Interactive Computer Theorem Proving

Lecture 6: A Crash Course on Proof Automation

CS294-9
September 28, 2006
Adam Chlipala
UC Berkeley

Projects!

- Decide to work individually or in groups of up to 2.
  - People only taking the class for 1 unit are welcome to participate in a less structured way....
- Choose a subject related to the theme of “interactive computer theorem proving.”
  - You don't need to use Coq, but I can help you best if you do.

Possible Topics

- Formalize an interesting problem from your research.
  - A programming language and its properties?
  - An algorithm and why it's correct?
- For the more adventurous: Implement tool support for reasoning in a domain that interests you.
  - Warning: Coq is architected very cleanly, but there is almost no documentation on extending it!

Timeline

- **10/12**: Submit to me by e-mail a project proposal.
  - [Weekly homework assignments stop at this point.]
  - List your group members.
  - Describe the overall problem.
  - Break it up into a number of possible formalization tasks.
  - I'll suggest which of these should be doable in the time frame we have.

Timeline

- **11/30 and 12/4** (and possibly 11/23, depending on the number of groups): Final project presentations
  - You get half a class period to tell us about your project.
  - Expect that a good amount of the time will be taken up with questions.
- **~12/11**: Short project report
  - Probably with some code

Proof Terms vs. Tactics

Proof Terms	**Tactics**
Mathematically elegant	Heuristically expedient
Statically typed	Dynamically typed
Small number of primitives	Everything but the kitchen sink
Brittle	Adapt to change (if used right)
Really a programming language	Really a programming language

Motivating Example: Type Safety

$\underline{n}$ ::= O | S $\underline{n}$

$\underline{b}$ ::= true | false

x ::= [variable]

$\underline{e}$ ::= $\underline{n}$ | $\underline{b}$ | x | $\underline{e}$ + $\underline{e}$ | $\underline{e}$ = $\underline{e}$

$$\frac{}{\textbf{value}(\underline{n})} \qquad \frac{}{\textbf{value}(\underline{b})}$$

$$\frac{}{x \Rrightarrow \sigma(x)} \qquad \frac{}{\underline{n}_1 + \underline{n}_2 \Rrightarrow n_1 \mathbf{+} n_2} \qquad \frac{}{\underline{n}_1 = \underline{n}_2 \Rrightarrow n_1 \mathbf{=} n_2}$$

$$\frac{\underline{e}_1 \Rrightarrow \underline{e}_1'}{\underline{e}_1 + \underline{e}_2 \Rrightarrow \underline{e}_1' + \underline{e}_2} \qquad \frac{\underline{e}_1 \Rrightarrow \underline{e}_1'}{\underline{e}_1 = \underline{e}_2 \Rrightarrow \underline{e}_1' = \underline{e}_2}$$

$$\frac{\textbf{value}(\underline{e}_1) \quad \underline{e}_2 \Rrightarrow \underline{e}_2'}{\underline{e}_1 + \underline{e}_2 \Rrightarrow \underline{e}_1 + \underline{e}_2'} \qquad \frac{\textbf{value}(\underline{e}_1) \quad \underline{e}_2 \Rrightarrow \underline{e}_2'}{\underline{e}_1 = \underline{e}_2 \Rrightarrow \underline{e}_1 = \underline{e}_2'}$$

Type System

$$\frac{}{\Gamma \vdash \underline{n} : \text{nat}} \qquad \frac{}{\Gamma \vdash \underline{b} : \text{bool}} \qquad \frac{}{\Gamma \vdash x : \Gamma(x)}$$

$$\frac{\Gamma \vdash \underline{e}_1 : \text{nat} \qquad \Gamma \vdash \underline{e}_2 : \text{nat}}{\Gamma \vdash \underline{e}_1 + \underline{e}_2 : \text{nat}}$$

$$\frac{\Gamma \vdash \underline{e}_1 : \text{nat} \qquad \Gamma \vdash \underline{e}_2 : \text{nat}}{\Gamma \vdash \underline{e}_1 = \underline{e}_2 : \text{bool}}$$

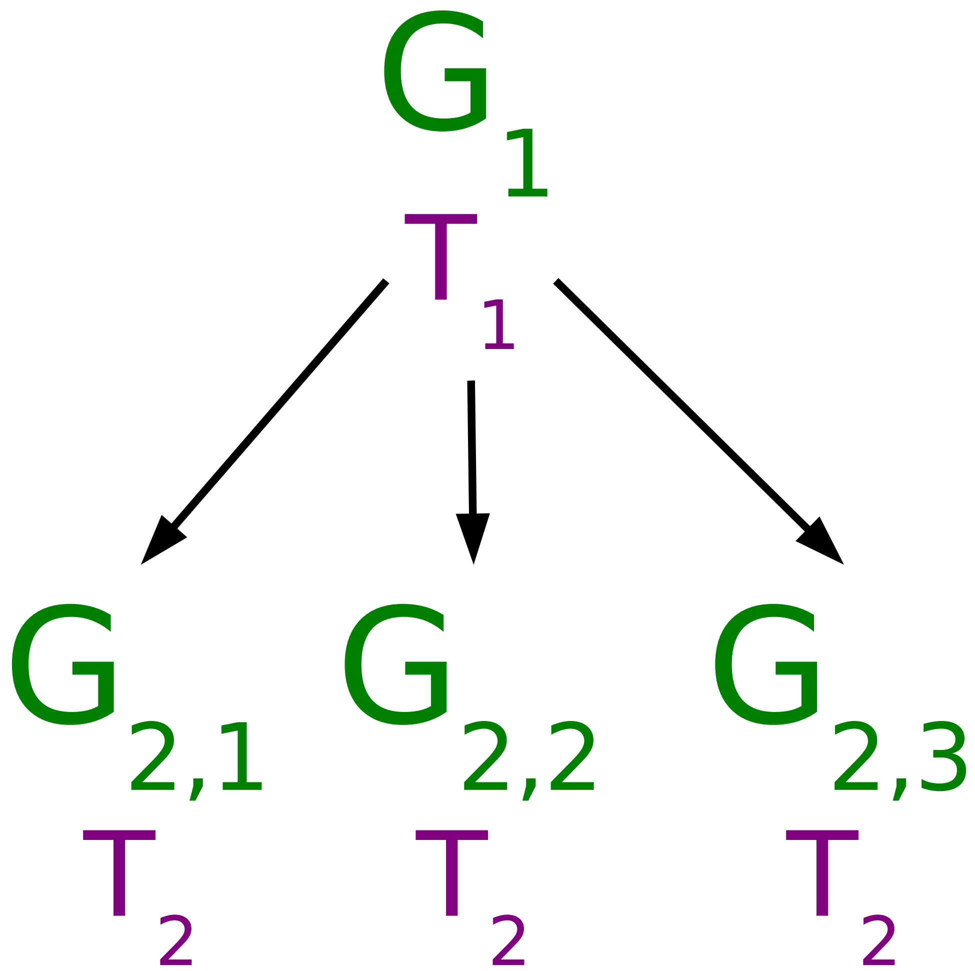
Type safety

Progress: If $\Gamma \vdash \underline{e} : \tau$, then either **value**($\underline{e}$) or $\underline{e} \Rightarrow \underline{e'}$ for some $\underline{e'}$.

Preservation: If $\Gamma \vdash \underline{e} : \tau$ and $\underline{e} \Rightarrow \underline{e'}$, then $\Gamma \vdash \underline{e'} : \tau$.

Sequencing

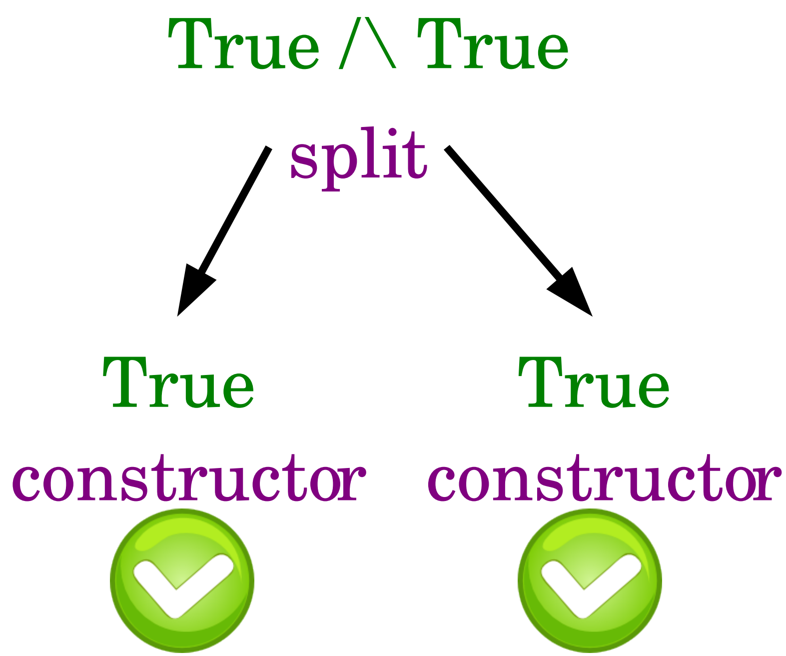
$T_1; T_2$



Example

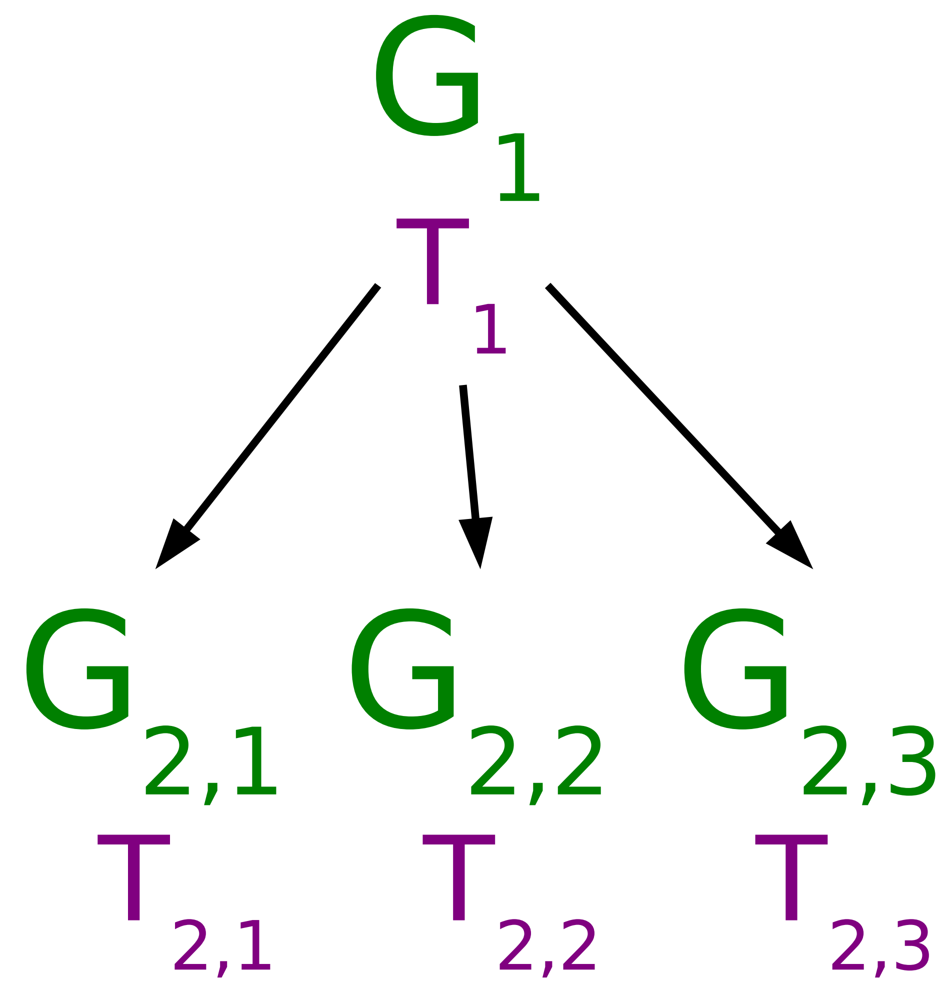
Goal: True /\ True

Tactic: split; constructor.



Non-uniform Sequencing

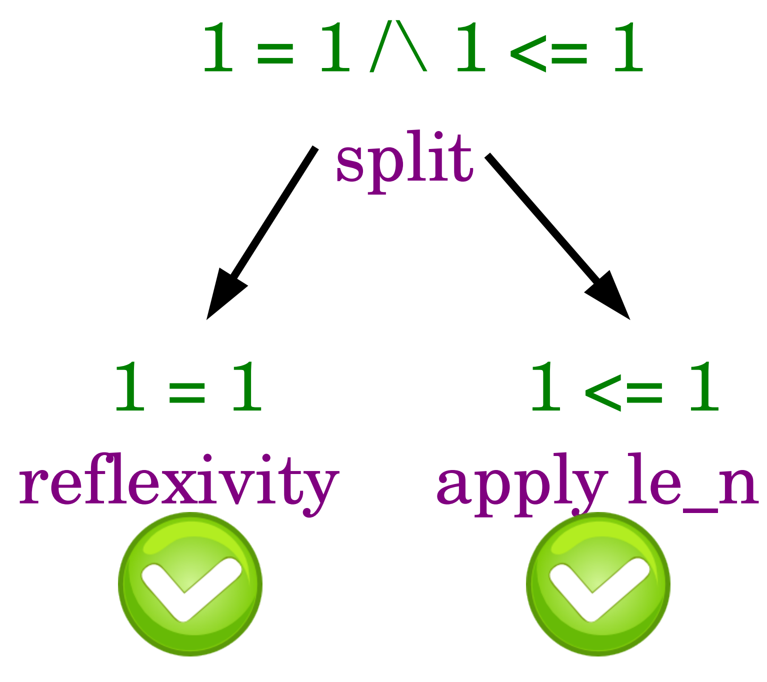
$T_1; [T_{2,1} \mid T_{2,2} \mid T_{2,3}]$



Example

Goal: 1 = 1 /\ 1 <= 1

Tactic: split; [reflexivity
| apply le_n].



Twiddling Your Thumbs

idtac

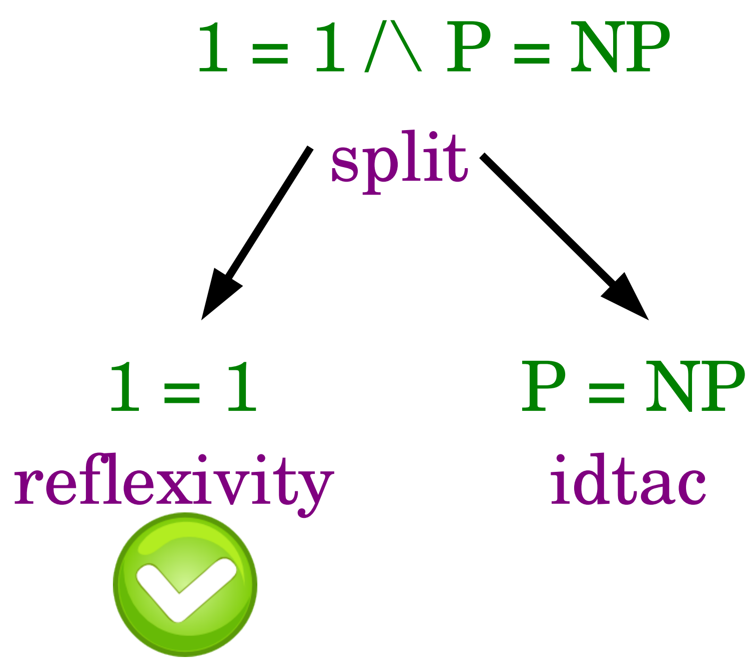
G

idtac

Example

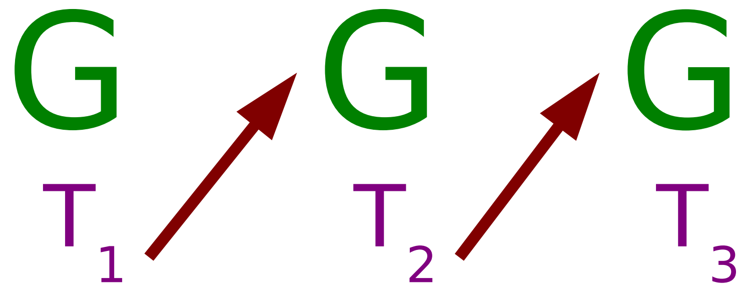
Goal: 1 = 1 /\ P = NP

Tactic: split; [reflexivity | idtac].



Alternatives

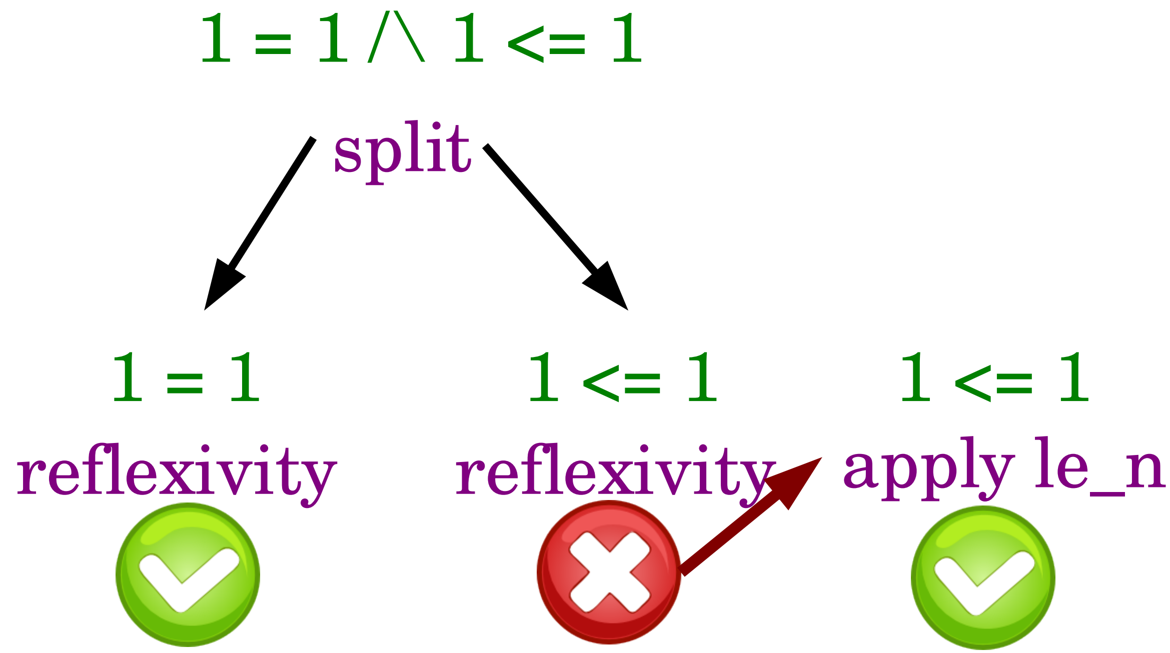
$T_1 \,||\, T_2 \,||\, T_3$



Example

Goal: 1 = 1 /\ 1 <= 1

Tactic: split; (reflexivity
|| apply le_n).

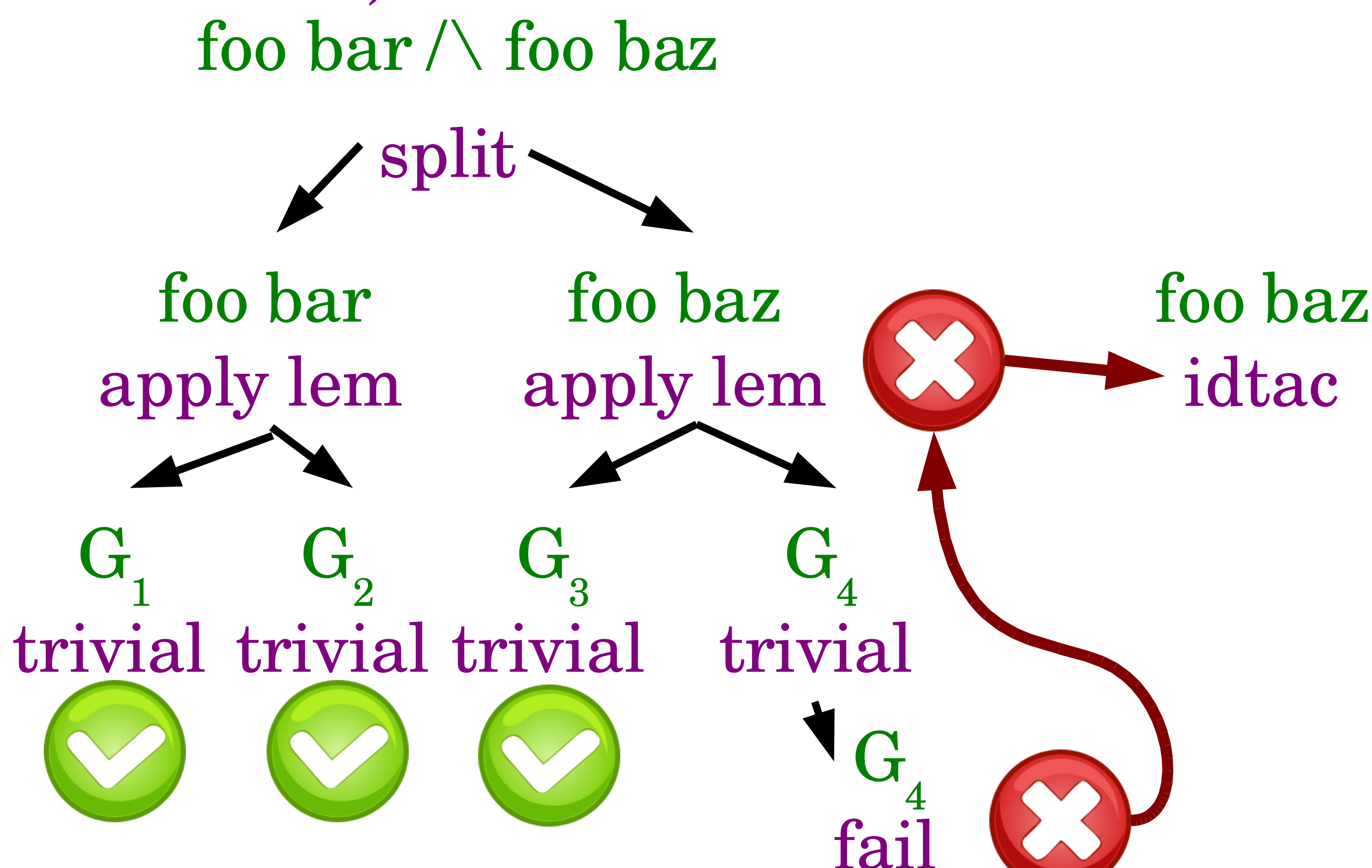


Explicit Failure

Example

Goal: foo bar /\ foo baz

Tactic: split; ((apply lem; trivial; fail)
|| idtac).



Graceful Failure

try T

Example

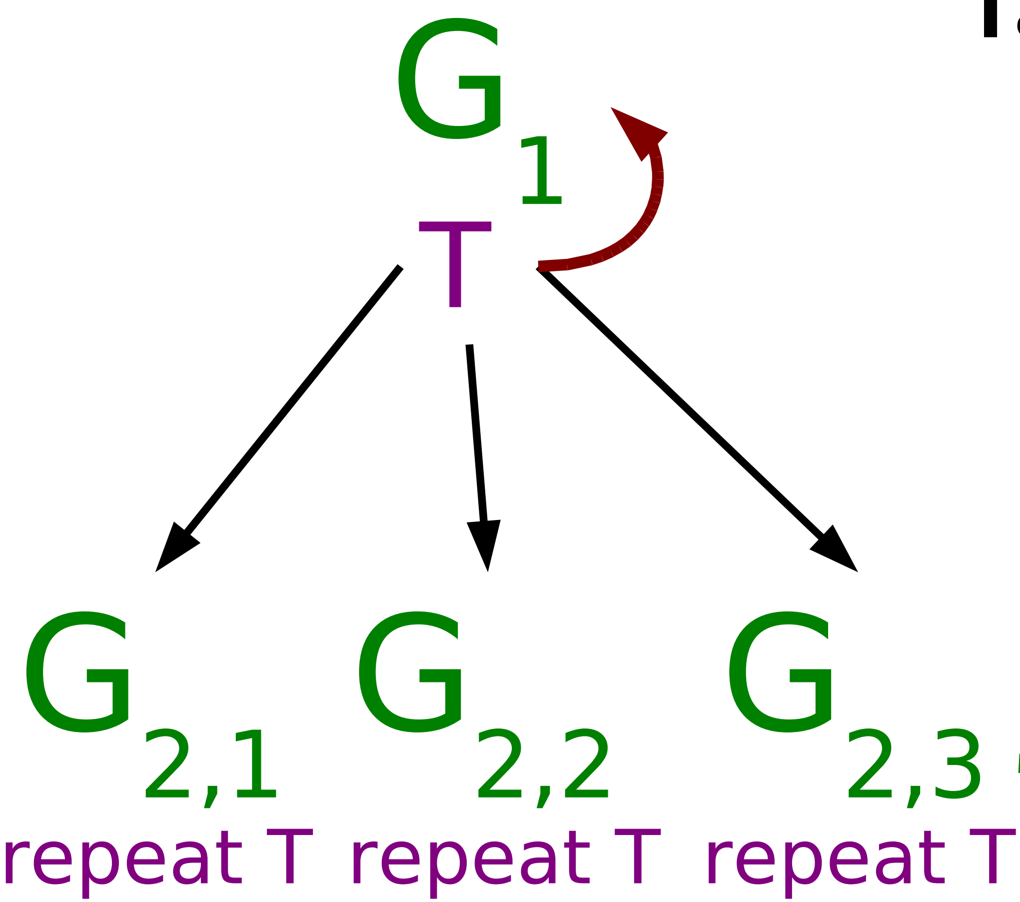
Goal: foo bar /\ foo baz

Tactic: split; try (apply lem; trivial; fail).

try T
is the same as
T || idtac

Repetition

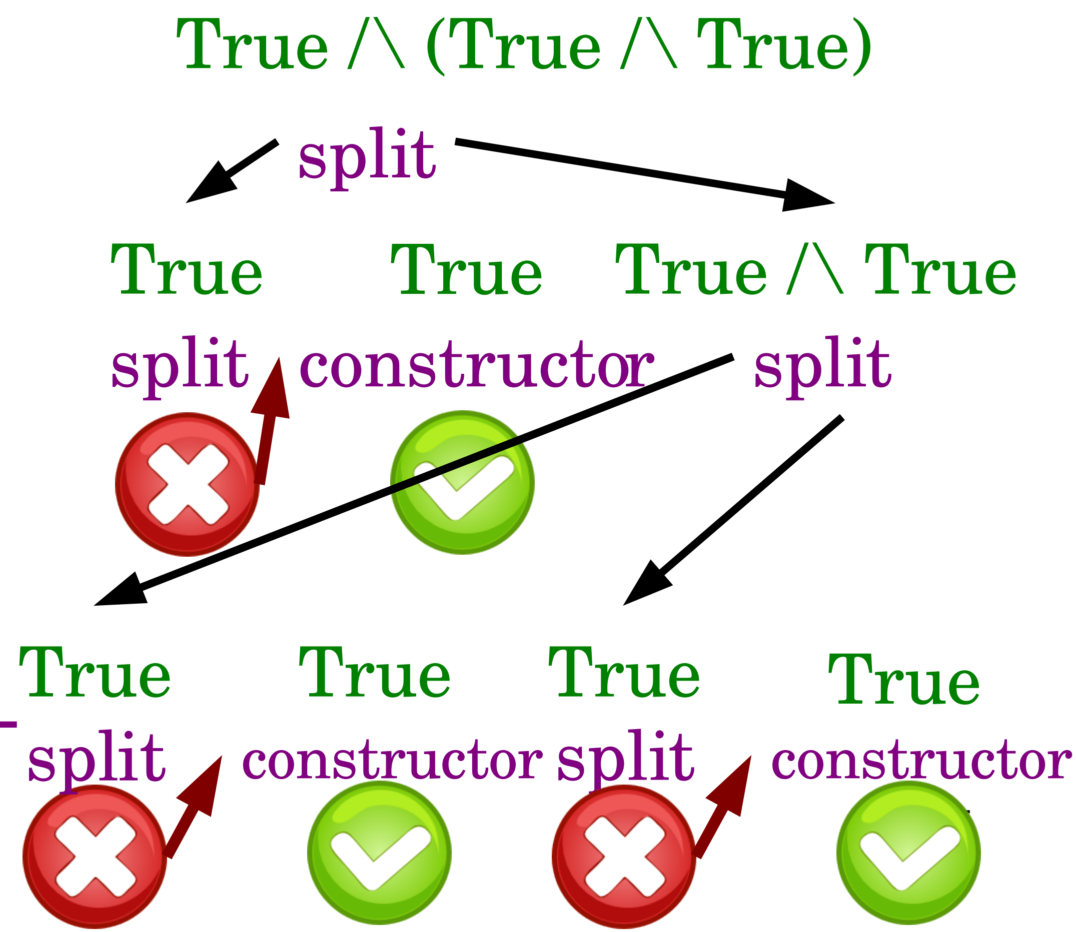
repeat T



Example

Goal: True /\ (True /\ True)

Tactic: repeat split; constructor.



Pattern Matching

H1 : x = y

H2 : y = S z

=============

x + y = q

tac1{H := H1, A := x, B := y}

tac1{H := H2, A := y, B := S z}

tac2{H := H2, A := y, B := z}

tac3{A := x + y, B := q}

```
match goal with
  | [ H : ?A = ?B |- _ ] => tac1
  | [ H : ?A = S ?B |- _ ] => tac2
  | [ |- ?A = ?B ] => tac3
end.
```