



The New Proof Assistant Ω MEGA: Developments and Applications

Serge Autexier
(joint work with the Ω MEGA Group*)

`serge@ags.uni-sb.de`

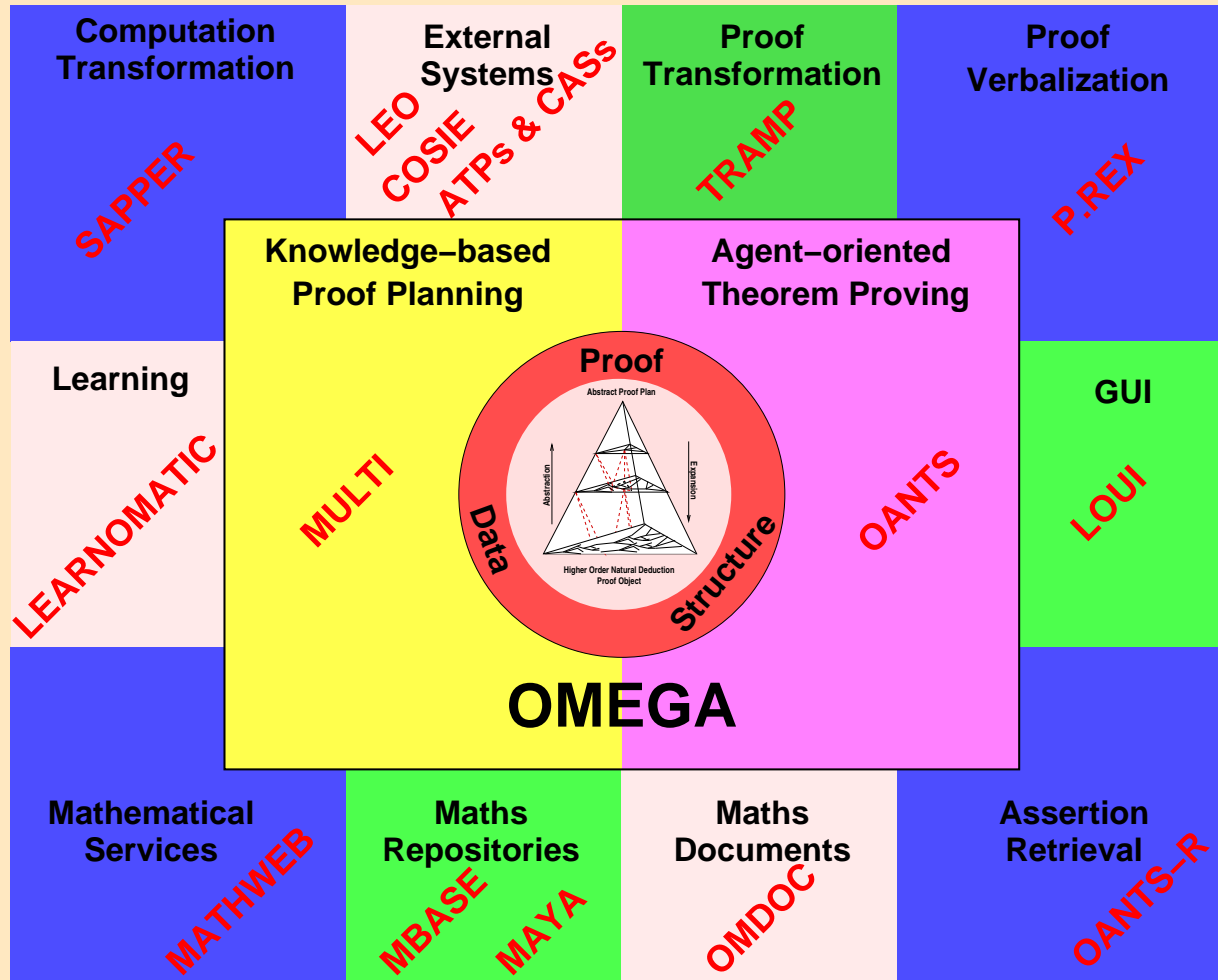
DFKI GmbH & CS Department, Saarland University, Saarbrücken, Germany
& CISA, University of Edinburgh, Scotland (May-July)

STP 2006, June 6, 2006

Glasgow, Scotland

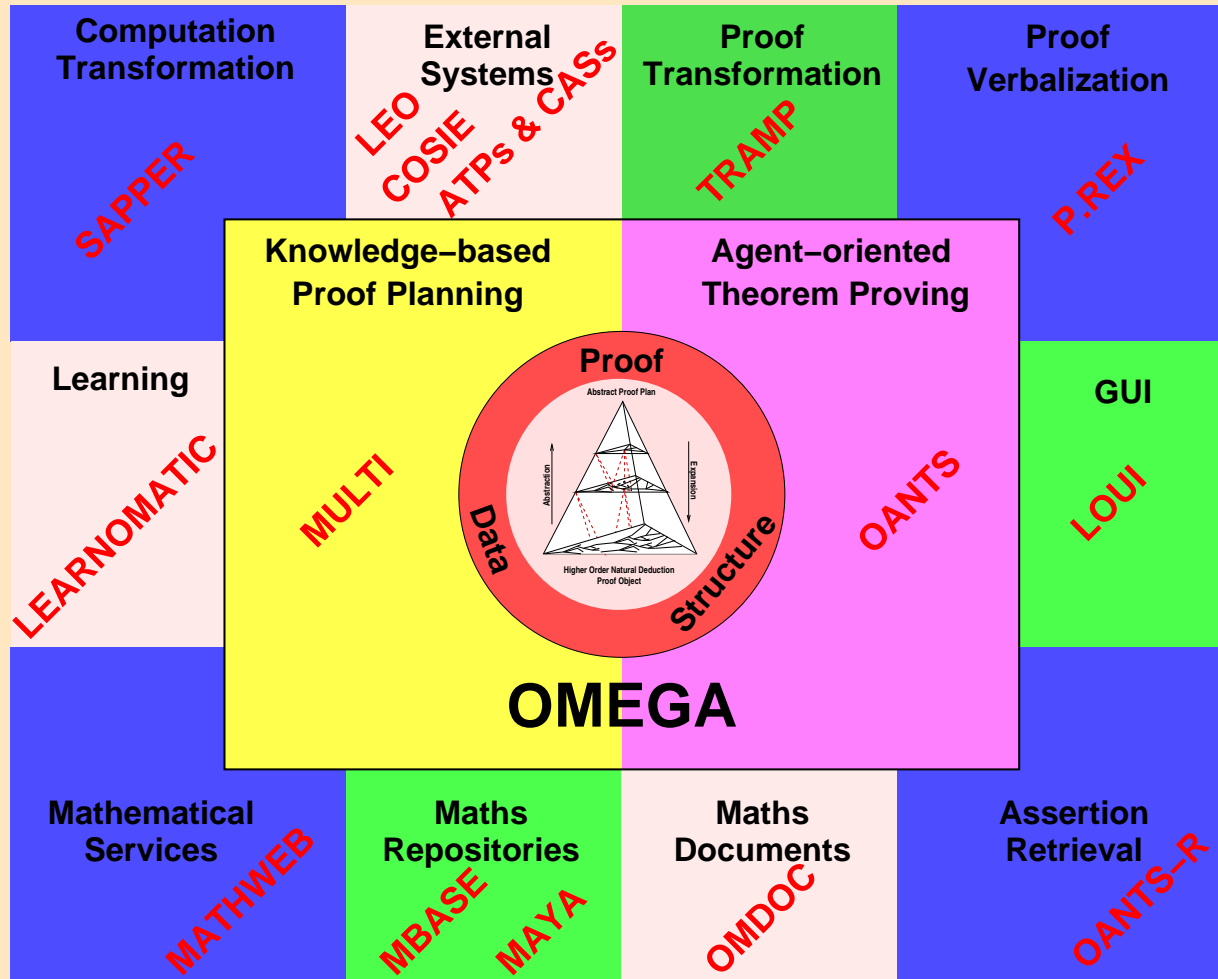
* Jörg Siekmann, Christoph Benz Müller, Chad E. Brown, Mark Buckley, Dominik Dietrich, Armin Fiedler (part-time), Andreas Franke, Henri Lesourd, Marvin Schiller, Frank Theiß, Marc Wagner, Jürgen Zimmer

Old Ω MEGA



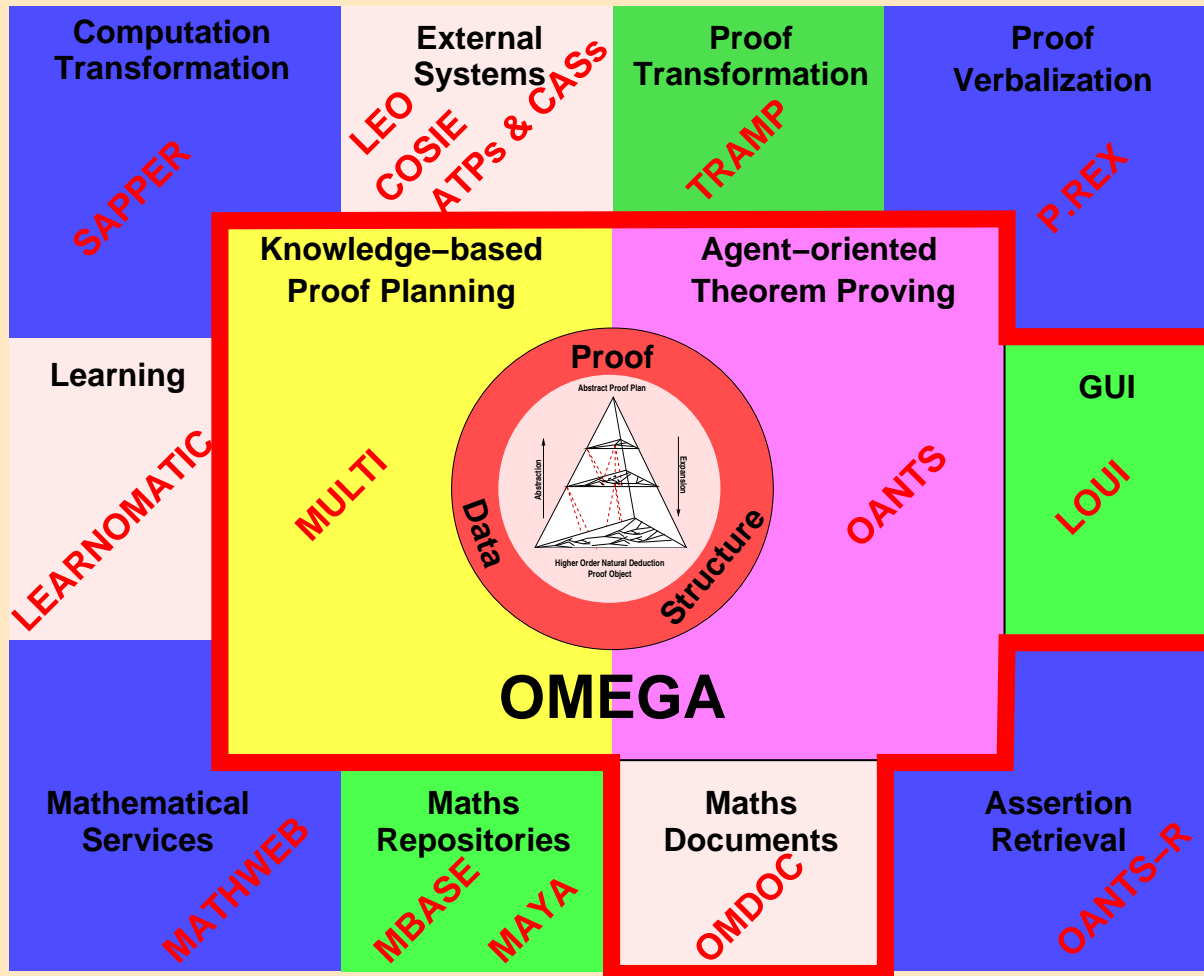
- Proof-Planning
(MULTI, E. Melis, A. Meier)
- Agent-based Theorem Proving
(Ω ANTS, V. Sorge)
- NL Verbalisation of Proofs
(*P.rex*, A. Fiedler)
- External systems (ATP, CAS)
(MATHWEB-SB J. Zimmer)
- Proof transformation
(TRAMP, A. Meier)
- Math. Knowledge base
(MBASE A. Franke)
- OMDOC
(M. Kohlhase)

Old Ω MEGA



- 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 *L Ω UI*

Old Ω MEGA



- 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 *LΩUI*

Why and What did we Change?



- Natural Deduction Calculus too cumbersome (not natural enough)

Why and What did we Change?



- Natural Deduction Calculus too cumbersome (not natural enough)
 - ▶ Want direct *reasoning with the assertions* (interactive & automatic)
 - ⇒ CoRE-calculus that directly supports that [[Autexier, CADE'05](#)]

Why and What did we Change?



- Natural Deduction Calculus too cumbersome (not natural enough)
 - ▶ Want direct *reasoning with the assertions* (interactive & automatic)
 - ⇒ CoRE-calculus that directly supports that [[Autexier, CADE'05](#)]
- The proof datastructure was completely overcrowded

Why and What did we Change?



- Natural Deduction Calculus too cumbersome (not natural enough)
 - ▶ Want direct *reasoning with the assertions* (interactive & automatic)
 - ⇒ CoRE-calculus that directly supports that [[Autexier, CADE'05](#)]
- The proof datastructure was completely overcrowded
 - ▶ New generic proof datastructure (PDS)

Why and What did we Change?



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

Why and What did we Change?



- Natural Deduction Calculus too cumbersome (not natural enough)
 - ▶ Want direct *reasoning with the assertions* (interactive & automatic)
 - ⇒ 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 *LΩMI* nice, but did not meet requirements of targeted users, say mathematicians

Why and What did we Change?



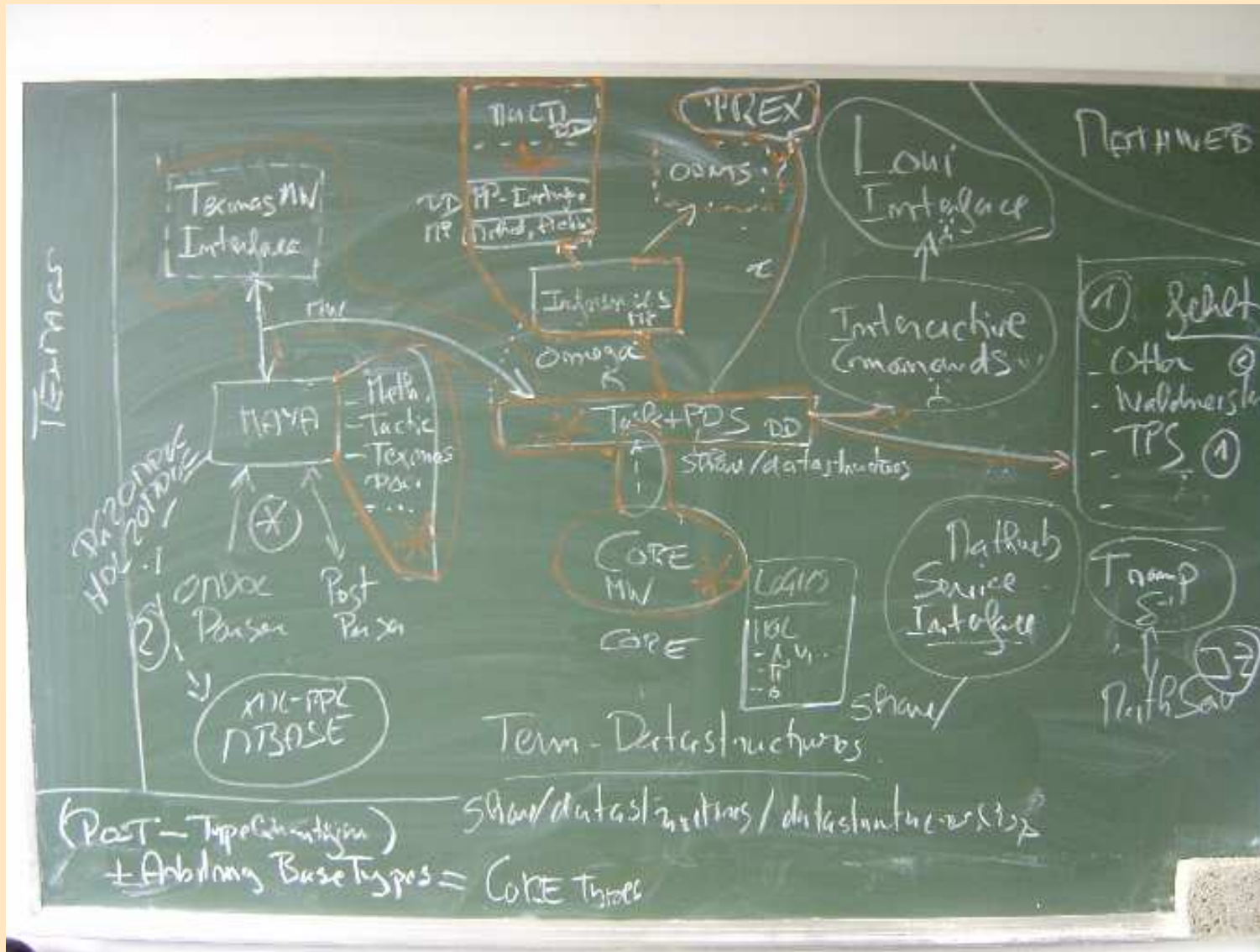
- Natural Deduction Calculus too cumbersome (not natural enough)
 - ▶ Want direct *reasoning with the assertions* (interactive & automatic)
 - ⇒ CORE-calculus that directly supports that [Autexier, CADE'05]
- The proof datastructure was completely overcrowded
 - ▶ New generic proof datastructure (PDS)
 - ▶ Definition of the *task-layer* as uniform reasoning platform
- GUI *L Ω UI* nice, but did not meet requirements of targeted users, say mathematicians
 - ▶ Plug Ω MEGA as reasoning service provider into tools anyway used by users, e.g. WYSIWYG text-editors like TEXMACS (analogy: grammar checkers)

Why and What did we Change?

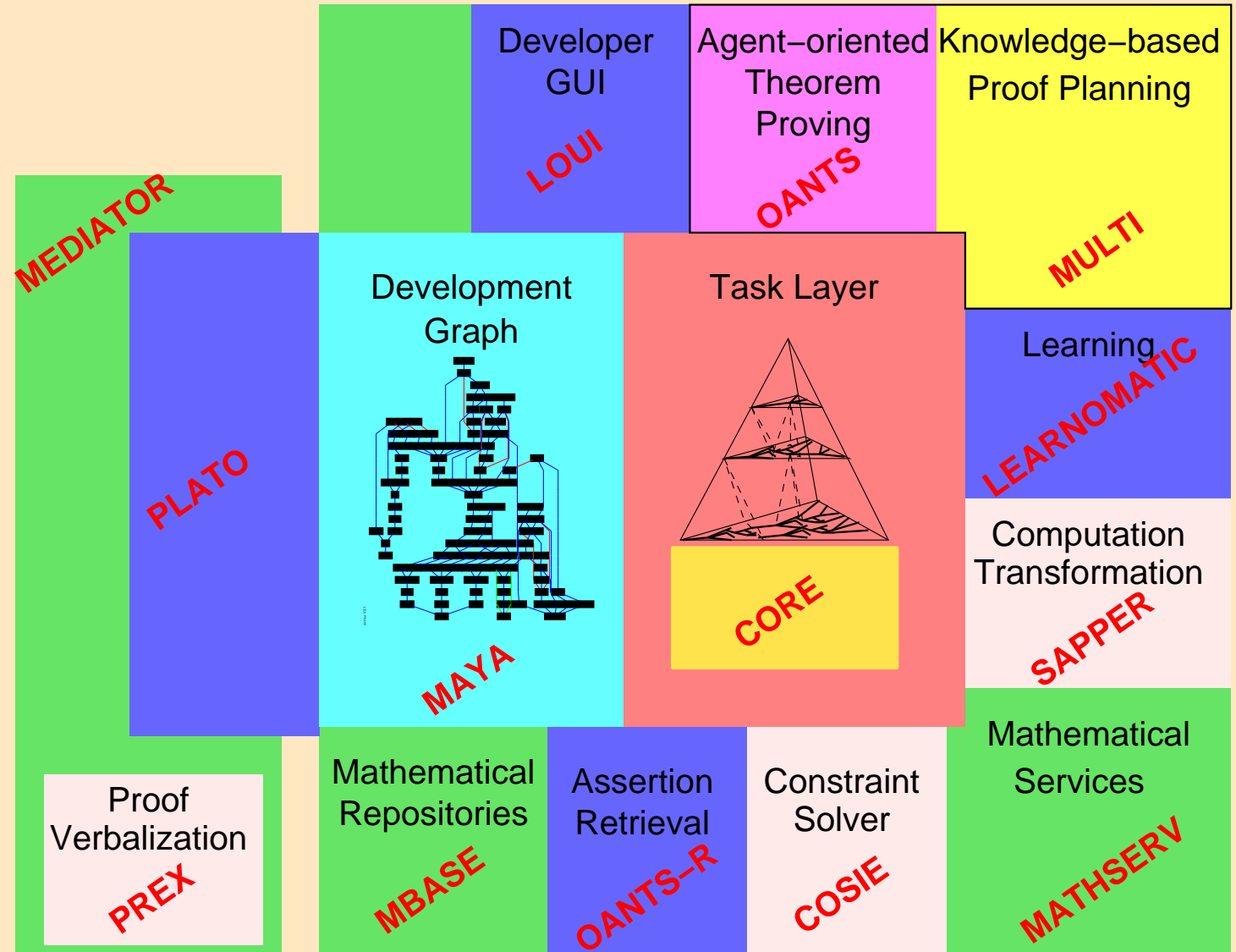
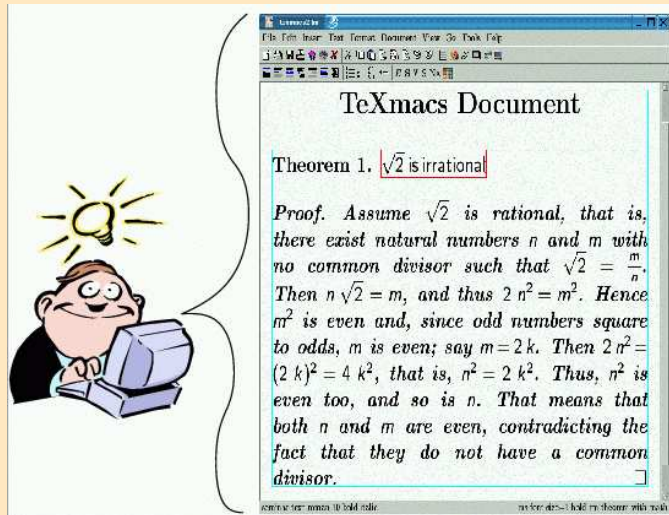
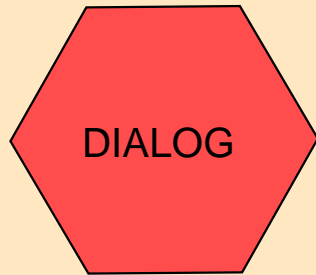


- Natural Deduction Calculus too cumbersome (not natural enough)
 - ▶ Want direct *reasoning with the assertions* (interactive & automatic)
 - ⇒ CORE-calculus that directly supports that [Autexier, CADE'05]
- The proof datastructure was completely overcrowded
 - ▶ New generic proof datastructure (PDS)
 - ▶ Definition of the *task-layer* as uniform reasoning platform
- GUI *LΩMI* 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

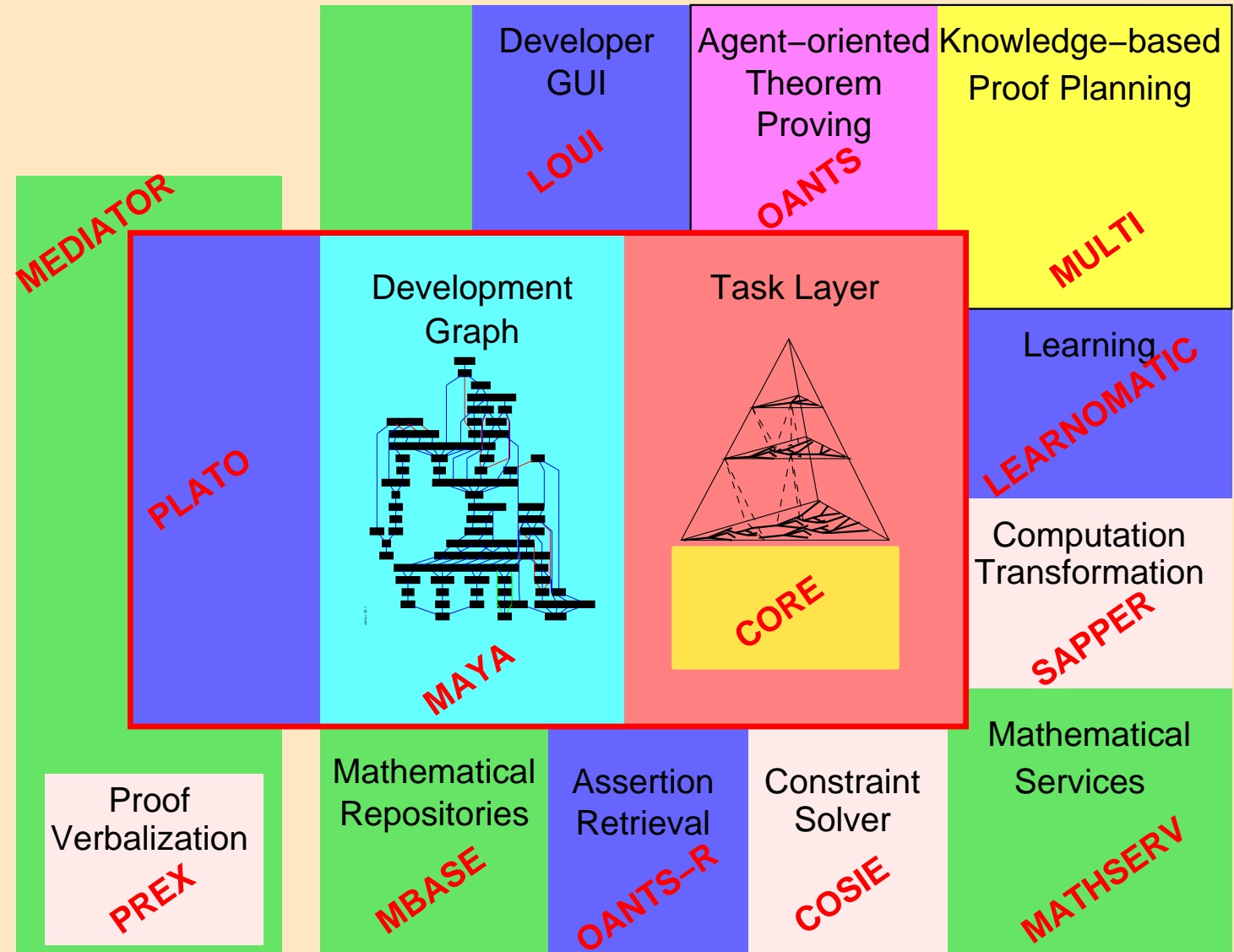
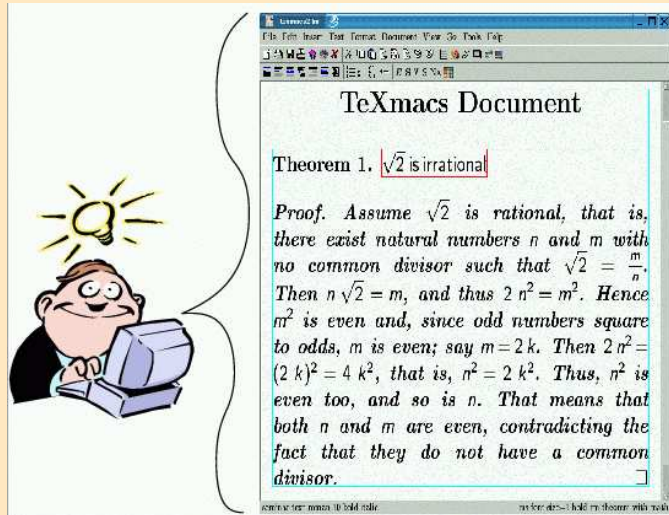
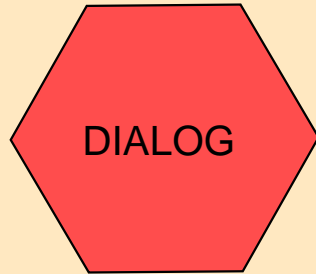
Towards the New Ω MEGA



New Architecture



This Talk

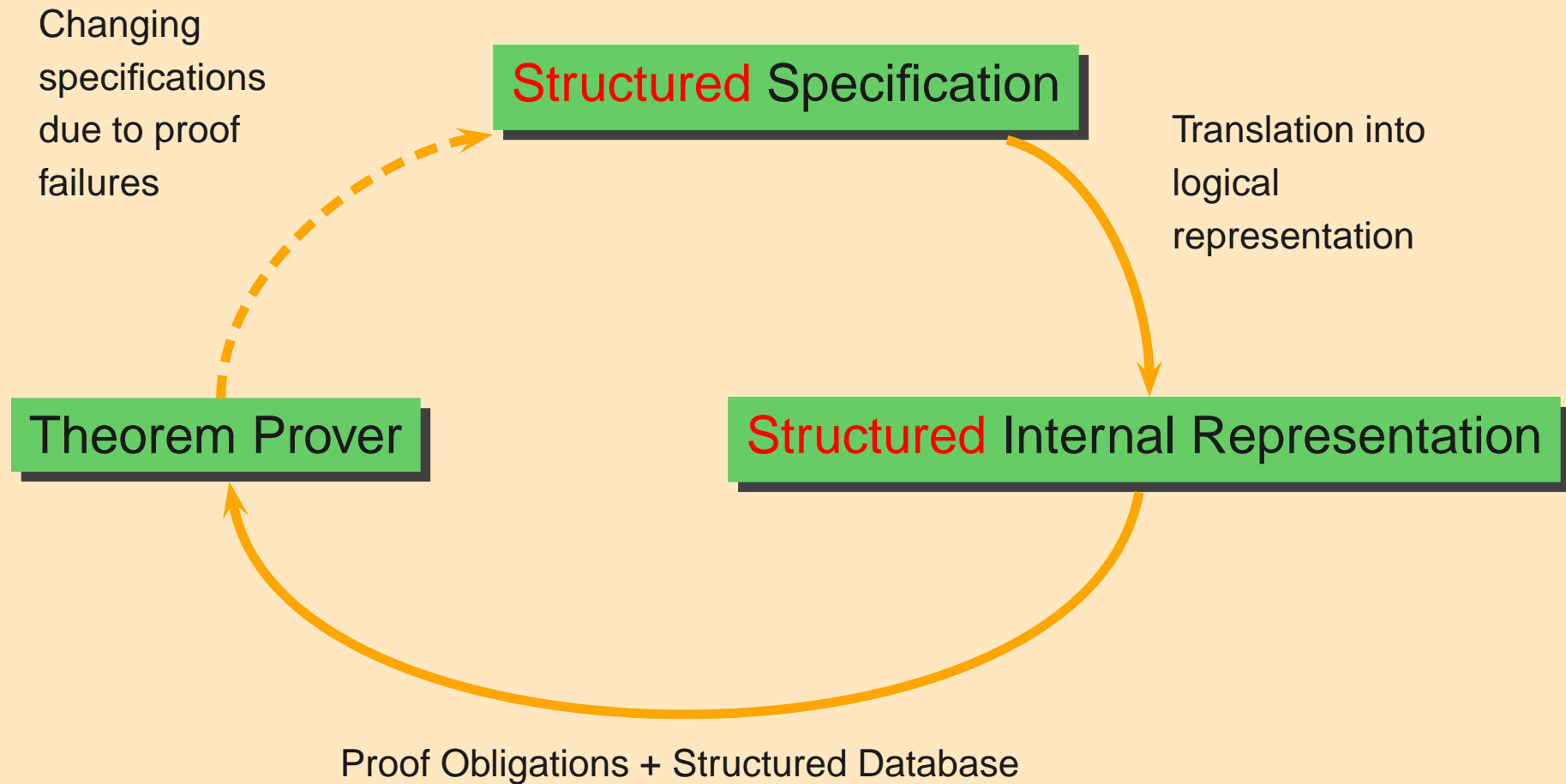


Development graphs: Maintaining structured mathematical theories

Context



Evolutionary Formal Development



Development Graphs



LIST[ELEM]

Local Signature:

- **Local sorts** List[Elem]
- **Local constants** [] : List[Elem], ...

Local Axioms: · [] ++ K = K, ...

Local Lemmata:

- reverse(K ++ L) = reverse(L) ++ reverse(K)
- ...

MONOID

Local Signature:

- Local sorts: Elem
- Local constants: * : Elem × Elem → Elem

Local Axioms: x * e = x, ...



id

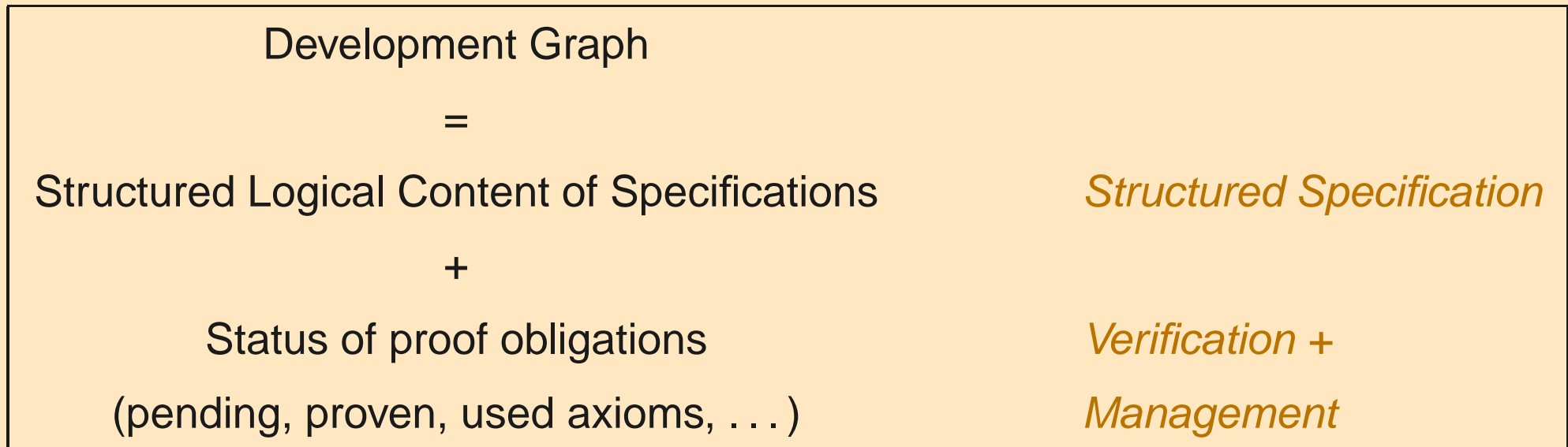
A blue arrow labeled "id" points from the bottom box to the LIST[ELEM] box, indicating an identity mapping.

Local Signature:

- Local sorts: Elem

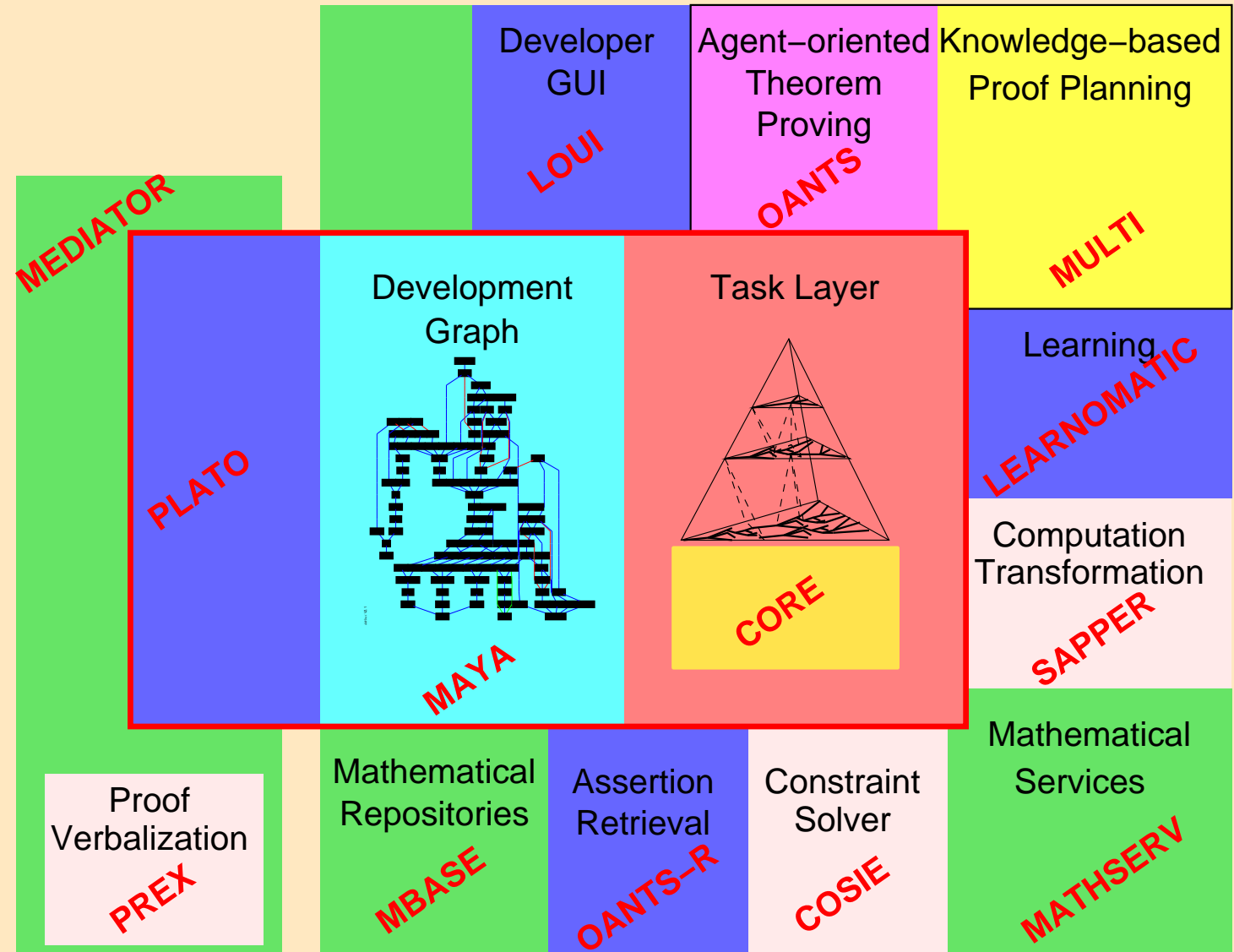
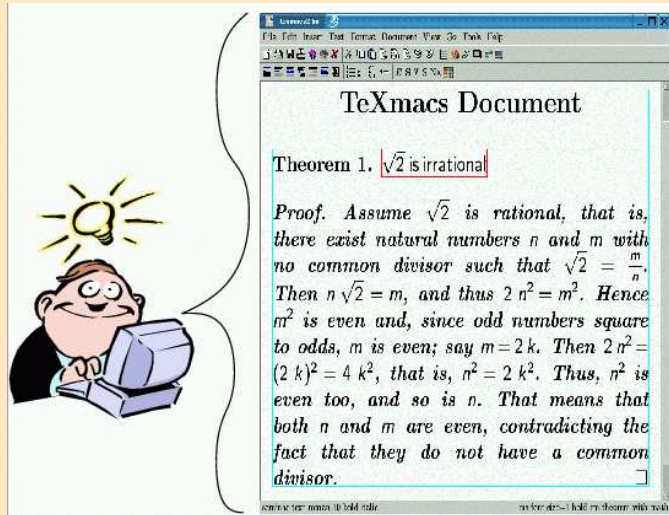
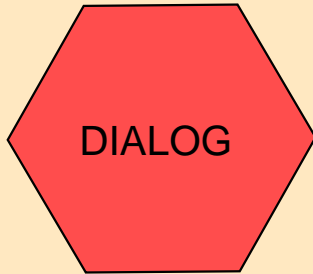
[Hutter'00], [AutexierHutter'02&'05], [MossakowskiAutexierHutter'01&'06]

Role of Development Graphs



- Deals with *evolution* of formal specifications and proofs
 - ⇒ 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

This Talk



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

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, Ω_{ANTS})

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, Ω_{ANTS})



The new Proof Datastructure (PDS)

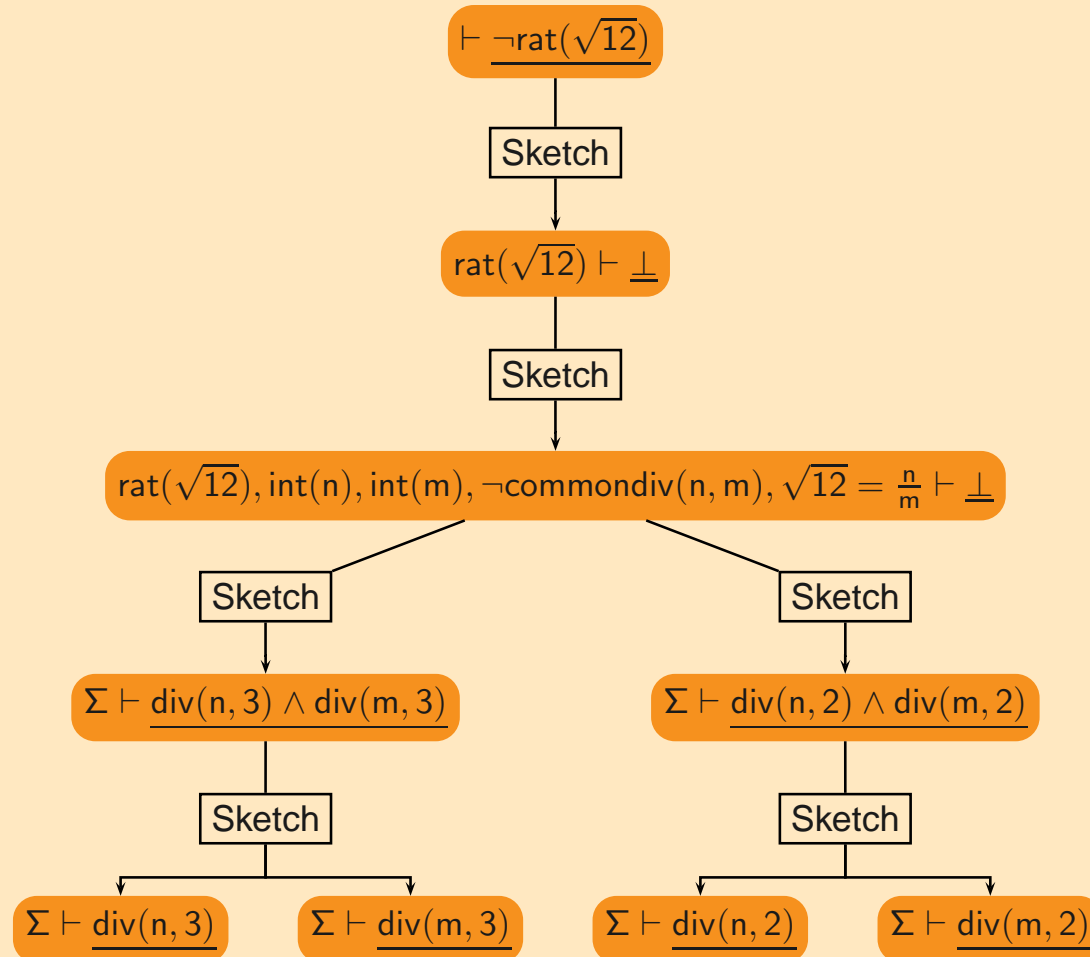
Proof Data Structure (PDS)



- 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]

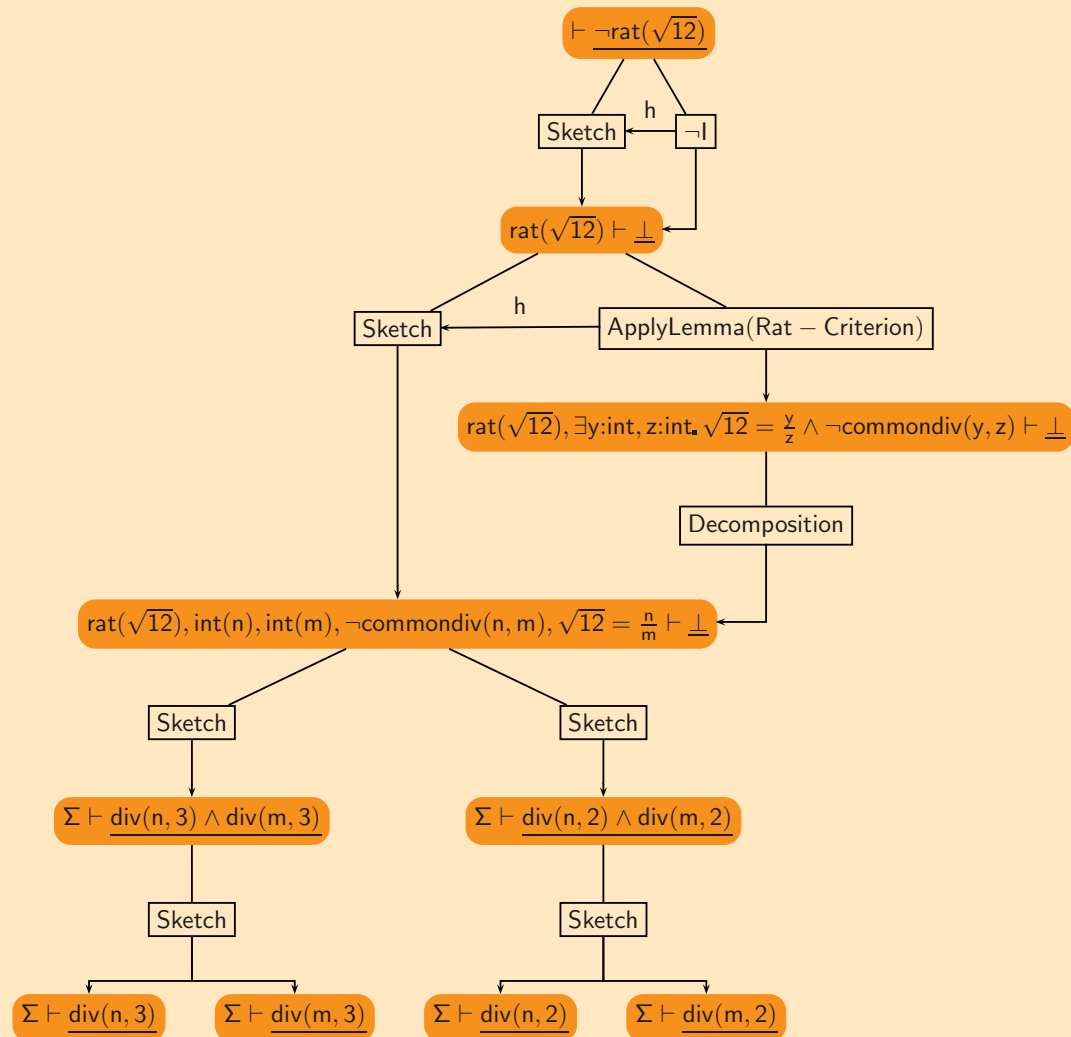
Example Abstract PDS



Example Complete PDS



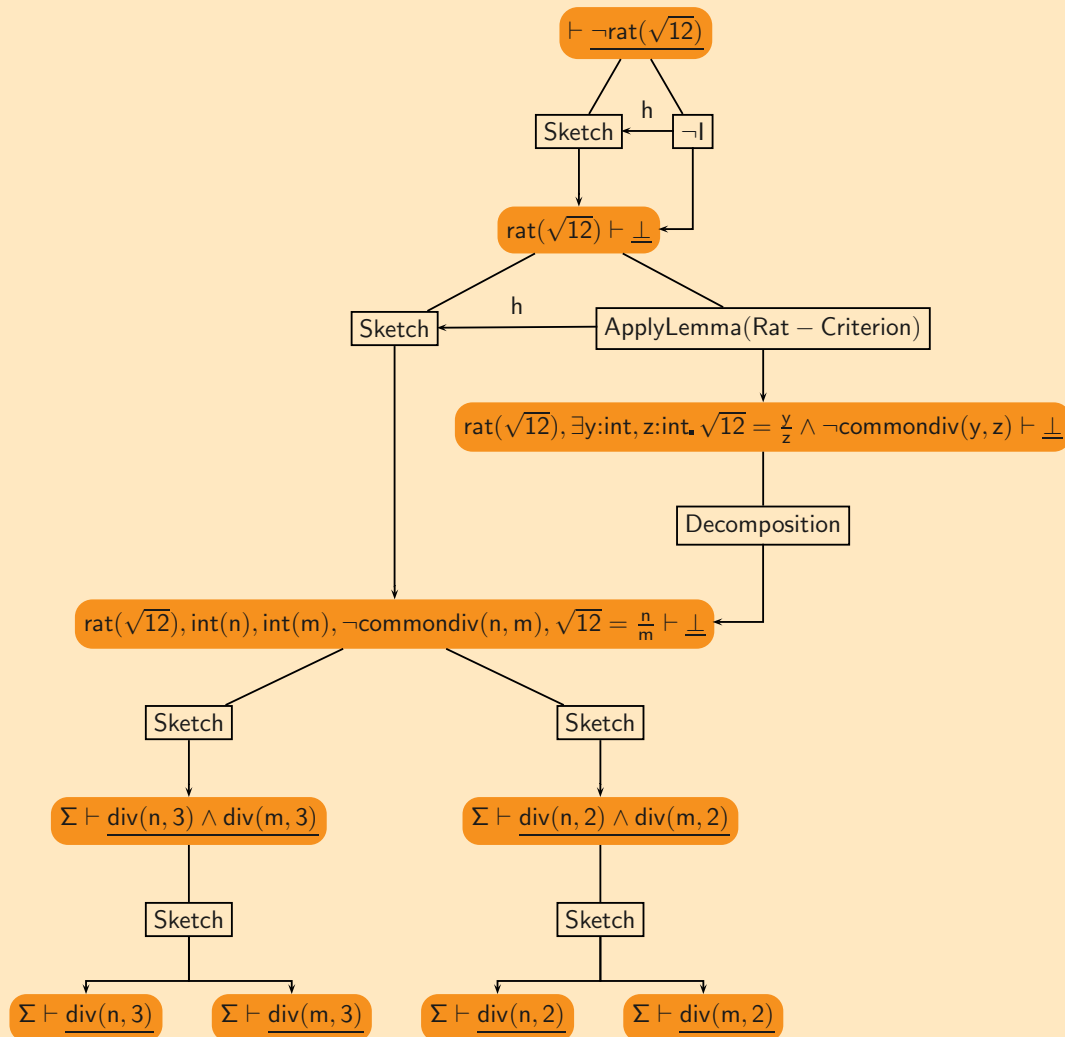
Complete PDS



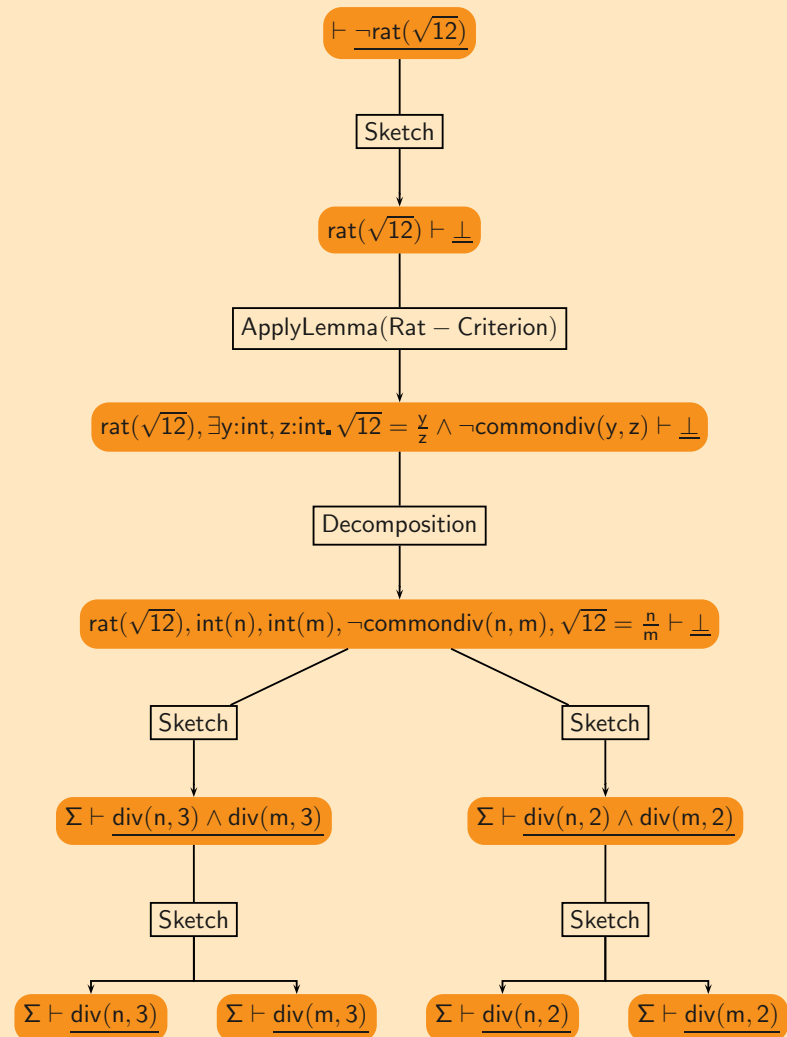
Example Complete PDS



Complete PDS



A PDS View



PDSs



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

[Dietrich, Diploma Thesis, 2006]

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

[Autexier&Sacerdoti-Coen, MKM'06]

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



Proof Construction by Inference Application

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:

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

Application Condition: –

Outline functions:

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

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

Application Condition: –

Outline functions:

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

- 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

The same as a single Inference...



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

Application Condition: –

Outline functions:

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

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

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

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

Examples of Inferences



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

Application Condition: —

Examples of Inferences



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

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

Examples of Inferences



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

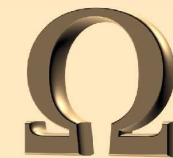
Application Condition: $O \in \{=, \leq, <, \geq, >\}$

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

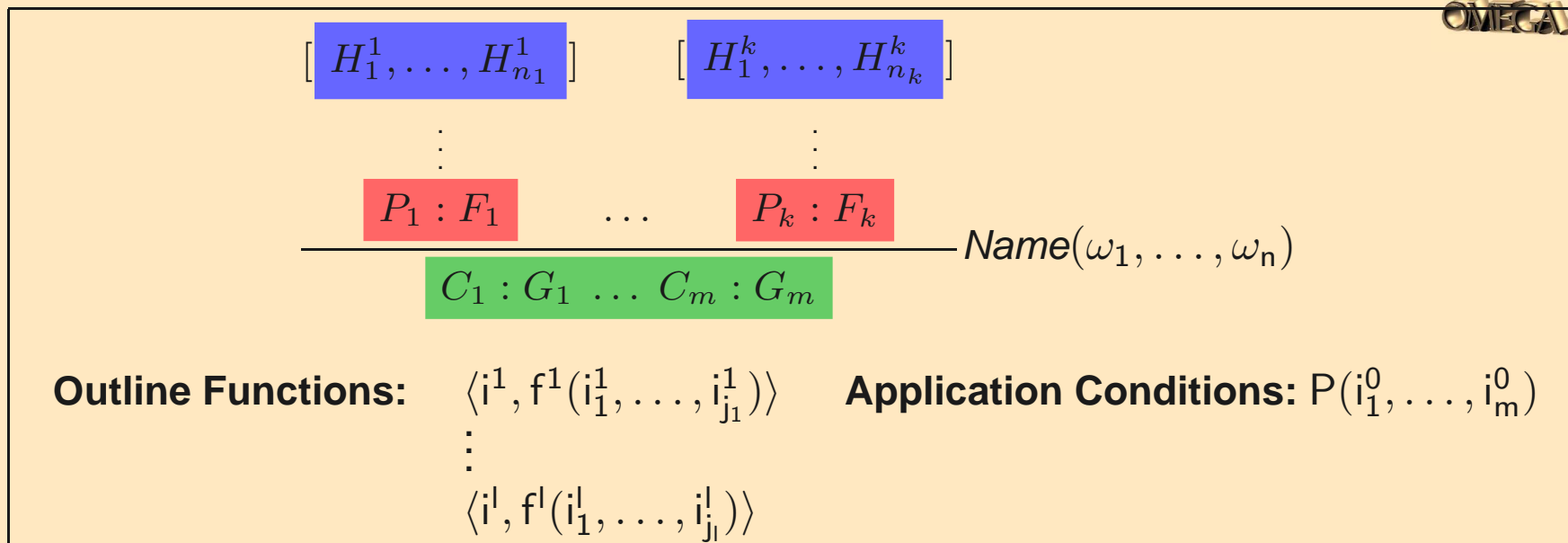
$\langle U' \text{ maple-simplify}(U) \rangle$

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

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



General Form of Inferences



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

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

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

$$\Rightarrow \lim_a f = l$$

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

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

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:

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

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:

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

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:

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

Suppose we map $P_1 \mapsto \langle 1, 2 \rangle$ and $C \mapsto \langle 2, 2 \rangle$ (Hence $P_2 \rightarrow B \subset C$).

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:

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

Suppose we map $P_1 \mapsto \langle 1, 2 \rangle$ and $C \mapsto \langle 2, 2 \rangle$ (Hence $P_2 \rightarrow B \subset C$).

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

Interesting Questions



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

- When is an inference applicable?

Application Condition: –

Outline functions:

$\langle C \quad \text{compute-}=\text{subst-}m(\pi) \rangle$

$\langle P_1 \quad \text{compute-}=\text{subst-}m(\pi) \rangle$

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
⇒ Sufficient information for the proof planner
⇒ 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
⇒ Overcomes manual definition of the agents as in the old ΩMEGA system

[Autexier&Dietrich, MKM'06]

Next Steps



- Automation of proof search at the TASK LAYER
 - ▶ Reactivate Ω_{ANTS} 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 PLAT Ω

Mediating between Texteditors and Proof Assistance Systems

—

The PLAT Ω System

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

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

Plato at a Glance



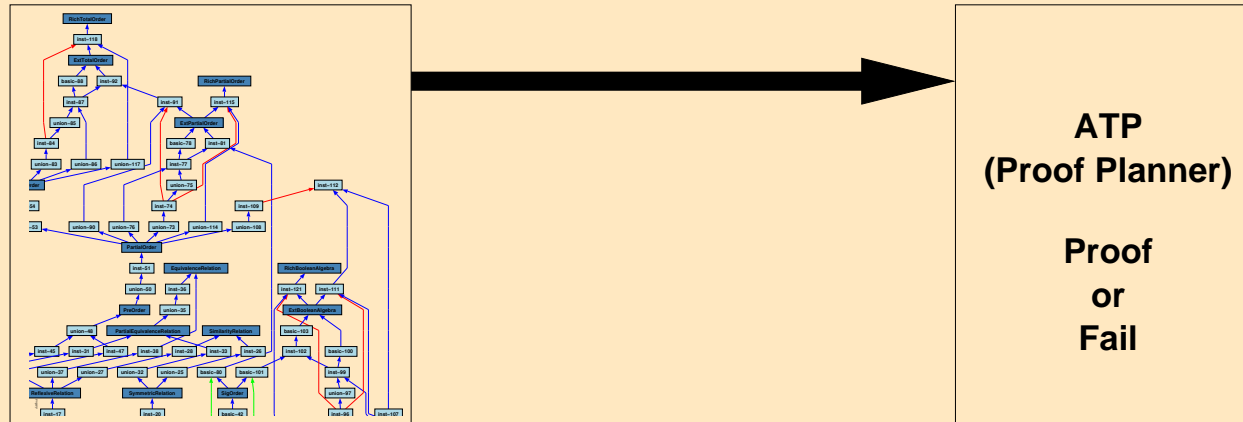
- The user authors his mathematical documents with a scientific WYSIWYG text-editor
- The primary task of PLATΩ (Marc Wagner, Diploma Thesis, ?June 2006?)
 - ▶ Maintain consistent formal (PA) and informal representations (Text-editor)
 - ▶ Relay service requests from the Text-editor to the PA
 - ▶ Propagate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

My work during the visit to the DREAM Group:

“Automated Reasoning in Large Structured Theories”

(EPSRC Visiting Researcher Project)

Motivation

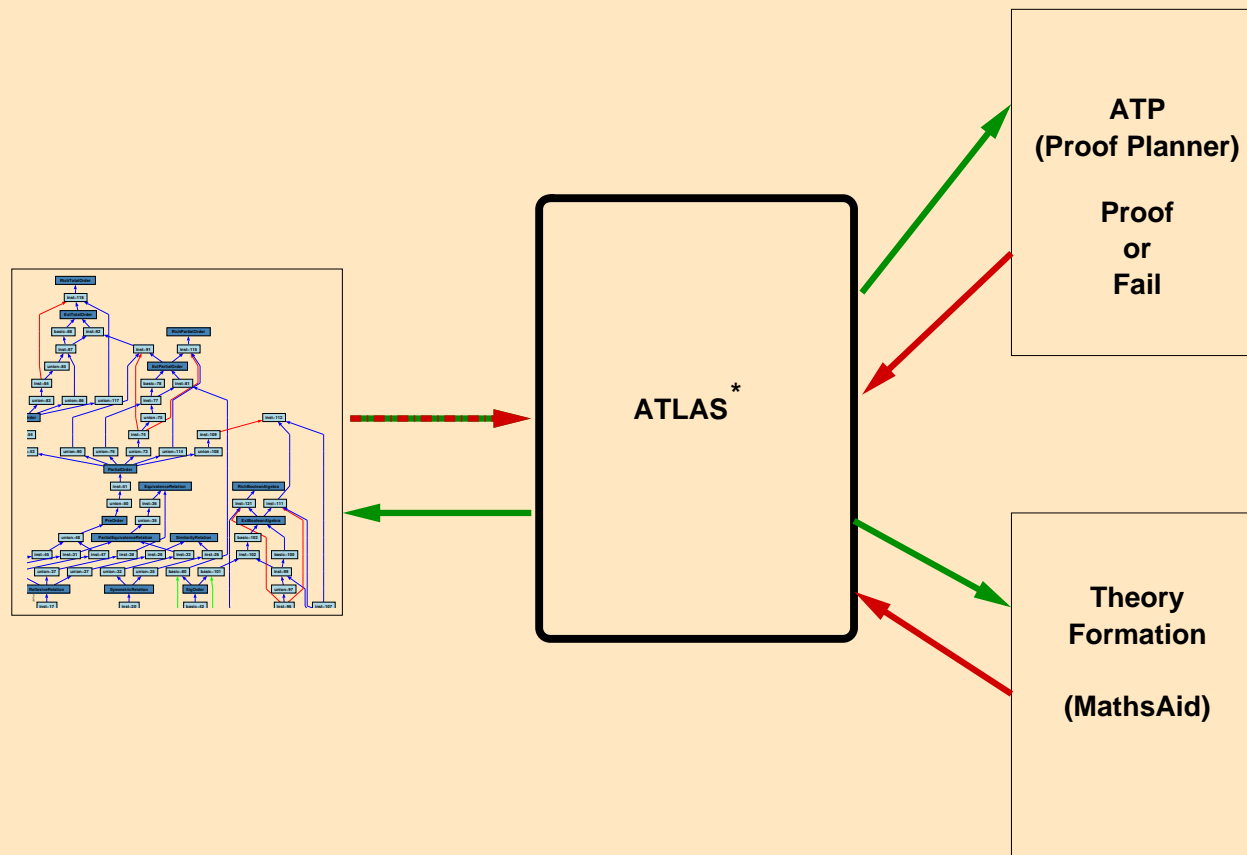


Failures of automatic proof search procedures (e.g. classical ATP, Proof Planner, ...) and reasons:

- 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)

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

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

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 Ω_{ANTS}
- PLAT Ω tool to plug Ω MEGA 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”

The initial Document in the Text-Editor



Theory [setdistr]

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

Theorem [Distributivity of Intersection] :

The following proposition follows at once from the definitions:

$$\forall A, B, C:\text{set} . (A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C))$$

Proof:

In order to prove this equation, we show that the following subset relations hold:

- (1) $\forall A, B, C:\text{set} . ((A \cap (B \cup C)) \subset ((A \cap B) \cup (A \cap C)))$
- (2) $\forall A, B, C:\text{set} . (((A \cap B) \cup (A \cap C)) \subset (A \cap (B \cup C)))$

Uploaded Theory in the MAYA system



CORE@kirana (Development: None)

Bigsys Development graph Logics Proof Interrupt Inka prover Tasklayer Theorem provers Help

Dgraph Proof

Development Graph

```

    graph BT
      set-theory[set-theory] --> setdistr[setdistr]
  
```

LML Proof Browser

File Help

Location:

Theory: setdistr

LOGIC: [hol+eq](#)

BASE TYPES:

CONSTANTS:

AXIOMS:

THEOREMS/LEMMATA:

Theorem DL#32

$$\pi (\lambda a. (\pi (\lambda b. (\pi (\lambda c. (\text{set= (intersection a (union b c)) (union (intersection a b) (intersection a c)))))))$$

Status: No proof [Prove](#) [Assume](#)

INCOMING LINKS:

[hol+eq](#) -> setdistr (global, axiom, Proved),
[set-theory](#) -> setdistr (global, axiom, Proved)

OUTGOING LINKS:

Output Message Error Warning Trace

```

*****
create new controlruleinterpreter
Create solution blackboard
Create control blackboard
Create Tasklayer with new (empty) taskforest
Create Tasklayer with new (empty) taskforest
Insert Element Basetype "PAI" into blackboard OBJ
Insert Element Core-Inference: "seteq" into blackb
Insert Element Core-Inference: "subset" into black
Insert Element Core-Inference: "intersect-" into b
Insert Element Core-Inference: "intersect+" into b
Insert Element Core-Inference: "union-" into black
Insert Element Core-Inference: "union+" into black
Insert Element Core-Inference: "same" into blackbo
  
```

Total: 8 Depth: 0 Command: Time: 0ms

Uploaded Proof in the TASK LAYER



CORE@kirana (Development: None)

Bigsys Development graph Logics Proof Interrupt Inka prover Tasklayer Theorem provers Help

Dgraph Proof

Map

LML Proof Browser

Label	Hypothesis	Term	Method	Premises
L1.2.1		==> nil (list (set= (inter		L1.2.1.1
L1.2.1.1			plato:proof	L1.2.2
L1.2.2		==> nil (list (π (λa.(π		L1.2.2.1
L1.2.2.1			plato:subgo	L1.2.3 L1.2.4 L1.
L1.2.3		==> nil (list (π (λa.(π		L1.2.3.1
L1.2.3.1			plato:done	
L1.2.4		==> nil (list (π (λa.(π		
L1.2.5		==> nil (list (π (λa.(π		

Pretty Term

```

==>
nil
(list
 (π
  (λa.(π
   (λb.(π
    (λc.(set=
     (intersection a (union b c))
     (union (intersection a b) (intersection a c)))))))
 (π
  (λa.(π
   (λb.(π
    (λc.( (intersection a (union b c))
           < (union (intersection a b) (intersection a c))))))))))

```

Output Message Error Warning Trace

```

*****
create new controlruleinterpreter
Create solution blackboard
Create control blackboard
Create Tasklayer with new (empty) taskForest
Create Tasklayer with new (empty) taskForest
Insert Element Basetype "PAI" into blackboard OBJ
Insert Element Core-Inference: "seteq" into blackb
Insert Element Core-Inference: "subset" into black
Insert Element Core-Inference: "intersect-" into b
Insert Element Core-Inference: "intersect+" into b
Insert Element Core-Inference: "union-" into black
Insert Element Core-Inference: "union+" into black
Insert Element Core-Inference: "same" into blackbo

```

Total: 8 Depth: 0 Command: Time: 0ms

User-Modified Document in the Text-Editor



Theory [setdistr]

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

Theorem [Distributivity of Intersection] :

The following proposition follows at once from the definitions:

$$\forall A, B, C:\text{set} . (A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C))$$

Proof:

In order to prove this equation, we show that the following subset relations hold:

$$(1) \forall A, B, C:\text{set} . ((A \cap (B \cup C)) \subset ((A \cap B) \cup (A \cap C)))$$

$$(2) \forall A, B, C:\text{set} . (((A \cap B) \cup (A \cap C)) \subset (A \cap (B \cup C)))$$

We prove first the proposition (1).

$$\text{It follows that } \forall A, B, C:\text{set } x:i. x \in (A \cap (B \cup C)) \Rightarrow x \in ((A \cap B) \cup (A \cap C))$$

Transformed User-Modified Proof in the TASK LAYER



OMEGA GROUP

CORE@kirana (Development: None)

Bigsys Development graph Logics Proof Interrupt Inka prover Tasklayer Theorem provers Help

Dgraph Proof

Map

LML Proof Browser

Label	Hypothesis	Term	Method	Premises
L1.2.1		==> nil (list (set= (inter		L1.2.1.1
L1.2.1.1			plato:proof	L1.2.2
L1.2.2		==> nil (list (π (λa.(π		L1.2.2.1
L1.2.2.1			plato:subgoal	L1.2.3 L1.2.4 L1.2.6
L1.2.3		==> nil (list (π (λa.(π		L1.2.3.1
L1.2.3.1			plato:done	
L1.2.4		==> nil (list (π (λa.(π		L1.2.4.1
L1.2.4.1			plato:proof	L1.2.6
L1.2.6		==> nil (list (π (λa.(π		L1.2.6.1
L1.2.6.1			plato:fact	L1.2.7
L1.2.7		==> (list (π (λa.(π (λb.(
L1.2.5		==> nil (list (π (λa.(π		

Pretty Term

```

==>
(list
  (π
    (λa.(π
      (λb.(π
        (λc.(π
          (λx.(impl
            (x ∈ (intersection a (union b c)))
            (x ∈ (union (intersection a b) (intersection a c))):
    (list
      (π
        (λa.(π
          (λb.(π
            (λc.(set=
              (intersection a (union b c))

```

Output Message Error Warning Trace

```

*****
create new controlruleinterpreter
Create solution blackboard
Create control blackboard
Create Tasklayer with new (empty) taskForest
Create Tasklayer with new (empty) taskForest
Insert Element Basetype "PAI" into blackboard OBJ
Insert Element Core-Inference: "seteq" into blackb
Insert Element Core-Inference: "subset" into black
Insert Element Core-Inference: "intersect-" into b
Insert Element Core-Inference: "intersect+" into b
Insert Element Core-Inference: "union-" into black
Insert Element Core-Inference: "union+" into black
Insert Element Core-Inference: "same" into blackbo

```

0 7 5 0 0 0 0 0 0 0 Total: 12 Depth: 0 Command: Time: 0ms

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:
 $\forall A, B, C:\text{set} . (A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C))$

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

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

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:
 $\forall A, B, C:\text{set} . (A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C))$

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

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

Patched Document in the Text-Editor



Theory [setdistr]

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

Theorem [Distributivity of Intersection] :

The following proposition follows at once from the definitions:

$$\forall A, B, C:\text{set} . (A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C))$$

Proof:

In order to prove this equation, we show that the following subset relations hold:

$$(1) \forall A, B, C:\text{set} . ((A \cap (B \cup C)) \subset ((A \cap B) \cup (A \cap C)))$$

$$(2) \forall A, B, C:\text{set} . (((A \cap B) \cup (A \cap C)) \subset (A \cap (B \cup C)))$$

We prove first the proposition (1).

$$\text{It follows that } \forall A, B, C:\text{set } x:i. x \in (A \cap (B \cup C)) \Rightarrow x \in ((A \cap B) \cup (A \cap C))$$

$$\forall A, B, C:\text{set } x:i. x \in A \wedge x \in (B \cup C) \Rightarrow x \in ((A \cap B) \cup (A \cap C))$$
