

Plenary Demo: Isabelle/jEdit — a Generic Frontend for Formal Mathematics

Walther Neuper

IIS Institute Integriert Studieren
Johannes Kepler University Linz
Austria

ThEdu'26 at FLoC
19. July 2026

Outline¹

- 1 Demo: Generic Tools of Isabelle/Isar for ...
Isabelle/MAWEN with ML
Isabelle/jEdit for Haskell
- 2 Demo: Some features of the generic tools:
Elegance and simplicity
A glimpse at Isabelle's architecture
- 3 Summary and Conclusion

¹For PDF click on  at <https://isac.miraheze.org/m/C3B>

Outline

- 1 Demo: Generic Tools of Isabelle/Isar for ...
Isabelle/MAWEN with ML
Isabelle/jEdit for Haskell
- 2 Demo: Some features of the generic tools:
Elegance and simplicity
A glimpse at Isabelle's architecture
- 3 Summary and Conclusion

Isabelle/MAWEN with ML mathematics engine

Project currently under development, file

`test/main/BridgePIDE/Test_VSCode_Example.thy`

- MAWEN inherits representation of formulas with
 - sub-term structure by mouse hovering
 - syntax errors indicated at correct place
 - linking elements with underlying knowledge
 - 'accessible' by blind people.
- To be shown: integration with suprising little effort

Isabelle/MAWEN with ML mathematics engine

Project currently under development, file
`test/main/BridgePIDE/Test_VSCode_Example.thy`

- MAWEN inherits representation of formulas with
 - sub-term structure by mouse hovering
 - syntax errors indicated at correct place
 - linking elements with underlying knowledge
 - 'accessible' by blind people.
- To be shown: integration with suprising little effort

Outline

- 1 Demo: Generic Tools of Isabelle/Isar for ...
Isabelle/MAWEN with ML
Isabelle/jEdit for Haskell
- 2 Demo: Some features of the generic tools:
Elegance and simplicity
A glimpse at Isabelle's architecture
- 3 Summary and Conclusion

Isabelle/jEdit for Haskell

<https://github.com/naproche/naproche>

- Proof Checking of Natural Mathematical Documents, with optional support for Isabelle Prover IDE (Isabelle/PIDE – Isabelle/jEdit).
- For End-Users:
You can get a fully integrated version of Naproche for Linux, macOS and Windows ...
- For Developers:
Isabelle/Naproche Prover IDE: Prerequisites, Repository Management, etc

... and lots of publications in mathematics and computer science,
e.g. <https://www.math.uni-bonn.de/ag/logik/teaching/2024SS/praktikum.shtml>

A glimpse at Naproche with Isabelle/jEdit

Generic Tools

ML
Haskell

features

simplicity
architecture

Summary and Conclusion

The screenshot shows the Isabelle/jEdit interface with the file `Isabelle2021/HOL - TUTORIAL.ftl.tex` open. The main editor displays the following LaTeX code:

```

988 \begin{definition} $$$ is infinite iff $$$ is not finite.
989 \end{definition}
990
991 \end{forthel}
992
993 \section{Euclid's Theorem}
994
995 Now everything is in place for the proof that there
996 are infinitely many prime numbers.
997 \begin{forthel}
998
999 \begin{signature} $\Primes$ is the class of prime natural numbers.
1000 \end{signature}
1001
1002 % \begin{lemma} Contradiction. \end{lemma}
1003
1004
1005 \begin{theorem}[Euclid]
1006 $\Primes$ is infinite.
1007 \end{theorem}
1008 \begin{proof}
1009 Assume that $r$ is a natural number and
1010 $p$ is a sequence of length $r$ and
1011 $\Set{p\{1\}\{r\}}$ is a subclass of $\Primes$

```

The right sidebar shows a document tree with the following structure:

- TUTORIAL.ftl.tex
 - About This Document
 - Notation
 - Notation
 - Notation
 - General Remarks
 - Natural Numbers - Introducing t
 - Natural Numbers - Postulating A
 - The Natural Order - Defining \R
 - Lemmas and Theorems
 - Linear and Discrete Orders
 - Induction
 - Division
 - An Interactive Proof
 - Primes
 - Classes
 - Finite Sequences and Products,
 - Finite and Infinite Sets
 - Euclid's Theorem
 - Thm: [Euclid]
 - Proof:

The bottom status bar shows the current line (1009,41) and various system metrics:

1009,41 (35031/35761) (latex_none,UTF-8-Isabelle) in mroUG JVM: 288/512MB ML: 169/169MB 3:55 PM

Outline

- 1 Demo: Generic Tools of Isabelle/Isar for ...
Isabelle/MAWEN with ML
Isabelle/jEdit for Haskell
- 2 Demo: Some features of the generic tools:
Elegance and simplicity
A glimpse at Isabelle's architecture
- 3 Summary and Conclusion

Elegance and simplicity

- Get all features of **formula-representation**:

Into file

test/main/BridgePIDE/Demo_FLoC_26.thy
we type ...

- ...two lines into Isabelle/Isar
- ...one line into Isabelle/ML
- ...and have all the achievements of Isabelle/jEdit
auto-magically.

- **System handling** for specific software:

- e.g. for Isabelle/MAWEN
menu 'MAWEN' > 'MAWEN Panel' > 'Browse
Knowledge' > 'Examples'
 - click on an item: student's view (rudimentary)
 - double click: author's view on the definition
- we have a look at the respective code.
Files src/scala/★

Elegance and simplicity

- Get all features of **formula-representation**:

Into file

`test/main/BridgePIDE/Demo_FLoC_26.thy`
we type ...

- ...two lines into Isabelle/Isar
- ...one line into Isabelle/ML
- ...and have all the achievements of Isabelle/jEdit
auto-magically.

- **System handling** for specific software:

- e.g. for Isabelle/MAWEN

`menu 'MAWEN' > 'MAWEN Panel' > 'Browse
Knowledge' > 'Examples'`

- click on an item: student's view (rudimentary)
 - double click: author's view on the definition
- we have a look at the respective code.
`Files src/scala/*`

Elegance and simplicity

- Get all features of **formula-representation**:

Into file

```
test/main/BridgePIDE/Demo_FLoC_26.thy  
we type ...
```

- ...two lines into Isabelle/Isar
- ...one line into Isabelle/ML
- ...and have all the achievements of Isabelle/jEdit
auto-magically.

- **System handling** for specific software:

- e.g. for Isabelle/MAWEN

```
menu 'MAWEN' > 'MAWEN Panel' > 'Browse  
Knowledge' > 'Examples'
```

- click on an item: student's view (rudimentary)
 - double click: author's view on the definition
- we have a look at the respective code.
Files `src/scala/`*

Outline

- 1 Demo: Generic Tools of Isabelle/Isar for ...
Isabelle/MAWEN with ML
Isabelle/jEdit for Haskell
- 2 Demo: Some features of the generic tools:
Elegance and simplicity
A glimpse at Isabelle's architecture
- 3 Summary and Conclusion

Isabelle's architecture

What is Isabelle ² ?

- Proof:
 - Isabelle/Isar: Intelligible semi-automated reasoning + **generative tools**
 - Document language: $\text{\LaTeX}$ type-setting of proof text
 - Programming:
 - Isabelle/Scala: System integration (JVM)
- | |
|--|
| Isabelle/jEdit: frontend (JVM) |
| Isabelle/Scala: (JVM) |
| Isabelle/ML --- interchangeable |
- Isabelle/ML: Tool implementation (Poly/ML)
 - Logic:
 - Isabelle/Pure: Logical framework and bootstrap
 - Isabelle/HOL: Theories and tools for applications

²<https://sketis.net/2025/>

Summary

- We have shown
 - generic tools of Isabelle/Isar
 - to connect various math SW with Isabelle/jEdit
 - examples for such SW in ML and Haskell
 - the connection **auto-magically** provides formulas in Isabelle quality:
 - sub-term structure by mouse hovering
 - syntax errors indicated at correct place
 - linking elements with underlying knowledge
 - 'accessible' handling by blind people.
 - tool specific features require coding in Isabelle/Scala

Summary

- We have shown
 - generic tools of Isabelle/Isar
 - to connect various math SW with Isabelle/jEdit
 - examples for such SW in ML and Haskell
 - the connection **auto-magically** provides formulas in Isabelle quality:
 - sub-term structure by mouse hovering
 - syntax errors indicated at correct place
 - linking elements with underlying knowledge
 - 'accessible' handling by blind people.
 - tool specific features require coding in Isabelle/Scala

Conclusion

- In conclusion, we note that
 - there is no separated API in Isabelle
(that would hinder further development)
 - rather there are
 - rigorous coding standards ('code is the documentation')
 - software architecture with a high level of abstraction
 - comprehensive documentation
 - Implementing Isabelle/jEdit as a user interface requires
 - a distinguished amount of development work
 - *a secure funding*
- We invite everyone to consider
whether their software in mathematics or logic
is worth the effort to create ...
 - ... **an interface with the high quality of Isabelle/jEdit**
that is constantly evolving.

Conclusion

- In conclusion, we note that
 - there is no separated API in Isabelle
(that would hinder further development)
 - rather there are
 - rigorous coding standards ('code is the documentation')
 - software architecture with a high level of abstraction
 - comprehensive documentation
 - Implementing Isabelle/jEdit as a user interface requires
 - a distinguished amount of development work
 - *a secure funding*
- We invite everyone to consider
whether their software in mathematics or logic
is worth the effort to create ...
 - ... **an interface with the high quality of Isabelle/jEdit**
that is constantly evolving.

Plenary Demo:

Isabelle/jEdit — a Generic Frontend for Formal Mathematics

Thank you for attention !

Surprises? Questions?