The New Proof Assistant ΩMEGA: Developments and Applications

Serge Autexier

(joint work with the ΩMEGA Group*)

serge@ags.uni-sb.de

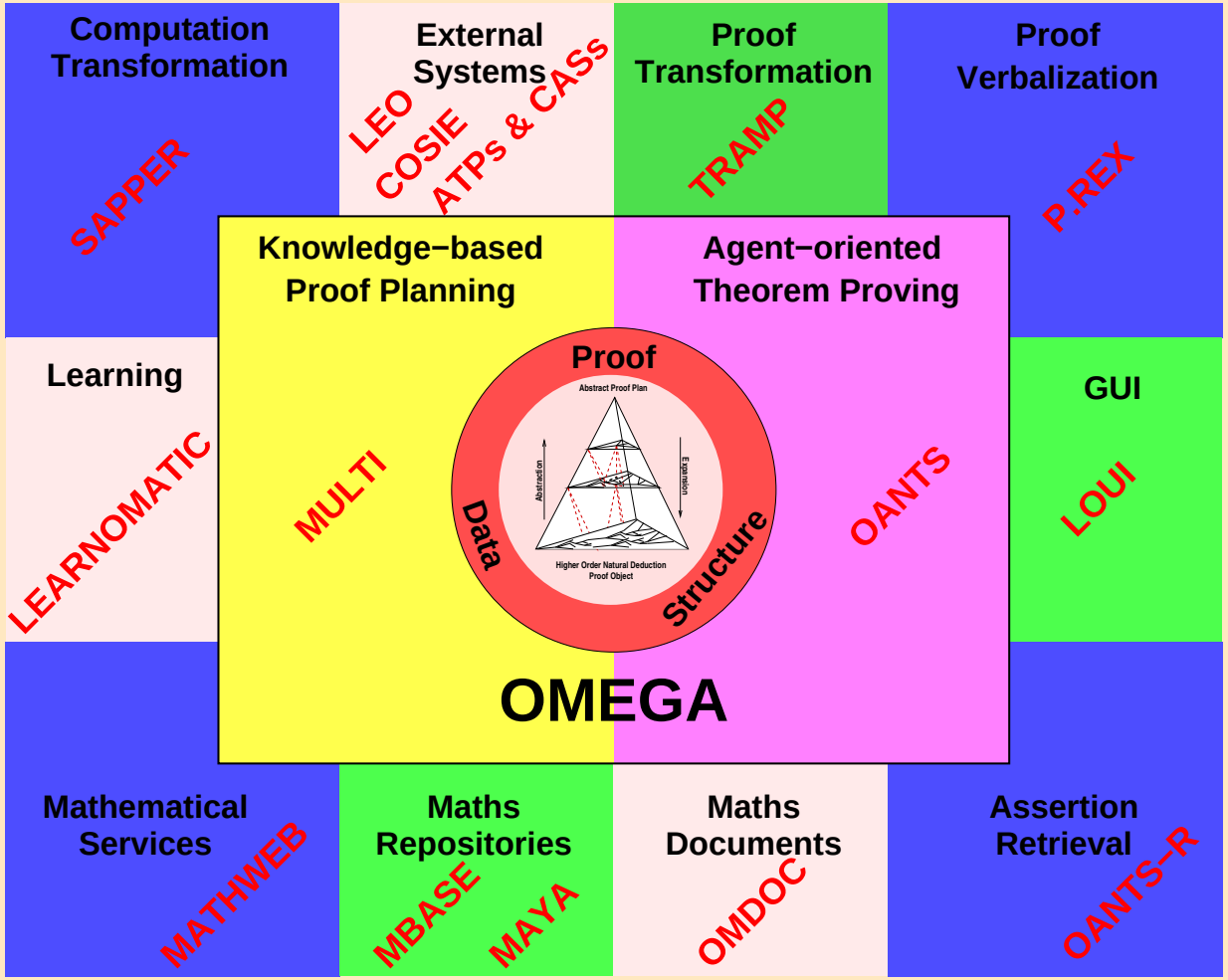
DFKI GmbH & CS Department, Saarland University, Saarbrücken, Germany
& CISA, University of Edinburgh, Scotland (May-July)

STP 2006, June 6, 2006

Glasgow, Scotland

* Jörg Siekmann, Christoph Benzmüller, Chad E. Brown, Mark Buckley, Dominik Dietrich, Armin Fiedler (part-time), Andreas Franke, Henri Lesourd, Marvin Schiller, Frank Theiß, Marc Wagner, Jürgen Zimmer

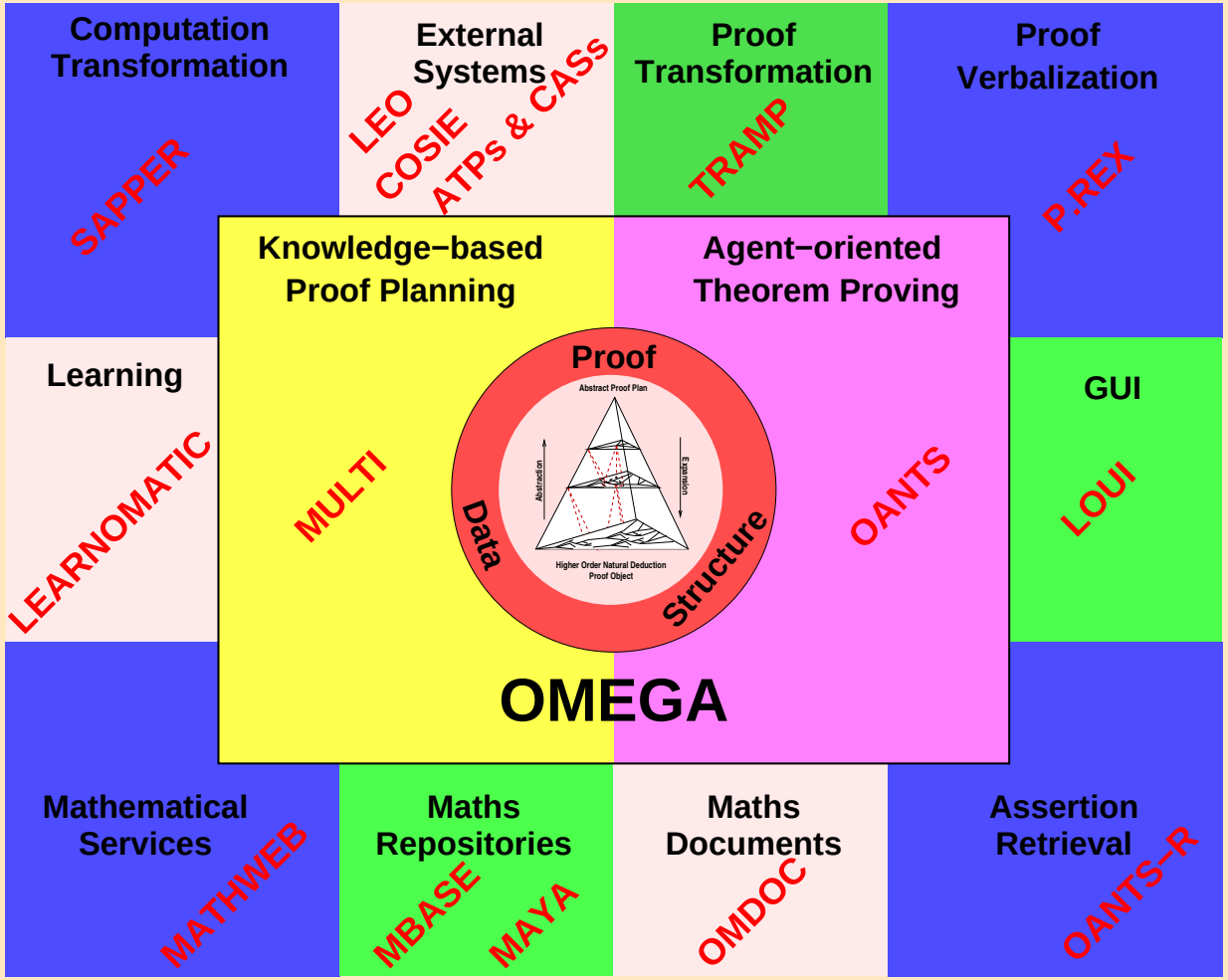
Old ΩMEGA



Source: Autexier

- Proof-Planning (MULTI, E. Melis, A. Meier)
- Agent-based Theorem Proving (ΩANTS, V. Sorge)
- NL Verbalisation of Proofs (*P.rex*, A. Fiedler)
- External systems (ATP, CAS) (MATHWEB-SB J. Zimmer)
- Proof transformation (TRAMP, A. Meier)
- Math. Knowledge base (MBASE A. Franke)
- OMDOC (M. Kohlhase)

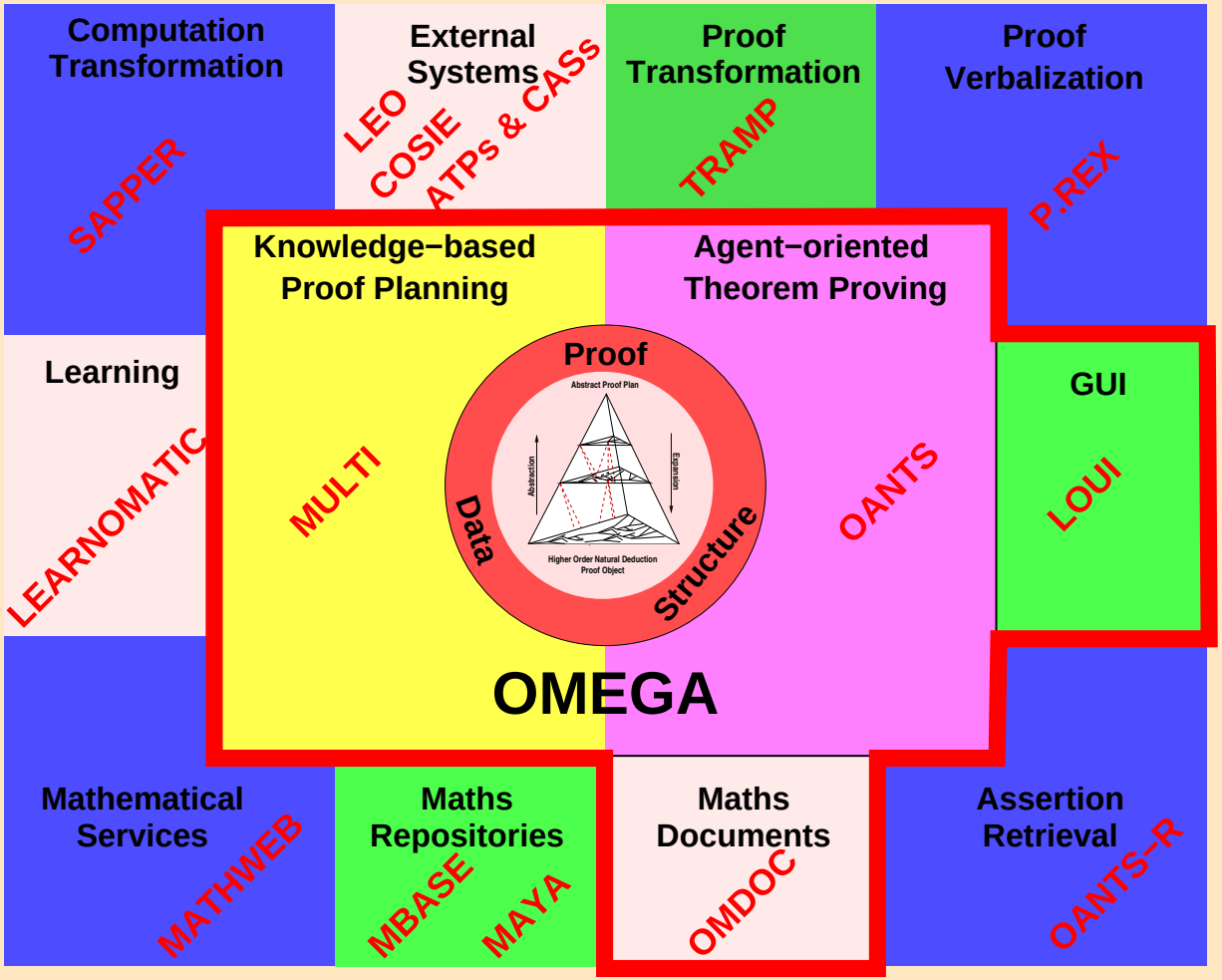
Old ΩMEGA



- Base Calculus: simply typed HOL, Natural Deduction
- Hierarchical proof datastructure (PDS) tailored to
  - Base ND calculus
  - Proof planning methods
  - Tactics
  - Heavily overcrowded
- Graphical user interface $\mathcal{L}\Omega\mathcal{UI}$

Source: Autexier

Old ΩMEGA



- Base Calculus: simply typed HOL, Natural Deduction
- Hierarchical proof datastructure (PDS) tailored to
  - Base ND calculus
  - Proof planning methods
  - Tactics
  - Heavily overcrowded
- Graphical user interface $\mathcal{L}\Omega\mathcal{UI}$

Source: Autexier

Why and What did we Change?

- Natural Deduction Calculus too cumbersome (not natural enough)

Source: Autexier

Why and What did we Change?

- Natural Deduction Calculus too cumbersome (not natural enough)
  - Want direct *reasoning with the assertions* (interactive & automatic)

    $\Rightarrow$ CoRe-calculus that directly supports that [Autexier, CADE'05]

Source: Autexier

Why and What did we Change?

- Natural Deduction Calculus too cumbersome (not natural enough)
  - Want direct *reasoning with the assertions* (interactive & automatic)

    $\Rightarrow$ CoRe-calculus that directly supports that [Autexier, CADE'05]
- The proof datastructure was completely overcrowded

Source: Autexier

Why and What did we Change?

- Natural Deduction Calculus too cumbersome (not natural enough)
  - Want direct *reasoning with the assertions* (interactive & automatic)

    $\Rightarrow$ CoRe-calculus that directly supports that [Autexier, CADE'05]
- The proof datastructure was completely overcrowded
  - New generic proof datastructure (PDS)

Source: Autexier

Why and What did we Change?

- Natural Deduction Calculus too cumbersome (not natural enough)
  - Want direct *reasoning with the assertions* (interactive & automatic)

    $\Rightarrow$ CoRe-calculus that directly supports that [Autexier, CADE'05]
- The proof datastructure was completely overcrowded
  - New generic proof datastructure (PDS)
  - Definition of the *task-layer* as uniform reasoning platform

Source: Autexier

Why and What did we Change?

- Natural Deduction Calculus too cumbersome (not natural enough)
  - Want direct *reasoning with the assertions* (interactive & automatic)

    $\Rightarrow$ CoRe-calculus that directly supports that [Autexier, CADE'05]
- The proof datastructure was completely overcrowded
  - New generic proof datastructure (PDS)
  - Definition of the *task-layer* as uniform reasoning platform
- GUI $\mathcal{L}\Omega\mathcal{UI}$ nice, but did not meet requirements of targeted users, say mathematicians

Source: Autexier

Why and What did we Change?

- Natural Deduction Calculus too cumbersome (not natural enough)
  - Want direct *reasoning with the assertions* (interactive & automatic)

    $\Rightarrow$ CoRe-calculus that directly supports that [Autexier, CADE'05]
- The proof datastructure was completely overcrowded
  - New generic proof datastructure (PDS)
  - Definition of the *task-layer* as uniform reasoning platform
- GUI $\mathcal{L}\Omega\mathcal{UI}$ nice, but did not meet requirements of targeted users, say mathematicians
  - Plug ΩMEGA as reasoning service provider into tools anyway used by users, e.g. WYSIWYG text-editors like TeXmacs (analogy: grammar checkers)

Source: Autexier

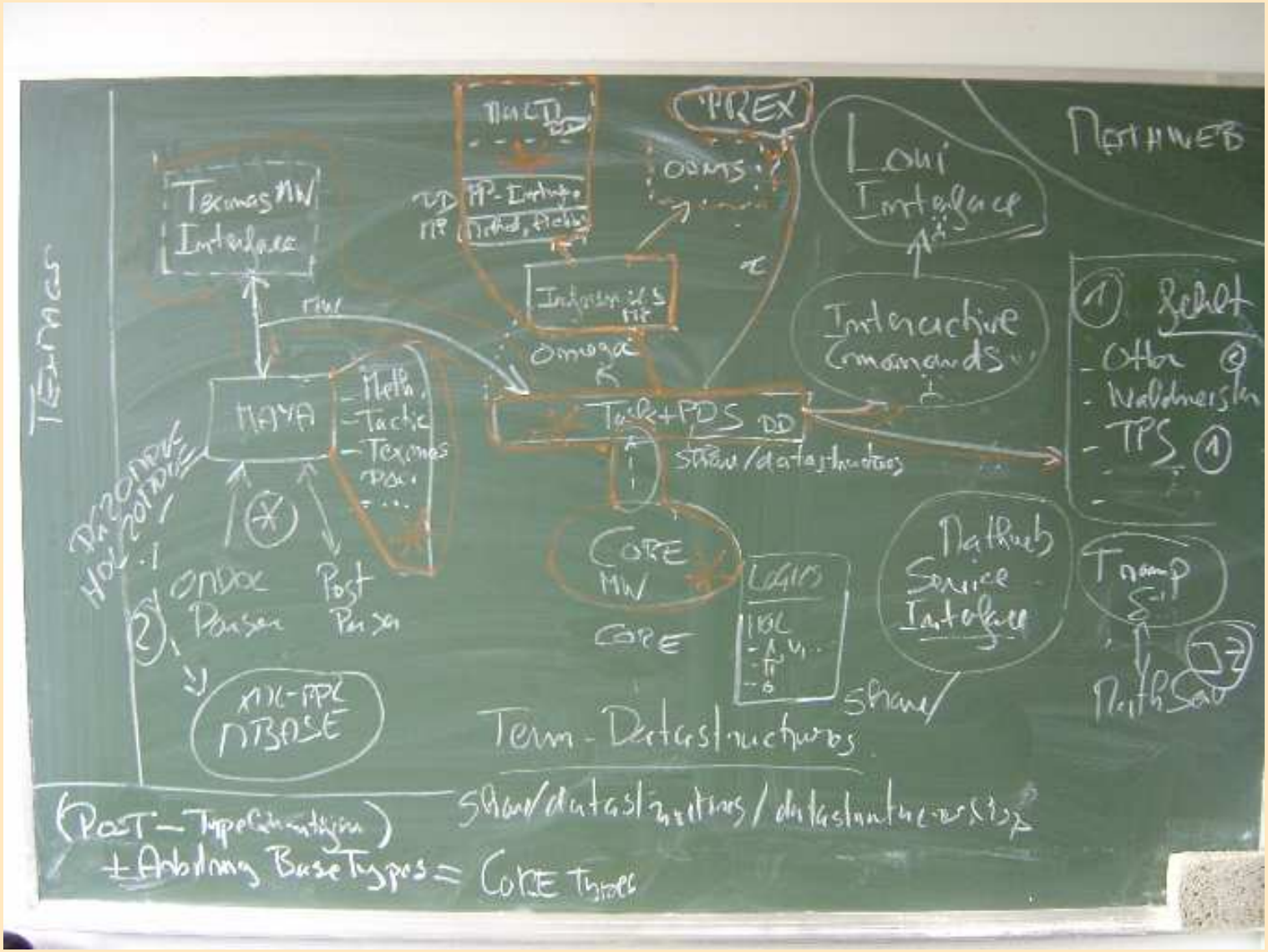
Why and What did we Change?

- Natural Deduction Calculus too cumbersome (not natural enough)
  - Want direct *reasoning with the assertions* (interactive & automatic)

    $\Rightarrow$ CoRe-calculus that directly supports that [Autexier, CADE'05]
- The proof datastructure was completely overcrowded
  - New generic proof datastructure (PDS)
  - Definition of the *task-layer* as uniform reasoning platform
- GUI $\mathcal{L}\Omega\mathcal{UI}$ nice, but did not meet requirements of targeted users, say mathematicians
  - Plug ΩMEGA as reasoning service provider into tools anyway used by users, e.g. WYSIWYG text-editors like TEXMACS (analogy: grammar checkers)
- Add development graph manager MAYA to efficiently maintain formal theories and proofs

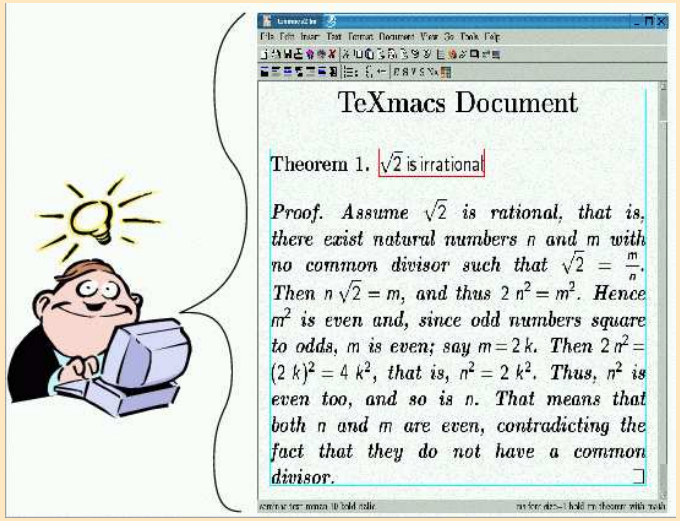
Source: Autexier

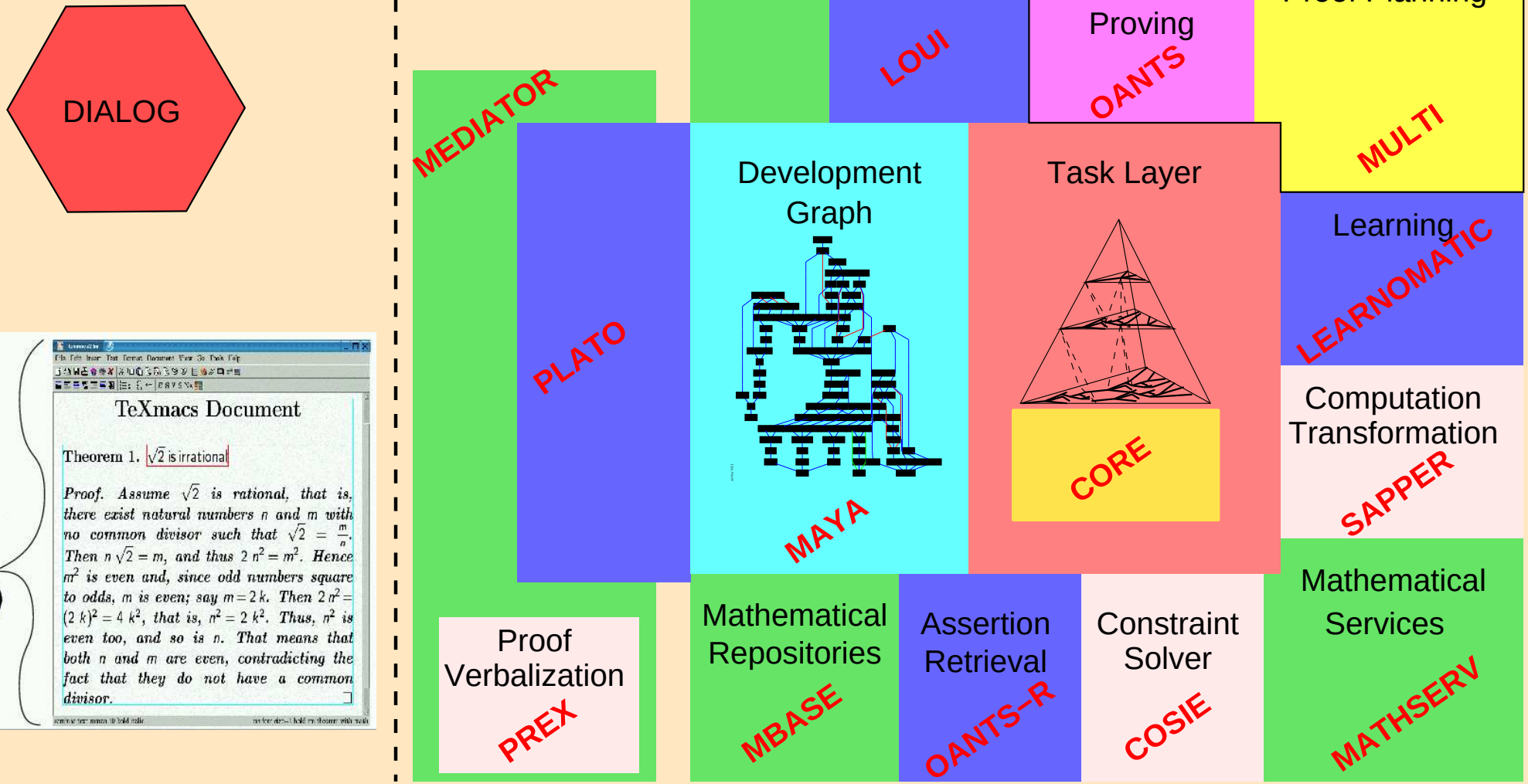
Towards the New ΩMEGA



Source: Autexier

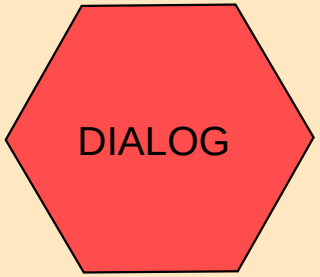
New Architecture

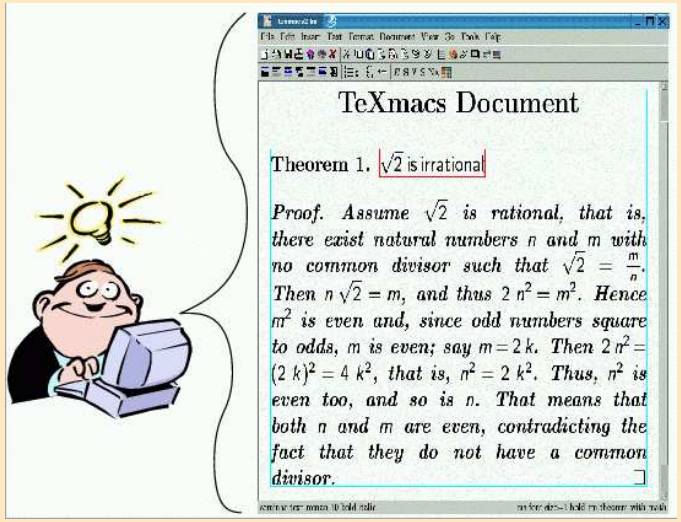


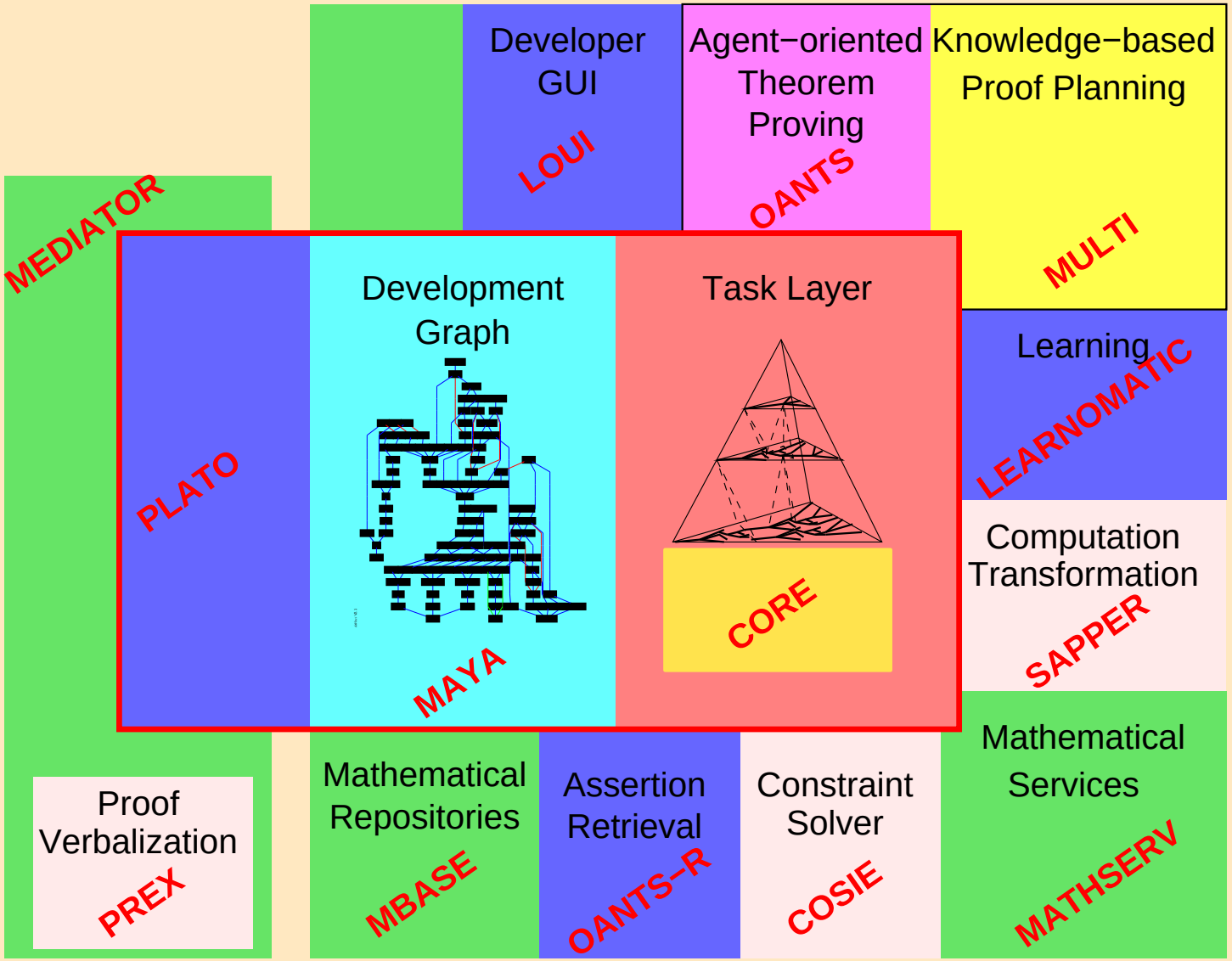


Source: Autexier

This Talk







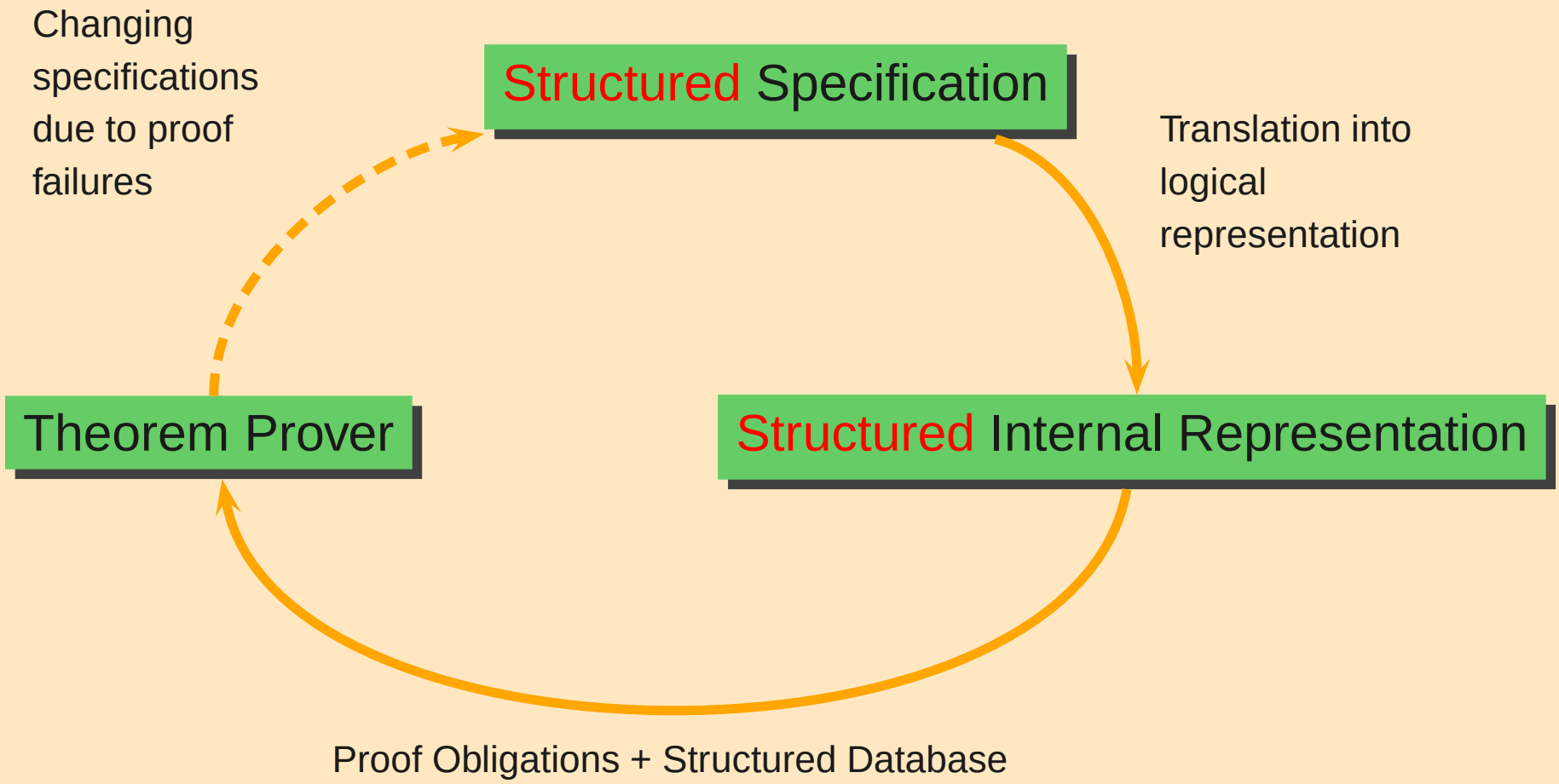
Source: Autexier

Development graphs: Maintaining structured mathematical theories

Source: Autexier

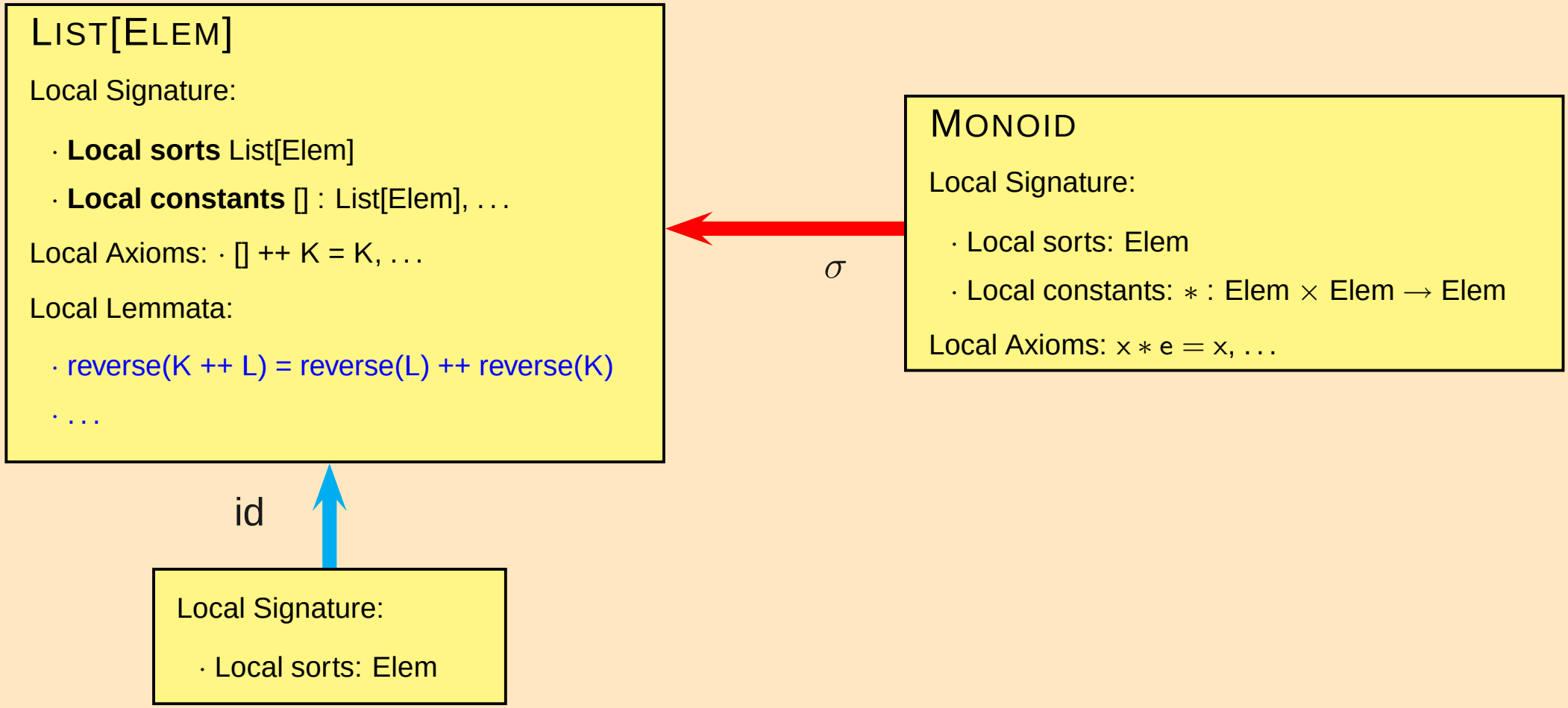
Context

Evolutionary Formal Development



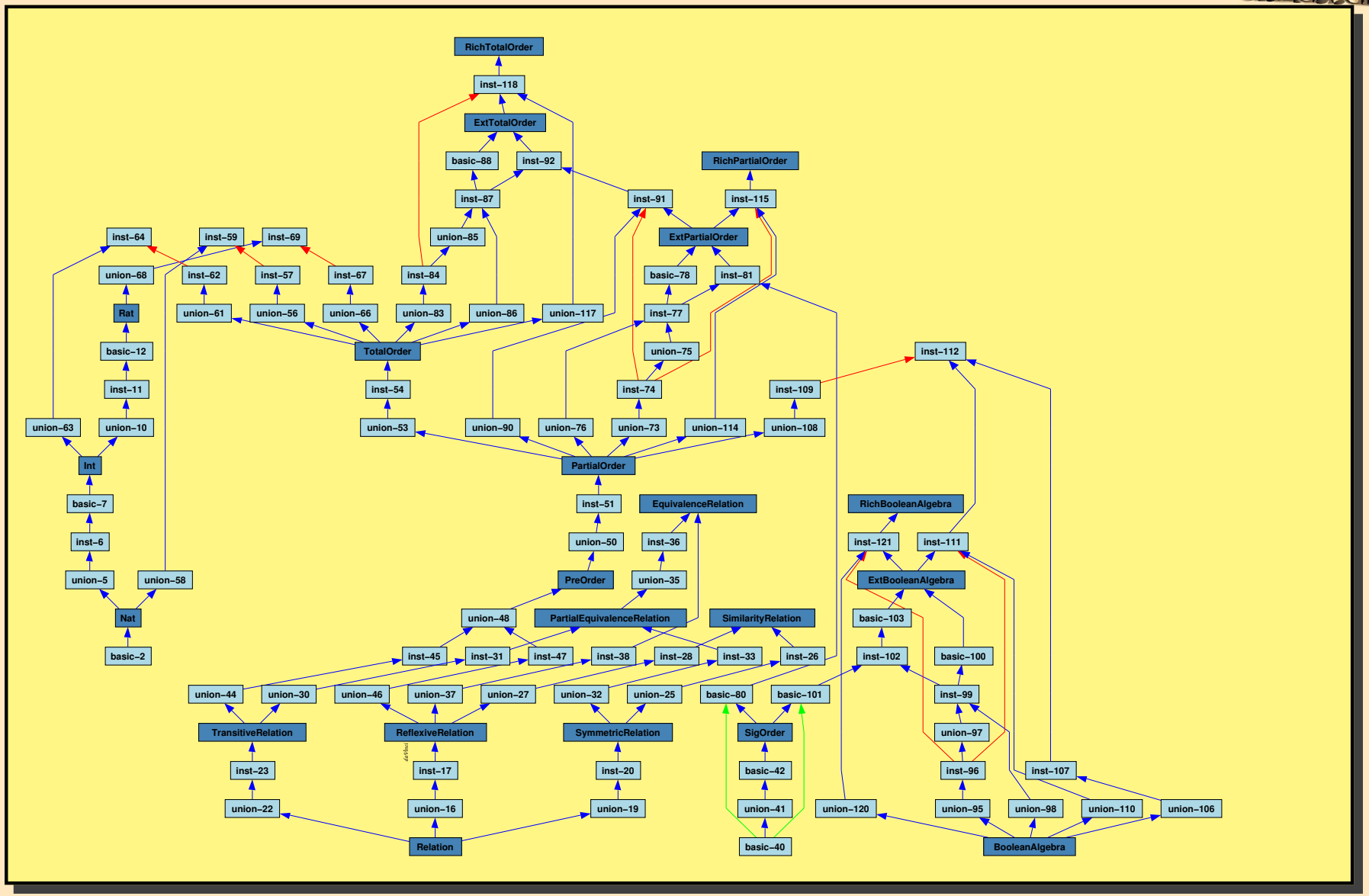
Source: Autexier

Development Graphs



[Hutter'00], [AutexierHutter'02&'05], [MossakowskiAutexierHutter'01&'06]

Source: Autexier



Source: Autexier

Role of Development Graphs

Development Graph	
=	
Structured Logical Content of Specifications	*Structured Specification*
+	
Status of proof obligations	*Verification +*
(pending, proven, used axioms, . . .)	*Management*

- Deals with *evolution* of formal specifications and proofs

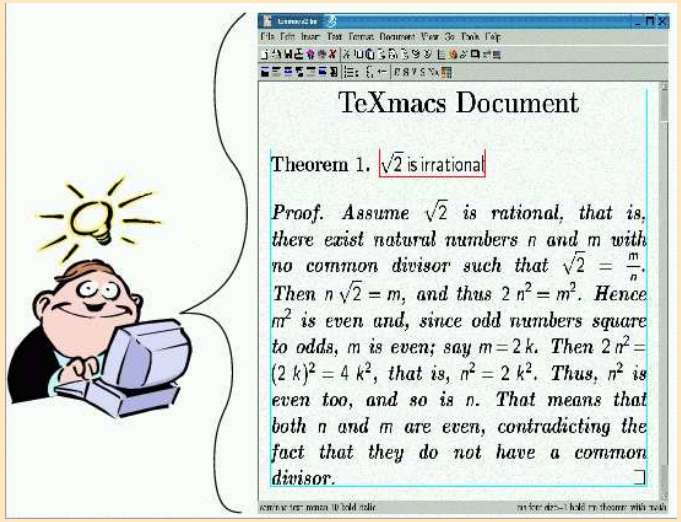
  $\Rightarrow$ Save as much proof work as possible!

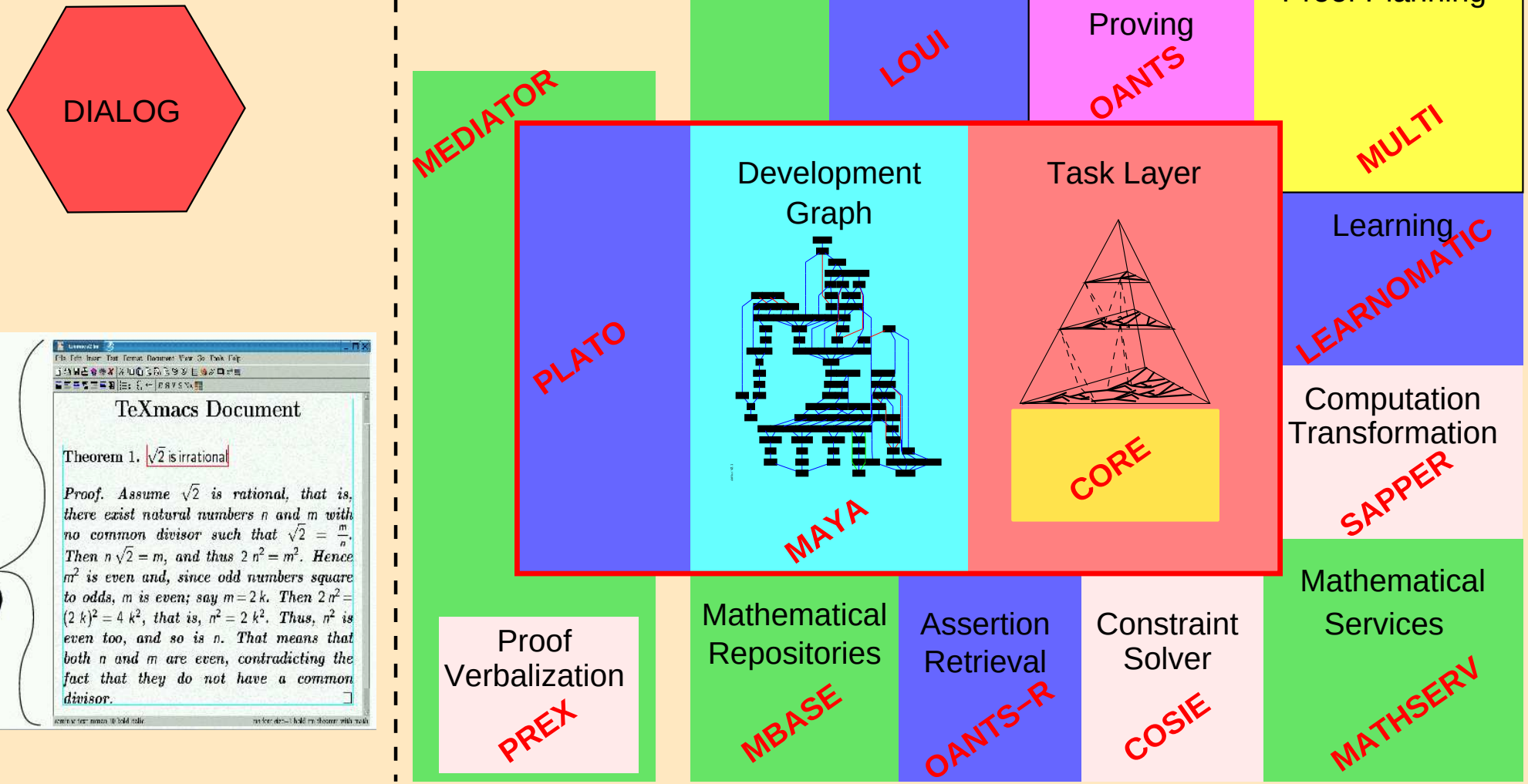
- Exploits graph structure of formal specifications to reduce effects of changes on existing proofs

- Difference analysis of formal specifications to determine fine grained changes

Source: Autexier

This Talk





Source: Autexier

The Task Layer:
Developing and maintaining proofs, proof plans, proof sketches, and alternatives

Source: Autexier

The TASK LAYER at a Glance

The uniform proof construction interface used by both the human user and the automated proof search procedures

- Instance of the new proof datastructure (PDS)
- *Tasks* = Gentzen-style multi-conclusion sequents + focuses of attention on subformulas
- Proof construction steps (task justifications):
  1. introduction of a proof sketch [Wiedijk, TYPES'03]
  2. deep structural rules for weakening and decomposition of subformulas
  3. the application of a lemma that can be postulated on the fly
  4. the application of a CORE calculus rule
  5. the application of an inference ($\Leftarrow$ Proof Planning, ΩANTS)

Source: Autexier

The TASK LAYER at a Glance

The uniform proof construction interface used by both the human user and the automated proof search procedures

- Instance of the new proof datastructure (PDS)
- *Tasks* = Gentzen-style multi-conclusion sequents + focuses of attention on subformulas
- Proof construction steps (task justifications):
  1. introduction of a proof sketch [Wiedijk, TYPES'03]
  2. deep structural rules for weakening and decomposition of subformulas
  3. the application of a lemma that can be postulated on the fly
  4. the application of a CoRe calculus rule
  5. the application of an inference ($\Leftarrow$ Proof Planning, ΩAnts)

Source: Autexier

The new Proof Datastructure (PDS)

Source: Autexier

Proof Data Structure (PDS)

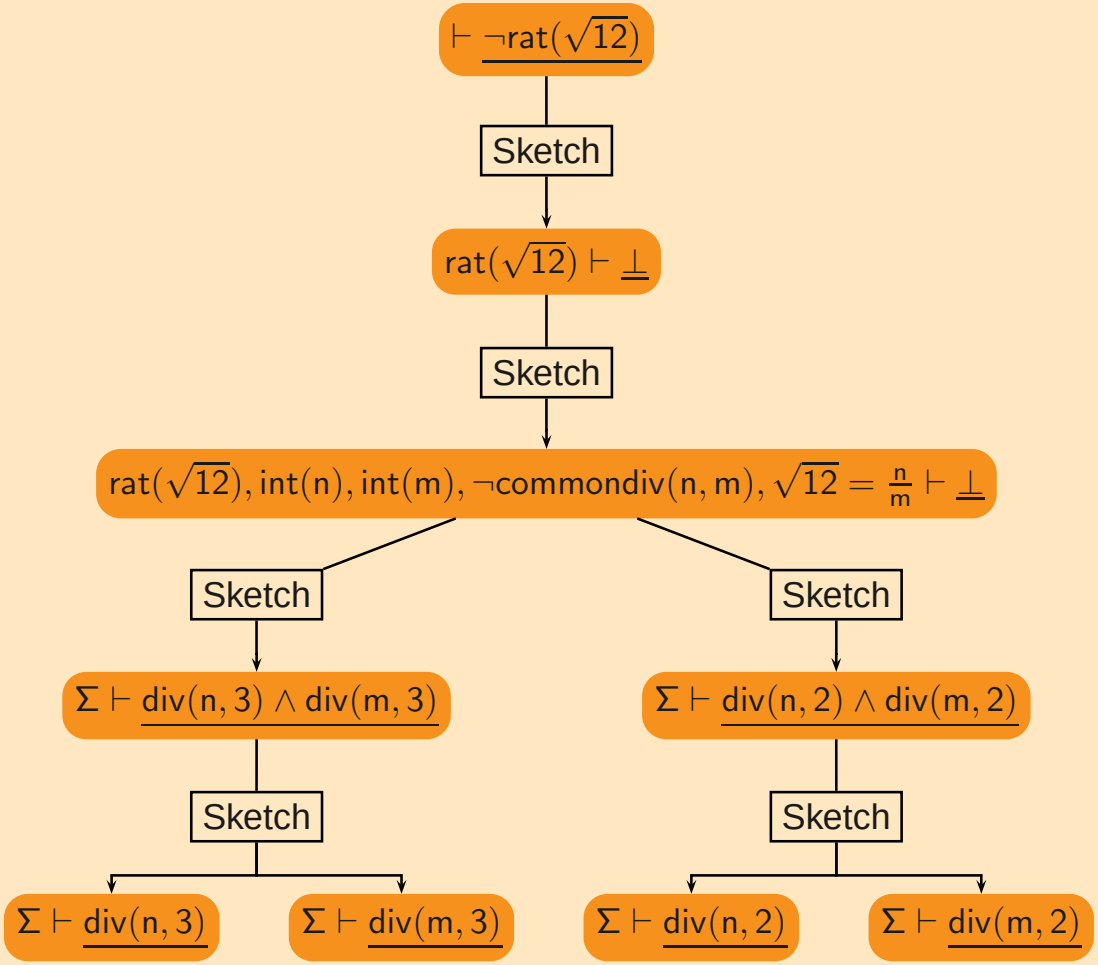
- Proof simultaneously at different levels of granularity
  - Representation of abstract proof ideas and their refinement

    (Proof Planning)
  - Representation of external systems proofs/computations and their refinement
  - Definable level of granularity (slices through the hierarchy) Views
    - Interactive proof development
    - Adaptive natural language proof explanations
  - Allows to postpone verification (expansion) of higher-level proof steps
- Alternative proof attempts (on the same level of granularity)
- Support for lemmatization (forest of PDS trees with links)

[AutexierBenzmüllerDietrichMeierWirth,MKM 2005]

Source: Autexier

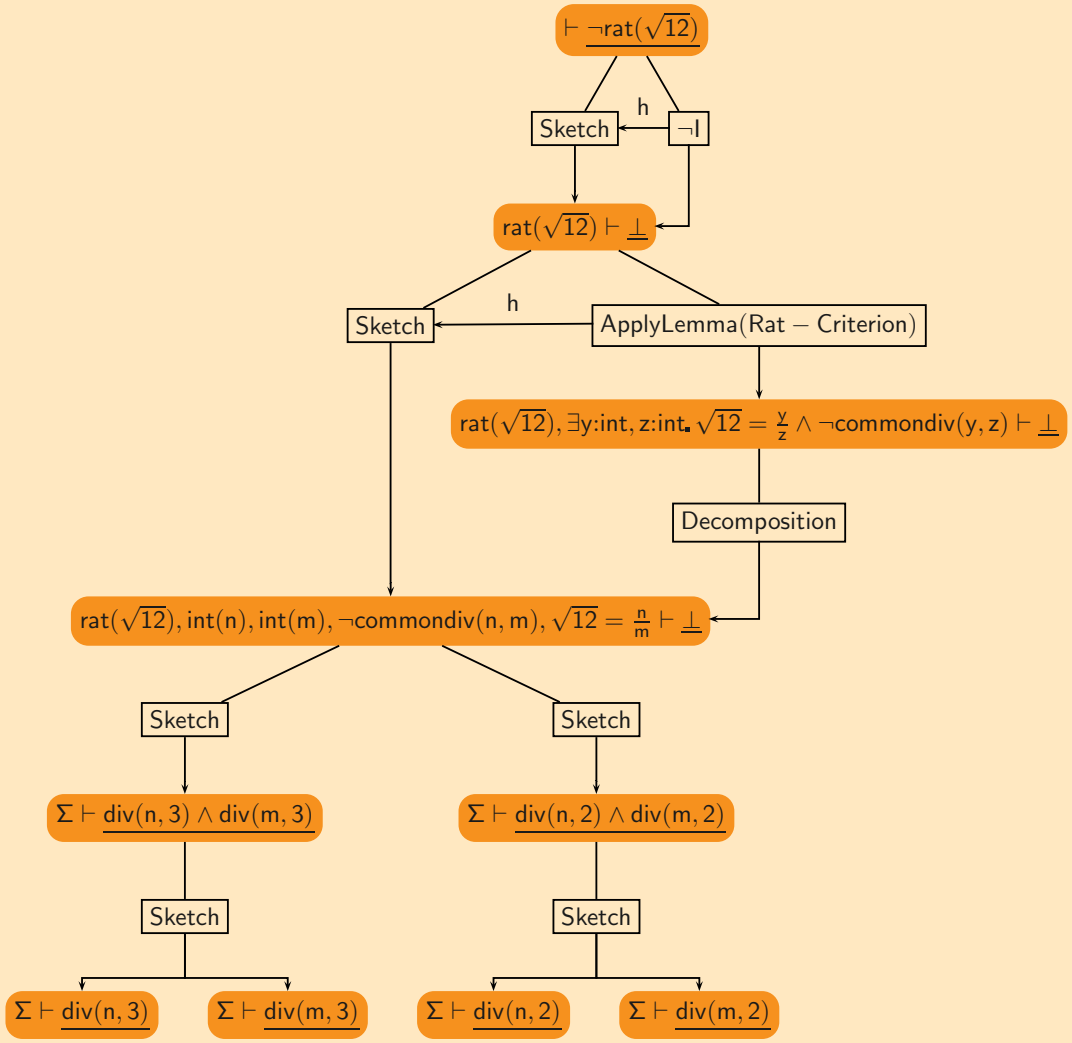
Example Abstract PDS



Source: Autexier

Example Complete PDS

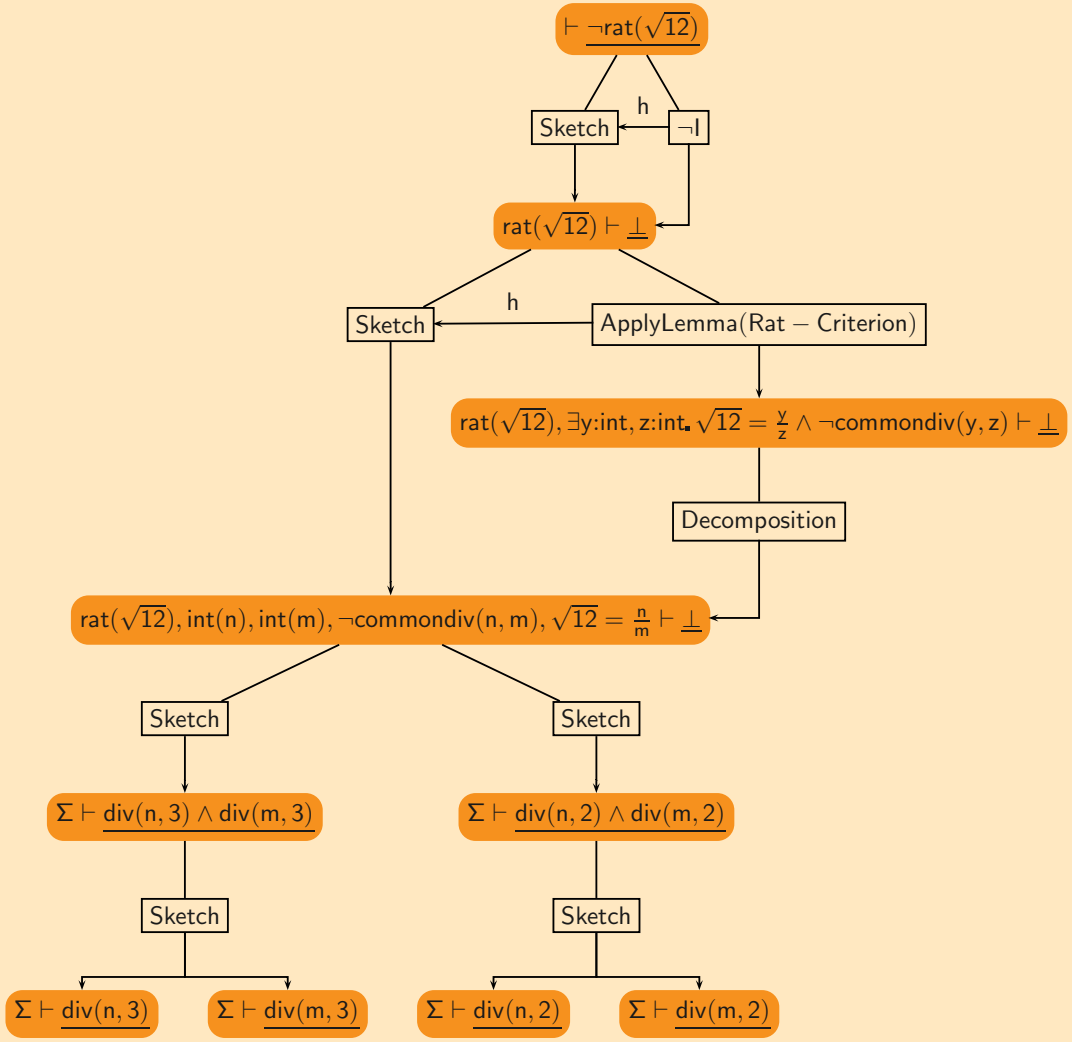
Complete PDS



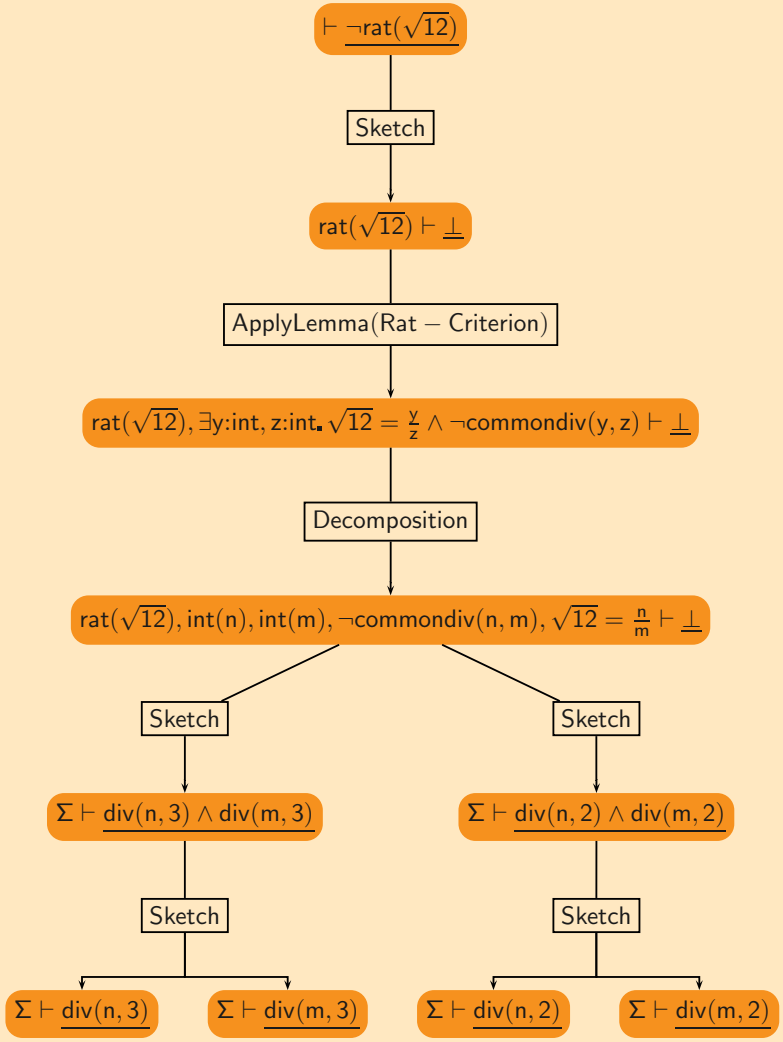
Source: Autexier

Example Complete PDS

Complete PDS



A PDS View



Source: Autexier

PDSs

- Alternative subproofs require to allow for alternative substitutions of Metavariables $\Longrightarrow$ Mechanism to support that efficiently

  [Dietrich, Diploma Thesis, 2006]

- PDS provided basis to use $\overline{\lambda}\mu\widetilde{\mu}$-calculus proof terms as semantics for OMDOC proofs extended by alternatives (= PDS)

  [Autexier&Sacerdoti-Coen, MKM'06]

  ($\overline{\lambda}\mu\widetilde{\mu}$-calculus proposed by Curien and Herbelin, Curry-Howard Isomorphism to classical SK, allows call-by-value/call-by-name)

Source: Autexier

Proof Construction by Inference Application

Source: Autexier

From Tactics and Methods. . .
. . . to Inferences

In the old ΩMEGA system there were

- (ΩMEGA-)tactics (interactive theorem proving)
- methods (proof planning)

For each of them, again *each application direction (AD)* had to be specified manually:

$$\frac{P_1 : F \qquad P_2 : U = V}{C : G} \text{=}\textit{subst-m-fw}(\pi)$$

Application Condition: –

Outline functions:

$\langle C \quad \texttt{compute-=subst-prem}(P_1, P_2, \pi)\rangle$

$$\frac{P_1 : F \qquad P_2 : U = V}{C : G} \text{=}\textit{subst-m-bw1}(\pi)$$

Application Condition: –

Outline functions:

$\langle P_1 \quad \texttt{compute-=subst-prem}(C, P_2, \pi)\rangle$

- Having P_1 and P_2, we can get C (fw)
- Having C and P_2, we can get P_1 (bw1)
- Having C and P_1, we can get P_2 (bw2)
- Having P_1, P_2 and C, we check the proof step

Source: Autexier

The same as a single Inference...

$$\frac{P_1 : F \qquad P_2 : U = V}{C : G} \textit{=subst-m}(\pi)$$

Application Condition: $-$

Outline functions:

$\langle C \quad \texttt{compute-=subst-prem}(P_1, P_2, \pi)\rangle$

$\langle P_1 \quad \texttt{compute-=subst-prem}(C, P_2, \pi)\rangle$

There were 4 tactics and 2 methods in the old ΩMEGA system!

- Inferences unify (ΩMEGA-)tactics and methods, overcome manual specification of ADs

Source: Autexier

Examples of Inferences

$$\frac{P_1 : A \subseteq B \qquad P_2 : B \subseteq C}{C : A \subseteq C}$$

Application Condition: $-$

Source: Autexier

Examples of Inferences

$$\frac{\begin{array}{c}[\epsilon > 0, D > 0, 0 <| x - A |, | x - A |< D] \\ \vdots \\ P :| F(x) - L |< \epsilon\end{array}}{C : \lim_{A} F = L}\ \mathit{Limit}(\epsilon, x)$$

Application Condition: $\mathsf{EV}(\epsilon, \{F, A, L\}) \wedge \mathsf{EV}(x, \{F, A, L, D\})$

Source: Autexier

Examples of Inferences

$$\frac{P : O(U, V)}{C : O(U', V')} \textit{Maple-Simplify}$$

Application Condition: $O \in \{=, \leq, <, \geq, >\}$

Outline functions: $\langle U \;\; \texttt{maple-simplify}(U') \rangle$

$\langle U' \;\; \texttt{maple-simplify}(U) \rangle$

$\langle V \;\; \texttt{maple-simplify}(V') \rangle$

$\langle V' \;\; \texttt{maple-simplify}(V) \rangle$

Source: Autexier

General Form of Inferences

$$\frac{\begin{array}{ccc} [H_1^1, \ldots, H_{n_1}^1] & & [H_1^k, \ldots, H_{n_k}^k] \\ \vdots & & \vdots \\ P_1 : F_1 & \ldots & P_k : F_k \end{array}}{C_1 : G_1 \ \ldots \ C_m : G_m} \; \mathit{Name}(\omega_1, \ldots, \omega_n)$$

Outline Functions: $\langle i^1, f^1(i_1^1, \ldots, i_{j_1}^1)\rangle$
$\vdots$
$\langle i^l, f^l(i_1^l, \ldots, i_{j_l}^l)\rangle$

Application Conditions: $P(i_1^0, \ldots, i_m^0)$

Source: Autexier

General Form of Inferences

$$\frac{\begin{array}{c}[H_1^1,\ldots,H_{n_1}^1] \quad\quad [H_1^k,\ldots,H_{n_k}^k] \\ \vdots \quad\quad\quad\quad \vdots \\ P_1 : F_1 \quad \ldots \quad P_k : F_k\end{array}}{C_1 : G_1 \ \ldots \ C_m : G_m}\ \mathit{Name}(\omega_1,\ldots,\omega_n)$$

Outline Functions: $\langle i^1, f^1(i_1^1,\ldots,i_{j_1}^1)\rangle$ $\vdots$ $\langle i^l, f^l(i_1^l,\ldots,i_{j_l}^l)\rangle$

Application Conditions: $P(i_1^0,\ldots,i_m^0)$

- Procedure to *automatically synthesise inferences* from arbitrary formulas contained in the mathematical theories.

$$\forall A, B, C : \mathrm{Set}.(A \subseteq B \wedge B \subseteq C) \Rightarrow (A \subseteq C)$$

$$\forall f, a, l.\forall \epsilon.\epsilon > 0 \Rightarrow \exists \delta.\delta > 0 \Rightarrow \forall x.(0 <| x - a | \wedge | x - a |< \delta) \Rightarrow | f(x) - l |< \epsilon$$

$$\Rightarrow \lim_a f = l$$

(Had *all* to be *encoded manually* in the old ΩMEGA system...)

- Only the other inferences must be manually encoded (e.g. `=subst-m`)

Source: Autexier

Application of Inferences

- Inference are applied on subformulas of a task
- Not all premises and conclusions need to be mapped.
- Premises are mapped to negative positions in a task
- Conclusions are mapped to positive positions in a task
- Example:

$$P \Rightarrow (A \subset B) \vdash Q \Rightarrow (A \subset C)$$

Source: Autexier

Application of Inferences

- Inference are applied on subformulas of a task
- Not all premises and conclusions need to be mapped.
- Premises are mapped to negative positions in a task
- Conclusions are mapped to positive positions in a task
- Example:

$$\mathsf{P} \Rightarrow (\mathsf{A} \subset \mathsf{B}) \vdash \mathsf{Q} \Rightarrow (\mathsf{A} \subset \mathsf{C})$$

$$\frac{P_1 : (A \subset B) \quad P_2 : (B \subset C)}{C : (A \subset C)} \mathsf{Subset}$$

Source: Autexier

Application of Inferences

- Inference are applied on subformulas of a task
- Not all premises and conclusions need to be mapped.
- Premises are mapped to negative positions in a task
- Conclusions are mapped to positive positions in a task
- Example:

$\mathsf{P} \Rightarrow (\mathsf{A} \subset \mathsf{B}) \vdash \mathsf{Q} \Rightarrow (\mathsf{A} \subset \mathsf{C})$

$$\frac{P_1 : (A \subset B) \quad P_2 : (B \subset C)}{C : (A \subset C)}\text{Subset}$$

Suppose we map $\mathsf{P}_1 \mapsto \langle 1, 2 \rangle$ and $\mathsf{C} \mapsto \langle 2, 2 \rangle$ (Hence $\mathsf{P}_2 \rightarrow \mathsf{B} \subset \mathsf{C}$).

Source: Autexier

Application of Inferences

- Inference are applied on subformulas of a task
- Not all premises and conclusions need to be mapped.
- Premises are mapped to negative positions in a task
- Conclusions are mapped to positive positions in a task
- Example:

$$\mathsf{P} \Rightarrow (\mathsf{A} \subset \mathsf{B}) \vdash \mathsf{Q} \Rightarrow (\mathsf{A} \subset \mathsf{C}) \qquad \frac{P_1 : (A \subset B) \quad P_2 : (B \subset C)}{C : (A \subset C)}\text{Subset}$$

Suppose we map $\mathsf{P}_1 \mapsto \langle 1, 2 \rangle$ and $\mathsf{C} \mapsto \langle 2, 2 \rangle$ (Hence $\mathsf{P}_2 \rightarrow \mathsf{B} \subset \mathsf{C}$).

$$\mathsf{P} \Rightarrow (\mathsf{A} \subset \mathsf{B}) \vdash \mathsf{Q} \Rightarrow (\mathsf{P} \wedge (\mathsf{B} \subset \mathsf{C}))$$

Source: Autexier

Interesting Questions

$$\frac{P_1 : F \qquad P_2 : U = V}{C : G} \text{=}\textit{subst-m}(\pi)$$

Application Condition: –

Outline functions:

$\langle C \quad \texttt{compute-=subst-prem}(P_1, P_2, \pi)\rangle$

$\langle P_1 \quad \texttt{compute-=subst-prem}(C, P_2, \pi)\rangle$

- When is an inference applicable?

  As soon as we have matched enough premises and conclusions such that we can compute any missing premise and conclusion using the outline functions

  Application directions = These sets of premises and conclusions
  $\Longrightarrow$ Sufficient information for the proof planner
  $\Longrightarrow$ No need for extra methods as in the old ΩMEGA system

- More enhanced "static analysis" of inferences allows for automatic generation of inference-specific agents required by ΩANTS
  $\Longrightarrow$ Overcomes manual definition of the agents as in the old ΩMEGA system

  [Autexier&Dietrich, MKM'06]

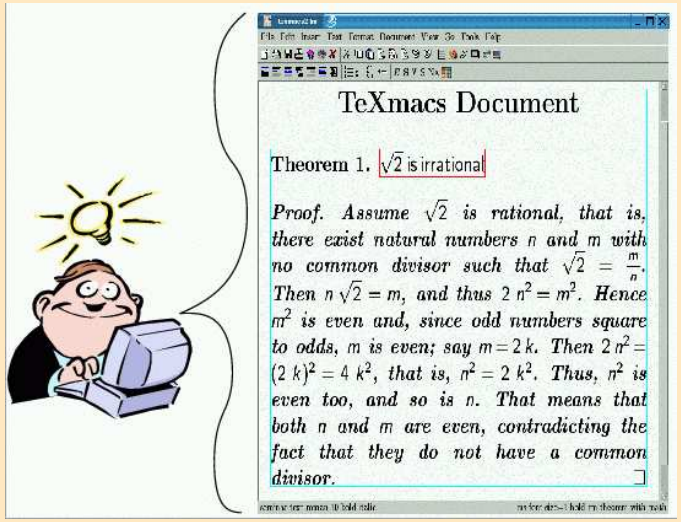
Source: Autexier

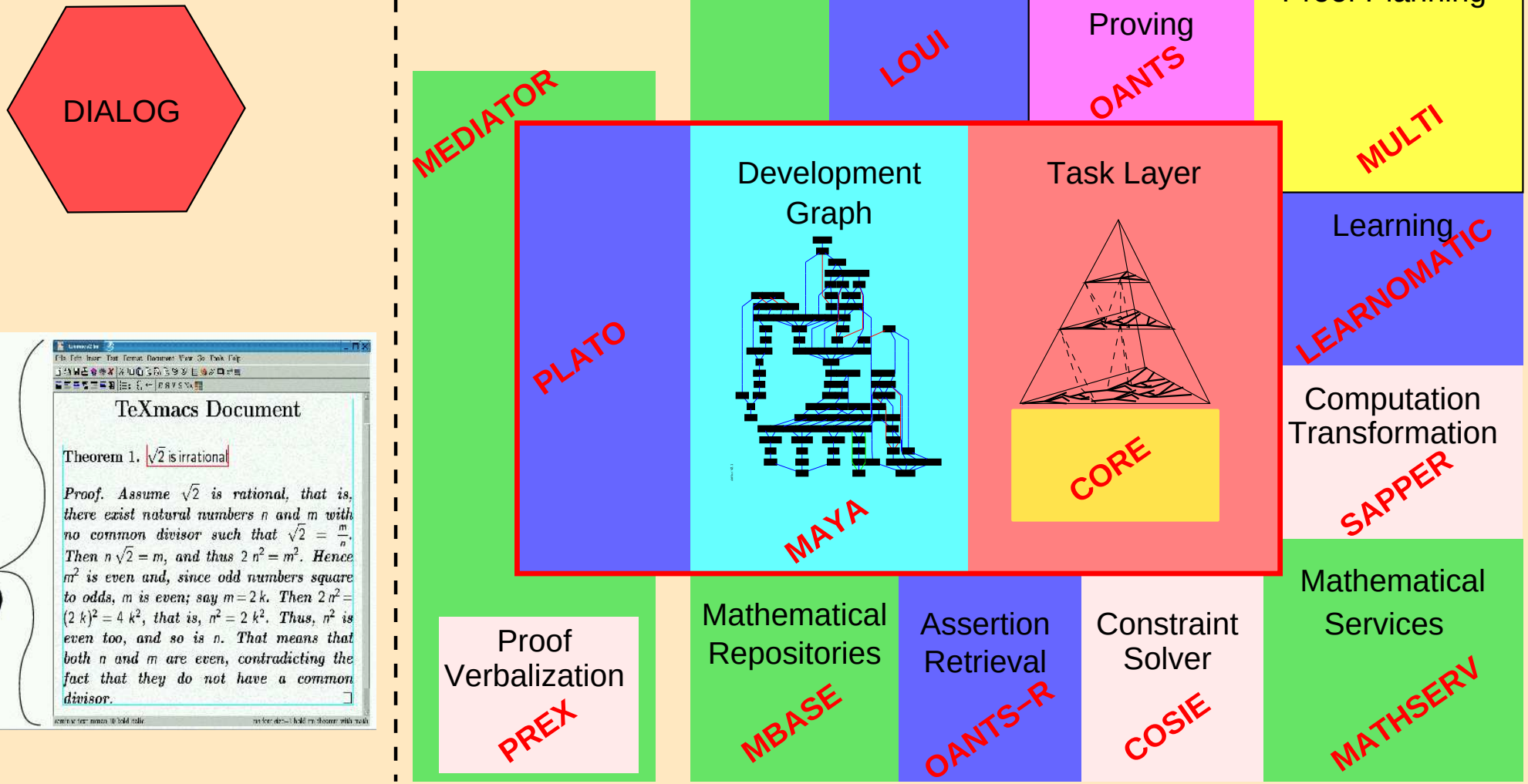
Next Steps

- Automation of proof search at the TASK LAYER
  - Reactivate ΩANTS on that basis
  - Reimplement proof planner MULTI
- Further evaluation:
  - Coding effort is already drastically reduced
  - Less inferences required by deep application of inferences
  - Benefit for automated proof search?
- Specific services required to support proof development in TEXMACS via PLATΩ

Source: Autexier

This Talk





Source: Autexier

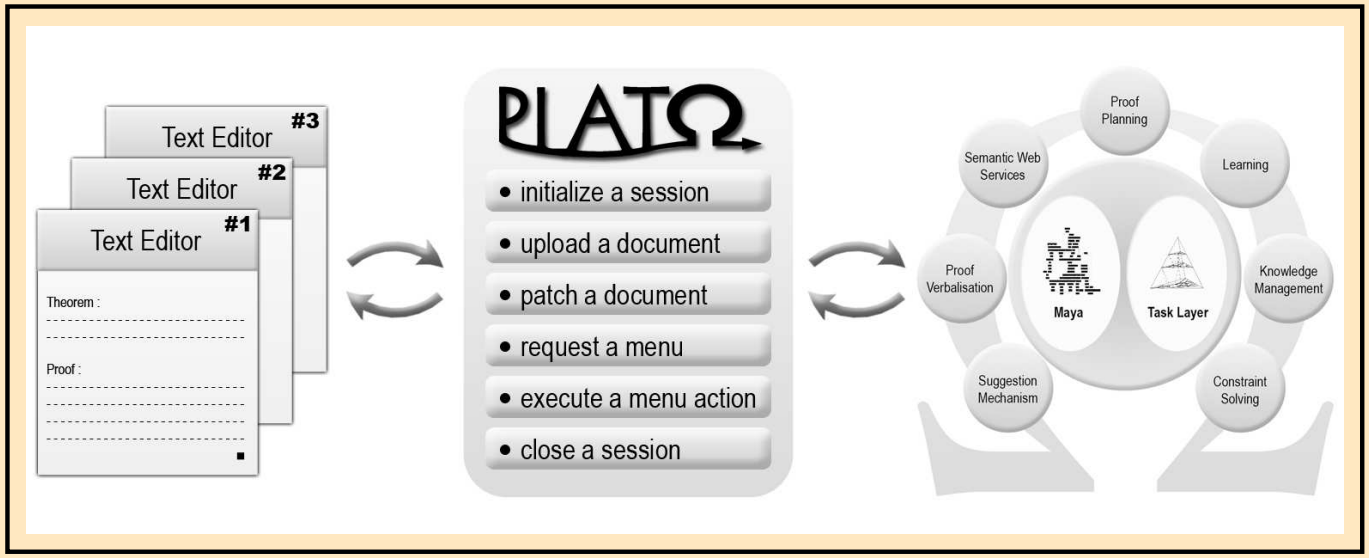
Mediating between Texteditors and Proof Assistance Systems

–

The PLATΩ System

Source: Autexier

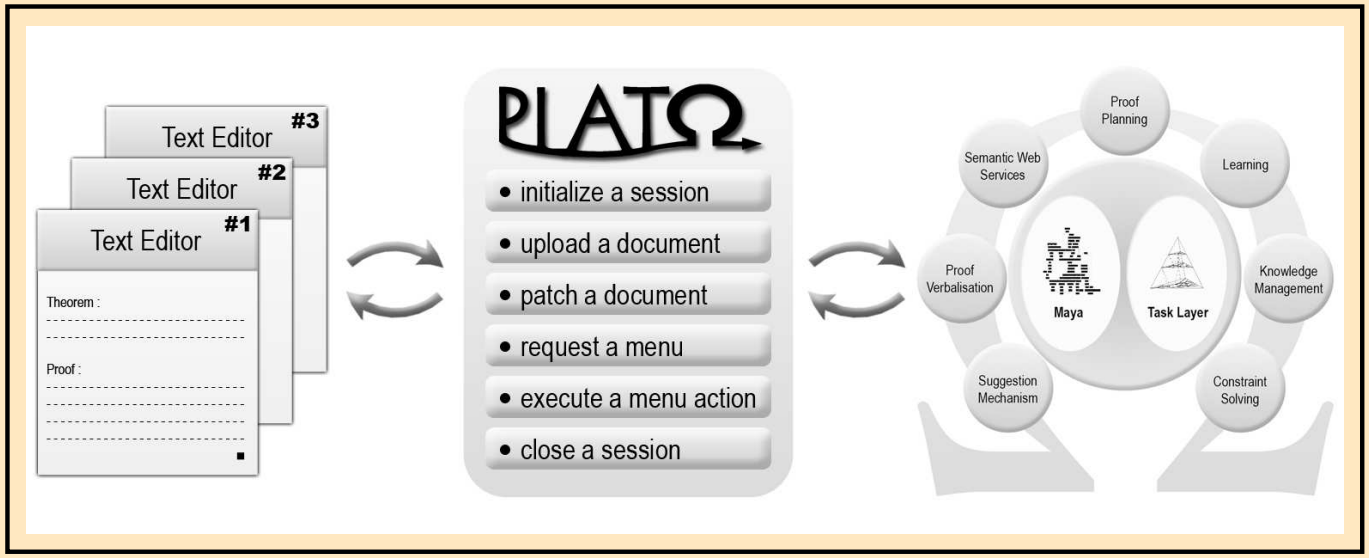
Plato at a Glance



- The user authors his mathematical documents with a scientific WYSIWYG text-editor
  - in the informal language he is used to, that is a mixture of natural language and formulas
  - using semantic annotations (PL) that follow the textual structure

Source: Autexier

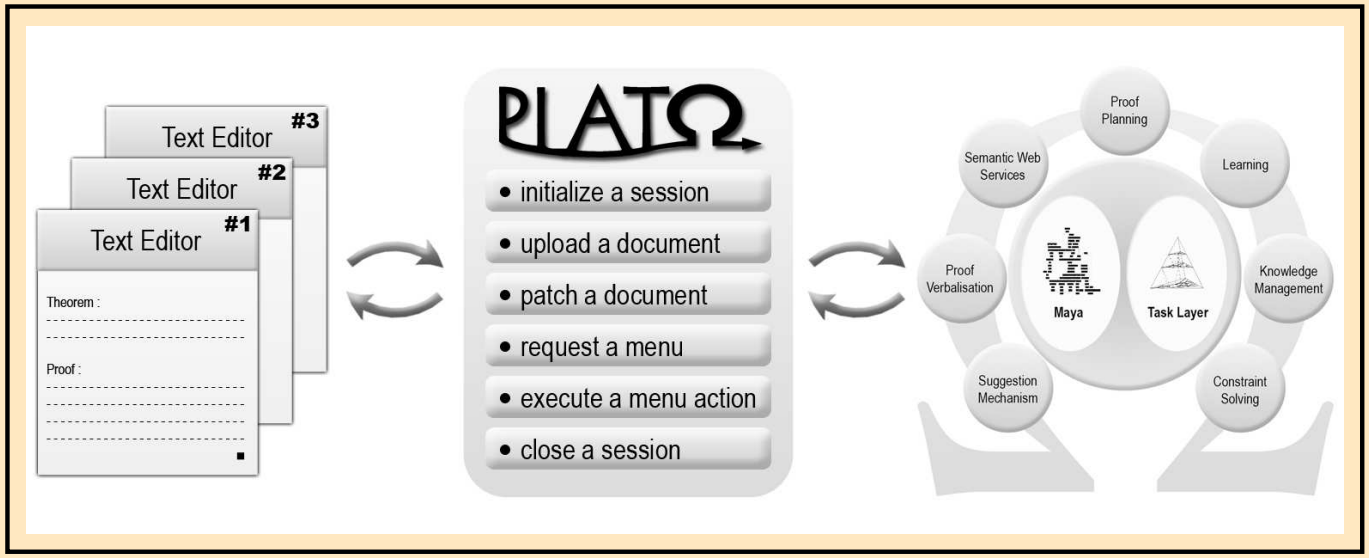
Plato at a Glance



- The user authors his mathematical documents with a scientific WYSIWYG text-editor
- Generate from this informal semantic representation the corresponding formal representation for a proof assistant

Source: Autexier

Plato at a Glance



- The user authors his mathematical documents with a scientific WYSIWYG text-editor
- The primary task of PLATΩ(Marc Wagner, Diploma Thesis, ?June 2006?)
  - Maintain consistent formal (PA) and informal representations (Text-editor)
  - Relay service requests from the Text-editor to the PA
  - Propagate changes from the PA to the Text-editor

Source: Autexier

My work during the visit to the DREAM Group:

“Automated Reasoning in Large Structured Theories”

(EPSRC Visiting Researcher Project)

Source: Autexier

Motivation

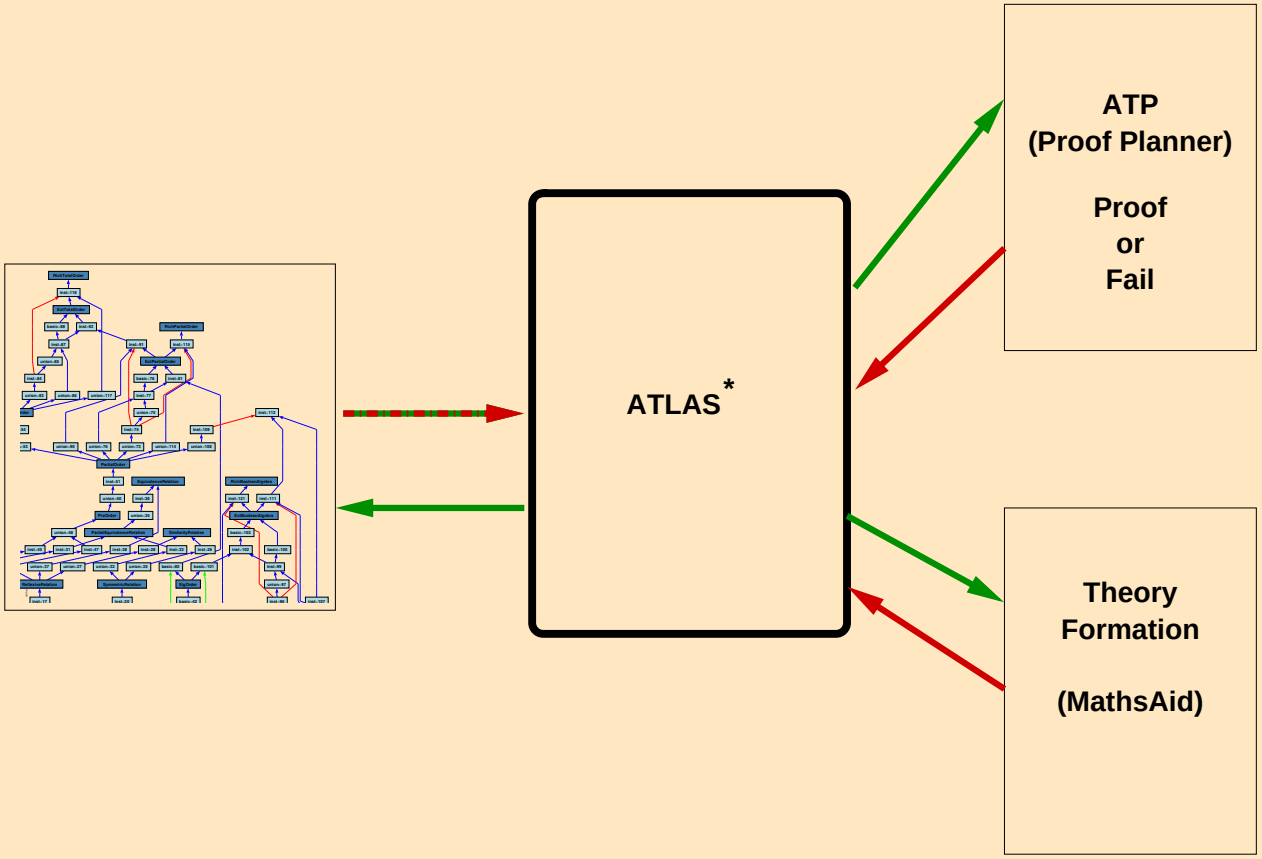
Failures of automatic proof search procedures (e.g. classical ATP, Proof Planner, . . .) and reasons:

- Too much knowledge/Search space too large (Restrict Knowledge)
- Not a theorem
  - Only part of knowledge has been provided: (Adjust provided knowledge)
  - All knowledge has been provided: (Definitely not a theorem)

Source: Autexier

Question

- How to automate restriction and adjustment of knowledge provided to the ATPs? (Based on Feedback provided in case of failure)



*The titan in the Greek mythology punished by Zeus by giving him the task to hold up the sky

Source: Autexier

Main Objectives

- System environment that supports the definition and evaluation of knowledge filters and in-the-large reasoning procedures.
- Criteria characterising the knowledge that can be provided by ATP, ATF and systems maintaining structured theories
- Basic set of filters to formulate queries for each subsystems
- Criteria to rate the response given by a subsystem to a query.
- Sample in-the-large reasoning heuristics:
  - Given the goal to prove a conjecture
  - Automate cooperation among the subsystems by formulating queries, evaluating responses and formulating new queries.
- *Status:* Interfacing systems nearly done

Source: Autexier

Summary

- Development Graphs to maintain formal specifications and proofs
- Hierarchical proof datastructure that allows for *alternative proofs* on *different levels of granularity*, manages alternative substitutions
- Unifying tactics and methods into *inferences*
  - *deep application* of inferences (inspired by CoRe's assertion application)
  - automatic synthesis of inferences from mathematical knowledge
  - automatic computation of additional information required by proof planner and ΩAnts
- PlatΩ tool to plug ΩMEGA into texteditors:
  - mediates between text structure and logical structure, relays service requests and propagates changes (ask for screenshots of a worked example)
- Project: "Automated Reasoning in Large, Structured Theories"

Source: Autexier

The initial Document in the Text-Editor

Theory [setdistr]

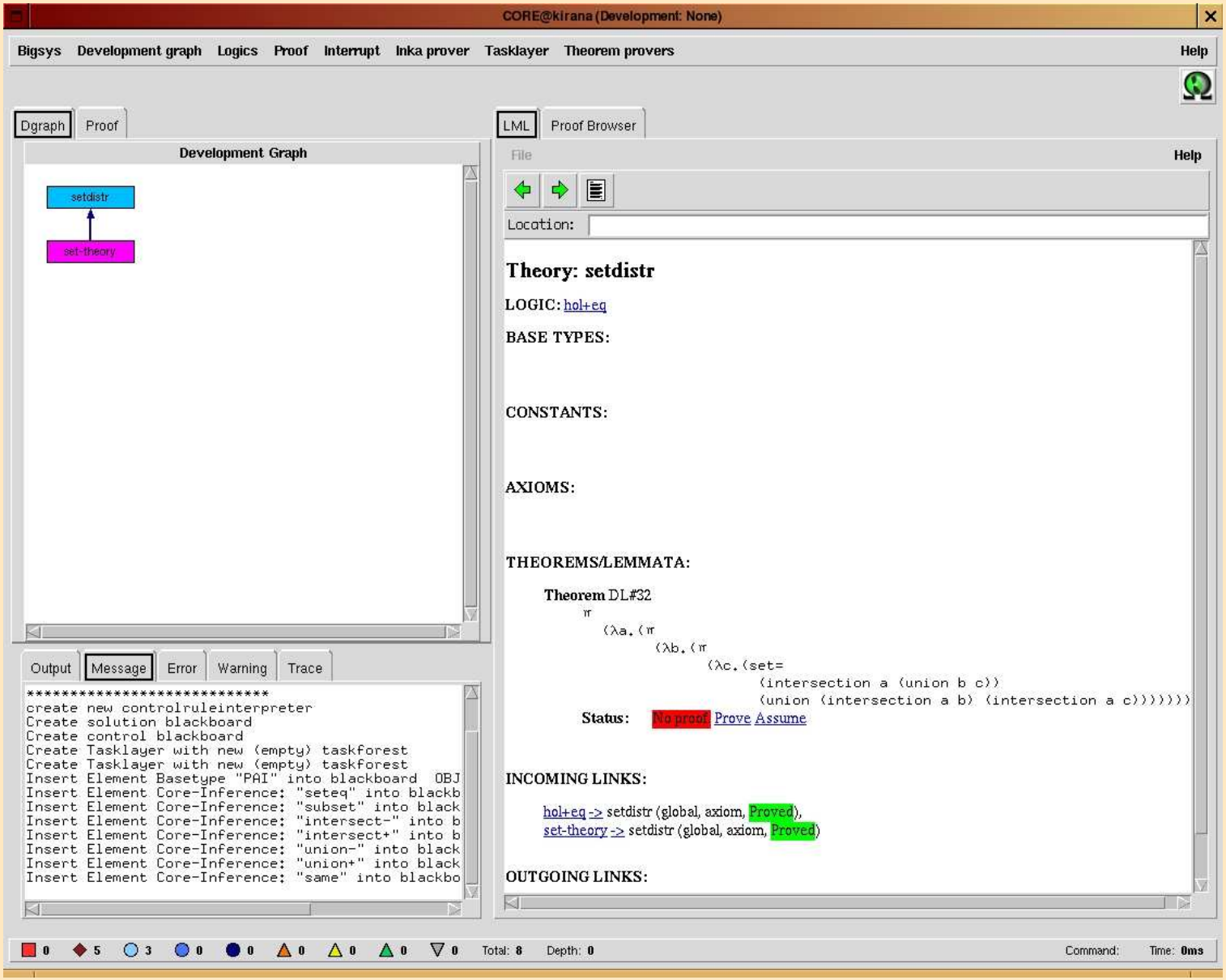
Context : We are going to prove a simple theorem in the theory of sets (->[set-theory])

Theorem [Distributivity of Intersection] :
The following proposition follows at once from the definitions:
$\forall A, B, C$:set . $(A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C))$

Proof:
In order to prove this equation, we show that the following subset relations hold:
(1) $\forall A, B, C$:set . $((A \cap (B \cup C)) \subset ((A \cap B) \cup (A \cap C)))$
(2) $\forall A, B, C$:set . $(((A \cap B) \cup (A \cap C)) \subset (A \cap (B \cup C)))$

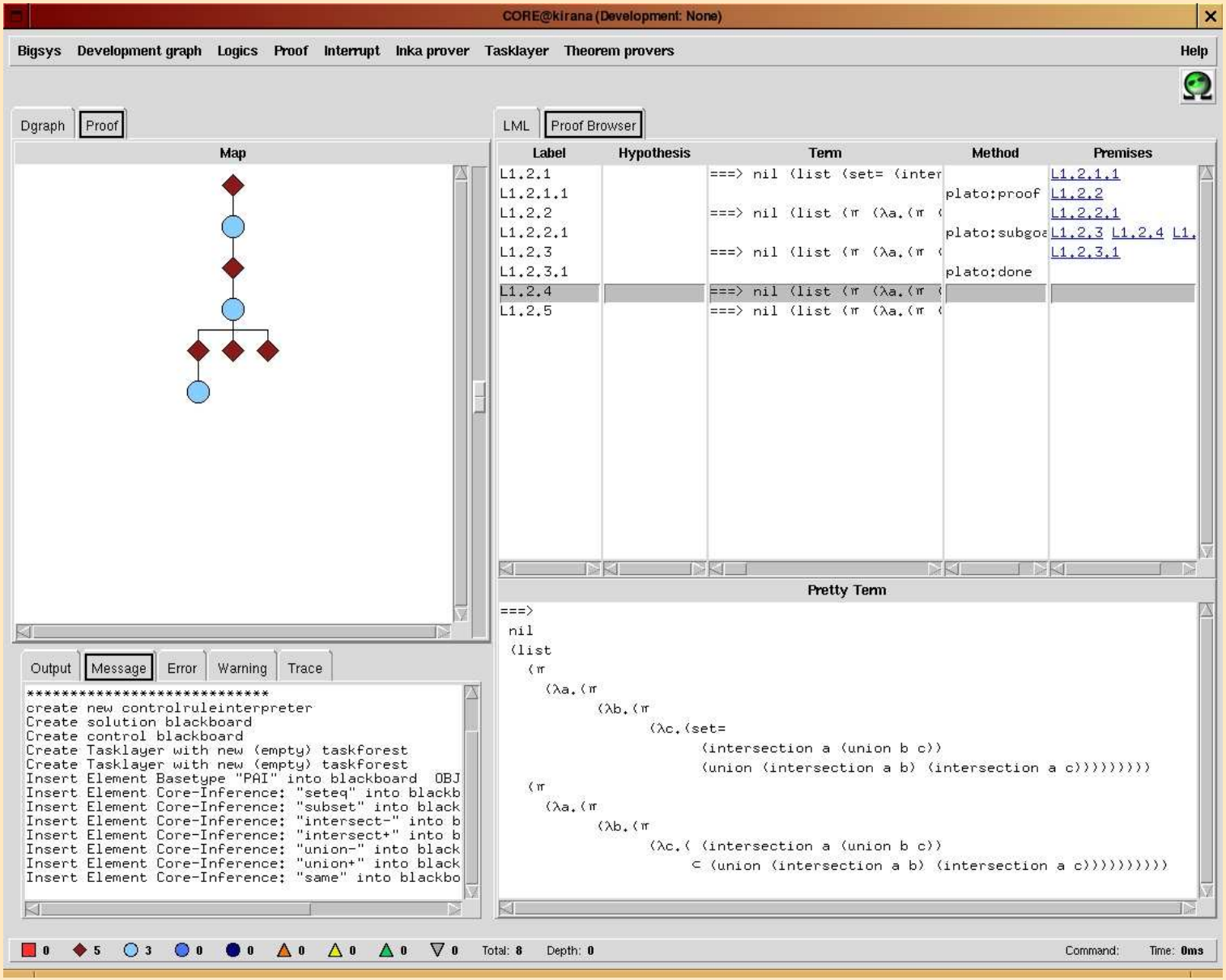
Source: Autexier

Uploaded Theory in the MAYA system



Source: Autexier

Uploaded Proof in the TASK LAYER



Source: Autexier

User-Modified Document in the Text-Editor

Theory [setdistr]

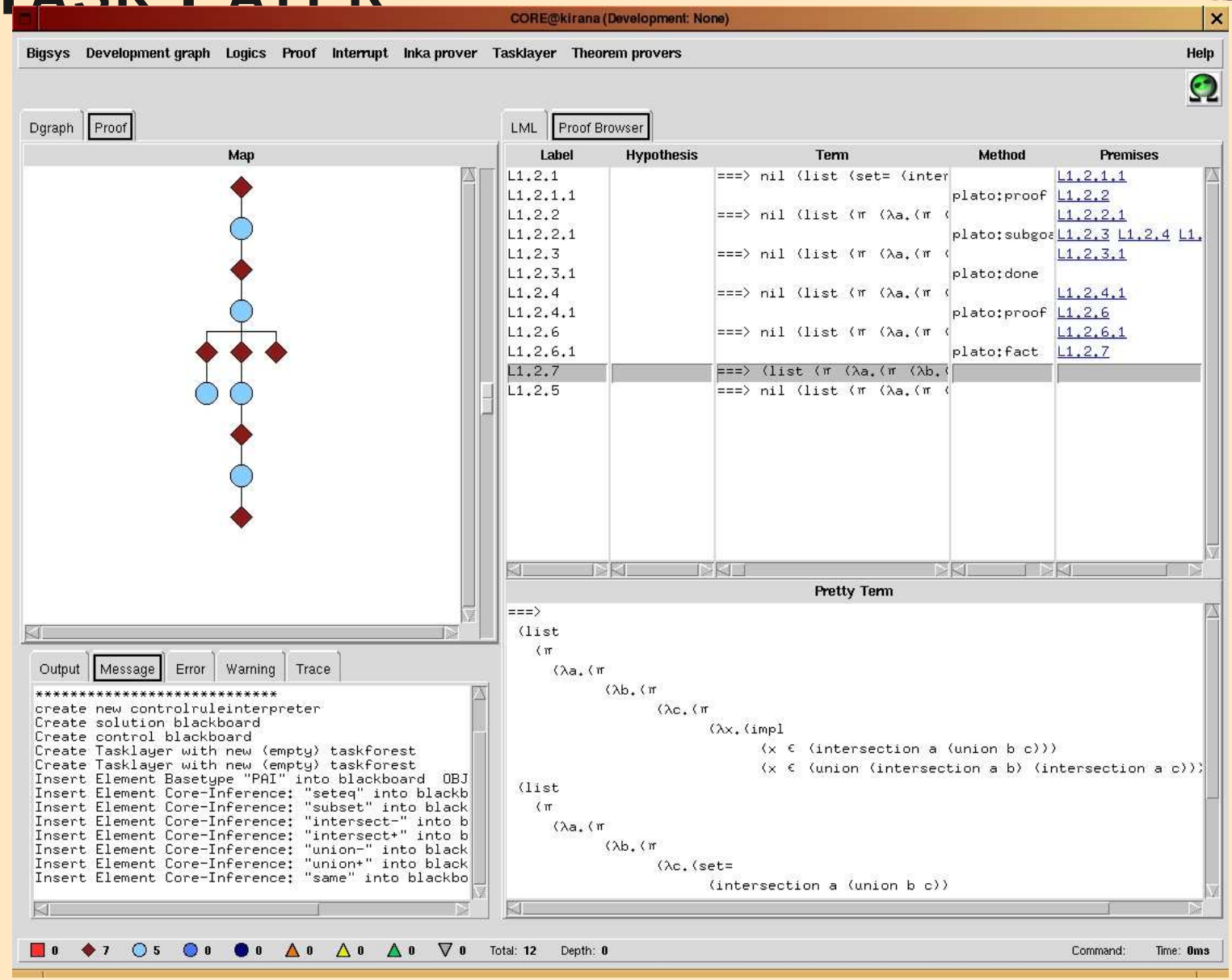
Context : We are going to prove a simple theorem in the theory of sets (->[set-theory])

Theorem [Distributivity of Intersection] :
The following proposition follows at once from the definitions:
$\forall A, B, C$:set . $(A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C))$

Proof:
In order to prove this equation, we show that the following subset relations hold:
(1) $\forall A, B, C$:set . $((A \cap (B \cup C)) \subset ((A \cap B) \cup (A \cap C)))$
(2) $\forall A, B, C$:set . $(((A \cap B) \cup (A \cap C)) \subset (A \cap (B \cup C)))$
We prove first the proposition (1).
It follows that $\forall A, B, C$:set x:i. $x \in (A \cap (B \cup C)) \Rightarrow x \in ((A \cap B) \cup (A \cap C))$

Source: Autexier

Transformed User-Modified Proof in the TASK LAYER



Source: Autexier

Menu Interaction in the Text-Editor

Theory [setdistr]

Context : We are going to prove a simple theorem in the theory of sets (->[set-theory])

Theorem [Distributivity of Intersection] :
The following proposition follows at once from the definitions:
$\forall A, B, C$:set . $(A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C))$

Proof:
In order to prove this equation, we show that the following subset relations hold:
(1) $\forall A, B, C$:set . $((A \cap (B \cup C)) \subset ((A \cap B) \cup (A \cap C)))$
(2) $\forall A, B, C$:set . $(((A \cap B) \cup (A \cap C)) \subset (A \cap (B \cup C)))$
We prove first the proposition (1).
It follows that $\forall A, B, C$:set x:i. $x \in (A \cap (B \cup C)) \Rightarrow x \in ((A \cap B) \cup (A \cap C))$

Apply same
Apply union+
Apply union-
Apply intersect+: Arguments: Compute Instantiations
Apply intersect-
Apply subset

Source: Autexier

Patched Menu in the Text-Editor

Theory [setdistr]

Context : We are going to prove a simple theorem in the theory of sets (->[set-theory])

Theorem [Distributivity of Intersection] :
The following proposition follows at once from the definitions:
$\forall A, B, C$:set . $(A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C))$

Proof:
In order to prove this equation, we show that the following subset relations hold:
(1) $\forall A, B, C$:set . $((A \cap (B \cup C)) \subset ((A \cap B) \cup (A \cap C)))$
(2) $\forall A, B, C$:set . $(((A \cap B) \cup (A \cap C)) \subset (A \cap (B \cup C)))$
We prove first the proposition (1).
It follows that $\forall A, B, C$:set x:i. $x \in (A \cap (B \cup C)) \Rightarrow x \in ((A \cap B) \cup (A \cap C))$

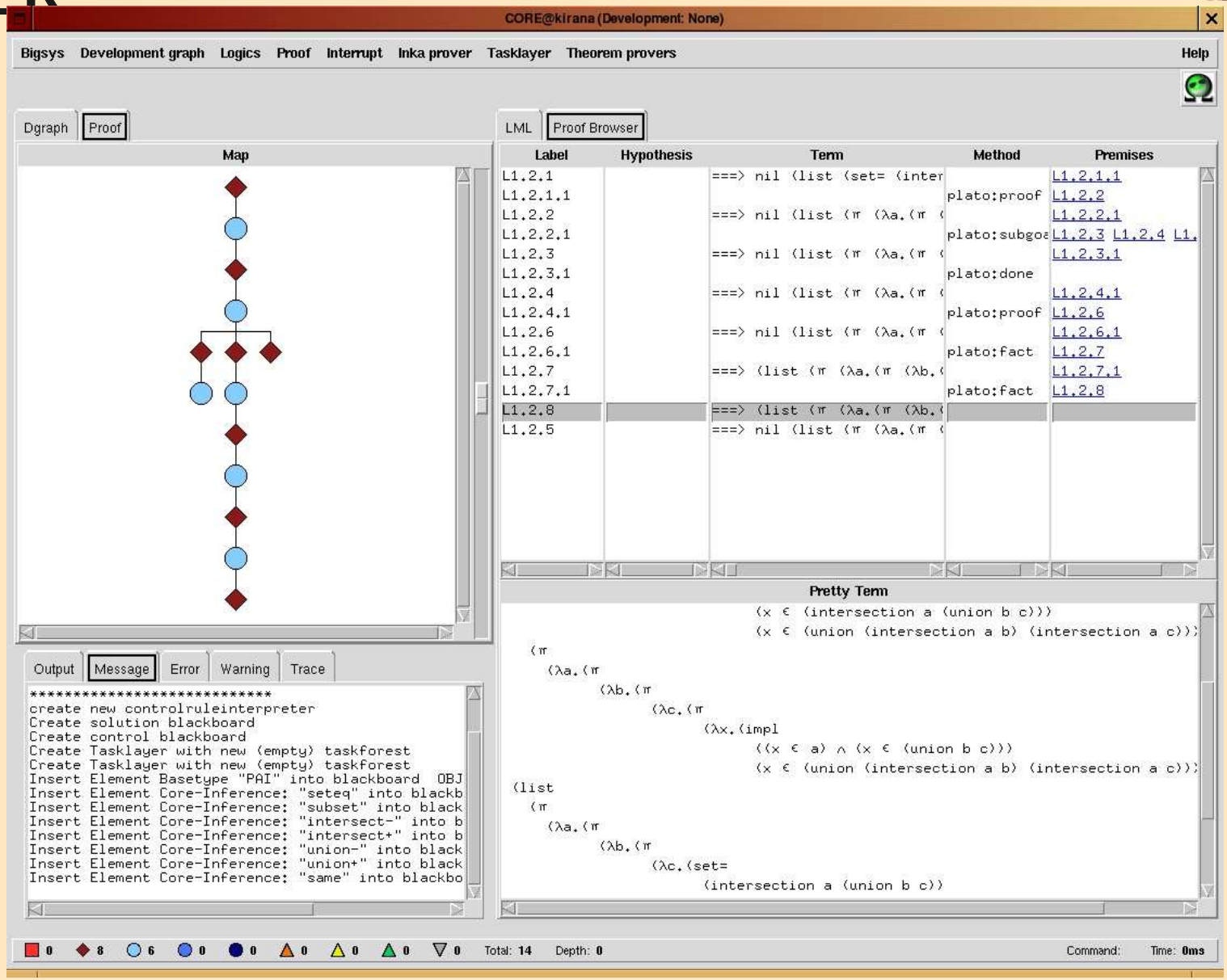
Apply same
Apply union+
Apply union-

Apply intersect+:	Arguments:	$P1 \mapsto x \in A \wedge x \in (B \cup C)$
		$C \mapsto x \in (A \cap (B \cup C))$
		GSETB $\mapsto (B \cup C)$
		GSETA $\mapsto A$
		$X \mapsto x$

Apply intersect-
Apply subset

Source: Autexier

System-Modified Proof in the TASK LAYER



Source: Autexier

Patched Document in the Text-Editor

Theory [setdistr]

Context : We are going to prove a simple theorem in the theory of sets (->[set-theory])

Theorem [Distributivity of Intersection] :
The following proposition follows at once from the definitions:
$\forall A, B, C$:set . $(A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C))$

Proof:
In order to prove this equation, we show that the following subset relations hold:
(1) $\forall A, B, C$:set . $((A \cap (B \cup C)) \subset ((A \cap B) \cup (A \cap C)))$
(2) $\forall A, B, C$:set . $(((A \cap B) \cup (A \cap C)) \subset (A \cap (B \cup C)))$
We prove first the proposition (1).
It follows that $\forall A, B, C$:set x:i. $x \in (A \cap (B \cup C)) \Rightarrow x \in ((A \cap B) \cup (A \cap C))$
$\forall A, B, C$:set x:i. $x \in A \wedge x \in (B \cup C) \Rightarrow x \in ((A \cap B) \cup (A \cap C))$

Source: Autexier