THE UNIVERSITY OF
NEW SOUTH WALES

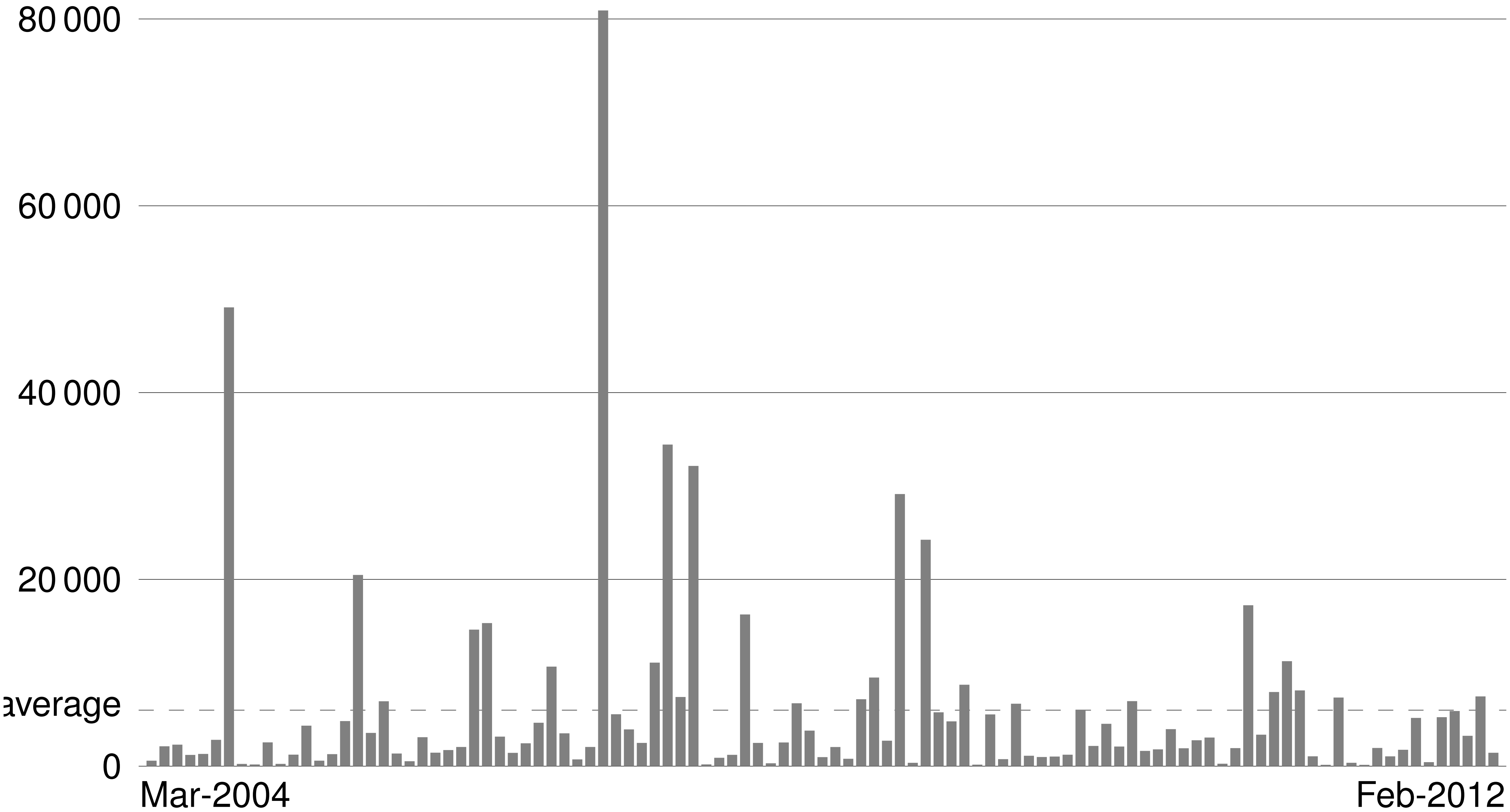
Eisbach: An Isabelle Proof Method Language

Daniel Matichuk

Makarius Wenzel, Toby Murray

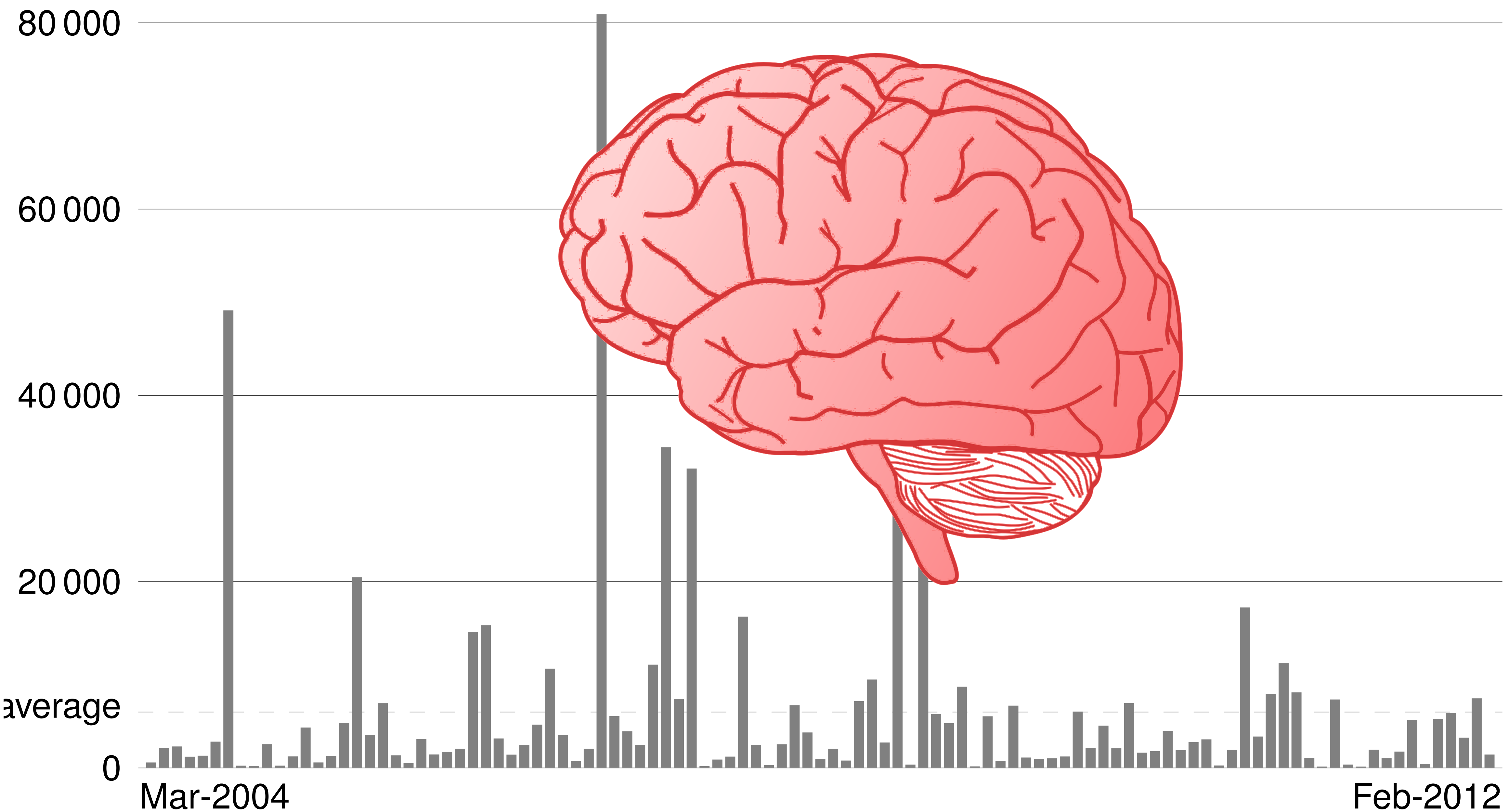
ITP 2014

Proof Engineering



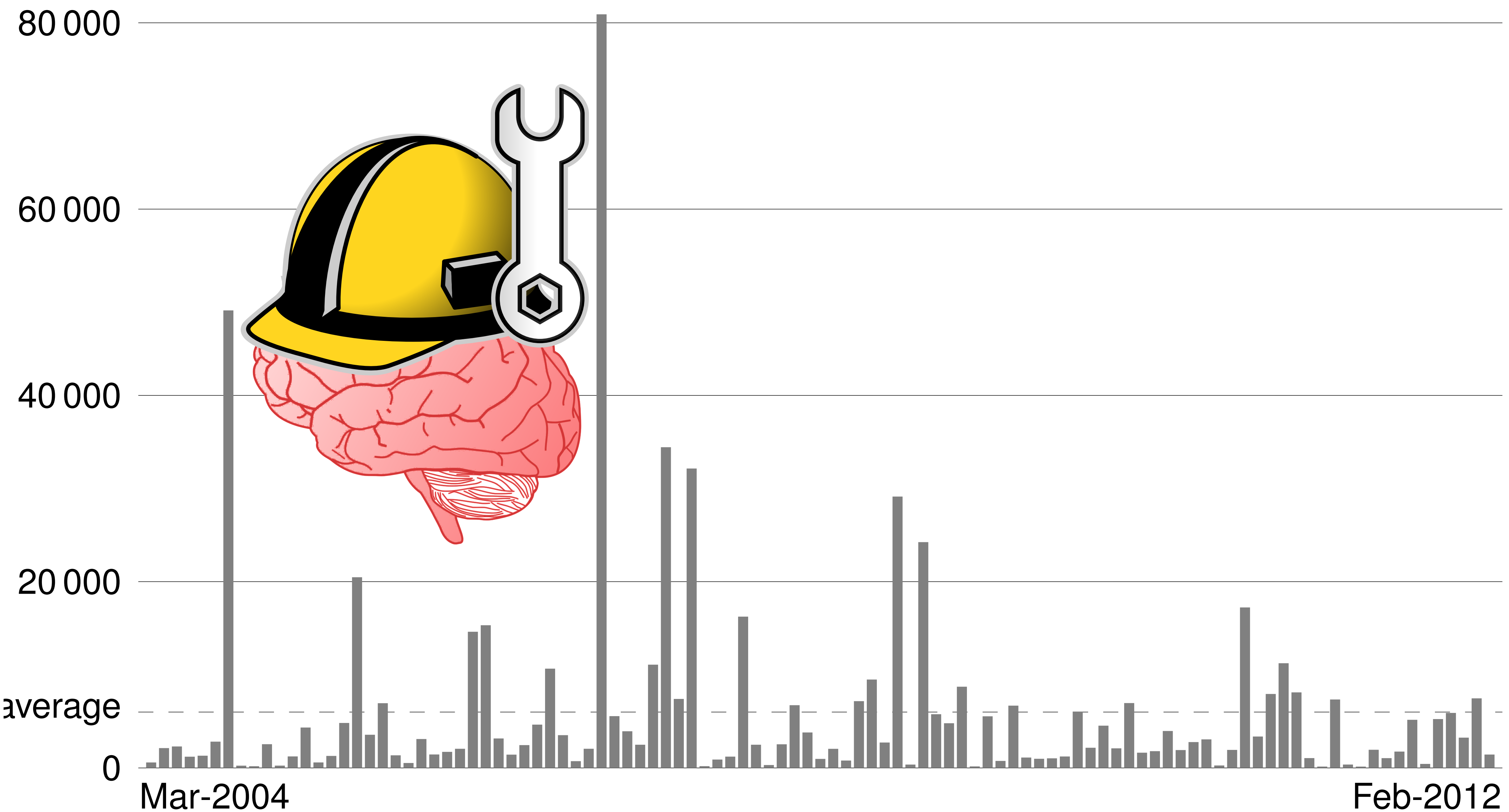
Size distribution of AFP entries in lines of proof,
sorted by submission date

Proof Engineering



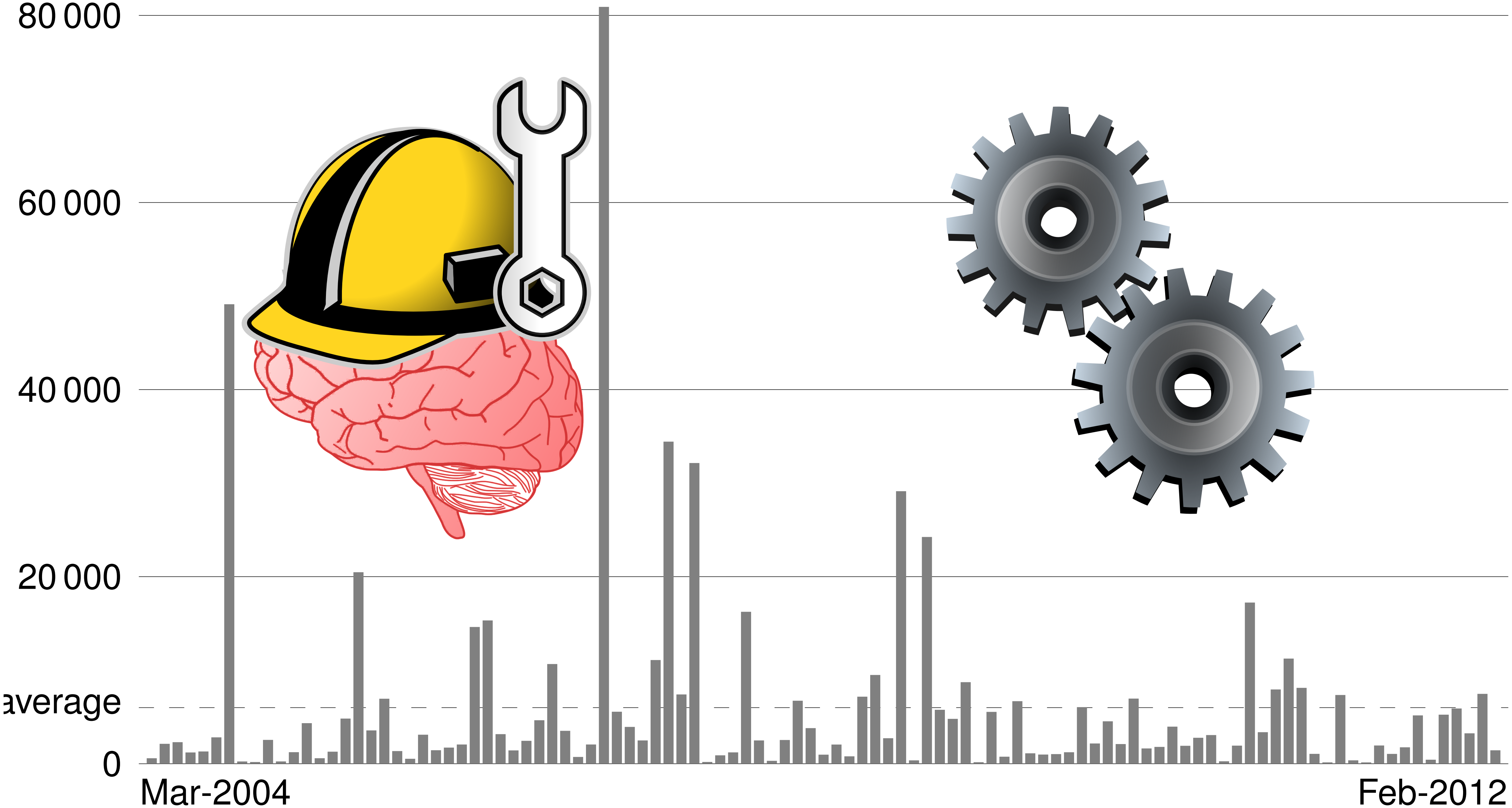
Size distribution of AFP entries in lines of proof, sorted by submission date

Proof Engineering



Size distribution of AFP entries in lines of proof,
sorted by submission date

Proof Engineering



Size distribution of AFP entries in lines of proof,
sorted by submission date

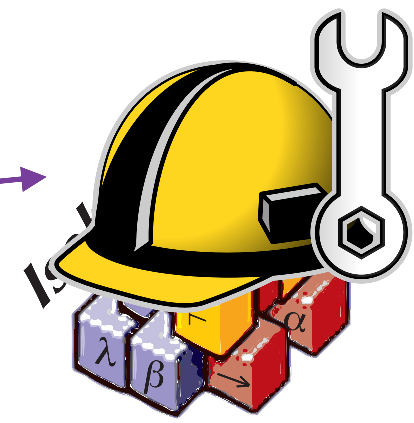
Outline

Isabelle Concepts
- Isar
- Proof Methods

Eisbach
- Easy Custom Proof Methods
- Demo

Evaluation/Future
-Existing method rewritten
-Tracing/Debugging…

Isabelle Concepts



Isabelle Concepts
- Isar
- Proof Methods

Eisbach
- Easy Custom Proof Methods
- Demo

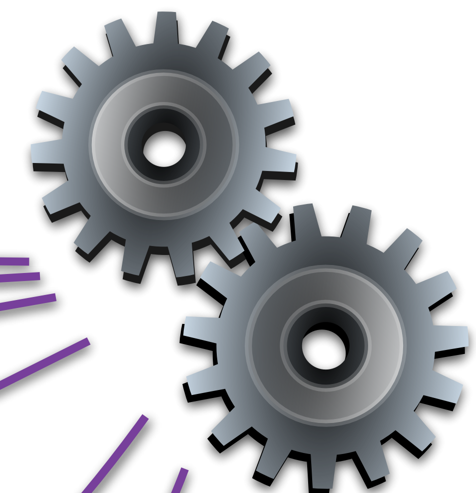
Evaluation/Future
-Existing method rewritten
-Tracing/Debugging…

Isabelle/Isar

theorem $\mathit{Knaster\text{-}Tarski}$:
  assumes mono: $\bigwedge x\ y.\ x \leq y \Longrightarrow f\ x \leq f\ y$
  shows $f\ (\bigsqcap \{x.\ f\ x \leq x\}) = \bigsqcap \{x.\ f\ x \leq x\}$ (**is** $f\ ?a = ?a$)
proof $-$
  have $*$: $f\ ?a \leq ?a$ (**is** - $\leq \bigsqcap ?H$)
  proof
    fix x **assume** H: $x \in ?H$
    then have $?a \leq x$ **..**
    also from H **have** $f\ \ldots \leq x$ **..**
    moreover note mono **finally show** $f\ ?a \leq x$ **.**
  qed
  also have $?a \leq f\ ?a$
  proof
    from mono **and** $*$ **have** $f\ (f\ ?a) \leq f\ ?a$ **.**
    then show $f\ ?a \in ?H$ **..**
  qed
  finally show $f\ ?a = ?a$ **.**
qed

Isabelle/Isar

theorem $\mathit{Knaster\text{-}Tarski}$:
  assumes mono: $\bigwedge x\ y.\ x \le y \Longrightarrow f\ x \le f\ y$
  shows $f\ (\bigsqcap\{x.\ f\ x \le x\}) = \bigsqcap\{x.\ f\ x \le x\}$ (**is** $f\ ?a = ?a$)
proof $-$
  have $*$: $f\ ?a \le ?a$ (**is** $\text{-} \le \bigsqcap\ ?H$)
  proof
    fix x **assume** H: $x \in ?H$
    then have $?a \le x$ **..**
    also from H **have** $f\ \ldots \le x$ **..**
    moreover note mono **finally show** $f\ ?a \le x$ **.**
  qed
  also have $?a \le f\ ?a$
  proof
    from mono **and** $*$ **have** $f\ (f\ ?a) \le f\ ?a$ **.**
    then show $f\ ?a \in ?H$ **..**
  qed
  finally show $f\ ?a = ?a$ **.**
qed

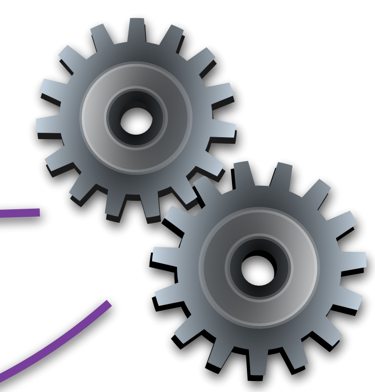


Isabelle/Isar

theorem *Knaster-Tarski'*:
  assumes *mono*[*intro*!]: $\bigwedge x\ y.\ x \le y \Longrightarrow f\ x \le f\ y$
 shows $f\ (\bigsqcap\ \{x.\ f\ x \le x\}) = \bigsqcap\ (\{x.\ f\ x \le x\})$ (**is** $f\ ?a = ?a$)
 proof –
  have $*$: $f\ ?a \le ?a$ **by** (*clarsimp*,*rule order*.*trans*, *fastforce*)
  also have $?a \le f\ ?a$ **by** (*fastforce intro*!: $*$)
  finally show $f\ ?a = ?a$ **.**
 qed

Isabelle/Isar

theorem *Knaster-Tarski'*:
  assumes *mono*[*intro*!]: $\bigwedge x\ y.\ x \le y \Longrightarrow f\ x \le f\ y$
 shows $f\ (\bigsqcap\ \{x.\ f\ x \le x\}) = \bigsqcap\ (\{x.\ f\ x \le x\})$ (**is** $f\ ?a = ?a$)
 proof –
  have $*$: $f\ ?a \le ?a$ **by** (*clarsimp*,*rule order.trans*, *fastforce*)
  also have $?a \le f\ ?a$ **by** (*fastforce intro*!: $*$)
  finally show $f\ ?a = ?a$ **.**
 qed



Proof Methods

$$\textbf{have}\ *\!: f\ ?a \le\ ?a\ \textbf{by}\ (\mathit{clarsimp}, \mathit{rule}\ \mathit{order.trans},\ \mathit{fastforce})$$

Goal

Method

Combinator

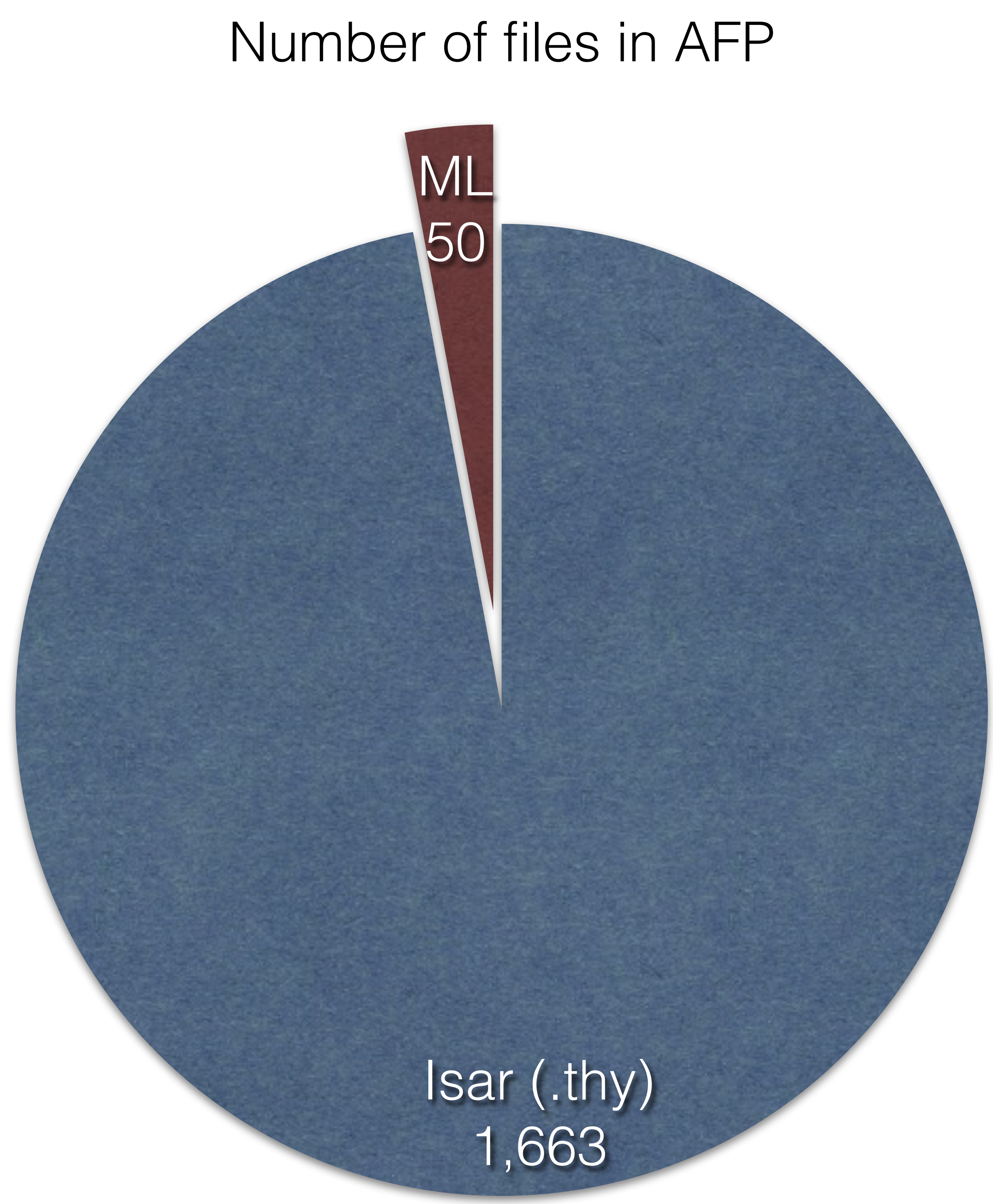
$$\textbf{also have}\ ?a \le f\ ?a\ \textbf{by}\ (\mathit{fastforce}\ \mathit{intro}!\!: *)$$

Method Parameter

Isabelle/ML

theorem *Knaster-Tarski'*: $(\bigwedge x\ y.\ x \le y \Longrightarrow f\ x \le f\ y) \Longrightarrow$
$f\ (\bigsqcap\ \{x.\ f\ x \le x\}) = \bigsqcap\ (\{x.\ f\ x \le x\})$
apply (*tactic* ⟪ (*EqSubst.eqsubst-tac* @{*context*} [*0*] @{*thms order-eq-iff*} *1*)
THEN (*Tactic.resolve-tac* @{*thms context-conjI*} *1*)
THEN (*Tactic.resolve-tac* @{*thms Inf-greatest*} *1*)
THEN (*Tactic.forward-tac* @{*thms Inf-lower*} *1*)
THEN (*Clasimp.fast-force-tac* @{*context*} *1*)
THEN (*Tactic.resolve-tac* @{*thms Inf-lower*} *1*)
THEN (*Clasimp.fast-force-tac* @{*context*} *1*)
⟫)
done

Isabelle's AFP



seL4 - our experience

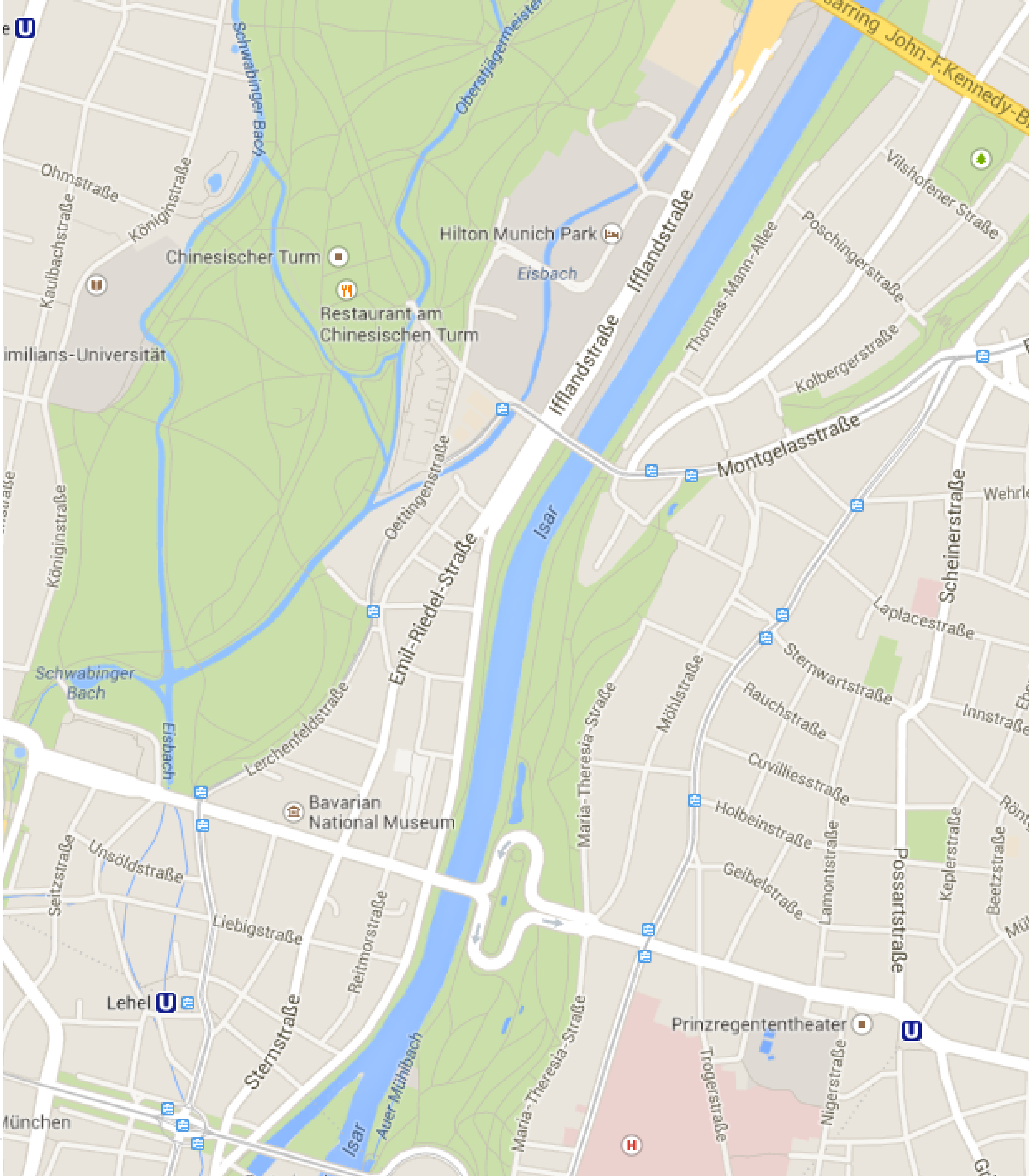
- Full functional correctness proof
  - Source code and Proof going open source!
  - http://seL4.systems for more info
  - July 29
- Isabelle proof methods developed
  - WP/WPC - vcg for monadic hoare logic
  - sep-* - automating separation logic
- Proof Engineers want more!
  - Languages like Ltac show this

Eisbach

Isabelle Concepts
- Isar
- Proof Methods

Eisbach
- Easy Custom Proof Methods
- Demo

Evaluation/Future
-Existing method rewritten
-Tracing/Debugging…


Ohmstraße
Königinstraße
Kaulbachstraße
Schwabinger Bach
Chinesischer Turm
Restaurant am Chinesischen Turm
Hilton Munich Park
Eisbach
Ifflandstraße
Thomas-Mann-Allee
Poschingerstraße
Vilshofener Straße
Kolbergerstraße
Montgelasstraße
Oettingenstraße
Emil-Riedel-Straße
Isar
Königinstraße
Schwabinger Bach
Eisbach
Lerchenfeldstraße
Bavarian National Museum
Scheinerstraße
Laplacestraße
Sternwartstraße
Rauchstraße
Innstraße
Cuvilliesstraße
Holbeinstraße
Geibelstraße
Lamontstraße
Possartstraße
Keplerstraße
Beetzstraße
Maria-Theresia-Straße
Möhlstraße
Unsöldstraße
Seitzstraße
Liebigstraße
Reitmorstraße
Sternstraße
Lehel
Isar
Auer Mühlbach
Maria-Theresia-Straße
Prinzregententheater
Trogerstraße
Nigerstraße
Praterinsel


Eisbach
Chinesischer Turm
Restaurant am Chinesischen Turm
Hilton Munich Park
Schwabinger Bach
Oberstjägermeister
Ohmstraße
Königinstraße
Kaulbachstraße
Ifflandstraße
Thomas-Mann-Allee
Poschingerstraße
Vilshofener Straße
Kolbergerstraße
Montgelasstraße
Oettingenstraße
Emil-Riedel-Straße
Isar
Lerchenfeldstraße
Bavarian National Museum
Unsöldstraße
Liebigstraße
Seitzstraße
Reitmorstraße
Sternstraße
Maria-Theresia-Straße
Möhlstraße
Sternwartstraße
Rauchstraße
Cuvilliesstraße
Holbeinstraße
Geibelstraße
Lamontstraße
Laplacestraße
Scheinerstraße
Possartstraße
Keplerstraße
Beetzstraße
Innstraße
Prinzregententheater
Trogerstraße
Nigerstraße
Auer Mühlbach
Praterinsel
Wehrle

NICTA

Language Elements

- Integrates existing/new methods
  - fastforce, simp, auto…
- Abstract over Terms/Facts/Methods
- Attributes for method hints
  - simp, intro, my_vcg_rules…
- Matching provides control flow
  - Match and bind higher-order patterns against focused subgoal elements

Eisbach

method-definition *induct-list* **facts** *simp* $=$
$($**match** *?concl* **in** *?P* $(?x :: {}'a\ list) \Rightarrow (induct\ ?x \mapsto fastforce\ simp: simp))$

$\Downarrow$

lemma $length\ (xs\ @\ ys) = length\ xs + length\ ys$ **by** *induct-list*

Eisbach - Design goals

- Easy for beginners and experts
  - Familiar method syntax from Isar
- Limited functionality - leave complexity to Isabelle/ML
- Integration with other Isabelle languages
- Readable proof procedures

Eisbach - Combinators

- Standard Isar Method Combinators
  - "|" - alternative composition
  - "," - sequential composition
  - "?" - suppress failure (try)
  - "+" - repeated application

- New Combinator
  - "$\mapsto$" - compose with emerging subgoals

$$\textbf{method-definition}\ \textit{prop-solver}_1 = ((\textit{rule impI}, (\textit{erule conjE})?) \mid \textit{assumption})+$$

Eisbach - Abstraction

- Parameterize over facts, terms, and methods

Method "Signature"

$$\textbf{method-definition}\ \textit{prop-solver}_2\ \textbf{facts}\ \textit{intro elim} = \\ ((\textit{rule intro}, (\textit{erule elim})?) \mid \textit{assumption})+$$

Abstracted Facts

Eisbach - Abstraction

- Parameterize over facts, terms, and methods

Method "Signature"

method-definition *prop-solver*$_2$ **facts** *intro elim* $=$
$((\textit{rule intro}, (\textit{erule elim})?) \mid \textit{assumption})+$

Abstracted Facts

Fact Arguments

lemma $P \wedge Q \longrightarrow P$ **by** $(\textit{prop-solver}_2\ \textit{intro}\colon \textit{impI}\ \textit{elim}\colon \textit{conjE})$

Eisbach - Attributes

- New command: **declare-attributes**

declare-attributes *intro elim*

– Managed with the usual Isar **declare** command

declare *implI* [*intro*] **and** *conjE* [*elim*]

– Used at run-time by methods

method-definition $\textit{prop-solver}_3$ **facts** [*intro*] [*elim*] $=$
$((\textit{rule intro}, (\textit{erule elim})?) \mid \textit{assumption})+$

Eisbach - Attributes

- New command: **declare-attributes**

declare-attributes *intro elim*

– Managed with the usual Isar **declare** command

declare *implI* [*intro*] **and** *conjE* [*elim*]

Square brackets indicate fact parameter is managed by attribute

– Used at run-time by methods

method-definition *prop-solver*$_3$ **facts** [*intro*] [*elim*] $=$
$((rule\ intro,\ (erule\ elim)?)\ |\ assumption)+$

Eisbach - Attributes

- New command: **declare-attributes**

declare-attributes *intro elim*

– Managed with the usual Isar **declare** command

declare *implI* [*intro*] **and** *conjE* [*elim*]

Square brackets indicate fact parameter is managed by attribute

– Used at run-time by methods

method-definition $\textit{prop-solver}_3$ **facts** [*intro*] [*elim*] =
  ((*rule intro*, (*erule elim*)?) | *assumption*)+

Contains implI

Contains conjE

Eisbach - Attributes

NICTA

- New command: **declare-attributes**

declare-attributes *intro elim*

– Managed with the usual Isar **declare** command

declare *implI* [*intro*] **and** *conjE* [*elim*]

Square brackets indicate fact parameter is managed by attribute

– Used at run-time by methods

method-definition *prop-solver*$_3$ **facts** [*intro*] [*elim*] $=$
((*rule intro*, (*erule elim*)?) | *assumption*)+

Contains $implI$

Contains $conjE$

lemma $P \wedge Q \longrightarrow P$ **by** *prop-solver*$_3$

Eisbach - Matching

- Higher-order matching for control flow
  - Bind matched patterns

$$\textbf{method-definition}\ \textit{solve-ex} =$$
$$(\textbf{match}\ \textit{?concl}\ \textbf{in}\ \exists x.\ ?Q\ x \Rightarrow$$
$$(\textbf{match}\ \textit{prems}\ \textbf{in}\ U\!: Q\ ?y \Rightarrow (\textit{rule exI}\ [\textit{where}\ x = y\ \textbf{and}\ P = Q,\ \textit{OF}\ U])))$$

Eisbach - Matching

- Higher-order matching for control flow
  - Bind matched patterns

Special term $?concl$ is current subgoal

$$
\begin{array}{l}
\textbf{method-definition}\ \mathit{solve\text{-}ex} = \\
\quad (\textbf{match}\ ?concl\ \textbf{in}\ \exists x.\ ?Q\ x \Rightarrow \\
\quad\quad (\textbf{match}\ \mathit{prems}\ \textbf{in}\ U\!: Q\ ?y \Rightarrow (\mathit{rule}\ \mathit{exI}\ [\mathit{where}\ x = y\ \textbf{and}\ P = Q,\ \mathit{OF}\ U])))
\end{array}
$$

Eisbach - Matching

- Higher-order matching for control flow
  - Bind matched patterns

Special term $?concl$ is current subgoal

Matched pattern $?Q$ is bound

$$\textbf{method-definition}\ \mathit{solve\text{-}ex} =$$
$$(\textbf{match}\ ?concl\ \textbf{in}\ \exists x.\ ?Q\ x \Rightarrow$$
$$(\textbf{match}\ \mathit{prems}\ \textbf{in}\ U\!: Q\ ?y \Rightarrow (\mathit{rule\ exI}\ [\mathit{where}\ x = y\ \textbf{and}\ P = Q,\ \mathit{OF}\ U])))$$

Eisbach - Matching

- Higher-order matching for control flow
  - Bind matched patterns

Special term $?concl$ is current subgoal

Matched pattern $?Q$ is bound

$$
\begin{array}{l}
\textbf{method-definition}\ \mathit{solve\text{-}ex} = \\
\quad (\textbf{match}\ \mathit{?concl}\ \textbf{in}\ \exists x.\ ?Q\ x \Rightarrow \\
\quad\quad (\textbf{match}\ \mathit{prems}\ \textbf{in}\ U\colon Q\ ?y \Rightarrow (\mathit{rule\ exI}\ [\mathit{where}\ x = y\ \textbf{and}\ P = Q,\ \mathit{OF}\ U])))
\end{array}
$$

Special fact $prems$ is current premises

Eisbach - Matching

- Higher-order matching for control flow
  - Bind matched patterns

Special term $?concl$ is current subgoal

Matched pattern $?Q$ is bound

$$\textbf{method-definition}\ solve\text{-}ex = \\ (\textbf{match}\ ?concl\ \textbf{in}\ \exists x.\ ?Q\ x \Rightarrow \\ (\textbf{match}\ prems\ \textbf{in}\ U\colon Q\ ?y \Rightarrow (rule\ exI\ [where\ x = y\ \textbf{and}\ P = Q,\ OF\ U])))$$

Special fact $prems$ is current premises

Matching singleton fact U is bound

Focus/Matching

- Problem: Raw subgoals are unstructured

$$\bigwedge x.\ A\ x \Longrightarrow B\ x \Longrightarrow A\ x \wedge B\ x$$

Focus/Matching

- Problem: Raw subgoals are unstructured

$$\bigwedge x.\ A\ x \Longrightarrow B\ x \Longrightarrow A\ x \wedge B\ x$$

$$\textbf{by}\ (\mathit{rule}\ \mathit{conjI}[\mathit{OF}\ \mathit{assms}(1)\ \mathit{assms}(2)])$$

Focus/Matching

- Problem: Raw subgoals are unstructured

$$\bigwedge x.\ A\ x \Longrightarrow B\ x \Longrightarrow A\ x \wedge B\ x$$

$$\cancel{\mathbf{by}\ (\mathit{rule}\ \mathit{conjI}[\mathit{OF}\ \mathit{assms}(1)\ \mathit{assms}(2)])}$$

Focus/Matching

- Problem: Raw subgoals are unstructured

$$\bigwedge x.\ A\ x \Longrightarrow B\ x \Longrightarrow A\ x \wedge B\ x$$

$$\xcancel{\textbf{by}\ (\mathit{rule}\ \mathit{conjI}[\mathit{OF}\ \mathit{assms}(1)\ \mathit{assms}(2)])}$$

- Goal:

$$\begin{array}{l}
\textbf{method-definition}\ \mathit{solve\text{-}conj} = \\
\quad (\textbf{match}\ \mathit{?concl}\ \textbf{in}\ ?P \wedge ?Q \Rightarrow \\
\qquad (\textbf{match}\ \mathit{prems}\ \textbf{in}\ U\colon P\ \textbf{and}\ U'\colon Q \Rightarrow \\
\qquad\quad (\mathit{rule}\ \mathit{conjI}[\mathit{OF}\ U\ U'])))
\end{array}$$

Focus/Matching

- Problem: Raw subgoals are unstructured

$$\bigwedge x.\ A\ x \Longrightarrow B\ x \Longrightarrow A\ x \wedge B\ x$$

$$\cancel{\textbf{by}\ (\textit{rule conjI}[\textit{OF assms}(1)\ \textit{assms}(2)])}$$

- Goal:

Find and name assumptions through matching

$$\begin{array}{l}
\textbf{method-definition}\ \textit{solve-conj} = \\
\quad (\textbf{match}\ \textit{?concl}\ \textbf{in}\ \textit{?P} \wedge \textit{?Q} \Rightarrow \\
\qquad (\textbf{match}\ \textit{prems}\ \textbf{in}\ U\colon P\ \textbf{and}\ U'\colon Q \Rightarrow \\
\qquad\quad (\textit{rule conjI}[\textit{OF}\ U\ U'])))
\end{array}$$

Focus

- Solution: Focusing
  - Based on existing work

$$\bigwedge x.\ A\ x \Longrightarrow B\ x \Longrightarrow A\ x \wedge B\ x$$

Focus

- Solution: Focusing
  - Based on existing work

$$\bigwedge x.\ A\ x \Longrightarrow B\ x \Longrightarrow A\ x \wedge B\ x$$

$$\begin{array}{l}\textbf{fixes}\ x\\ \textbf{assumes}\ A\ x\ \textbf{and}\ \ B\ x\\ \textbf{shows}\ \ A\ x \wedge B\ x\end{array}$$

Focus

- Solution: Focusing
  - Based on existing work

$$\bigwedge x.\ A\ x \Longrightarrow B\ x \Longrightarrow A\ x \wedge B\ x$$

$$\Downarrow$$

fixes x

assumes $A\ x$ **and** $B\ x$ $\longrightarrow$ *prems*

shows $A\ x \wedge B\ x$ $\longrightarrow$ *?concl*

Demo

Evaluation/Future work

Isabelle Concepts
- Isar
- Proof Methods

Eisbach
- Easy Custom Proof Methods
- Demo

Evaluation/Future
-Existing method rewritten
-Tracing/Debugging…

Tactic Languages are not new

- Ltac
  - Untyped High-level tactic language for Coq
  - Goal matching, iteration, recursion
- VeriML
  - Dependently typed tactic language
  - Provides strong static guarantees
- Mtac
  - Typed tactic language for Coq
  - Leverages built-in Coq notion of computation
  - Strong static guarantees

Current Results

- Eisbach
  - Extension of Isar, Isabelle's proof language
  - Integrates with existing Isar syntax
    - methods
    - attributes
- Evaluation
  - Existing methods rewritten in Eisbach
    - WP, WPC: l4.verified invariant proof successfully checked
- Future Work
  - Tracing/Debugging
  - Optimisations

Conclusion

- Proof Engineers need tools
  - to write proofs at scale
- Isar provides structure/syntax for **proofs**
  - Most Isabelle users most familiar with Isar
- Eisbach provides easy mechanisms for writing **automation**
  - abstraction
  - matching
  - backtracking
  - recursion
- Coming soon…

Thank You!