### Terminology

**Context**

\( x : T \)

\( S : (\text{nat} \times T) \)

\( S \times x : x \in S \)

**The bar**

\[(\forall y, y == x \rightarrow y \in S) \]

**Goal**

\[(\forall n, P n) \rightarrow G \]

\((\forall n, P n) \rightarrow G \)

- **Assumptions**
- **Conclusion**

**Top** is the first assumption, \( y \) here.

**Stack** alternative name for the list of Assumptions

---

### Popping from the stack

**Note:** in the following example we assume \( cmd \) does nothing, exactly like \( move \), to focus on the effect of the intro pattern.

**cmd** \( x \times px \)

Run \( cmd \), then pop Top, put it in the context naming it \( x \) then pop the new Top and names it \( px \) in the context.

**cmd** \( \{ x \times x \} / \)

Run \( cmd \), then reason by cases on Top. In the first branch do nothing, in the second one pop two assumptions naming then \( x \) and \( x \times x \). Then get rid of trivial goals.

Note that, since only the first branch is trivial, one can write \( \{ / \{ 1 \times x \} x \} \) too. Immediately after \( case \) and \( elim \) it does not perform any case analysis, but can still introduce different names in different branches.

**cmd** \( \{ x = x \} \)

Run \( cmd \), then apply the view \( andP \) to Top, then destruct the conjunction and introduce in the context the two parts naming \( pa \) and \( pb \).

**cmd** \( / \{ x \} \)

Run \( cmd \) then simplify the goal then discard \( px \) from then context.

**cmd** \( y \rightarrow \{ x \} \)

Run \( cmd \) then destruct the existential, then introduce \( y \), then rewrite with \( Top \) left to right and discard the equation, then clear \( x \).

---

### Cheat Sheet

**cmd** \( /\{ x \} \times b \)

Introduce \( h \) specialized to \( x \)

- **P** : \( \text{nat} \rightarrow \text{Prop} \)
- **x** : \( \text{nat} \)
- **\( x \rightarrow h : P x \)\)
- **\( (\forall n, P n) \rightarrow G \)\)

**Pushing to the stack**

**Note:** in the following \( cmd \) is not apply or **exact**. Moreover we display the goal just before \( cmd \) is run.

**cmd** \( \{ x \} y \)

Push \( x \) then push \( x \) on the stack. \( y \) is also cleared

- **x** : \( \text{nat} \)
- **\( y : \text{nat} \)\)
- **\( px : P x \)\)
- **\( \{ x \rightarrow \forall x, Q x \} \)\)

**cmd** \( \{ x \} y \)

Push the type of \( (\text{eref} x) \), then push \( x \) on the stack binding all but the second occurrence

- **x** : \( \text{nat} \)
- **\( x : \text{nat} \)\)
- **\( px : P x \)\)
- **\( \forall y, x = y \rightarrow Q x \)\)
- **\( \forall y, x = y \rightarrow Q x \)\)

**cmd** \( \{ x \} y \)

Clear \( px \) and generalize the goal with respect to the first match of the pattern \( \{ x \} y \)

- **x** : \( \text{nat} \)
- **\( x : \text{nat} \)\)
- **\( px : P x \)\)
- **\( \forall x, x < x + 1 \)\)

**cmd** \( \{ x \} y \)

Clear the goal, then generalize the goal with respect to the next match of the pattern \( \{ x \} y \)

**cmd** \( \{ x \} y \)

Open a new goal for \( P a \)

**cmd** \( \{ x \} y \)

Rewrite with \( Eab \) left to right, then with \( Exc \) by instantiating the first argument with \( b \)

- **Eab** : \( a = b \)
- **Exc** : \( (\forall x, x = c) \rightarrow \) \( \forall x, x = c \)
- **P a**
- **P c**

**cmd** \( \{ x \} y \)

Rewrite with \( Eab \) right to left then with \( Exc \) left to right, finally clear \( Eac \)

- **Eab** : \( a = b \)
- **Eac** : \( a = c \)
- **P b**

**cmd** \( \{ x \} y \)

Fold back the local definition \( \& \& \)

- **a** : \( \text{bool} \)
- **\( a \rightarrow \)\)
- **\{ if a then a else false = a \} \)
Reflect and views
reflect P b
States that P is logically equivalent to b
apply: (iffP V)
Proves a reflection goal, applying the view lemma V to the propositional form of P.

Misc notations
"f1 \ o f2" := (comp f1 f2)
"x \ \in\ A" := (\in (\text{mem} x (\text{mem} A))

Misc
"f1 \ o f2" := (comp f1 f2)
"x \ \in\ A" := (\in (\text{mem} x (\text{mem} A))

Notations for natural numbers: nat

Notations for lists: seq T

Rewrite patterns
rewrite [part1][lem in pat2][lem2] X
Rewrite the subterms selected by the pattern inferred from lem1 and Lem2 to identify the sub terms of X to be rewritten. Of these terms, rewrite only the third one.

Example: rewrite [3](\text{In} x in f \ \text{X}) E.

Rewrite patterns
rewrite [part1][lem in pat2][lem2] X
Rewrite the subterms selected by the pattern inferred from lem1 and Lem2 to identify the sub terms of X to be rewritten. Of these terms, rewrite only the third one.

Example: rewrite [3](\text{In} x in f \ \text{X}) E.