The Present and Future of Agda

Andreas Abel\textsuperscript{1}

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

Programming Logic Day
Ågrenska Villa, Gothenburg, Sweden
7 March 2014
Agda: Outside View

- A language to formalize math and computer science
- A language to write papers (96 on the wiki)
- A language for teaching (15 courses on the wiki)
- A brand: *Coq and Agda*
- A user community
  - Martin Escardo, Wolfram Kahl, Dan Licata, Serge Mechveliani, . . .
Competition

- Coq: different league
- Epigram? (prefectioned to death?)
- Idris!
  - Built-in types and Prelude
  - Easy-to-use FFI
  - Good compilers
  - Emphasis on programming
  - Catching up in interactive development
  - But: still more “prototype”
- Haskell
Agda: Inside View

- 3 main developers (part-time)
- a dozen contributors (sporadically active)
- a long tail of single-patch submitters
- 70,000 loc
- 1076 issues on the bug tracker (165 open)
  - 2007-2010 100 bugs/year
  - 2010-2013 200 bugs/year
  - 2013- 300 bugs/year ?
## Agda: Where do the 70kloc come from?

<table>
<thead>
<tr>
<th>Component</th>
<th>loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Utility functions</td>
<td>4100</td>
</tr>
<tr>
<td>Syntax (Concrete, Abstract)</td>
<td>6600</td>
</tr>
<tr>
<td>Parser</td>
<td>3300</td>
</tr>
<tr>
<td>Printer</td>
<td>1900</td>
</tr>
<tr>
<td>Scope checker</td>
<td>2700</td>
</tr>
<tr>
<td>Syntax (Internal)</td>
<td>1400</td>
</tr>
<tr>
<td>Type checker (+ eval, coverage)</td>
<td>29600</td>
</tr>
<tr>
<td>Termination checker</td>
<td>4600</td>
</tr>
<tr>
<td>Interaction (+imports)</td>
<td>6600</td>
</tr>
<tr>
<td>Agsy</td>
<td>4100</td>
</tr>
<tr>
<td>Compiler</td>
<td>5200</td>
</tr>
</tbody>
</table>
Introduction

Open issues

- 2007: 0
- 2008: 10
- 2009: 20
- 2010: 30
- 2011: 50
- 2012: 70
- 2013: 100
- 2014: 150

The number of open issues has increased significantly from 2007 to 2014.
Long on the Wish List

- User manual
- Packaging
- Type classes
- Universe cumulativity
- Reflection/tactics
- Efficient type-checking
- Usable compiler
Core Language / Internal Syntax

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

- Equality (HoTT, OTT)
- Parametricity/colors
- Sized dependent types
- Proof-instance search and unification
- Foundation for hidden/named arguments
- Telescopes/$\Sigma$-types at framework level
- Printing
Get Involve(d!/ments)

- Critique: closed developing process
- Move to github
- Developer documentation
- Advertise projects
- More funding?
Related Work