Eating Your Own Dog Food

Andreas Abel\textsuperscript{1}

\textsuperscript{1}Department of Computer Science and Engineering
Chalmers and Gothenburg University, Sweden

25th Agda Implementor’s Meeting (AIM XXI)
Teknikparken, Chalmers, Gothenburg, Sweden
9 May 2017
Agda: Inside View

- 3 main developers (part-time)
- 1 packaging, integration, and release engineer (part-time)
- A dozen contributors (sporadically active)
- A long tail of single-patch submitters
- 103,000 loc (3.9MB) [2015: 93,000 loc; 2014: 70,000 loc]
- 2,461 issues on the bug tracker (404 open) [2015: 1,528 (249); 2014: 1,076 (165)]

<table>
<thead>
<tr>
<th>Year</th>
<th>Issues</th>
<th>Bugs/Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-2010</td>
<td>300</td>
<td>100</td>
</tr>
<tr>
<td>2010-2013</td>
<td>600</td>
<td>200</td>
</tr>
<tr>
<td>2013-2015</td>
<td>600</td>
<td>300</td>
</tr>
<tr>
<td>2015-2017</td>
<td>900</td>
<td>&gt;400</td>
</tr>
</tbody>
</table>
# Agda: code by components

<table>
<thead>
<tr>
<th>Component</th>
<th>loc (’17)</th>
<th>loc (’15)</th>
<th>loc (’14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Utility functions</td>
<td>6800</td>
<td>8000</td>
<td>4100</td>
</tr>
<tr>
<td>Syntax (parse print scope)</td>
<td>23500</td>
<td>19000</td>
<td>16000</td>
</tr>
<tr>
<td>Type checker (eval. cov. pos.)</td>
<td><strong>49000</strong></td>
<td>39000</td>
<td>30000</td>
</tr>
<tr>
<td>Termination checker</td>
<td>4900</td>
<td>5800</td>
<td>4600</td>
</tr>
<tr>
<td>Interaction (imp. highl. LaTeX)</td>
<td>8600</td>
<td>7400</td>
<td>6600</td>
</tr>
<tr>
<td>Agsy</td>
<td>4200</td>
<td>4200</td>
<td>4100</td>
</tr>
<tr>
<td>Compiler</td>
<td>5500</td>
<td>8000</td>
<td>5200</td>
</tr>
<tr>
<td>total</td>
<td><strong>103000</strong></td>
<td><strong>93000</strong></td>
<td><strong>71000</strong></td>
</tr>
</tbody>
</table>
Closed issue age (cumulative), 2015-08-08 to 2017-05-08

Days

Abel (GU)  Own Dog Food  AIM XXV
Closed issue age logarithmically (cumulative), 2015-08-08 to 2017-05-08
Eating Our Own Dog Food

- We are developing a language for verification.
- Shouldn’t we eat our own dog food?
- Develop Agda in Agda?
- Certainly a grand challenge.
Looking at our development process

Agda in Agda is out of reach, for a while

- We do not even have rigorous pen or paper proofs for our theory.
- We do not even have a theory.
- Most of the theoretical work focuses on the semantics.
- Semantics of small fragments of the language.
- Formalization of decidability $\Pi + \text{Set}_0 + \mathbb{N}$: half a year of dedicated Agda grinding (Joakim hman).
- No advanced technology formalized: unification, pattern matching, termination . . .
Santa Claus

Let’s get millions of EUR from the EU!
Can we at least apply some dependent types to implement dependent types?

- Well-scoped syntax via Haskell GADTs.
- Could take care of de Bruijn index bugs.
- How does this scale to pattern matching, unification, with-abstraction?
- Type-checking monad needs to be indexed by the context length.
- First try this on a prototype!
- An advanced master thesis?
Prior to Formal Methods

- Agda is a programming language, so it should have a **SPECIFICATION**
- Write an informal specification!
- Flesh it out with test cases.
- We will have plenty of deviations from the specification.
- We can have testcases that *should* work but do not yet.
- We can gradually approach correctness.
Code quality: documentation

- Module documentation.
- Algorithm explanation by example.
- Documentation of our data structures.
- Pre- and postconditions of functions.
- 100% haddock coverage.
- Can only be reached by code reviewing requesting documentation.
Looking at our development process

Code quality: structure

- Break long procedures into several components!
- Write out the properties of the components.
- We have legacy spaghetti code with impenetrable control flow: `Interaction.Imports`
- Genesis: patches-over-patches
- Counterculture: refactorings, data structure evolution!
- We have monolithic state (`TCState`): pre-OO imperative programming.
- *Gegenentwurf*: modular monadic programming.
Long on the Wish List (2015)

- User manual
- Packaging
- Type classes (WIP)
- Universe cumulativity
- Reflection/tactics (WIP)
- Efficient type-checking
- Usable compiler (WIP)
What happened to the 2015 wish list?

Core Language / Internal Syntax (2015)

- Sharing
- Independent checking
- Termination certificates
- Shared optimizations/transformations used by compiler backends
Research topics (2015)

- Equality (HoTT, OTT)
- Parametricity/colors
- Sized dependent types
- Proof-instance search and unification
- Foundation for hidden/named arguments
- Telescopes/Σ-types at framework level
- Printing
Action items for us

- Install a coding code of conduct (à la style guide).
- Install mandatory code reviews.
- Framework for test-backed specification.
- Stabilize core features of Agda.