User Interfaces for Theorem Provers
August 21th 2006
Seattle, Washington, USA

PLATΩ: A Mediator between Text-Editors and Proof Assistance Systems

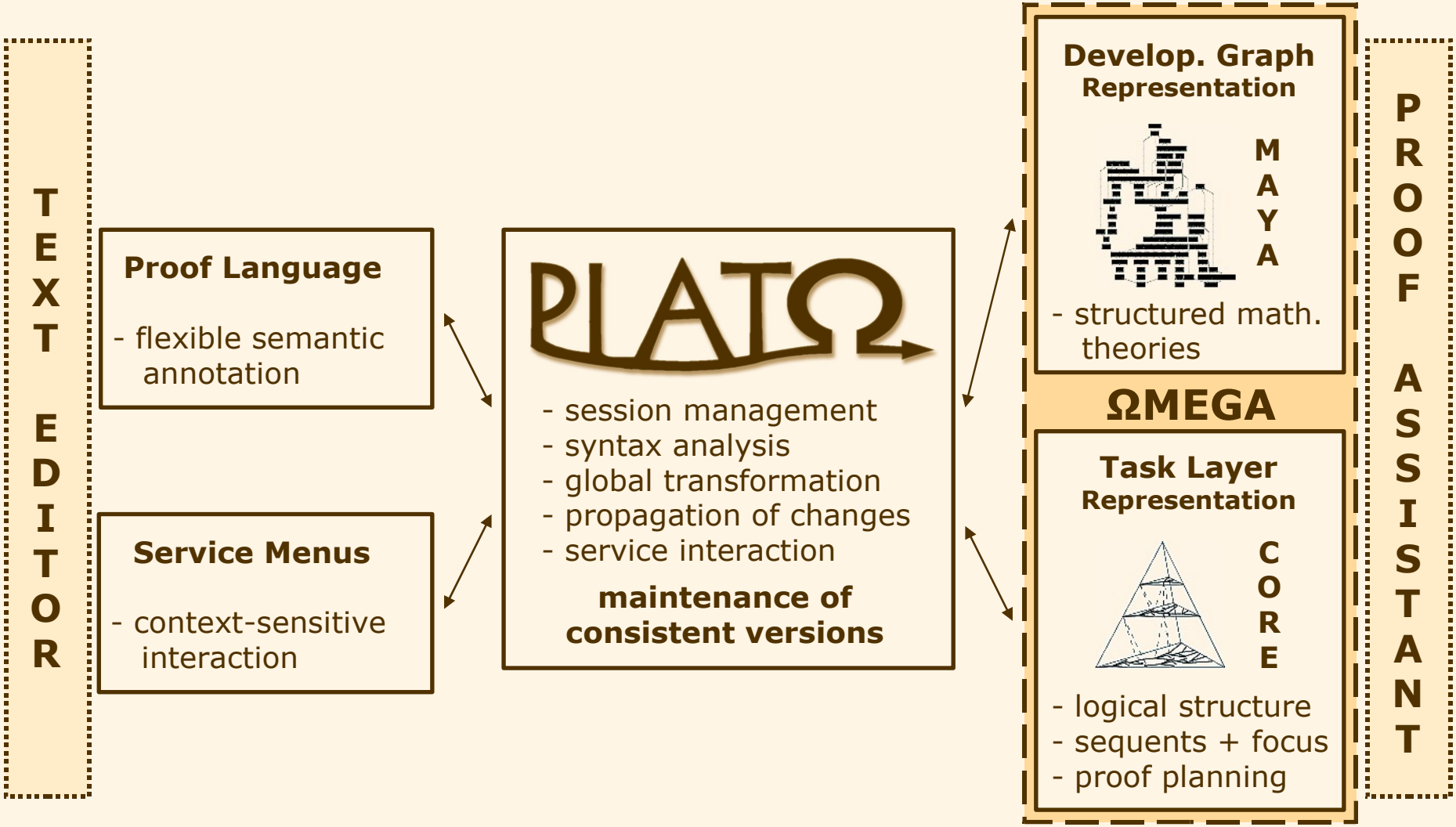
Marc Wagner
Serge Autexier
Christoph Benzmüller

Saarland University
DFKI GmbH
Saarbrücken, Germany

Motivation

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

Mediation Problem



Proof 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
- Parameterized over
  - languages for formulas, definitions and references

```
[document]
 └─[theory]
    ├ [context]
    │  └─[reference]
    ├ [axiom]
    ├ [definition]
    ├ [theorem]
    │  ├─[precondition]
    │  │  └─ [formula]
    │  └─[conclusion]
    │     └─ [formula]
    └ [proof]
       ├─[cases]
       │  ├─ [case]
       │  └─ [proof]
       ├─[subgoals]
       ├─[assumption]
       ├─[fact]
       │  ├─ [by]
       │  └─ [obtain]
       └─[done]
```

Maintenance of Consistency

- Global Transformation
  - normalizing the flexible semantic annotation
  - separating proofs from theory knowledge
  - transforming linear proofs into the treelike datastructure

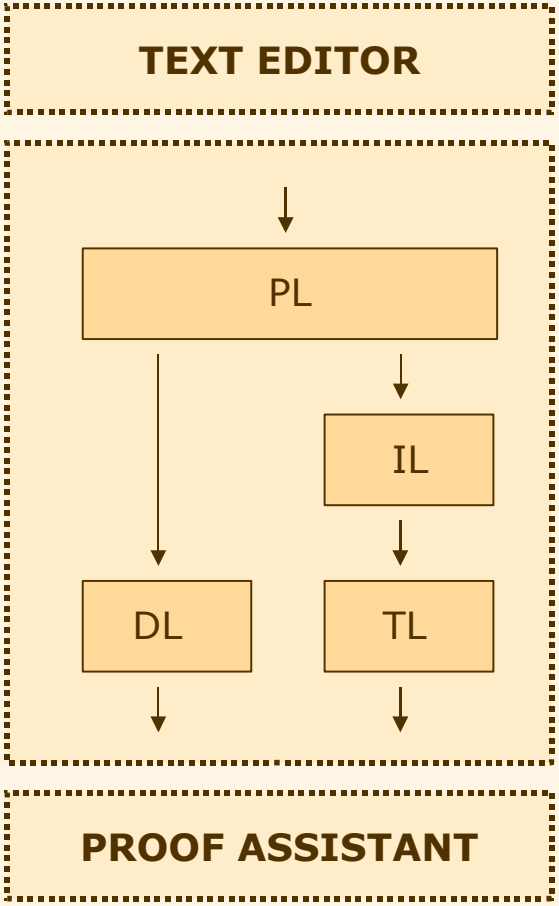
Proof Language (PL)
- theorems (flexible)
- proofs (flexible, linear)
- natural language text

Development Graph Language (DL)
- theorems (rigid)

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

Task Language (TL)
- proofs (rigid, tree)

Global Transformation



Maintenance of Consistency

- Propagation of Changes
  - semantic-based difference computation
  - efficient transformation of differences
  - preserving partial verifications in the proof datastructure

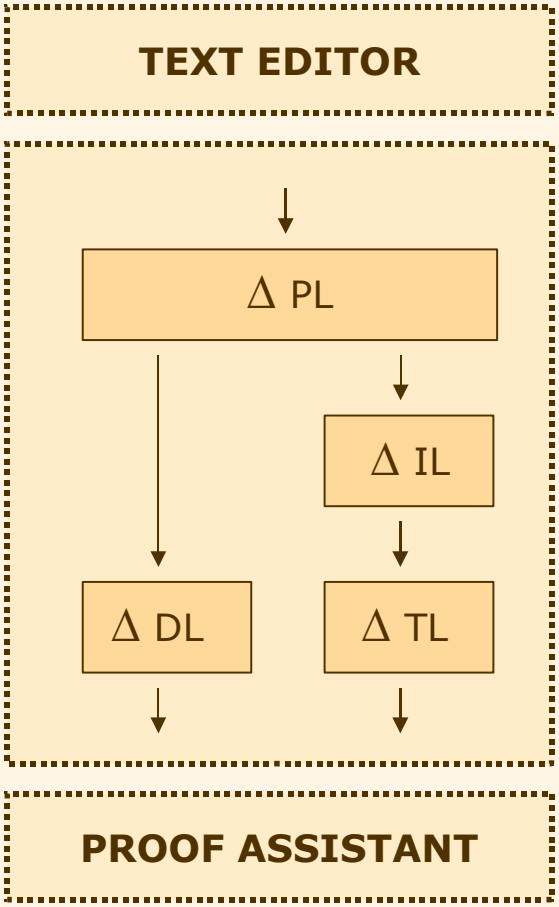
Proof Language (PL)
- theorems (flexible)
- proofs (flexible, linear)
- natural language text

Development Graph Language (DL)
- theorems (rigid)

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

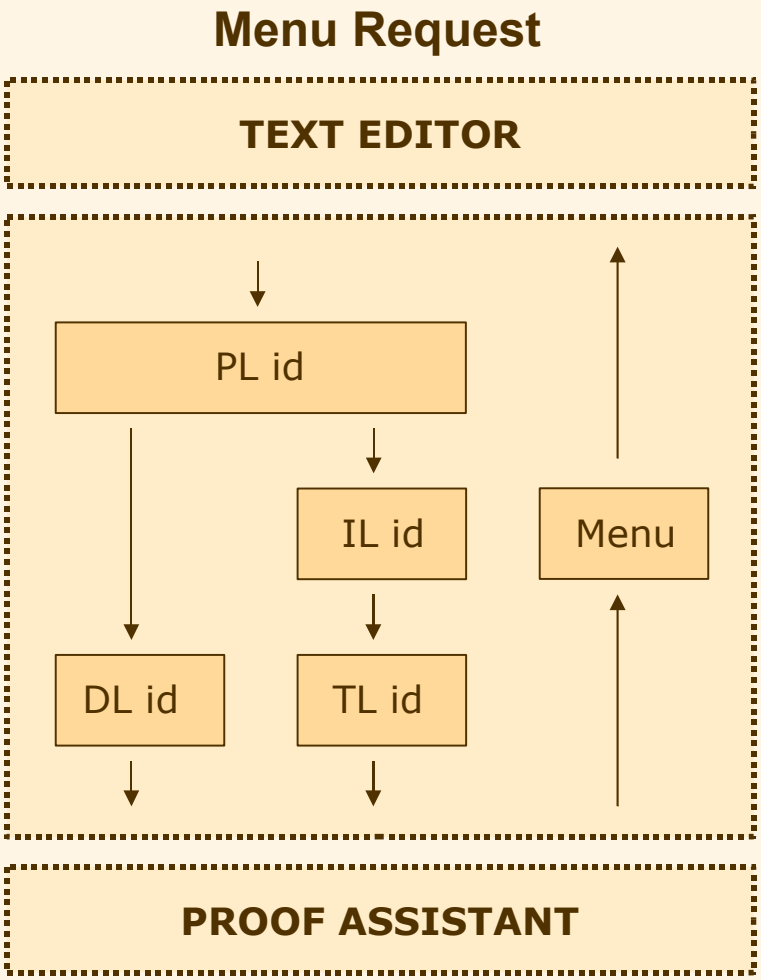
Task Language (TL)
- proofs (rigid, tree)

Propagation of Changes



Service Interaction

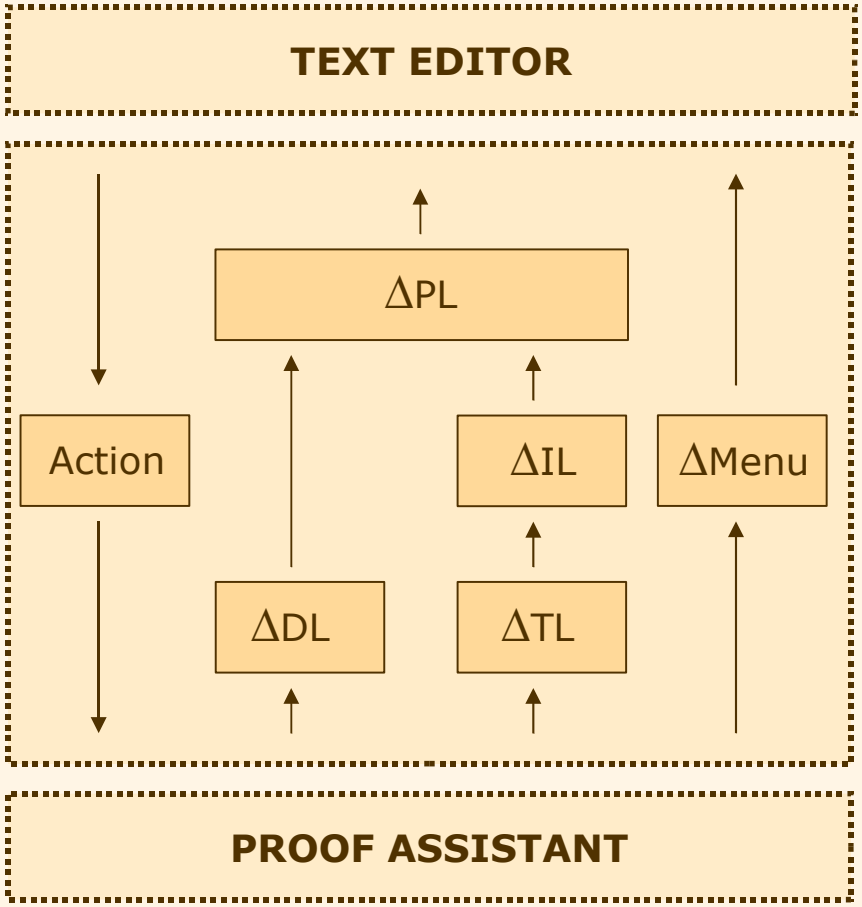
- Menu Request
  - user selects object in the text-editor
  - look up all related objects in the maptable of the transformation
  - request menu from the proof assistance system for these objects
  - display the menu in the text-editor



Service Interaction

- Action Execution
  - user selects an action and its arguments in the menu
  - evaluate this action in the proof assistance system
  - evaluation result:
    - menu differences (nested evaluation)
    - document differences (top level evaluation)
  - propagation of changes

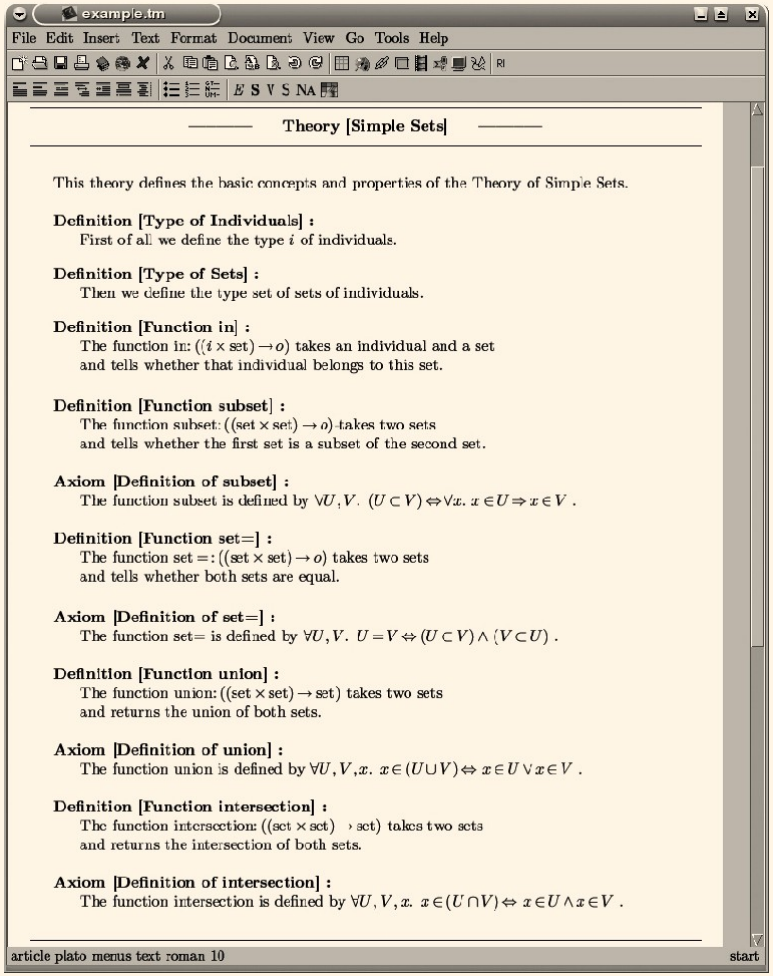
Action Execution



PLATΩ : System Demo

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

PLATΩ : System Demo



Theory [Simple Sets]

This theory defines the basic concepts and properties of the Theory of Simple Sets.

Definition [Type of Individuals] :
First of all we define the type i of individuals.

Definition [Type of Sets] :
Then we define the type set of sets of individuals.

Definition [Function in] :
The function in: $((i \times \text{set}) \rightarrow o)$ takes an individual and a set and tells whether that individual belongs to this set.

Definition [Function subset] :
The function subset: $((\text{set} \times \text{set}) \rightarrow o)$ takes two sets and tells whether the first set is a subset of the second set.

Axiom [Definition of subset] :
The function subset is defined by $\forall U, V.\ (U \subset V) \Leftrightarrow \forall x.\ x \in U \Rightarrow x \in V$.

Definition [Function set=] :
The function set = : $((\text{set} \times \text{set}) \rightarrow o)$ takes two sets and tells whether both sets are equal.

Axiom [Definition of set=] :
The function set= is defined by $\forall U, V.\ U = V \Leftrightarrow (U \subset V) \wedge (V \subset U)$.

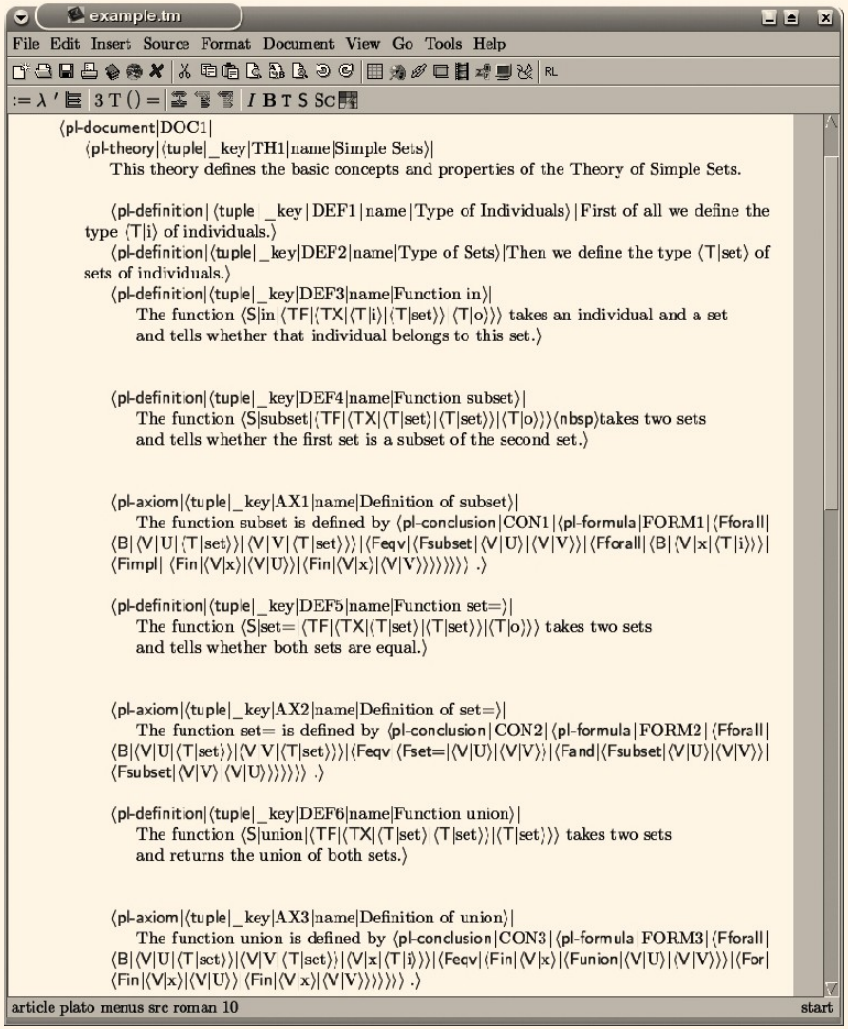
Definition [Function union] :
The function union: $((\text{set} \times \text{set}) \rightarrow \text{set})$ takes two sets and returns the union of both sets.

Axiom [Definition of union] :
The function union is defined by $\forall U, V, x.\ x \in (U \cup V) \Leftrightarrow x \in U \vee x \in V$.

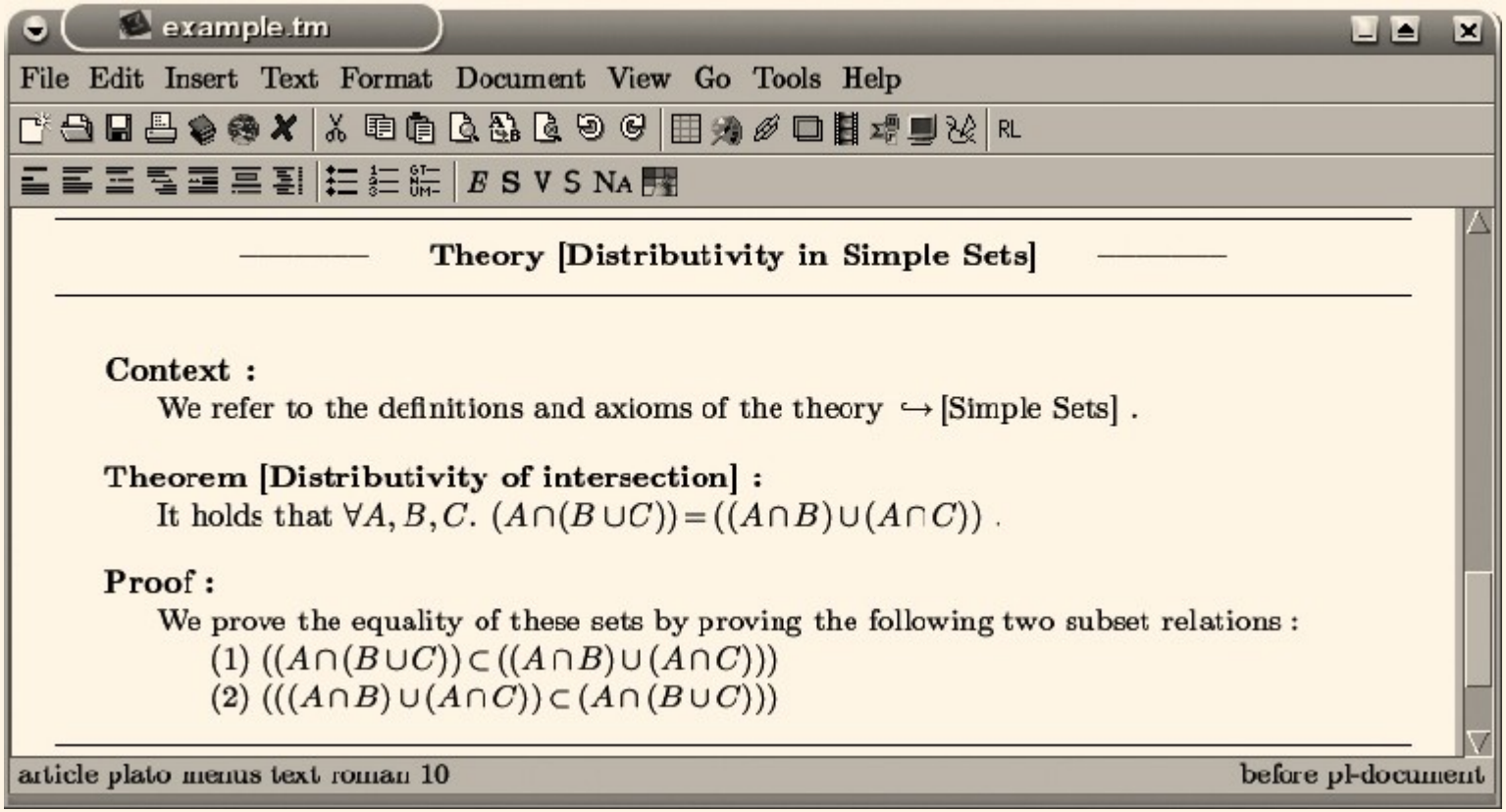
Definition [Function intersection] :
The function intersection: $((\text{set} \times \text{set}) \rightarrow \text{set})$ takes two sets and returns the intersection of both sets.

Axiom [Definition of intersection] :
The function intersection is defined by $\forall U, V, x.\ x \in (U \cap V) \Leftrightarrow x \in U \wedge x \in V$.

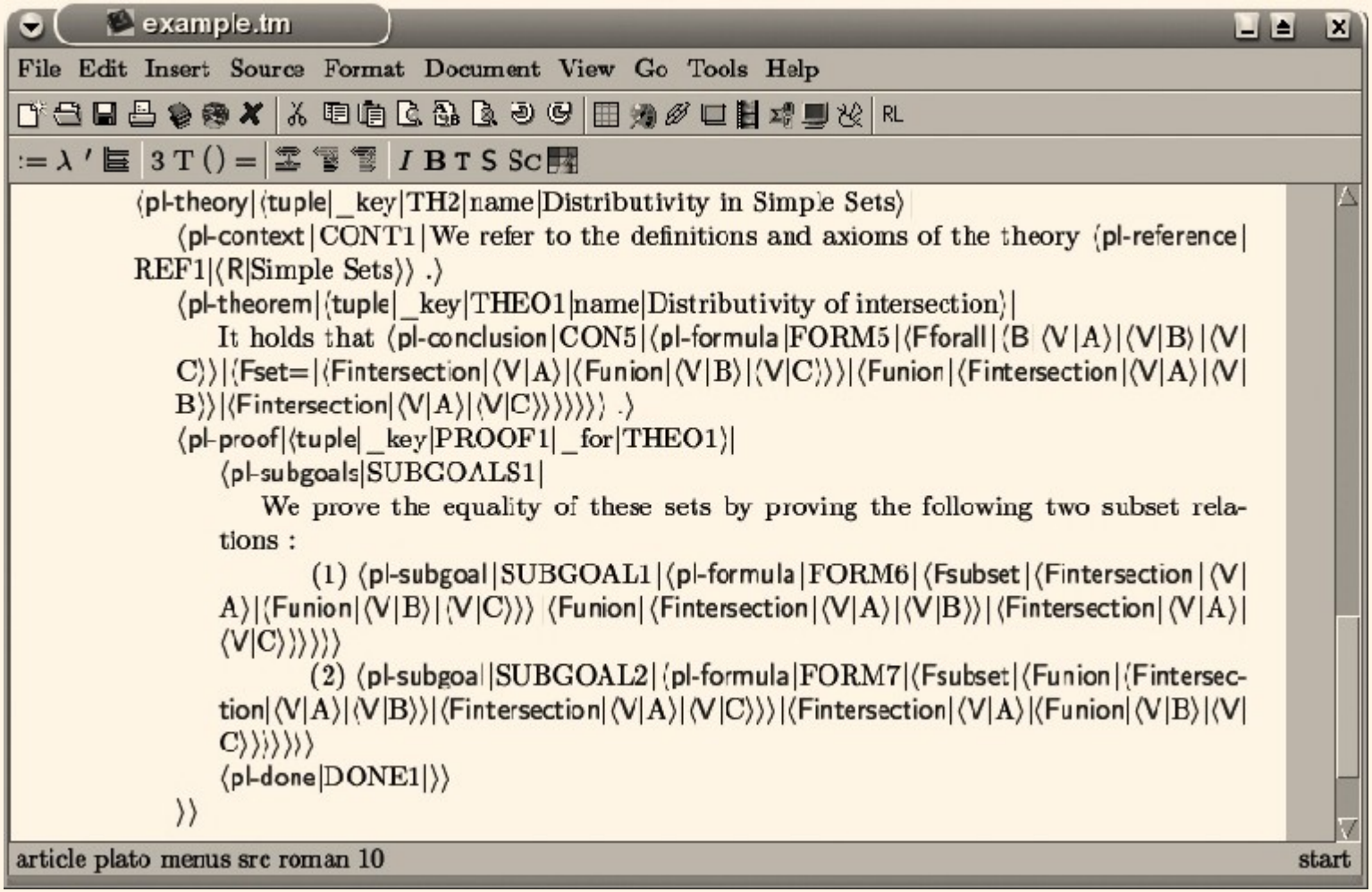
PLATΩ : System Demo



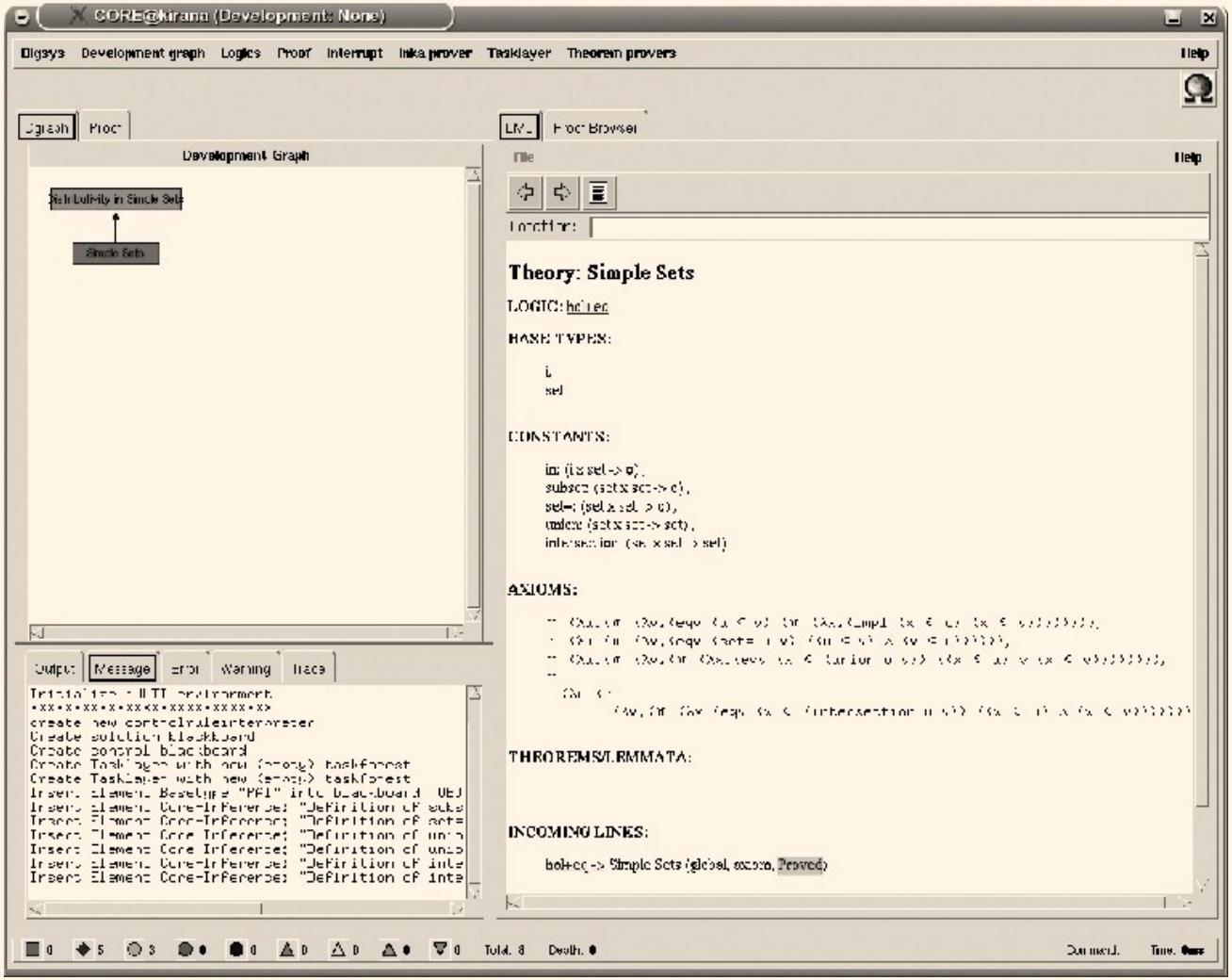
PLATΩ : System Demo



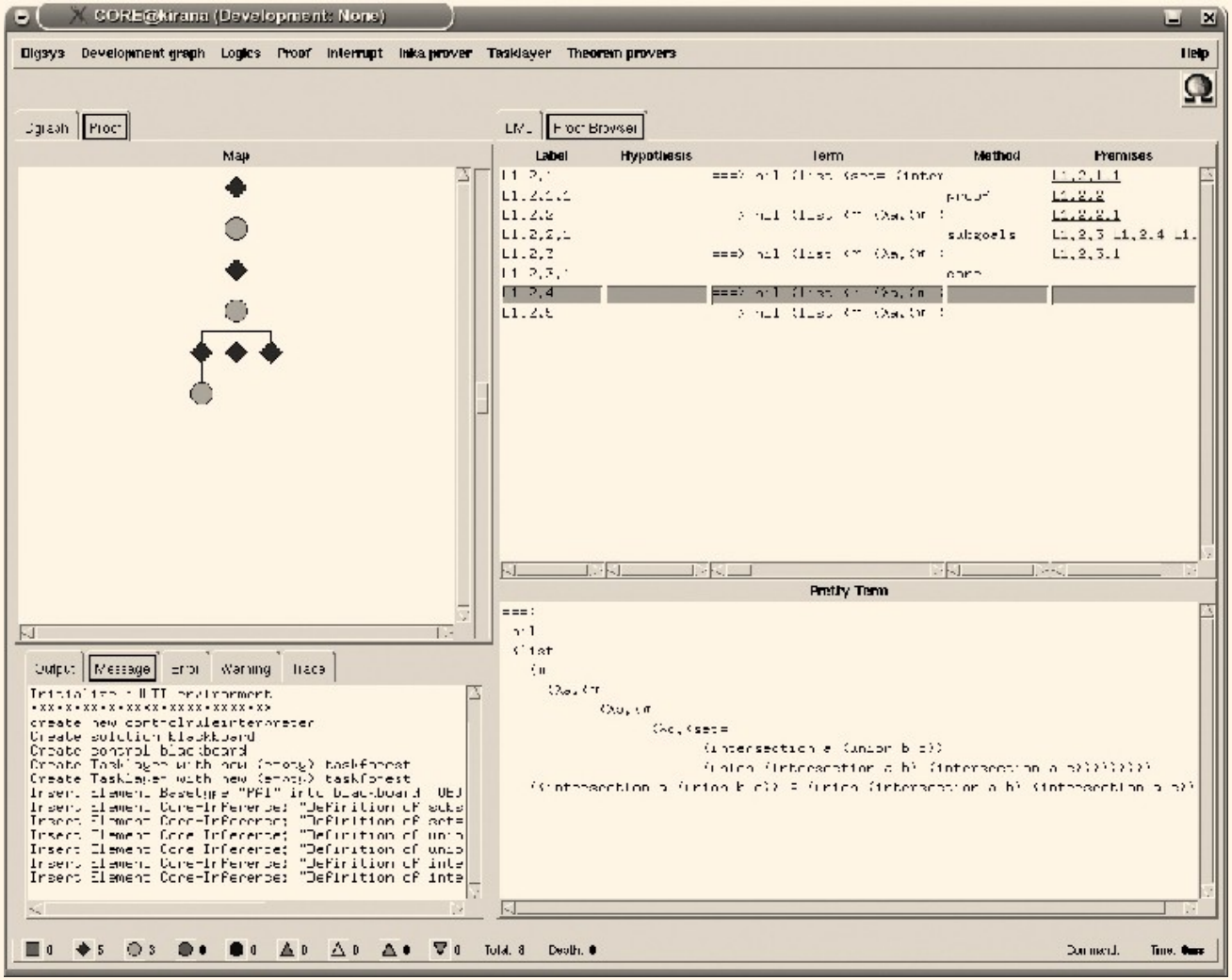
PLATΩ : System Demo



PLATΩ : System Demo



PLATΩ : System Demo



PLATΩ : System Demo

example.tm

File Edit Insert Text Format Document View Go Tools Help

Theory [Distributivity in Simple Sets]

Context :

We refer to the definitions and axioms of the theory ↪ [Simple Sets] .

Theorem [Distributivity of intersection] :

It holds that $\forall A, B, C.\ (A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C))$.

Proof :

We prove the equality of these sets by proving the following two subset relations :

(1) $((A \cap (B \cup C)) \subset ((A \cap B) \cup (A \cap C)))$

(2) $(((A \cap B) \cup (A \cap C)) \subset (A \cap (B \cup C)))$

We start by proving the first subgoal.

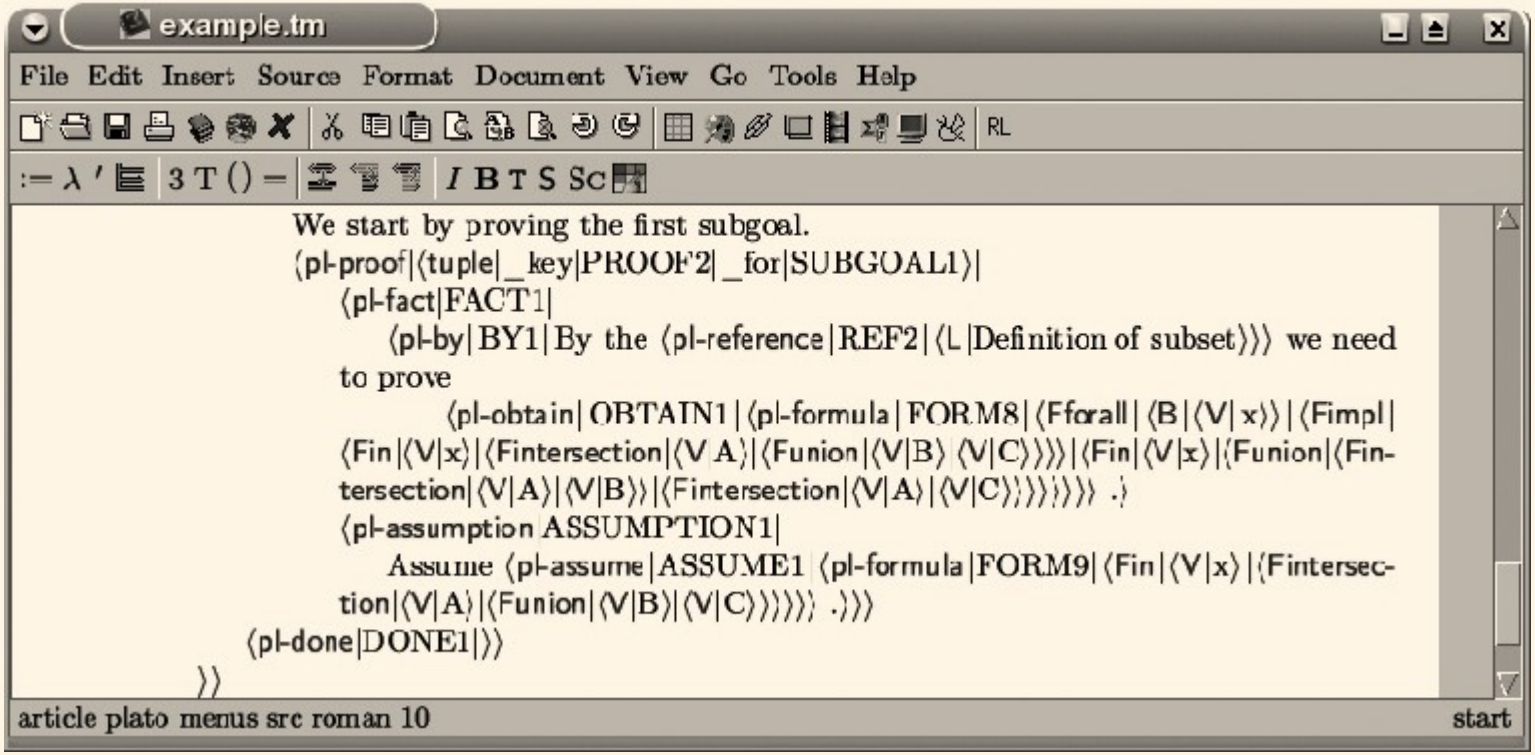
By the → [Definition of subset] we need to prove

$\forall x.\ x \in (A \cap (B \cup C)) \Rightarrow x \in ((A \cap B) \cup (A \cap C))$.

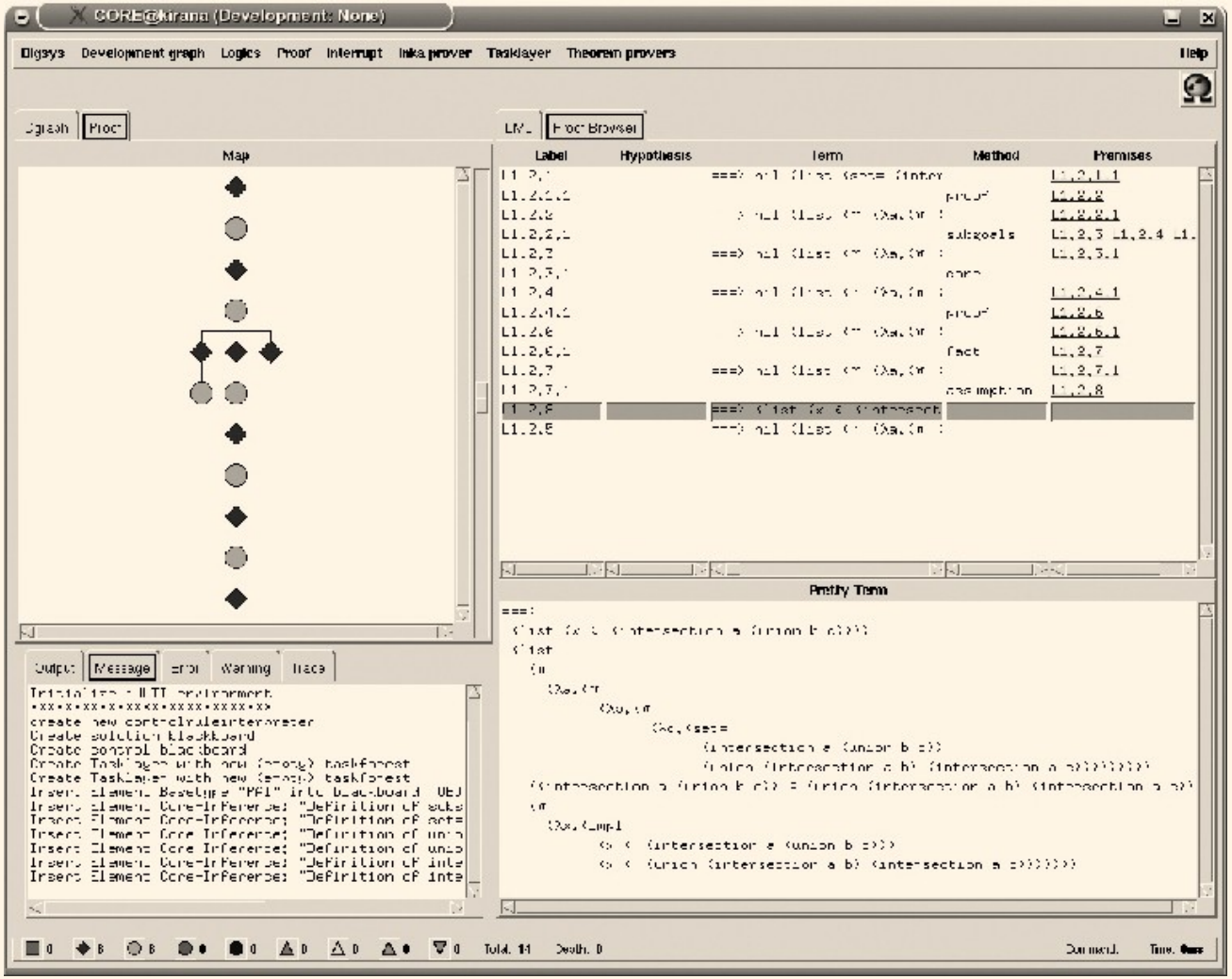
Assume $x \in (A \cap (B \cup C))$.

article plato menus text roman 10 start

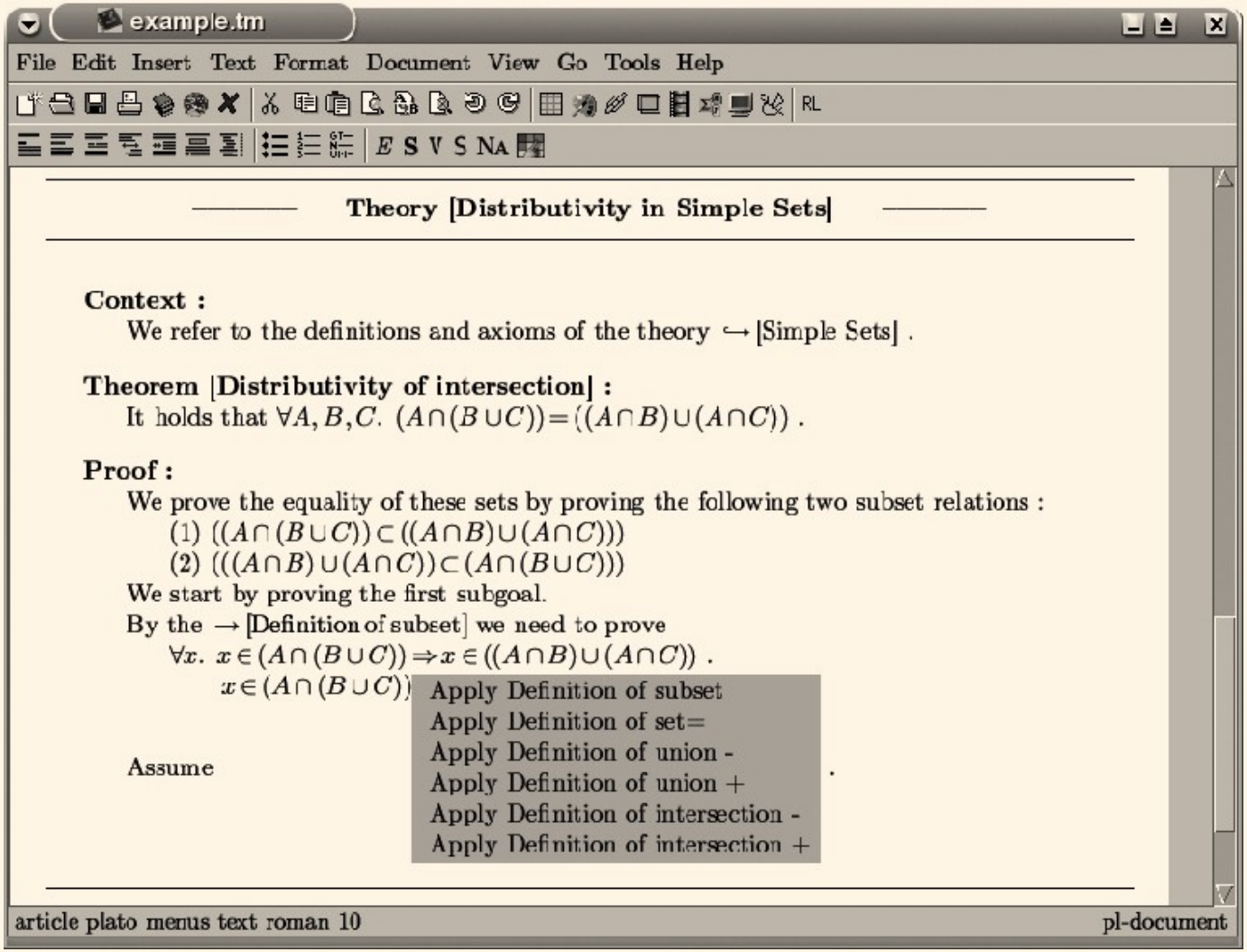
PLATΩ : System Demo



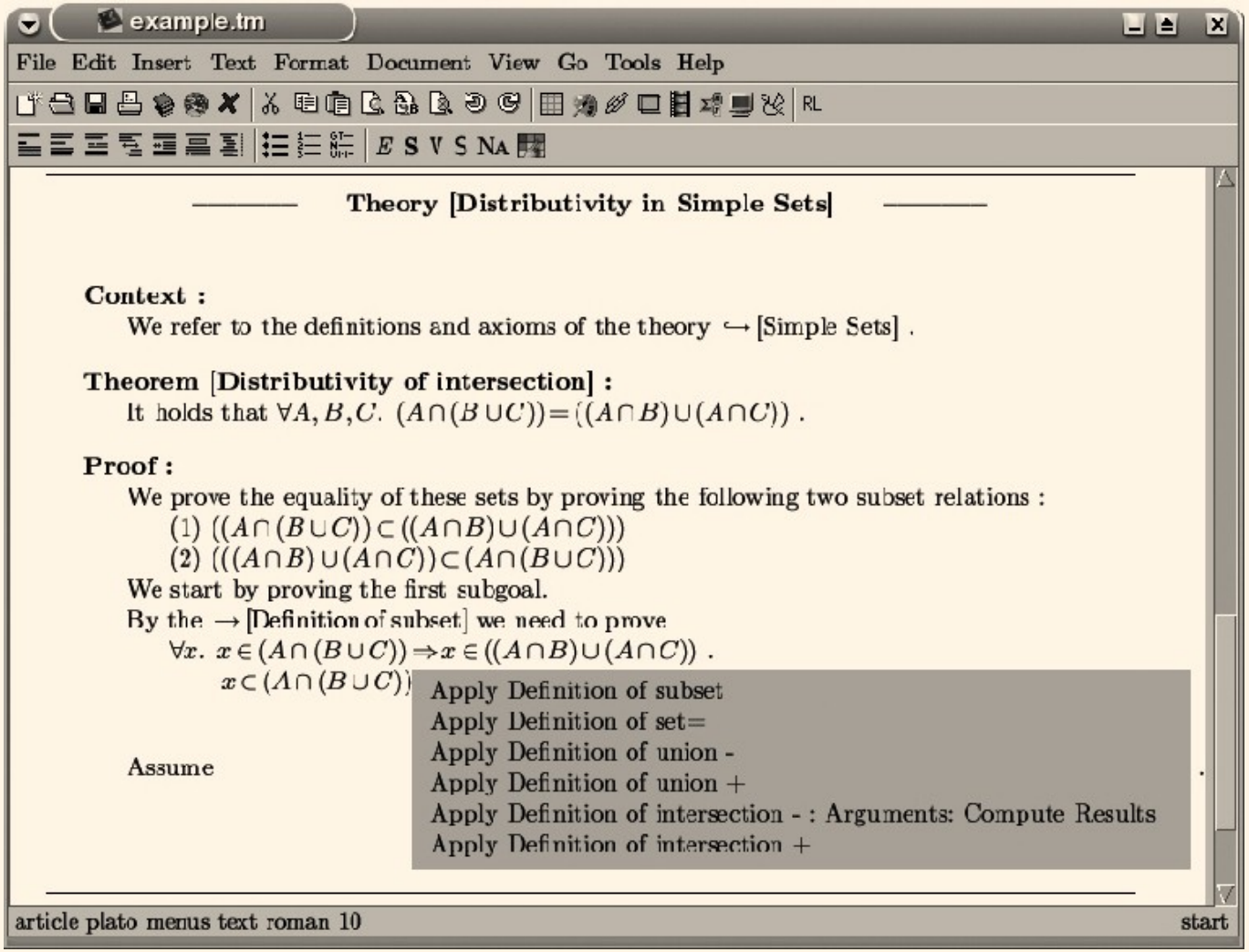
PLATΩ : System Demo



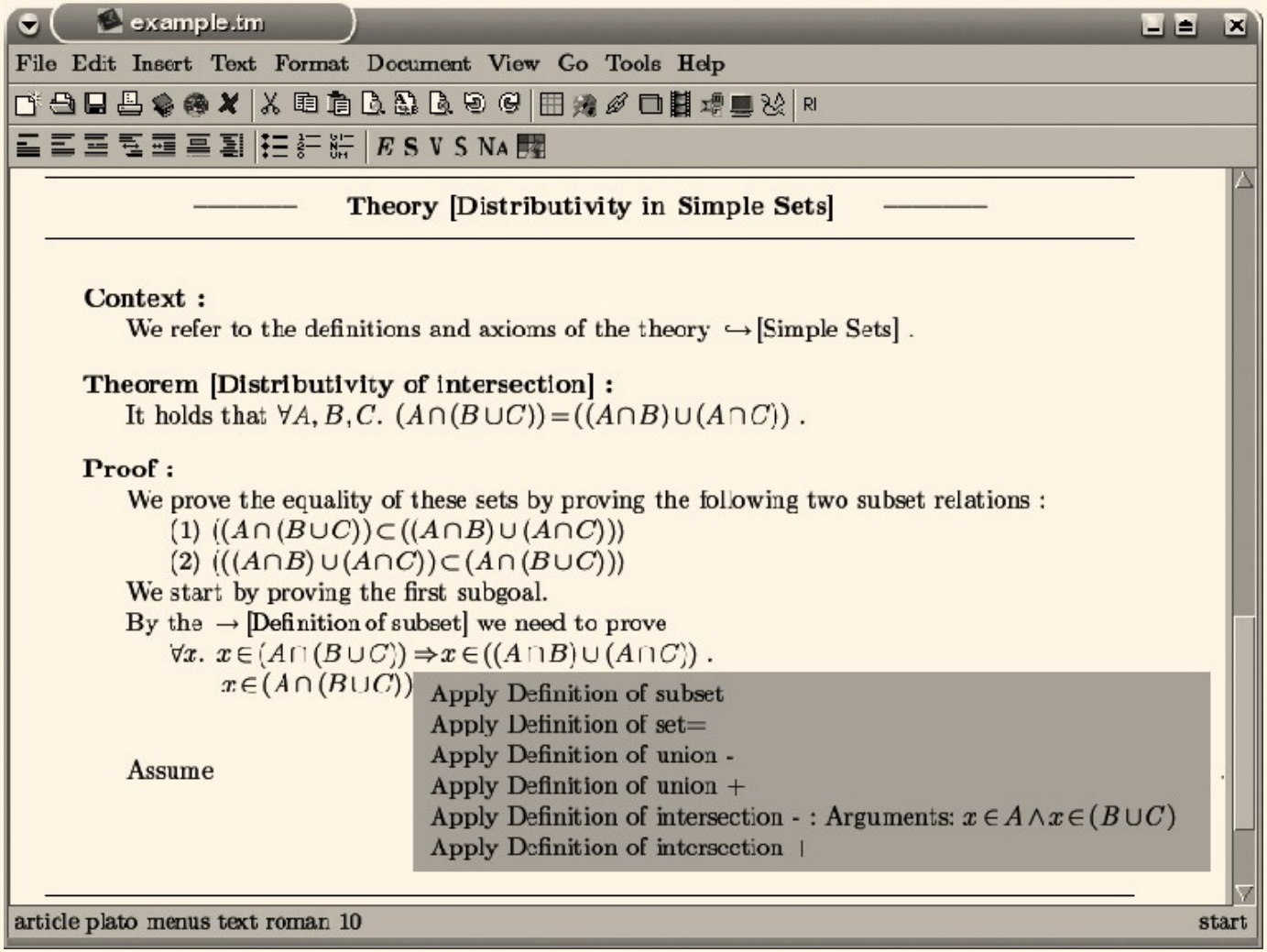
PLATΩ : System Demo



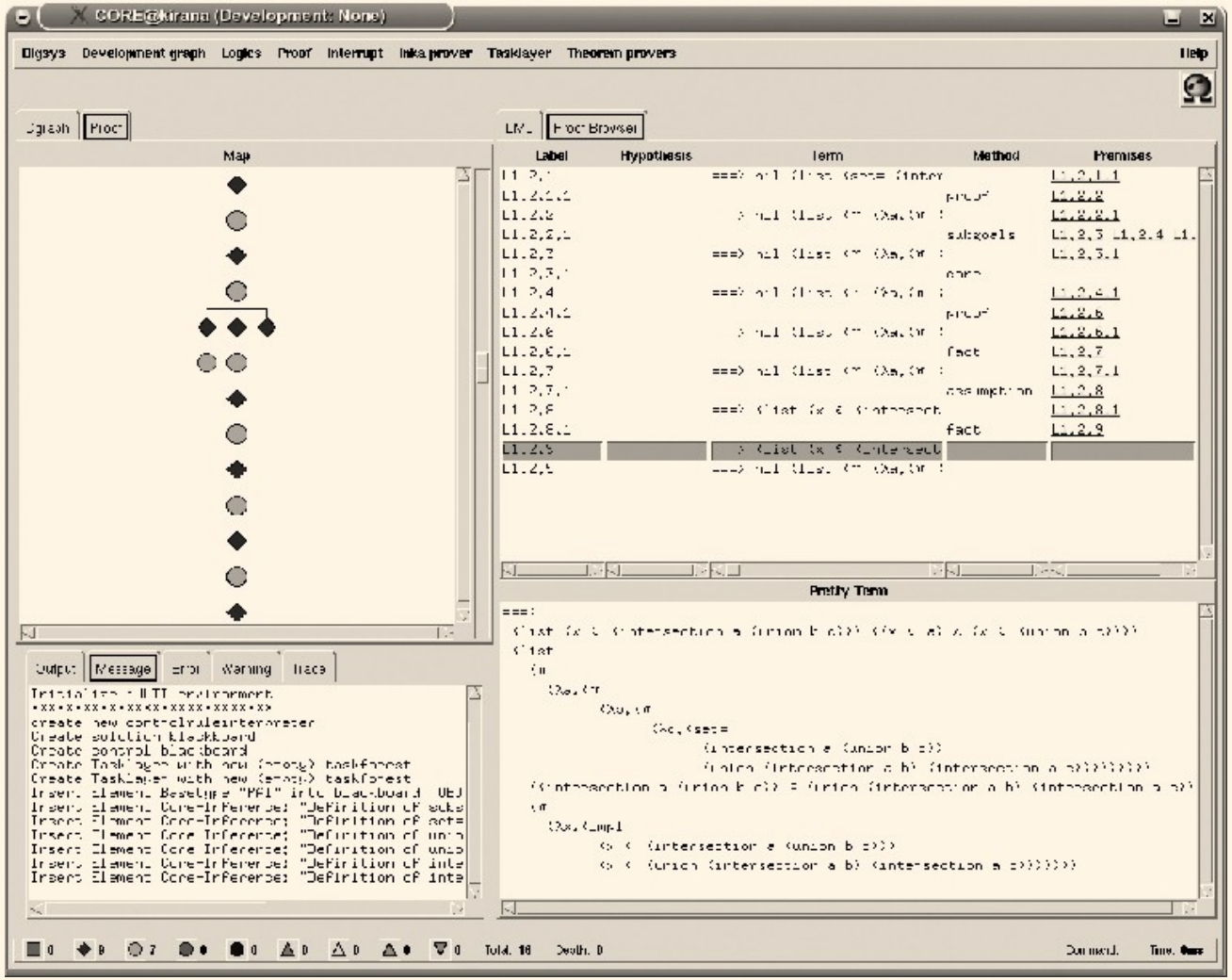
PLATΩ : System Demo



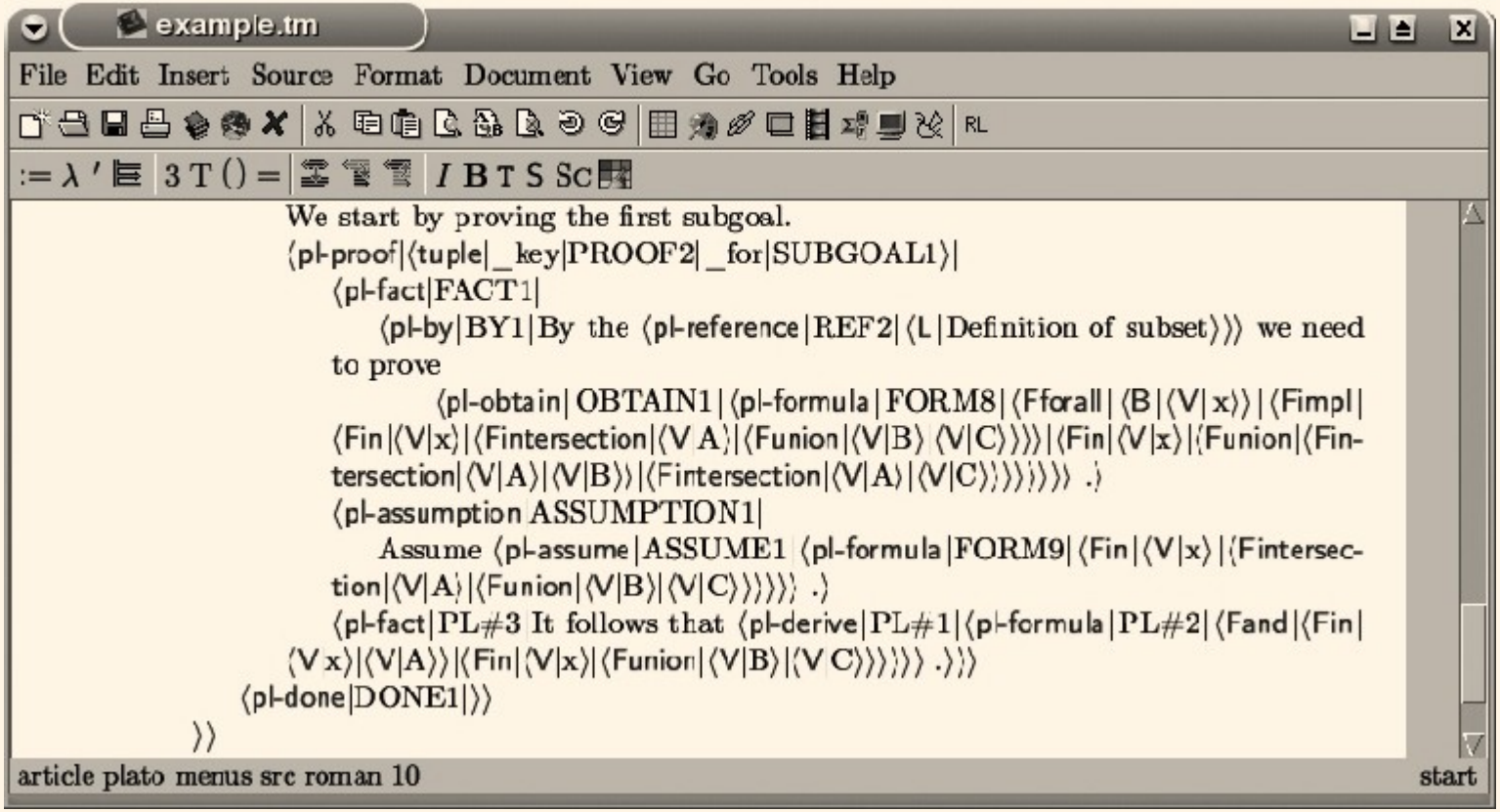
PLATΩ : System Demo



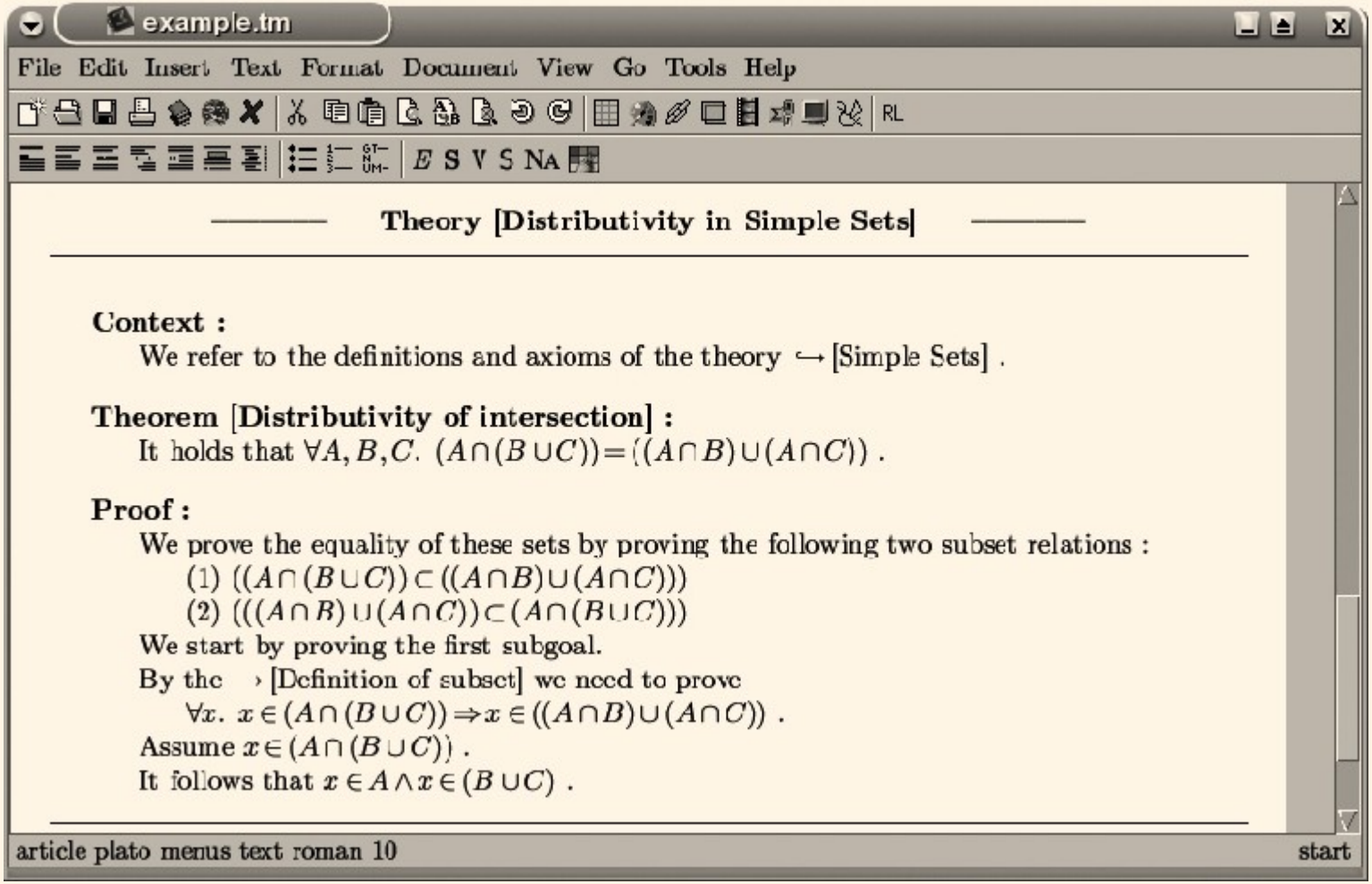
PLATΩ : System Demo



PLATΩ : System Demo



PLATΩ : System Demo



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 processsing technique to generate proof descriptions

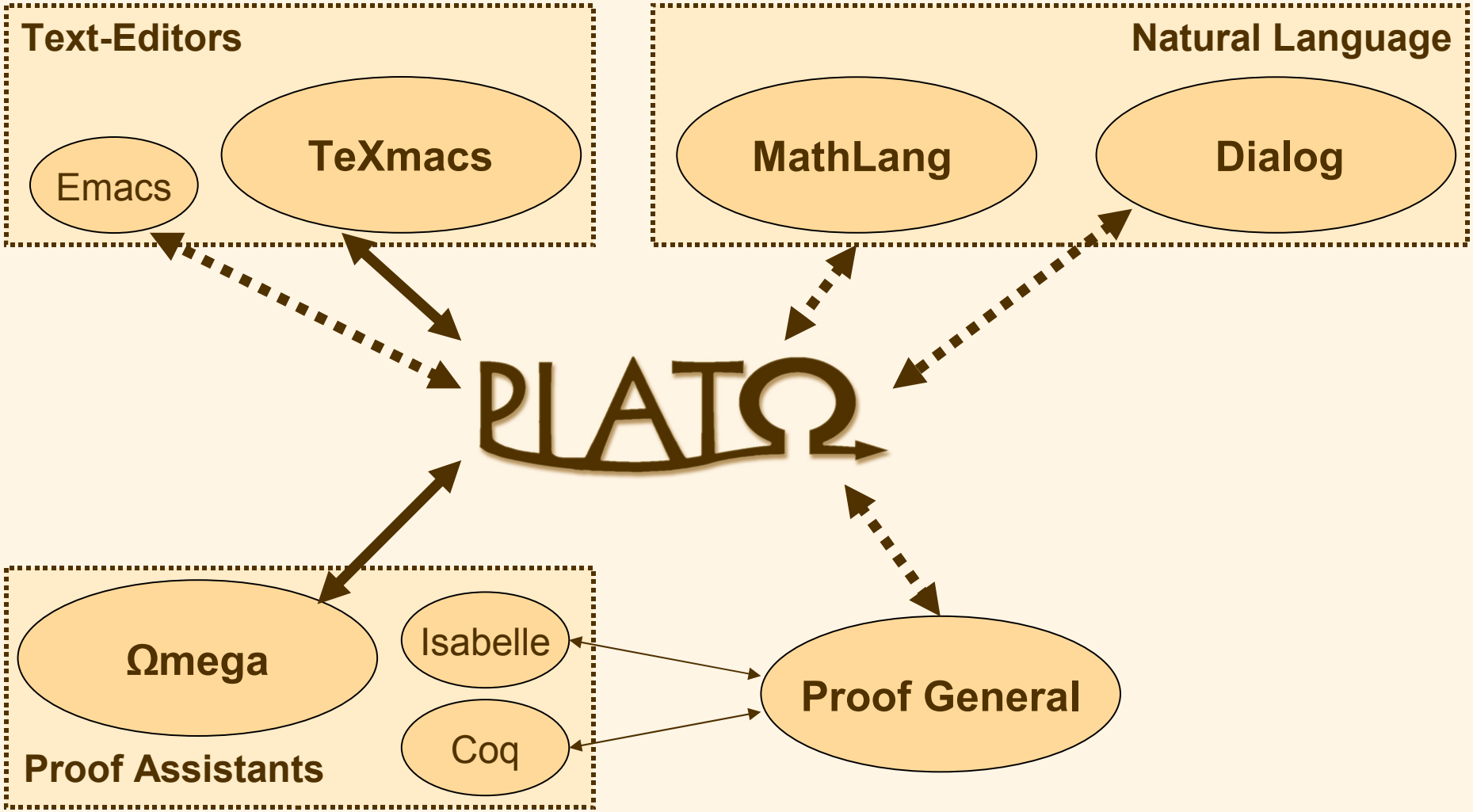
Related Work (cont'd)

- 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

- Interactive Mathematical Authoring with PLATΩ
  - Generic mediation between text-editors and PAs
  - Integrated support in text-editors as local plugin or online webservice
  - Parametric semantic annotation language
  - Automatic generation of the formal representation for the proof assistance system
  - Propagation of changes for efficiency and preservation of partial verifications
  - Context-sensitive service interaction
  - Maintenance of consistent formal and informal representations

Future Work


Text-Editors
Emacs
TeXmacs
Natural Language
MathLang
Dialog
PLATΩ
Ωmega
Isabelle
Coq
Proof Assistants
Proof General