Calculemus Autumn School

Matthew Fluet

October 28, 2002

Theorem Proving Systems

- Mizar
- Coq
- NuPrl
- Isabelle
- Ωmega

Mizar

An attempt to develop a software environment for writing traditional mathematical papers, where classical logic and set theory form the basis for future developments.

- Mizar Language
  - human readable, machine verifiable proofs
  - multi-sorted (typed) first-order predicate logic with functions, schemata, and second-order free variables
- Mizar Mathematical Library
  - mathematical knowledge base
  - $>$ 720 cross-referenced articles
- Mizar Tools
  - Accommodator, Verifier, Extractor

Mizar: Example Theorems

- Euler's Theorem
- Rolle's Theorem (average value theorem for real functions of one variable)
- Heine-Borel's Covering Theorem
- Fundamental Theorem of Algebra
- Tarski-Knastera Theorem (fixpoints in complete lattices)
- Stone's Representation Theorem (boolean algebras)

Coq

Formalizing mathematics using type theory.

- Extended Calculus of (Co)Inductive Constructions

Exploiting the type checker

- Implicit syntax
  - omit arguments that can be *inferred* by type-checker
    if `f : (S:Set)->S->S->S`, then write `(f ? a b)`
- Coercions
  - declare a term as a coercion from type A to type B
    `Coercion k : A >-> B`
    if `(f a)` cannot be typed, then type checker will
    try `((k f) a)` and `(f (k a))`.

Coq: Case Study

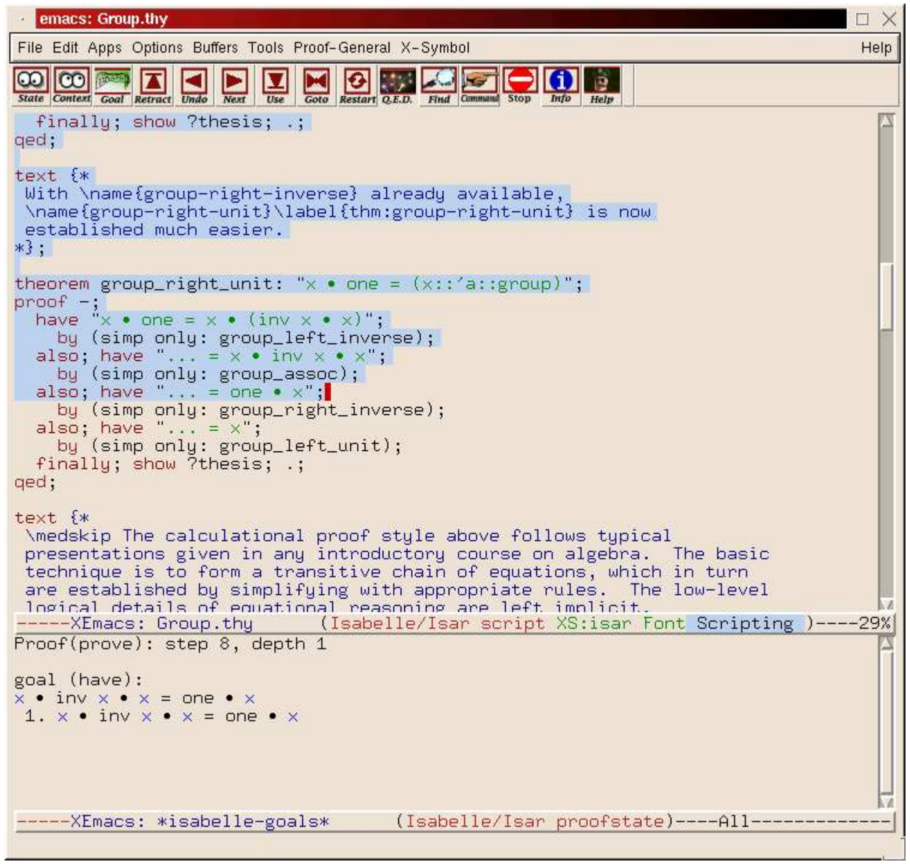
Constructive proof of the Fundamental Theorem of Algebra:

Sets and Basics	41 kb
Algebra (upto Ordered Fields)	165 kb
Reals	52 kb
Polynomials	113 kb
Real-valued functions / Basic Analysis	30 kb
Complex numbers	98 kb
FTA proof	70 kb
Construction of $\mathbb{R}$	309 kb
Rational Tactic	49 kb
LaTeX Proof	25 pp

Isabelle/HOL

Isabelle: a generic system for implementing logical formalizisms

- Isabelle/HOL: specialization of Isabelle to Higher-Order Logic



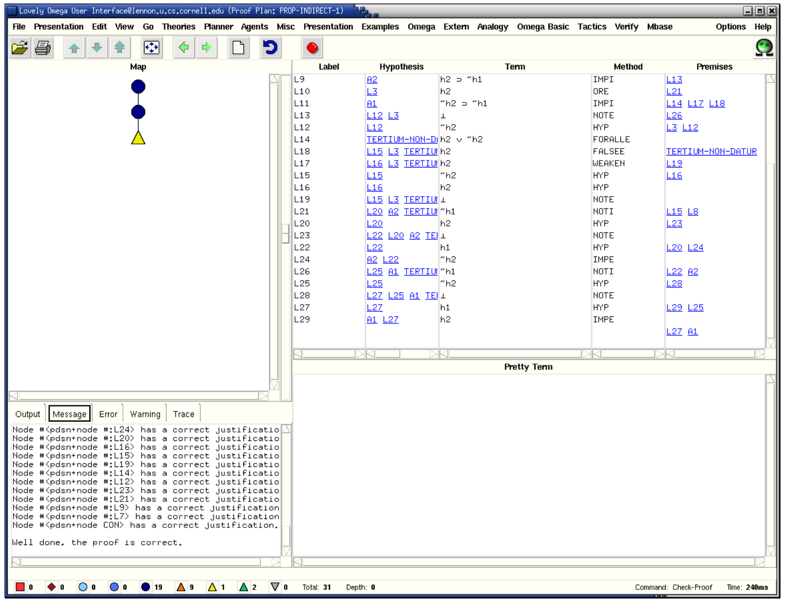
Isabelle/Isar/HOL

Isar: Intelligible semi-automated reasoning

- structured, (human) readable proofs
- includes tactic system as “improper” sub-language
- linguistic/condensed forms of standard rules and methods
- references to proof state via keywords: `this`, `?thesis`
- pattern matching to name sub-formulae

Ωmega

A system with the ultimate purpose of supporting theorem proving in main-stream mathematics and mathematics education. The current system consists of a proof planner and an integrated collection of tools for formulating problems, proving subproblems, and proof presentation.



Ωmega: Case Study

Proof of the irrationality of $\sqrt{2}$

- Ω-ANTS: suggestion mechanism
  - compute possible "next" steps
  - run as concurrent processes
- island tactics
  - proposed proof steps; details need to be expanded

Additional topics

- Proof Planning
- Exploration

Impressions and Thoughts

- focus on mathematics, rather than program specification and correctness
- variety of user interface choices
- Ωmega demonstrated most clearly connections with concurrently running systems

Planning

- Bruno Buchberger
  - Formal Theory Exploration (The Lazy Thinking Paradigm)
- Alan Bundy
  - Proof Planning (with XBarnacle)

Formal Theory Exploration

- opposed to Isolated Theorem Proving/Reasoning
- describe "completely" the interplay of (one or several) new notions with the given notion
- mathematical knowledge management (MKM)
  - "all of mathematics" can be stored/generated/retrieved in compact form by a hierachy of functors and their truth transportation properties

The Lazy Thinking Paradigm

- attempt a proof
  - problem/knowledge/algorithm schemes
- analyze the failure
- conjecture a lemma
  - generalization of failure
  - extract "re-usable" knowledge
  - add lemma to the knowledge base
- Case study: list sorting
  - divide-and-conquer algorithm scheme

$$S[X] = \left\{ \begin{array}{ll} special[X] & , is{-}basic[X] \\ compose[S[one{-}part[X]], S[other{-}part[X]]] & , otherwise \end{array} \right\}$$

Proof Planning

Rippling: proof planning for induction

- selective, bi-directional, terminating
- offers heuristic solution to problems of choosing lemmas, generalizations, and inductions
  - ripple analysis: induction rules chosen to support rippling
  - pattern of proof breakdown suggest different patches

Proof Planning: XBarnacle

Implementation of proof planning with rippling.

- Critics support choosing lemmas and generalizations