

ProofViz

An Interactive Visual Proof Explorer

Daniel Melcer, Stephen Chang
Northeastern University, University of Massachusetts Boston
TFP/Lambda Days February 18, 2021

Theorem Proving (as an undergraduate student)

```
Theorem addleq : forall (n : nat)
(j : nat), (n + j).+1 = (n + j.+1).
```

```
1 goal
```

```
⋀ n j : ℕ, eq (S (addn n j)) (addn n (S j))
```

Theorem Proving (as an undergraduate student)

```
Theorem addleq : forall (n : nat)
(j : nat), (n + j).+1 = (n + j.+1).
intros n j.
```

```
1 goal
  n, j : ℕ
  -----
  eq (S (addn n j)) (addn n (S j))
```

Theorem Proving (as an undergraduate student)

```
Theorem addleq : forall (n : nat)
(j : nat), (n + j).+1 = (n + j.+1).
intros n j.
elim n.
```

2 goals

$n, j : \mathbb{N}$

$\text{eq } (S \text{ (addn 0 j)}) \text{ (addn 0 (S j))}$

subgoal 2 is:

$\forall (n : \mathbb{N}) (_ : \text{eq } (S \text{ (addn n j)}) \text{ (addn n (S j))}),$
 $\text{eq } (S \text{ (addn (S n) j)}) \text{ (addn (S n) (S j))}$

Theorem Proving (as an undergraduate student)

```
Theorem addleq : forall (n : nat)
(j : nat), (n + j).+1 = (n + j.+1).
intros n j.
elim n.
reflexivity.
```

```
1 goal
  n, j : ℕ
  -----
  ∀ (n : ℕ) (_ : eq (S (addn n j)) (addn n (S j))),
  eq (S (addn (S n) j)) (addn (S n) (S j))
```

Theorem Proving (as an undergraduate student)

```
Theorem addleq : forall (n : nat)
(j : nat), (n + j).+1 = (n + j.+1).
intros n j.
elim n.
reflexivity.
intros n1 IH.
```

```
1 goal
n, j, n1 : ℕ
IH : eq (S (addn n1 j)) (addn n1 (S j))
-----
eq (S (addn (S n1) j)) (addn (S n1) (S j))
```

Theorem Proving (as an undergraduate student)

```
Theorem addleq : forall (n : nat)
  (j : nat), (n + j).+1 = (n + j.+1).
intros n j.
elim n.
reflexivity.
intros n1 IH.
simpl.
```

1 goal

```
n, j, n1 : ℕ
IH : eq (S (addn n1 j)) (addn n1 (S j))
-----
eq (S (addn (S n1) j)) (addn (S n1) (S j))
```

Theorem Proving (as an undergraduate student)

```
Theorem addleq : forall (n : nat)
  (j : nat), (n + j).+1 = (n + j.+1).
intros n j.
elim n.
reflexivity.
intros n1 IH.
simpl.
have why_doesnt_simpl_do_this :
  (addn (S n1) (S j)) = (S (addn n1
    (S j))).
done.
```

```
1 goal
  n, j, n1 : ℕ
  IH : eq (S (addn n1 j)) (addn n1 (S j))
  why_doesnt_simpl_do_this : eq (addn (S n1) (S j))
    (S (addn n1 (S j)))
  -----
  eq (S (addn (S n1) j)) (addn (S n1) (S j))
```

Theorem Proving (as an undergraduate student)

```
Theorem addleq : forall (n : nat)
  (j : nat), (n + j).+1 = (n + j.+1).
```

```
intros n j.
```

```
elim n.
```

```
reflexivity.
```

```
intros n1 IH.
```

```
simpl.
```

```
have why_doesnt_simpl
```

```
(addn (S n1) (S j)) = (S (addn n1
```

```
(S j))).
```

```
done.
```

```
rewrite why_doesnt_simpl_do_this.
```

```
1 goal
```

```
n, j, n1 : ℕ
```

```
IH : eq (S (addn n1 j)) (addn n1 (S j))
```

```
  this : eq (addn (S n1) (S j))
```

```
    )) (S (addn n1 (S j)))
```

Where is the proof term?

Tactics (and tactic-based proof assistants)

- + Track proof state
- + Automation
- Opaque for beginners
- No intuition about the proof itself

Example: *The Little Typer*¹

What if we *only* focus on the proof term?

¹D. Friedman, D. Christiansen (2018)

Example: *The Little Typer*¹

```
(define step-add1+=+add1
  (λ (j n-1)
    (λ (add1+=+add1n-1)
      (cong add1+=+add1n-1
        (+ 1) ) ) ) )
```

¹D. Friedman, D. Christiansen (2018)

Example: *The Little Typer*

```
(define step-add1+=+add1
  (λ (j n-1)
    (λ (add1+=+add1n-1)
      (cong add1+=+add1n-1
        (+ 1))))))
```

```
(define add1+=+add1
  (λ (n j)
    (ind-Nat n
      (mot-add1+=+add1 j)
      (same (add1 j))
      (step-add1+=+add1 j))))
```



Proof Strategies

```
(define step-add1+=+add1
  (λ (j n-1)
    (λ (add1+=+add1n-1)
      (cong add1+=+add1
        (+ 1))))))
```

```
(define add1+=+add1
  (λ (n j)
    (ind-Nat n
      (mot-add1+=+add1 j)
      (same (add1 j))
      (step-add1+=+add1 j))))
```

```
Theorem add1eq : forall (n : nat)
  (j : nat), (n + j).+1 = (n + j.+1).
  intros n j.
```

How are these
connected?

```
  (S j)) = (S (addn n1
    (S j))).  reflexivity.
  rewrite why_doesnt_simpl_do_this.
  by f_equal.
Qed.
```

Proof Strategies

```
(define step-add1+=+add1
  (λ (j n-1)
    (λ (add1+=+add1n-1)
      (cong add1+=+add1n-1
        (+ 1))))))
```

```
(define add1+=+add1
  (λ (n j)
    (ind-Nat n
      (mot-add1+=+add1 j)
      (same (add1 j))
      (step-add1+=+add1 j))))
```

```
Theorem add1eq : forall (n : nat)
  (j : nat), (n + j).+1 = (n + j.+1).
  intros n j.
  elim n.
  reflexivity.
  intros n1 IH.
  have why_doesnt_simpl_do_this :
    (addn (S n1) (S j)) = (S (addn n1
      (S j))). reflexivity.
  rewrite why_doesnt_simpl_do_this.
  by f_equal.
Qed.
```

Proof Strategies

```
(define step-add1+=+add1  
  (λ (j) n-1)  
    (λ (add1+=+add1 n-1)  
      (cong add1+=+add1 n-1  
        (+ 1))))))
```

```
(define add1+=+add1  
  (λ (n j) n j)  
    (ind-Nat n  
      (mot-add1+=+add1 j)  
      (same (add1 j)) (same (add1 j))  
      (step-add1+=+add1 j))))
```

```
Theorem add1eq : forall (n : nat)  
  (j : nat), (n + j).+1 = (n + j.+1).  
  intros n j.  
  elim n.  
  reflexivity.  
  intros n1 IH.  
  have why_doesnt_simpl_do_this :  
    (addn (S n1) (S j)) = (S (addn n1  
      (S j))). reflexivity.  
  rewrite why_doesnt_simpl_do_this.  
  by f_equal.  
Qed.
```

Bottom-up proofs

- + Explicit
- + Transparent
- Verbose- no automation
- Most proofs aren't book chapters

Introducing ProofViz

The screenshot displays the GUI Proof Explorer interface, which is used for visualizing and managing proofs in a theorem prover. The interface is divided into several panels:

- Left Panel (Proof Tree):** Shows a hierarchical view of the proof. The root is `⋄(Focused) Top Level`. It branches into `⋄(λ (n : Nat) (Subterm 0))`, which further branches into `⋄Context: (n : Nat)`, `⋄(λ (j : Nat) (Subterm 0))`, `⋄Context: (j : Nat)`, `⋄((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))`, `⋄(λ (λ (Subterm 0)))`, `⋄Context:`, `⋄(λ (Subterm 0))`, `⋄Context:`, `(refl Nat (s j)) : (== Nat (s j) (s j))`, `⋄(λ X1 IH3 (λ (Subterm 0)))`, `⋄Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))`, `⋄(f-equal Nat Nat (λ X1 (s X1)) (s (plus X1 j)) (plus X1 (s j)) (Subterm 0))`, and `IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j)))`.
- Context Panel:** Displays the current context, showing `n : Nat`.
- Goal Panel:** Shows the current goal, which is `(Π (j : Nat) (== Nat (s (plus n j)) (plus n (s j))))`.
- Apply Subterms Panel:** Shows the subterms of the goal, with `0 : (== Nat (s (plus n j)) (plus n (s j)))`.
- Result Panel:** Shows the result of the current step, which is `(λ (j : Nat) (Subterm 0))`.
- Complete Proof Term Panel:** Shows the complete proof term, which is `(λ (j : Nat) ((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (λ (λ (λ (refl Nat (s j))))) (λ X1`.
- Generated by tactic Panel:** Shows the tactic used to generate the result, which is `(by-intros n j)`.
- Tactic Panel:** Lists available tactics, including `(by-intros n j)`, `(by-induction n)`, `reflexivity`, `(by-apply f-equal #::with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))`, and `by-assumption` (which is highlighted).

At the bottom of the interface, there are buttons for `Expand all`, `Collapse all except focus`, `Undo`, and `Redo`.

ProofViz: Tree View

The screenshot displays the ProofViz Tree View interface. The main area shows a proof tree for a theorem. The tree is structured as follows:

- ▼(Focused) Top Level
 - ▼(λ (n : Nat) (Subterm 0))
 - ▼Context: (n : Nat)
 - ▼(λ (j : Nat) (Subterm 0))
 - ▼Context: (j : Nat)
 - ▼((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))
 - ▼(λ (λ (Subterm 0)))
 - ▼Context:
 - ▼(λ (Subterm 0))
 - ▼Context:
 - (refl Nat (s j)) : (== Nat (s j) (s j))
 - ▼(λ X1 IH3 (λ (Subterm 0)))
 - ▼Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))
 - ▼(f-equal Nat Nat (λ X1 (s X1)) (s (plus X1 j)) (plus X1 (s j)) (Subterm 0))
 - IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j)))

The right sidebar shows the current goal: `Nat s (s (plus X1 j)) (plus X1 (s j))`. Below the goal is an `Undo` button. The bottom of the interface includes buttons for `Expand all`, `Collapse all except focus`, and `Redo`.

ProofViz: Node Information

The screenshot displays the ProofViz interface, which provides a visual representation of a proof state. A red rectangle highlights the central area containing the Context, Goal, and Apply Subterms panels.

Left Panel (Proof Tree): Shows the hierarchical structure of the proof. The root is `⋄(Focused) Top Level`. It branches into `⋄(λ (n : Nat) (Subterm 0))`, which further branches into `⋄Context: (n : Nat)`, `⋄(λ (j : Nat) (Subterm 0))`, `⋄Context: (j : Nat)`, `⋄((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j)))) (λ (λ (Subterm 0))))`, `⋄Context:`, `⋄(λ (Subterm 0))`, `⋄Context:`, `(refl Nat (s j)) : (== Nat (s j)) (s j))`, `⋄(λ X1 IH3 (λ (Subterm 0)))`, `⋄Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus n (s j))))`, `⋄(f-equal Nat Nat (λ X1 (s X1)) (s (plus X1 j)) (plus n (s j)))`, and `IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j)))`.

Context Panel: Displays the current context, showing `n : Nat`.

Goal Panel: Displays the current goal, showing `(Π (j : Nat) (== Nat (s (plus n j)) (plus n (s j))))`.

Apply Subterms Panel: Displays the subterms being applied, showing `0 : (== Nat (s (plus n j)) (plus n (s j)))`.

Bottom Panel: Shows the command `(by-intros n j)` generated by the system.

Right Panel: Shows the current state of the proof, with a blue highlight on the goal statement `(Nat Nat s (s (plus X1 j)) (plus X1 (s j)))`.

Buttons: The interface includes buttons for `Undo` and `Redo` at the bottom right, and `Expand all` and `Collapse all except focus` at the bottom left.

ProofViz: Node Information

GUI Proof Explorer

▼(Focused) Top Level

- ▼(λ (n : Nat) (Subterm 0))
 - ▼Context: (n : Nat)
 - ▼[λ (j : Nat)] (Subterm 0))
 - ▼Context: (j : Nat)
 - ▼((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j)))))) (Subterm 0) (Subterm 1)))
 - ▼(λ (λ (Subterm 0)))
 - ▼Context:
 - ▼(λ (Subterm 0))
 - ▼Context:
 - (refl Nat (s j)) : (== Nat (s j) (s j))
- ▼(λ X1 IH3 (λ (Subterm 0)))
 - ▼Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))
 - ▼(f-equal Nat Nat (λ X1 (s X1)) (s (plus X1 j)) (plus X1 (s j)))
 - IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j)))

Context

```
n      :    Nat
```

Goal

Tactic

- (by-intros n j)
- (by-induction n)
- reflexivity
- (by-apply f-equal #:with Nat s (s (plus X1 j)) (plus X1 (s j)))
- by-assumption

Result

```
(λ (j : Nat) (Subterm 0))
```

Complete Proof Term

```
(λ (j : Nat)
  ((new-elim
    n
    (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))))
   (λ (λ (λ (refl Nat (s j)))))
   (λ X1
```

Generated by tactic

```
(by-intros n j)
```

Undo

Redo

Expand all
Collapse all

ProofViz: Interaction

The screenshot displays the GUI Proof Explorer interface, which is used for interacting with a proof. The interface is divided into several panels:

- Left Panel (Proof Tree):** Shows a hierarchical view of the proof. The root is `⊢ (Focused) Top Level`. It branches into `⊢ (λ (n : Nat) (Subterm 0))`, which further branches into `⊢ Context: (n : Nat)`, `⊢ (λ (j : Nat) (Subterm 0))`, and `⊢ Context: (j : Nat)`. The `⊢ Context: (j : Nat)` node is highlighted in blue. Below it, the proof tree continues with `⊢ ((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j)))) (Subterm 0) (Subterm 1))))`, `⊢ (λ (λ (Subterm 0)))`, `⊢ Context: (λ (Subterm 0))`, `⊢ Context: (λ (Subterm 0))`, `⊢ Context: (refl Nat (s j)) : (== Nat (s j)) (s j))`, `⊢ (λ X1 IH3 (λ (Subterm 0)))`, `⊢ Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))`, `⊢ (f-equal Nat Nat (λ X1 (s X1)) (s (plus X1 j)) (plus X1 (s j)) (Subterm 0))`, and `IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j)))`.
- Context Panel:** Shows the current context, which is `n : Nat`.
- Goal Panel:** Shows the current goal, which is `(Π (j : Nat) (== Nat (s (plus n j)) (plus n (s j)))) (Subterm 0) (Subterm 1))`.
- Apply Subterms Panel:** Shows the subterms of the goal, which are `0 : (== Nat (s (plus n j)) (plus n (s j)))` and `1 : (== Nat (s (plus n j)) (plus n (s j)))`.
- Result Panel:** Shows the result of the tactic, which is `(λ (j : Nat) (Subterm 0))`.
- Complete Proof Term Panel:** Shows the complete proof term, which is `(λ (j : Nat) (new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j)))) (Subterm 0) (Subterm 1)) (λ (λ (Subterm 0)) (λ (λ (refl Nat (s j)) (s j)) (Subterm 0)) (λ X1 IH3 (λ (Subterm 0)) (f-equal Nat Nat (λ X1 (s X1)) (s (plus X1 j)) (plus X1 (s j)) (Subterm 0)) IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))`.
- Generated by tactic Panel:** Shows the tactic used to generate the result, which is `(by-intros n j)`.
- Tactics Panel:** A list of available tactics, including `(by-intros n j)`, `(by-induction n)`, `reflexivity`, `(by-apply f-equal #.with Nat Nat s (plus X1 j)) (plus X1 (s j)))`, and `by-assumption` (which is highlighted in blue).
- Buttons:** `Undo` and `Redo` buttons are located at the bottom right of the interface.

ProofViz: Goals

- + Explicit
- + Transparent

ProofViz: Goals

- + Explicit
- + Transparent
- + Automation with tactics
- + Track proof state

Cur

- A dependently-typed proof assistant¹
 - Created with Racket²

¹Dependent Type Systems as Macros, S. Chang et al. (2019)

²The Racket Manifesto, M. Felleisen et al. (2015)

Cur

- A dependently-typed proof assistant¹
 - Created with Racket²
- Extensible (experiment without core language modification)
 - New tactic systems: Ntac
 - Experimental type systems: Sized types
 - Solver integration

¹Dependent Type Systems as Macros, S. Chang et al. (2019)

²The Racket Manifesto, M. Felleisen et al. (2015)

Cur

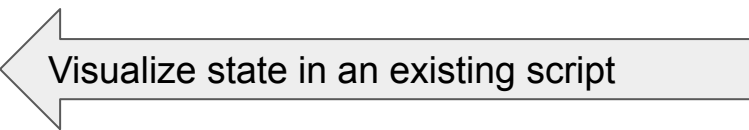
- A dependently-typed proof assistant¹
 - Created with Racket²
- Extensible (experiment without core language modification)
 - New tactic systems: Ntac
 - Experimental type systems: Sized types
 - Solver integration
 - This presentation: ProofViz

¹Dependent Type Systems as Macros, S. Chang et al. (2019)

²The Racket Manifesto, M. Felleisen et al. (2015)

Invoking ProofViz

```
(define add1+=+add1
  (ntac (Π (n : Nat)
    (j : Nat)
    (== Nat
      (s (plus n j))
      (plus n (s j))))))
display-focus-tree
```



Visualize state in an existing script

Invoking ProofViz

```
(define add1+=+add1
  (ntac/visual (Π (n : Nat)
    (j : Nat)
    (== Nat
      (s (plus n j))
      (plus n (s j))))))
```



Develop proofs in a visual environment

Invoking ProofViz

```
(define add1+=+add1
  (ntac/visual (Π (n : Nat)
    (j : Nat)
    (== Nat
      (s (plus n j))
      (plus n (s j))))))
  (by-intros n j)
  (by-induction n)
  reflexivity
  (by-apply f-equal #:with Nat Nat s (s (plus X1 j))
    (plus X1 (s j)))
  by-assumption))
```



Pre-fill the undo buffer

GUI Proof Explorer

Top Level

Focused

$$(\lambda (n : \text{Nat}) (j : \text{Nat}) == \text{Nat } (s \text{ (plus } n \text{ j)}) \text{ (plus } n \text{ (s j))}))$$

Context

Goal

$$(\lambda (n : \text{Nat}) (j : \text{Nat}) == \text{Nat } (s \text{ (plus } n \text{ j)}) \text{ (plus } n \text{ (s j))}))$$

Hole

Focus here

Generated by tactic

Passed into tactic

Tactic

Undo

(by-intros n j)

(by-induction n)

reflexivity

(by-apply f-equal #:=with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))

by-assumption

Redo

Expand all

Collapse all except focus

Top Level

Top Level

$\forall (\lambda (n : \text{Nat}) (\text{Subterm } 0))$

$\forall \text{Context: } (n : \text{Nat})$

$\forall (\lambda (j : \text{Nat}) (\text{Subterm } 0))$

$\forall \text{Context: } (j : \text{Nat})$

$(\text{Focused}) (== \text{Nat } (s \text{ (plus } n \text{ j)}) \text{ (plus } n \text{ (s j))})$

(by-intros n j)

Undo

Focus here

(by-induction n)

reflexivity

(by-apply f-equal #:=with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))

by-assumption

Generated by tactic

(by-intros n j)

Expand all

Collapse all except focus

Redo

GUL Proof Explorer

Top Level

- λ (n : Nat) (Subterm 0)
 - Context: (n : Nat)
 - λ (j : Nat) (Subterm 0)
 - Context: (j : Nat)
(Focused) (== Nat (s (plus n j)) (plus n (s j)))

Subterms

0 : (== Nat (s (plus n j)) (plus n (s j)))

Result

λ (j : Nat) (Subterm 0)

Generated by tactic

(by-intros n j)

Tactic

(by-intros n j)

(try-induction n)
reflexivity
(by-apply f-equal #with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))
by-assumption

Undo

Expand all

Collapse all except focus

Redo

GUL Proof Explorer

Top Level

- λ (n : Nat) (Subterm 0))
- Context: (n : Nat)
- λ (j : Nat) (Subterm 0))
- Context: (j : Nat)
- (Focused) (== Nat (s (plus n j)) (plus n (s j)))

Context

n : Nat

Tactic

(by-intros n j)

Apply

Subterms

0 : (== Nat (s (plus n j)) (plus n (s j)))

Result

(λ (j : Nat) (Subterm 0))

Generated by tactic

(by-intros n j)

Expand all

Undo

Redo

GUL Proof Explorer

Top Level

Top Level

$\lambda (n : \text{Nat})$ (Subterm 0))

Context: $(n : \text{Nat})$

$\lambda (j : \text{Nat})$ (Subterm 0))

Context: $(j : \text{Nat})$

$((\text{new-elim } n \ (\lambda n \ (\Pi \ (== \text{Nat } (s \ (\text{plus } n \ j)) \ (\text{plus } n \ (s \ j)))))) \ (\text{Subterm } 0) \ (\text{Subterm } 1)))$

$\lambda (\lambda \ (\text{Subterm } 0)))$

Context:

$(\text{Focused}) \ (== \text{Nat } (s \ j) \ (s \ j))$

$\lambda X1 \text{ IH3 } (\lambda \ (\text{Subterm } 0)))$

Context: $(X1 : \text{Nat}) \ (\text{IH3} : (== \text{Nat } (s \ (\text{plus } X1 \ j)) \ (\text{plus } X1 \ (s \ j))))$

$(== \text{Nat } (s \ (s \ (\text{plus } X1 \ j))) \ (s \ (\text{plus } X1 \ (s \ j))))$

Context

$(n : \text{Nat})$

Tactic

$(\text{by-intros } n \ j)$

$(\text{by-induction } n)$

Undo

Generated by tactic

$(\text{by-induction } n)$

reflexivity

$(\text{by-apply f-equal \#::with Nat Nat } s \ (s \ (\text{plus } X1 \ j)) \ (\text{plus } X1 \ (s \ j)))$

by-assumption

Expand all

Collapse all except focus

Redo

GUI Proof Explorer

Top Level
λ (n : Nat) (Subterm 0))

Context

n : Nat

Goal

(== Nat (s (plus n j)) (plus n (s j)))

Tactic

(by-intros n j)
(by-induction n)

Undo

reflexivity
(by-apply f-equal #with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))
by-assumption

Redo

Expand all

Collapse all except focus

GUI Proof Explorer

Top Level
λ (n : Nat) (Subterm 0))

Context
n : Nat

Goal
(== Nat (s (plus n j)) (plus n (s j)))

Apply
Subterms
0 : (== Nat (s (plus n j)) (plus n (s j)))
1 : (== Nat (s (plus n j)) (plus n (s j)))

Result
((new-elim
 n
 (λ n (Π (== Nat (s (plus n j)) (plus n (s j)))))
 (Subterm 0)
 (Subterm 1)))

Generated by tactic
(by-induction n)

Expand all

Collapse all except focus

GUI Proof Explorer

Top Level

Top Level

$\lambda (n : \text{Nat})$ (Subterm 0))

Context: $(n : \text{Nat})$

$\lambda (j : \text{Nat})$ (Subterm 0))

Context: $(j : \text{Nat})$

$((\text{new-elim } n \ (\lambda n \ (\Pi \ (== \text{Nat } (s \ (\text{plus } n \ j)) \ (\text{plus } n \ (s \ j)))))) \ (\text{Subterm } 0) \ (\text{Subterm } 1)))$

$\lambda (\lambda \ (\text{Subterm } 0)))$

Context:

$(\text{Focused } (== \text{Nat } (s \ j) \ (s \ j)))$

$\lambda X1 \text{ IH3 } (\lambda \ (\text{Subterm } 0)))$

Context: $(X1 : \text{Nat}) \ (\text{IH3} : (== \text{Nat } (s \ (\text{plus } X1 \ j)) \ (\text{plus } X1 \ (s \ j))))$

$(== \text{Nat } (s \ (s \ (\text{plus } X1 \ j))) \ (s \ (\text{plus } X1 \ (s \ j))))$

Tactic

$(\text{by-intros } n \ j)$

$(\text{by-induction } n)$

Undo

Generated by tactic

$(\text{by-induction } n)$

reflexivity

$(\text{by-apply } f\text{-equal } \#:\text{with Nat Nat } s \ (s \ (\text{plus } X1 \ j)) \ (\text{plus } X1 \ (s \ j)))$

by-assumption

Expand all

Collapse all except focus

Redo

GUI Proof Explorer

▼Top Level

▼(λ (n : Nat) (Subterm 0))

▼Context: (n : Nat)

▼(λ (j : Nat) (Subterm 0))

▼Top Level

▼(λ (n : Nat) (Subterm 0))

▼Context: (n : Nat)

▼(λ (j : Nat) (Subterm 0))

▼Context: (j : Nat)

▼((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))

▼(λ (λ (Subterm 0)))

▼Context:

(== Nat (s j) (s j))

▼(λ X1 IH3 (λ (Subterm 0)))

▼Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))

(Focused) (== Nat (s (s (plus X1 j))) (s (plus X1 (s j))))

Context

j : Nat

X1 : Nat

Tactic

(by-intros n j)

(by-induction n)

(navigate (path-down-done) (path-down-apply 0) (path-down-context) (path-down-apply 0) (path-down-context))

Undo

Focus here

Generated by tactic

(by-induction n)

Expand all

Collapse all except focus

Redo

GUL Proof Explorer

▼Top Level

▼(λ (n : Nat) (Subterm 0))

▼Context: (n : Nat)

▼(λ (j : Nat) (Subterm 0))

▼Top Level

▼(λ (n : Nat) (Subterm 0))

▼Context: (n : Nat)

▼(λ (j : Nat) (Subterm 0))

▼Context: (j : Nat)

▼((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j)))

▼(λ (λ (Subterm 0)))

▼Context:

(== Nat (s j) (s j))

▼(λ X1 IH3 (λ (Subterm 0)))

▼Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))

(Focused) (== Nat (s (s (plus X1 j))) (s (plus X1 (s j))))

Cont

J

X1

new

Tactic

(by-intros n j)

(by-induction n)

(navigate (path-down-done) (path-down-apply 0) (path-down-context) (path-down-apply 0) (path-down-context))

Context

Undo

Focus here

Generated by tactic

(by-induction n)

Expand all

Collapse all except focus

Redo

▼To
▼

▼Top Level

▼(λ (n : Nat) (Subterm 0))

▼Context: (n : Nat)

▼(λ (j : Nat) (Subterm 0))

▼Context: (j : Nat)

▼((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))

▼(λ (λ (Subterm 0)))

▼Context:

▼(λ (Subterm 0))

▼Context:

(refl Nat (s j)) : (== Nat (s j) (s j))

▼(λ X1 IH3 (λ (Subterm 0)))

▼Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))

(Focused) (== Nat (s (s (plus X1 j))) (s (plus X1 (s j))))

Generated by tactic
(by-induction n)

Expand all

Collapse all except focus

Redo

Undo

with Nat Nat s (s (plus X1 j)) (plus X1 (s j))

Top Level
 (λ (n : Nat) (Subterm 0))
 Context: (n : Nat)
 (λ (j : Nat) (Subterm 0))
 Context: (j : Nat)
 (new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j)))) (Subterm 0) (Subterm 1)))
 (λ (λ (Subterm 0))
 Context:
 (λ (Subterm 0))
 Context:
 (refl Nat (s j)) : (== Nat (s j) (s j)))
 (λ X1 IH3 (λ (Subterm 0))
 Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))
 (f-equal Nat Nat (λ X1 (s X1)) (s (plus X1 j)) (plus X1 (s j)) (Subterm 0))
 (Focused) (== Nat (s (plus X1 j)) (plus X1 (s j)))

Context:
 j : Nat
 X1 : Nat
 IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j)))

Goal:
 (== Nat (s (plus X1 j)) (plus X1 (s j)))

Hole
 Focus here

Generated by tactic
 (by-apply f-equal #:with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))

Tactic

(by-intros n j)
 (by-induction n)
 reflexivity
 (by-apply f-equal #:with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))

Undo

by-assumption

Expand all

Collapse all except focus

Redo

(by-intros n j)

GUI Proo

$$\nabla(\lambda (n : \text{Nat}) (\text{Subterm } 0))$$

▽Context: (n : Nat)

$$\nabla(\lambda (j : \text{Nat}) (\text{Subterm } 0))$$

▽Context: (j : Nat)

$$\nabla((\text{new-elim } n \ (\lambda n \ (\prod \ (== \text{Nat } (s \ (\text{plus } n \ j)) \ (\text{plus } n \ (s \ j)))))) \ (\text{Subterm } 0) \ (\text{Subterm } 1)))$$
$$\nabla(\lambda (\lambda (\text{Subterm } 0)))$$

▼ Context:

$$\nabla(\lambda (\text{Subterm } 0))$$

▼ Context:

```
(refl Nat (s j)) : (== Nat (s j) (s j))
```

$$\nabla(\lambda X1 \text{ IH3 } (\lambda (\text{Subterm } 0)))$$

▽Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))

$$\nabla(\text{f-equal Nat Nat } (\lambda X1 \text{ (s X1)}) \text{ (s (plus X1 j)) (plus X1 (s j)) (Subterm 0))$$

```
(Focused) (== Nat (s (plus X1 j)) (plus X1 (s j)))
```

Generated by tactic

```
(by-apply f-equal #::with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))
```

Expand all

[Collapse all except focus](#)

Undo

Redo

GUI Proof Explorer

Top Level

- λ (n : Nat) (Subterm 0))
- Context: (n : Nat)
- λ (j : Nat) (Subterm 0))
- Context: (j : Nat)
- (new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))))
- λ (λ (Subterm 0))
- Context:
- λ (Subterm 0))
- Context:
- (refl Nat (s j)) : (== Nat (s j) (s j))
- λ X1 IH3 (λ (Subterm 0))
- Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))
- (f-equal Nat Nat (λ X1 (s X1)) (s (plus X1 j)) (plus X1 (s j)))
- (Focused) (== Nat (s (plus X1 j)) (plus X1 (s j)))

Context

J : Nat

X1 : Nat

IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j)))

Goal

(== Nat (s (plus X1 j)) (plus X1 (s j)))

Expand all

Collapse all except focus

Undo

Redo

GUI Proof Explorer

Context

Tactic

▼ [Focused] Top Level

▼ (λ (n : Nat) (Subterm 0))

▼ Context: (n : Nat)

▼ (λ (j : Nat) (Subterm 0))

▼ Context: (j : Nat)

▼ ((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j)))) (Subterm 0) (Subterm 0)) (λ (λ (Subterm 0))

▼ Context:

▼ (λ (Subterm 0))

▼ Context:

(refl Nat (s j)) : (== Nat (s j) (s j))

▼ (λ X1 IH3 (λ (Subterm 0))

▼ Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))

▼ (f-equal Nat Nat (λ X1 (s X1)) (s (plus X1 j)) (plus X1 (s j)) (Subterm 0))

IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j)))

Tactic

(by-intros n j)

(by-induction n)

reflexivity

(by-apply f-equal #:with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))

by-assumption

Complete Proof Term

Undo

(λ (n : Nat)

(λ (j : Nat)

((new-elim

n

(λ n (Π (== Nat (s (plus n j)) (plus n (s j))))

(λ (λ (λ (refl Nat (s j)))))

(λ X1

IH3

(λ (f-equal

Nat

Nat

...

Generated by tactic

(Passed into tactic)

Expand all

Collapse all except focus

Redo

GUI Proof Explorer

Context

Tactic

▼ (Focused) Top Level

▼ (λ (n : Nat) (Subterm 0))

▼ Context: (n : Nat)

▼ (λ (j : Nat) (Subterm 0))

▼ Context: (j : Nat)

▼ ((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))

▼ (λ (λ (Subterm 0)))

▼ Context:

▼ (λ (Subterm 0))

▼ Context:

(refl Nat (s j)) : (== Nat (s j) (s j))

▼ (λ X1 IH3 (λ (Subterm 0)))

▼ Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))

▼ (f-equal Nat Nat (λ X1 (s X1)) (s (plus X1 j)) (plus X1 (s j)) (Subterm 0))

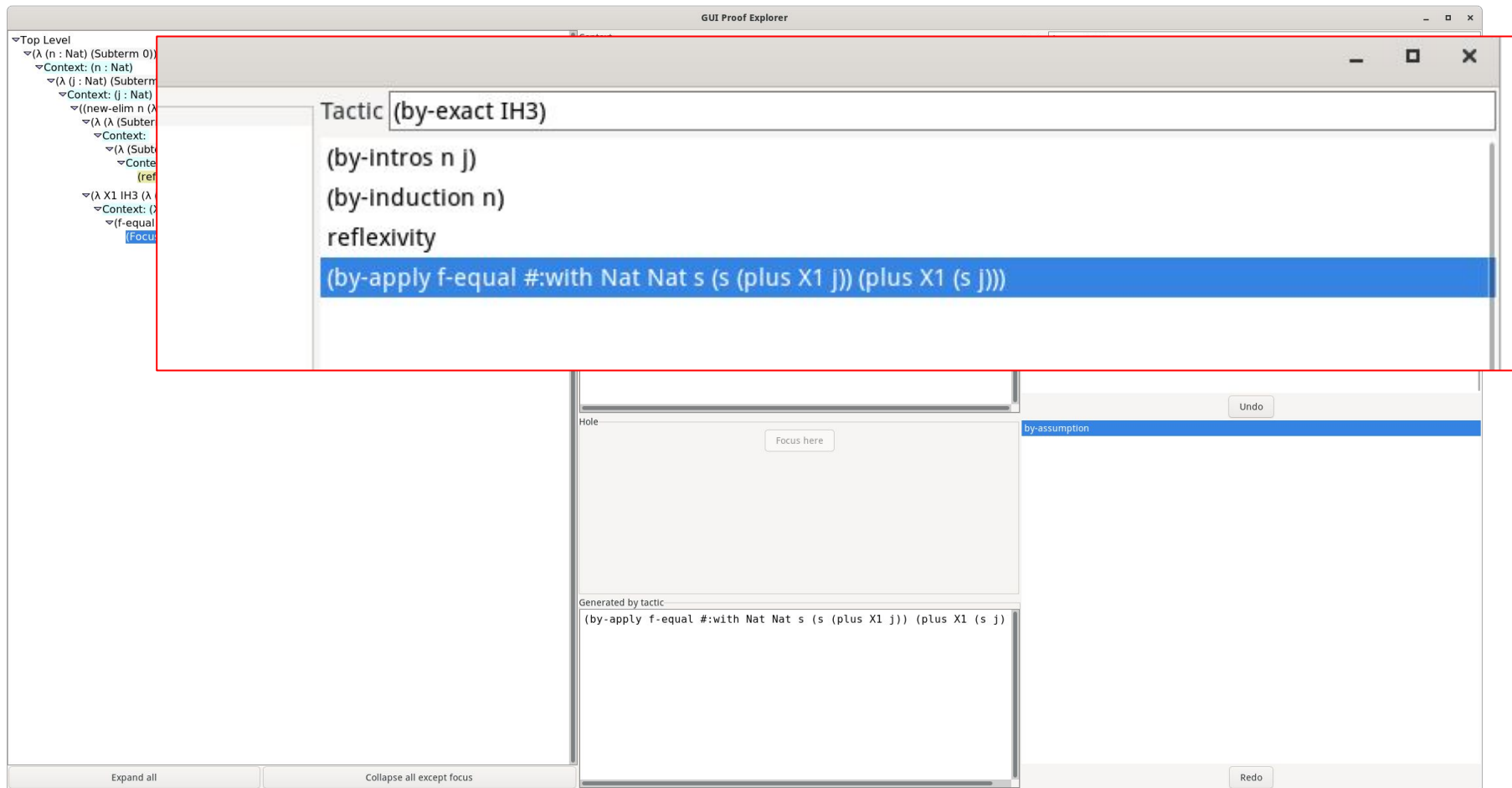
IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j)))

Undo

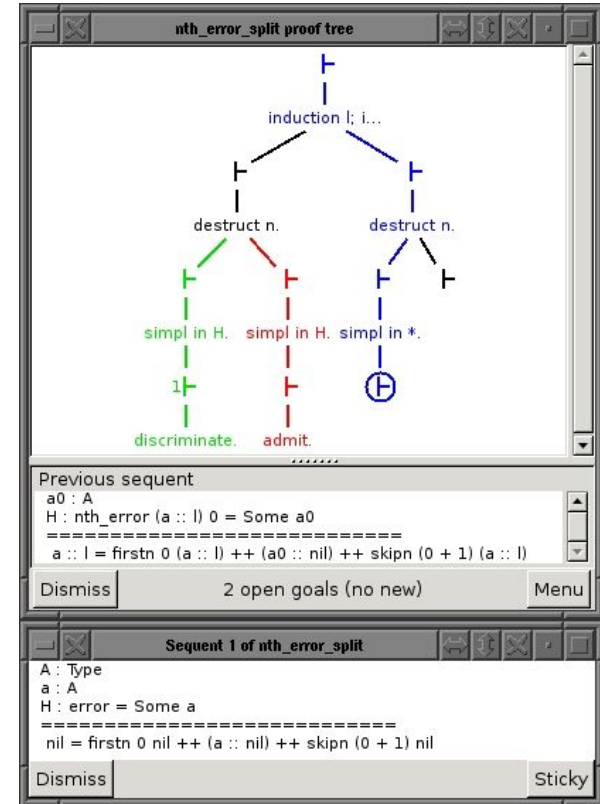
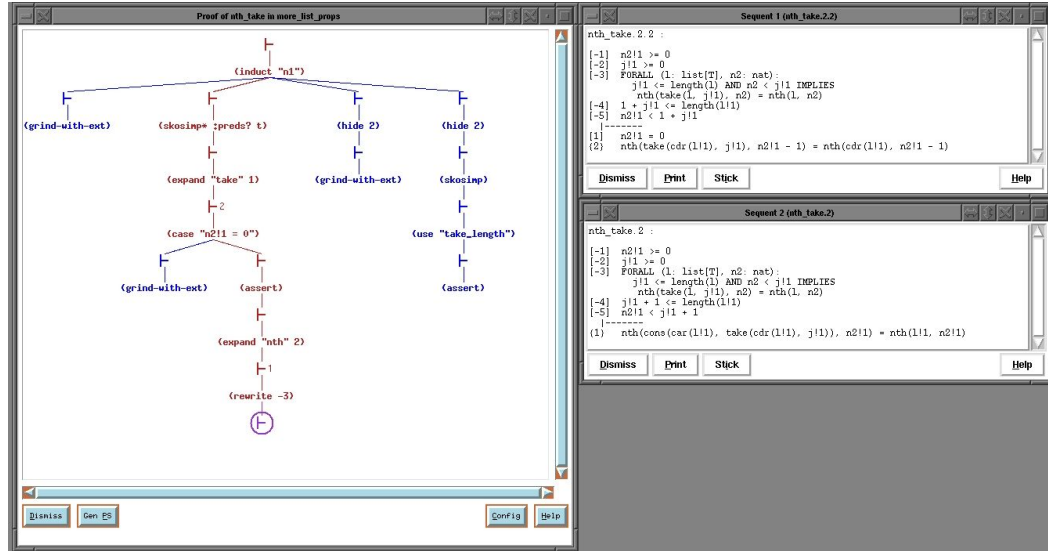
Expand all

Collapse all except focus

Redo

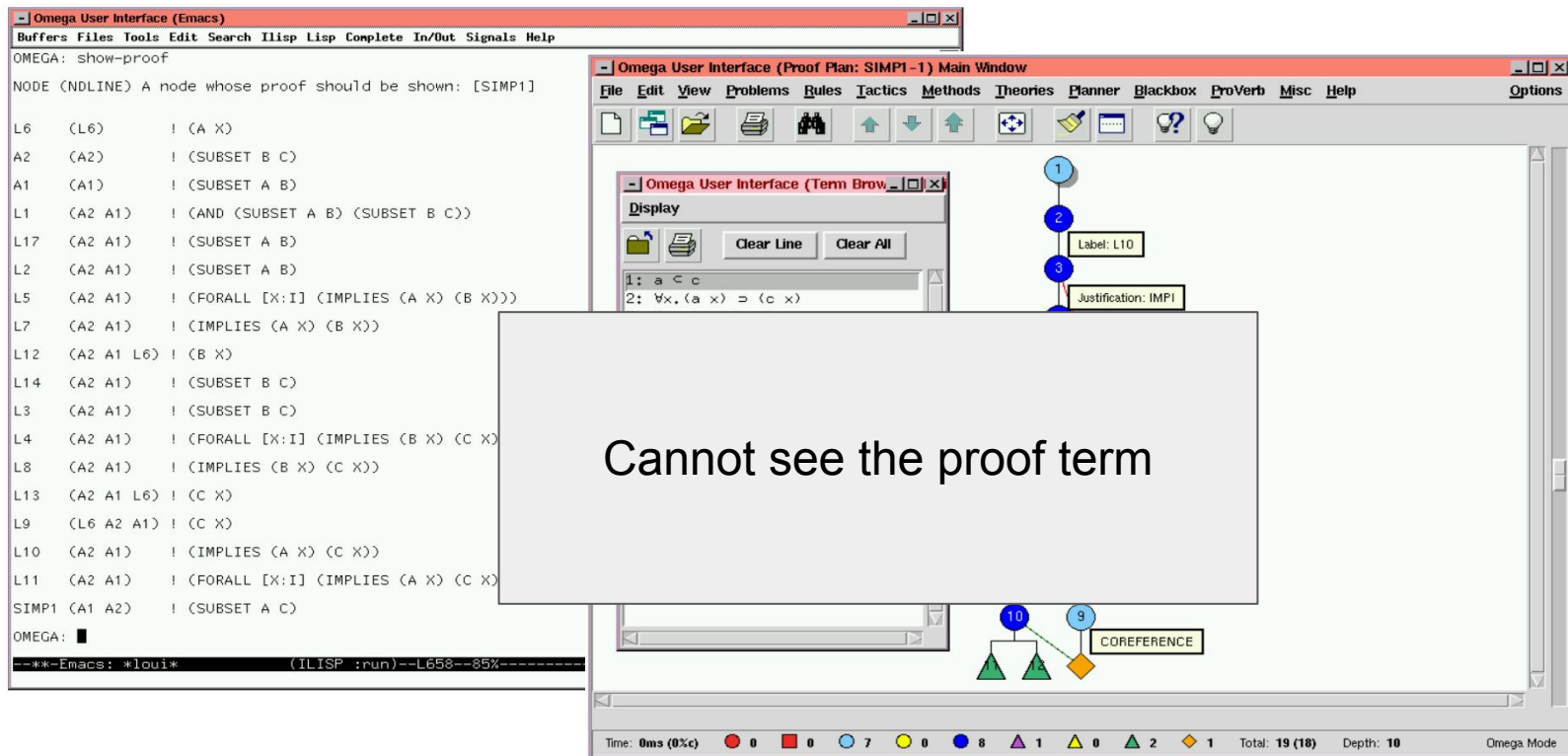


Related Work



(Left) Prototype Verification System, Owre S., Rushby J.M., Shankar N. (1992)

(Right) Proof Tree Visualization for Proof General, Tews H. (2011)



A Distributed Graphical User Interface for the Interactive Proof System
Siekmann et al. (1998)

ProofTool

File Edit View LK Proof LKS Proof Sunburst Help Tests

$$\begin{array}{c}
 = ((n_0 + k_0) + (1 + k_1)) \vdash \quad \quad \quad f(((n_0 + k_0) + 1) + k_1) \\
 \hline
 \text{---} \varepsilon.l1 \quad \quad \quad f((n_0 + k_0)) = 1, f((n_0 + k_0)) = 1 \\
 \hline
 \text{---} \neg.r \quad \quad \quad f((n_0 + k_0)) = 1, f(((n_0 + k_0) + 1) + k_1) \\
 \hline
 ((n_0 + k_0) + 1) + k_1) = 1 \vdash \neg((n_0 + k_0) = (((n_0 + k_0) + 1) + k_1) \wedge (f((n_0 + k_0)) = f(((n_0 + k_0) + 1) + k_1))) \\
 \hline
 f((n_0 + k_0)) = 1, f(((n_0 + k_0) + 1) + k_1) = 1 \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q))) \\
 \hline
 f((n_0 + k_0)) = 1, (\exists k)(f(((n_0 + k_0) + 1) + k)) = 1 \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q))) \\
 \hline
 f((n_0 + k_0)) = 1, (\forall n)(\exists k)(f((n + k)) = 1) \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q))) \\
 \hline
 (\forall n)(\exists k)(f((n + k)) = 1), (\exists k)(f((n_0 + k)) = 1) \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q))) \\
 \hline
 (\forall n)(\exists k)(f((n + k)) = 1), (\forall n)(\exists k)(f((n + k)) = 1) \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q))) \\
 \hline
 (\forall n)(\exists k)(f((n + k)) = 1) \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q))) \text{---} d.l \\
 \hline
 I(1) \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q))) \\
 \hline
 \text{---} cut
 \end{array}$$

Sunburst view of the-proof

Export as: PDF Ctrl-D PNG Ctrl-N

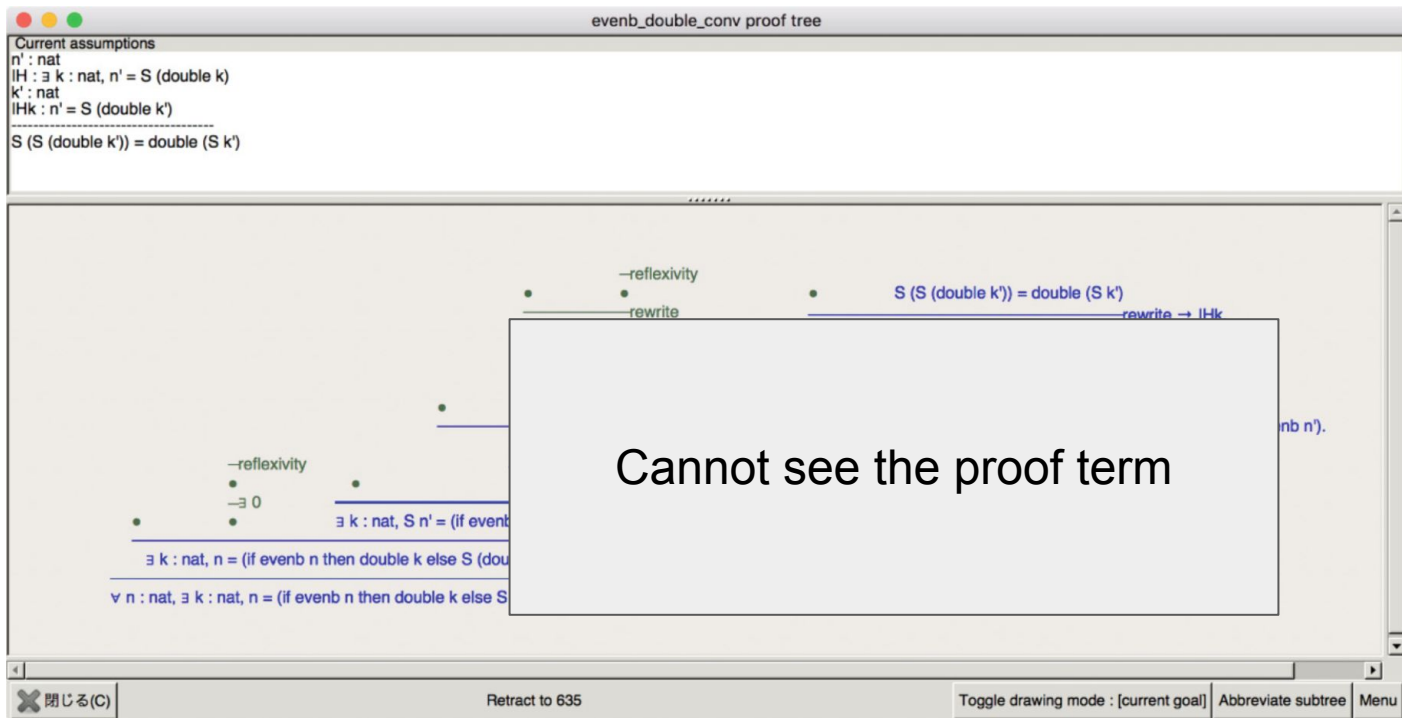
Inference: $d : l$

Auxiliary: $(\forall n)(\exists k)(f((n + k)) = 1)$

Primary: $I(1) \vdash$

Substitution:

Advanced Proof Viewing in ProofTool
 Libal T., Riener M., Rukhaia M. (2014)



Traf: A Graphical Proof Tree Viewer Cooperating with Coq Through Proof General
Kawabata H., Tanaka Y., Kimura M., Hironaka T. (2018)

Lemma Gauss: $\forall n, 2 * (\text{nsum } n \text{ (fun } i \Rightarrow i)}) = n * (n + 1).$ —

$$\forall n : \mathbb{N}. 2 \times \sum_{i=0}^n i = n \times (n + 1)$$

Proof. —
induction n; cbn [nsum]. —
- (* n + 0 *) —

$$2 \times 0 = 0 \times (0 + 1)$$

reflexivity.
- (* n + S _ *) —

$$n : \mathbb{N} \quad \text{IHn} : 2 \times \sum_{i=0}^n i = n \times (n + 1)$$

$$2 \times (S n + \sum_{i=0}^n i) = S n \times (S n + 1)$$

rewrite Mult.mult_plus_distr_l. —

$$2 \times S n + 2 \times \sum_{i=0}^n i = S n \times (S n + 1)$$

rewrite IHn. —

$$2 \times S n + n \times (n + 1) = S n \times (S n + 1)$$

ring.
Qed.

Untangling Mechanized Proofs, Pit-Claudel C. (2020)

Vernacular "Show Proof"

```
Theorem add1eq : forall
  (n : nat) (j : nat),
  (n + j).+1 =
    (n + j.+1).
intros n j.
elim n.
Show Proof.
```

```
(fun n j : nat =>
  nat_ind
    (fun n0 : nat =>
      eq (S (addn n0 j))
        (addn n0 (S j)))
    ?Goal ?Goal0 n)
```

Future Work

- Better IDE integration
- Tree diff view before/after tactics
- Save proof script to existing file
- Less verbose modes

ProofViz: Summary

- Bridge tactics and proof terms for beginners
- Develop new proofs or visualize existing scripts
- Minimal changes to rest of proof assistant

Thank you