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**
- Mathematically elegant
- Statically typed
- Small number of primitives
- Brittle
- Really a programming language

**Tactics**
- Heuristically expedient
- Dynamically typed
- Everything but the kitchen sink
- Adapt to change (if used right)
- Really a programming language
Motivating Example: Type Safety

\[ n ::= O \mid S \, n \]
\[ b ::= true \mid false \]
\[ x ::= [\text{variable}] \]
\[ e ::= n \mid b \mid x \mid e + e \mid e = e \]

\[ \text{value}(n) \]
\[ \text{value}(b) \]

\[ x \Rightarrow \sigma(x) \]
\[ n_1 + n_2 \Rightarrow n_1 + n_2 \]
\[ e_1 \Rightarrow e_1' \]
\[ e_1 + e_2 \Rightarrow e_1' + e_2 \]
\[ \text{value}(e_1) \quad e_2 \Rightarrow e_2' \]
\[ e_1 + e_2 \Rightarrow e_1 + e_2' \]
\[ n_1 = n_2 \Rightarrow n_1 = n_2 \]
\[ e_1 \Rightarrow e_1' \]
\[ e_1 = e_2 \Rightarrow e_1' = e_2 \]
\[ \text{value}(e_1) \quad e_2 \Rightarrow e_2' \]
\[ e_1 = e_2 \Rightarrow e_1 = e_2' \]
Type System

\[
\begin{align*}
\Gamma & \vdash n : \text{nat} & \Gamma & \vdash b : \text{bool} & \Gamma & \vdash x : \Gamma(x) \\
\Gamma & \vdash e_1 : \text{nat} & \Gamma & \vdash e_2 : \text{nat} & \Gamma & \vdash e_1 + e_2 : \text{nat} \\
\Gamma & \vdash e_1 : \text{nat} & \Gamma & \vdash e_2 : \text{nat} & \Gamma & \vdash e_1 = e_2 : \text{bool}
\end{align*}
\]

Type safety

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

**Preservation**: If \( \Gamma \vdash e : \tau \) and \( e \Rightarrow e' \), then \( \Gamma \vdash e' : \tau \).
Sequencing

Example

Goal: $\text{True} \land \text{True}$

Tactic: split; constructor.

True $\land$ True

split

True

constructor

True

constructor
Non-uniform Sequencing

Example

Goal: $1 = 1 \land 1 \leq 1$

Tactic: split; [reflexivity | apply le_n].

$$1 = 1 \land 1 \leq 1$$

split

$$1 = 1$$ reflexivity

$$1 \leq 1$$ apply le_n

$$\checkmark$$

$$\checkmark$$
Example

Goal: \( 1 = 1 \land P = NP \)

Tactic: split; [reflexivity | idtac].

\[
\begin{align*}
1 = 1 \land P = NP \\
\text{split} \\
1 = 1 & \text{reflexivity} \\
P = NP & \text{idtac}
\end{align*}
\]
Alternatives

Example

Goal: $1 = 1 \land 1 \leq 1$

Tactic: \texttt{split; (reflexivity || apply le_n)}.

$1 = 1 \land 1 \leq 1$

\texttt{split}

\texttt{reflexivity} \quad \texttt{reflexivity} \quad \texttt{apply le_n}
Explicit Failure

Example

Goal: foo bar /
\ foo baz

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

| idtac).

foo bar /
\ foo baz

split

foo bar

apply lem

G_1

trivial


foo baz

apply lem

G_2

trivial


G_3

trivial


G_4

trivial

fail

G_4

idtac

foo baz
Graceful Failure

Example

Goal: foo bar \(\land\) foo baz

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

try T

G

T

try T

is the same as

T || id tac
Repetition

Example

Goal: \( \text{True} \land (\text{True} \land \text{True}) \)

Tactic: \text{repeat split; constructor.}

\[
\begin{array}{c}
\text{True} \\
\text{split} \\
\text{True} \land \text{True} \\
\text{split} \\
\text{constructor} \\
\text{split}
\end{array}
\]
Pattern Matching

H1 : \( x = y \)
H2 : \( y = S \, z \)

\[ x + y = q \]

match goal with

\[ \begin{align*}
| \ [ \ H : \ ?A = \ ?B \ | \ - \ _ \ ] & \Rightarrow \ tac1 \\
| \ [ \ H : \ ?A = S \ ?B \ | \ - \ _ \ ] & \Rightarrow \ tac2 \\
| \ [ \ | \ - \ ?A = \ ?B \ ] & \Rightarrow \ tac3 \\
\end{align*} \]

end.