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

Serge Autexier
(Joint work with Christoph Benzmüller, Andreas Franke, Dominik Dietrich, Henri Lesourd, Marc Wagner)

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

6 February, 2007

LORIA, Nancy, France
Research Goals

Background: Artificial Intelligence, Formal methods, deduction
(eespecially, Inductive Theorem Proving)

- The potential of using formal logic based techniques is clearly recognized (mathematics, reliability of software systems, ...)
- Formal logic based tools are not routinely employed like, for instance, CAS by mathematicians, physicists or engineers.
- Two reasons:
  - the high hurdle posed by the deep formal logical knowledge that is necessary to effectively use these techniques and tools
  - that the formal techniques and tools are not integrated with standard (software) development environments.
Past and Ongoing Research

Semantics-based management of change:

- Evolutionary nature of specification and verification: MAYA
  - Development graphs to represent and exploit theory structure (now semantics for theories and locales in Isabelle)
  - Difference Analysis & Propagation of changes to preserve proof knowledge
  with D. Hutter and T. Mossakowski

- Offspring: Semantics based XML difference computations (Masters, S. Radzevich)
  with D. Hutter
Past and Ongoing Research

Usability of formal reasoning tools:

- Assertion-Level (Huang’93) reasoning
  - (CORE calculus, PhD 2003)
  - Tasklayer of the new version of the proof assistant $\Omega$MEGA
    (Diploma thesis D. Dietrich)

- The next version of the $\Omega$MEGA Proof assistant (Project in CRC “Ressource-adaptive cognitive processes (SFB-378)”)

- Integrating reasoning systems as Web Services: MATHSERVE
  (PhD J. Zimmer)

- Automated reasoning in large structured theories: ATLAS
  EPSRC Visiting Researcher (with A. Bundy and R. McCasland)
Augmenting authoring and development environments by formal techniques:

- Evolution of computer-supported mathematics (and hence to all areas that employ mathematics)
- Low user-friendliness of actual proof assistants
- Interfacing scientific text-editors instead of yet another proof assistant GUI (Project VeriMathDoc (with A. Fiedler) and Diploma thesis M. Wagner)
Past and Ongoing Research

Augmenting authoring and development environments by formal techniques:

- Evolution of computer-supported mathematics (and hence to all areas that employ mathematics)
- Low user-friendliness of actual proof assistants
- Interfacing scientific text-editors instead of yet another proof assistant GUI (Project VeriMathDoc (with A. Fiedler) and Diploma thesis M. Wagner)

Brings together research developments and experience from

- Evolutionary software development (MAYA),
- Semantics-based XML-DIFF,
- Proof construction at the assertion-level (CORE, Tasklayer)
This theory defines the basic concepts and properties of the Theory of Simple Sets.

**Definition 1.1 [Type of Elements]**:
First of all we define the type \texttt{elem}.

**Definition 1.2 [Type of Sets]**:
Then we define the type \texttt{set}.

**Definition 1.3 [Function \( \in \)]**:
The function \( \in_{\texttt{elem} \times \texttt{set}} \rightarrow \texttt{bool} \) takes an individual and a set
and tells whether that individual belongs to this set.

**Notation 1.4**:
Let \( x \) be an individual and \( A \) a set, then we write
\( x \in A \), \( x \) is an element of \( A \), \( x \) is in \( A \) or \( A \) contains \( x \).

**Definition 1.5 [Function \( \subset \)]**:
The function \( \subset_{\texttt{set} \times \texttt{set}} \rightarrow \texttt{bool} \) takes two sets
and tells whether the first set is a subset of the second set.

**Notation 1.6**:
Let \( A \) and \( B \) be sets, then we write \( A \subset B \).
System Architecture

TeXmacs

XML
RPC

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

LISP

OMEGA
CORE

Source: Autexier
The Mediator PLATΩ
Mediation Problem

Proof Language
- flexible semantic annotation

Service Menus
- context-sensitive interaction

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

ΩMEGA

Develop. Graph Representation
- structured math. theories

Task Layer Representation
- logical structure
- sequents + focus
- proof planning

Source: Autexier
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)

Source: Autexier
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
The Plugin in TeXmacs

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

Source: Autexier
Rôle of the Plugin

- Key and session management
- Patch applications on the \( \TeX_{\text{MACS}} \) 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:
      \[
      \text{\texttt{\textbackslash imp\{\text{\textbackslash in}\{\text{\textbackslash V}\{x\}\}\{\text{\textbackslash V}\{U\}\}\}\{\text{\textbackslash in}\{\text{\textbackslash V}\{x\}\}\{\text{\textbackslash V}\{V\}\}\}}}
      \]

... by [Definition of \( \subseteq \)] ...
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 \( x \in A, \) \( x \) is in \( A, \) \( x \) is element of \( A, \) or \( A \) 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
Features of the Proof Assistant

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

\[
\begin{align*}
A \subseteq B & \quad B \subseteq A \\
\hline
A = B
\end{align*}
\]

Mixture of CORE ideas and B. Wacks’ super natural deduction

- 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 MATHS AID
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 $\text{T}_{\text{MACS}}$ and $\Omega$MEGA
  - $\Omega$LAT 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
  - $\Omega$LAT plugin:
    - User-definable notation used when parsing formulas
    - Add more NL analysis to automatically annotate text(parts)
    - Add NL generation (incremental)
  - $\Omega$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
  - Semantics-based versioning, collaborative editing

- **Proof support**
  - Automation of proof search at the assertion-level
  - Exploit structure of and in theories to enhance automatic proof search
Future Work

Text-Editors
- Emacs
- TeXmacs

Natural Language
- MathLang
- NL Analysis & Generation

Proof Assistants
- Omega
- Isabelle
- Coq
- Proof General

OMDOC