Towards Merging *PlatΩ* and PGIP

David Aspinall[1] Serge Autexier[2]
Christoph Lüth[2] Marc Wagner[2,3]

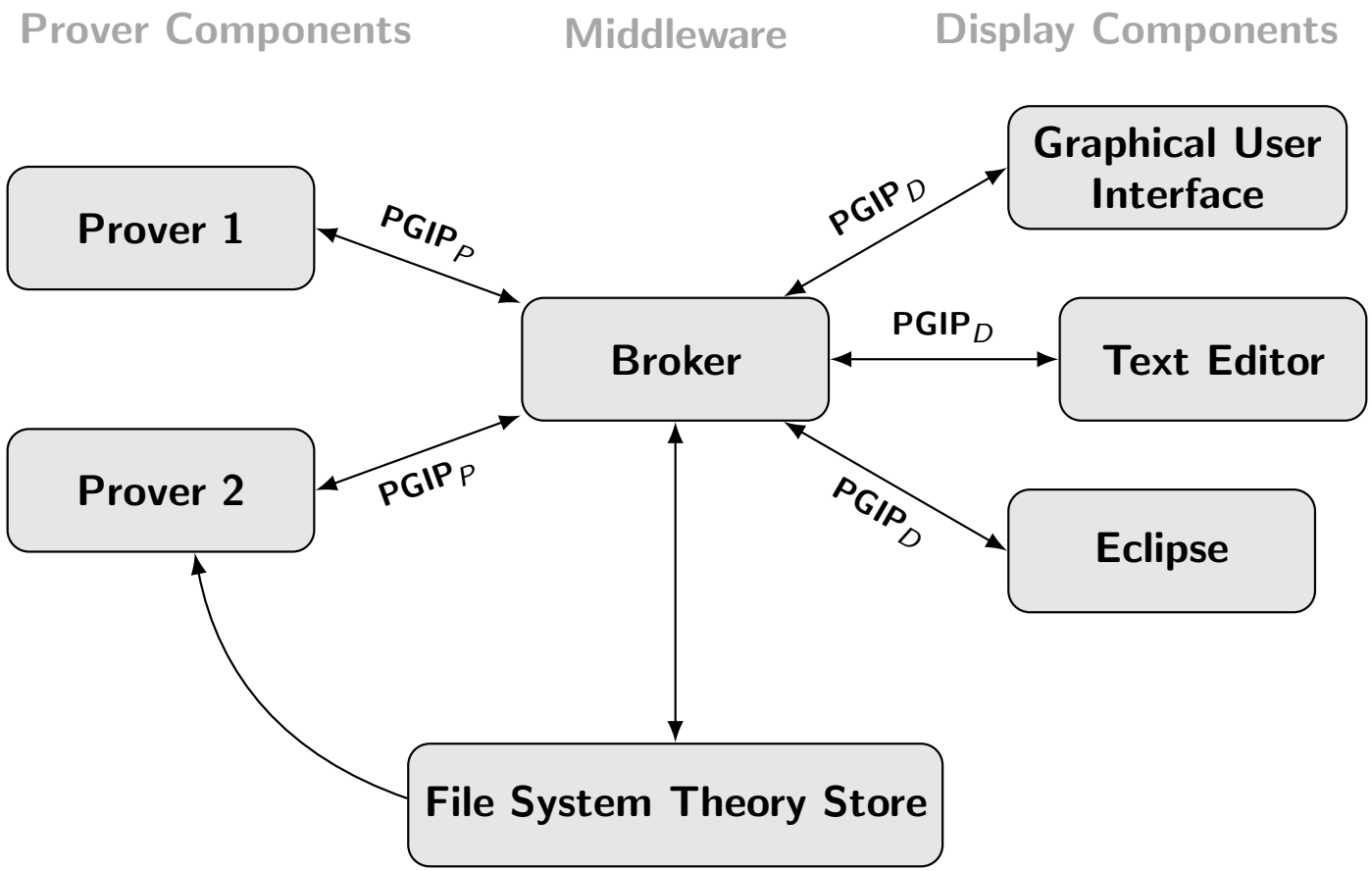
[1]University of Edinburgh, Scotland

[2]DFKI Bremen, Germany

[3]Saarland University, Saarbrücken, Germany

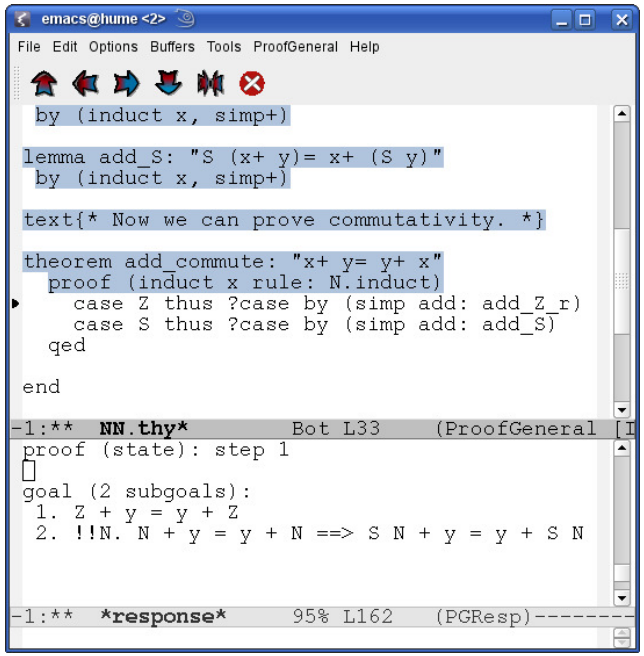
8th International Workshop
User Interfaces for Theorem Provers
Montréal, Canada
22. August 2008

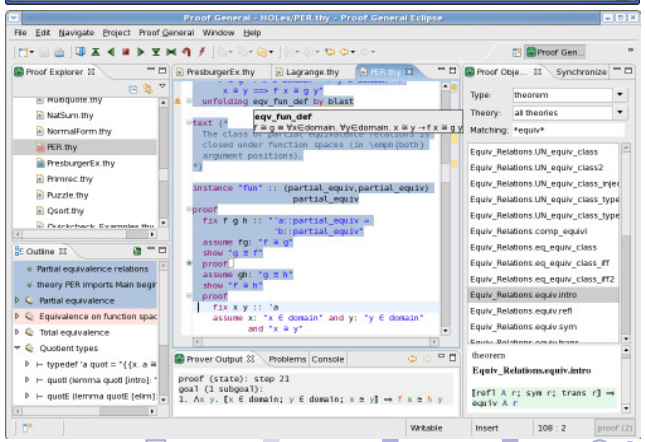
PGKIT Architecture



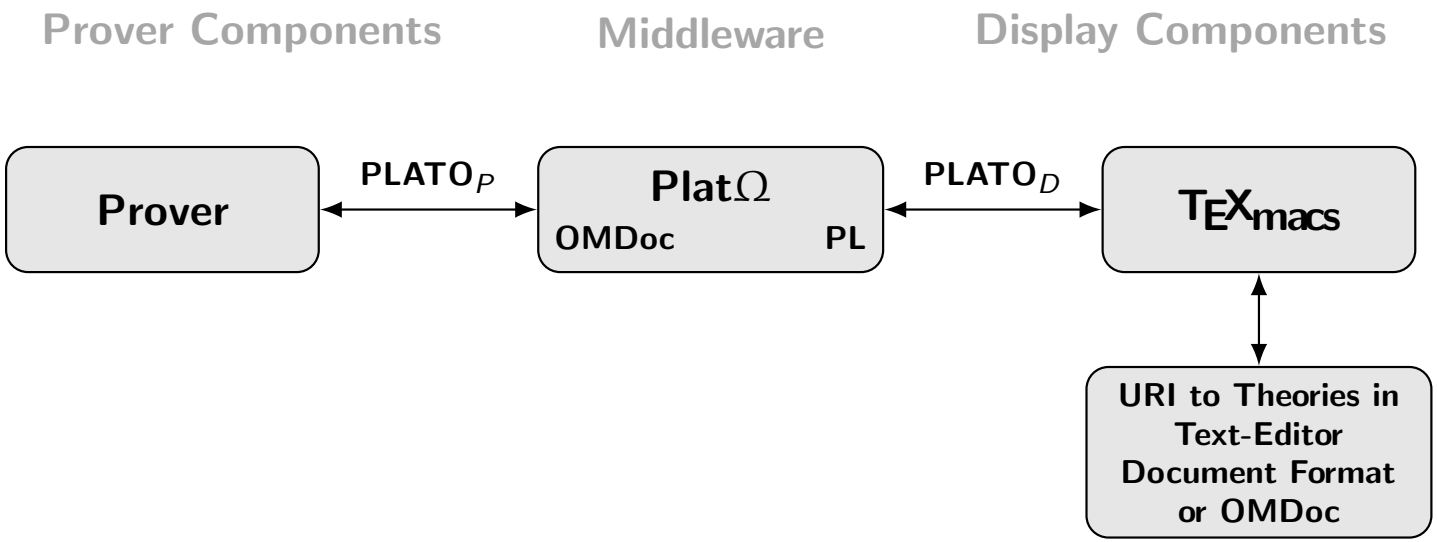
PGKIT Architecture

- ACL2, Isabelle, HOL, Lambda-Clam, Lego, Plastic, Phox, ...
- ASCII Files (Input-only)
- Specific parsers
- Command-Line oriented
- Single focus
- Display of proof state



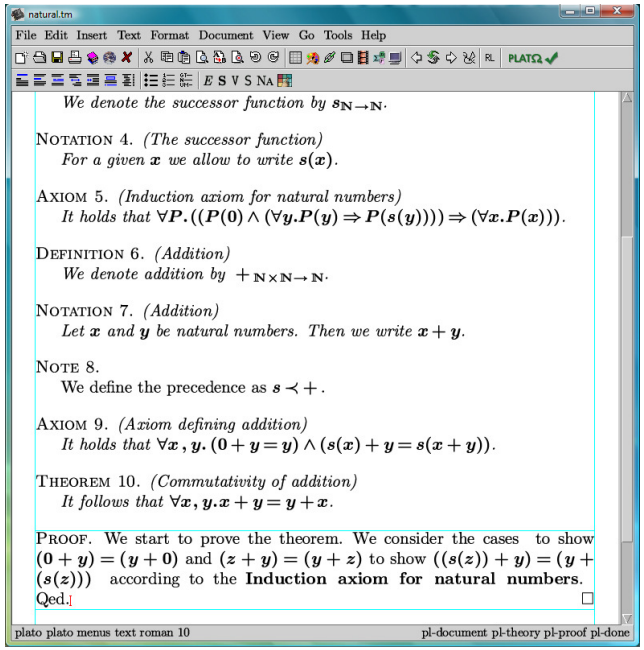


PlatΩ Architecture



PlatΩ Architecture

- Writing TEX$_{\text{MACS}}$ documents assisted by ΩMEGA
- ΩMEGA: OMDOC Input/Output
- TEX$_{\text{MACS}}$: Specific, flexible proof document style (PL)
- Multiple foci
- Bidirectional: proof steps done by the prover are propagated into the TEX$_{\text{MACS}}$ document
- XUPDATE/XMLDIFF based
- On the fly conversions between OMDOC and PL
- Contect-sensitive menus for assistance



Comparision

	PlatΩ Display	PG Kit Display	*PGIP 2* Display
Document format	XML	Plain text	XML
Document syntax	$\TeX_{\text{MACS}}$	ASCII	Generic
Change protocol	XUpdate	PGIP_{D}	XUpdate
Operations	Context-dependent menus	Global menus, typed operations	Context-dependent menus, typed operations

Comparision

	PlatΩ Prover	PG Kit Prover	*PGIP 2* Prover
Document format	XML	Plain text	XML
Document syntax	OMDoc	Native prover syntax	Generic
Change protocol	XUpdate	$\mathrm{PGIP_P}$	XUpdate
Prover support	ΩMEGA	Generic (Coq, Isabelle, etc)	Generic (Coq, Isabelle, ΩMEGA, etc)
Operations	Context-dependent menus	Global menus, typed operations	Context-dependent menus, typed operations

Overview of Extensions for PGIP 2

1. Support for Semi Structured Documents
   - Multiple Foci
   - XUpdate/Semantic XMLDiff
   - Protocols
2. Context-Sensitive Service Menus
3. Multiple Editing Displays in Parallel
4. Mutiple Document Formats

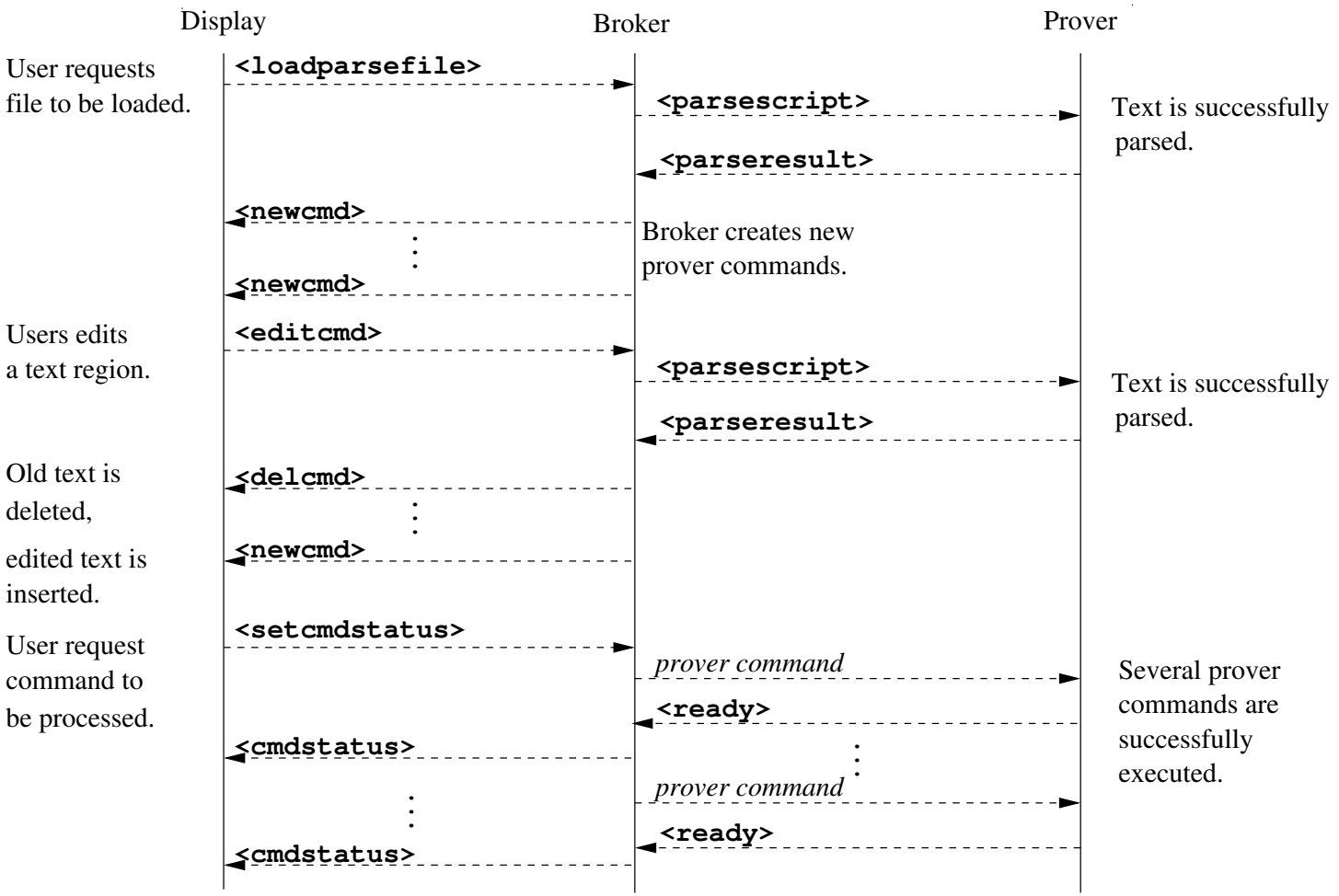
Working hypothesis: Do not require any change for current displays/provers when using the new PGIP 2

1. Support for Semi Structured Documents
   - Multiple Foci
   - XUpdate/Semantic XMLDiff
   - Protocols
2. Context-Sensitive Service Menus
3. Multiple Editing Displays in Parallel
4. Mutiple Document Formats

Unstructured Documents

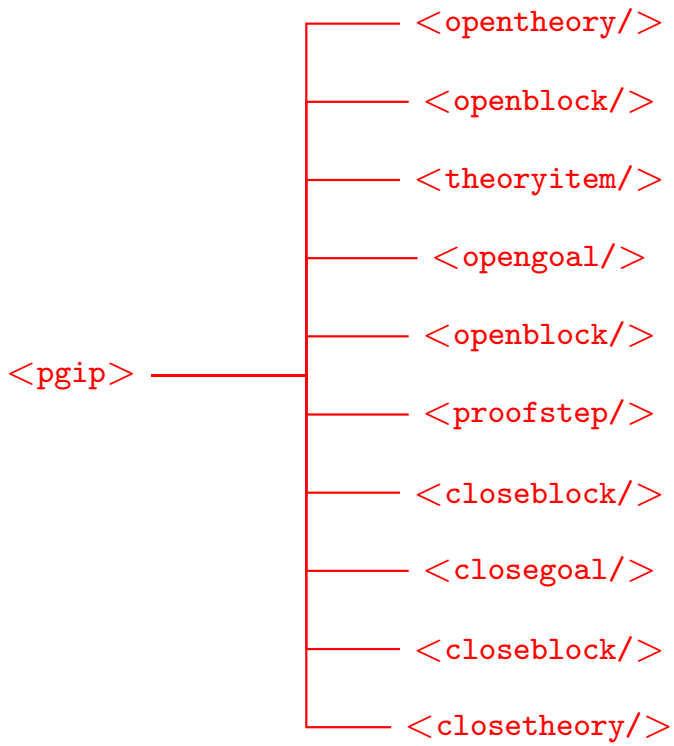

```
theory NN
imports Main
begin

text {* The datatype of natural numbers\footnote{Not
        the real ones!}. *}
datatype N = Z | S N

text{* Induction is a derived concept: @{thm N.induct}
        *}

text{* We define addition via recursive
 equations, and declare its usual syntax. *}

fun "add" :: "N \<Rightarrow> N \<Rightarrow> N"
where
   "add   Z x = x"
 | "add (S x) y= S (add x y)"

notation
  add (infixr "+" 65)

text{* We first need two lemmas. *}

lemma add_Z_r: "x + Z= x"
 by (induct x, simp+)

lemma add_S: "S (x+ y)= x+ (S y)"
 by (induct x, simp+)

text{* Now we can prove commutativity. *}

theorem add_commute: "x+ y= y+ (x::N)"
  proof (induct x rule: N.induct)
    case Z thus ?case by (simp add: add_Z_r)
    case S thus ?case by (simp add: add_S)
  qed

end
```

Unstructured Documents



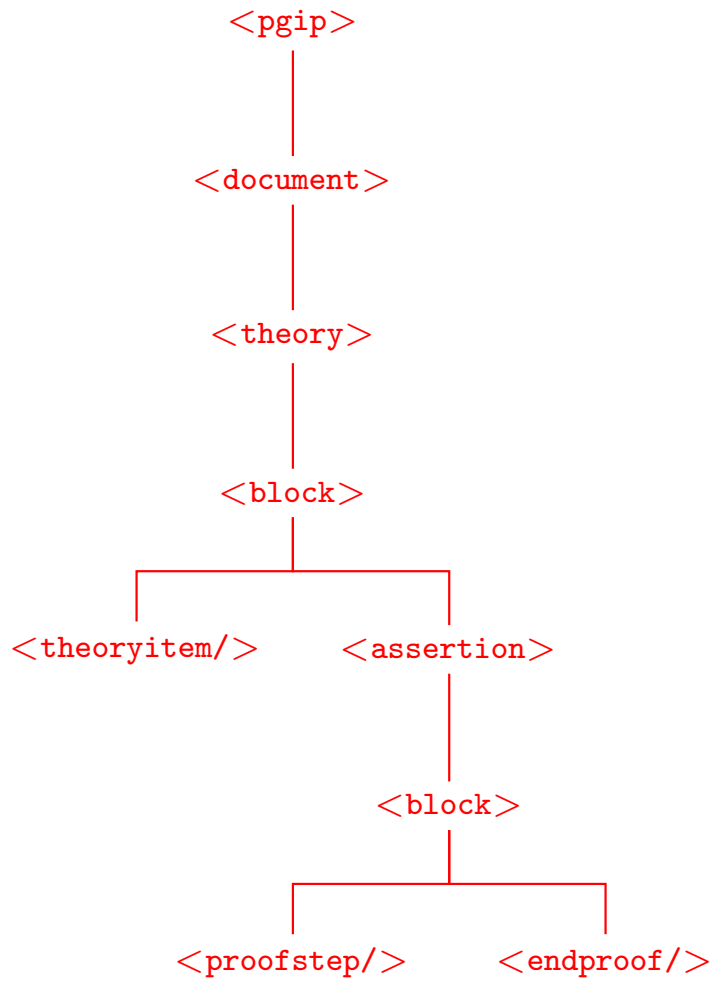
Unstructured Documents


```
<pgip tag = "Isabelle/Isar" id = "hume/cxl/27572/1208530140.785"
      destid =
  "broker:hume/2658/27571/2008318-144859-Z" class = "pg" refid =
  "broker:hume/2658/27571/2008318-144859-Z" refseq = "18" seq =
  "24"><parseresult>

<opentheory thyname = "NN" parentnames = "Main">theory NN
imports Main
begin</opentheory><openblock objtype = "theory body"/>

<theoryitem objtype = "theory declaration">text {* The datatype of
      natural numbers\footnote{Not the real ones!}. *}</theoryitem
      >
<theoryitem objtype = "theory declaration">datatype N = Z | S N</
      theoryitem>

<theoryitem objtype = "theory declaration">text{* Induction is a
      derived concept: @{thm N.induct} *}</theoryitem>

<theoryitem objtype = "theory declaration">text{* We define addition
      via recursive equations,
 and declare its syntax as the usual infix operator. *}</theoryitem>

<theoryitem objtype = "theory declaration">fun &quot;add&quot; :: &
      quot;N \&lt;Rightarrow&gt; N \&lt;Rightarrow&gt; N&quot;
where
   &quot;add  Z x = x&quot;
 | &quot;add (S x) y= S (add x y)&quot;</theoryitem>
```



Semi Structured Documents

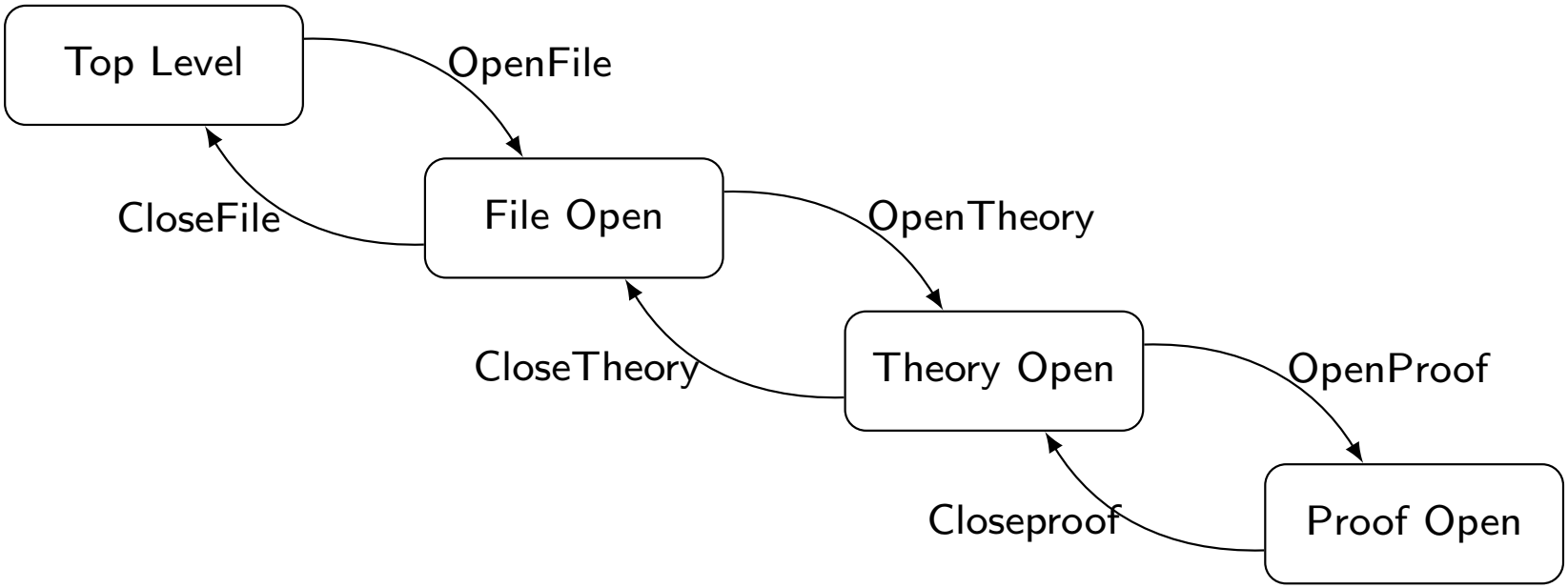

```
<pgip tag = "Isabelle/Isar" id = "hume/cxl/27572/1208530140.785"
      destid =
  "broker:hume/2658/27571/2008318-144859-Z" class = "pg" refid =
  "broker:hume/2658/27571/2008318-144859-Z" refseq = "18" seq =
  "24">
<parseresult>
<document>
<theory thyname="NN" parentnames="Main">theory NN
imports Main
begin
<block objtype="theory body">

<theoryitem>text {* The datatype of natural numbers\footnote{Not
      the real ones!}. *}</theoryitem>
<theoryitem type="adt">datatype N = Z | S N</theoryitem>

<theoryitem>text{* Induction is a derived concept: @{thm N.induct}
      *}</theoryitem>

<theoryitem>text{* We define addition via recursive equations,
 and declare its syntax as the usual infix operator. *}</theoryitem>

<theoryitem objtype="fundef">fun &quot;add&quot; :: &quot;N \&lt;
      Rightarrow&gt; N \&lt;Rightarrow&gt; N&quot;
where
   &quot;add  Z x = x&quot;
 | &quot;add (S x) y= S (add x y)&quot;</theoryitem>

<theoryitem objtype="notation">notation
```



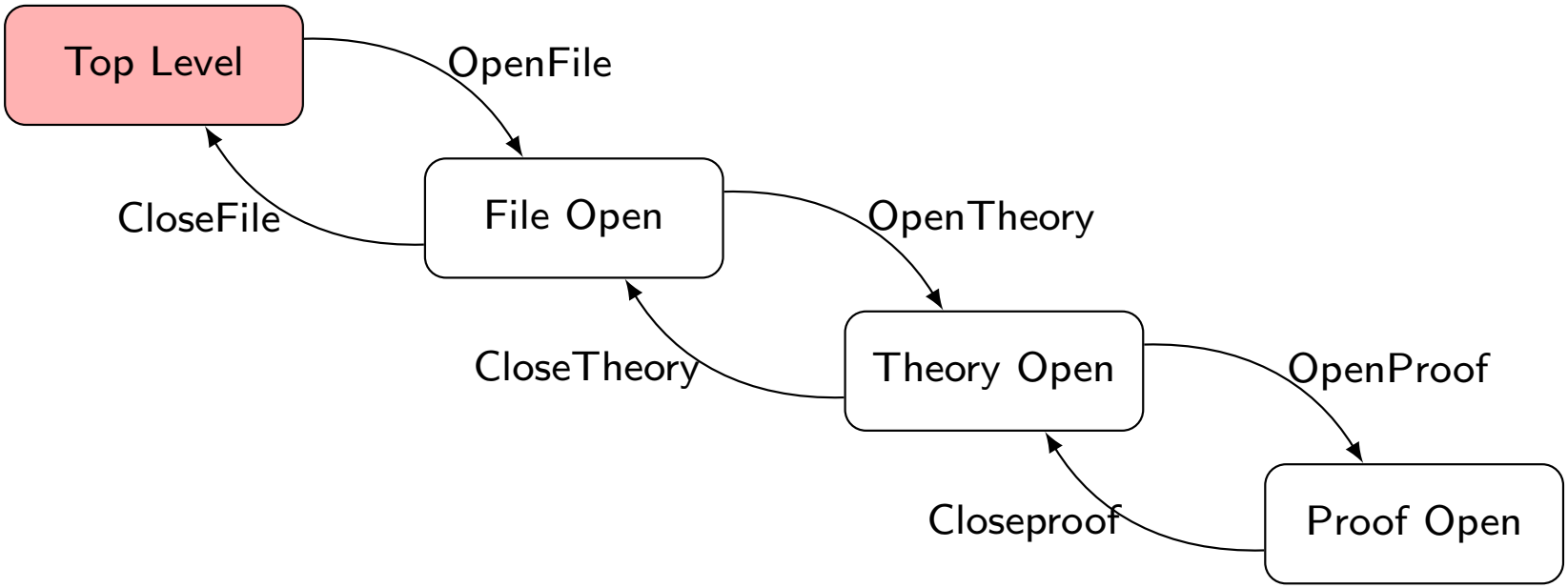
Supporting Semi Structured Documents

- New internal PGIP markup (PGML), tree compatible
  - Use `xml:id` to flexibilise document structure
    Example: theorems and proofs need not occur together physically
  - Rigid structure computed by the broker for the prover
- Provide on-the-fly converters between old and new markup
  $\Rightarrow$ provers need not be adapted
- Configuration

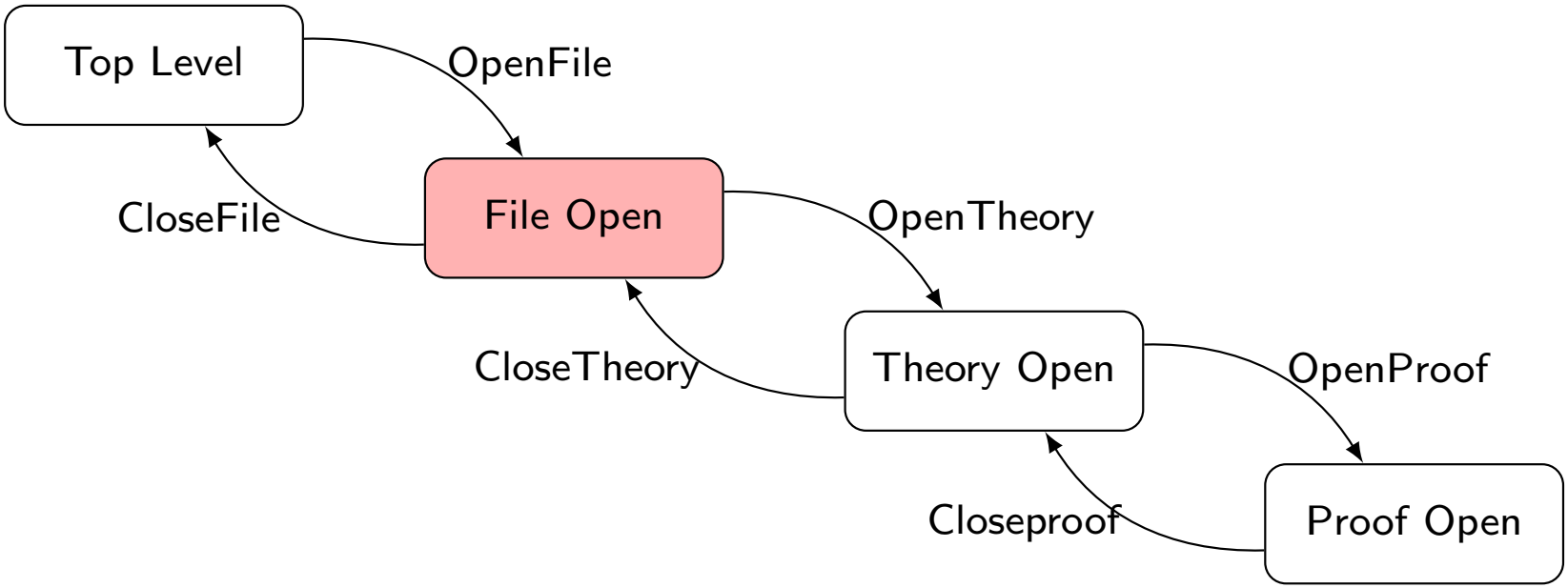
Single Focus



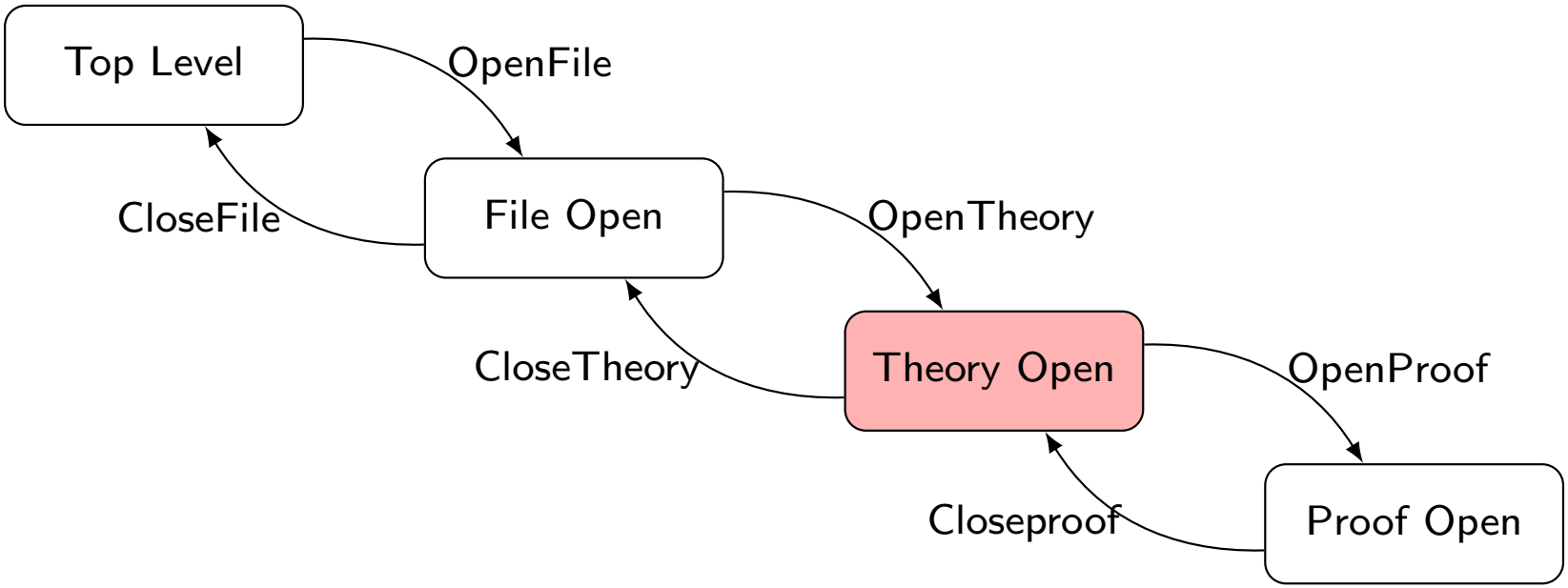
Single Focus



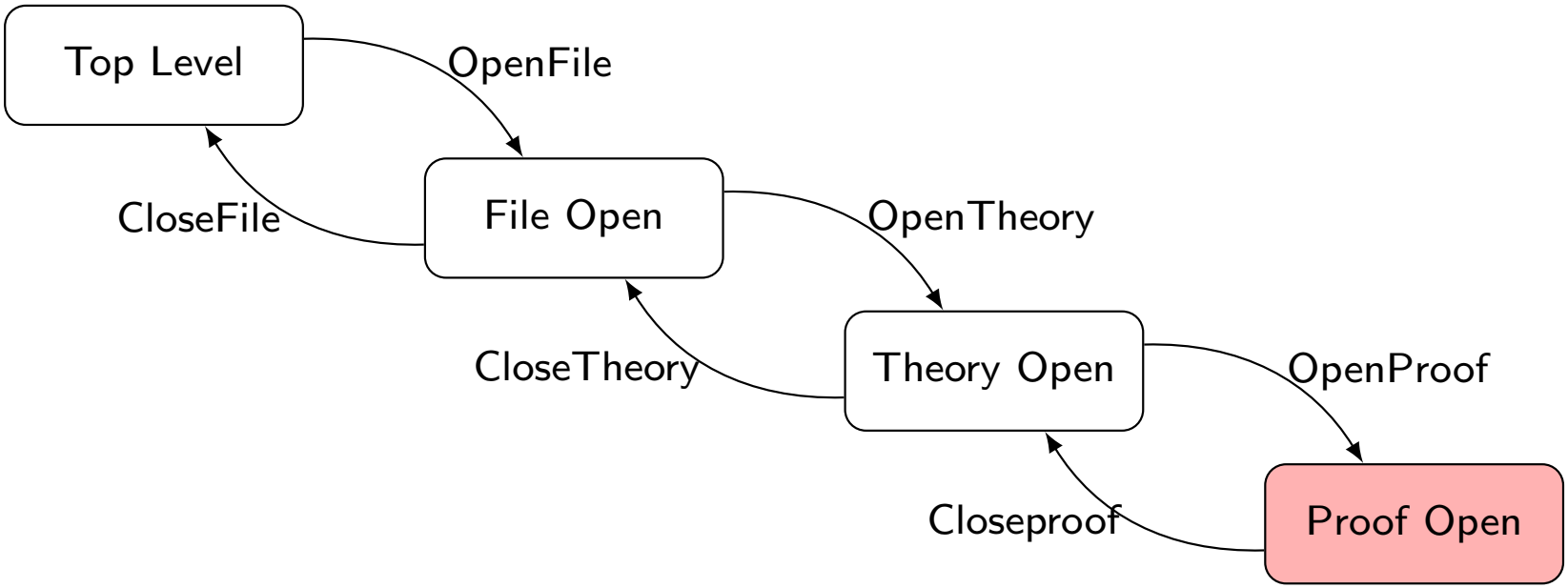
Single Focus



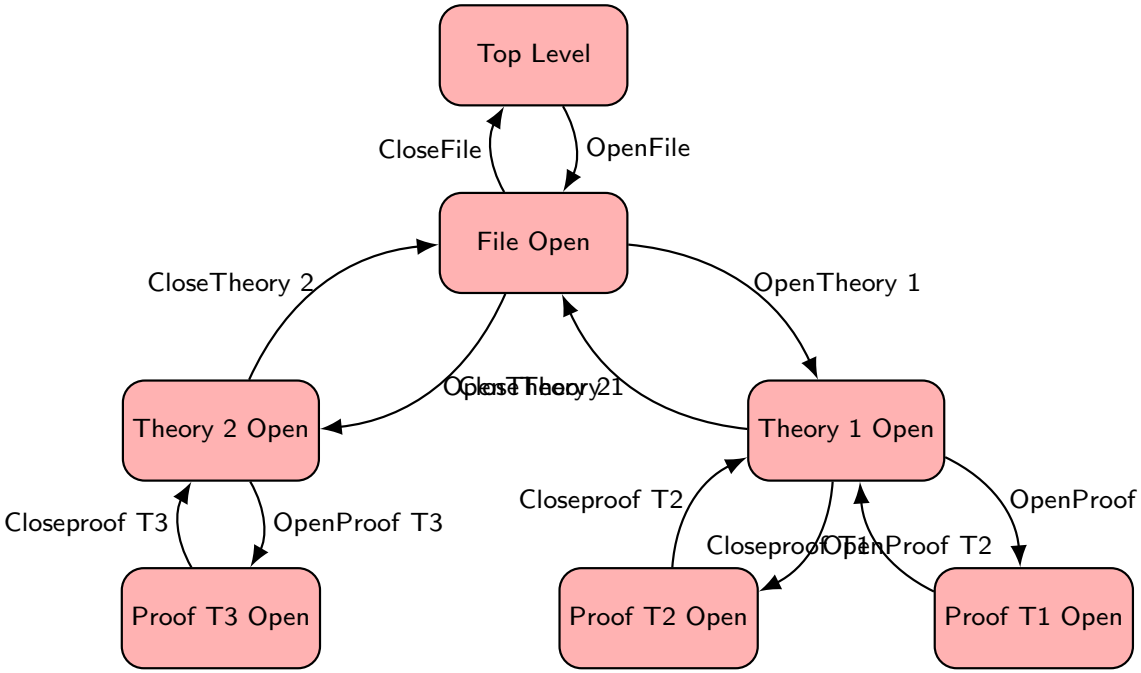
Single Focus

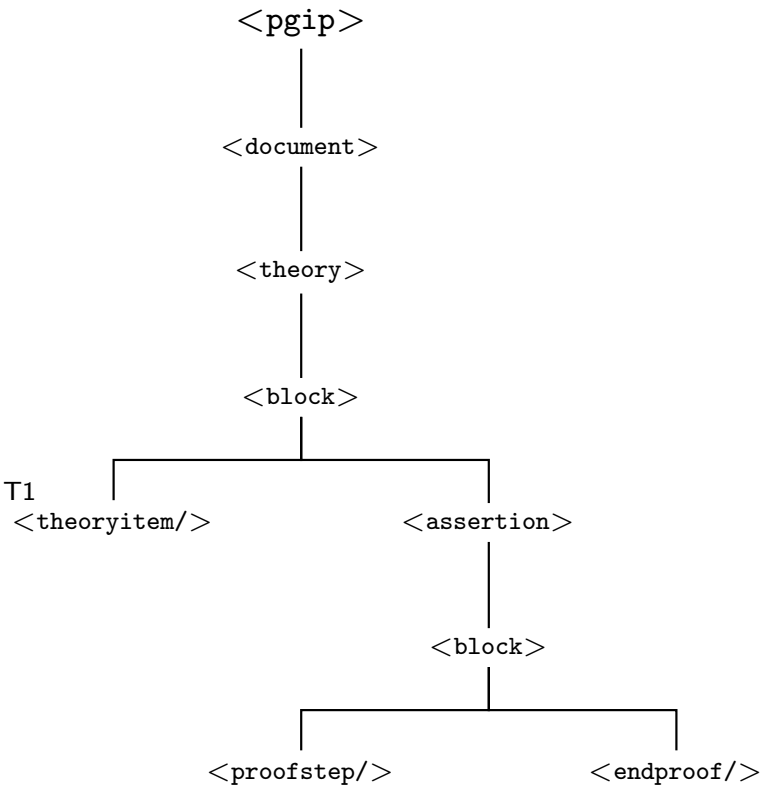


Single Focus



Multiple Foci





Supporting Semi Structured Documents

- New internal PGIP markup, tree compatible
- Provide on-the-fly converters between old and new markup
  $\Rightarrow$ provers need not be adapted
- Configuration

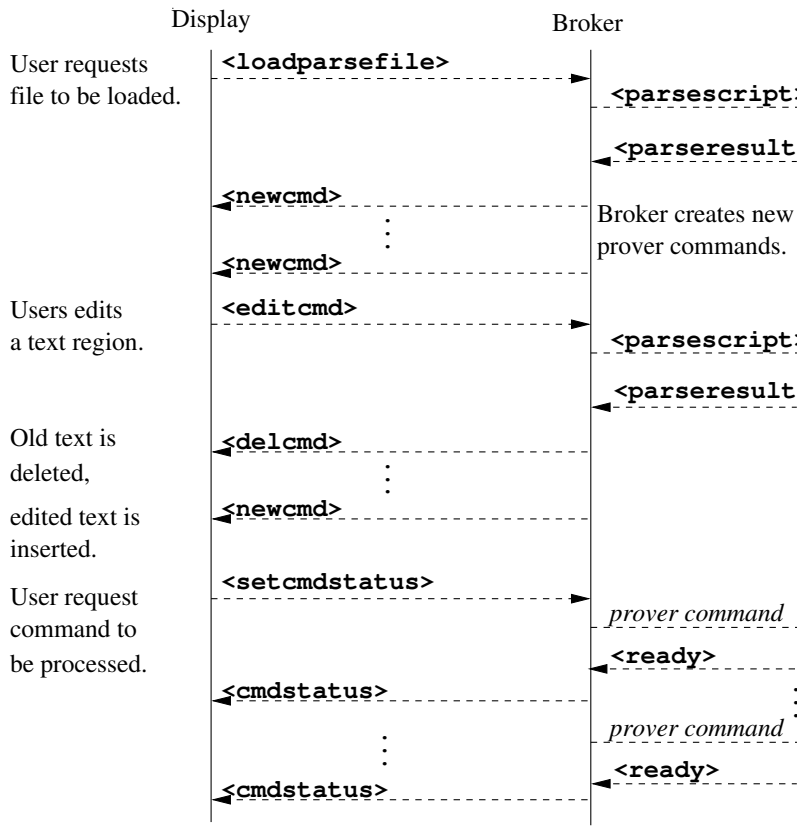
XUpdates

Old PGIP$_D$ protocol (asymmetric):

- Display → Broker: `<createcmd>`, `<editcmd>`
- Broker → Display: `<newcmd>`, `<delcmd>`, `<replacecmd>`

New PGIP$_D$ protocol:

- Rephrase in terms of Xupdate-modifications `<insert-after>`, `<insert-before>`, `<remove>`



Compatibility

- XUPDATEs cannot be generated by all displays or provers
- Allow components to send only new document version and compute differences inside the broker
- Use semantic XMLDIFF tool inside the broker
  - Normal tree diff
  - Configurable to take some “semantics” into account
  - Configurable level of granularity for modifications
- Configuration if component can handle XUPDATES
  Broker generates appropriate representation for each component

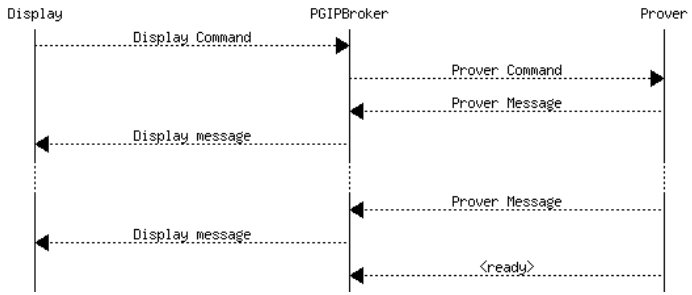
Protocol between Prover and Broker

- <parsescript> and <parseresult>
- Prover commands in native prover command syntax
- prover answers <normalresponse>, <errorresponse>, <ready>

Unchanged, but contextualized (multiple foci)

Extension:

- <normalresponse> can contain XUPDATES for document which are relayed to the display



Protocols between Display with Broker

In addition to PGIP_D provide

- XML-RPC interface and
- HTTP in plain REST architecture

HTTP Request	**HTTP answer**
`/initsession`	path to new session (`/session1`)
`/session1/initdocument?CONTENT`	HTML rendered document after parsing
`/session1/document/1`	version 1 of the document
`/session1/document/HEAD`	the current version of the document

1. Support for Semi Structured Documents
   - Multiple Foci
   - XUpdate/Semantic XMLDiff
   - Protocols
2. Context-Sensitive Service Menus
3. Multiple Editing Displays in Parallel
4. Mutiple Document Formats

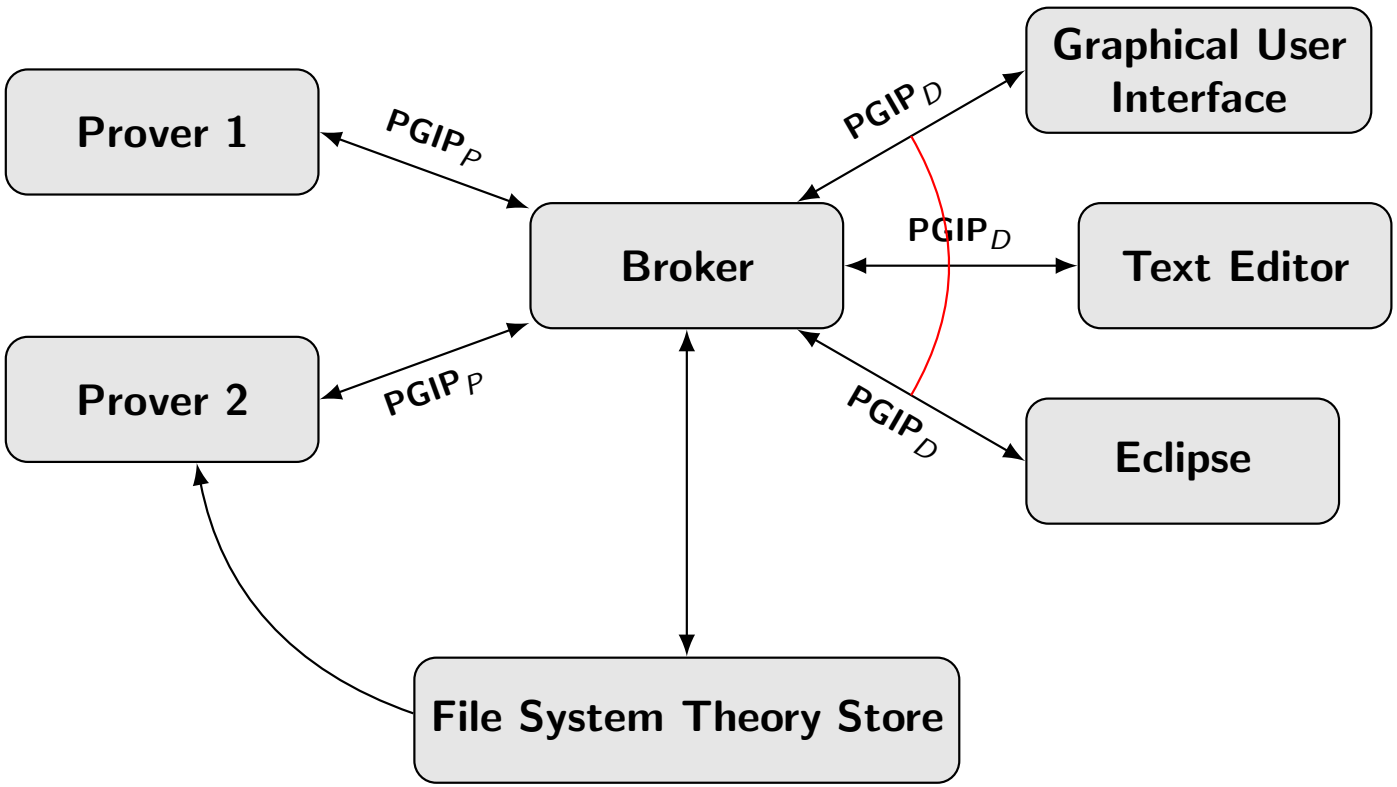
Service Menus

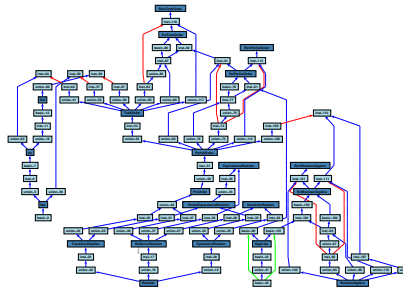
- Allow requests of service menus for arbitrary elements in the display
- Requests relayed by the broker to the prover
- Prover returns menu description (PGML menu format)
  - Menus contain descriptions and actions
  - Actions can be
    - tactic calls (traditional)
    - requests for axiom/lemma applications (*Plat*Ω/TEX$_{\text{MACS}}$)
- Computing the complete menus is expensive, most of the menu is discarded
- Support lazy menu computation
  - Menu entries trigger computations which results are modifications of the menu

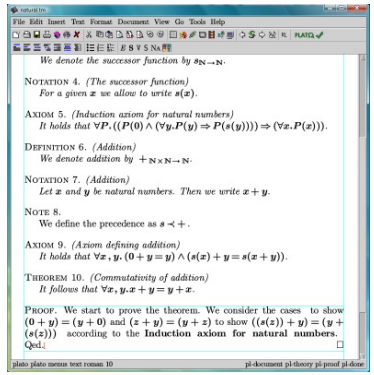
1. Support for Semi Structured Documents
   - Multiple Foci
   - XUpdate/Semantic XMLDiff
   - Protocols
2. Context-Sensitive Service Menus
3. Multiple Editing Displays in Parallel
4. Mutiple Document Formats

Multiple Editing Displays

Prover Components | Middleware | Display Components

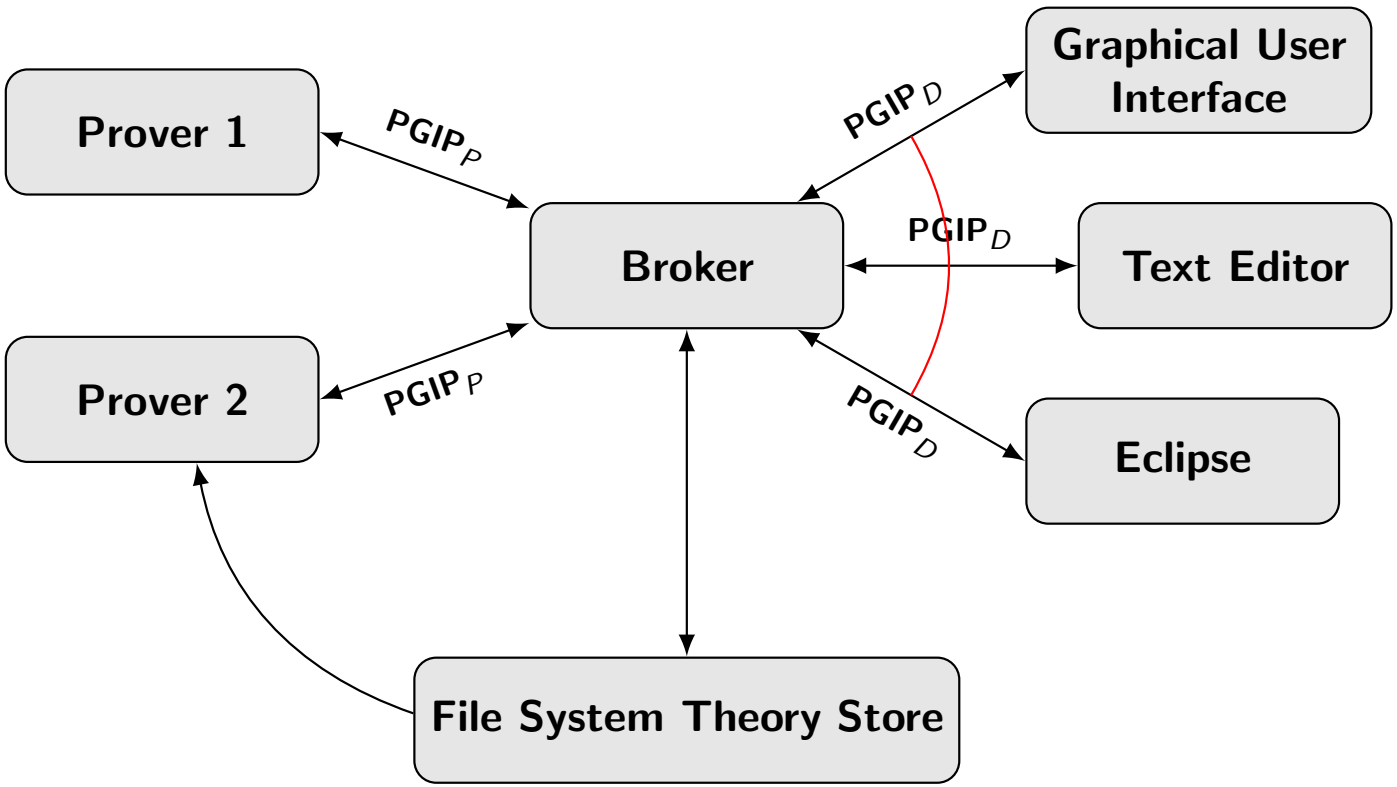


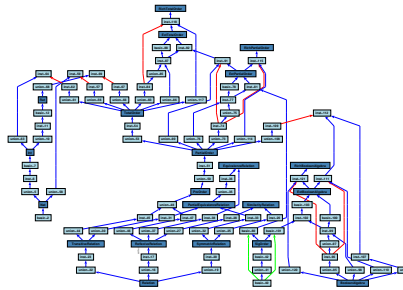


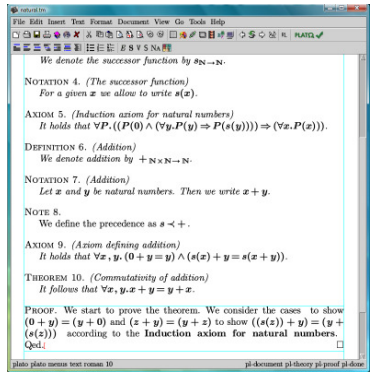


Multiple Editing Displays

Prover Components | Middleware | Display Components



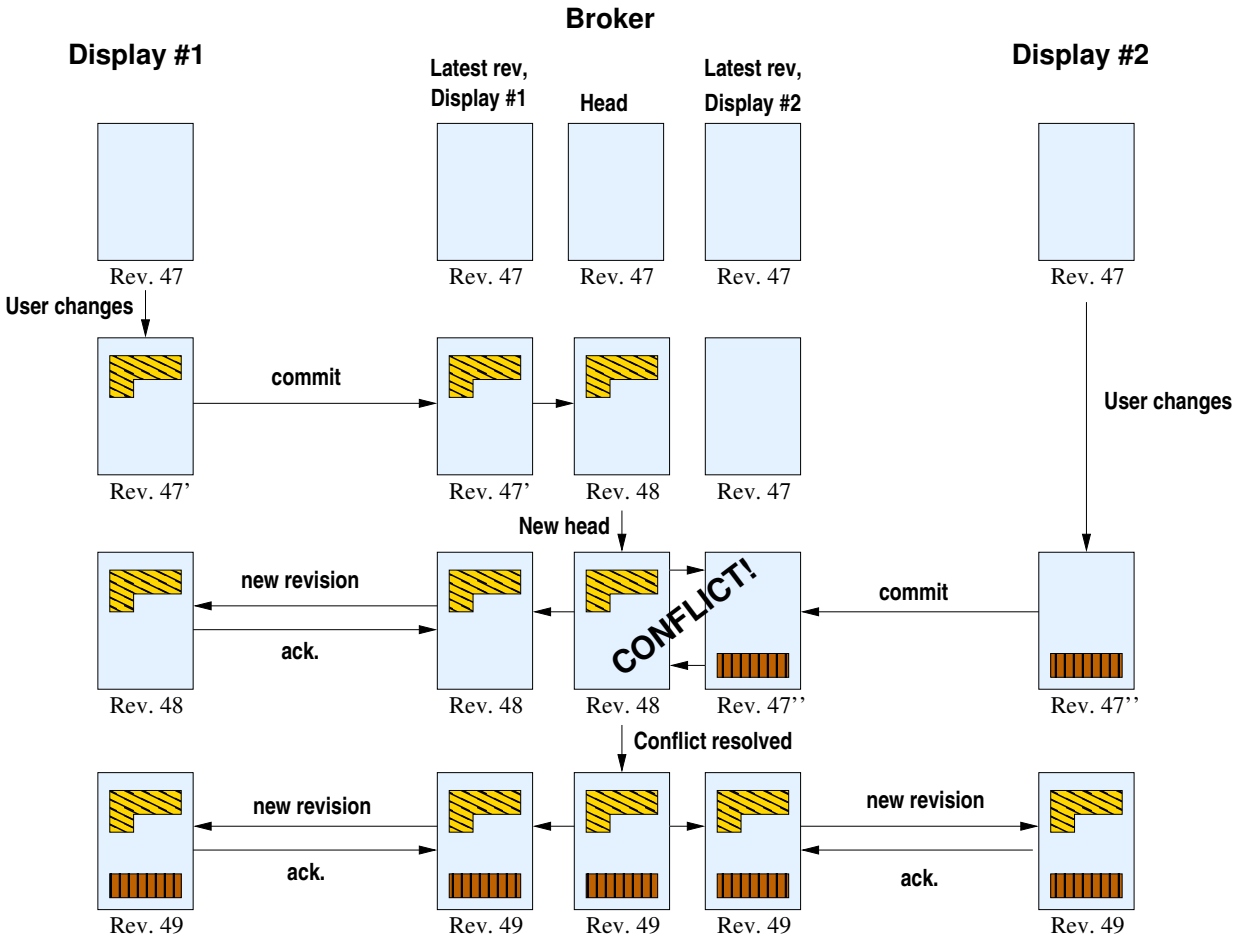




Requires Synchronisation by the Broker

Multiple Displays

- Simple kind of version control inside the broker
- Broker stores for each display the latest version communicated
- Uses semantic XMLMerge (diff3)
- Conflicts are marked in special markup extending XUpdate markup.

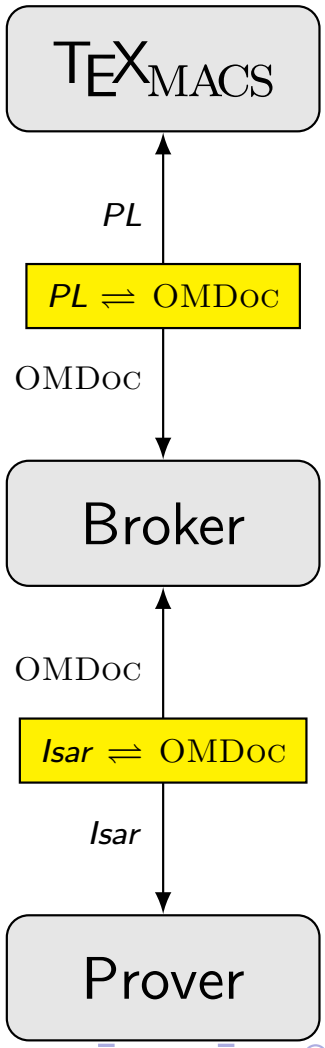


Multiple Document Formats

- Have: Display D communicates in format *A*
- Have: Prover P communicates in format *B*
- Wanted: Use display D with prover P

For instance:

- *PlatΩ* connects TEXMACS with format PL to ΩMEGA with format OMDoc
- Now want to use TEXMACS with Isabelle (Isar)



Multiple Document Formats

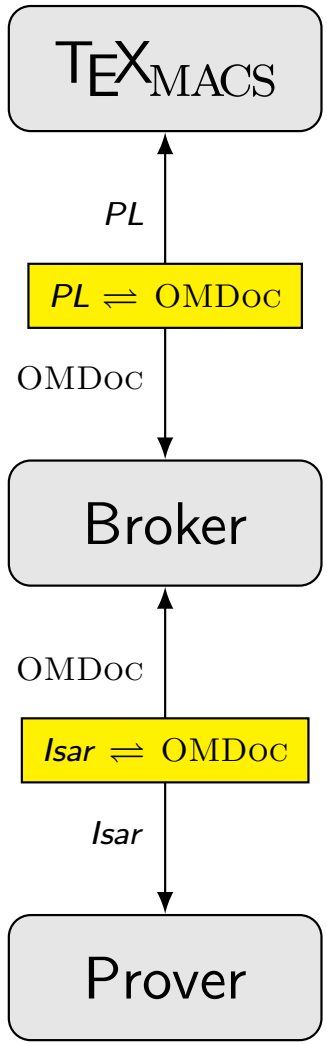
- Have: Display D communicates in format *A*
- Have: Prover P communicates in format *B*
- Wanted: Use display D with prover P

For instance:

- *PlatΩ* connects TEXMACS with format PL to ΩMEGA with format OMDOC
- Now want to use TEXMACS with Isabelle (Isar)

Provide autonomous converters from *PL* to OMDOC and OMDOC and Isar

- Bidirectional
- Conversion of Xupdates
- Maintain mappings to relay service requests



Related Work

- Semi-structured input documents
  MathsTiles (structured input to multiple domain reasoners, unidirectional)
- Multiple foci
  ΩMEGA (status management added), Isabelle (multi-foci under the way)
- Multiple displays
  MATITA (proof script,goal display), LΩUI, GEOPROOF/ COQ
- Support modifications to the document from the prover:
  MATITA (tinycals)
  Could be realised by the generic functionality proposed here?

Next Steps

- Implementation (any feedback/remarks welcome before we start. . .)
- Implement *Plat*Ω functionality on PGIP 2 basis
- Connect other editing displays (Word 2007, OpenOffice, Netbeans)
- Multiple foci, changes from prover could serve interaction styles in other provers (Agda, Mizar, PVS)