Integrating the Text-Editor $\text{TEX}_{\text{MACS}}$ with the Proof Assistance System $\Omega\text{MEGA}$ using $\text{PLAT}$

Serge Autexier, Christoph Benzmüller, Andreas Franke, Dominik Dietrich, Henri Lesourd, Marc Wagner

DFKI GmbH & CS Department, Saarland University, Saarbrücken, Germany

CHIT-CHAT’06, 21st December 2006

Nijmegen, Netherlands
Motivation

- Evolution of computer-supported mathematics
- Low user-friendliness of actual proof assistants
- Interfacing scientific text-editors instead of yet another proof assistant GUI
System Architecture

- Initializing a session
- Uploading a document
- Patching a document
- Requesting a menu
- Executing a menu action
- Closing a session
The Mediator PLATΩ
Mediation Problem

- Proof Language
  - flexible semantic annotation

- Service Menus
  - context-sensitive interaction

- PLATO
  - session management
  - syntax analysis
  - global transformation
  - propagation of changes
  - service interaction
  - maintenance of consistent versions

- OMEGA
  - structured math. theories
  - logical structure
  - sequents + focus
  - proof planning

Source: Autexier, Wagner
Document Language

- Semantic Annotation of Natural Language
  - Macros in the text-editor
  - Individual layout style

- Designed to support
  - Textual structure of proofs
  - Flexible and multiple positioning
  - Underspecification
  - Alternative proof attempts

- Parameters
  - (Sub-)Languages for formulas, definitions and references
Global Transformation

- Normalizing the flexible semantic annotations
- Separating proofs from theory knowledge
- Transforming linear proofs into the tree-like proof structure

**Proof Language (PL)**
- Theorems (flexible)
- Proofs (flexible, linear)
- Natural language text

**Development Graph Language (DL)**
- Theorems (rigid)

**Intermediate Language (IL)**
- Proofs (rigid, linear)

**Task Language (TL)**
- Proofs (rigid, tree)
Propagation of Changes

- Semantic-based difference computation
- Efficient transformation of differences
- Preserve partial verifications in proof datastructure of PA

Proof Language (PL)
  - Theorems (flexible)
  - Proofs (flexible, linear)
  - Natural language text

Intermediate Language (IL)
  - Proofs (rigid, linear)

Development Graph Language (DL)
  - Theorems (rigid)

Task Language (TL)
  - Proofs (rigid, tree)
Service Interaction: Requesting a Menu

- The user selects object in the text-editor
- Lookup all related objects in the maptable of the transformation
- Request a menu from the PA for these objects
- Display the menu in the text-editor
Service Interaction: Executing Actions

- User selects an action and its arguments in the menu
- Evaluate this action in the proof assistance system
- Evaluation result:
  - Patches to the menu differences (nested evaluation)
  - Patches to the DL and TL documents (top level evaluation)
- Transformation of DL/TL patches into PL patches
The Mediator PLATΩ

- XML-RPC server as interface for the text-editor
- Connection to proof assistant in Lisp
- Operating as online webservice or local plugin

Source: Autexier, Wagner
The Plugin in TeXmacs
Rôle of the Plugin

- Key and session management
- Patch applications on the $\TeX_{MACS}$ document
- Resolve and check references
- Communicates with $\text{PLAT}_\Omega$ by fully annotated documents
- Manually writing a fully annotated document is tedious
  - Have a context sensitive menu to write the annotations
  - May be acceptable for the user to write large structures (begin definition, theorem, proof, etc.)
  - But certainly not for formulas . . .
    - To obtain: $x \in U \Rightarrow x \in V$
    - Write:
      \[
      \text{imp}\{\text{in}\{\text{V}\{x\}\}\{\text{V}\{U\}\}\}\{\text{in}\{\text{V}\{x\}\}\{\text{V}\{V\}\}\}\}
      \]
Writing Formulas

■ Use a parser for formulas

■ Allow the user to define its own syntax for any concepts

\notation{for=in}{
Let \declare{x} be an element and \declare{A} be a set. Then we write \denote{x \in A}, \denote{x is in A}, \denote{x is element of A}, or \denote{A contains x}.
}

**Notation.** Let \( x \) be an element and \( A \) be a set. Then we write \( x \in A, x \text{ is in } A, x \text{ is element of } A, \text{ or } A \text{ contains } x \).

■ Procedure:
  
  ► Scan all notation definitions, convert automatically into parser grammar rules
  
  ► Using parser generator create (document) specific parser
  
  ► Use that parser to parse formulas
The Parser Generator

- Standard LALR(1) parser generator implemented in Scheme
- Creates parser that generates all possible readings
- Allows specification of external call-backs to use to single-out invalid readings.
  Example: Could be used to integrate a “refiner”
- Sophisticated specification mechanism for precedences of operators
The Interface of the Proof Assistant

- Initializing a session
- Uploading a document
- Patching a document
- Requesting a menu
- Executing a menu action
- Closing a session

TeXmacs → PLATO → ΩMEGA CORE

Source: Autexier, Wagner
Features of the Proof Assistant

- Create inference rules from axioms/lemmas
  Example: $\forall A, B : \text{set.} (A \subseteq B \land B \subseteq A) \Rightarrow A = B$

\[ \frac{A \subseteq B \quad B \subseteq A}{A = B} \]

- Proof construction on that level (assertion level)
- Requires proof representation that allows encoding of proof continuations
- Every (additional) feature of the PA is immediately available on the corresponding text part.
  Example: Automatic Theory Exploration System \textsc{MathSAID}
System Demo

- Definition of concepts and their notations
- Writing Axioms, Conjectures using pre-defined and user-defined notations
- Proof support: interactive and automatic
Related Work

- Automath, Mizar, Isar:
  - Balanced compromise between machine processable and human readable

- Grammatical Framework:
  - Framework to define grammar for an abstract and a concrete syntax

- PCOQ:
  - Schematic quasi-natural language output

- Nuprl, Clam, Omega/P.Rex:
  - Natural language processing technique to generate proof descriptions
Related Work

- **Theorema:**
  - Strictly separated formal and informal parts

- **Mathlang:**
  - Top-down from natural language
  - Use annotations for structure, no parser as well
  - Still even more far away from PAs

- **ProofGeneral:**
  - Top-down processing of documents
  - Documents are input format of PA rather than of some typesetting program.
Conclusion

- Have a stable connection between $\text{T}_{\text{TEX}}\text{MACS}$ and $\Omega\text{MEGA}$
  - $\text{PLAT}\Omega$ deals with all mediating (translation, consistency, patching, relationship between parts of text and formal parts in PA, menus)
  - Clean interface to text-editor
  - Parameterized over language for formulas, definitions and references

- Either side can be enhanced without affecting the mediator
  - $\text{PLAT}\Omega$ plugin:
    - User-definable notation used when parsing formulas
    - Add more NL analysis to automatically annotate text(parts)
    - Add NL generation (incremental)
  - $\Omega\text{MEGA}$:
    - Added theory exploration, classical ATPs are available
    - Increase proof search automation
Future Work

■ Technicalities:
  ► XML-RPC Interface for PAs (OMDOC based)
  ► Asynchronous communications

■ Editing:
  ► Library mechanism
  ► Dependent types (Scunak, ΩMEGA’07, others are welcome)
  ► Support overloading
  ► More natural language analysis and NL generation
  ► Collaborative editing

■ Use it to formalise mathematics
Future Work

Text-Editors
- Emacs
- TeXmacs

Natural Language
- MathLang
- NL Analysis & Generation

Proof Assistants
- Omega
- Isabelle
- Coq

OMDOC
- Proof General