The New Proof Assistant \( \Omega \text{MEGA} \): Developments and Applications

Serge Autexier

(joint work with the \( \Omega \text{MEGA} \) 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 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 $\Omega$MEGA

- Proof-Planning
  (MULTI, E. Melis, A. Meier)
- Agent-based Theorem Proving
  ($\Omega$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)

Source: Autexier
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

---

Source: Autexier
- 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 \( \text{LOUI} \)
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)
    - \[\text{CORE-calculus that directly supports that} \] [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)
    - \( \Rightarrow \) 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
Why and What did we Change?

- Natural Deduction Calculus too cumbersome (not natural enough)
  - Want direct *reasoning with the assertions* (interactive & automatic)
    \[ \Rightarrow \text{CORE-calculus that directly supports that} \ [\text{Autexier, CADE’05}] \]

- The proof datastructure was completely overcrowded
  - New generic proof datastructure (PDS)
  - Definition of the *task-layer* as uniform reasoning platform

- GUI *\Omega 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)
    \[ \Rightarrow \text{CORE-calculus that directly supports that [Autexier, CADE’05]} \]

- The proof datastructure was completely overcrowded
  - New generic proof datastructure (PDS)
  - Definition of the *task-layer* as uniform reasoning platform

- GUI *LΩUT* 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)
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 L\(\Omega\)UT nice, but did not meet requirements of targeted users, say mathematicians
  - Plug \(\Omega\)MEGA 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 ΩMEGA
New Architecture

Source: Autexier
This Talk

Development

Graph

Task Layer

Proof Verbalization

Mathematical Repositories

Assertion Retrieval

Constraint Solver

Knowledge-based Proof Planning

Agent-oriented Theorem Proving

Developer GUI

Loui

OANTS

MULTI

Learning

Computation Transformation

Mathematical Services

Mathematical Repositories Proof

Verbalization

Core

constraint

Sapper

Rich Partial Order

Total Order

Ext Total Order

Mathematical Services

PreOrder

Partial Order

Ext Boolean Algebra

Boolean Algebra

Rich Boolean Algebra

Rat

Int

Nat

Development

Graph

Task Layer
Development graphs:
Maintaining structured mathematical theories
Context

Evolutionary Formal Development

Structured Specification

Theorem Prover

Structured Internal Representation

Changing specifications due to proof failures

Translation into logical representation

Proof Obligations + Structured Database

Source: Autexier
**Development Graphs**

**LIST[Elem]**

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

Local Axioms: · [] ++ K = K, ...

Local Lemmata:
- reverse(K ++ L) = reverse(L) ++ reverse(K)
- ...

**MONOID**

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

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

**[Hutter’00], [AutexierHutter’02&’05], [MossakowskiAutexierHutter’01&’06]**
Role of Development Graphs

Development Graph

\[ \text{Structured Logical Content of Specifications} + \text{Status of proof obligations} \]

\( \Rightarrow \) Save as much proof work as possible!

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

STP 2006, June 6, 2006 – p.11
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 (⇔ Proof Planning, ΩANTS)
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 (⇔ Proof Planning, ΩANTS)
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)
    - 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

⊢ ¬rat(\sqrt{12})

Sketch

rat(\sqrt{12}) ⊢ ⊥

Sketch

rat(\sqrt{12}), \text{int}(n), \text{int}(m), \neg \text{commondiv}(n, m), \sqrt{12} = \frac{n}{m} ⊢ ⊥

Sketch

Σ ⊢ \text{div}(n, 3) \land \text{div}(m, 3)

Sketch

Σ ⊢ \text{div}(n, 3)

Sketch

Σ ⊢ \text{div}(m, 3)

Sketch

Σ ⊢ \text{div}(n, 2)

Sketch

Σ ⊢ \text{div}(m, 2)

Sketch

Σ ⊢ \text{div}(m, 2)
Example Complete PDS

Complete PDS

⊢ ¬\(\text{rat}(\sqrt{12})\)

Sketch

⊢ \(\perp\)

ApplyLemma(Rat – Criterion)

rat(\(\sqrt{12}\)) \(\vdash \perp\)

Decomposition

rat(\(\sqrt{12}\)), \(\exists y: \text{int}. z: \text{int}. \sqrt{12} = \frac{y}{z} \land \neg \text{commondiv}(y, z) \vdash \perp\)

Sketch

Σ \(\vdash \text{div}(n, 3) \land \text{div}(m, 3)\)

Sketch

Σ \(\vdash \text{div}(n, 2) \land \text{div}(m, 2)\)

Sketch

Σ \(\vdash \text{div}(n, 3)\)

Σ \(\vdash \text{div}(m, 3)\)

Σ \(\vdash \text{div}(n, 2)\)

Σ \(\vdash \text{div}(m, 2)\)
Example Complete PDS

Complete PDS

\[ \vdash \neg \text{rat}(\sqrt{12}) \]

Sketch

\[ \text{rat}(\sqrt{12}) \vdash \bot \]

ApplyLemma(Rat – Criterion)

Sketch

\[ \text{rat}(\sqrt{12}), \exists y : \text{int}, z : \text{int}, \sqrt{12} = \frac{y}{z} \land \neg \text{commondiv}(y, z) \vdash \bot \]

Decomposition

Sketch

\[ \Sigma \vdash \text{div}(n, 3) \land \text{div}(m, 3) \]

\[ \Sigma \vdash \text{div}(n, 2) \land \text{div}(m, 2) \]

Sketch

\[ \Sigma \vdash \text{div}(n, 3) \]

\[ \Sigma \vdash \text{div}(m, 3) \]

\[ \Sigma \vdash \text{div}(n, 2) \]

\[ \Sigma \vdash \text{div}(m, 2) \]

A PDS View

\[ \vdash \neg \text{rat}(\sqrt{12}) \]

Sketch

\[ \text{rat}(\sqrt{12}) \vdash \bot \]

ApplyLemma(Rat – Criterion)

Sketch

\[ \text{rat}(\sqrt{12}), \exists y : \text{int}, z : \text{int}, \sqrt{12} = \frac{y}{z} \land \neg \text{commondiv}(y, z) \vdash \bot \]

Decomposition

Sketch

\[ \Sigma \vdash \text{div}(n, 3) \land \text{div}(m, 3) \]

\[ \Sigma \vdash \text{div}(n, 2) \land \text{div}(m, 2) \]

Sketch

\[ \Sigma \vdash \text{div}(n, 3) \]

\[ \Sigma \vdash \text{div}(m, 3) \]

\[ \Sigma \vdash \text{div}(n, 2) \]

\[ \Sigma \vdash \text{div}(m, 2) \]
PDSs

- Alternative subproofs require to allow for alternative substitutions of Metavariabes $\Rightarrow$ Mechanism to support that efficiently
  
  [Dietrich, Diploma Thesis, 2006]

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

  [Autexier&Sacerdoti-Coen, MKM’06]

($\lambda\mu\tilde{\mu}$-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 $\Omega$MEGA system there were

- $(\Omega$MEGA-)tactics (interactive theorem proving)
- methods (proof planning)

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

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

Application Condition: –
Outline functions:
\[ \langle C \ compute=\text{subst-prem}(P_1, P_2, \pi) \rangle \]

- Having $P_1$ and $P_2$, we can get $C$ (fw)
- Having $C$ and $P_2$, we can get $P_1$ (bw1)

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

Application Condition: –
Outline functions:
\[ \langle P_1 \ compute=\text{subst-prem}(C, P_2, \pi) \rangle \]

- 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:**

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

There were 4 tactics and 2 methods in the old \(\text{OMEGA}\) system!

- Inferences unify (\(\text{OMEGA}\)-)tactics and methods, overcome manual specification of ADs
Examples of Inferences

\[
P_1 : A \subseteq B \quad P_2 : B \subseteq C
\]

\[
C : A \subseteq C
\]

Application Condition: –
Examples of Inferences

\[ \epsilon > 0, D > 0, 0 < |x - A|, |x - A| < D \]

\[ \vdash P : |F(x) - L| < \epsilon \]

\[ \frac{}{C : \lim_{A} F = L} \]

Limit(\(\epsilon, x\))

Application Condition: \( \text{EV}(\epsilon, \{F, A, L\}) \land \text{EV}(x, \{F, A, L, D\}) \)
Examples of Inferences

\[
P : O(U, V) \\
\frac{\text{Maple-Simplify}}{C : O(U', V')}
\]

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

\[
\begin{aligned}
\left[ H_1^1, \ldots, H_{n_1}^1 \right] & \quad \cdots \quad & \left[ H_1^k, \ldots, H_{n_k}^k \right] \\
\vdots & \quad \cdots \quad & \vdots \\
P_1 : F_1 & \quad \cdots \quad & P_k : F_k \\
C_1 : G_1 & \cdots \quad & C_m : G_m
\end{aligned}
\]

Name(\(\omega_1, \ldots, \omega_n\))

Outline Functions: \(\langle i^1, f^1(i_1^1, \ldots, i_{j_1}^1) \rangle\)  
Application Conditions: \(P(i_1^0, \ldots, i_m^0)\)
General Form of Inferences

\[
\begin{align*}
\left[ H_1^{1}, \ldots, H_{n_1}^{1} \right] & \quad \left[ H_1^{k}, \ldots, H_{n_k}^{k} \right] \\
\vdots & \quad \vdots \\
P_1 : F_1 & \quad \cdots \quad P_k : F_k \\
C_1 : G_1 & \quad \cdots \quad C_m : G_m \\
\name(\omega_1, \ldots, \omega_n)
\end{align*}
\]

Outline Functions: \( \langle i^1, \text{f}^1(i_1^1, \ldots, i_j^1) \rangle \)

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

- Procedure to \textit{automatically synthesise inferences} from arbitrary formulas contained in the mathematical theories.

\[
\forall A, B, C : \text{Set.} (A \subseteq B \land B \subseteq C) \Rightarrow (A \subseteq C)
\]

\[
\forall f, a, \epsilon. \epsilon > 0 \Rightarrow \exists \delta. \delta > 0 \Rightarrow \forall x. (0 < |x - a| \land |x - a| < \delta) \Rightarrow |f(x) - l| < \epsilon
\]

\[
\Rightarrow \lim_{a \to l} f = l
\]

(Had \textit{all} to be \textit{encoded manually} in the old \(\Omega\)MEGA system...)

- Only the other inferences must be manually encoded (e.g. \(=\text{subst-m}\))
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:

\[
P \Rightarrow (A \subset B) \vdash Q \Rightarrow (A \subset C)
\]

\[
P_1 : (A \subset B) \quad P_2 : (B \subset C)
\]

\[
\frac{P_1 \quad P_2}{C : (A \subset C)} \text{ Subset}
\]
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:

\[
P \Rightarrow (A \subset B) \vdash Q \Rightarrow (A \subset C)
\]

\[
\begin{align*}
P_1 & : (A \subset B) \\
P_2 & : (B \subset C) \\
C & : (A \subset C)
\end{align*}
\]

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:

\[ P \Rightarrow (A \subset B) \vdash Q \Rightarrow (A \subset C) \]

\[ \begin{align*}
P_1 & : (A \subset B) \\
C & : (A \subset C) \\
\end{align*} \]

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

\[ P \Rightarrow (A \subset B) \vdash Q \Rightarrow (P \land (B \subset C)) \]
Interesting Questions

■ 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
⇒ 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

Source: Autexier

[Autexier & Dietrich, MKM’06]
Next Steps

- Automation of proof search at the Task Layer
  - Reactivate \( \Omega\text{ANTS} \) on that basis
  - Reimplement proof planner \( \text{MULTI} \)

- 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 \( \text{TEXMACS} \) via \( \text{PLAT\Omega} \)
This Talk

TeXmacs Document

Theorem 1. $\sqrt{2}$ is irrational.

Proof. Assume $\sqrt{2}$ is rational, that is, there exist natural numbers $n$ and $m$ with no common divisor such that $\sqrt{2} = \frac{n}{m}$. Then $n\sqrt{2} = m$, and thus $2n^2 = m^2$. Hence $n^2$ is even, since odd numbers squared are odd, and hence $n$ is even; say $n = 2k$. Then $2k^2 = 2k^2$, that is, $n^2 = 2k^2$. Thus, $n^2$ is even too, and so is $n$. That means that both $n$ and $m$ are even, contradicting the fact that they do not have a common divisor.
Mediating between Texteditors and Proof Assistance Systems

–

The PLATO 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
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)
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

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
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”
**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
Uploaded Proof in the Task Layer
User-Modified Document in the Text-Editor

---

Theory \( [\text{setdistr}] \)

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

**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 \( \forall A, B, C: \text{set} \): \( x \in (A \cap (B \cup C)) \Rightarrow x \in ((A \cap B) \cup (A \cap C)) \)
Transformed User-Modified Proof in the Task Layer
Menu Interaction in the Text-Editor

<table>
<thead>
<tr>
<th>Theory [setdistr]</th>
</tr>
</thead>
</table>

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

We prove first the proposition (1).

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

- Apply same
- Apply union+
- Apply union-
- Apply intersect+: Arguments: Compute Instantiations
- Apply intersect-
- Apply subset
### 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. \(\forall A, B, C:\text{set} . \ ((A \cap (B \cup C)) \subseteq ((A \cap B) \cup (A \cap C)))\)
2. \(\forall A, B, C:\text{set} . \ ((A \cap B) \cup (A \cap C)) \subseteq (A \cap (B \cup C))\)

We prove first the proposition (1).

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

<table>
<thead>
<tr>
<th>Apply same</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply union+</td>
</tr>
<tr>
<td>Apply union-</td>
</tr>
<tr>
<td>Apply intersect+</td>
</tr>
<tr>
<td>P1 (\iff x \in A \land x \in (B \cup C))</td>
</tr>
<tr>
<td>C (\iff x \in (A \cap (B \cup C)))</td>
</tr>
<tr>
<td>GSETB (\mapsto \ (B \cup C))</td>
</tr>
<tr>
<td>GSETA (\mapsto \ A)</td>
</tr>
<tr>
<td>X (\mapsto \ x)</td>
</tr>
<tr>
<td>Apply intersect-</td>
</tr>
<tr>
<td>Apply subset</td>
</tr>
</tbody>
</table>
System-Modified Proof in the TASK LAYER

---

Source: Autexier

---

STP 2006, June 6, 2006 – p.45
Context: We are going to prove a simple theorem in the theory of sets (\(\rightarrow\text{[set-theory]}\))

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

We prove first the proposition (1).

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

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