A Generic Modular Data Structure for Proof Attempts Alternating on Ideas and Granularity

Serge Autexier, Christoph Benzmüller, Dominik Dietrich, Andreas Meier, Claus-Peter Wirth

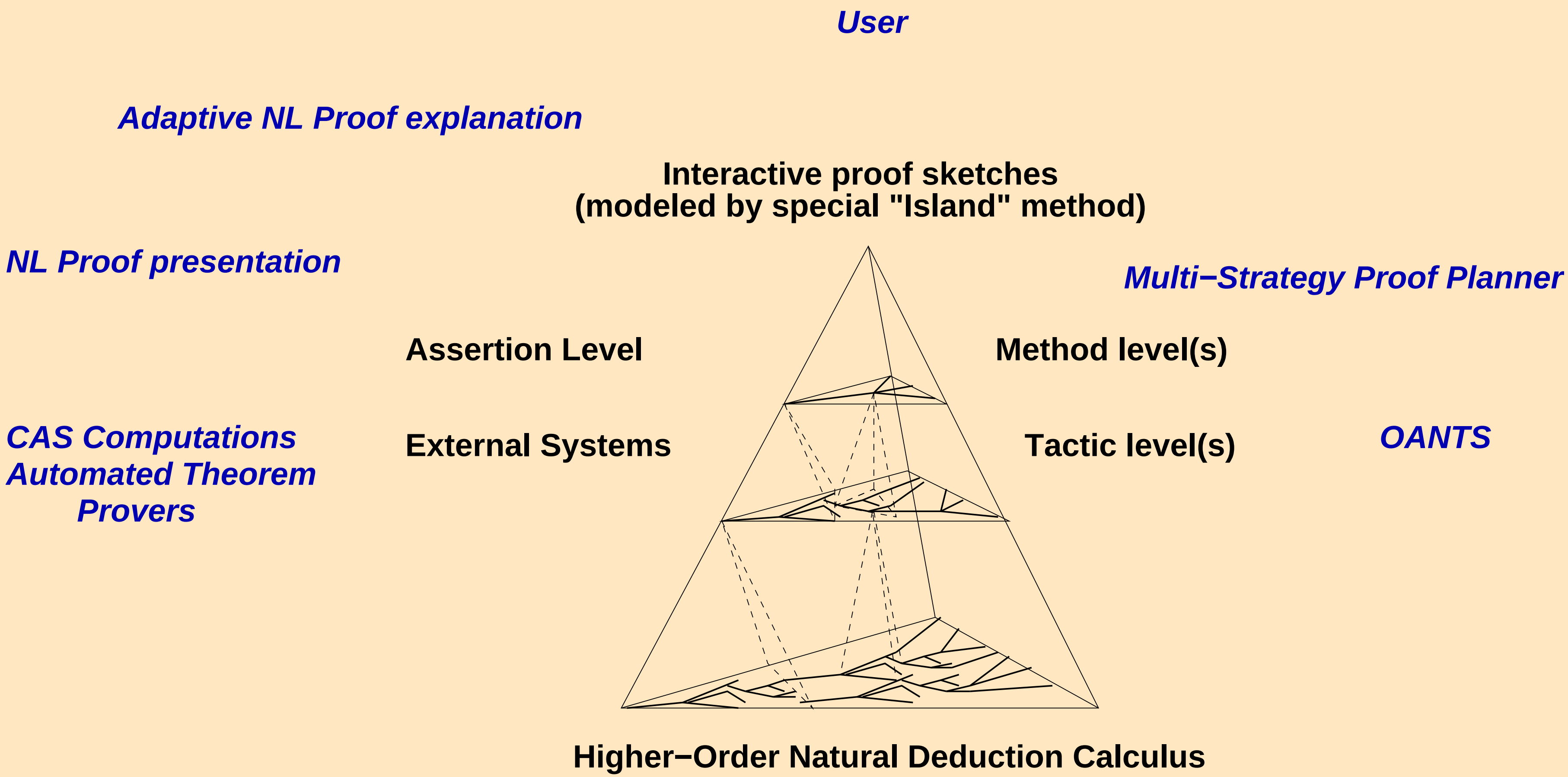
serge@ags.uni-sb.de

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

MKM'05, July 15th 2005

IU Bremen, Germany

ΩMEGA's Old PDS



Source: Autexier

Features

- Simultaneous representation of the proofs at different levels of granularity

Features

- Simultaneous representation of the proofs at different levels of granularity
  - Representation of abstract proof ideas and their refinement *(Proof Planning)*

Source: Autexier

Features

- Simultaneous representation of the proofs at different levels of granularity
  - Representation of abstract proof ideas and their refinement *(Proof Planning)*
  - Representation of external systems proofs/computations and their refinement

Features

- Simultaneous representation of the proofs 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)

Source: Autexier

Features

- Simultaneous representation of the proofs 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

Features

- Simultaneous representation of the proofs 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

Source: Autexier

Features

- Simultaneous representation of the proofs 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

Source: Autexier

Drawbacks

- No alternative proof attempts (on the same level of granularity)

Drawbacks

- No alternative proof attempts (on the same level of granularity)
  - Alternatives represented internally to the algorithms/programming language

Drawbacks

- No alternative proof attempts (on the same level of granularity)
  - Alternatives represented internally to the algorithms/programming language
  - No means to communicate to other “participants”

Source: Autexier

Drawbacks

- No alternative proof attempts (on the same level of granularity)
  - Alternatives represented internally to the algorithms/programming language
  - No means to communicate to other “participants”
- No good support for lemmatization

Source: Autexier

Drawbacks

- No alternative proof attempts (on the same level of granularity)
  - Alternatives represented internally to the algorithms/programming language
  - No means to communicate to other “participants”
- No good support for lemmatization
- Almost impossible to exchange the base calculus and have something different than the methods and tactics as abstract justifications

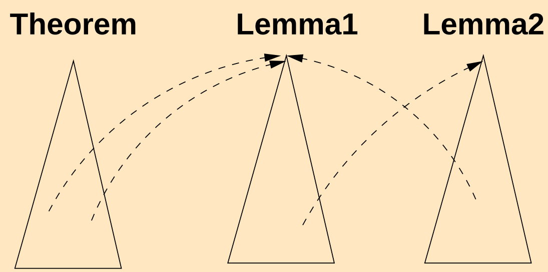
QuodLibet's Proof Representation

Quodlibet

- Tactic-based inductive theorem prover specialized on induction in the style of Descente infinie
  [Avenhaus,Kühler,Schmidt-Samoa,Wirth]

The quodlibet proof representation

- Alternative proof attempts (OR-branching)
- Support for lemmatization by
  - forests of proof trees
  - links between proof trees



Source: Autexier

Goals for the New PDS

- Preserve hierarchical representation of the proof at different granularity
- Support representation of alternative proof ideas
- Be independent of specific justifications and content of node
- Support for lemmatization

Source: Autexier

Generic PDS Node & Justifications

- Each PDS node has a *content* c

  For instance:

  - a single-conclusion sequent $\Gamma \Longrightarrow \varphi$.
  - tasks $\varphi_1, \ldots \varphi_n \vdash \underline{\psi_1}, \psi_2, \ldots, \psi_m$: multi-conclusion sequents with a selected focus of attention

Source: Autexier

Generic PDS Node & Justifications

- Each PDS node has a *content* c

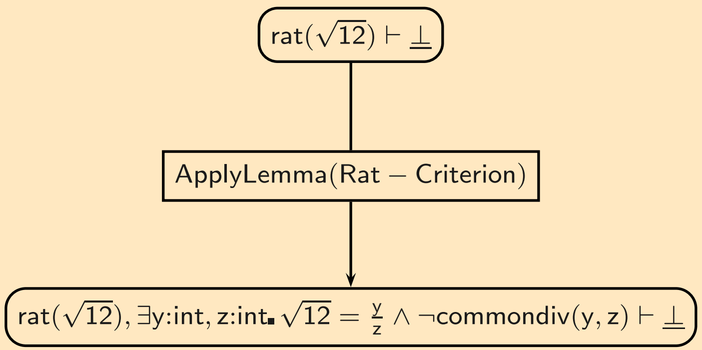
  For instance:

  - a single-conclusion sequent $\Gamma \Longrightarrow \varphi$.
  - tasks $\varphi_1, \ldots \varphi_n \vdash \underline{\psi_1}, \psi_2, \ldots, \psi_m$: multi-conclusion sequents with a selected focus of attention

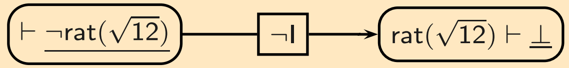
- A PDS justification links a PDS node to a set of PDS nodes and is annotated with information about the used reasoning technique

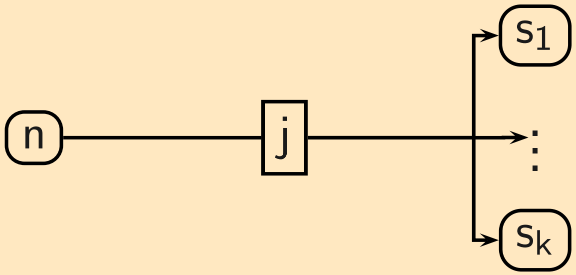
  For instance:

  - Some lemma



Generic PDS Node & Justifications

- A rule of the base calculus: 

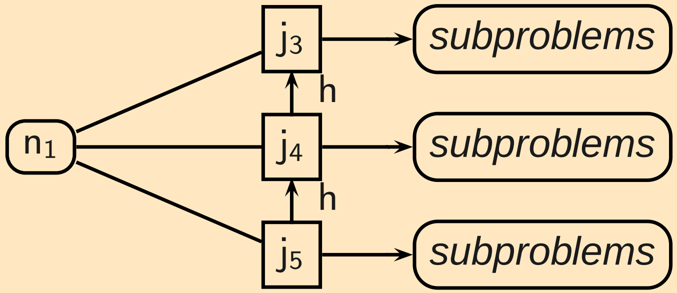
- In general 

"Given justifications for $s_1, \ldots, s_k$, j *justifies* n*"*

Alternatives

- Vertical alternatives: *Layers of granularity*

  Alternative justifications at different layers of granularity

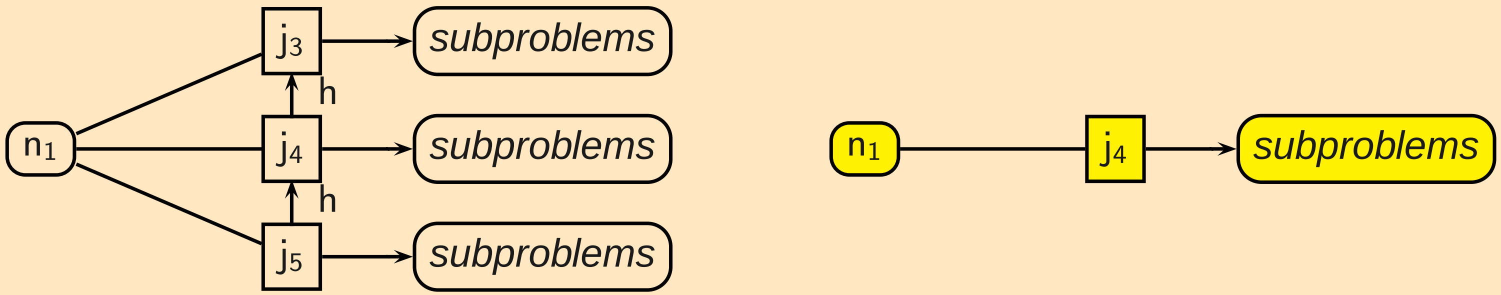


  - Totally ordered set of justifications. **1:1**

Alternatives

- Vertical alternatives: *Layers of granularity*

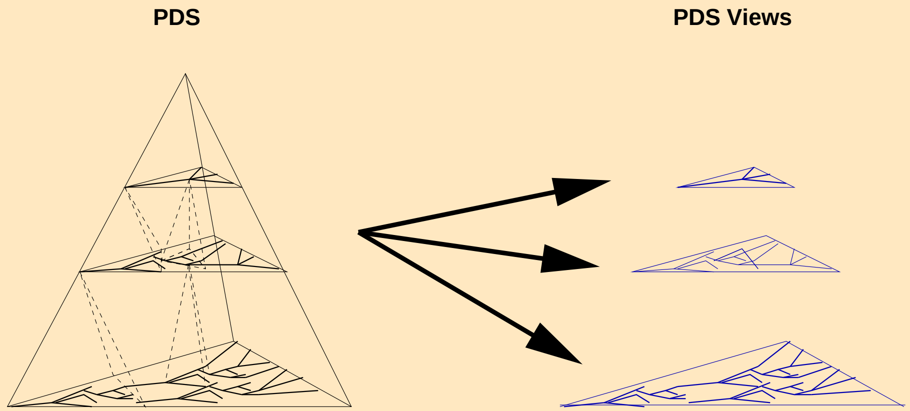
  Alternative justifications at different layers of granularity



  - Totally ordered set of justifications. **1:1**
  - Select a layer of granularity by selecting a justification.

Selection of a level of Granularity

Selecting *one* justification for each node . . .



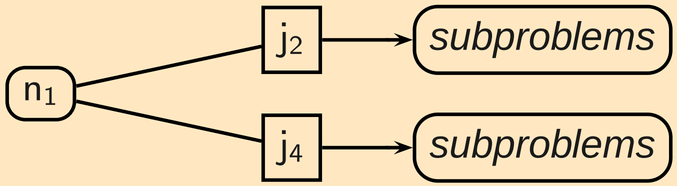
. . . determines a specific layer of granularity to view the PDS

$\Longrightarrow$ (Old ΩMEGA PDS)

Alternatives

- Horizontal alternatives:

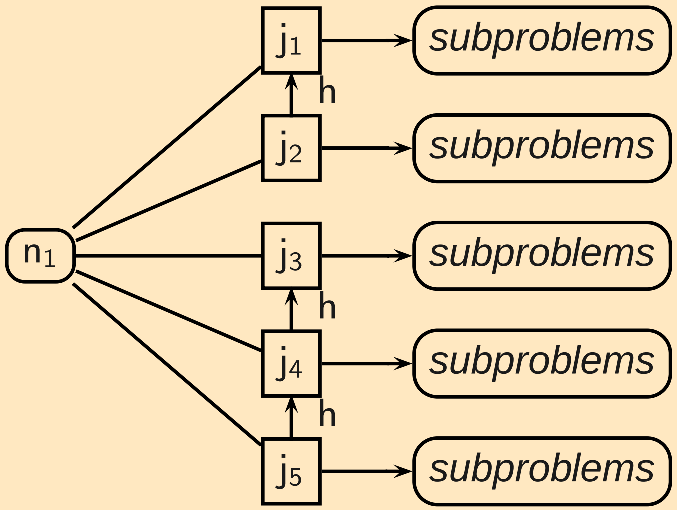
  Alternative proof ideas on the same level of granularity



  Unordered set of justifications

Bringing Alternatives together

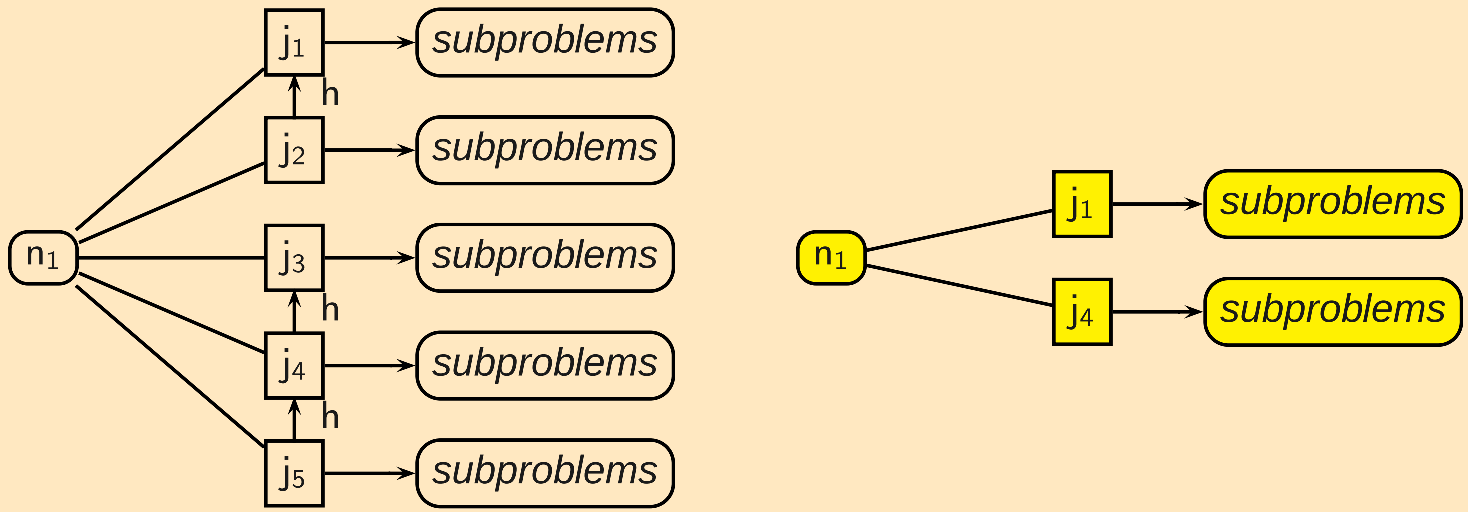
The simple approach:



- Disjoint sets of totally ordered justifications.

Bringing Alternatives together

The simple approach:



- Disjoint sets of totally ordered justifications.
- Select layer of granularity by selecting *one* justification from *each* set.

Source: Autexier

What cannot be modeled yet. . .

- We cannot model alternative refinements of a *same* abstract justification **n:1**

  For instance: Abstract justification "*By induction*" cannot be refined

What cannot be modeled yet. . .

- We cannot model alternative refinements of a *same* abstract justification **n:1**

  For instance: Abstract justification “*By induction*” cannot be refined

  - by using different induction orderings or

Source: Autexier

What cannot be modeled yet. . .

- We cannot model alternative refinements of a *same* abstract justification **n:1**

  For instance: Abstract justification "*By induction*" cannot be refined

  - by using different induction orderings or
  - doing the induction proof in different base logics

What cannot be modeled yet. . .

- We cannot model alternative refinements of a *same* abstract justification **n:1**

  For instance: Abstract justification "*By induction*" cannot be refined

  - by using different induction orderings or
  - doing the induction proof in different base logics

- We cannot share common initial proof sequences among the refinements of different abstract justifications **1:m**

  For instance: Sharing a same initial *simplification* tactic among the refinements of alternative, high-level proof attempts

Source: Autexier

What cannot be modeled yet. . .

- We cannot model alternative refinements of a *same* abstract justification **n:1**

  For instance: Abstract justification “*By induction*” cannot be refined

  - by using different induction orderings or
  - doing the induction proof in different base logics

- We cannot share common initial proof sequences among the refinements of different abstract justifications **1:m**

  For instance: Sharing a same initial *simplification* tactic among the refinements of alternative, high-level proof attempts

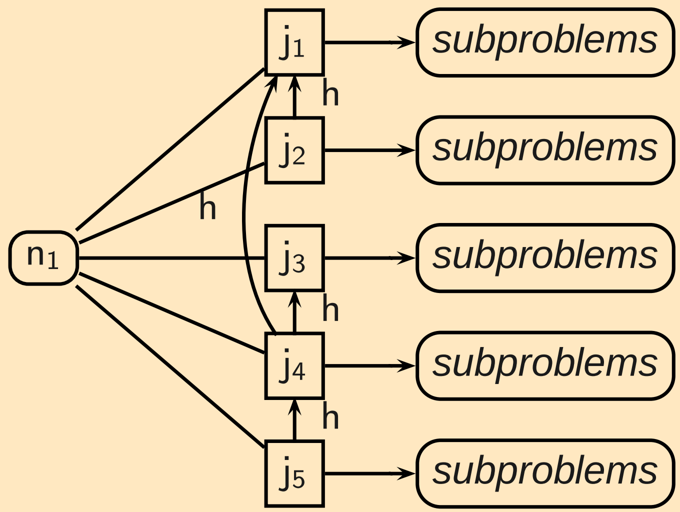
- To support this, we have to allow for *a single* set of *partially ordered* justifications

  (instead of *disjoint sets* of *totally ordered* justifications)

Source: Autexier

Bringing Alternatives together

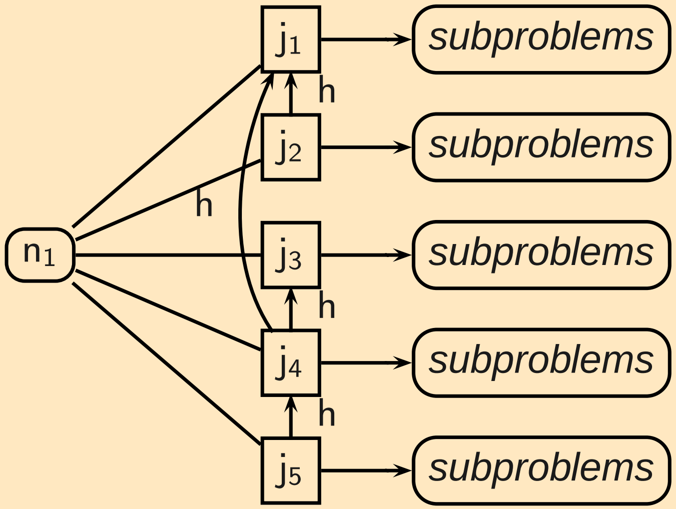
The advanced approach:



- A single set of partially ordered set of justifications **n:m**

Bringing Alternatives together

The advanced approach:

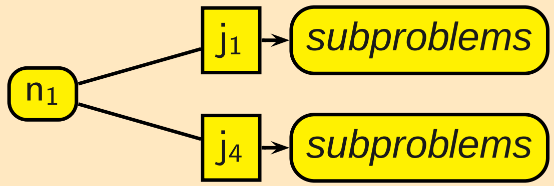


- A single set of partially ordered set of justifications **n:m**
- How to consistently select a layer of granularity?

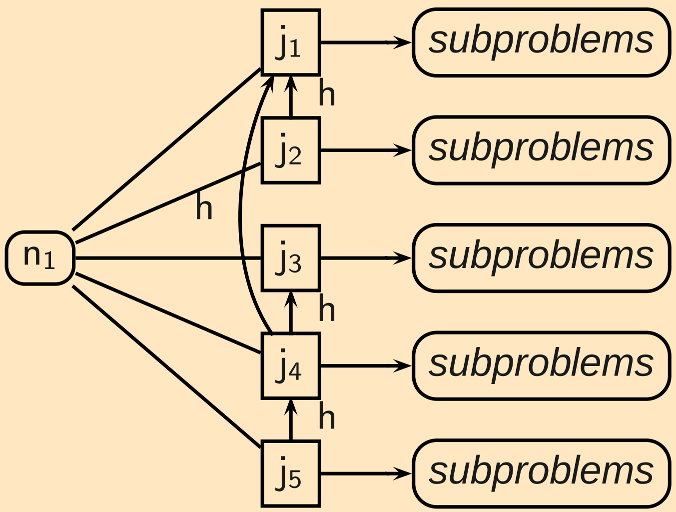
Source: Autexier

Bringing Alternatives together

The advanced approach:



? (not adequate)

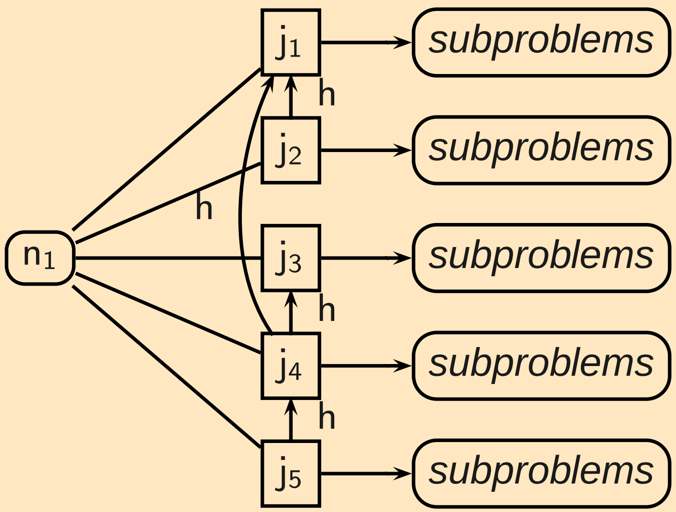


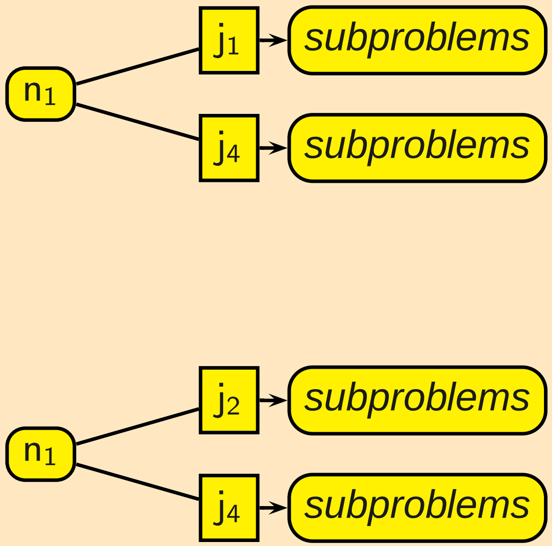
- A single set of partially ordered set of justifications **n:m**
- How to consistently select a layer of granularity?

Source: Autexier

Bringing Alternatives together

The advanced approach:





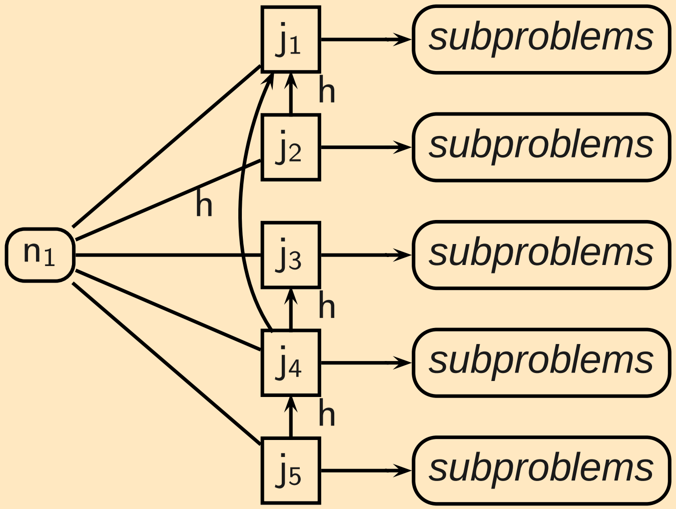
? (not adequate)

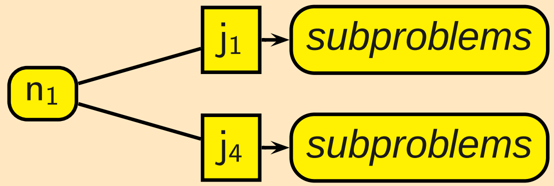
- A single set of partially ordered set of justifications **n:m**
- How to consistently select a layer of granularity?

Source: Autexier

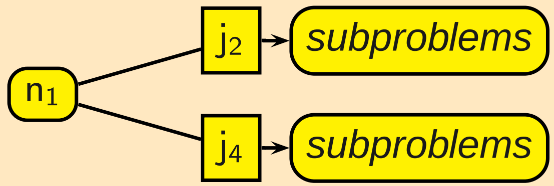
Bringing Alternatives together

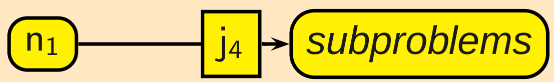
The advanced approach:





? (not adequate)





? (not complete)

- A single set of partially ordered set of justifications **n:m**
- How to consistently select a layer of granularity?

Source: Autexier

Formally: Sets of Alternatives

Assume a straightforward mathematical formalization of a PDS as an acyclic graph with justifications as hyperlinks and hierarchical links among justifications.

Let [...] $A \subseteq O_n$ *[be] a set of justifications for* n*.*

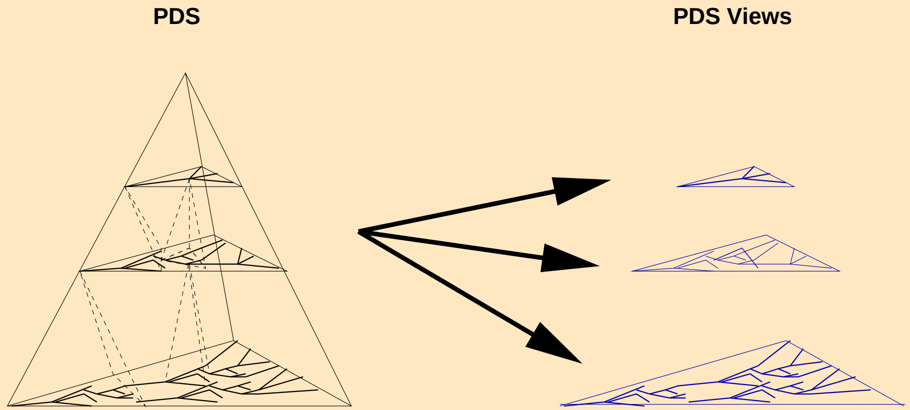
- A *is adequate if there are no* $k, k' \in A$ *such that* $k < k'$*.*
- A *is complete if for all* $k \in O_n$ *there is a* $k' \in A$ *such that* $k \leq k'$ *or* $k' \leq k$*.*

A *is a set of alternatives for* n *if it is adequate and complete.*

Source: Autexier

Selection of A level of Granularity

Fix a set of alternatives for each node of the PDS. . .



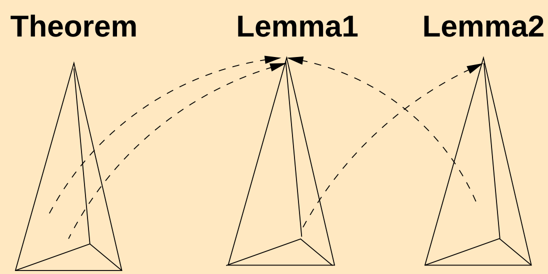
. . . gives you a proof on a specific granularity *including all alternative proof ideas* with that granularity.

Goals for the New PDS

- Preserve hierarchical representation of the proof at different granularity
- Support representation of alternative proof ideas
- Be independent of specific justifications and content of node
- Support for lemmatization

Support for Lemmatization

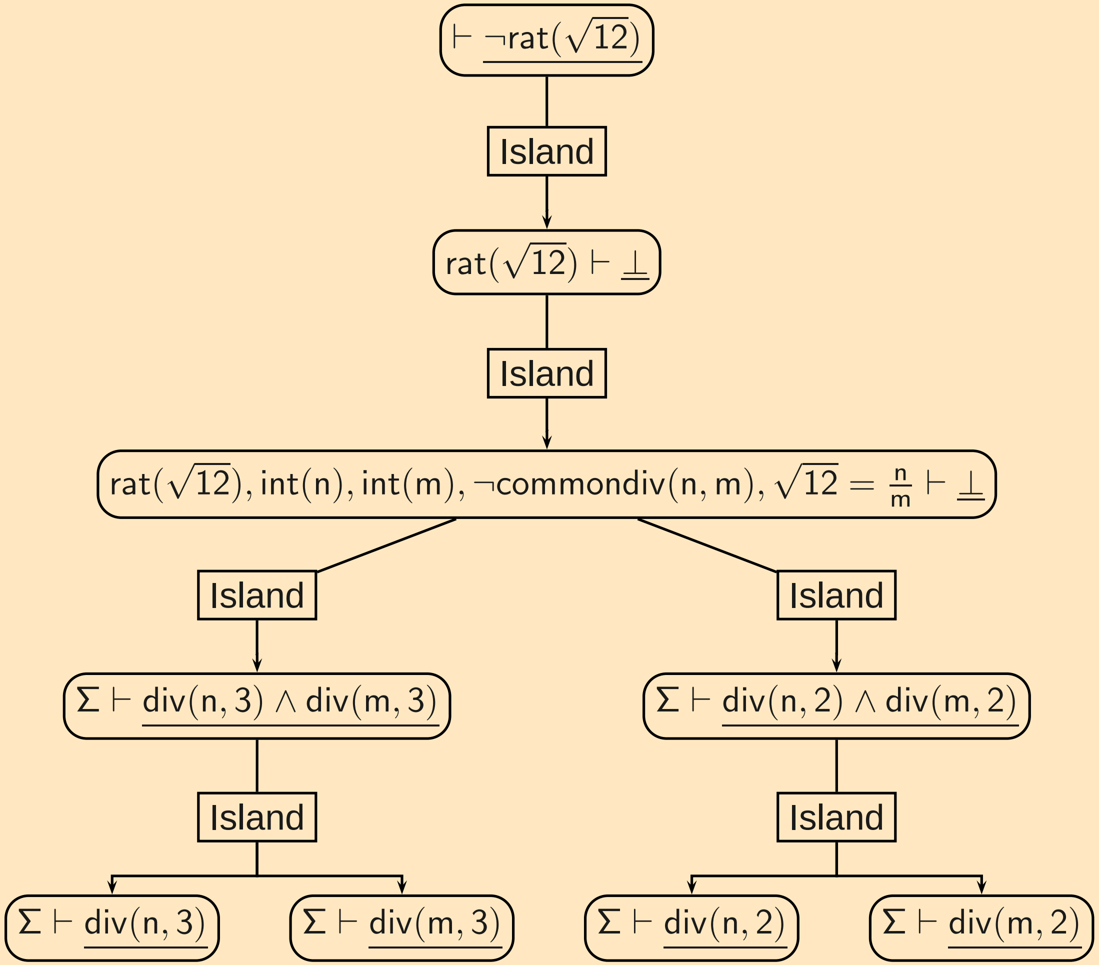
- Make a *forest* of PDSs
- Allow *inter-PDS-edges* (forest-edges) from a justification to some root node of a PDS



Intuitively: the lemma of the referenced PDS is used in the justification

- *Forest-View* is a “forest” of PDS-views (consistent)

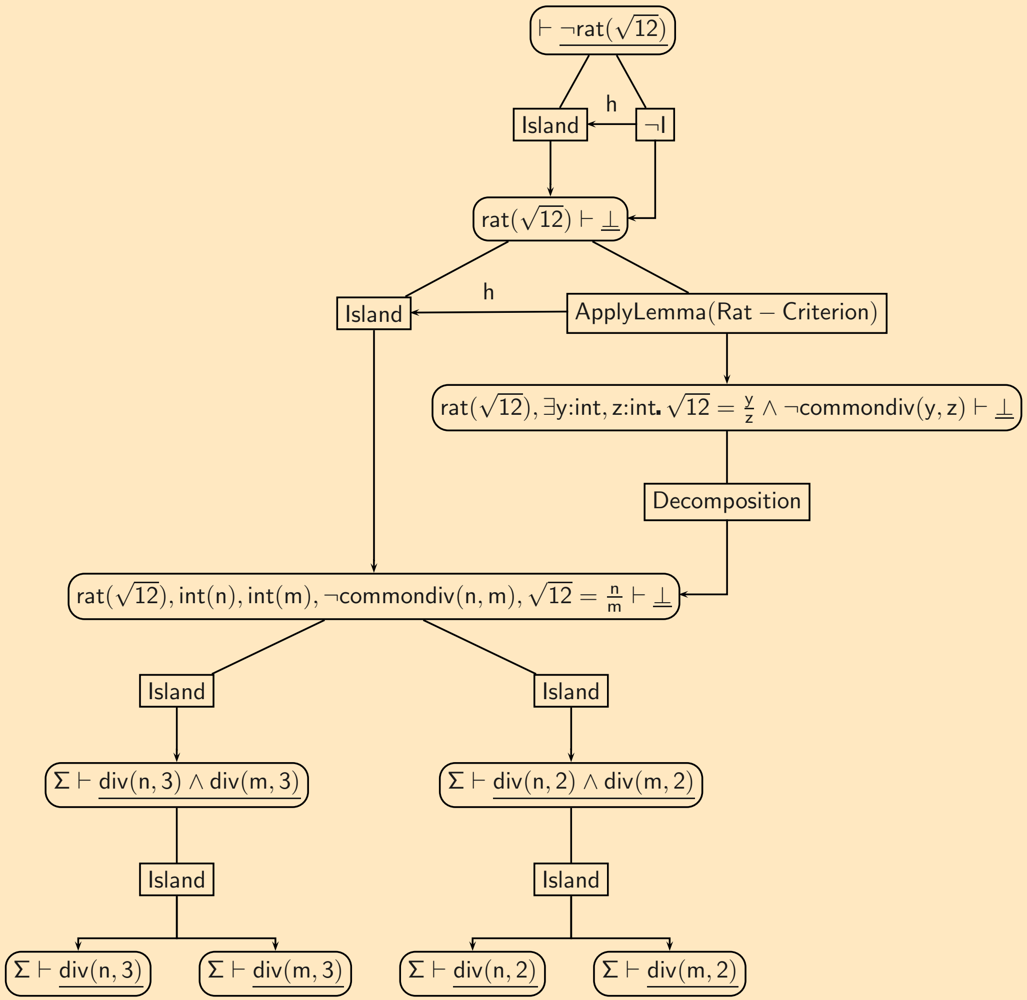
Example Abstract PDS



Source: Autexier

Example Complete PDS

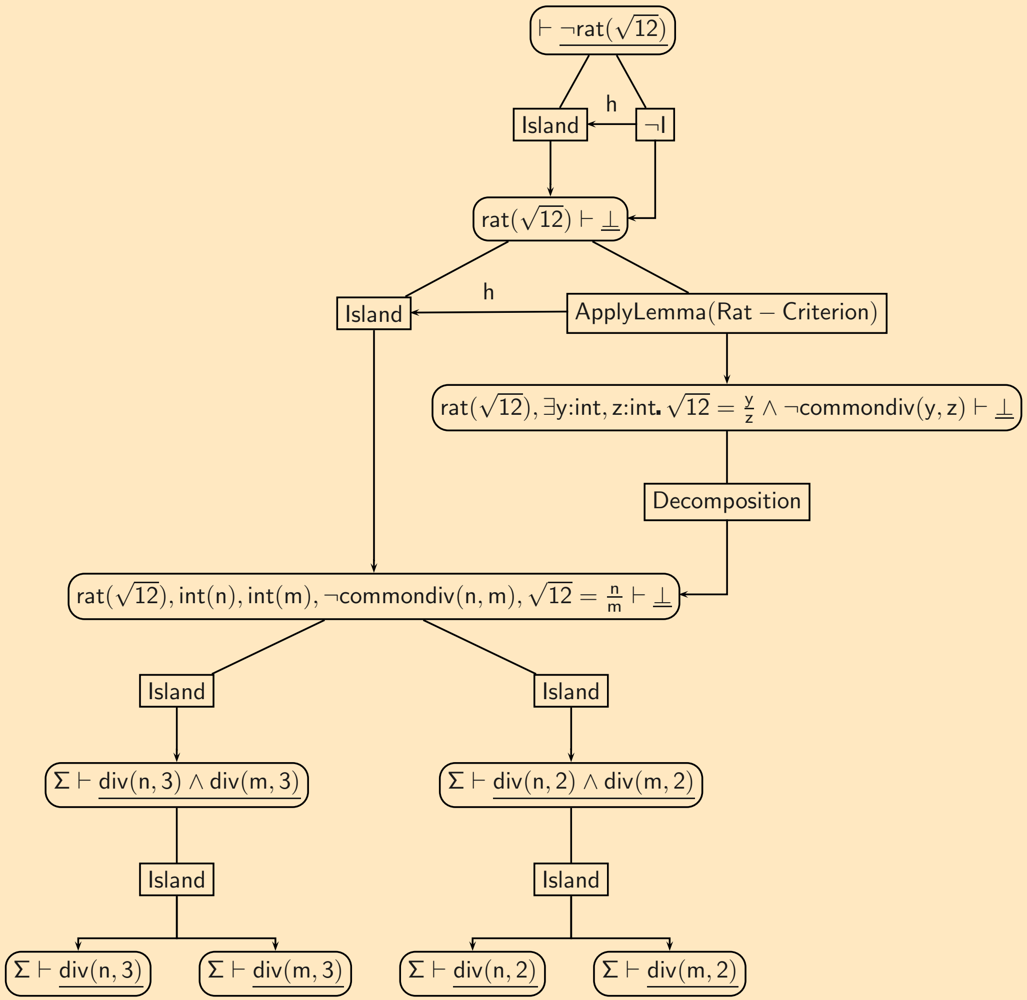
Complete PDS



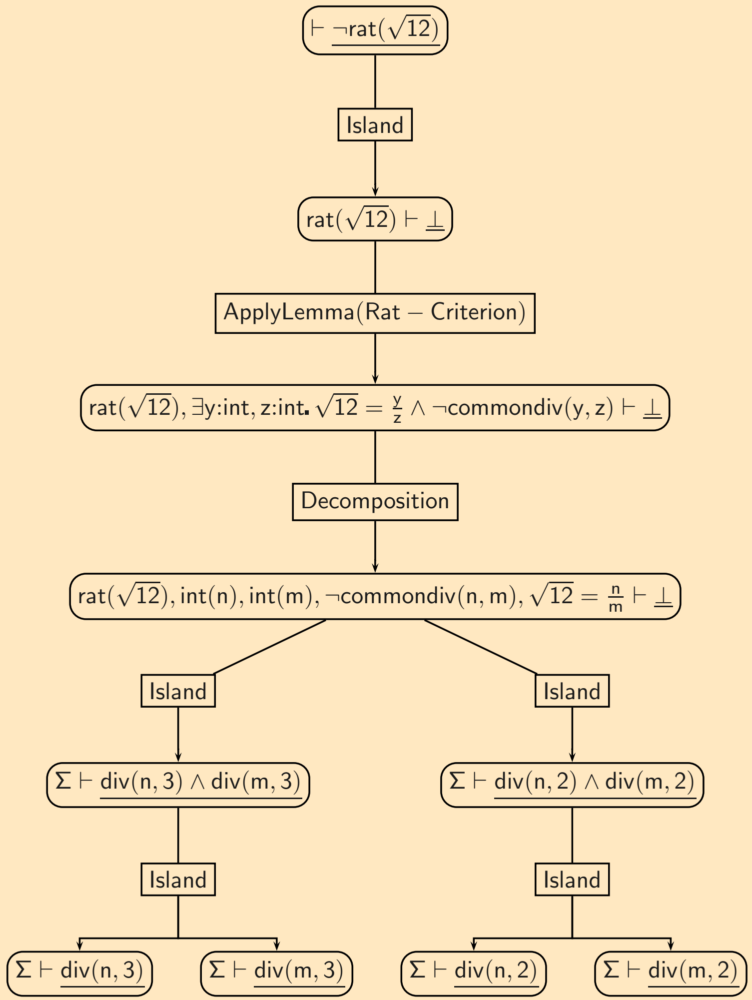
Source: Autexier

Example Complete PDS

Complete PDS



A PDS View



Source: Autexier

Implementation

- Implemented the generic PDS in Common Lisp
  - Basic functionality to introduce new justifications and changing the view
  - Provides dependency directed pruning for backtracking
  - Parameterized over generic classes for content of nodes and justifications
- Defined a content independent XML format for exporting and importing forests, trees, or parts of them.
- Storing proofs, alternative proofs, proofs under construction in our *Mathematical Knowledge Base*

Source: Autexier

XML Representation for the PDS

Parameterized over *node and justification contents*

```
<!ELEMENT forest (time,treelist,fedgelist)>
<!ELEMENT treelist (tree*)>
<!ELEMENT tree (time, assume?, (node|hedge|justification)*)>
<!ELEMENT assume (node|hedge|justification)*)>
<!ELEMENT content (#PCDATA)>
<!ELEMENT node (symid,time,content)>
<!ELEMENT justification (symid,time,content,source,targetlist)>
<!ELEMENT hedge (symid,time,content,source,target)>
<!ELEMENT symid (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT source (symid)>
<!ELEMENT target (symid)>
<!ELEMENT targetlist (symid*)>
<!ELEMENT fedgelist (fedge)>
<!ELEMENT fedge (time, content, source, targetlist)>

<!ATTLIST justification
 selected (0|1) "0" >
```

Source: Autexier

Conclusion

The presented PDSs and Forests support:

- the representation of alternative proof steps for *both*

Conclusion

The presented PDSs and Forests support:

- the representation of alternative proof steps for *both*
  - the reduction of a goal as well as

Conclusion

The presented PDSs and Forests support:

- the representation of alternative proof steps for *both*
  - the reduction of a goal as well as
  - for the expansion of a complex proof step to lower granularity

Conclusion

The presented PDSs and Forests support:

- the representation of alternative proof steps for *both*
  - the reduction of a goal as well as
  - for the expansion of a complex proof step to lower granularity
- the structuring of proof parts (i.e. lemmatization) into separate but connected parts of the data structure

Source: Autexier

Conclusion

The presented PDSs and Forests support:

- the representation of alternative proof steps for *both*
  - the reduction of a goal as well as
  - for the expansion of a complex proof step to lower granularity
- the structuring of proof parts (i.e. lemmatization) into separate but connected parts of the data structure
- the generic representation of proof statements and justifications, biased

Source: Autexier

Conclusion

The presented PDSs and Forests support:

- the representation of alternative proof steps for *both*
  - the reduction of a goal as well as
  - for the expansion of a complex proof step to lower granularity
- the structuring of proof parts (i.e. lemmatization) into separate but connected parts of the data structure
- the generic representation of proof statements and justifications, biased
  - neither to any specific calculus

Source: Autexier

Conclusion

The presented PDSs and Forests support:

- the representation of alternative proof steps for *both*
  - the reduction of a goal as well as
  - for the expansion of a complex proof step to lower granularity
- the structuring of proof parts (i.e. lemmatization) into separate but connected parts of the data structure
- the generic representation of proof statements and justifications, biased
  - neither to any specific calculus
  - nor to any specific formalism for representing abstract proof plans

Source: Autexier

Conclusion

The presented PDSs and Forests support:

- the representation of alternative proof steps for *both*
  - the reduction of a goal as well as
  - for the expansion of a complex proof step to lower granularity
- the structuring of proof parts (i.e. lemmatization) into separate but connected parts of the data structure
- the generic representation of proof statements and justifications, biased
  - neither to any specific calculus
  - nor to any specific formalism for representing abstract proof plans
- Any further *semantics* must be provided by the using system

  (e.g. scope of variables, resolution of cycles introduced by forest links, . . .)

This allows...

- Represent alternative proof ideas (Horizontal alternatives)
- Represent the same proof idea in different underlying calculi.

  Organize proof with subproofs in different calculi

  (Generic, Hierarchies, Alternative Expansions)
- Sharing of common initial proof parts for expansions

  (Hierarchies, alternative expansions)
- Represent the search space explored by automated proof techniques

  can serve for debugging of automated proof techniques
- XML for storing proofs, alternative proofs, proofs under construction

  Discussion: suitable extension for OMDOC to represent proofs of a same theorem with different formalisms and/or different proof ideas

Source: Autexier