Coq as a Tutor in Formal Reasoning

Guiding through the proof

Students' Problems

- Confusing assumptions and conclusions.
  - „Now we prove that f is injective. A function f is injective if for all x, y we have f (x) = f (y) implies x = y. Let us consider arbitrary x, y such that f (x) = f (y). Since this implies x = y, the function f is injective.”
- Quantifier problems.
  - Translating „if there exist three elements in the set A, two of them must be equal” into formal language.
- „Forgetting to unfold definition” problems.
- Bad understanding of definitions.
  - Show $\phi : P(N) \to (P(N) \to P(N))$ given by $\phi(a)(b) = a \cap b$ is injective: “Let a and a' be subsets of N. Let us consider ϕ(a) and show that it is an injective function.”
- Negating logical sentences.

Naive Type Theory better than „sets everywhere”

- <*a*, *a*> = {{*a*}}? Not teaching how to encode things as sets (implementation) not considered a problem.
- Basic types: **Prop**, **empty**, **unit**, **bool**, **nat**, and type constructors: **->** (function), **+** (sum), ***** (Cartesian product)
- Predicates *Predicate T := T -> Prop*.
  - Sum, difference, intersection, inclusion of predicates.
- Functions: image, injectivity, surjectivity, etc.
  - Problem with distinction of types and predicates=subsets.
    - Checking properties limited to predicates no problem.
    - Problem with equality for functions „defined” on predicates.
- Binary predicates *Relation (A : Type) := A -> A -> Prop*: reflexivity, transitivity, symmetry, etc.

Elements of Coq

- Calculus of Inductive Constructions: a dependently typed lambda-calculus with a structural recursion operator over inductive types, plus global and local definitions.
- Vernacular: language of commands manipulating CIC, generating inductive schemes etc.
- Inside Vernacular, language of tactics and tacticals to interactively write proof scripts.
- Ltac: language for defining tactics.
- Sessions allow to use variables which are then generalized „en bloc”.
- Arguments marked as implicit are (type-) inferred.
- Existential meta-variables for subterms to be instantiated later in the proof (e.g. *eapply ex_intro*).

Naive Type Theory and Coq

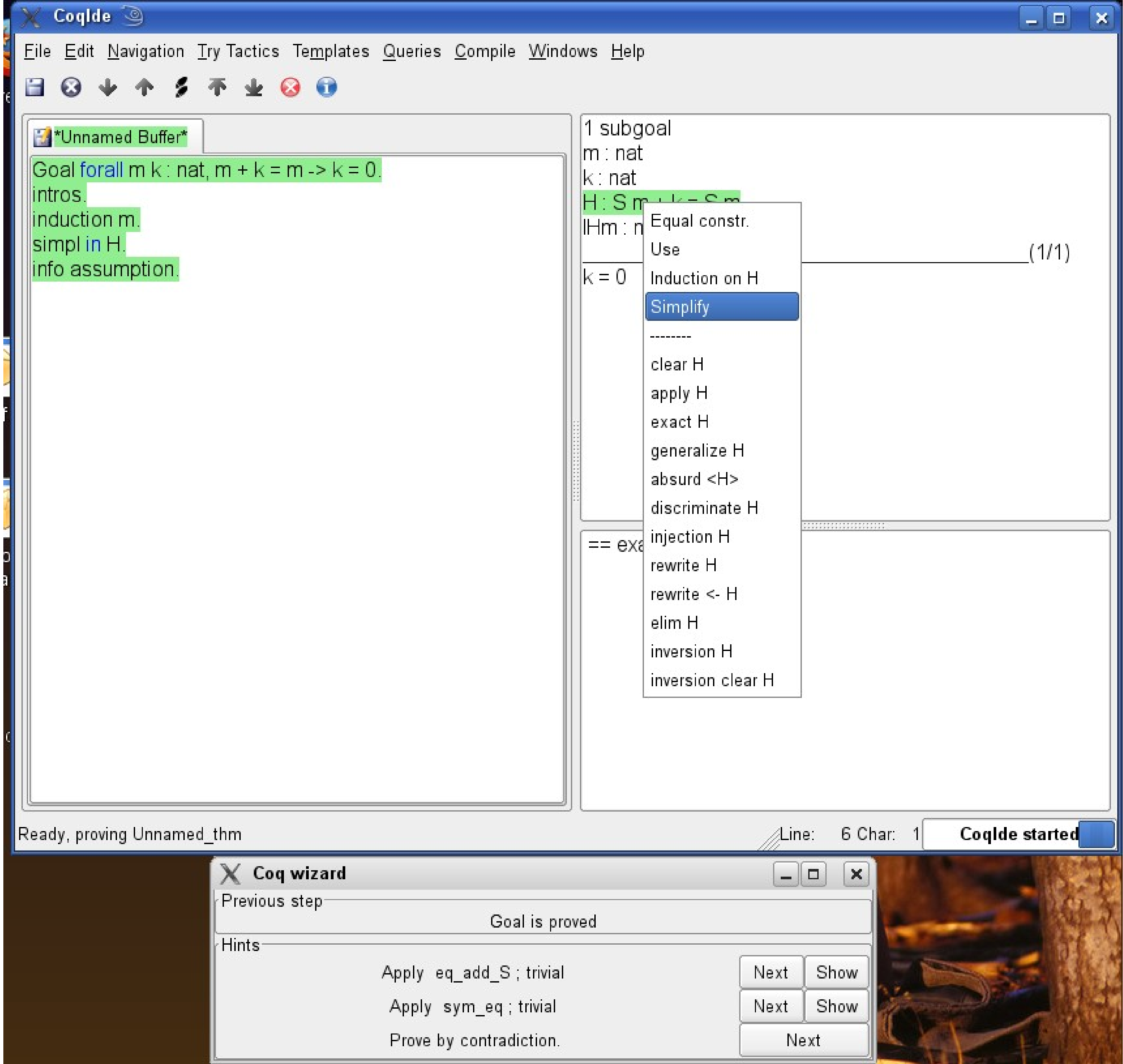
- Only four axioms have to be added: excluded middle, extensionality of predicate equality, extensionality of functional equality and the principle of description.
  - There exists a function corresponding to a functional relation.
- „IN” is defined as application, the „universe” is an implicit argument.
  - Sets are subsets of some type.
- Notations, e.g.
  - *Notation "A 'n' B" := (Intersection A B) (at level 11).*
  - *Notation "A 'c' B" := (Subset A B) (at level 100).*

CoqIde + Papug

- CoqIde allows to work with multiple files,
- marks and protects processed fragment of the script,
- separate windows for proof state and for inspection/search results,
- menus with inspection commands, tactics and script command templates.

- Papuq adds a help window proposing simple actions,
- lets inspect the lemmas it proposed to apply,
- extends the context menu of CoqIde showing most worthwhile actions first,
- „wizard” automation tries out several tactics.

CoqIde + Papug: example



Digression: Proof-by-Pointing

- Not present (yet) in CoqIde, but present in earlier Coq interfaces (IDEs): CtCoq (mid-nineties) and Pcoq (2003).
- Similar to the idea of context menu present in CoqIde, performs multiple actions from a single mouse interaction.
- Already proved theorems should be accessible similarly to assumptions.
- Drag-and-drop: e.g. drag equality to the subterm to rewrite (match the terms and generate subgoals for assumptions of the equality).
- Point-and-shoot: while pointing, select the tactic to apply to the subterm after it is brought to the surface.

Proof-by-Pointing

Rules descend recursively to the position pointed by mouse.

$$\wedge\, left_1 : \frac{\boxed{A}, B, A \wedge B, \Gamma \vdash C}{\boxed{A} \wedge B, \Gamma \vdash C} \qquad \wedge\, right_1 : \frac{\Gamma \vdash \boxed{A} \qquad \Gamma \vdash B}{\Gamma \vdash \boxed{A} \wedge B}$$

$$\wedge\, left_2 : \frac{A, \boxed{B}, A \wedge B, \Gamma \vdash C}{A \wedge \boxed{B}, \Gamma \vdash C} \qquad \wedge\, right_2 : \frac{\Gamma \vdash A \qquad \Gamma \vdash \boxed{B}}{\Gamma \vdash A \wedge \boxed{B}}$$

$$\vee\, left_1 : \frac{\boxed{A}, A \vee B, \Gamma \vdash C \qquad B, A \vee B, \Gamma \vdash C}{\boxed{A} \vee B, \Gamma \vdash C} \qquad \vee\, right_1 : \frac{\Gamma \vdash \boxed{A}}{\Gamma \vdash \boxed{A} \vee B}$$

$$\vee\, left_2 : \frac{A, A \vee B, \Gamma \vdash C \qquad \boxed{B}, A \vee B, \Gamma \vdash C}{A \vee \boxed{B}, \Gamma \vdash C} \qquad \vee\, right_2 : \frac{\Gamma \vdash \boxed{B}}{\Gamma \vdash A \vee \boxed{B}}$$

$$\supset left_1 : \frac{A \supset B, \Gamma \vdash \boxed{A} \qquad B, A \supset B, \Gamma \vdash C}{\boxed{A} \supset B, \Gamma \vdash C} \qquad \supset right_1 : \frac{\boxed{A}, \Gamma \vdash B}{\Gamma \vdash \boxed{A} \supset B}$$

$$\supset left_2 : \frac{A \supset B, \Gamma \vdash A \qquad \boxed{B}, A \supset B, \Gamma \vdash C}{A \supset \boxed{B}, \Gamma \vdash C} \qquad \supset right_2 : \frac{A, \Gamma \vdash \boxed{B}}{\Gamma \vdash A \supset \boxed{B}}$$

$$\forall\, left : \frac{\boxed{A[x \backslash e]}, \forall x\, A, \Gamma \vdash C}{\forall x\, \boxed{A}, \Gamma \vdash C} \qquad \forall\, right : \frac{\Gamma \vdash \boxed{A[x \backslash c]}}{\Gamma \vdash \forall x\, \boxed{A}}$$

type-specific
induction scheme:

$$\frac{\Gamma \vdash P(0) \qquad P(n), \Gamma \vdash P(n+1)}{\Gamma \vdash \forall \boxed{n : int}\ P(n)}$$

$$\exists\, left : \frac{\boxed{A[x \backslash c]}, \exists x\, A, \Gamma \vdash}{\exists x\, \boxed{A}, \Gamma \vdash C} \qquad \exists\, right : \frac{\Gamma \vdash \boxed{A[x \backslash e]}}{\Gamma \vdash \exists x\, \boxed{A}}$$

Proof-by-Pointing: 5-click example

- *Goal (p a \/ q b) /\ (forall x, p x -> q x) -> (exists x, q x).*
- Point to *p a* in conclusion.
  - Two subgoals with *p a* resp. *q b*, and *forall x, p x -> q x* in assumptions, the one with *p a* selected.
- Point to *p x* from *forall x, p x -> q x* in assumptions.
  - *p x* is automatically proved, and thus *q a* is added to assumptions.
- Twice: point to *q x* in conclusion.

Digression: Extracting Text from Proof

Rules for abstraction

$(\lambda l\colon A_{Type}.\, M)_\tau$	$\triangleright$	Let l : A M We have proved τ
$(\lambda h\colon A_{Prop}.\, M)_\tau$	$\triangleright$	Assume A (h) M We have proved τ
$(\lambda x\colon A_{Set}.\, M)_\tau$	$\triangleright$	Consider an arbitrary x in A M We have τ, since x is arbitrary

Rules for application

$(M_{\forall x:P.\,Q}\, N)_\tau$	$\triangleright$	M In particular τ
$(M_{P\supset Q}\, N)_\tau$	$\triangleright$	- N - M We deduce τ

Rules for identifiers

h_τ	$\triangleright$	By h we have τ
T_τ	$\triangleright$	Using T we get τ

Analogous (compact) rules are built for repeated abstractions and applications.

Extracting Text from Proof

Rules for introduction theorems

	$(\mathtt{C}intro\ M^1 \cdots M^n\ N^1 \cdots N^i)_\tau$	$\triangleright$	$-N^1$ $\vdots$ $-N^i$ So by definition of $\mathtt{C}$ we have τ
$i = 0$	$(\mathtt{C}intro\ M^1 \cdots M^n)_\tau$	$\triangleright$	By definition of $\mathtt{C}$ we have τ
$i = 1$	$(\mathtt{C}intro\ M^1 \cdots M^n N)_\tau$	$\triangleright$	N By definition of $\mathtt{C}$ we have τ

Rules for elimination theorems

$(\mathtt{C}elim\ M^1 \cdots M^n\ N^1 \cdots N^i\ P)_\tau$	$\triangleright$	P Therefore by definition of $\mathtt{C}$, to prove τ we have i cases: Case_1: N^1 $\vdots$ Case_i: N^i So we have τ
$i = 0$ $(\mathtt{C}elim\ M^1 \cdots M^n P)_\tau$	$\triangleright$	P, by definition of $\mathtt{C}$ there is a contradiction So we can assert τ
$i = 1$ $(\mathtt{C}elim\ M^1 \cdots M^n\ N\ P)_\tau$	$\triangleright$	P Therefore by definition of $\mathtt{C}$ to prove τ N So we have τ

Extracting Text from Proof: examples

Let U: $Type$
Let P, Q: $U \rightarrow Prop$
Let a: U
Assume $(P\,a)$ (h) and $\forall x{:}U.\,(P\,x) \supset (Q\,x)$ (h_0)
  Applying h_0 with h we get $(Q\,a)$
We have proved $(P a) \supset (\forall x{:}U.\,(P\,x) \supset (Q\,x)) \supset (Q\,a)$
We have proved $\forall U{:}Type.\,\forall P, Q{:}U \rightarrow Prop.\,\forall a{:}U.\,(P\,a) \supset (\forall x{:}U.\,(P\,x) \supset (Q\,x)) \supset (Q\,a)$

By definition of $\mathbb{N}$ to prove $\forall n{:}\mathbb{N}.\,0 \leq n$, we have two cases:
Case_1:
  By definition of $\leq$ we have $0 \leq 0$
Case_2:
  Let $m{:}\mathbb{N}$
  Assume $0 \leq m$ (h)
    From h and the definition of $\leq$, we have $0 \leq (\mathrm{Suc}\,m)$
  We have proved $0 \leq m \supset 0 \leq (\mathrm{Suc}\,m)$
  We have proved $\forall m{:}\mathbb{N}.\,0 \leq m \supset 0 \leq (\mathrm{Suc}\,m)$
So we have $\forall n{:}\,\mathbb{N}.\,0 \leq n$

Let A, B : $Prop$
Assume $A \vee B$ (h)
  Assume A (i)
    From i and the definition of $\vee$, we have $B \vee A$
  -We have proved $A \supset B \vee A$
  Assume B (j)
    From j and the definition of $\vee$, we have $B \vee A$
  -We have proved $B \supset B \vee A$
  -We have h
  Applying $\vee elim$ we get $B \vee A$
We have proved $A \vee B \supset B \vee A$
We have proved $\forall A, B{:}Prop.\,A \vee B \supset B \vee A$

Coq Selected Tactics

- ***eapply*** *term* tries to unify current goal with the conclusion of given term, turns uninstantiated variables in premises into existential meta-variables. (***apply*** forces all *term*'s variables be matched)
- ***rewrite*** *term* and ***rewrite <-*** *term* rewrites equality shown by *term* (**<-** right-to-left) in the goal. *term* can be an assumption name.
- ***destruct*** *ident* „destructs” *ident* generating subgoals for each constructor of inductive predicate which is the type of ident,
- ***injection*** *ident* reduces equality between inductive objects to equalities of their arguments.
- *[**e**]**auto** [**with** ident ... ident | **with** *] [**using** lemma ... lemma]* Prolog-like (depth-first) resolution procedure: reduces goal to an atomic one (intros), tries tactics associated with goal head in turn (lower cost tactics first; theorems used with ***apply***); recurses to subgoals. Either solves the goal completely or leaves intact. *ident*s name hint databases, * means uses all hints, *lemma*s are additional hints. ***eauto*** uses ***eapply***.

Hints provided by Papug

- Tactics for 1st order classical logic.
  - E.g. using *(~A -> B) -> A \/ B* on alternative.
- Axioms for equality of predicates and functions.
- Hints from the auto database.
  - Coq provides lemmas registered with *Hint* that match the goal head, with command *Print Hint*. Papug lets show and apply the applicable lemmas.
- Simplified use of assumption.
  - A generic *Use*, using e.g. *destruct* or *apply* or *injection*
  - *Rewrite*, *Rewrite backwards* for equalities
  - *Induction on H* for inductive objects, *Simplify* (e.g. computation)
- Unfolding definitions, marking obvious goals, proof by contradiction.

Sources

- *„Proof by Pointing”* Yves Bretot, Gilles Kahn, Laurent Thery, 1994
- *„Extracting Text from Proof”* Yann Coscoy, Gilles Kahn, Laurent Thery
- *„The Coq Proof Assistant Reference Manual Version 8.1”* The Coq Development Team: LogiCal Project
- *„Papuq: a Coq assistant”* Jakub Sakowicz and Jacek Chrząszcz, 2007