output
Query
Sledgehammer
Symbols
Theories

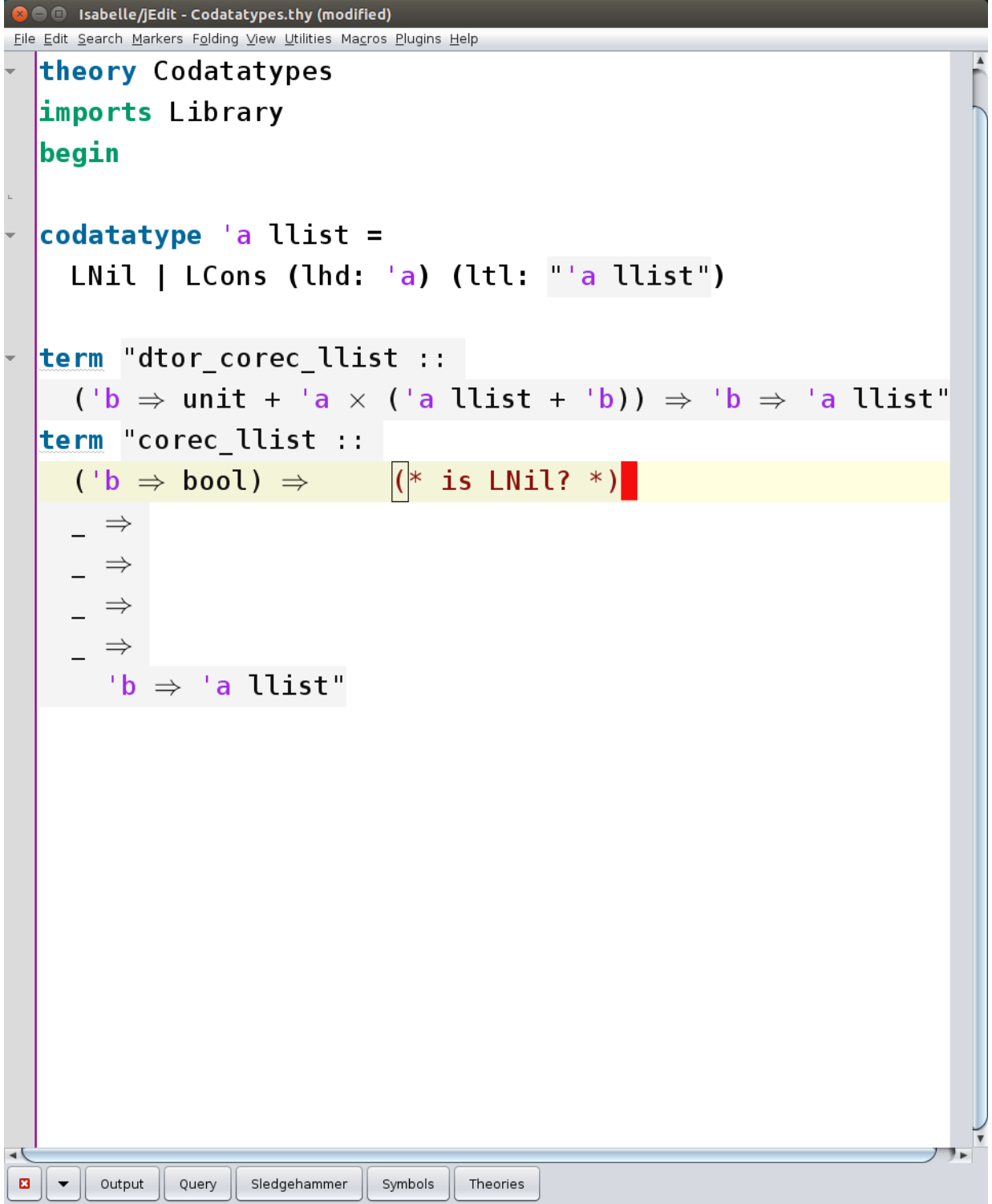

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
term "ctor_list :: unit + 'a × 'a list ⇒ 'a list"
thm Nil_def Cons_def
thm list.ctor_induct list.induct
▪ (⋀xb. (⋀z. z ∈ set2_pre_list xb ⟹ P z) ⟹
P (ctor_list xb)) ⟹
P z
▪ P Nil ⟹
(⋀x1 x2. P x2 ⟹ P (Cons x1 x2)) ⟹ P list
Output
Query
Sledgehammer
Symbols
Theories

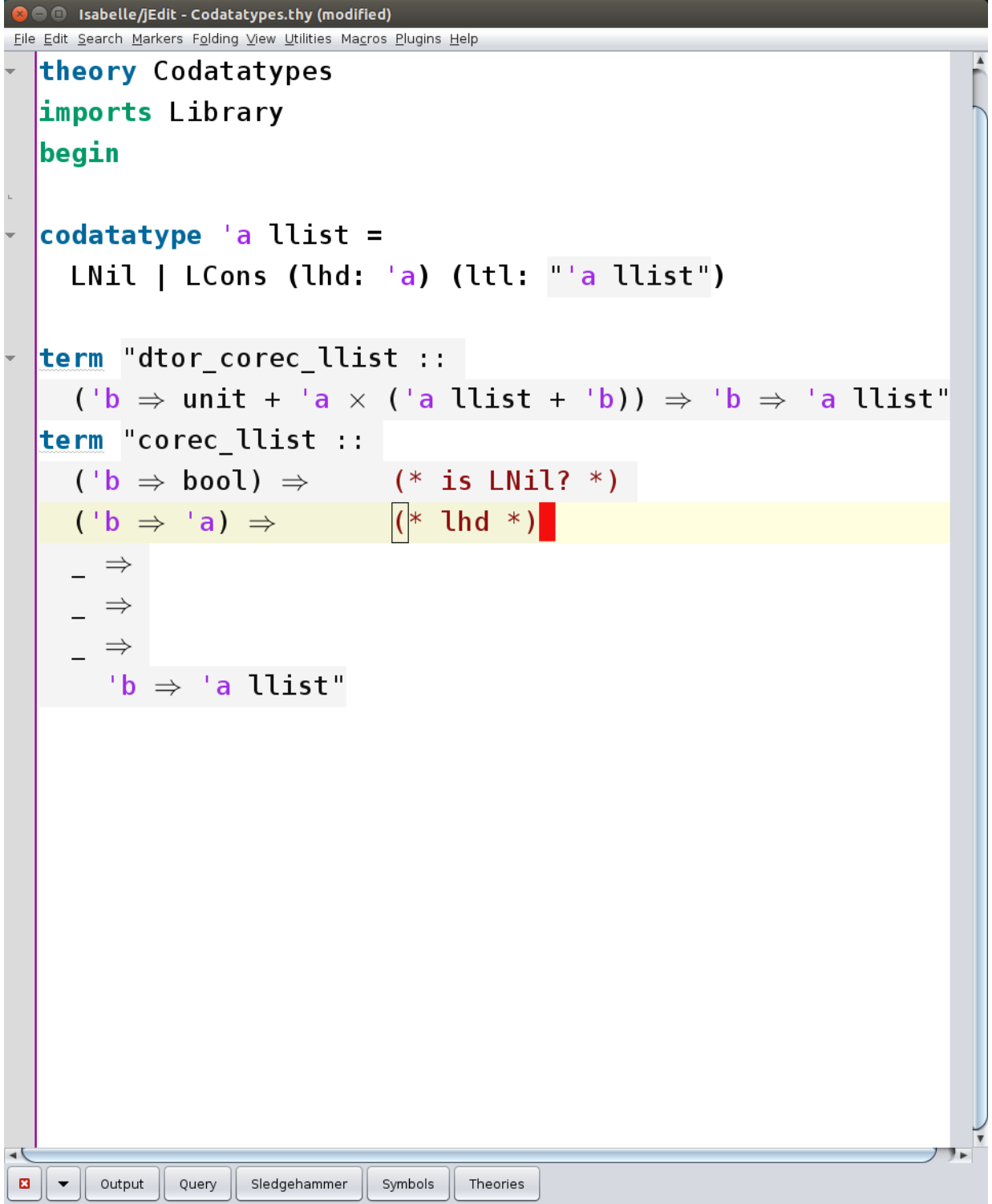

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
term "ctor_list :: unit + 'a × 'a list ⇒ 'a list"
thm Nil_def Cons_def
thm list.ctor_induct list.induct
term "ctor_rec_list ::
(unit + 'a × 'a list × 'b ⇒ 'b) ⇒ 'a list ⇒ 'b"
term "rec_list ::
'b ⇒ ('a ⇒ 'a list ⇒ 'b ⇒ 'b) ⇒ 'a list ⇒ 'b"
thm list.rec
rec_list f1 f2 Nil = f1
rec_list f1 f2 (Cons x21 x22) =
f2 x21 x22 (rec_list f1 f2 x22)
Output
Query
Sledgehammer
Symbols
Theories

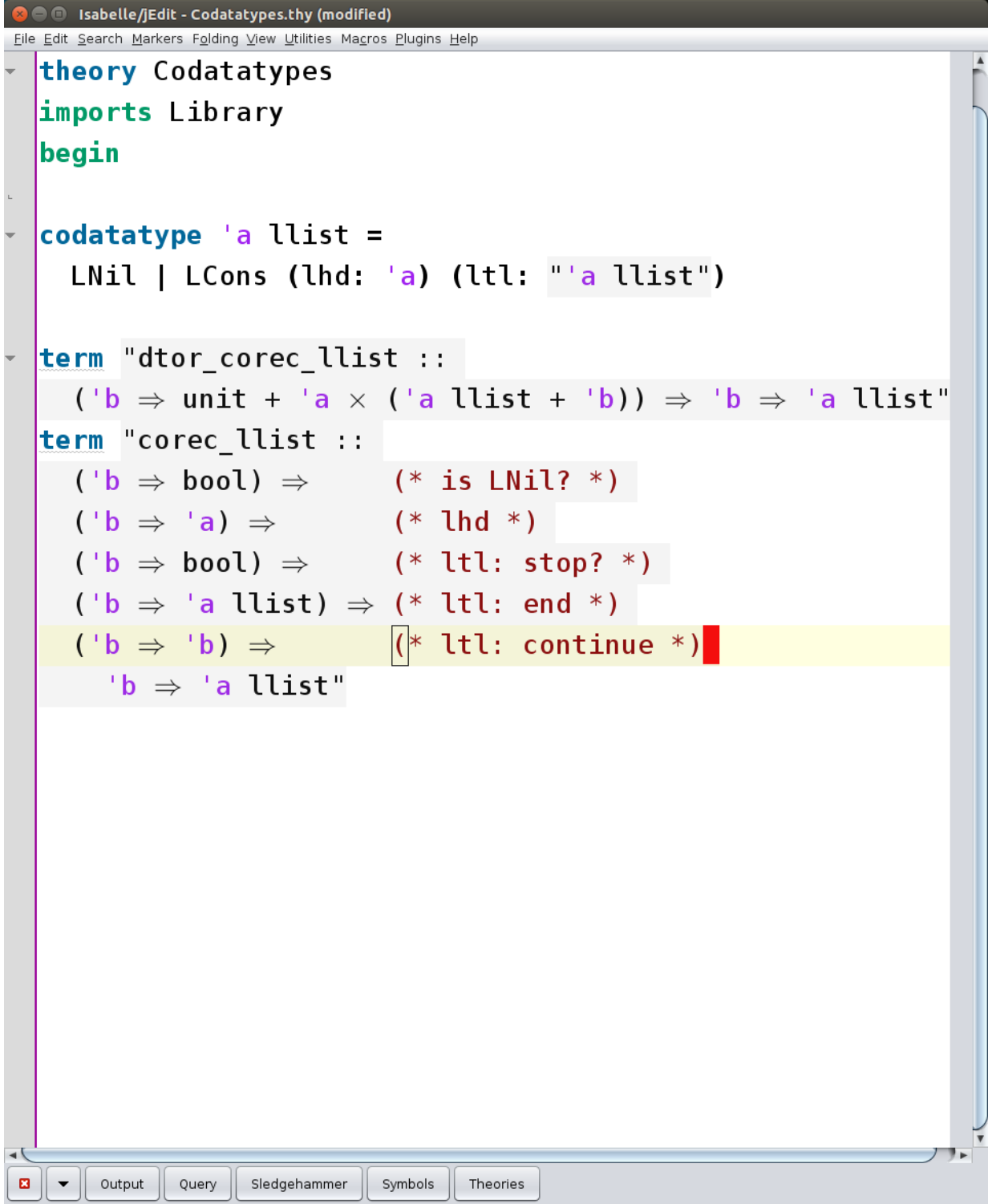

Isabelle/jEdit - Datatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Datatypes
imports Library
begin
datatype_new 'a list = Nil | Cons 'a "'a list"
term "ctor_list :: unit + 'a × 'a list ⇒ 'a list"
thm Nil_def Cons_def
thm list.ctor_induct list.induct
term "ctor_rec_list ::
(unit + 'a × 'a list × 'b ⇒ 'b) ⇒ 'a list ⇒ 'b"
term "rec_list ::
'b ⇒ ('a ⇒ 'a list ⇒ 'b ⇒ 'b) ⇒ 'a list ⇒ 'b"
thm list.rec
primrec tails :: "'a list ⇒ 'a list list" where
"tails Nil = Cons Nil Nil"
| "tails (Cons x xs) = Cons (Cons x xs) (tails xs)"
Output
Query
Sledgehammer
Symbols
Theories

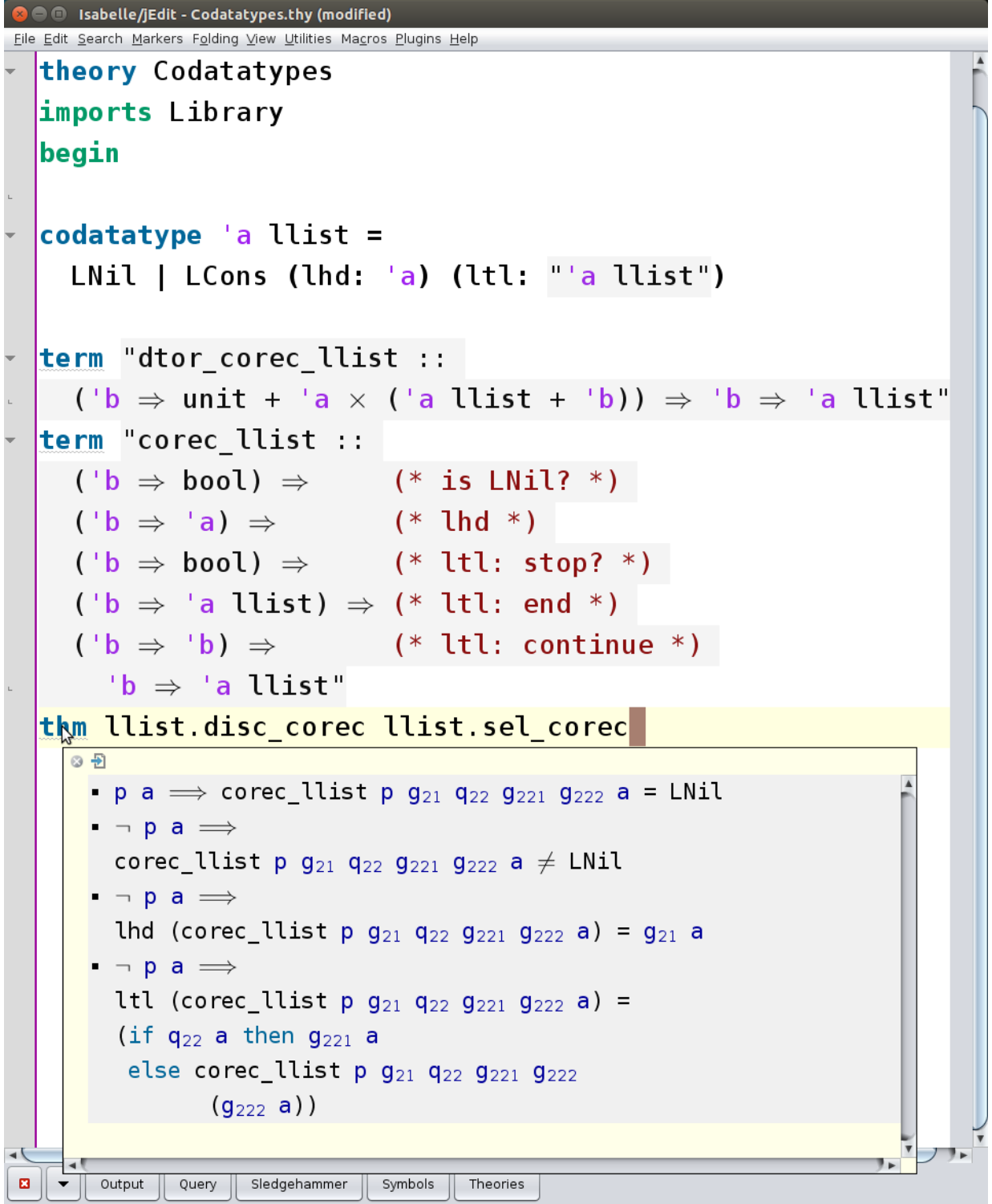

Isabelle/jEdit - Codatatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Codatatypes
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
Output
Query
Sledgehammer
Symbols
Theories

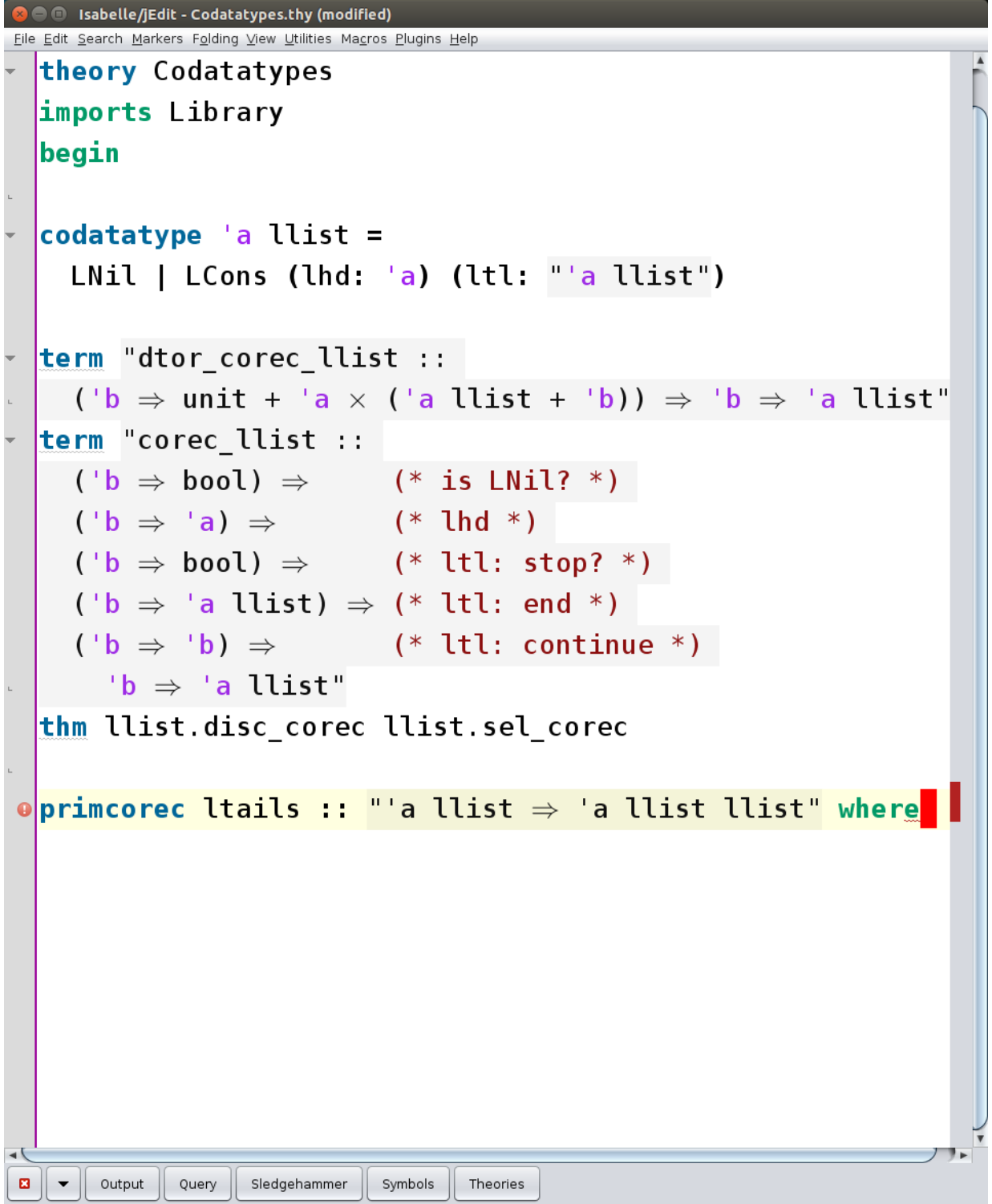

Isabelle/jEdit - Codatatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Codatatypes
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
term "dtor_corec_llist ::
('b ⇒ unit + 'a × ('a llist + 'b)) ⇒ 'b ⇒ 'a llist"
term "corec_llist ::
_ ⇒
_ ⇒
_ ⇒
_ ⇒
_ ⇒
'b ⇒ 'a llist"
Output
Query
Sledgehammer
Symbols
Theories

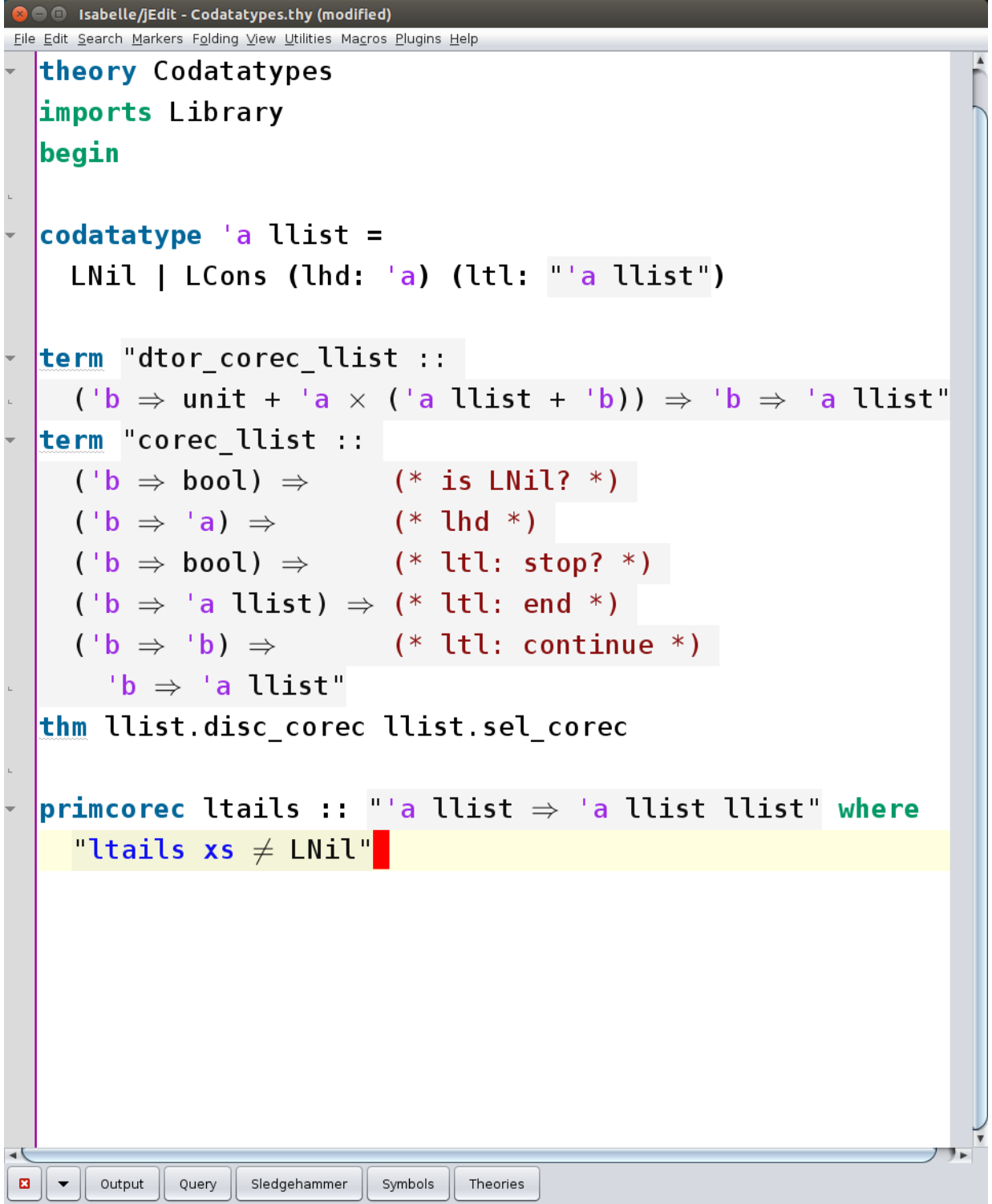

Isabelle/jEdit - Codatatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Codatatypes
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
term "dtor_corec_llist ::
('b ⇒ unit + 'a × ('a llist + 'b)) ⇒ 'b ⇒ 'a llist"
term "corec_llist ::
('b ⇒ bool) ⇒ (* is LNil? *)
_ ⇒
_ ⇒
_ ⇒
_ ⇒
'b ⇒ 'a llist"
Output
Query
Sledgehammer
Symbols
Theories

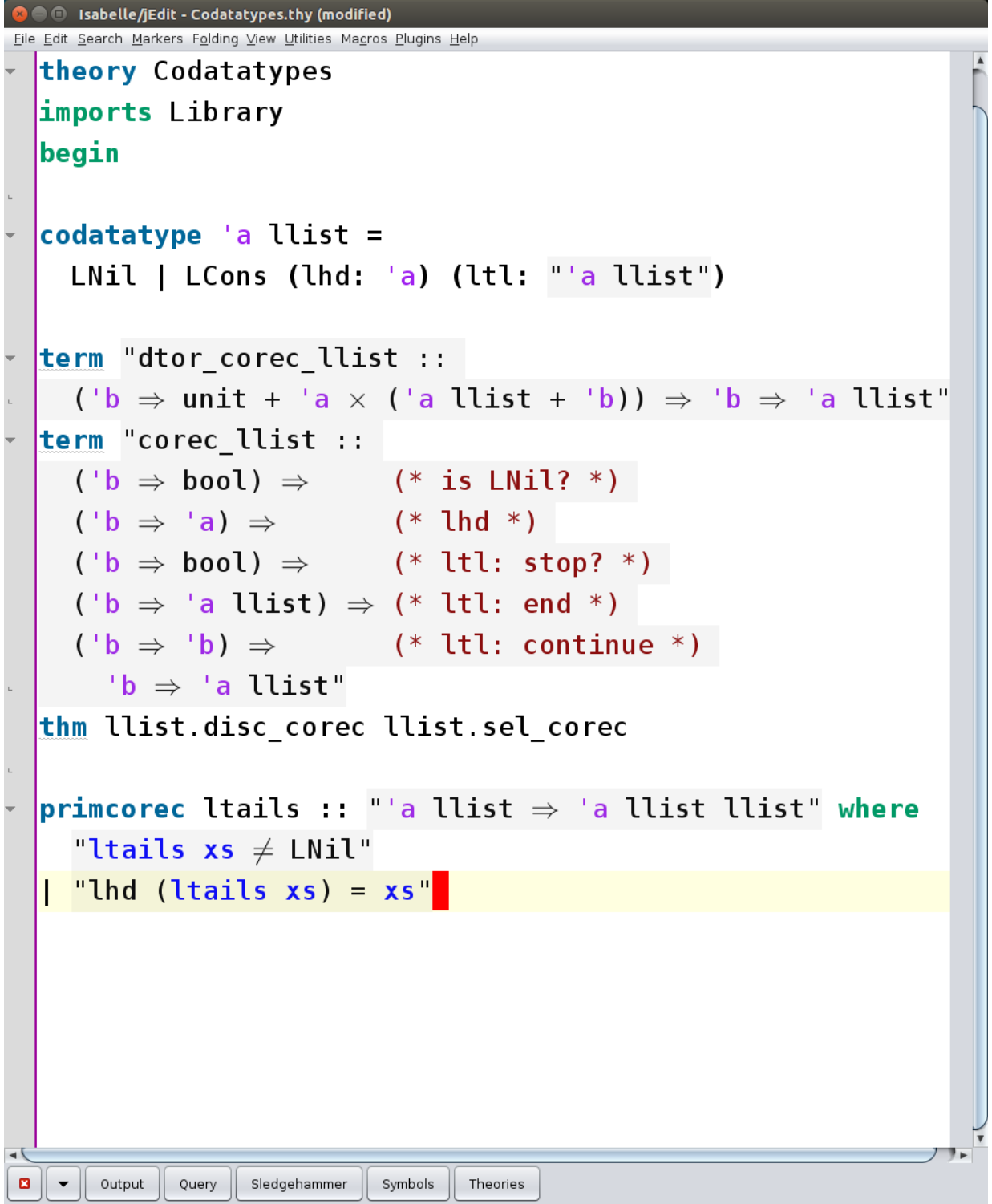

Isabelle/jEdit - Codatatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Codatatypes
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
term "dtor_corec_llist ::
('b ⇒ unit + 'a × ('a llist + 'b)) ⇒ 'b ⇒ 'a llist"
term "corec_llist ::
('b ⇒ bool) ⇒ (* is LNil? *)
('b ⇒ 'a) ⇒ (* lhd *)
_ ⇒
_ ⇒
_ ⇒
'b ⇒ 'a llist"
Output
Query
Sledgehammer
Symbols
Theories

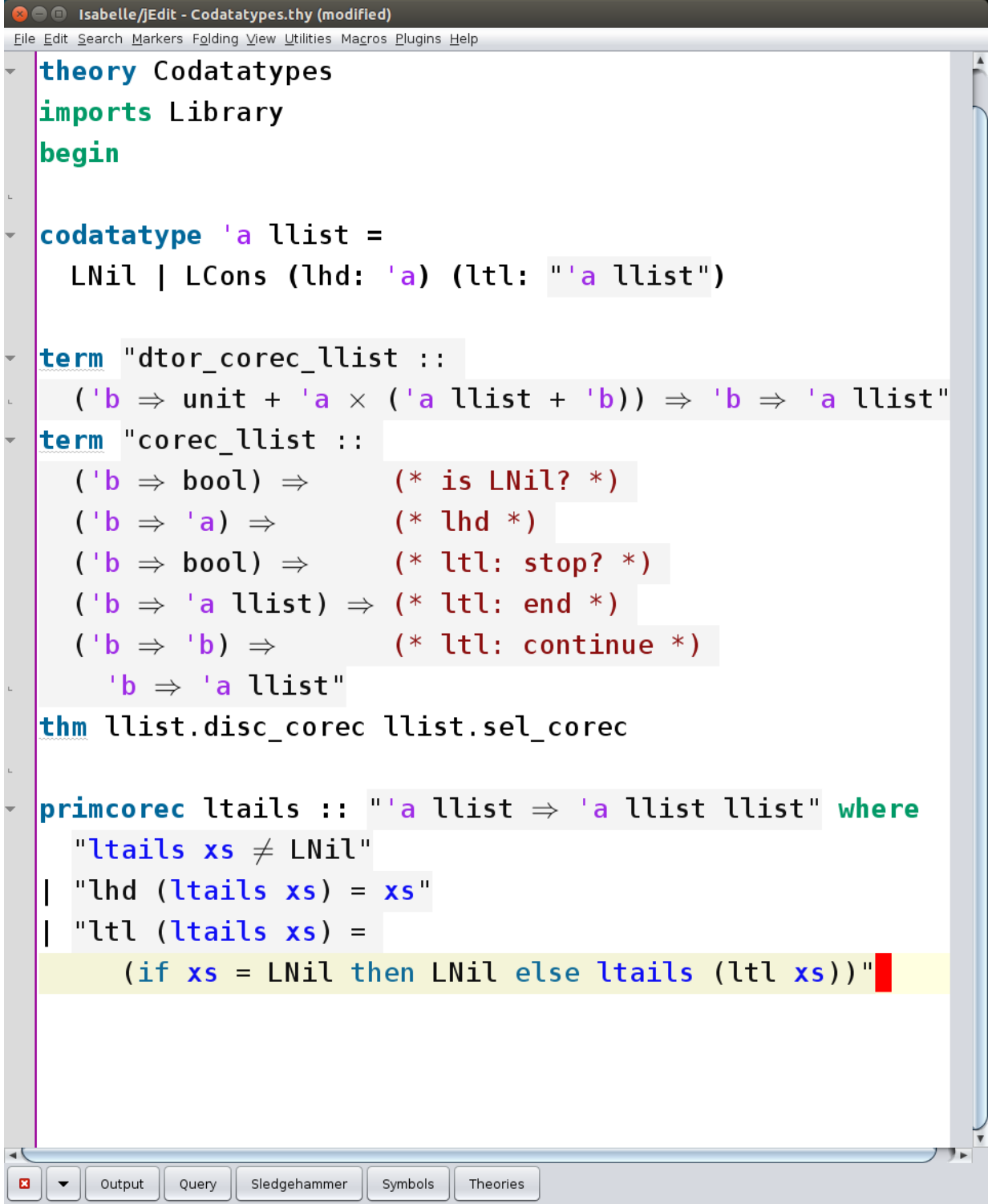

Isabelle/jEdit - Codatatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Codatatypes
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
term "dtor_corec_llist ::
('b ⇒ unit + 'a × ('a llist + 'b)) ⇒ 'b ⇒ 'a llist"
term "corec_llist ::
('b ⇒ bool) ⇒ (* is LNil? *)
('b ⇒ 'a) ⇒ (* lhd *)
('b ⇒ bool) ⇒ (* ltl: stop? *)
('b ⇒ 'a llist) ⇒ (* ltl: end *)
('b ⇒ 'b) ⇒ (* ltl: continue *)
'b ⇒ 'a llist"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Codatatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Codatatypes
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
term "dtor_corec_llist ::
('b ⇒ unit + 'a × ('a llist + 'b)) ⇒ 'b ⇒ 'a llist"
term "corec_llist ::
('b ⇒ bool) ⇒ (* is LNil? *)
('b ⇒ 'a) ⇒ (* lhd *)
('b ⇒ bool) ⇒ (* ltl: stop? *)
('b ⇒ 'a llist) ⇒ (* ltl: end *)
('b ⇒ 'b) ⇒ (* ltl: continue *)
'b ⇒ 'a llist"
thm llist.disc_corec llist.sel_corec
▪ p a ⟹ corec_llist p g21 q22 g221 g222 a = LNil
▪ ¬ p a ⟹
corec_llist p g21 q22 g221 g222 a ≠ LNil
▪ ¬ p a ⟹
lhd (corec_llist p g21 q22 g221 g222 a) = g21 a
▪ ¬ p a ⟹
ltl (corec_llist p g21 q22 g221 g222 a) =
(if q22 a then g221 a
else corec_llist p g21 q22 g221 g222
(g222 a))
Output
Query
Sledgehammer
Symbols
Theories

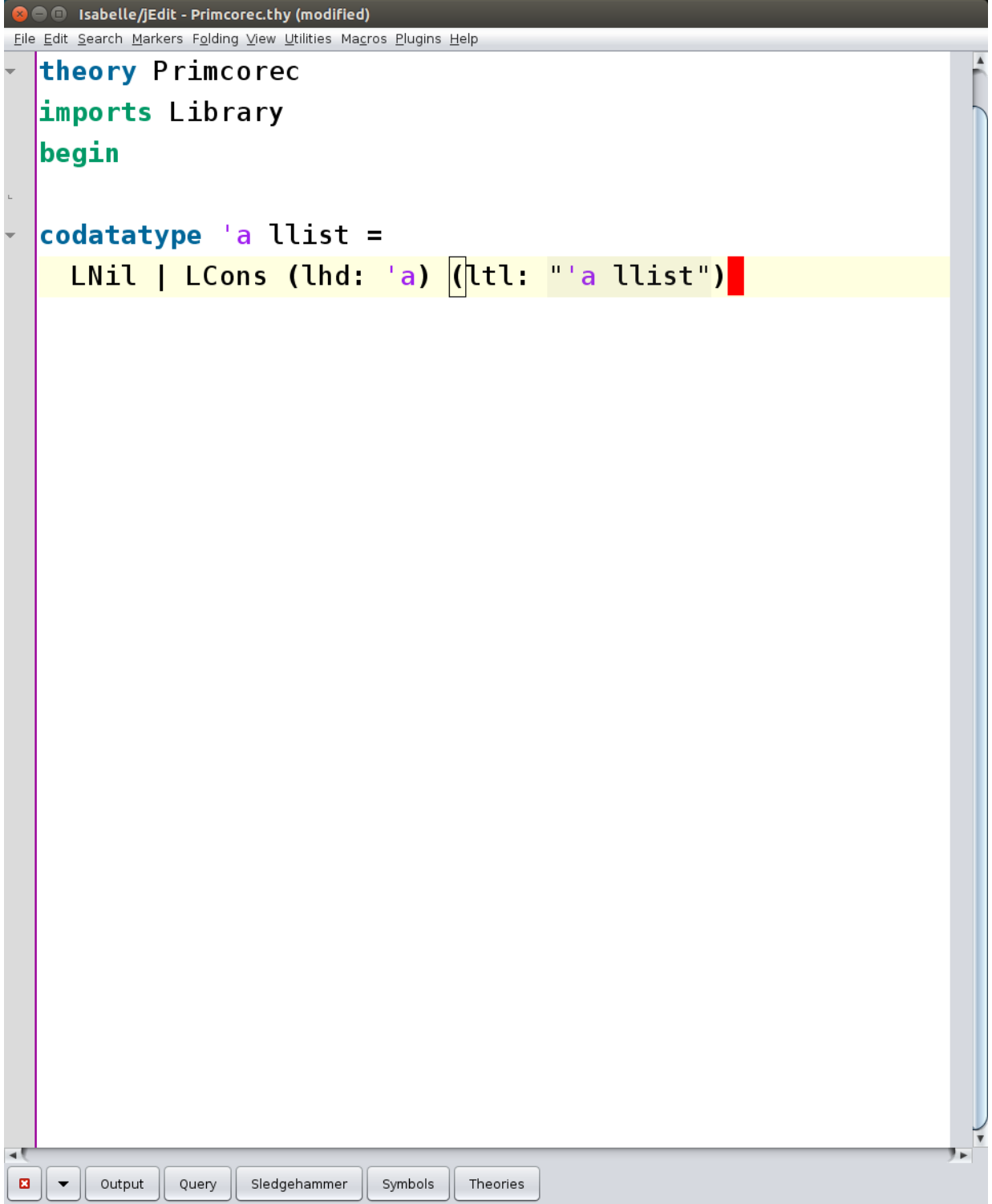

Isabelle/jEdit - Codatatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Codatatypes
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
term "dtor_corec_llist ::
('b ⇒ unit + 'a × ('a llist + 'b)) ⇒ 'b ⇒ 'a llist"
term "corec_llist ::
('b ⇒ bool) ⇒ (* is LNil? *)
('b ⇒ 'a) ⇒ (* lhd *)
('b ⇒ bool) ⇒ (* ltl: stop? *)
('b ⇒ 'a llist) ⇒ (* ltl: end *)
('b ⇒ 'b) ⇒ (* ltl: continue *)
'b ⇒ 'a llist"
thm llist.disc_corec llist.sel_corec
primcorec ltails :: "'a llist ⇒ 'a llist llist" where
Output
Query
Sledgehammer
Symbols
Theories

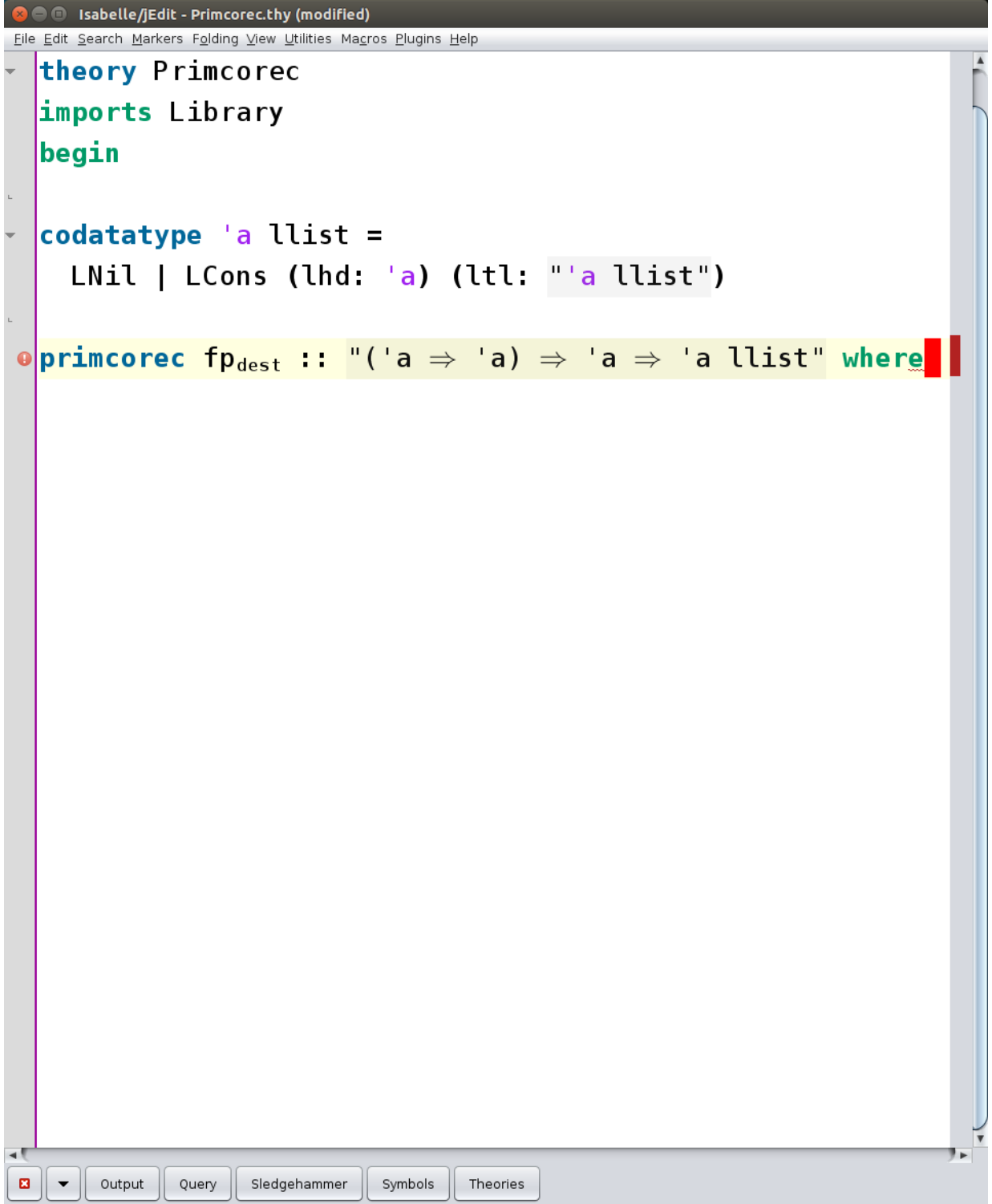

Isabelle/jEdit - Codatatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Codatatypes
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
term "dtor_corec_llist ::
('b ⇒ unit + 'a × ('a llist + 'b)) ⇒ 'b ⇒ 'a llist"
term "corec_llist ::
('b ⇒ bool) ⇒ (* is LNil? *)
('b ⇒ 'a) ⇒ (* lhd *)
('b ⇒ bool) ⇒ (* ltl: stop? *)
('b ⇒ 'a llist) ⇒ (* ltl: end *)
('b ⇒ 'b) ⇒ (* ltl: continue *)
'b ⇒ 'a llist"
thm llist.disc_corec llist.sel_corec
primcorec ltails :: "'a llist ⇒ 'a llist llist" where
"ltails xs ≠ LNil"
Output
Query
Sledgehammer
Symbols
Theories

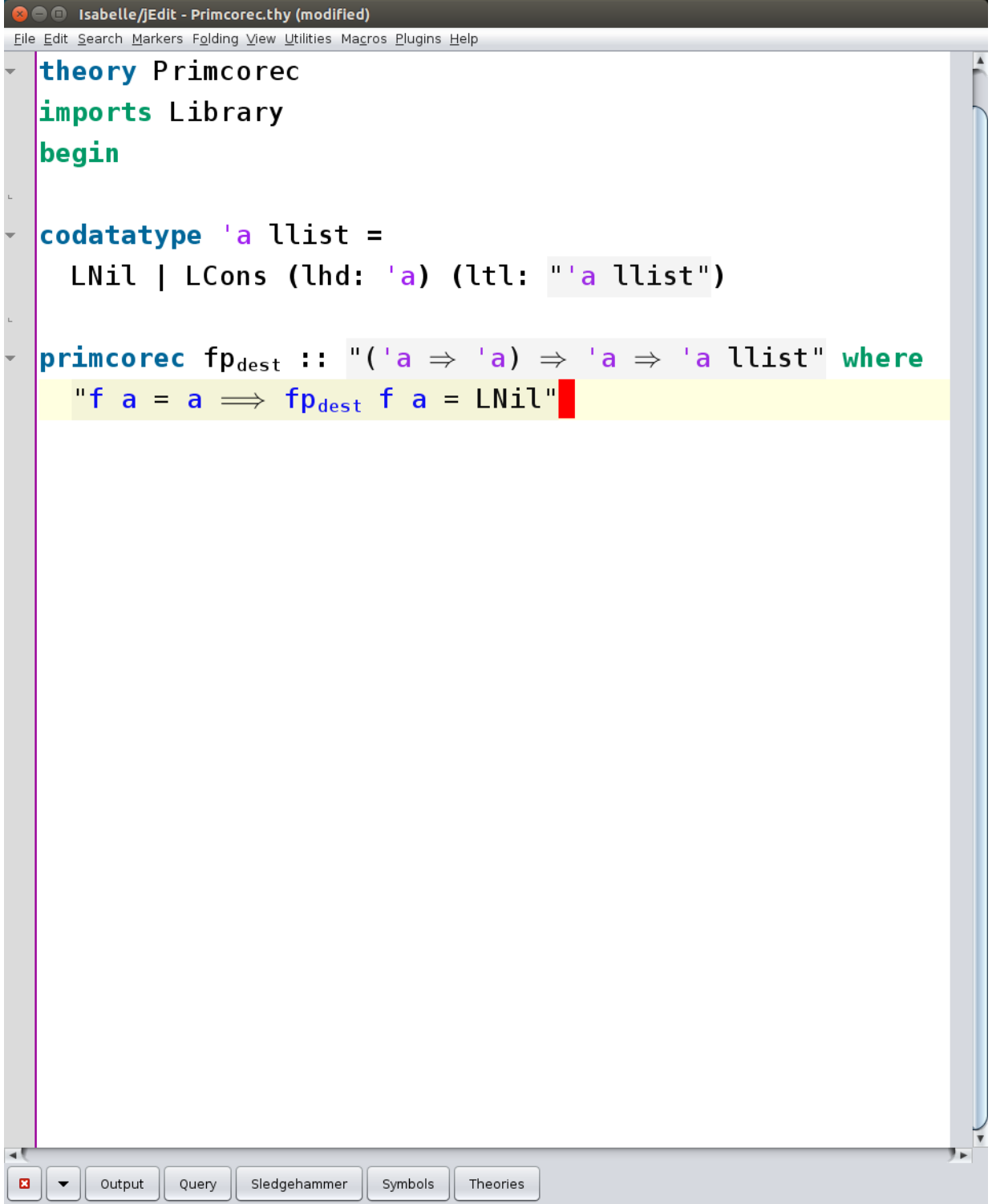

Isabelle/jEdit - Codatatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Codatatypes
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
term "dtor_corec_llist ::
('b ⇒ unit + 'a × ('a llist + 'b)) ⇒ 'b ⇒ 'a llist"
term "corec_llist ::
('b ⇒ bool) ⇒ (* is LNil? *)
('b ⇒ 'a) ⇒ (* lhd *)
('b ⇒ bool) ⇒ (* ltl: stop? *)
('b ⇒ 'a llist) ⇒ (* ltl: end *)
('b ⇒ 'b) ⇒ (* ltl: continue *)
'b ⇒ 'a llist"
thm llist.disc_corec llist.sel_corec
primcorec ltails :: "'a llist ⇒ 'a llist llist" where
"ltails xs ≠ LNil"
| "lhd (ltails xs) = xs"
Output
Query
Sledgehammer
Symbols
Theories

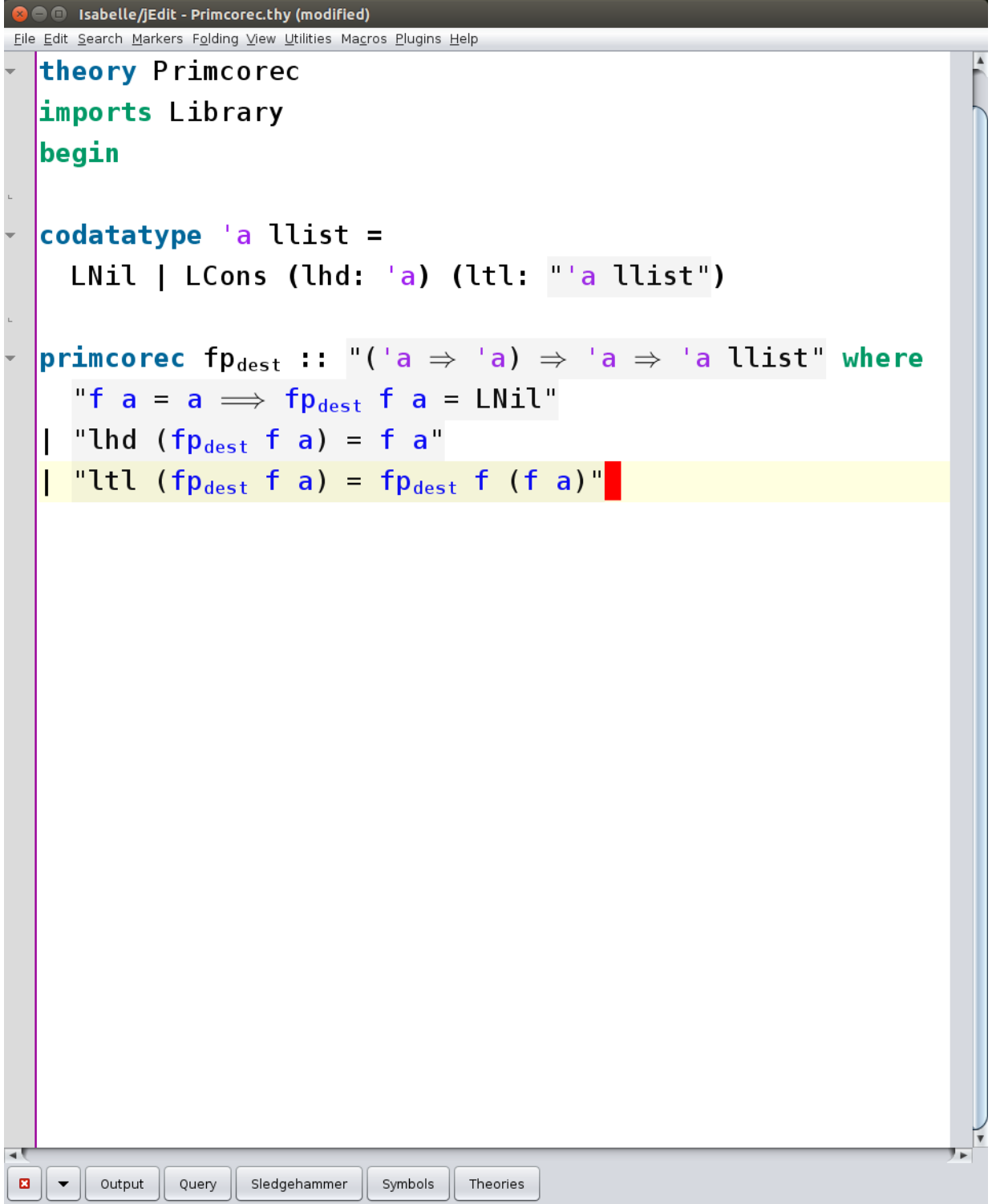

Isabelle/jEdit - Codatatypes.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Codatatypes
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
term "dtor_corec_llist ::
('b ⇒ unit + 'a × ('a llist + 'b)) ⇒ 'b ⇒ 'a llist"
term "corec_llist ::
('b ⇒ bool) ⇒ (* is LNil? *)
('b ⇒ 'a) ⇒ (* lhd *)
('b ⇒ bool) ⇒ (* ltl: stop? *)
('b ⇒ 'a llist) ⇒ (* ltl: end *)
('b ⇒ 'b) ⇒ (* ltl: continue *)
'b ⇒ 'a llist"
thm llist.disc_corec llist.sel_corec
primcorec ltails :: "'a llist ⇒ 'a llist llist" where
"ltails xs ≠ LNil"
| "lhd (ltails xs) = xs"
| "ltl (ltails xs) =
(if xs = LNil then LNil else ltails (ltl xs))"
Output
Query
Sledgehammer
Symbols
Theories

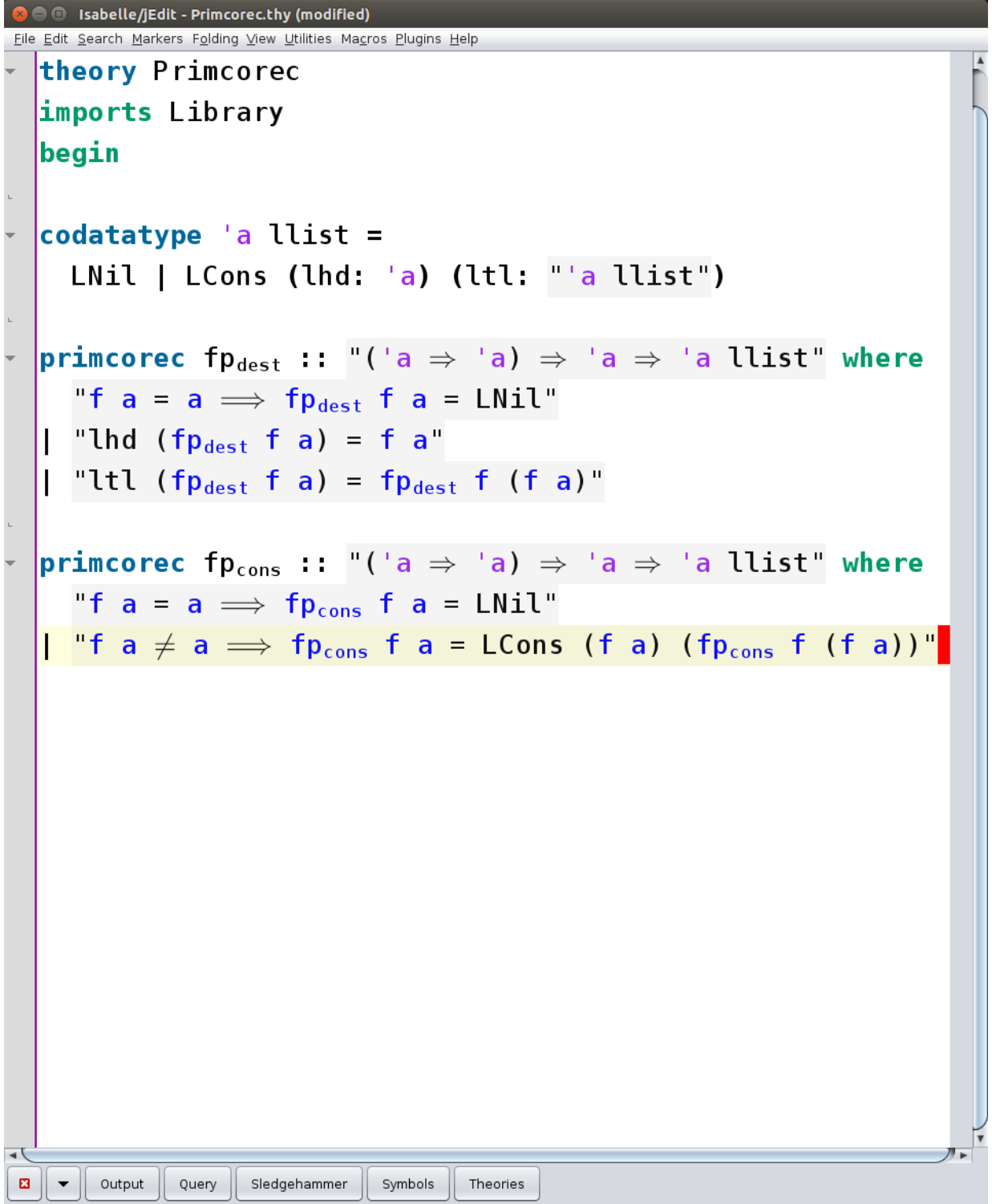
Outline

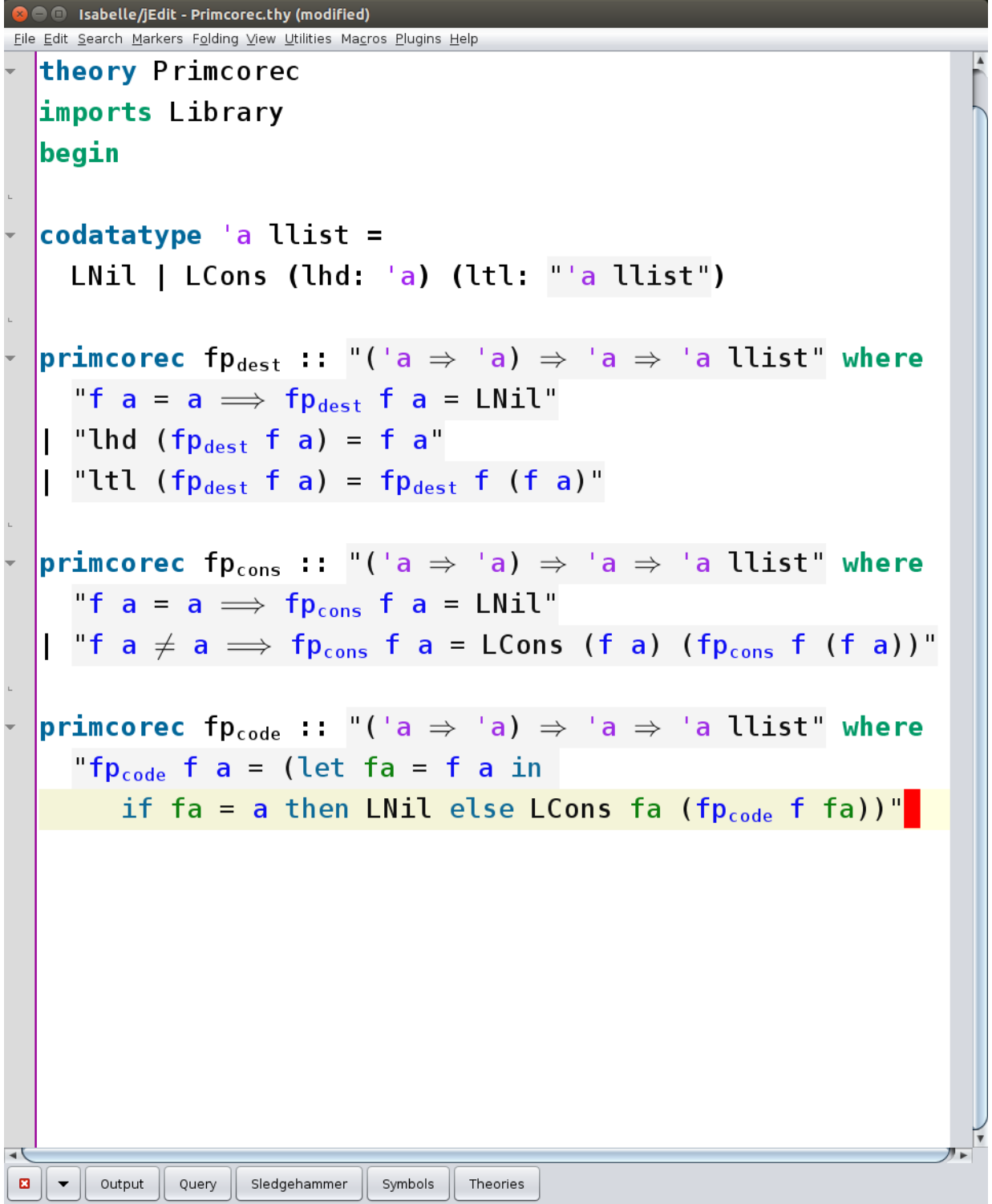

Isabelle/jEdit - Primcorec.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Primcorec
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
Output
Query
Sledgehammer
Symbols
Theories

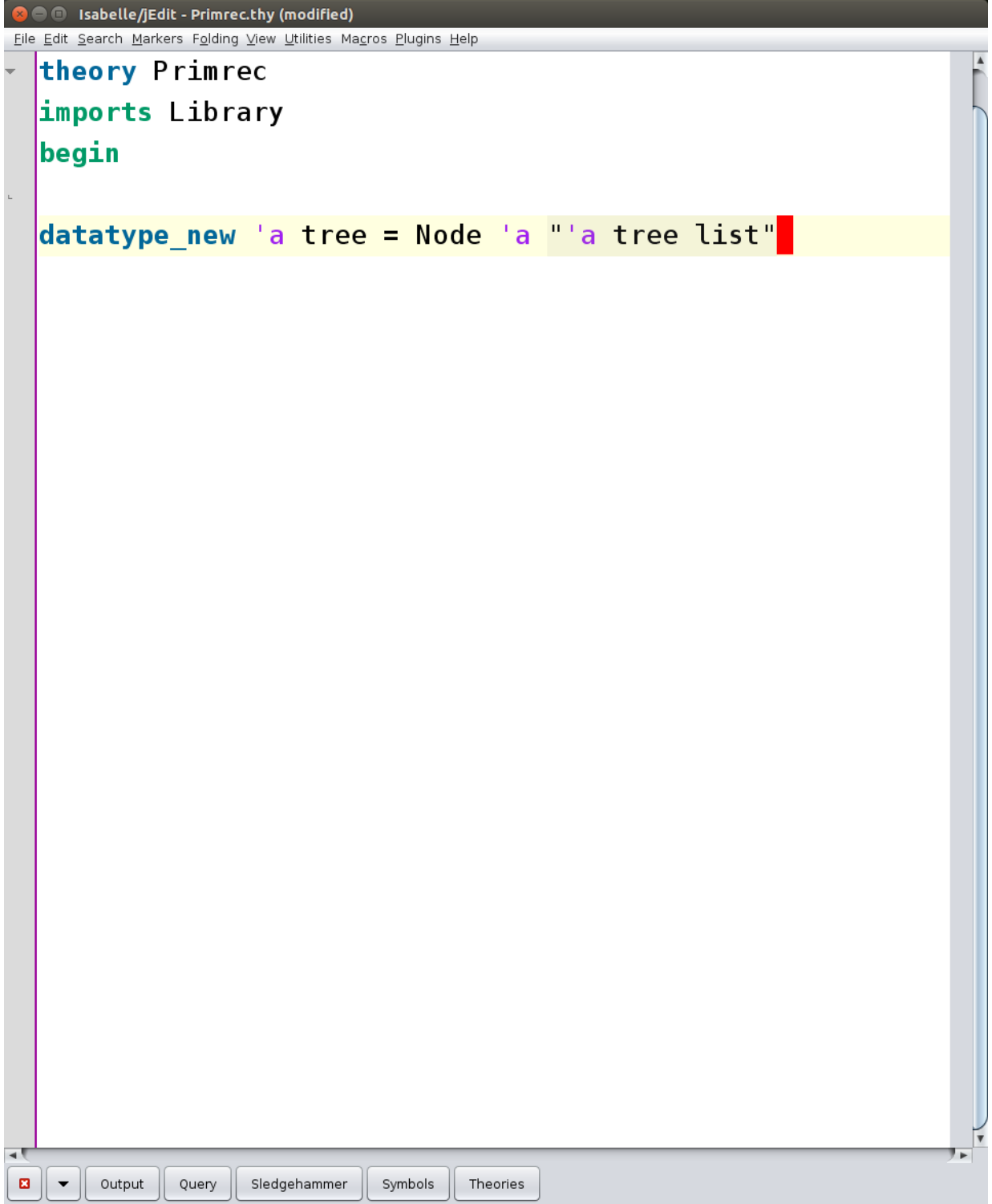

Isabelle/jEdit - Primcorec.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Primcorec
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
primcorec fp_dest :: "('a ⇒ 'a) ⇒ 'a ⇒ 'a llist" where
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Primcorec.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Primcorec
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
primcorec fp_dest :: "('a ⇒ 'a) ⇒ 'a ⇒ 'a llist" where
"f a = a ⟹ fp_dest f a = LNil"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Primcorec.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Primcorec
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
primcorec fp_dest :: "('a ⇒ 'a) ⇒ 'a ⇒ 'a llist" where
"f a = a ⟹ fp_dest f a = LNil"
| "lhd (fp_dest f a) = f a"
| "ltl (fp_dest f a) = fp_dest f (f a)"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Primcorec.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Primcorec
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
primcorec fp_dest :: "('a ⇒ 'a) ⇒ 'a ⇒ 'a llist" where
"f a = a ⟹ fp_dest f a = LNil"
| "lhd (fp_dest f a) = f a"
| "ltl (fp_dest f a) = fp_dest f (f a)"
primcorec fp_cons :: "('a ⇒ 'a) ⇒ 'a ⇒ 'a llist" where
"f a = a ⟹ fp_cons f a = LNil"
| "f a ≠ a ⟹ fp_cons f a = LCons (f a) (fp_cons f (f a))"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Primcorec.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Primcorec
imports Library
begin
codatatype 'a llist =
LNil | LCons (lhd: 'a) (ltl: "'a llist")
primcorec fp_dest :: "('a ⇒ 'a) ⇒ 'a ⇒ 'a llist" where
"f a = a ⟹ fp_dest f a = LNil"
| "lhd (fp_dest f a) = f a"
| "ltl (fp_dest f a) = fp_dest f (f a)"
primcorec fp_cons :: "('a ⇒ 'a) ⇒ 'a ⇒ 'a llist" where
"f a = a ⟹ fp_cons f a = LNil"
| "f a ≠ a ⟹ fp_cons f a = LCons (f a) (fp_cons f (f a))"
primcorec fp_code :: "('a ⇒ 'a) ⇒ 'a ⇒ 'a llist" where
"fp_code f a = (let fa = f a in
if fa = a then LNil else LCons fa (fp_code f fa))"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Primrec.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Primrec
imports Library
begin
datatype_new 'a tree = Node 'a "'a tree list"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Primrec.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Primrec
imports Library
begin
datatype_new 'a tree = Node 'a "'a tree list"
primrec sum_tree :: "nat tree ⇒ nat" where
"sum_tree (Node a ts) =
a + sum_list (map sum_tree ts)"
Output
Query
Sledgehammer
Symbols
Theories


Isabelle/jEdit - Primrec.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
theory Primrec
imports Library
begin
datatype_new 'a tree = Node 'a "'a tree fset"
primrec sum_tree :: "nat tree ⇒ nat" where
"sum_tree (Node a ts) =
a + sum_fset (fimage sum_tree ts)"
Output
Query
Sledgehammer
Symbols
Theories

Outline

Some Numbers

development size

19 kLoC ML, 7 kLoC thy

Some Numbers

development size

19 kLoC ML, 7 kLoC thy

automation gains

In the Coinductive library,
we now prove 36% more
lemmas in 11% fewer lines

Some Numbers

development size

19 kLoC ML, 7 kLoC thy

performance: mutual (m = 15)

$\frac{\text{new}}{\text{old}} \approx 2.5$

automation gains

In the Coinductive library,
we now prove 36% more
lemmas in 11% fewer lines

Some Numbers

development size

19 kLoC ML, 7 kLoC thy

automation gains

In the Coinductive library,
we now prove 36% more
lemmas in 11% fewer lines

performance: mutual (m = 15)

$\frac{\text{new}}{\text{old}} \approx 2.5$

performance: nested (n = 15)

$\frac{\text{new}}{\text{old}} = 0.04$

Some Numbers

development size

19 kLoC ML, 7 kLoC thy

automation gains

In the Coinductive library, we now prove 36% more lemmas in 11% fewer lines

performance: mutual (m = 15)

$\frac{\text{new}}{\text{old}} \approx 2.5$

performance: nested (n = 15)

$\frac{\text{new}}{\text{old}} = 0.04$

performance: real world

The IsaFoR session Proof-Checker now compiles in 10 minutes instead of 50!

René Thiemann

Related Talks

Lochbihler, Hölzl (next talk)

Recursive Functions on Lazy Lists via Domains and Topologies

Blanchette, Popescu, Traytel (tomorrow 12:15)

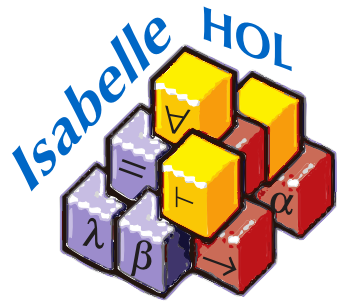
Cardinals in Isabelle/HOL

Blanchette, Popescu, Traytel (Saturday 11:45, IJCAR)

Unified Classical Logic Completeness: A Coinductive Pearl

Truly Modular (Co)datatypes

for



Jasmin Blanchette
Lorenz Panny
Dmitriy Traytel

Johannes Hölzl
Andrei Popescu

Technische Universität München

Andreas Lochbihler

ETH zürich