

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

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

Theorem Proving (as an undergraduate student)

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

1 goal

$$\frac{n, j : \mathbb{N}}{\text{eq } (\text{s } (\text{addn } n j)) (\text{addn } n (\text{s } j))}$$

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}

eq (S (addn 0 j)) (addn 0 (S j))

subgoal 2 is:

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

1 goal

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

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

$$\frac{n, j, n_1 : \mathbb{N} \quad \text{IH} : \text{eq } (\text{s } (\text{addn } n_1 j)) (\text{addn } n_1 (\text{s } j))}{\text{eq } (\text{s } (\text{addn } (\text{s } n_1) j)) (\text{addn } (\text{s } n_1) (\text{s } j))}$$

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

$$\frac{n, j, n_1 : \mathbb{N} \quad \text{IH} : \text{eq } (\text{s } (\text{addn } n_1 j)) (\text{addn } n_1 (\text{s } j))}{\text{eq } (\text{s } (\text{addn } (\text{s } n_1) j)) (\text{addn } (\text{s } n_1) (\text{s } j))}$$

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

$n, j, n_1 : \mathbb{N}$

$\text{IH} : \text{eq } (\text{S } (\text{addn } n_1 j)) (\text{addn } n_1 (\text{S } j))$

$\text{why_doesnt_simpl_do_this} : \text{eq } (\text{addn } (\text{S } n_1) (\text{S } j))$
 $(\text{S } (\text{addn } n_1 (\text{S } j)))$

$\text{eq } (\text{S } (\text{addn } (\text{S } n_1) j)) (\text{addn } (\text{S } n_1) (\text{S } j))$

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 : \mathbb{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+=+add
        (+ 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?

```
nt_simpl_do_this :
  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+=+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.

Bottom-up proofs

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

Introducing ProofViz

GUI Proof Explorer

Context
n : Nat

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

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

Apply Subterms
0 : $(\text{== Nat} (s (\text{plus } n j)) (\text{plus } n (s j)))$

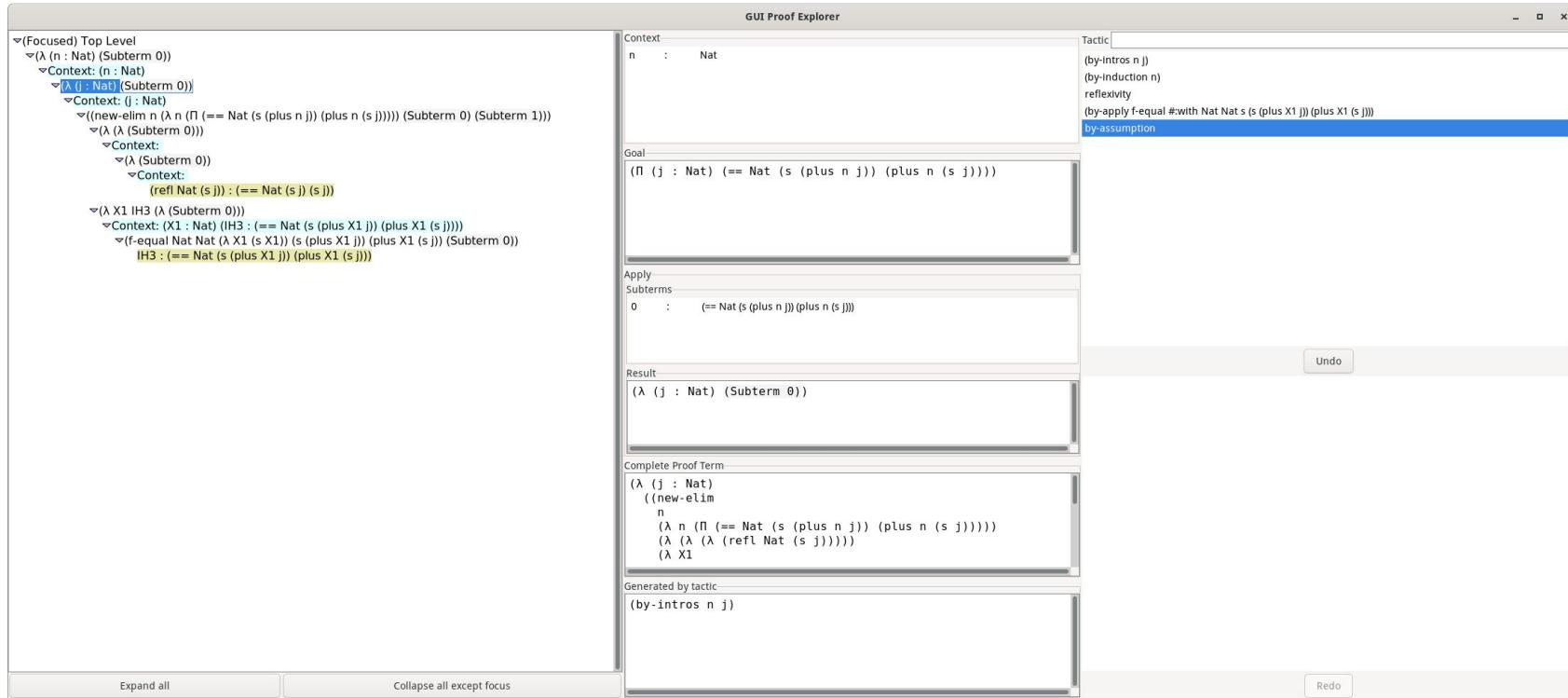
Result
 $(\lambda (j : \text{Nat}) (\text{Subterm } 0))$

Complete Proof Term
 $(\lambda (j : \text{Nat}) ((\text{new-elim} n (\lambda n (\Pi (\text{== Nat} (s (\text{plus } n j)) (\text{plus } n (s j)))))))$
 $\quad (\lambda (X1 (\lambda (X1 (\text{f-equal} \text{Nat Nat} (\lambda (X1 (s X1)) (s (\text{plus } X1 j)) (\text{plus } X1 (s j)))) (\text{Subterm } 0)))$
 $\quad \text{IH3} : (\text{== Nat} (s (\text{plus } X1 j)) (\text{plus } X1 (s j))))$

Generated by tactic
(by-intros n j)

Undo Redo

Expand all Collapse all except focus

The screenshot shows a window titled "GUI Proof Explorer". On the left, there is a tree view of the proof state, starting with "(Focused) Top Level" and expanding into various contexts and subterms. In the center, there are several panes: "Context" showing "n : Nat"; "Tactic" listing proof tactics like "by-intros", "by-induction", "reflexivity", and "by-apply f-equal"; "Goal" showing a complex type involving a universal quantifier over j and equality; "Apply Subterms" showing a single term; "Result" showing a lambda abstraction; "Complete Proof Term" showing the full proof term with tactic applications; and "Generated by tactic" showing the generated proof term. At the bottom, there are buttons for "Expand all", "Collapse all except focus", "Undo", and "Redo".

ProofViz: Tree View

The screenshot shows a window titled "ProofViz: Tree View" displaying a term structure tree. The tree is rooted at the top level, which contains a lambda abstraction $\lambda(n : \text{Nat})$. This abstraction has one subterm, n . The context for this abstraction is $(n : \text{Nat})$, which is expanded to show a lambda abstraction $\lambda(j : \text{Nat})$ with one subterm, j . The context for this second abstraction is $(j : \text{Nat})$, which is further expanded to show a new elimination rule $\text{new-elim } n (\lambda n (\Pi (== \text{Nat} (s (\text{plus} n j)) (\text{plus} n (s j)))))$. This rule has two subterms: Subterm 0 and Subterm 1 . The context for this rule is $(\lambda (\lambda (\text{Subterm 0})))$. The Subterm 0 is expanded to show a lambda abstraction $\lambda(\text{Subterm 0})$ with one subterm, Subterm 0 . The context for this third abstraction is $(\lambda (\text{Subterm 0}))$, which is expanded to show a context $(\text{refl Nat} (s j)) : (== \text{Nat} (s j) (s j))$. The Subterm 1 is expanded to show a lambda abstraction $\lambda X1 \text{IH3} (\lambda (\text{Subterm 0}))$ with one subterm, $X1$. The context for this fourth abstraction is $(X1 : \text{Nat}) (\text{IH3} : (== \text{Nat} (s (\text{plus} X1 j)) (\text{plus} X1 (s j))))$, which is expanded to show a f-equality check $(\text{f-equal Nat} \text{Nat} (\lambda X1 (s X1)) (s (\text{plus} X1 j)) (\text{plus} X1 (s j)) (\text{Subterm 0}))$. The context for this fifth abstraction is $(\text{IH3} : (== \text{Nat} (s (\text{plus} X1 j)) (\text{plus} X1 (s j))))$.

Top Level

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

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

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

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

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

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

Context:

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

Context:

$(\text{refl Nat} (s j)) : (== \text{Nat} (s j) (s j))$

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

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

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

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

Expand all

Collapse all except focus

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

Undo

Redo

ProofViz: Node Information

The screenshot shows the ProofViz interface with a red box highlighting the central proof state area. The interface is divided into several sections:

- (Focused) Top Level**: Shows the current term structure:
 - $\lambda(n : \text{Nat}) (\text{Subterm } 0)$
 - Context: $(n : \text{Nat})$
 - Subterm 0: $\lambda(j : \text{Nat}) (\text{Subterm } 0)$
 - Context: $(j : \text{Nat})$
 - Subterm 0: $\lambda(n : \text{Nat}) (\Pi (n == \text{Nat}(s(\text{plus } n j)) (\text{plus } n(s j))))$
 - Context: $(\text{new_elim } n (\lambda n (\Pi (n == \text{Nat}(s(\text{plus } n j)) (\text{plus } n(s j))))$
 - Subterm 0: $\lambda(\lambda(\text{Subterm } 0))$
 - Context: $(\lambda(\text{Subterm } 0))$
 - Subterm 0: $\lambda(\text{Subterm } 0)$
 - Context: $(\text{refl } \text{Nat}(s j) : (n == \text{Nat}(s j)) (s j))$
 - Subterm 0: $\lambda X1 \text{IH3 } (\lambda(\text{Subterm } 0))$
 - Context: $(X1 : \text{Nat}) (\text{IH3} : (n == \text{Nat}(s(\text{plus } X1 j)) (\text{plus } X1(s j)))$
 - Subterm 0: $\text{f-equal } \text{Nat } \text{Nat } (\lambda X1 (s X1)) (s(\text{plus } X1 j)) (\text{plus } X1(s j))$
 - Subterm 0: $\text{IH3} : (n == \text{Nat}(s(\text{plus } X1 j)) (\text{plus } X1(s j)))$
- Context**: Shows the context variable $n : \text{Nat}$.
- Goal**: Shows the goal term: $(\Pi (j : \text{Nat}) (n == \text{Nat}(s(\text{plus } n j)) (\text{plus } n(s j))))$.
- Apply Subterms**: Shows the subterm $0 : (n == \text{Nat}(s(\text{plus } n j)) (\text{plus } n(s j)))$.
- Generated by tactic**: Shows the tactic command: $(\text{by-intros } n j)$.

At the bottom of the interface are buttons for **Expand all**, **Collapse all except focus**, **Undo**, **Redo**, and **(by-intros n j)**.

ProofViz: Node Information

GUI Proof Explorer

(Focused) Top Level

Context: (n : Nat)

Subterm 0: (λ (j : Nat) (Subterm 0))

Context: (j : Nat)

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

Context: (λ (Subterm 0))

Subterm 0: (λ (Subterm 0))

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

Subterm 0: (λ X1 IH3 (λ (Subterm 0)))

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

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)))

Context: (Goal)

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

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)
```

Expand all

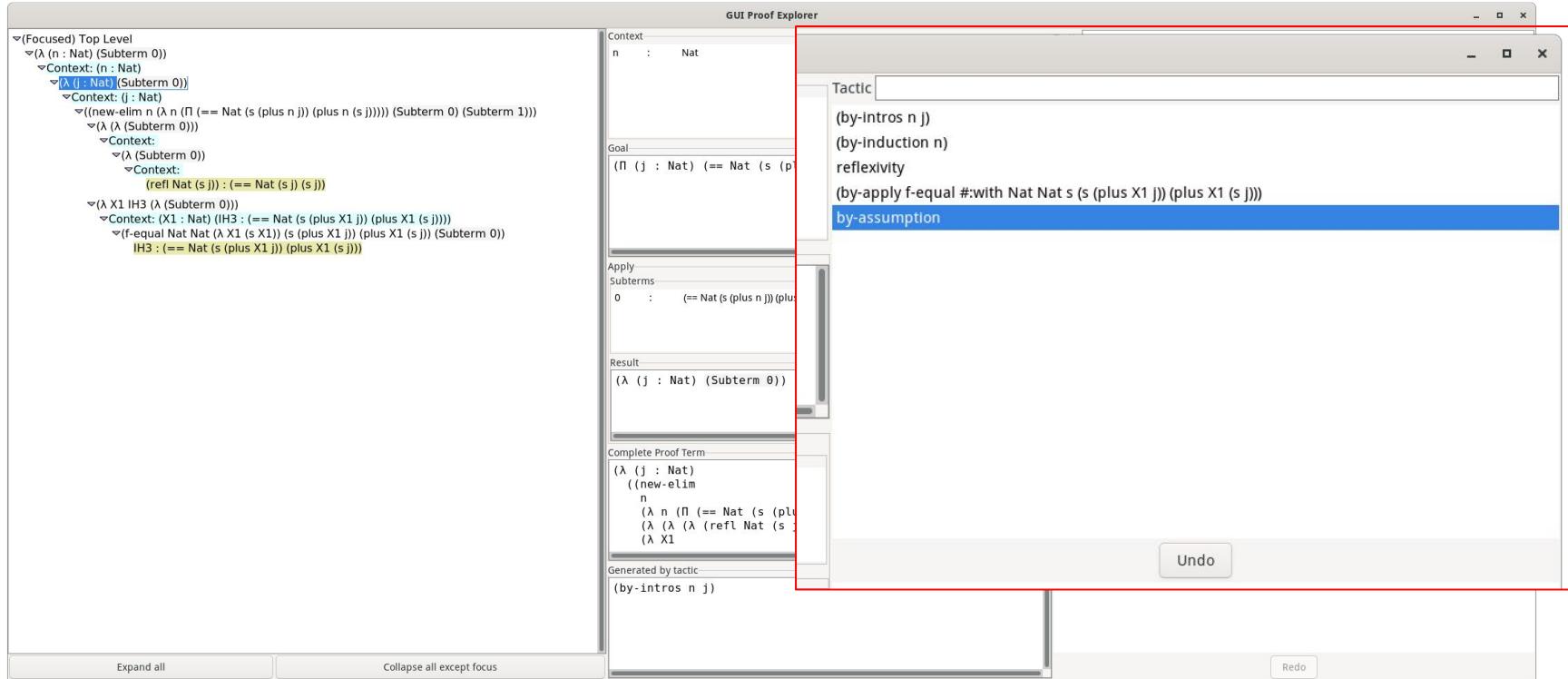
Collapse

Undo

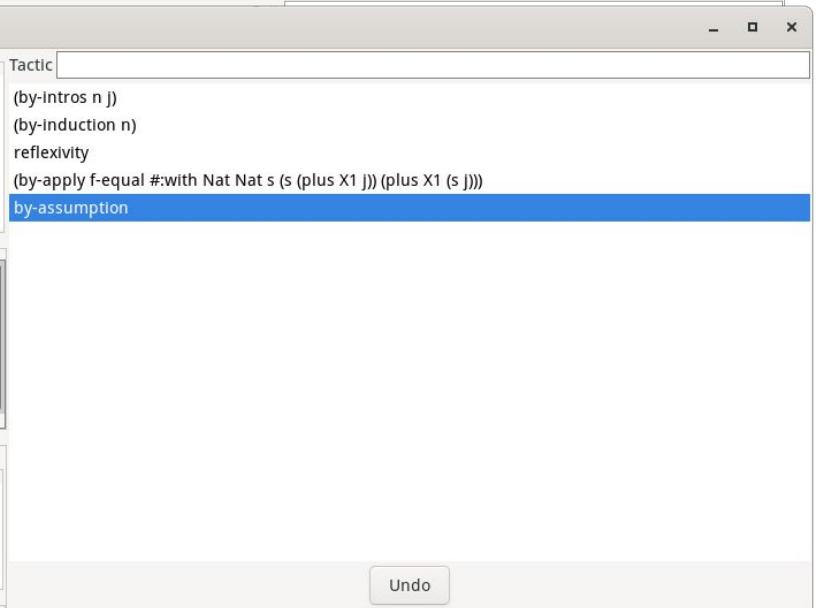
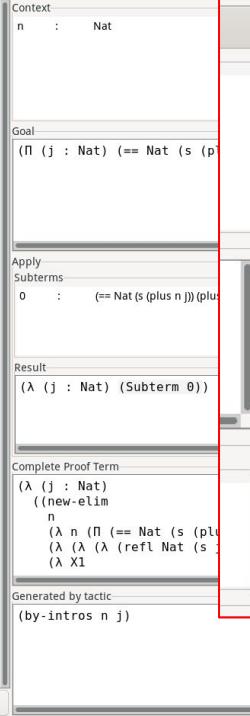
Redo

The screenshot shows a window titled "GUI Proof Explorer" with a "Result" panel containing the term $(\lambda (j : \text{Nat}) (\text{Subterm } 0))$. Below it is a "Complete Proof Term" panel showing the full proof structure, which includes a call to the (new-elim) tactic. The "Generated by tactic" panel shows the specific tactic used: (by-intros n j). The left sidebar displays a detailed proof tree with various nodes, contexts, and subterms. The right sidebar lists available tactics. The bottom of the window has standard "Undo" and "Redo" buttons.

ProofViz: Interaction



GUI Proof Explorer



Expand all

Collapse all except focus

Redo

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

Context

Tactic

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

Hole

Focus here

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

Generated by tactic
(Passed into tactic)

Expand all

Collapse all except focus

Redo

The screenshot shows the GUI Proof Explorer window with the following details:

- Top Level:** Focused on the goal $(\Pi (n : \text{Nat}) (j : \text{Nat}) (== \text{Nat} (s (\text{plus } n j)) (\text{plus } n (s j))))$.
- Context:** An empty panel.
- Tactic:** An empty panel.
- Goal:** The main goal expression is shown again: $(\Pi (n : \text{Nat}) (j : \text{Nat}) (== \text{Nat} (s (\text{plus } n j)) (\text{plus } n (s j))))$.
- Hole:** A box labeled "Focus here" is positioned over the hole in the goal expression.
- Tactic Suggestions:** A list of tactics is displayed:
 - (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
- Generated by tactic:** A panel containing the text "(Passed into tactic)".
- Bottom Buttons:** "Expand all", "Collapse all except focus", "Undo", "Redo".

GUI Proof Explorer

Context:

- n : Nat
- j : Nat

Tactic

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

Goal

```
(== Nat (s (plus
```

Hole

Focus here

Generated by tactic

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

Undo

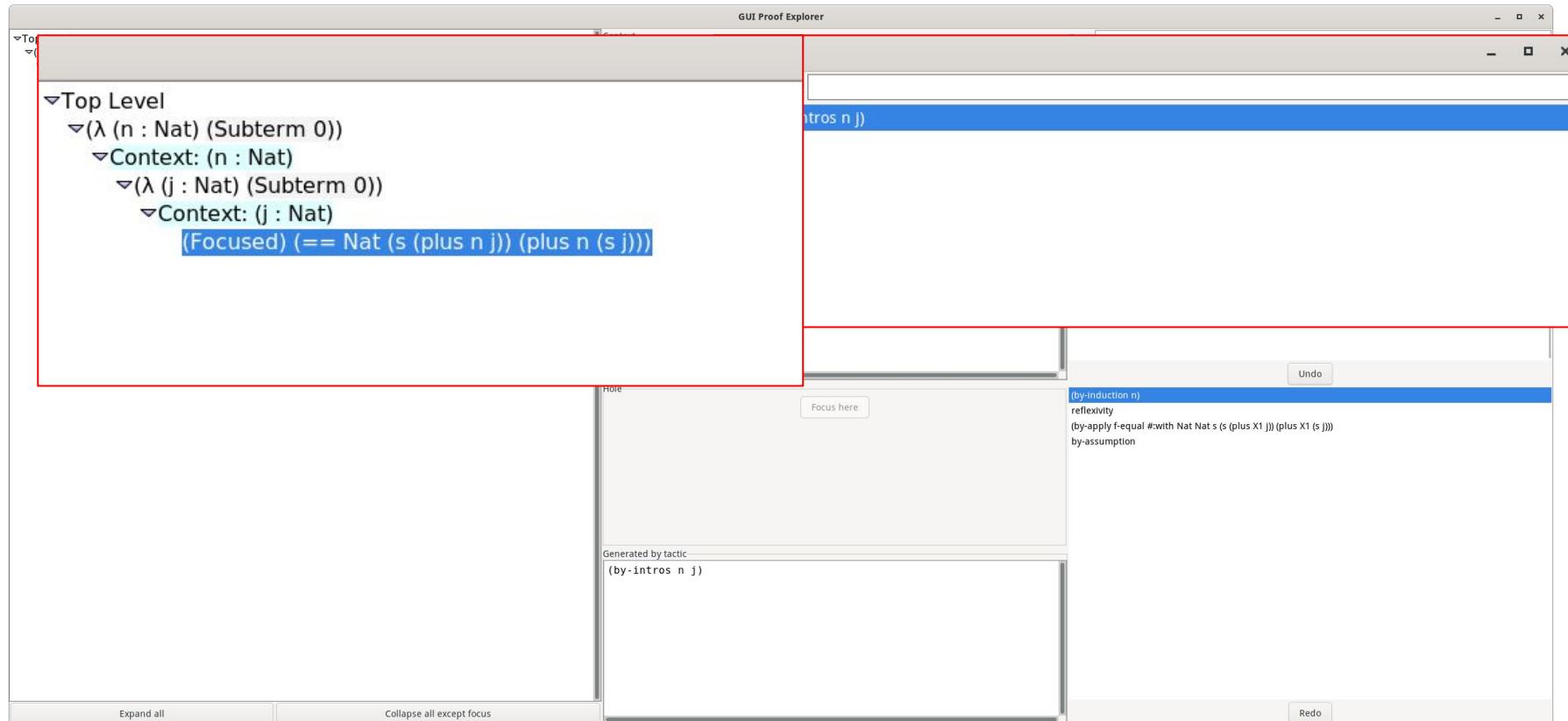
(by-induction n)
reflexivity
(by-apply f-equal #:with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))
by-assumption

Redo

Expand all

Collapse all except focus

The screenshot shows a window titled "GUI Proof Explorer" with a red box highlighting the "Tactic" input field. The "Tactic" field contains the command "(by-intros n j)". To the left, the "Context" pane shows declarations for variables n and j of type Nat. Below the tactic, the "Goal" pane shows a goal involving Nat and s. A "Hole" is present, indicated by a "Focus here" button. The bottom section shows generated tactic code: "(by-intros n j)". On the right, there is a stack of tactic steps: "(by-induction n)", "reflexivity", "(by-apply f-equal #:with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))", and "by-assumption". Navigation buttons "Undo" and "Redo" are at the bottom right, along with "Expand all" and "Collapse all except focus" buttons at the bottom left.



GUI Proof Explorer

To

Top Level

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

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

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

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

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

Tactic: (by-intros n j)

0 : $(\equiv \text{Nat} (s (\text{plus} n j)) (\text{plus} n (s j)))$

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

Generated by tactic: (by-intros n j)

Undo

Redo

Expand all

Collapse all except focus

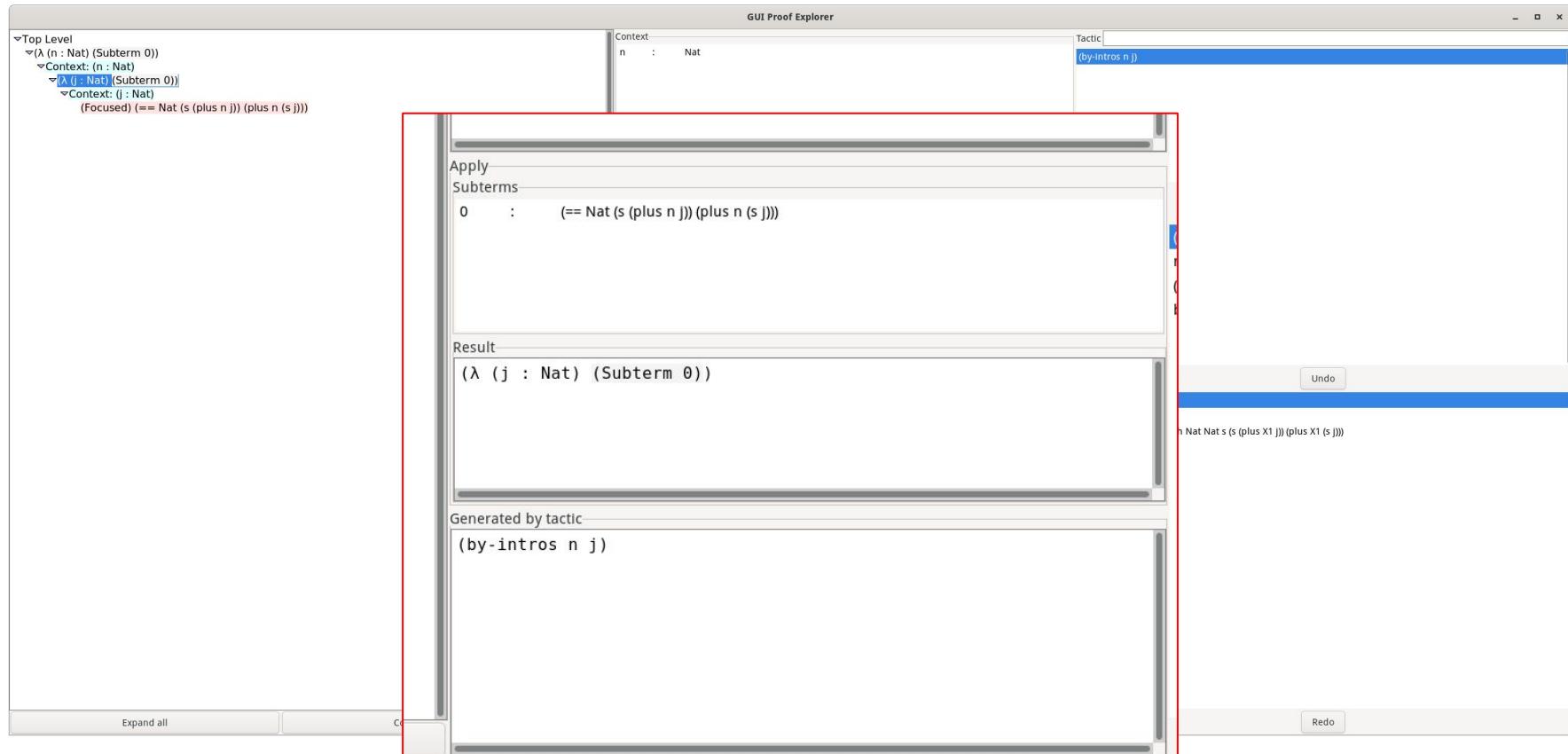
The screenshot shows a window titled "GUI Proof Explorer" with a tree view on the left and a tactic history and result view on the right.

Left Panel (Tree View):

- Top Level
 - $\triangleright (\lambda (n : \text{Nat}) (\text{Subterm} 0))$
 - $\triangleright \text{Context: } (n : \text{Nat})$
 - $\triangleright (\lambda (j : \text{Nat}) (\text{Subterm} 0))$
 - $\triangleright \text{Context: } (j : \text{Nat})$
 - (Focused) $(\equiv \text{Nat} (s (\text{plus} n j)) (\text{plus} n (s j)))$

Right Panel (Tactic History and Result):

- Tactic:** (by-intros n j)
- 0 :** $(\equiv \text{Nat} (s (\text{plus} n j)) (\text{plus} n (s j)))$
- Result:** $(\lambda (j : \text{Nat}) (\text{Subterm} 0))$
- Generated by tactic:** (by-intros n j)
- Tactic History (from bottom to top):**
 - (by-induction n)
 - reflexivity
 - (by-apply f-equal #with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))
 - by-assumption



GUI Proof Explorer

Top Level

Context: Nat

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

Top Level

Subterm 0

Context: (n : Nat)

Subterm 1

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

Context: (j : Nat)

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

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

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

Generated by tactic
(by-induction n)

Expand all | Collapse all except focus | Redo

GUI Proof Explorer

Top Level

(λ)

Top Level

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

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

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

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

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

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

Context:

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

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

Context: ($X1 : \text{Nat}$) (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

intros n j
induction n

Undo

Redo

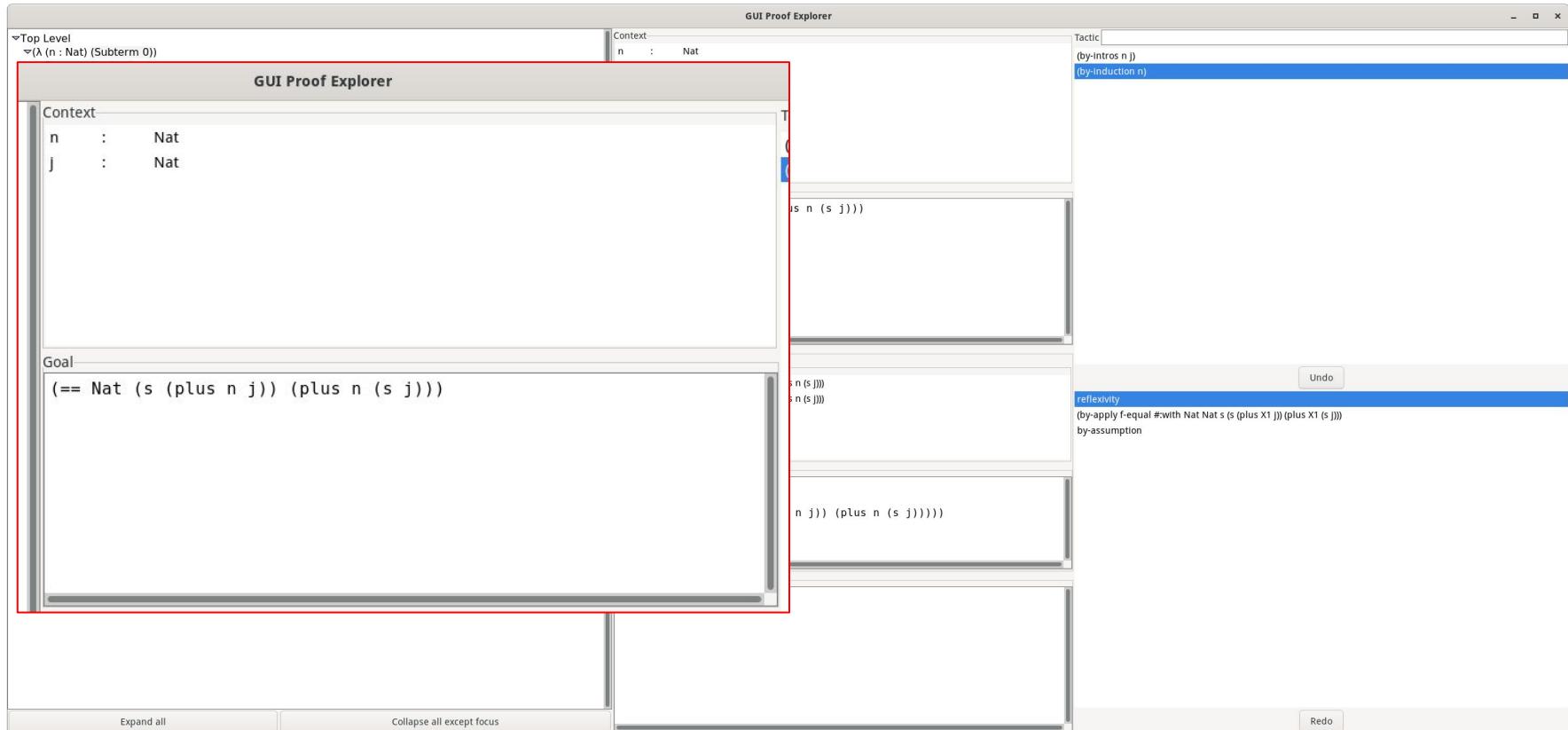
Generated by tactic
(by-induction n)

ativity
apply f-equal #with Nat Nat s (s (plus X1 j)) (plus X1 (s j))
umption

Expand all

Collapse all except focus

The screenshot shows a proof state in the 'Top Level' of a theorem. The proof consists of two main quantified statements, each with a context and a body. The first body contains a new elimination tactic application, which itself has a context and a body. The second body contains an induction tactic application, also with a context and a body. A red box highlights the entire proof structure. Below the proof, a tactic history window is open, showing the 'new-elim' and 'induction' tactics used. At the bottom, there are buttons for 'Expand all', 'Collapse all except focus', 'Undo', and 'Redo'.



GUI Proof Explorer

Top Level
λ (n : Nat) (Subterm 0)

Context

```
n      :  Nat
j      :  Nat
```

Goal

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

GUI Proof Explorer

Context

```
n      :  Nat
```

Apply

Subterms

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

Result

```
((new-elem
  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

(λ)

Top Level

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

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

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

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

((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:

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

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

Context: ($X1 : \text{Nat}$) (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

(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

Undo

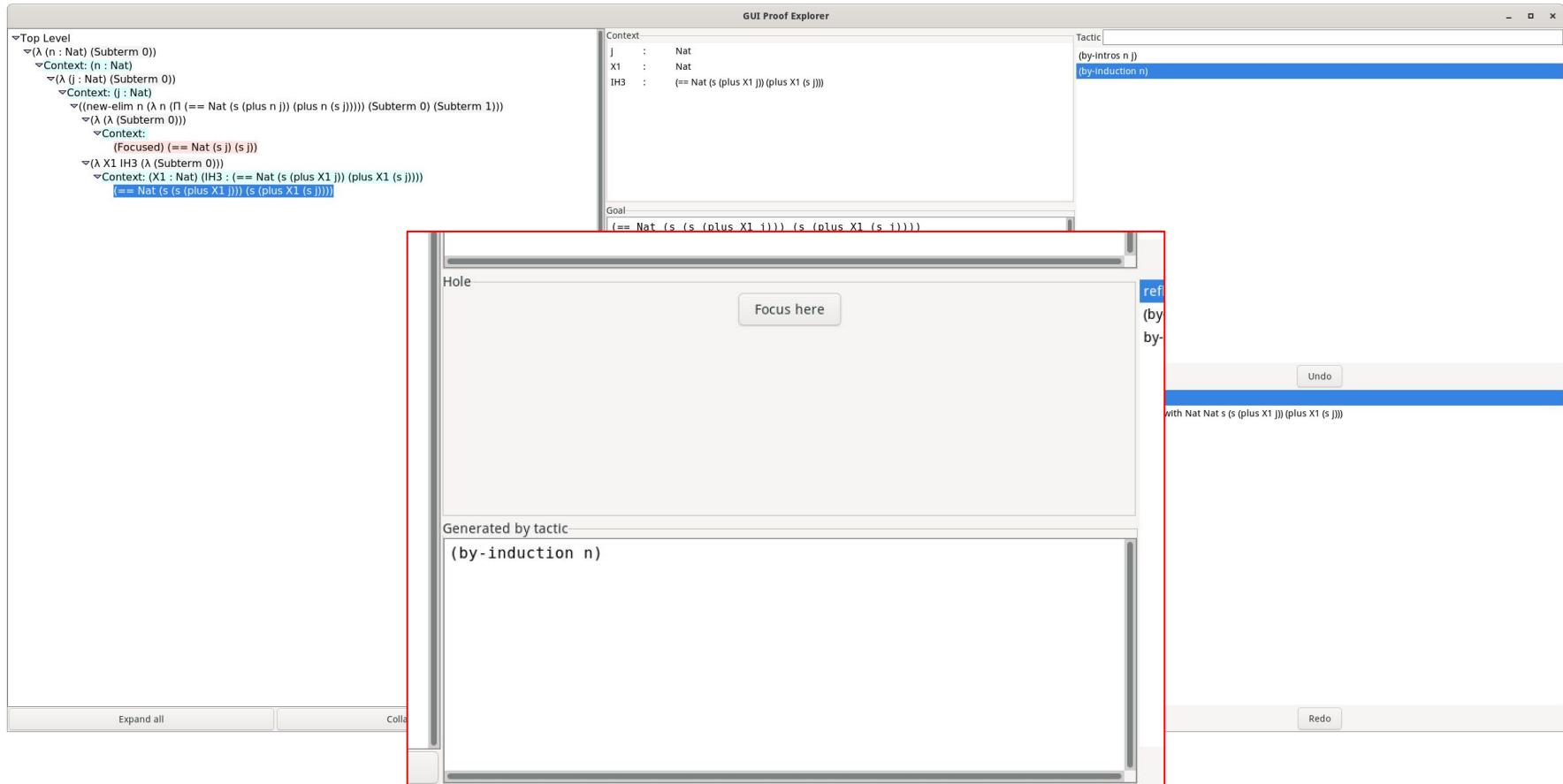
Generated by tactic

(by-induction n)

Redo

Expand all

Collapse all except focus



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

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))
```

Goal

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

Hole

Focus here

Generated by tactic

```
(by-induction n)
```

Buttons

Expand all | Collapse all except focus | Undo | Redo

GUI Proof Explorer

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

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))

Top Level

- $\lambda(n : \text{Nat}) (\text{Subterm } 0)$
 - Context: (n : Nat)
 - $\lambda(j : \text{Nat}) (\text{Subterm } 0)$
 - Context: (j : 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:
 - $(== \text{Nat} (s j) (s j))$

Focus here

Undo

Generated by tactic
(by-induction n)

Redo

Expand all

Collapse all except focus

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))))
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)))

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))

Focus here

Generated by tactic
(by-induction n)

Expand all | Collapse all except focus | Undo | Redo

GUI Proof Explorer

Top Level

Context:

- ($\lambda (n : \text{Nat}) (\text{Subterm} 0)$)
- Context: ($n : \text{Nat}$)
- Context: ($j : \text{Nat}$)
- Context: ($i : \text{Nat}$)
- ((new-elim n ($\lambda n (\prod (== \text{Nat} (s (\text{plus} n j)) (\text{plus} n (s j)))) (\text{Subterm} 0) (\text{Subterm} 1)$)))
- Context: ($\lambda (s : \text{Subterm} 0)$)
- Context: ($\lambda (s : \text{Subterm} 0)$)
- Context: ($\lambda (s : \text{Subterm} 0)$)
- (refl Nat (s j)) : ($== \text{Nat} (s j) (s j)$)
- ($\neg (\lambda X1 IH3 (\lambda (\text{Subterm} 0)))$)
- Context: ($X1 : \text{Nat}$) (IH3 : ($== \text{Nat} (s (\text{plus} X1 j)) (\text{plus} X1 (s j))$))
- (Focused) ($== \text{Nat} (s (s (\text{plus} X1 j))) (\text{plus} X1 (s j))$)

Context

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

Tactic

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

Goal

Hole

Focus here

Generated by tactic

(by-induction n)

Undo

Redo

Expand all

Collapse all except focus

The screenshot shows a proof assistant interface with a red box highlighting the Tactic panel. The Tactic panel lists three tactics: (by-intros n j), (by-induction n), and reflexivity. The word 'reflexivity' is highlighted with a blue background. The rest of the interface shows the proof context, goal, hole, and generated tactic code.

GUI Proof Explorer

Top Level

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

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

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

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

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

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

Context:

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

Context:

(refl Nat (s j)) : ($== \text{Nat} (s j) (s j)$)

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

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

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

Generated by tactic
(by-induction n)

Expand all | Collapse all except focus | Undo | Redo

GUI Proofs

Top Level

```
(λ (n : Nat) (Subterm 0))  
  Context: (n : Nat)  
    λ (j : Nat) (Subterm 0)  
      Context: (j : Nat)  
        ((new-elem n (λ n (Π (== Nat (s (plus n j)) (plus n (s j)))) (Subterm 0) (Subterm 1)))  
         λ (A (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 (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)))
```

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)))
```

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)))
```

Undo

by-assumption

Redo

Expand all

Collapse all except focus

GUI Proo

Tactic
(by-intros n j)

To

Top Level

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

Context: $n : \text{Nat}$

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

Context: $j : \text{Nat}$

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

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

Context:

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

Context:

$(\text{refl } \text{Nat} (s j)) : (== \text{Nat} (s j) (s j))$

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

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

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

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

S (plus X1 j)) (plus X1 (s j)))

Assumption

Undo

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

Redo

The screenshot shows a proof assistant interface with a top-level tactic state and a history of generated steps.

Tactic State:

- GUI Proo
- Tactic: (by-intros n j)

Top Level:

- $\lambda(n : \text{Nat})(\text{Subterm } 0)$
- Context: $n : \text{Nat}$
- $\lambda(j : \text{Nat})(\text{Subterm } 0)$
- Context: $j : \text{Nat}$
- $((\text{new-elem } n (\lambda n (\Pi (== \text{Nat} (s (\text{plus } n j)) (\text{plus } n (s j)))))) (\text{Subterm } 0) (\text{Subterm } 1))$
- $\lambda(\lambda(\text{Subterm } 0))$
- Context:
- $\lambda(\text{Subterm } 0)$
- Context:
- $(\text{refl } \text{Nat} (s j)) : (== \text{Nat} (s j) (s j))$
- $\lambda(X1 \text{ IH3 } (\lambda(\text{Subterm } 0)))$
- Context: $X1 : \text{Nat}$ ($\text{IH3} : (== \text{Nat} (s (\text{plus } X1 j)) (\text{plus } X1 (s j)))$)
- $\lambda(\text{f-equal } \text{Nat} \text{ Nat } (\lambda X1 (s X1)) (s (\text{plus } X1 j)) (\text{plus } X1 (s j)) (\text{Subterm } 0))$
- (Focused) $(== \text{Nat} (s (\text{plus } X1 j)) (\text{plus } X1 (s j)))$

History:

- S (plus X1 j)) (plus X1 (s j)))
- Assumption

Buttons:

- Undo
- Redo

Bottom Panel:

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

Bottom Buttons:

- Expand all
- Collapse all except focus

GUI Proof Explorer

Top Level

```
¬(λ (n : Nat) (Subterm 0))
  Context: (n : Nat)
    ¬(λ (j : Nat) (Subterm 0))
      Context: (j : Nat)
        ¬((new-elem n (λ (n (Π (== Nat (s (plus n j)) (plus n (s j))))))) (λ (A (Subterm 0)))
          Context: (A : Subterm 0)
            ¬(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)))
```

Undo

Redo

Expand all

Collapse all except focus

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
          ¬(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

```
(λ (n : Nat)
  (λ (j : Nat)
    ((new-elim
      n
      (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))))
      (λ (λ (refl Nat (s j))))))
  (λ X1
    IH3
    (λ (f-equal
      Nat
      Nat
      ....
      ....)))
```

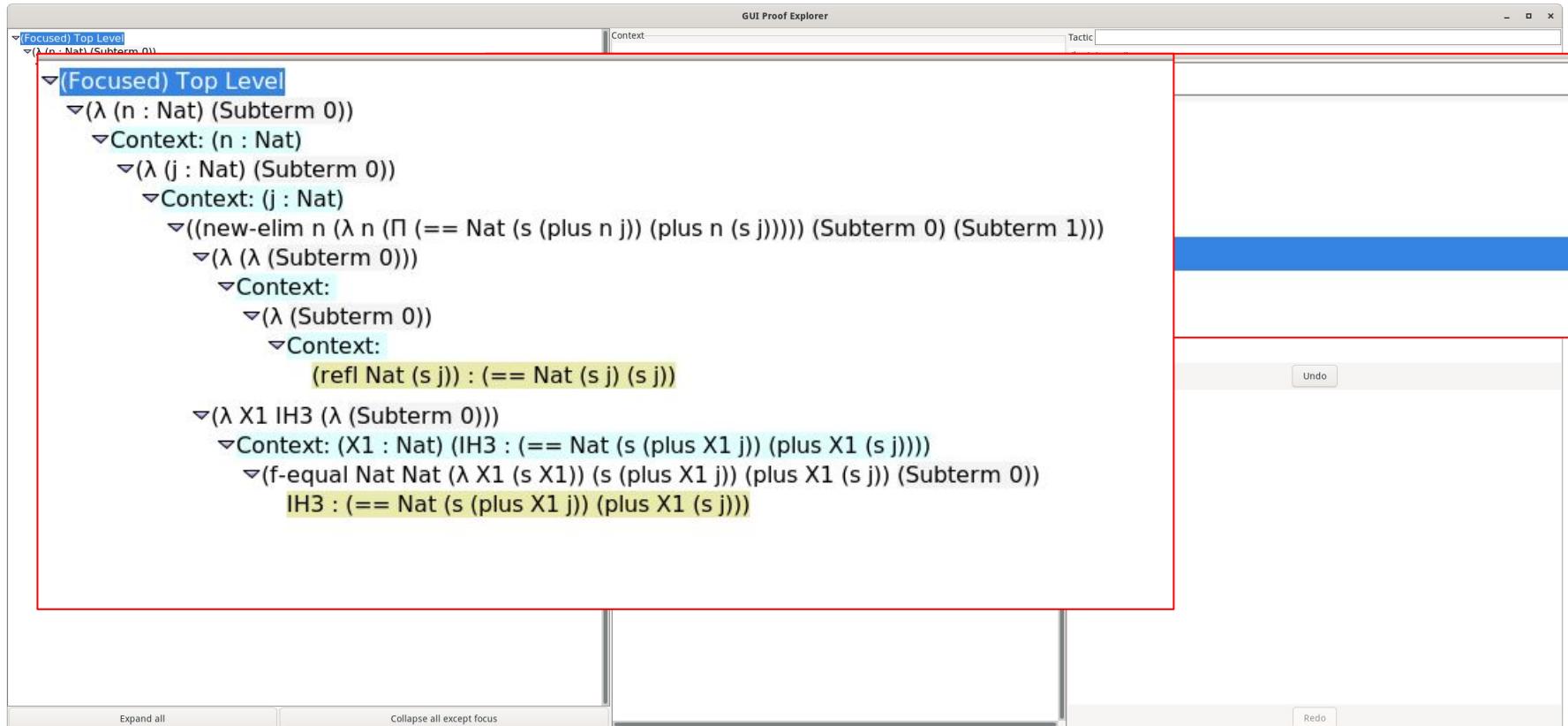
Generated by tactic
(Passed into tactic)

Expand all

Collapse all except focus

Undo

Redo



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))
          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

Tactic

Goal

```
(Π (n : Nat) (j : Nat) (== Nat (s (plus n j)) (plus n (s j))))
```

Complete Proof Term

```
(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)

Top Level
 $\lambda(n : \text{Nat}) (\text{Subterm } 0)$
 Context: $(n : \text{Nat})$
 $\lambda(j : \text{Nat}) (\text{Subterm } 1)$
 Context: $(j : \text{Nat})$
 $((\text{new-elim } n) (\lambda(s : \text{Subterm } 2) (\text{ref})))$
 Context:
 $\lambda(s : \text{Subterm } 2) (\text{Conte} (\text{ref}))$
 $\lambda(X1 : \text{IH3}) (\lambda(j : \text{Nat}) (\text{Subterm } 3))$
 Context: $(j : \text{Nat})$
 $\text{if-equal } (\text{Focus })$

Tactic (by-exact IH3)

(by-intros n j)

(by-induction n)

reflexivity

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

Hole

Focus here

Undo

by-assumption

Generated by tactic

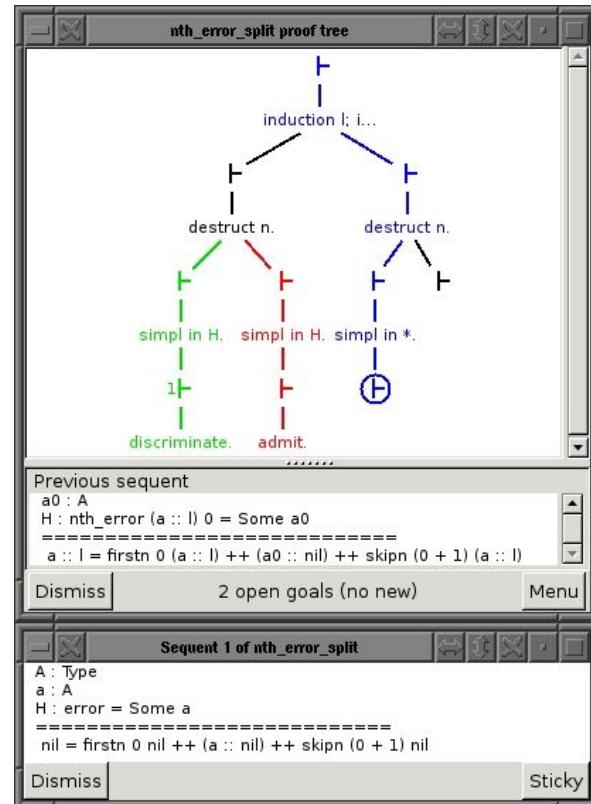
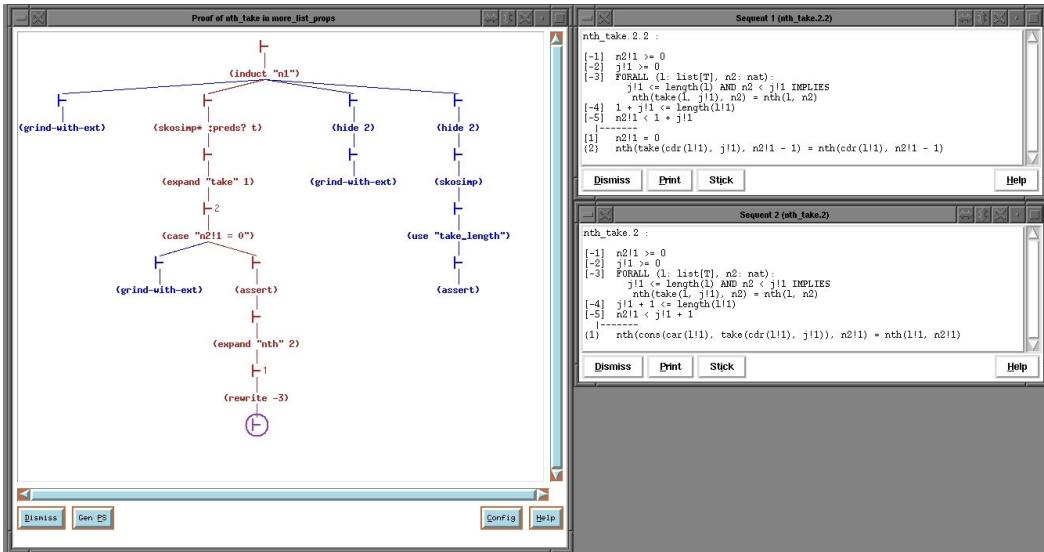
(by-apply f-equal #:with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))

Expand all

Collapse all except focus

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)

-| Omega User Interface (Emacs)

Buffers Files Tools Edit Search Ilisp Lisp Complete In/Out Signals Help

OMEGA: show-proof

NODE (NDLINE) A node whose proof should be shown: [SIMP1]

```

L6  (L6)      ! (A X)
A2  (A2)      ! (SUBSET B C)
A1  (A1)      ! (SUBSET A B)
L1  (A2 A1)   ! (AND (SUBSET A B) (SUBSET B C))
L17 (A2 A1)   ! (SUBSET A B)
L2  (A2 A1)   ! (SUBSET A B)
L5  (A2 A1)   ! (FORALL [X:I] (IMPLIES (A X) (B X)))
L7  (A2 A1)   ! (IMPLIES (A X) (B X))
L12 (A2 A1 L6) ! (B X)
L14 (A2 A1)   ! (SUBSET B C)
L3  (A2 A1)   ! (SUBSET B C)
L4  (A2 A1)   ! (FORALL [X:I] (IMPLIES (B X) (C X)))
L8  (A2 A1)   ! (IMPLIES (B X) (C X))
L13 (A2 A1 L6) ! (C X)
L9  (L6 A2 A1) ! (C X)
L10 (A2 A1)   ! (IMPLIES (A X) (C X))
L11 (A2 A1)   ! (FORALL [X:I] (IMPLIES (A X) (C X)))
SIMP1 (A1 A2) ! (SUBSET A C)

```

OMEGA: ■

--**-Emacs: *louie* (ILISP :run)--L658--85%

-| Omega User Interface (Proof Plan: SIMP1-1) Main Window

File Edit View Problems Rules Tactics Methods Theories Planner Blackbox ProVerb Misc Help Options

|- Omega User Interface (Term Brow...)

Display

Clear Line Clear All

1: a ⊂ c
2: ∀x.(a x) ⊃ (c x)

1
2
3
Label: L10
Justification: IMPI

Time: 0ms (0%) 0 0 7 0 8 1 0 2 1 Total: 19 (18) Depth: 10 Omega Mode

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

$$=((n_0 + k_0) + (1 + k_1)) \vdash$$

$$\frac{e;l}{\neg r}$$

$$f((n_0 + k_0)) = 1 \vdash f((n_0 + k_0)) = 1$$

$$f((n_0 + k_0)) = 1 , f(((n_0 + k_0) + 1 + k_1)) \vdash$$

$$((n_0 + k_0) + 1 + k_1)) = 1 \vdash \neg((n_0 + k_0) = ((n_0 + k_0) + 1 + k_1)) \wedge (f((n_0 + k_0)) = f((n_0 + k_0)) = 1 , f(((n_0 + k_0) + 1 + k_1)) = 1 \vdash (\exists q)(\neg((n_0 + k_0) = q) \wedge (f((n_0 + k_0)) = f((n_0 + k_0)) = 1 , f(((n_0 + k_0) + 1 + k_1)) = 1 \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q)))$$

$$f((n_0 + k_0)) = 1 , (\exists k)(f(((n_0 + k_0) + 1 + k_1)) = 1) \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q)))$$

$$f((n_0 + k_0)) = 1 , (\forall n)(\exists k)(f((n + k)) = 1) \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q)))$$

$$(\forall n)(\exists k)(f((n + k)) = 1) , (\exists k)(f((n_0 + k)) = 1) \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q)))$$

$$(\forall n)(\exists k)(f((n + k)) = 1) , (\forall n)(\exists k)(f((n + k)) = 1) \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q)))$$

$$(\forall n)(\exists k)(f((n + k)) = 1) \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q)))$$

$$I(1) \vdash (\exists p)(\exists q)(\neg(p = q) \wedge (f(p) = f(q)))$$

$$\frac{d;l}{\text{cut}}$$

Sunburst view of the-proof

Export as: PDF Ctrl-D PNG Ctrl-N

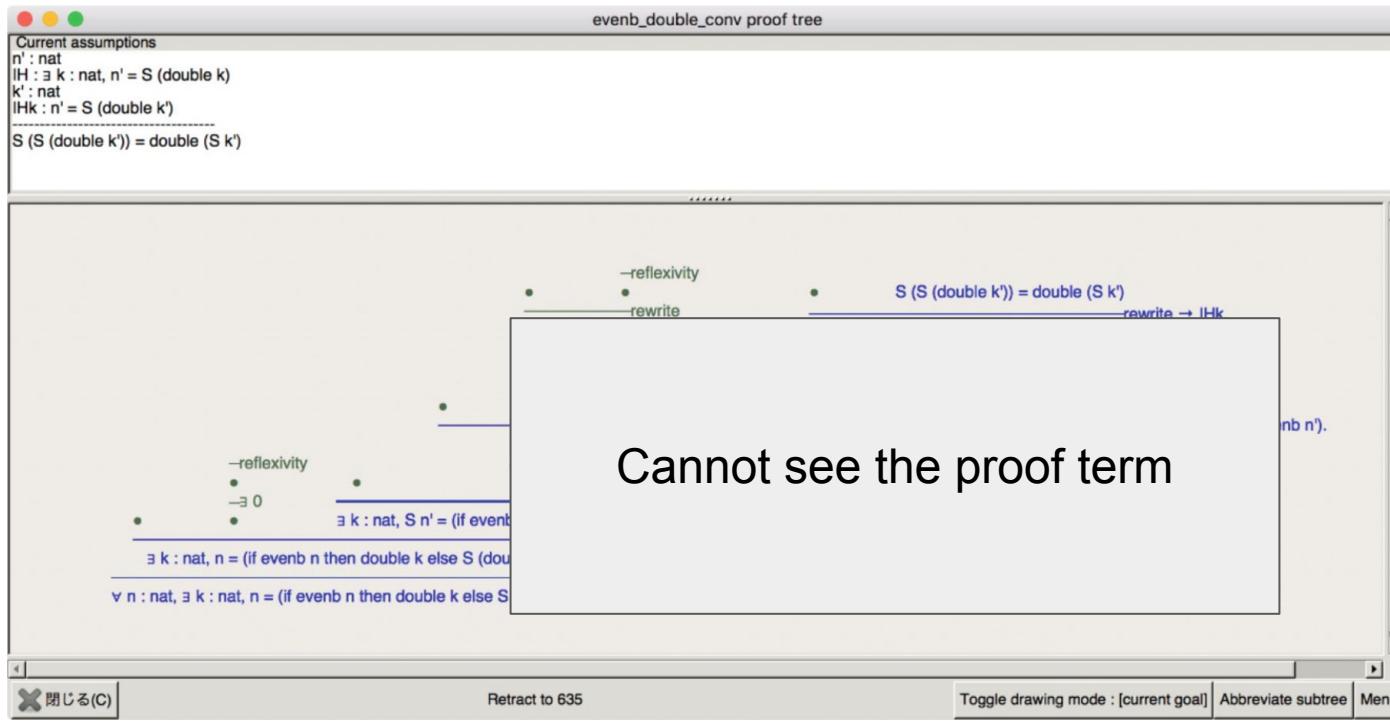
Inference: $d : l$

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

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 ← θ *) =

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

reflexivity.
- (* n ← S _ *) =

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

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

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          (fun n j : nat =>
  (n : nat) (j : nat),           nat_ind
  (n + j).+1 =                  (fun n0 : nat =>
  (n + j.+1).                   eq (S (addn n0 j))
intros n j.                     (addn n0 (S j)))
elim n.                          ?Goal ?Goal0 n)
Show Proof.
```

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