# Basic Proof Methods

If **thmA:** $P \Longrightarrow Q$, **thmEq:** $P \Longrightarrow t = s$ and proof obligation is $R_1\, t \Longrightarrow R_2\, t$ then

**apply(**frule thmA**)** forward reasoning: if $P$ matches $R_1\, t$, add $Q$ as assumption and new proof obligation is $P \Longrightarrow Q \Longrightarrow R_2\, t$

**apply(**rule thmA**)** backward reasoning: if $Q$ matches $R_2\, t$, eliminate $R_2\, t$ and new proof obligation is $R_1\, t \Longrightarrow P$

**apply(**drule thmA**)** if $P$ matches $R_1\, t$, eliminate $R_1\, t$, add $Q$ as assumption, and new proof obligation is $Q \Longrightarrow R_2\, t$

**apply(**erule thmA**)** if $Q$ matches $R_2\, t$, and $P$ matches $R_1\, t$, eliminate $R_2\, t$ as obligation and $R_1\, t$ as a hypothesis

**apply(**subst thmEq**)** rewrite $s$ for $t$ everywhere in $R_2\, t$ and add $P$ as an obligation $\begin{cases} 1. & R_1\, t \Longrightarrow P \\ 2. & R_1\, t \Longrightarrow R_2\, s \end{cases}$

**apply(**subst (asm) thmEq**)** rewrite $s$ for $t$ everywhere in the first assumption and add $P$ as an obligation

**apply(**method args**)+** repeat the method as long as possible at least one time

**apply(**method args**)?** try the method (never fails)

**apply(**method1 args, method2 args**)** sequential composition: do method2 after method1

**apply(**method1 args; method2 args**)** structural composition: do method2 to all subgoals emerging after applying method1

**apply(**method1 args; method2 args**)** do method1, if that fails, do method 2

# Automated Proof Methods

| Method | Description heuristic | Calls Simplifier | Goals |
|---|---|---|---|
| **intro** | Recursively apply in backward reasoning its arguments, e.g. apply(rule …)+, until it is no longer possible. | ✗ | 1 |
| **elim** | Recursively do elimination to its arguments, e.g. apply(erule …)+, until it is no longer possible. | ✗ | 1 |
| **simp** | Recursively rewrite the proof obligation according to its arguments as rewrite rules until it is no longer possible. | ✓ | 1 |
| **simp_all** | Apply the simplifier to all subgoals | ✓ | All |
| **clarify** | Safe first-order logic (FOL) rules that do not augment the number of goals. | ✗ | 1 |
| **clarsimp** | clarify + simplifier | ✓ | 1 |
| **safe** | Safe FOL rules including those that augment the number of goals. | ✗ | All |
| **fast** | weak FOL solver | ✗ | 1 |
| **blast** | FOL solver | ✗ | 1 |
| **force** | simp + blast + more | ✓ | 1 |
| **fastforce** | simp + fast + more | ✓ | 1 |
| **auto** | simp + safe + more | ✓ | All |

use indiscriminately (as unsafe)

add to simplifier — add to drule — do not use

**apply(**method *simp add*: T1 … T(n) *intro!*: T(n+1) … T(m) *dest*: T(m+1) … T(l) *elim!*: T(l+1) … T(k) *simp del*: T(k+1) … T(j)**)**

alternatively: *simp only*: T1 … T(n)

add to intro rules — add to erule