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

```
2 goals

  n, j : ℕ
  ──────────────────────────────
  eq (S (addn O j)) (addn O (S j))

subgoal 2 is:
 ∀ (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 add1eq : 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 add1eq : 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 add1eq : 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 add1eq : 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 add1eq : 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?

[1]D. Friedman, D. Christiansen (2018)

Example: *The Little Typer*[1]

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

[1]D. Friedman, D. Christiansen (2018)

Example: *The Little Typer*

```
(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)
    (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+=+add1 n-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.
```

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

How are these connected?

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

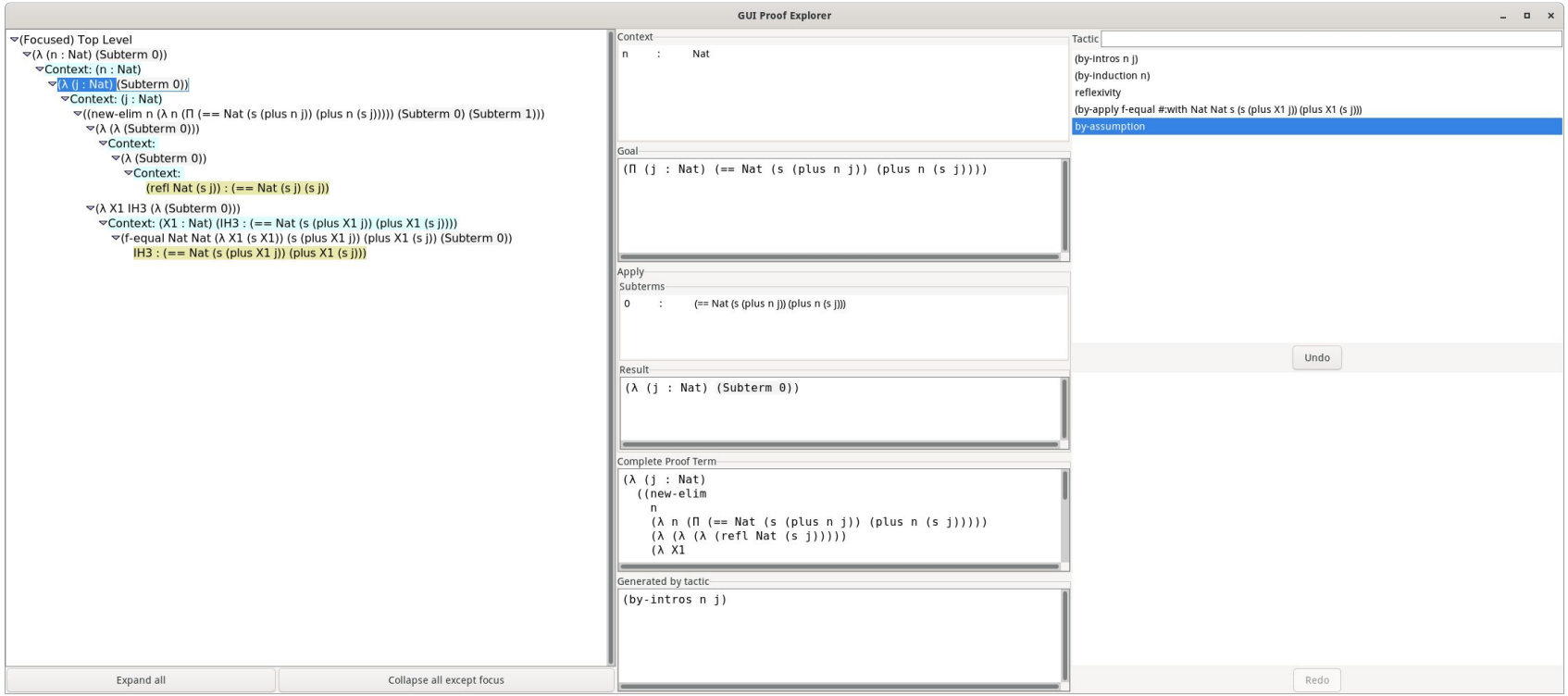
```
Theorem add1eq : forall (n : nat)
(j : nat), (n + j).+1 = (n + j.+1).
intros n j.
elim n.
reflexivity.
intros n1 IH.
 have why_doesnt_simpl_do_this :
(addn (S n1) (S j)) = (S (addn n1
(S j))).  reflexivity.
 rewrite why_doesnt_simpl_do_this.
by f_equal.
Qed.
```

Bottom-up proofs

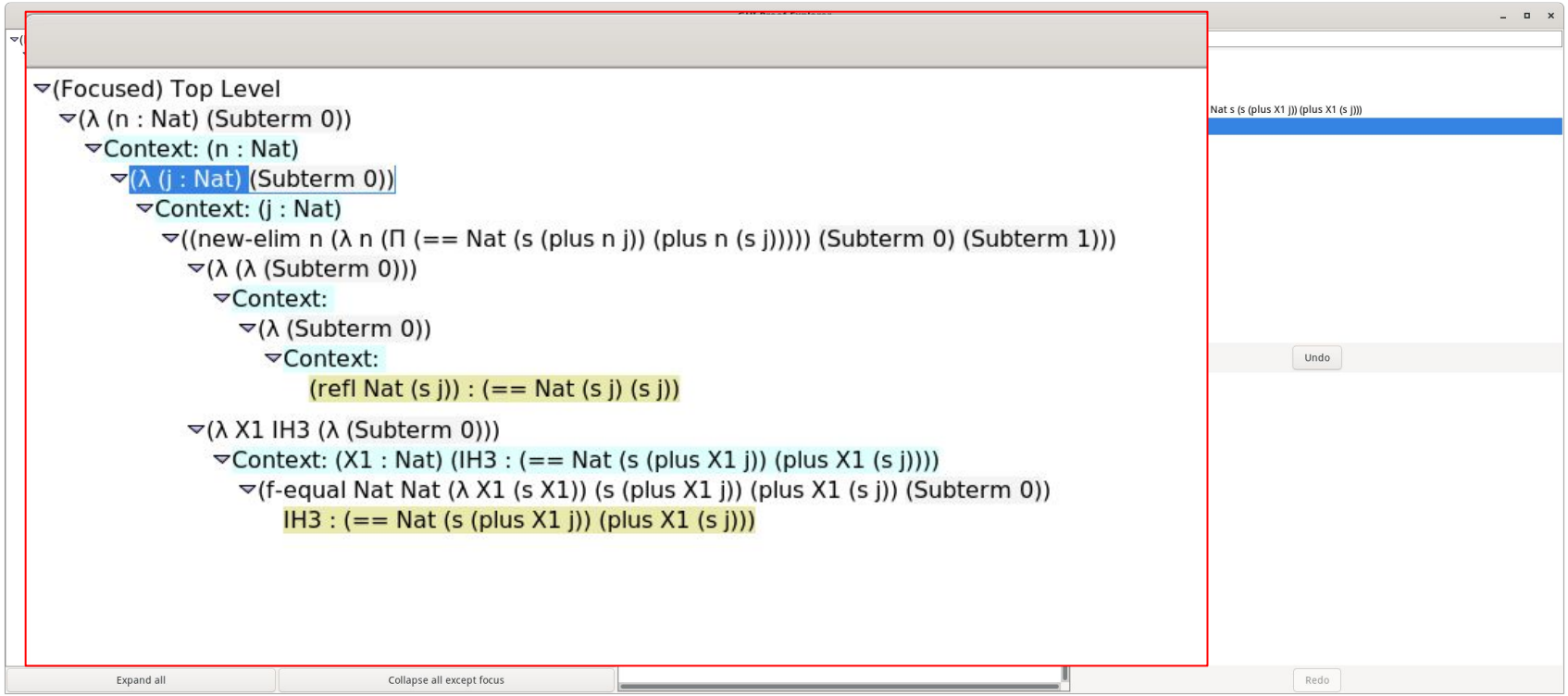
+ Explicit
+ Transparent

− Verbose- no automation
− Most proofs aren't book chapters

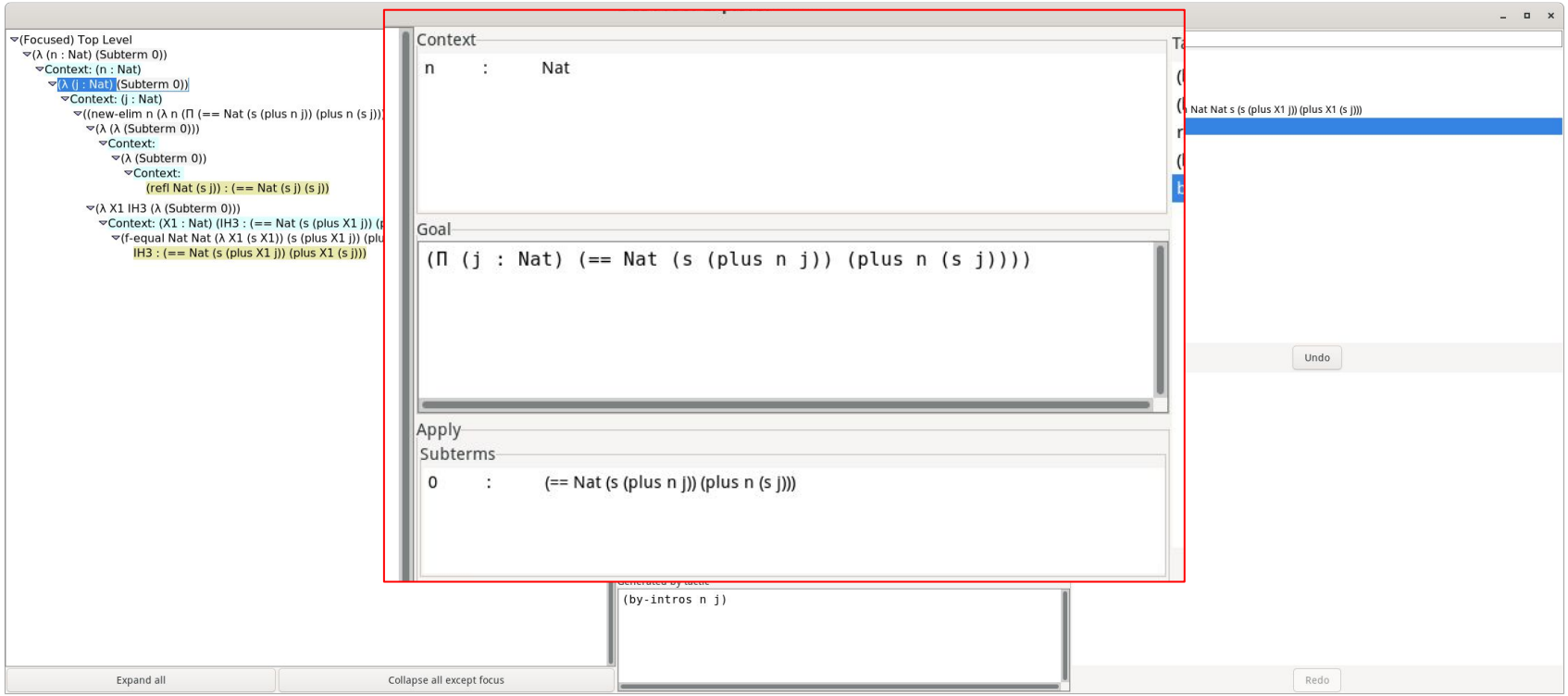
Introducing ProofViz



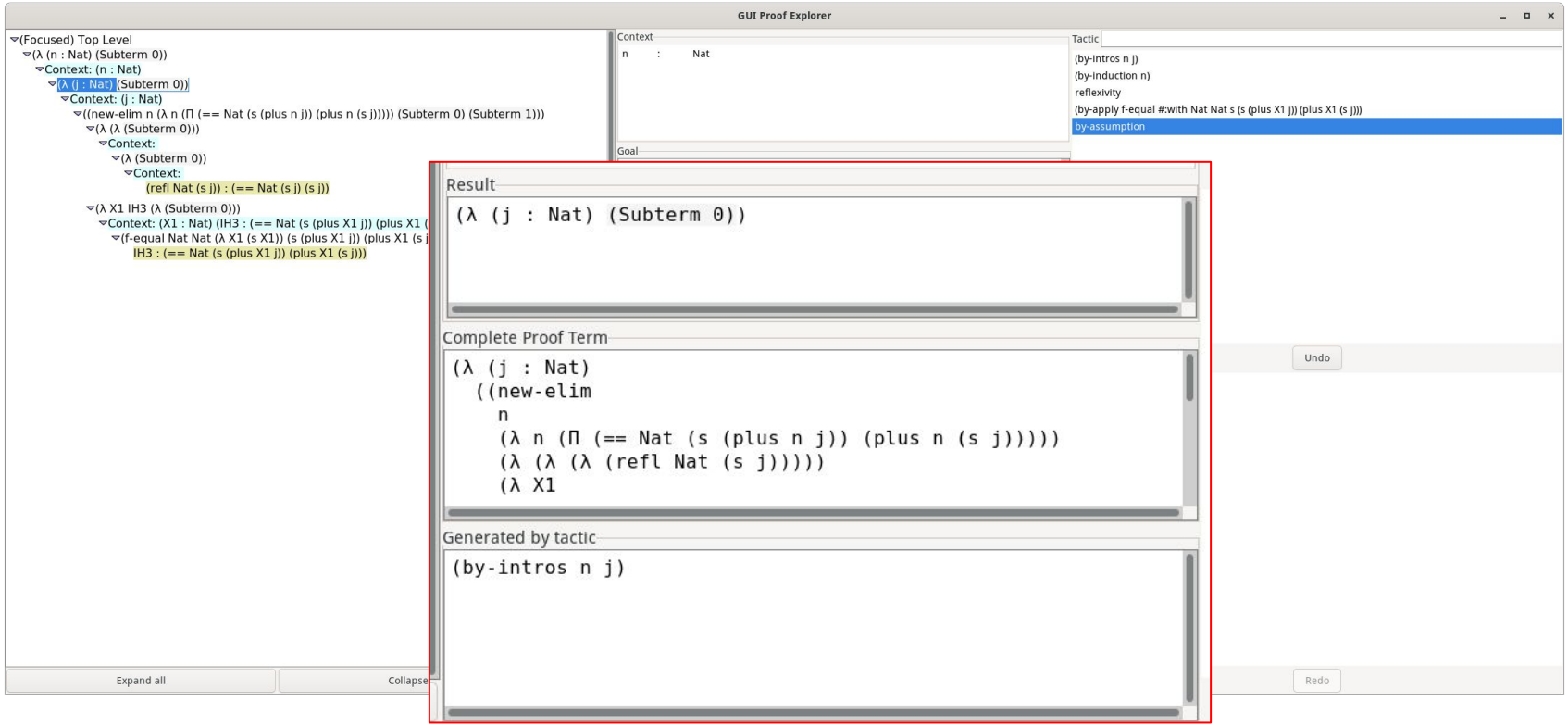
ProofViz: Tree View



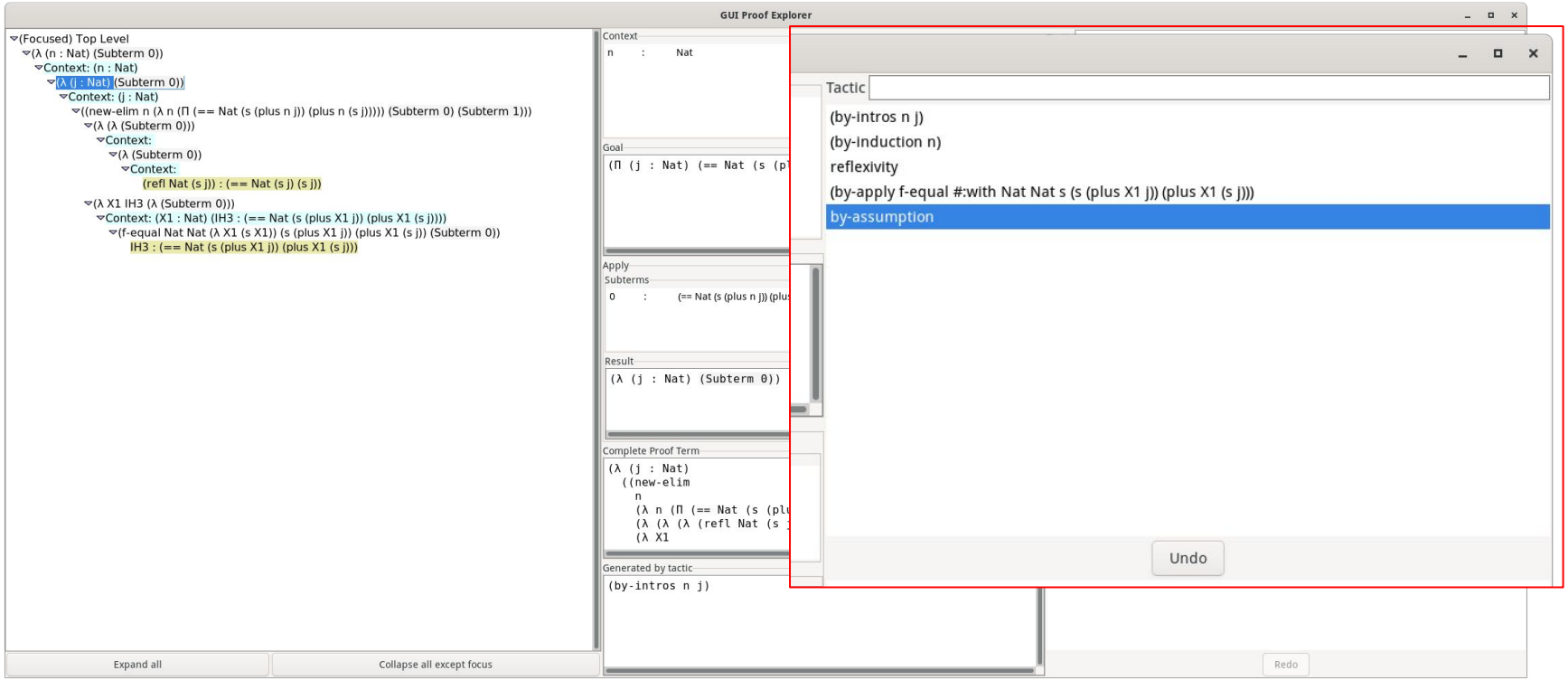
ProofViz: Node Information



ProofViz: Node Information



ProofViz: Interaction



ProofViz: Goals

+ Explicit
+ Transparent

ProofViz: Goals

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

Cur

- A dependently-typed proof assistant[1]
  - Created with Racket[2]

[1]Dependent Type Systems as Macros, S. Chang et al. (2019)
[2]The Racket Manifesto, M. Felleisen et al. (2015)

Cur

- A dependently-typed proof assistant[1]
  - Created with Racket[2]
- Extensible (experiment without core language modification)
  - New tactic systems: Ntac
  - Experimental type systems: Sized types
  - Solver integration

[1]Dependent Type Systems as Macros, S. Chang et al. (2019)
[2]The Racket Manifesto, M. Felleisen et al. (2015)

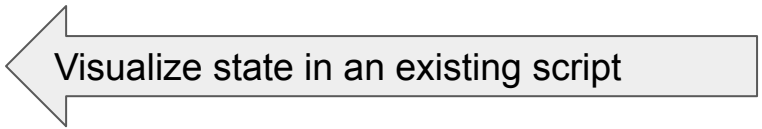
Cur

- A dependently-typed proof assistant[1]
  - Created with Racket[2]
- Extensible (experiment without core language modification)
  - New tactic systems: Ntac
  - Experimental type systems: Sized types
  - Solver integration
  - This presentation: ProofViz

[1]Dependent Type Systems as Macros, S. Chang et al. (2019)
[2]The 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
```

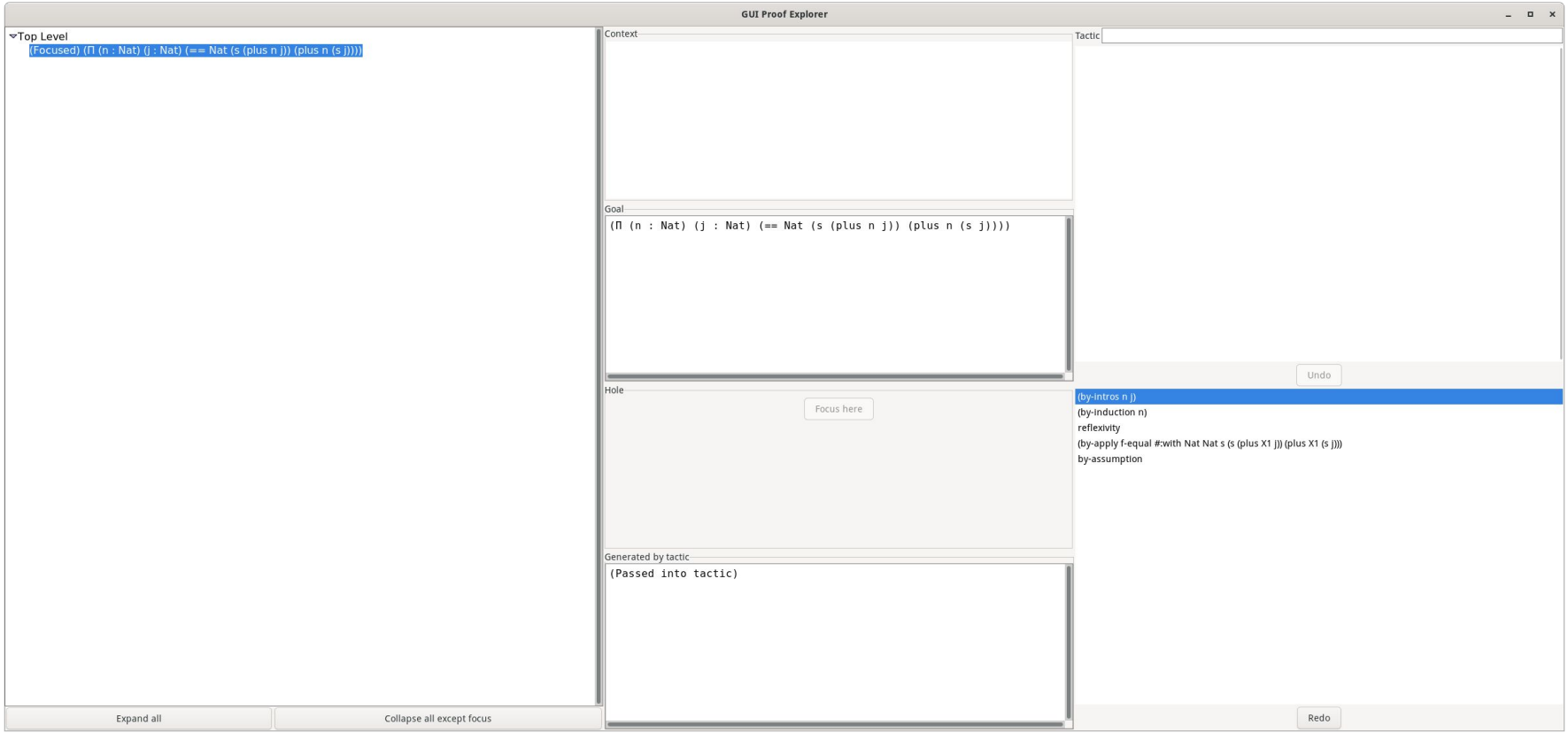


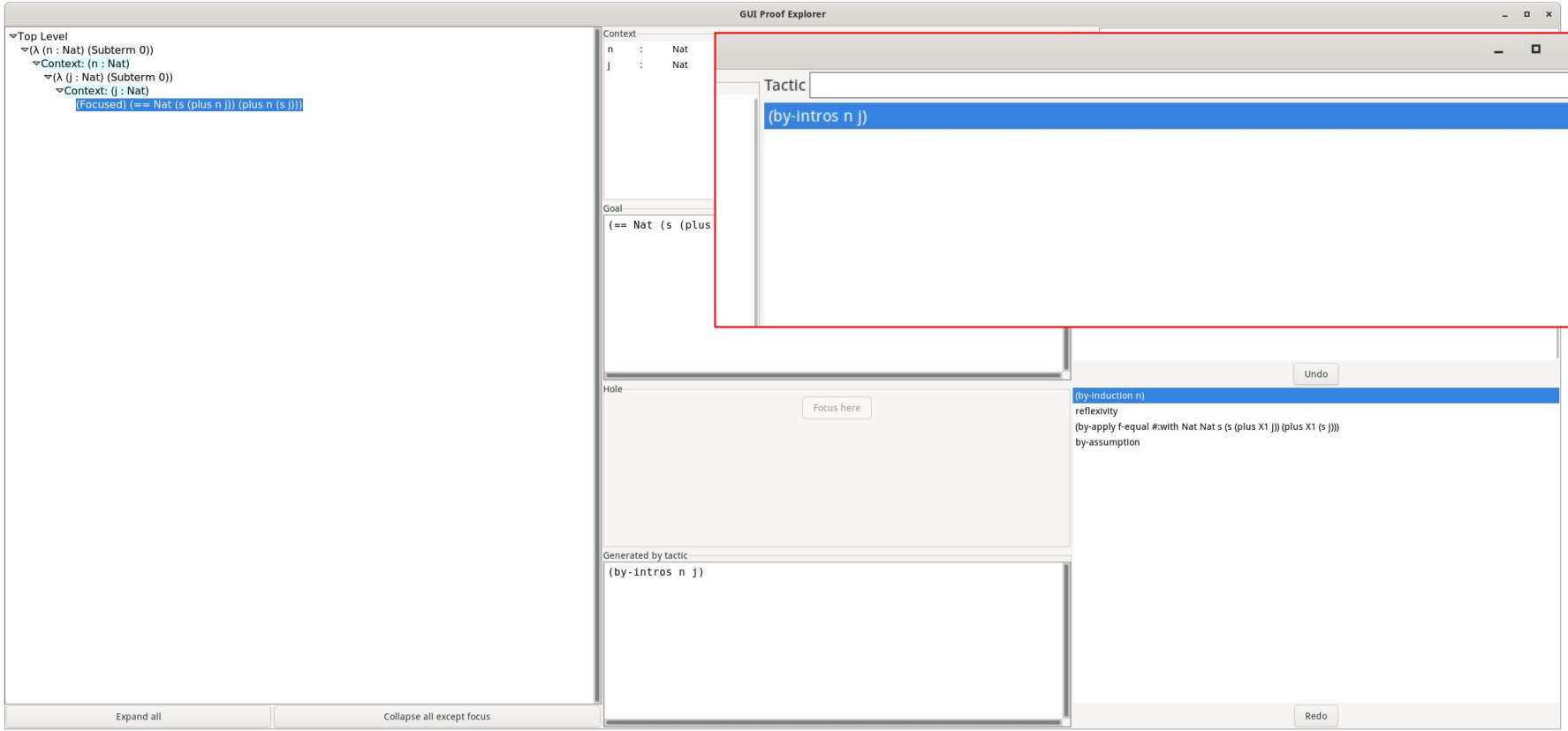
Invoking ProofViz

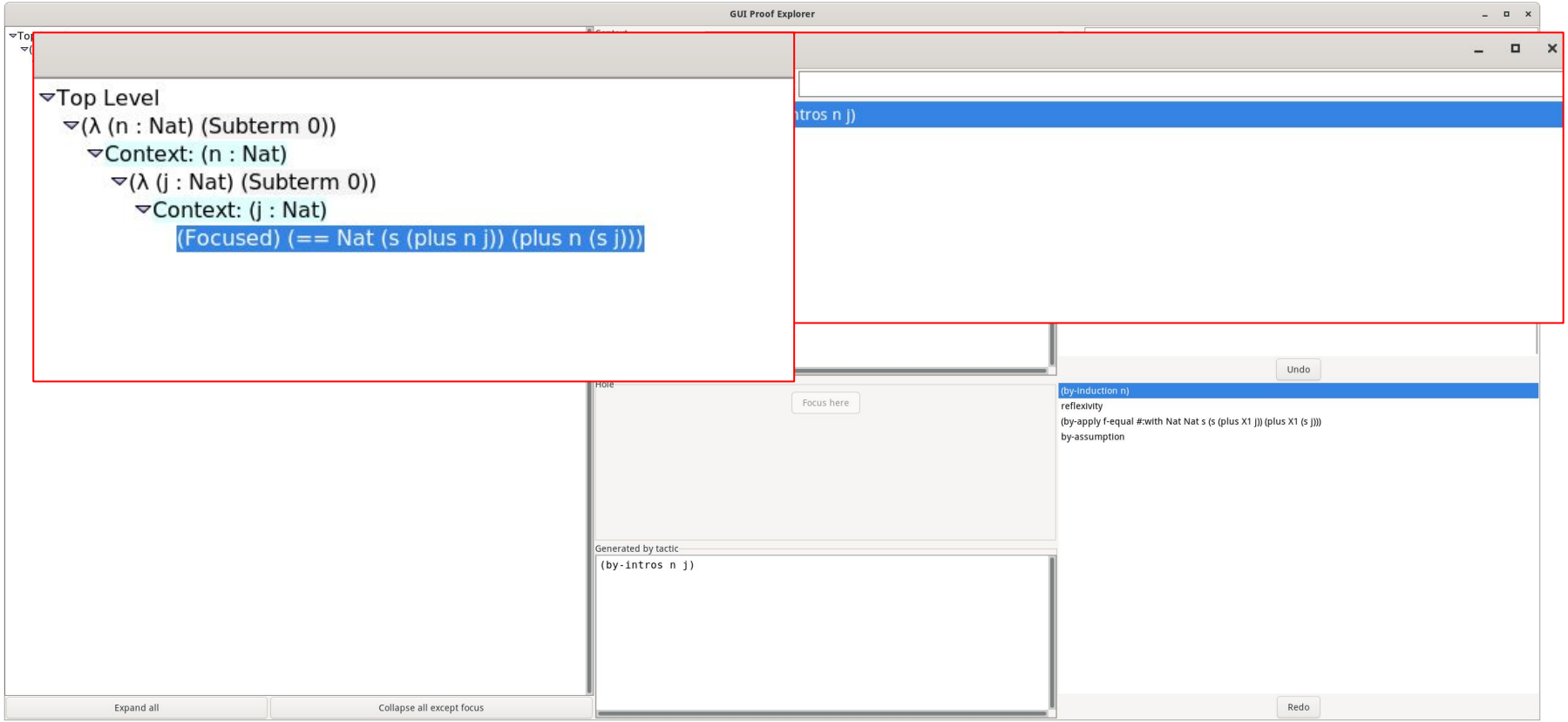
```
(define add1+=+add1
  (ntac/visual (Π (n : Nat)
                  (j : Nat)
                  (== Nat
                      (s (plus n j))
                      (plus n (s j))))))
```

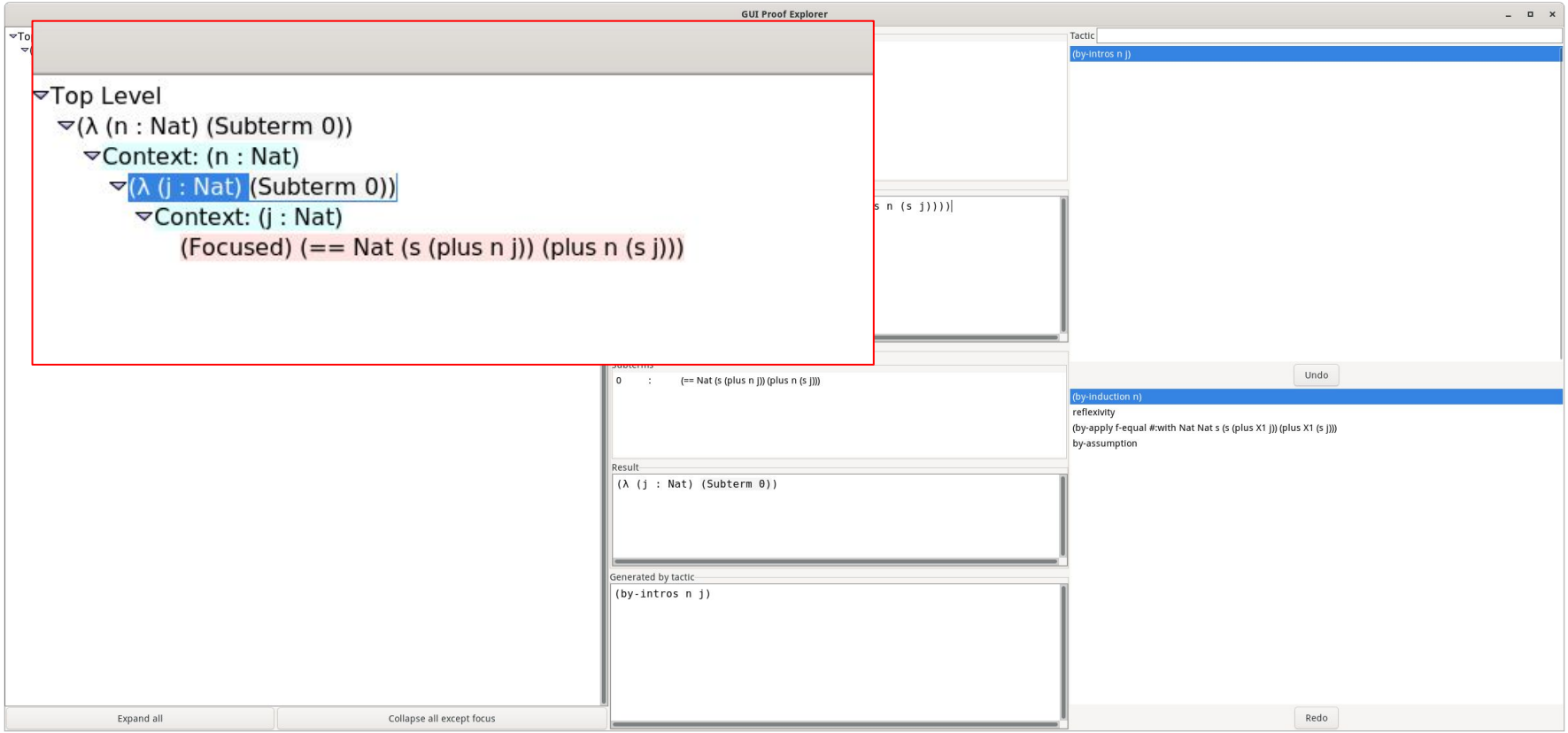
Invoking ProofViz

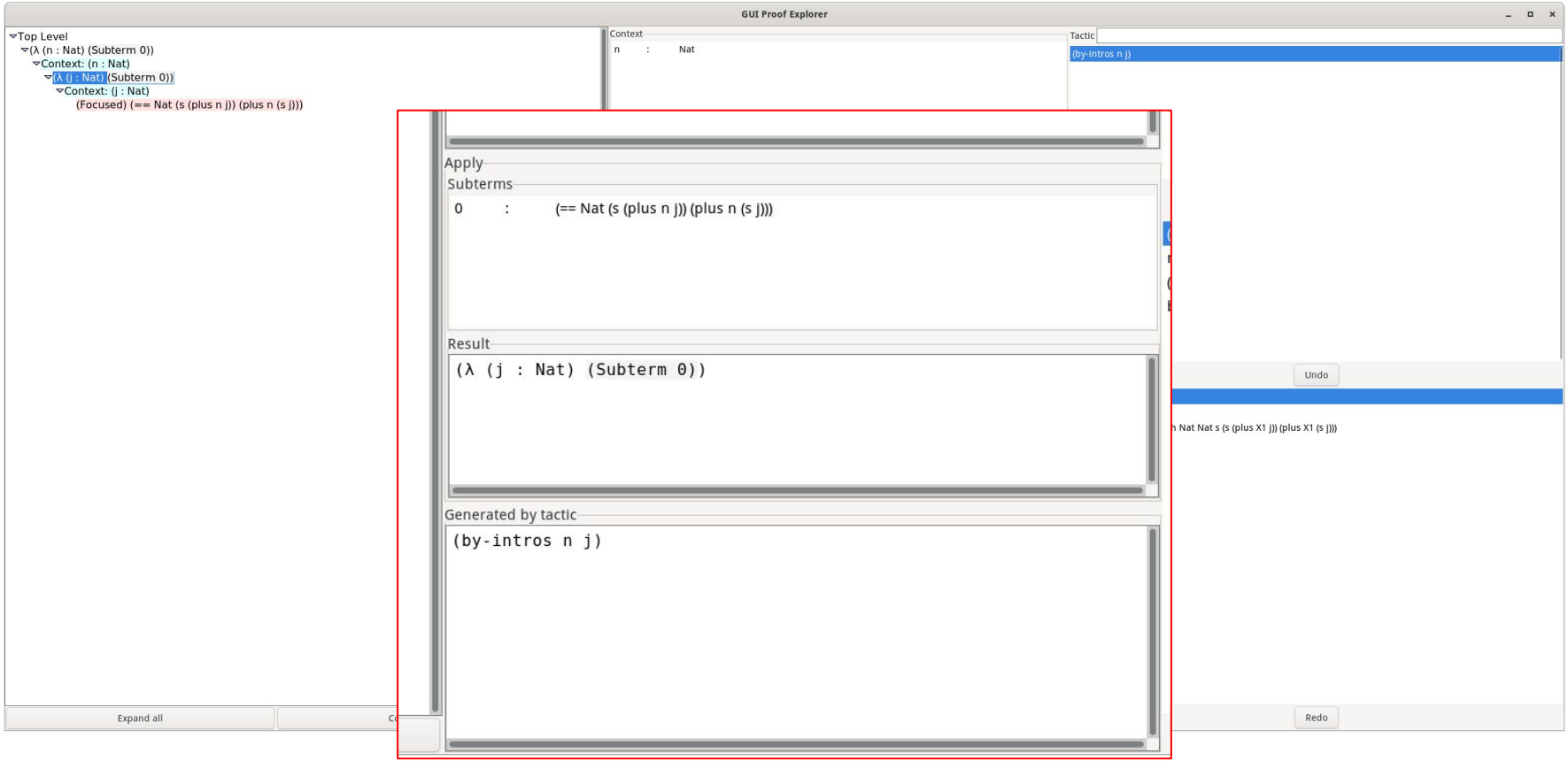
```
(define add1+=+add1
  (ntac/visual (Π (n : Nat)
                  (j : Nat)
                  (== Nat
                      (s (plus n j))
                      (plus n (s j))))
       (by-intros n j)
       (by-induction n)
       reflexivity
       (by-apply f-equal #:with Nat Nat s (s (plus X1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

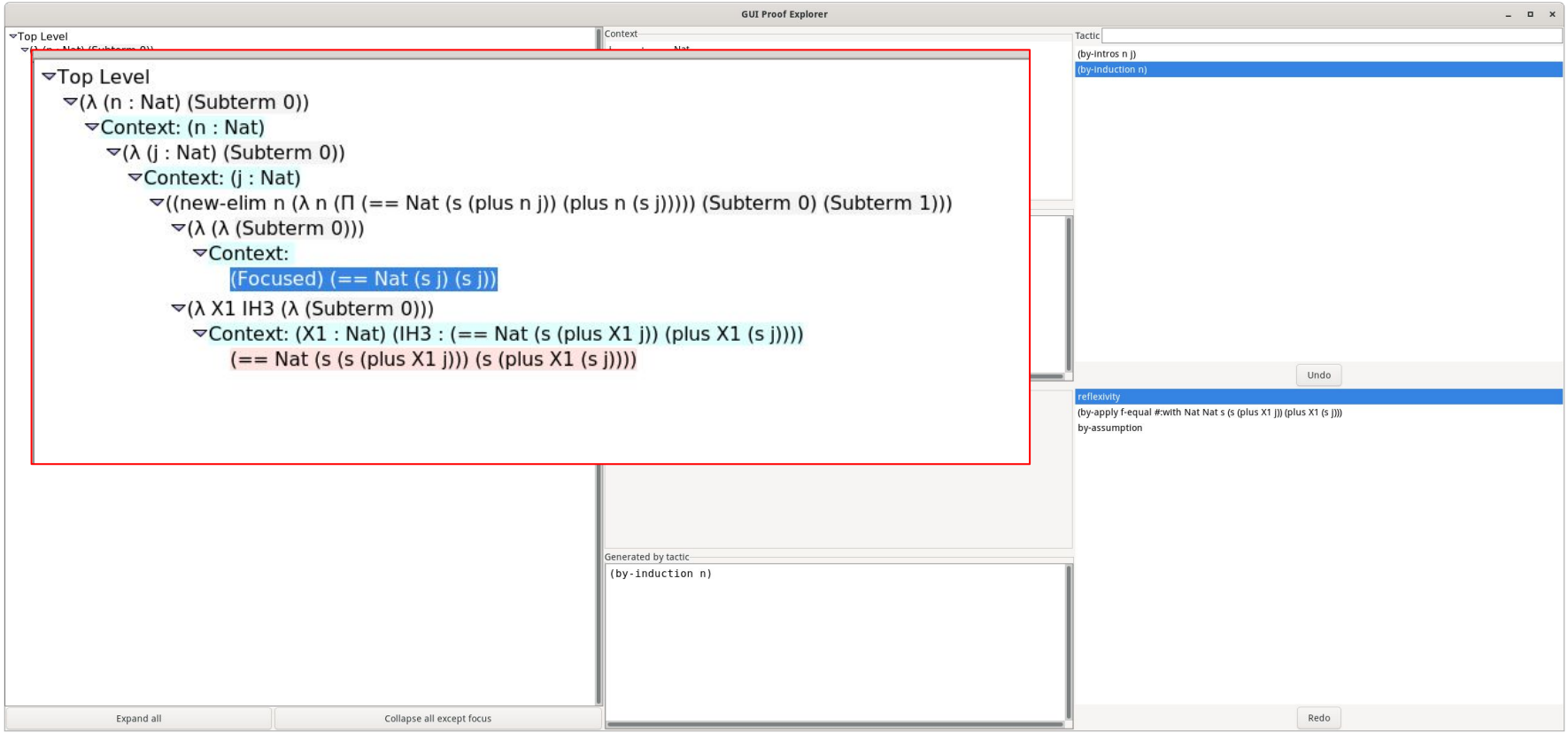

GUI Proof Explorer
Top Level
(Focused) (Π (n : Nat) (j : Nat) (== Nat (s (plus n j)) (plus n (s j))))
Expand all
Collapse all except focus
Context
Goal
(Π (n : Nat) (j : Nat) (== Nat (s (plus n j)) (plus n (s j))))
Hole
Focus here
Generated by tactic
(Passed into tactic)
Tactic
Undo
(by-intros n j)
(by-induction n)
reflexivity
(by-apply f-equal #:with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))
by-assumption
Redo

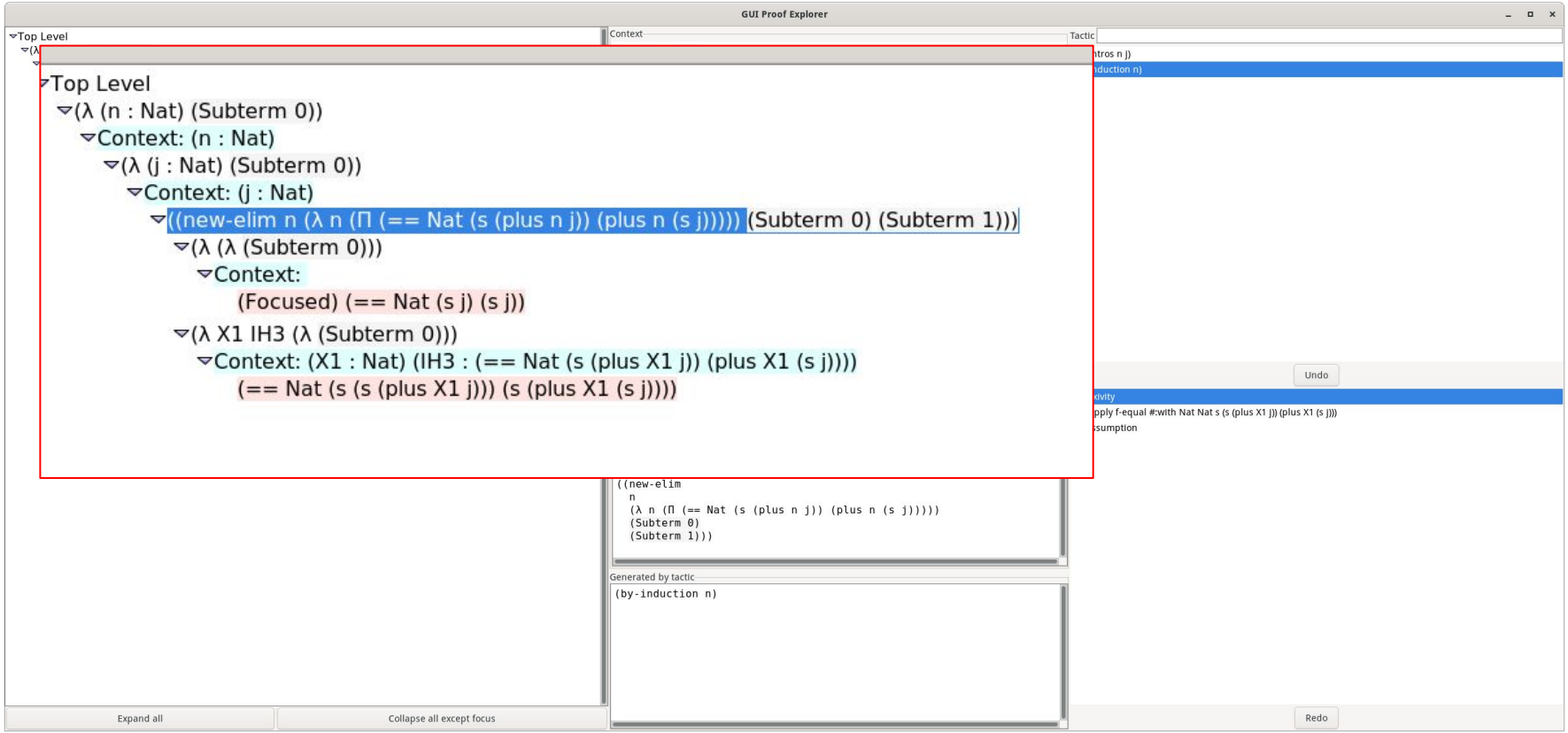

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
(Focused) (== Nat (s (plus n j)) (plus n (s j)))
Context
n : Nat
j : Nat
Goal
(== Nat (s (plus
Hole
Focus here
Generated by tactic
(by-intros n j)
Tactic
(by-intros n j)
Undo
(by-induction 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

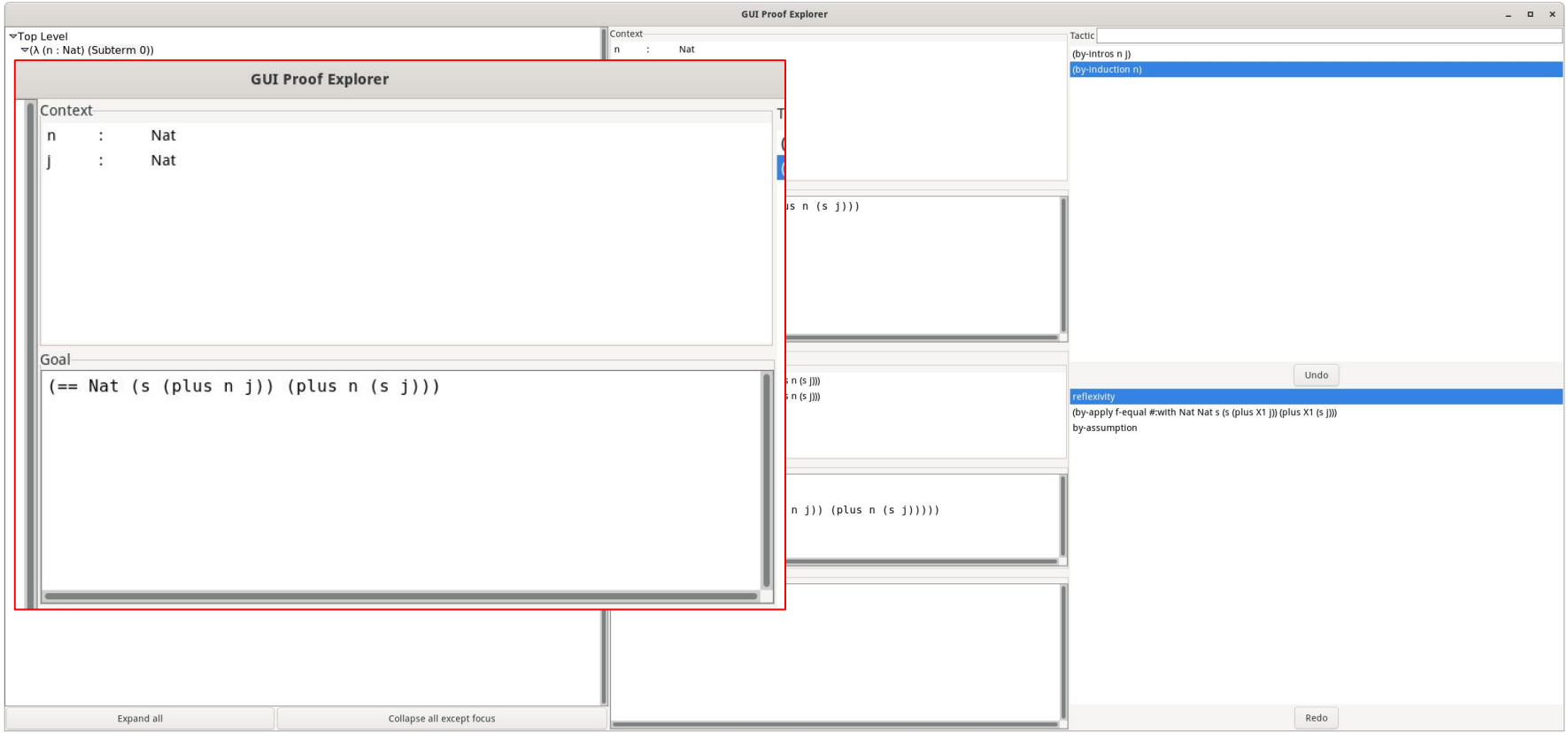

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
(Focused) (== Nat (s (plus n j)) (plus n (s j)))
ntros n j)
Undo
Hole
Focus here
(by-induction n)
reflexivity
(by-apply f-equal #:with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))
by-assumption
Generated by tactic
(by-intros n j)
Expand all
Collapse all except focus
Redo

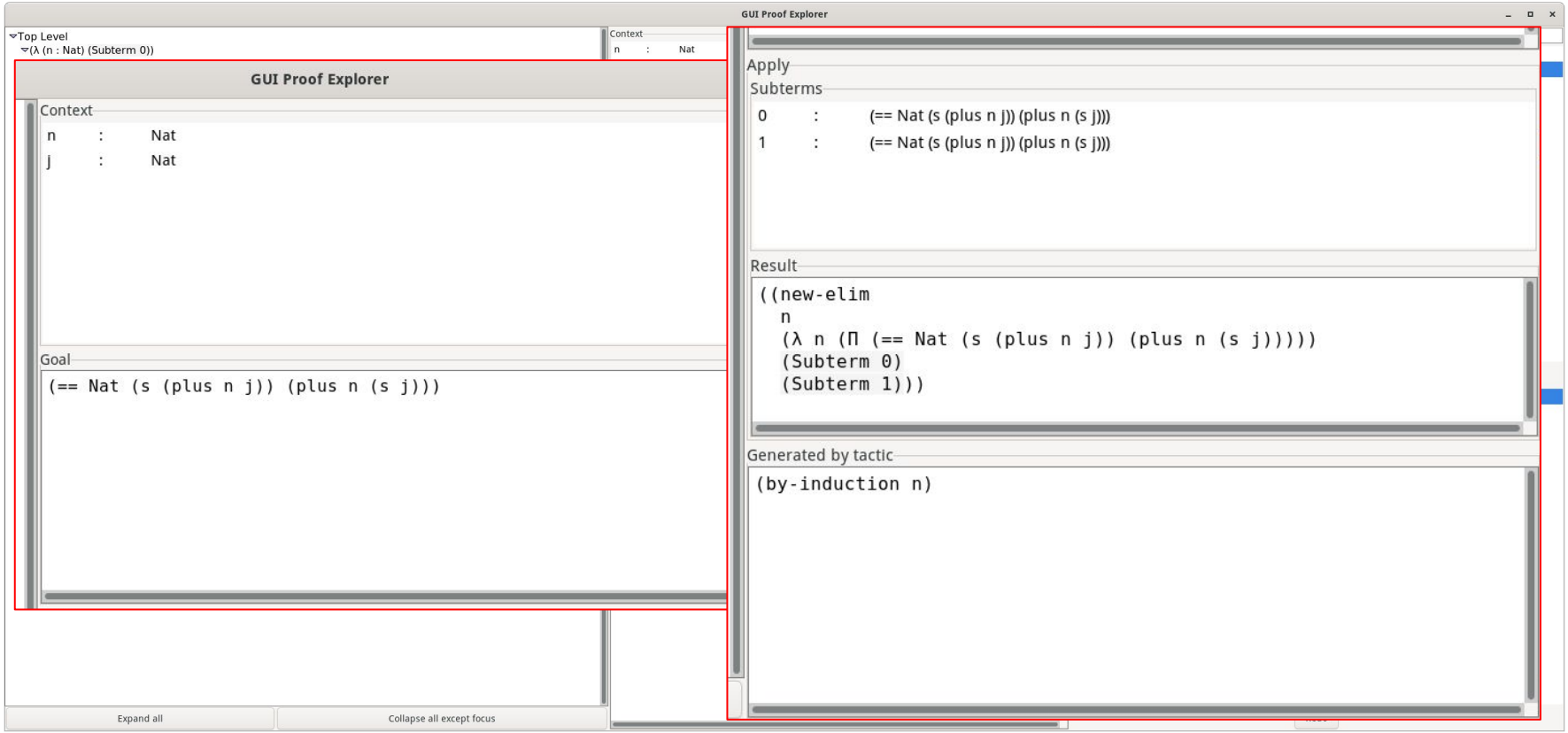

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
(Focused) (== Nat (s (plus n j)) (plus n (s j)))
Tactic
(by-intros n j)
s n (s j))))
0 : (== Nat (s (plus n j)) (plus n (s j)))
Result
(λ (j : Nat) (Subterm 0))
Generated by tactic
(by-intros 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
Expand all
Collapse all except focus
Redo

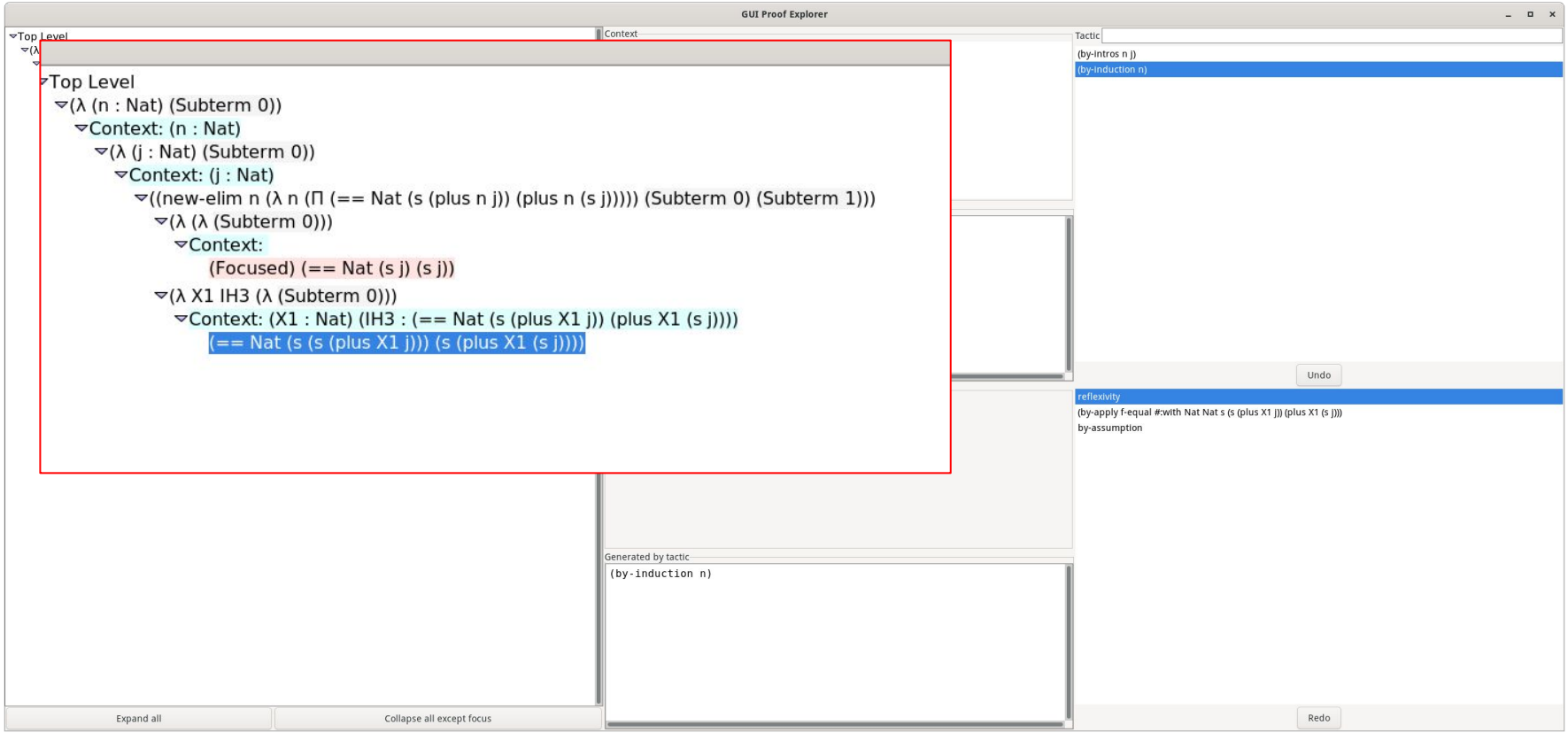

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
(Focused) (== Nat (s (plus n j)) (plus n (s j)))
Context
n : Nat
Tactic
(by-intros n j)
Apply
Subterms
0 : (== Nat (s (plus n j)) (plus n (s j)))
Result
(λ (j : Nat) (Subterm 0))
Generated by tactic
(by-intros n j)
Undo
Redo
Expand all

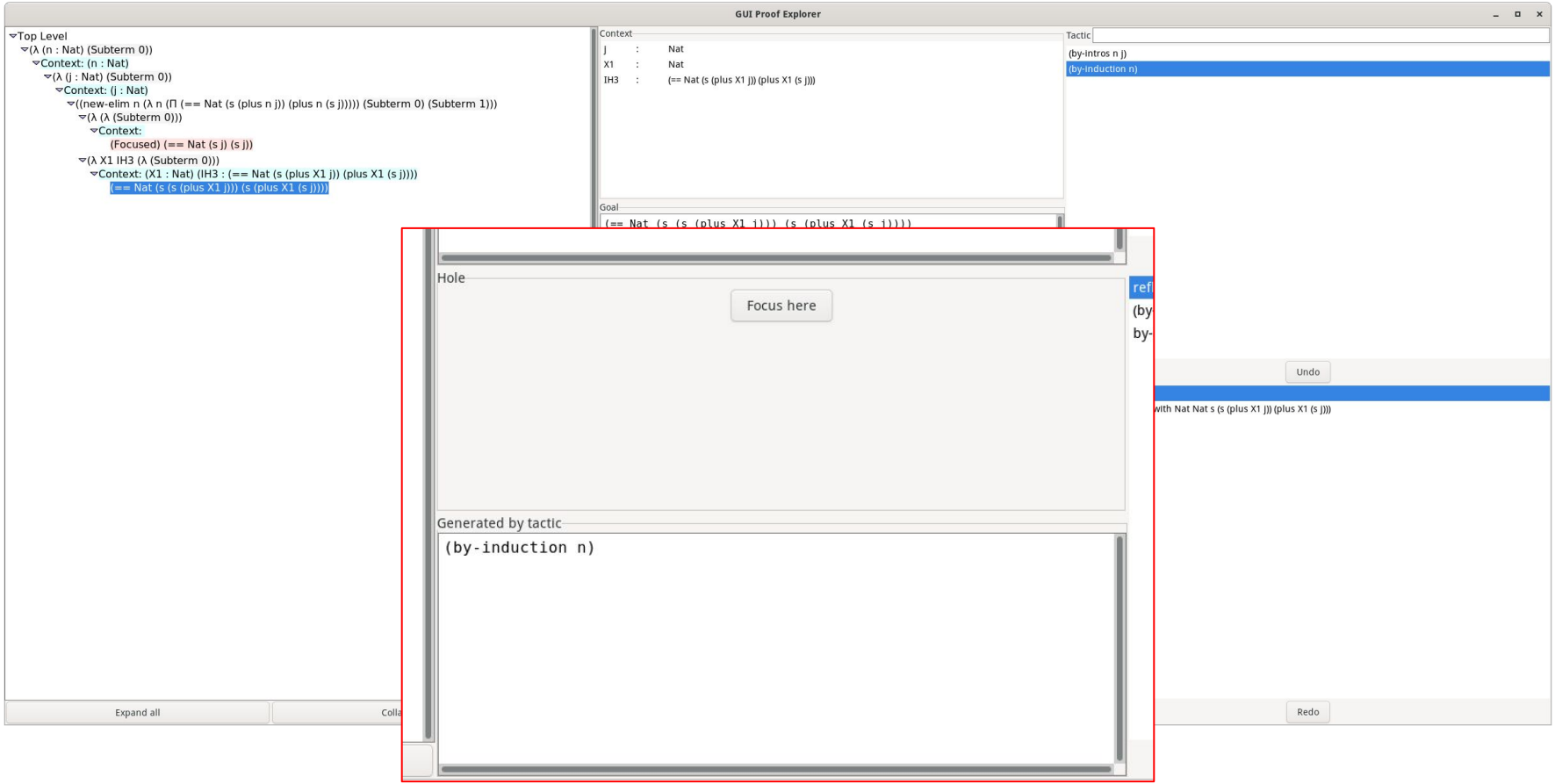

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))
(λ (λ (Subterm 0)))
Context:
(Focused) (== Nat (s j) (s j))
(λ X1 IH3 (λ (Subterm 0)))
Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))
(== Nat (s (s (plus X1 j))) (s (plus X1 (s j))))
Context
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
Generated by tactic
(by-induction n)
Expand all
Collapse all except focus
Redo

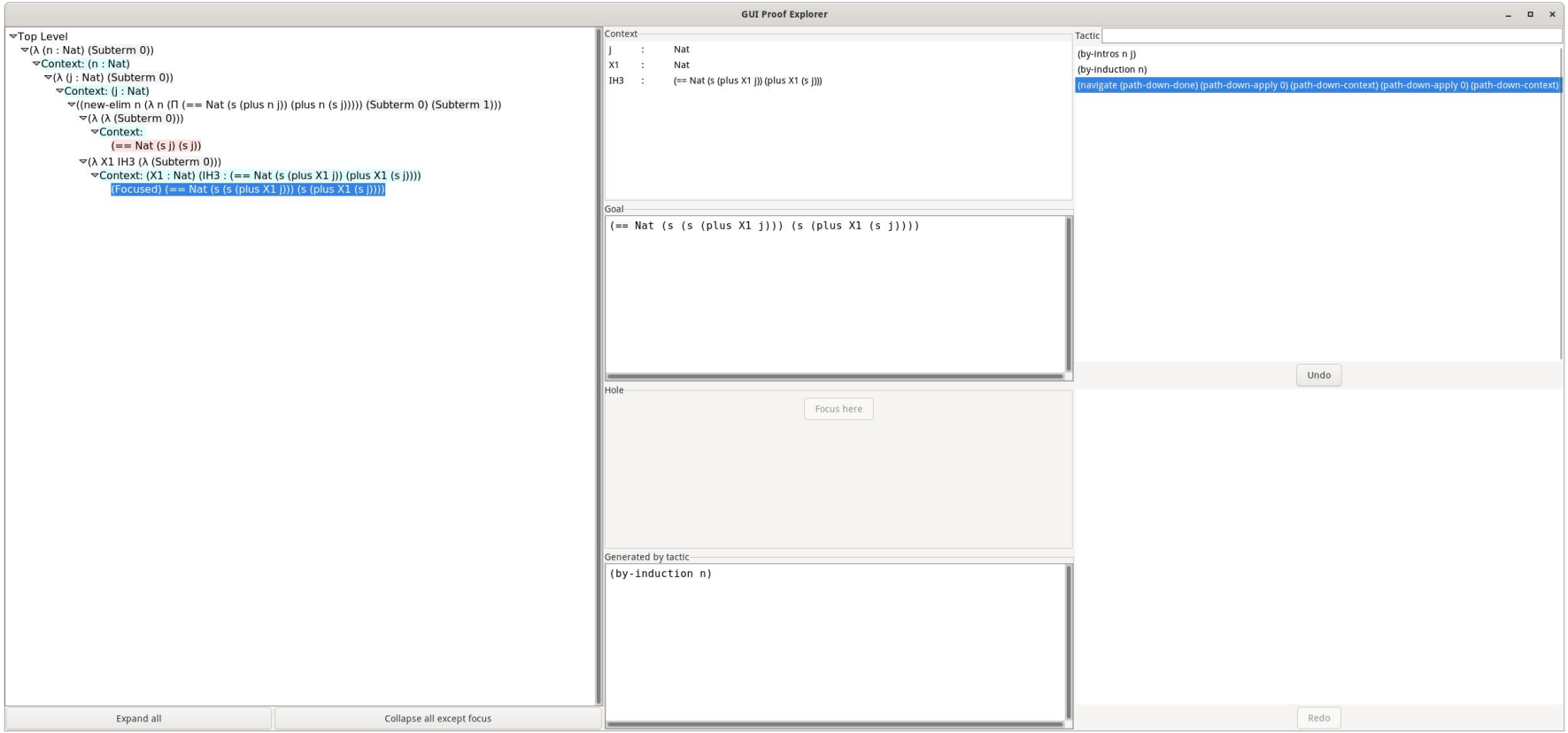

GUI Proof Explorer
Top Level
Context
Tactic
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))
(λ (λ (Subterm 0)))
Context:
(Focused) (== Nat (s j) (s j))
(λ X1 IH3 (λ (Subterm 0)))
Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))
(== Nat (s (s (plus X1 j))) (s (plus X1 (s j))))
((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)
Undo
Redo
Expand all
Collapse all except focus

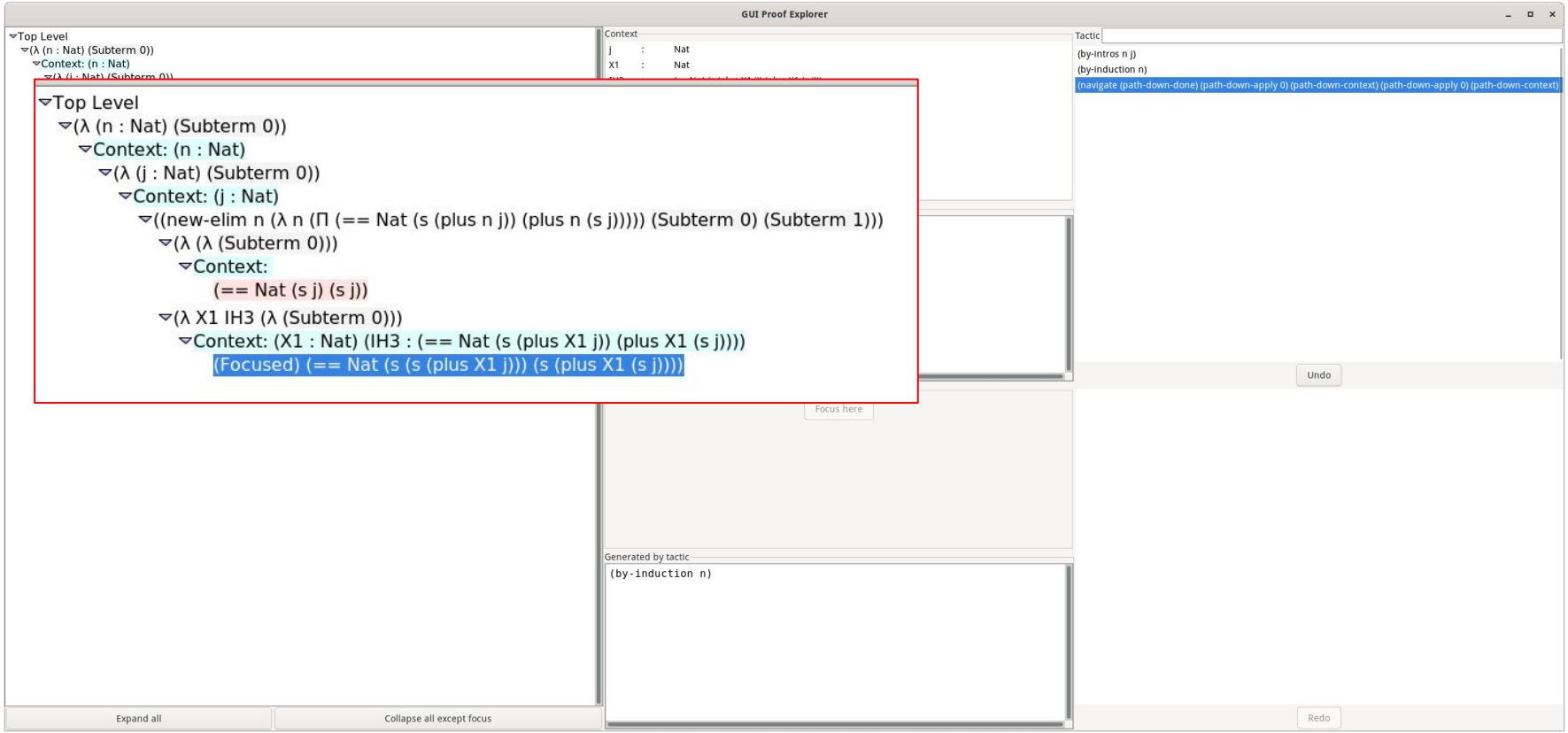

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context
n : Nat
Tactic
(by-intros n j)
(by-induction n)
GUI Proof Explorer
Context
n : Nat
j : Nat
Goal
(== Nat (s (plus n j)) (plus n (s j)))
Undo
reflexivity
(by-apply f-equal #:with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))
by-assumption
Redo
Expand all
Collapse all except focus

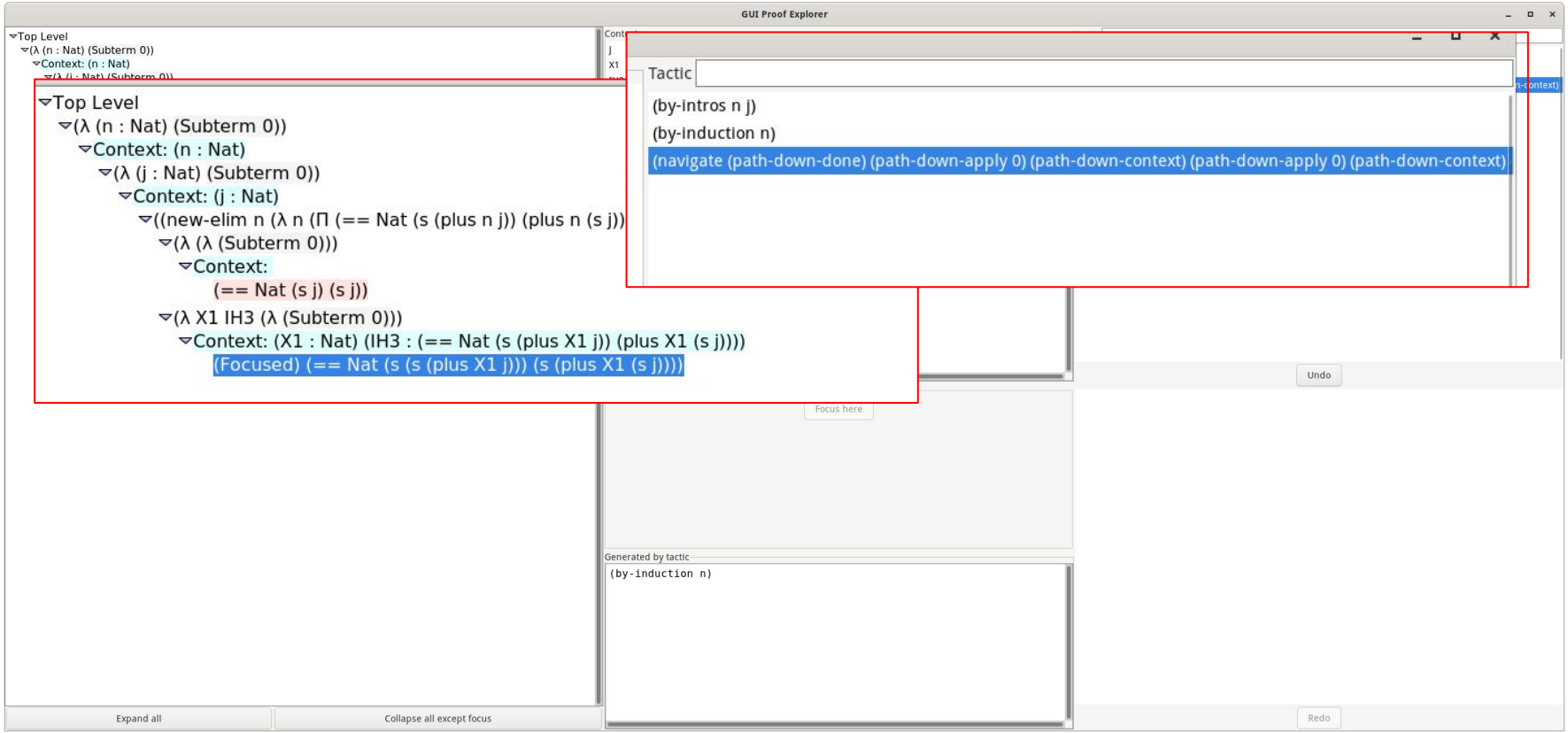

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context
n : Nat
GUI Proof Explorer
Context
n : Nat
j : Nat
Goal
(== Nat (s (plus n j)) (plus n (s j)))
Expand all
Collapse all except focus
GUI Proof Explorer
Apply
Subterms
0 : (== Nat (s (plus n j)) (plus n (s j)))
1 : (== Nat (s (plus n j)) (plus 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)

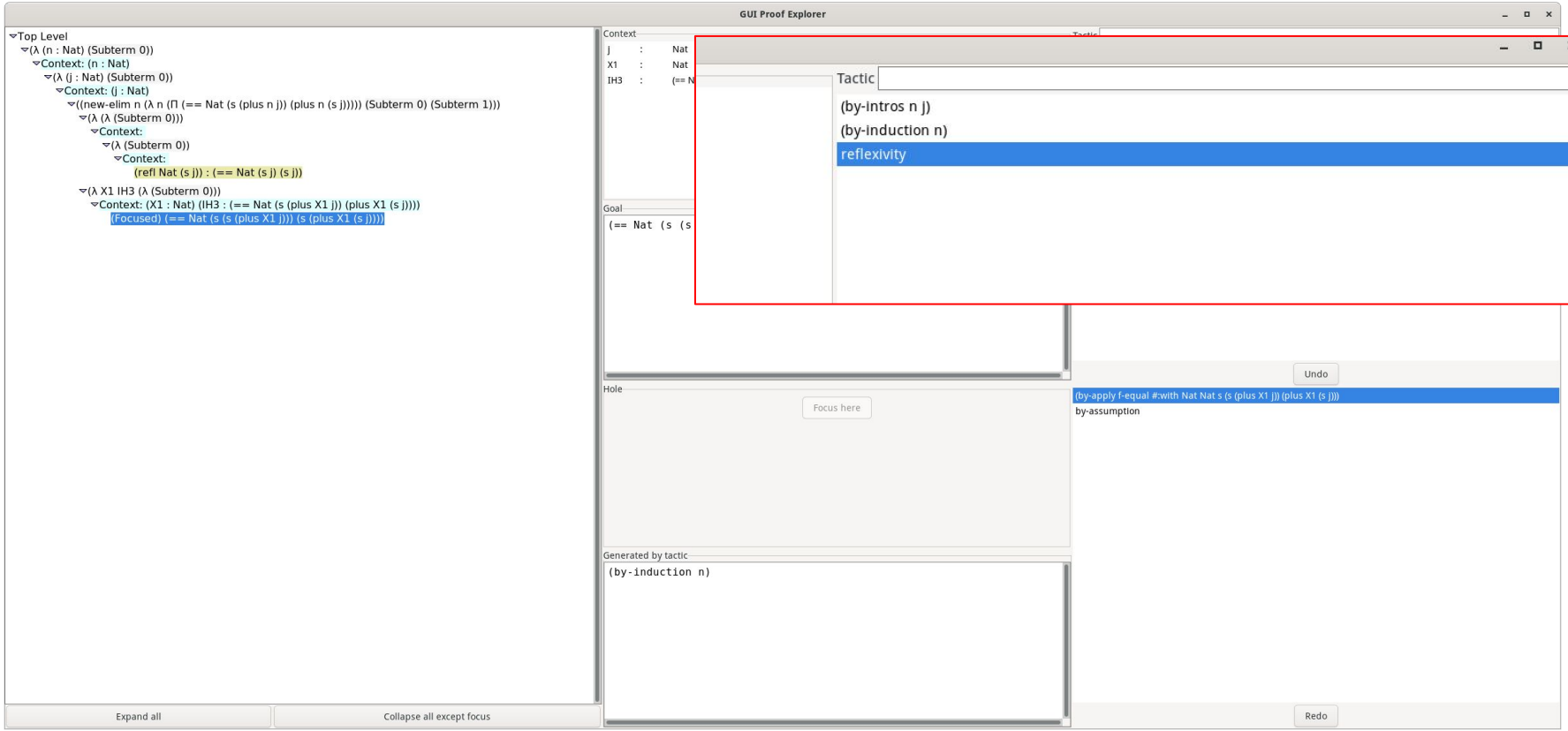

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))
(λ (λ (Subterm 0)))
Context:
(Focused) (== Nat (s j) (s j))
(λ X1 IH3 (λ (Subterm 0)))
Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))
(== Nat (s (s (plus X1 j))) (s (plus X1 (s j))))
Context
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
Generated by tactic
(by-induction n)
Expand all
Collapse all except focus
Redo

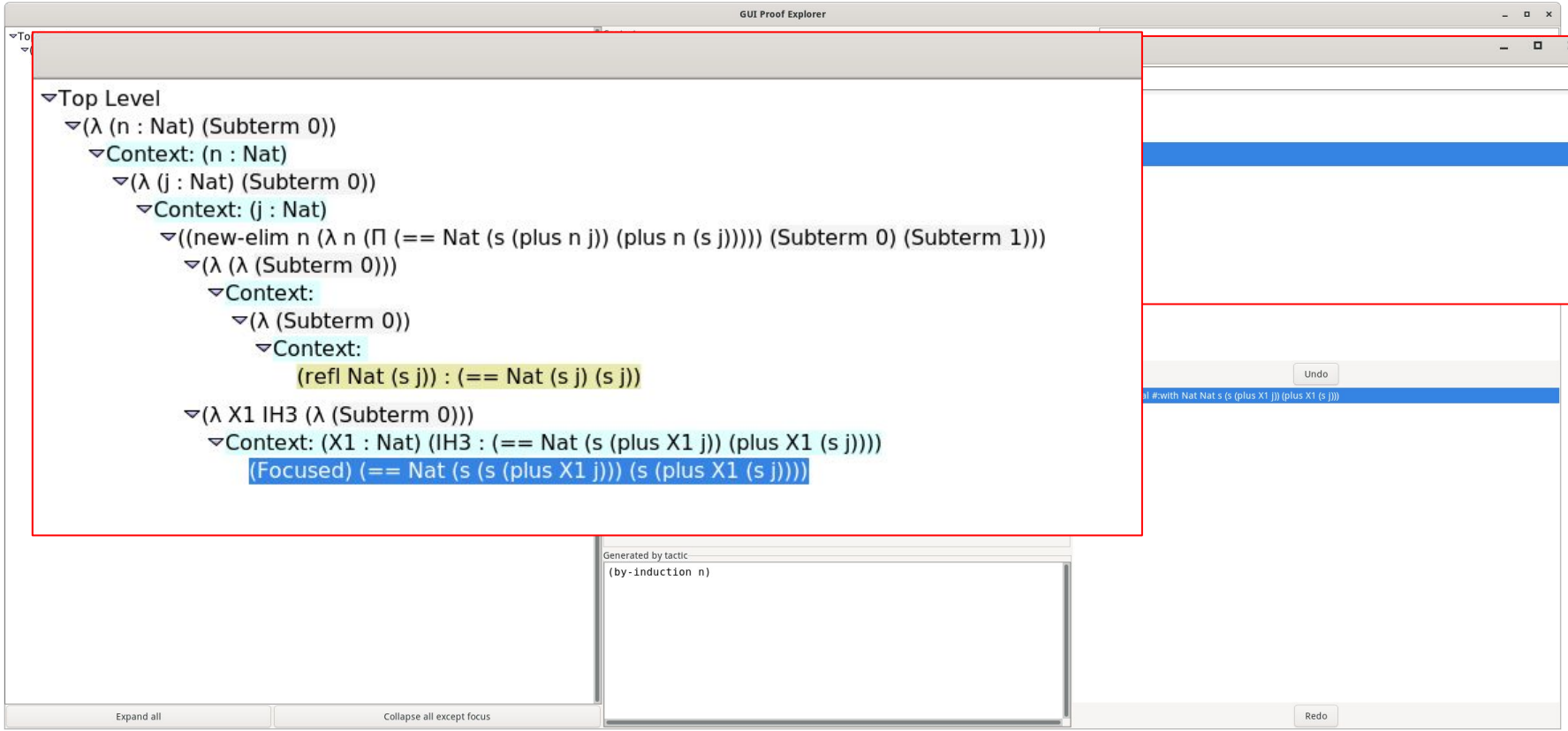

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))
(λ (λ (Subterm 0)))
Context:
(Focused) (== Nat (s j) (s j))
(λ X1 IH3 (λ (Subterm 0)))
Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))
(== Nat (s (s (plus X1 j))) (s (plus X1 (s j))))
Expand all
Context
j : Nat
X1 : Nat
IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j)))
Goal
(== Nat (s (s (plus X1 j))) (s (plus X1 (s j))))
Tactic
(by-intros n j)
(by-induction n)
Undo
Redo
Hole
Focus here
Generated by tactic
(by-induction n)

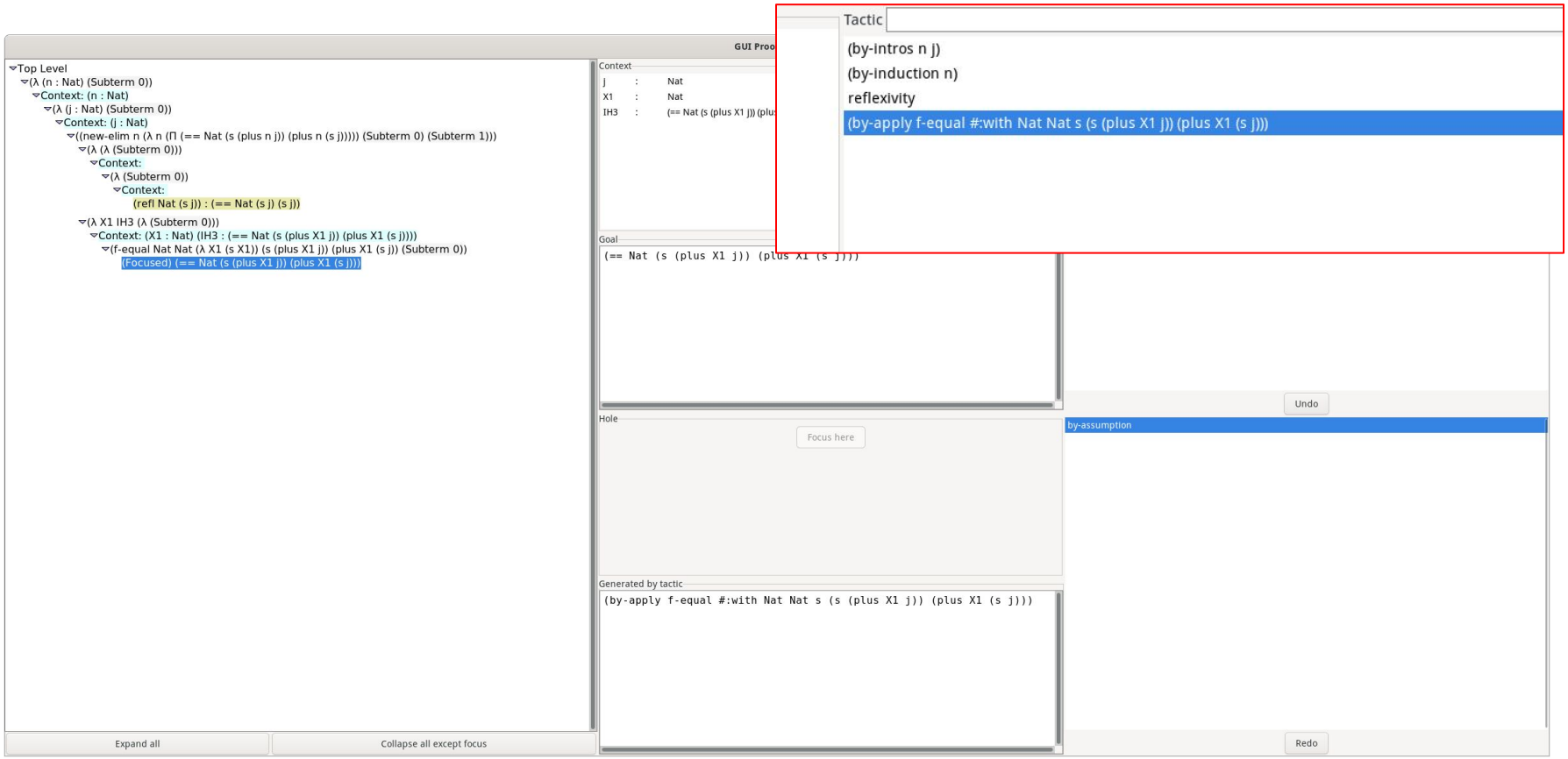

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))
(λ (λ (Subterm 0)))
Context:
(== Nat (s j) (s j))
(λ X1 IH3 (λ (Subterm 0)))
Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j))))
(Focused) (== Nat (s (s (plus X1 j))) (s (plus X1 (s j))))
Expand all
Collapse all except focus
Context
j : Nat
X1 : Nat
IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j)))
Goal
(== Nat (s (s (plus X1 j))) (s (plus X1 (s j))))
Hole
Focus here
Generated by tactic
(by-induction n)
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
Redo

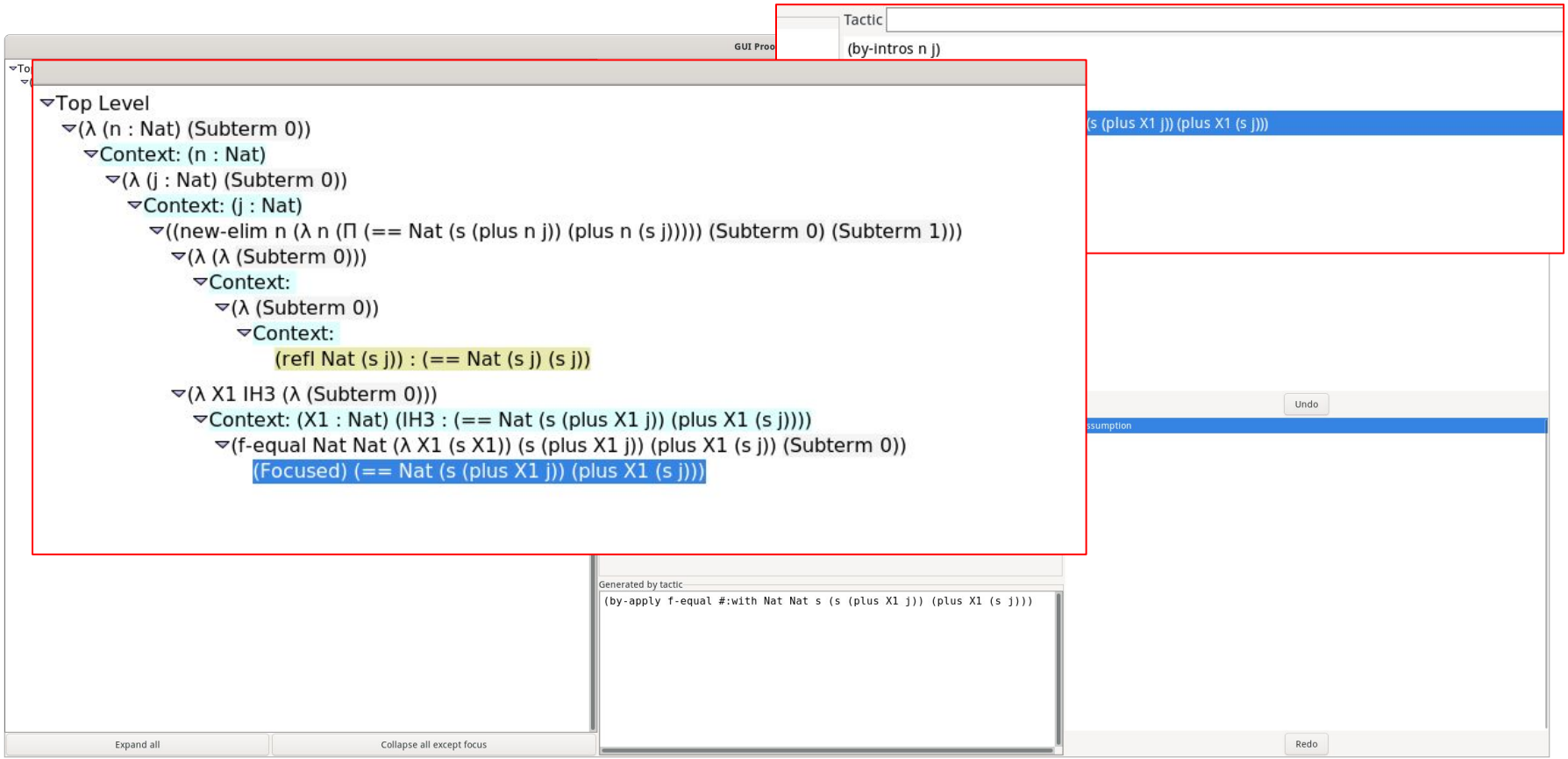

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))
(λ (λ (Subterm 0)))
Context:
(== Nat (s j) (s j))
(λ X1 IH3 (λ (Subterm 0)))
Context: (X1 : Nat) (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

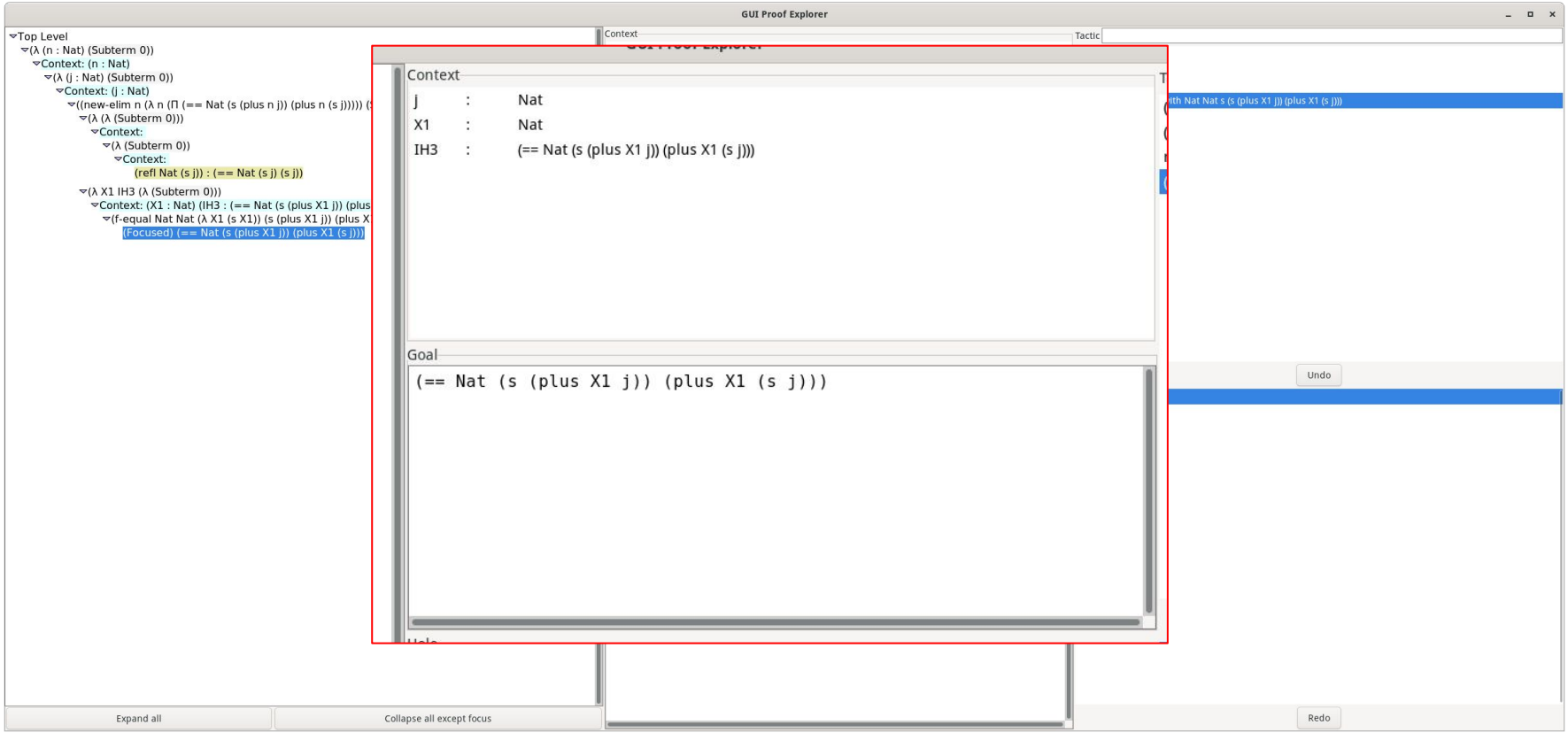

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))
(λ (λ (Subterm 0)))
Context:
(== Nat (s j) (s j))
(λ X1 IH3 (λ (Subterm 0)))
Context: (X1 : Nat) (IH3 : (== Nat (s (plus X1 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)
Undo
Focus here
Generated by tactic
(by-induction n)
Expand all
Collapse all except focus
Redo

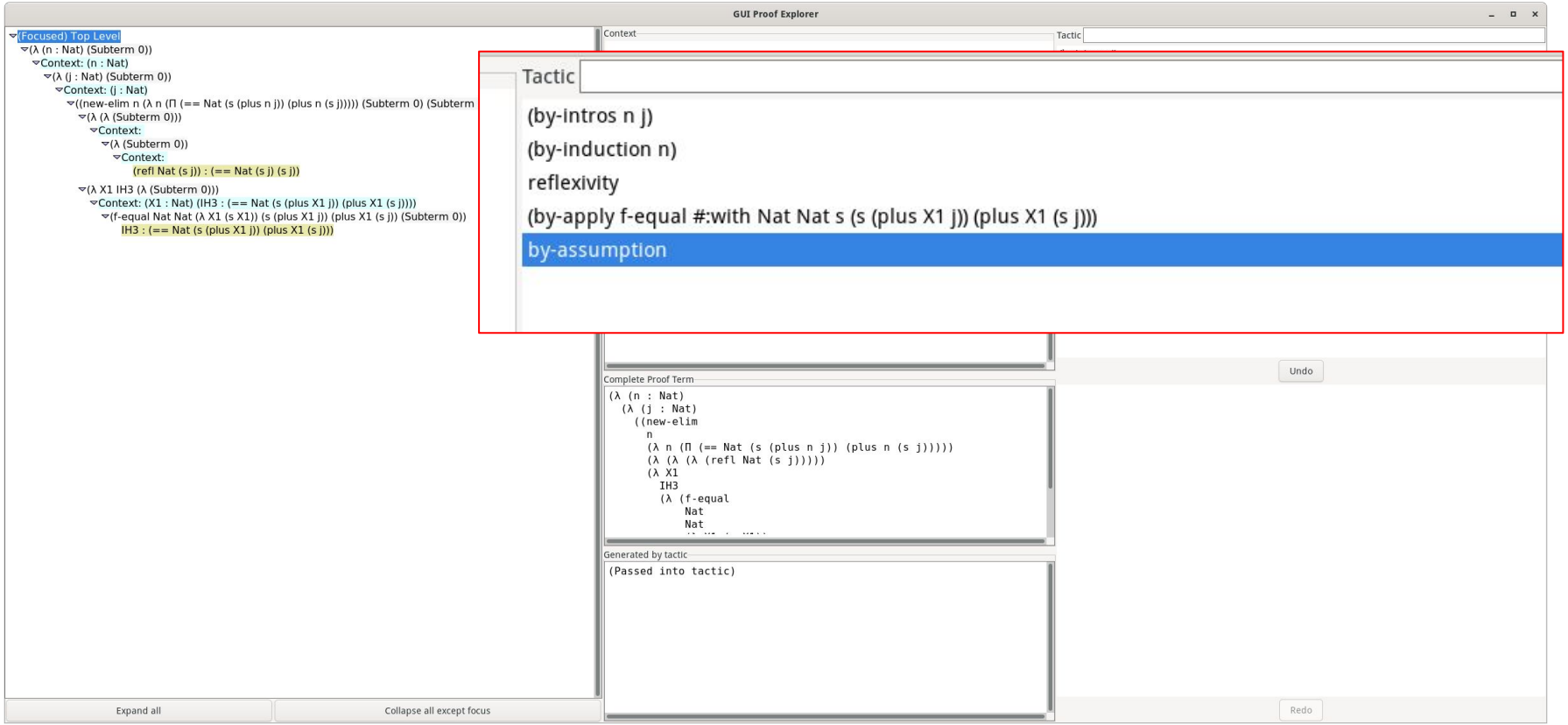

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))
(λ (λ (Subterm 0)))
Context:
(λ (Subterm 0))
Context:
(refl Nat (s j)) : (== Nat (s j) (s j))
(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))))
Expand all
Collapse all except focus
Context
J : Nat
X1 : Nat
IH3 : (== N
Goal
(== Nat (s (s
Hole
Focus here
Generated by tactic
(by-induction n)
Tactic
(by-intros n j)
(by-induction n)
reflexivity
Undo
(by-apply f-equal #:with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))
by-assumption
Redo

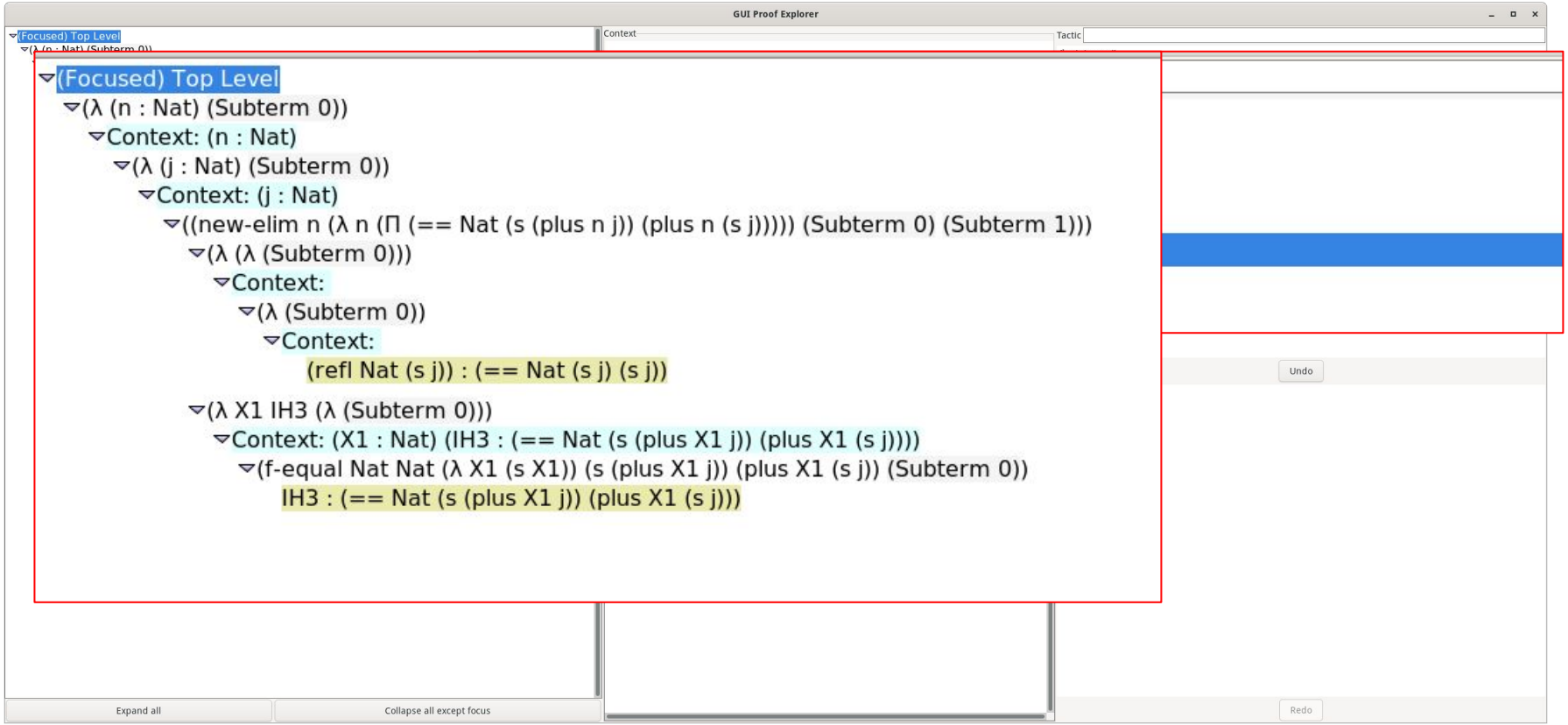

GUI Proof Explorer
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))
(λ (λ (Subterm 0)))
Context:
(λ (Subterm 0))
Context:
(refl Nat (s j)) : (== Nat (s j) (s j))
(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))))
Undo
Generated by tactic
(by-induction n)
Expand all
Collapse all except focus
Redo

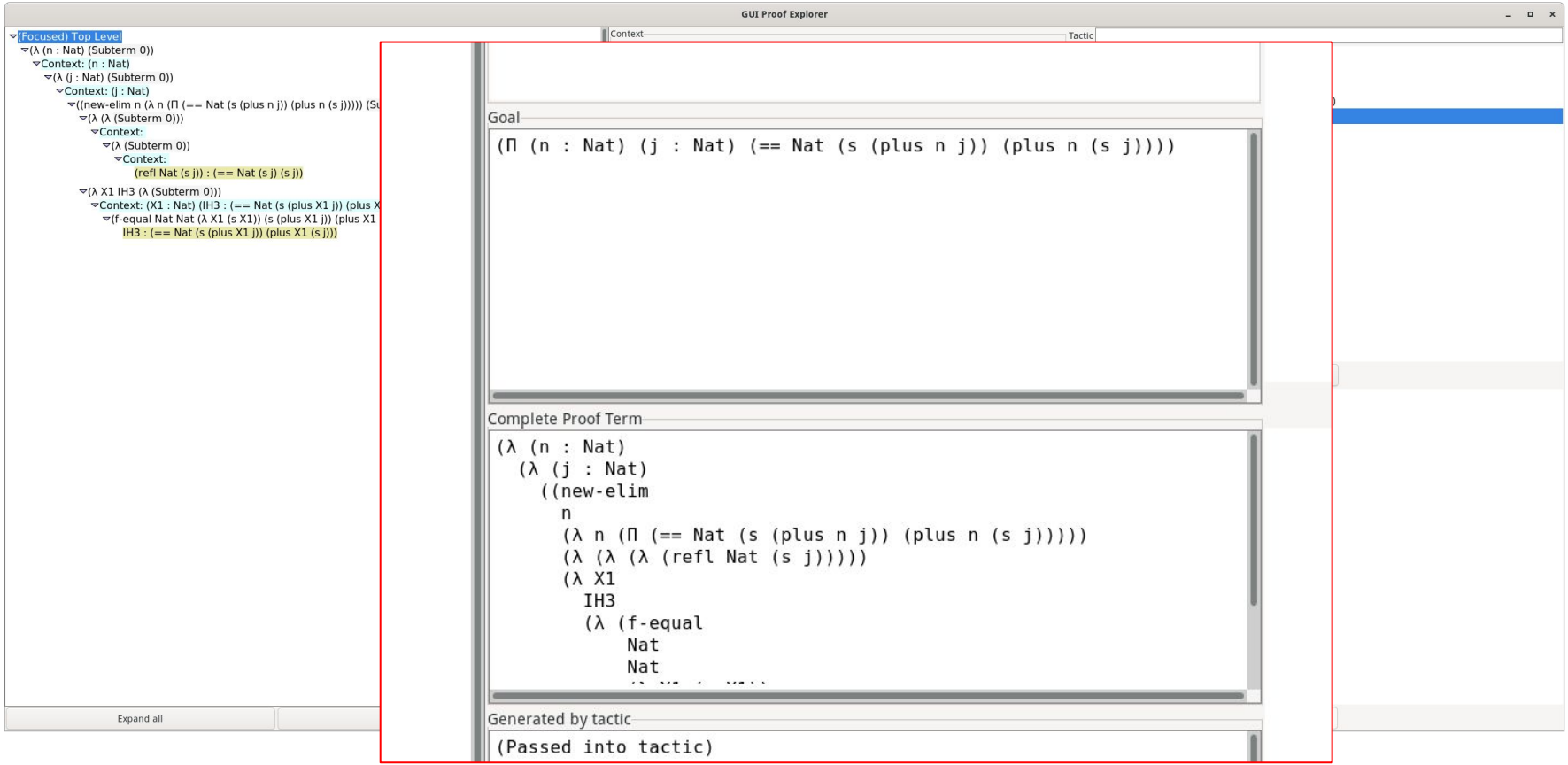

GUI Proo
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))
(λ (λ (Subterm 0)))
Context:
(λ (Subterm 0))
Context:
(refl Nat (s j)) : (== Nat (s j) (s j))
(λ X1 IH3 (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)))
Expand all
Collapse all except focus
Context
J : Nat
X1 : Nat
IH3 : (== Nat (s (plus X1 j)) (plu
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
Redo

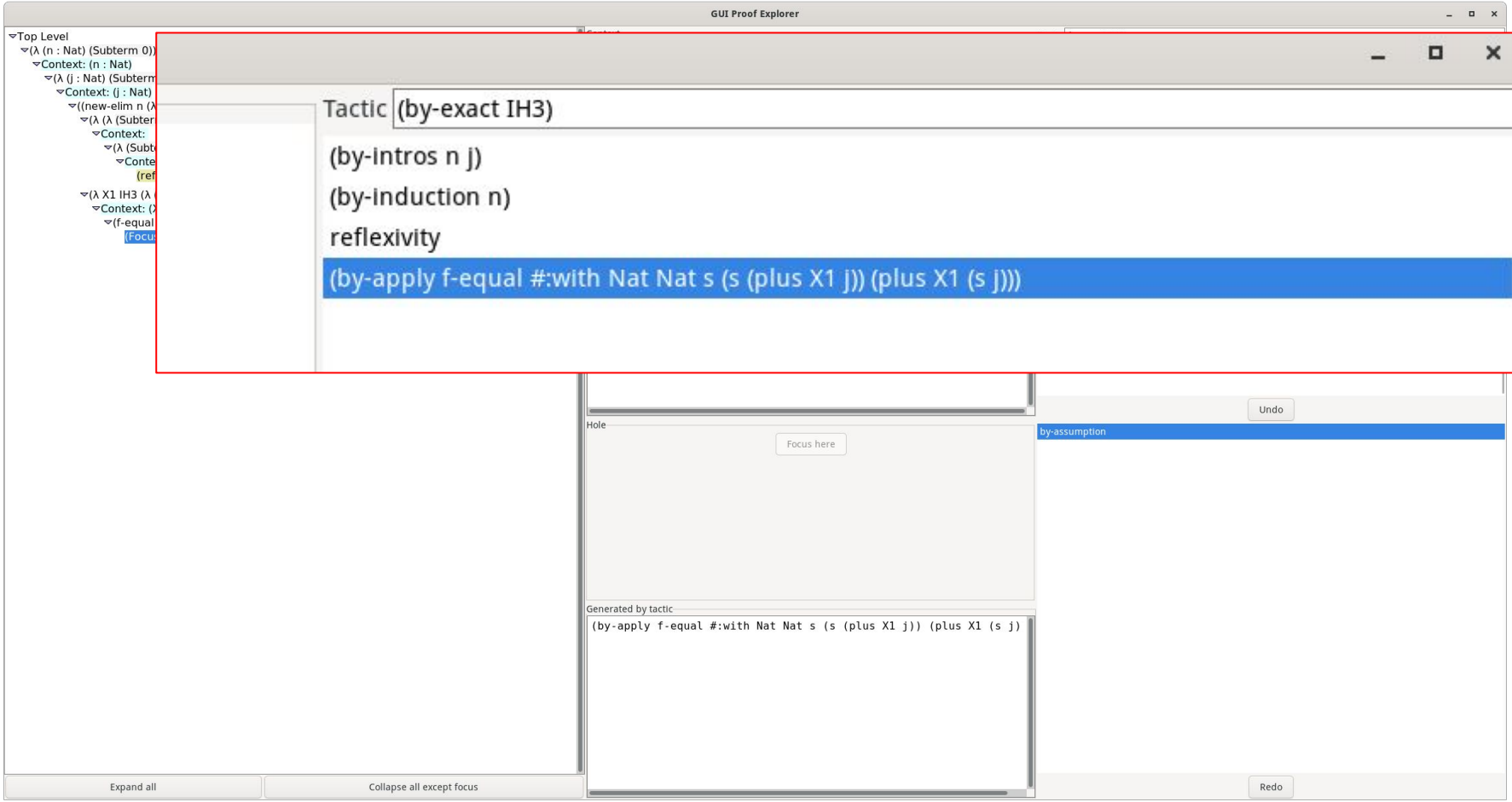

Tactic
(by-intros n j)
(s (plus X1 j)) (plus X1 (s j)))
Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))
(λ (λ (Subterm 0)))
Context:
(λ (Subterm 0))
Context:
(refl 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)))
Undo
ssumption
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


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
(f-equal Nat Nat (λ X1 (s X1)) (s (plus X1 j)) (plus X
(Focused) (== Nat (s (plus X1 j)) (plus X1 (s j))))
Context
Tactic
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

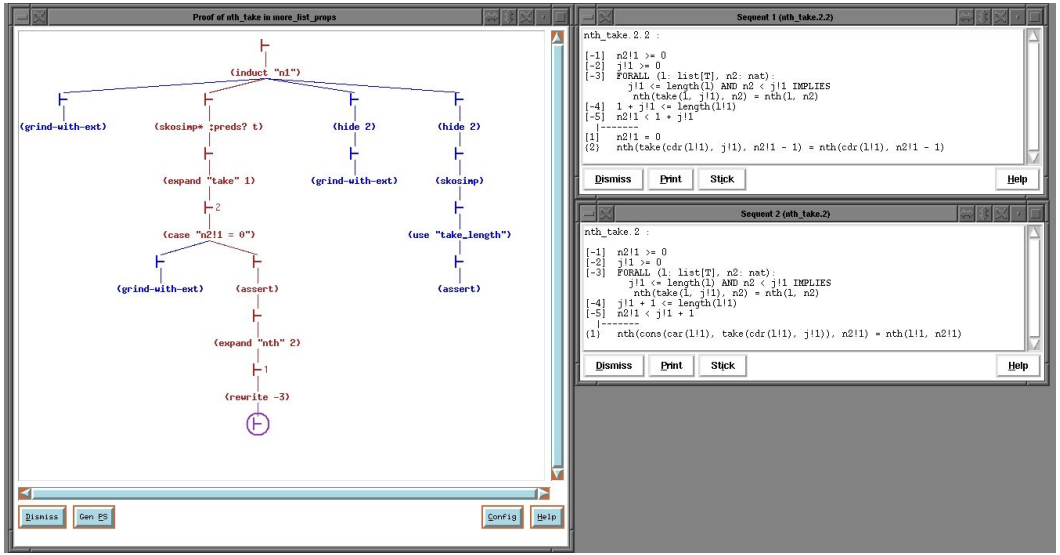

GUI Proof Explorer
(Focused) Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm
(λ (λ (Subterm 0)))
Context:
(λ (Subterm 0))
Context:
(refl Nat (s j)) : (== Nat (s j) (s 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)))
Expand all
Collapse all except focus
Context
Tactic
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
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)
Redo

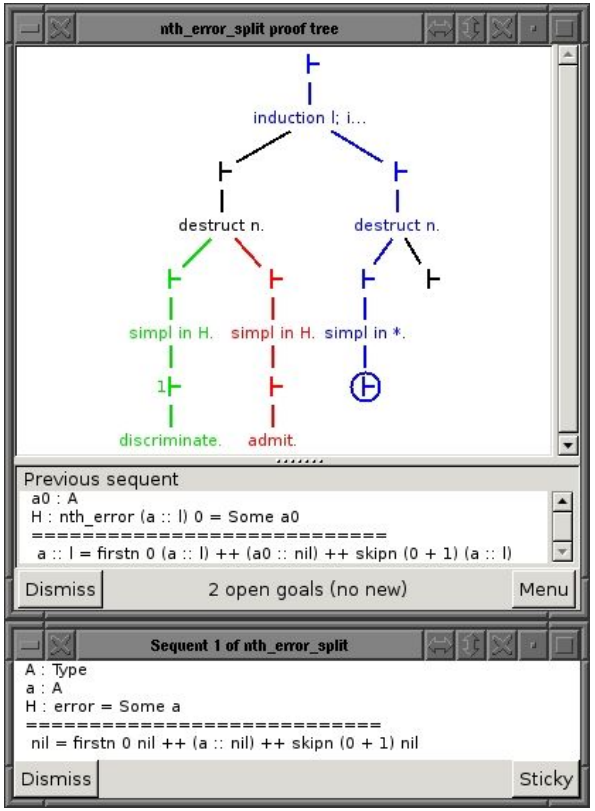

GUI Proof Explorer
(Focused) Top Level
(λ (n : Nat) (Subterm 0))
Context: (n : Nat)
(λ (j : Nat) (Subterm 0))
Context: (j : Nat)
((new-elim n (λ n (Π (== Nat (s (plus n j)) (plus n (s j))))) (Subterm 0) (Subterm 1)))
(λ (λ (Subterm 0)))
Context:
(λ (Subterm 0))
Context:
(refl Nat (s j)) : (== Nat (s 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)))
Context
Tactic
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))))) (Su
(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 X
(f-equal Nat Nat (λ X1 (s X1)) (s (plus X1 j)) (plus X1
IH3 : (== Nat (s (plus X1 j)) (plus X1 (s j)))
Expand all
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)

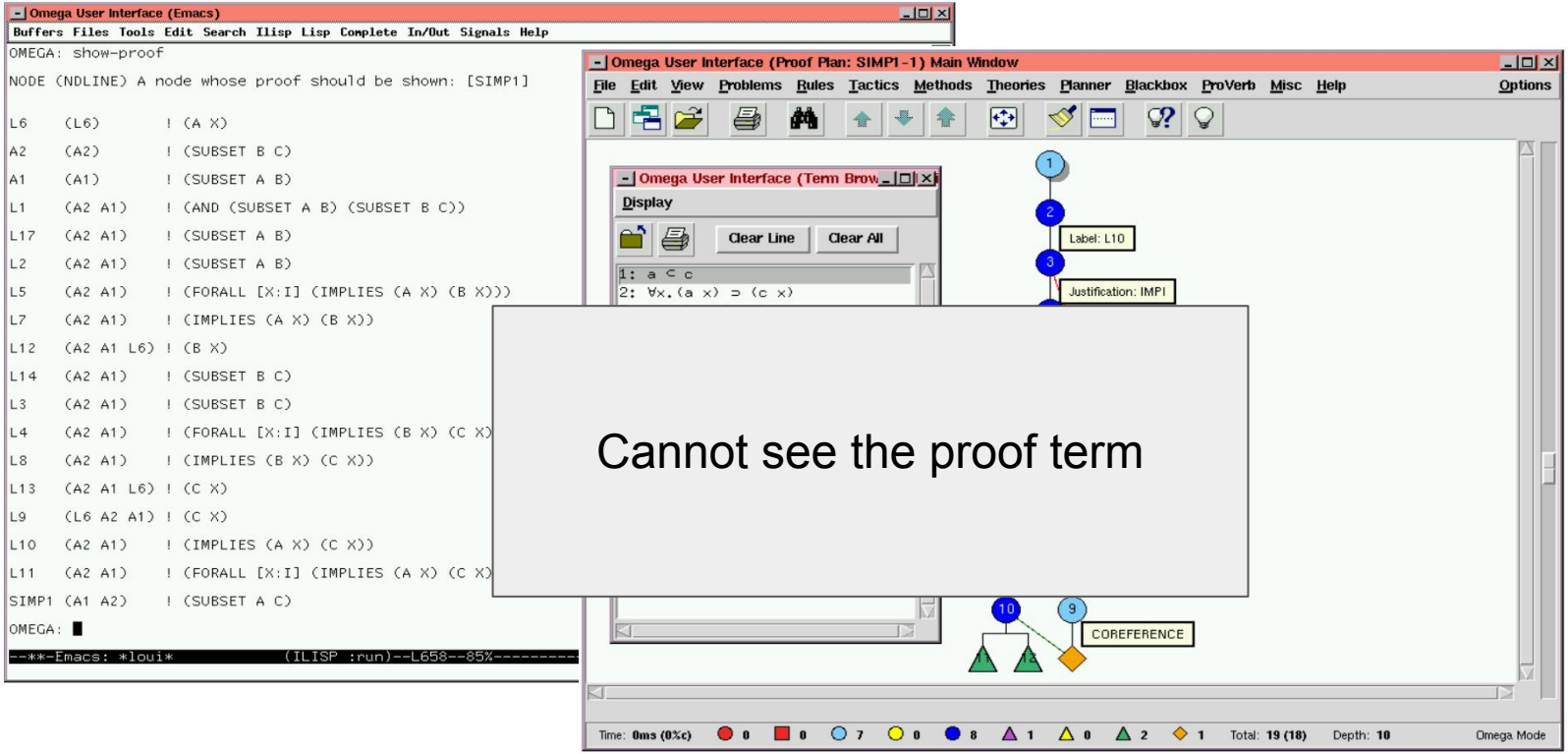

GUI Proof Explorer
Top Level
Tactic
(by-exact IH3)
(by-intros n j)
(by-induction n)
reflexivity
(by-apply f-equal #:with Nat Nat s (s (plus X1 j)) (plus X1 (s j)))
Undo
Hole
Focus here
by-assumption
Generated by tactic
(by-apply f-equal #:with Nat Nat s (s (plus X1 j)) (plus X1 (s j)
Expand all
Collapse all except focus
Redo

Related Work

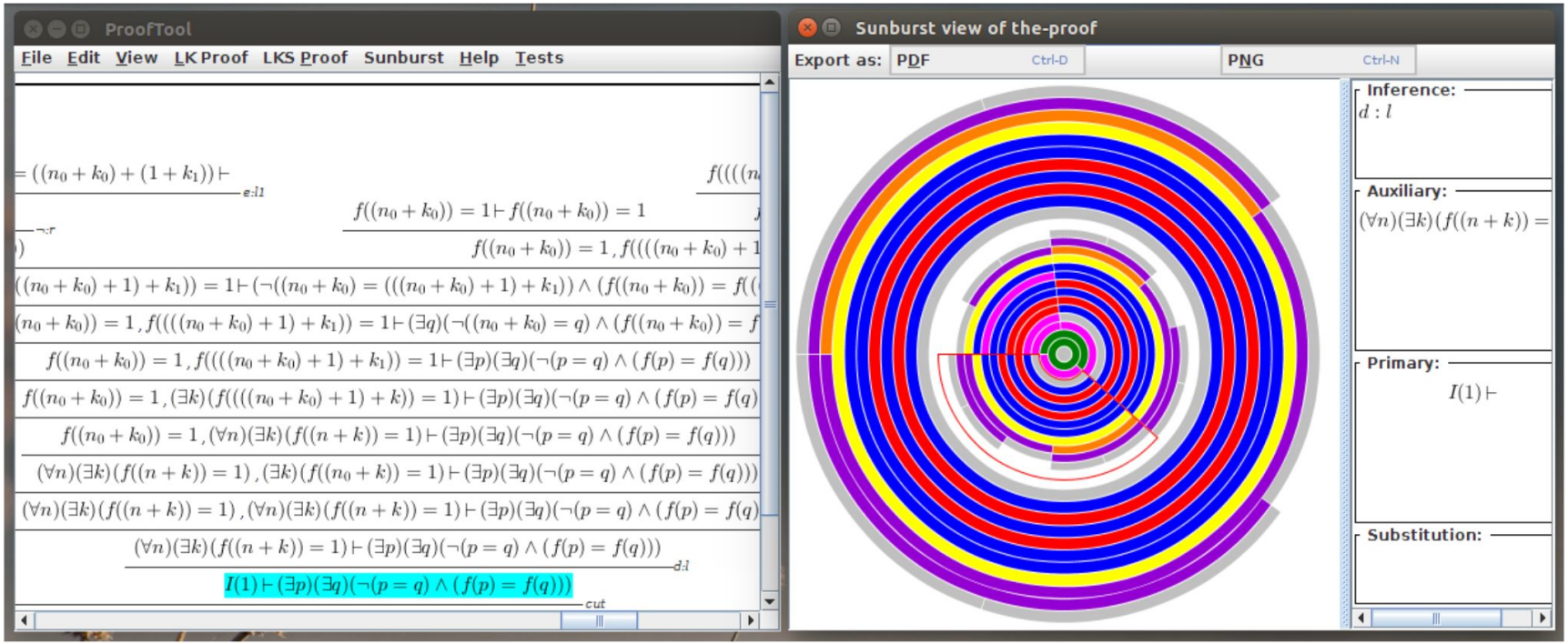




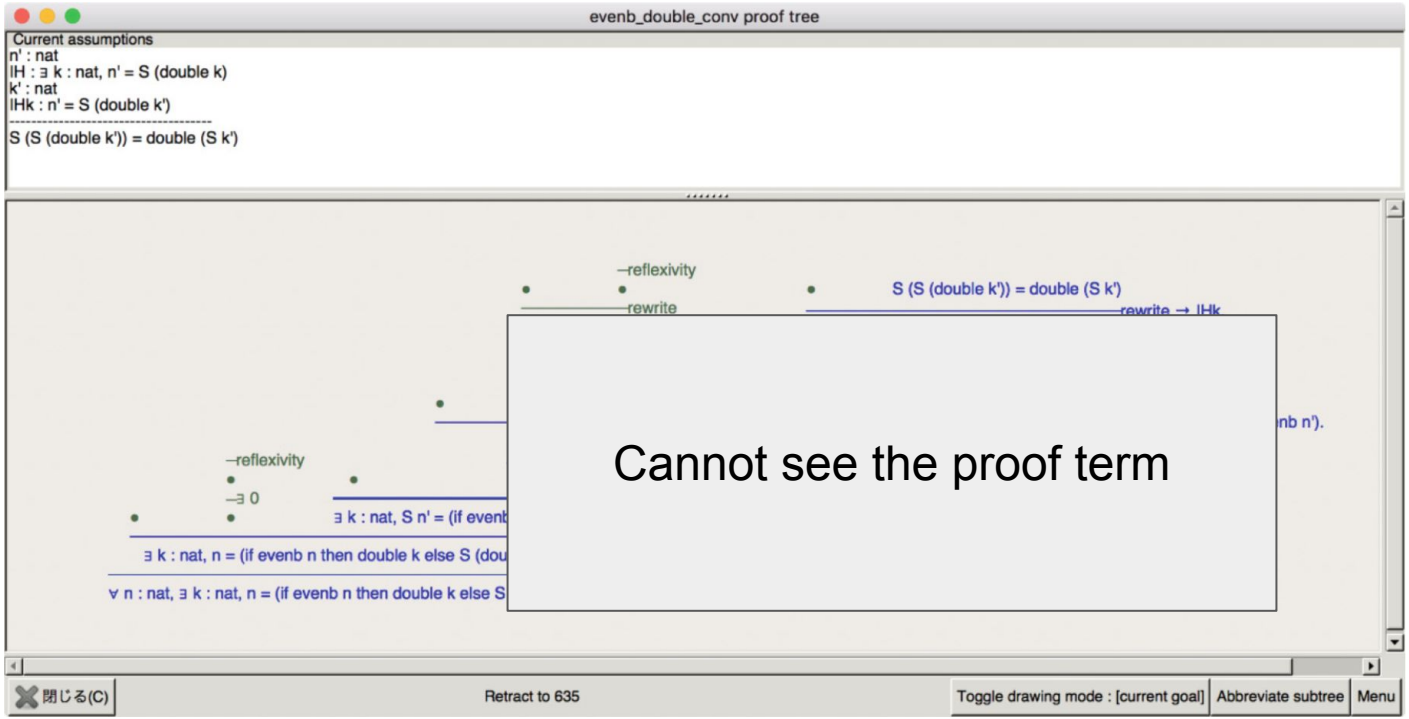
(Left) Prototype Verification System, Owre S., Rushby J.M., Shankar N. (1992)
(Right) Proof Tree Visualization for Proof General, Tews H. (2011)



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



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: ∀ n, 2 * (nsum n (fun i ⇒ 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 _ *)
```

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

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

```
    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