Tock: 18 Months On

(this was originally “One Year On”,
but it's been a while since the last CPA...)

Adam T. Sampson
Neil C. C. Brown

Computing Laboratory, University of Kent

University of Kent | Computing

What's Tock?

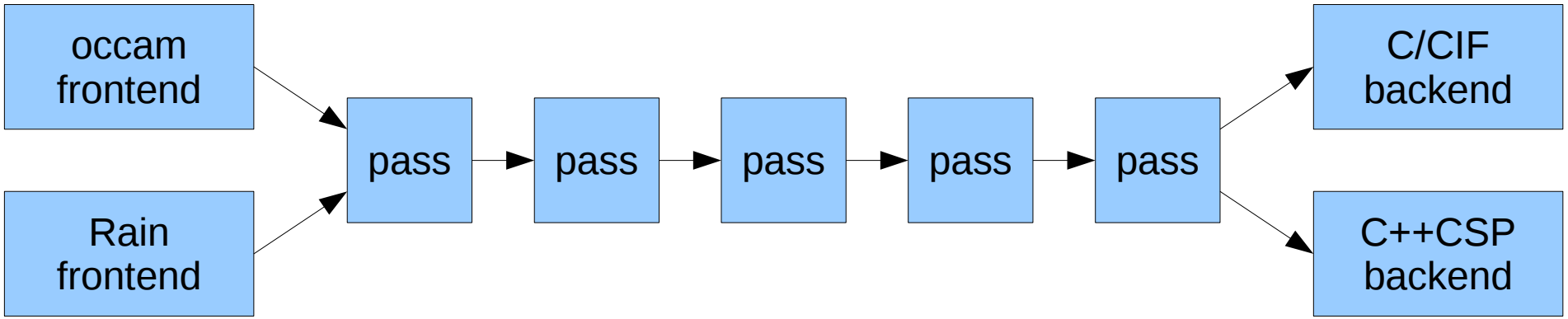
- A new compiler for concurrent languages
- Provide a solid basis for future work
- Makes experimenting with new features easy
- Compiles into C or C++
  - Highly portable (e.g. embedded devices, supercomputers)
  - Can use existing C compiler as backend
  - Decent performance for straightline code

Implementation

- Written in Haskell
  - Statically-typed, lazy, purely-functional language
  - Seems to have lots of users interested in compilers and concurrency already (see other presentations)
  - Our undergrads have to learn Haskell anyway

Nanopasses

- Uses “nanopass” approach
  - Lots of small passes
  - Easy to extend and to test



Overview

- We've done various things with Tock lately:
  - Rain frontend
  - C++CSP backend
  - Usage checking
  - Automated testing
  - Smart pattern-matching
  - Performance improvements
- Some of these may be of use to you...

Rain

- Tock supports multiple source languages
- We've implemented a frontend for Rain
  - Neil's new concurrent language
- We reused most of the existing code
  - A few Rain-specific bits
  - Other parts have been refactored to be more general

C++CSP

- Added a backend for C++CSP – process-oriented runtime library for C++
  - A bit more portable than CCSP (but also slower)
- We developed some useful Haskell techniques to share a lot of code between the C and C++ backends
  - e.g. “WHILE” is the same; “PAR” isn't

Usage checking, from alpha...

- The existing occ21 parallel usage checker has several problems
  - Works by expanding out PAR blocks and checking each case individually
  - No dynamic PAR replication counts
  - Very slow for large programs
- We need a better solution

... to Omega

- We use Pugh's Omega test algorithm
  - Very efficient integer constraint solver
  - Designed for this sort of application (e.g. SPoC)
- Extended algorithm to support occam's remainder operator \
- Translate code to constraints, then solve
  - e.g. “P writes to i[n], Q reads from i[n+1]”
  - If there's a solution, it's not safe!

Automated testing

- Tock now has a large automated test suite
- *HUnit:* unit tests for functions and passes
- *QuickCheck*: test passes with randomly-generated data, and ensure properties hold
  - Works best for mathsy stuff (e.g. Omega test)
- *Full-toolchain tests*: real applications, and code fragments that test particular cases
  - ... including the existing occam-pi test suite

Smart pattern-matching

- Pattern-matching is very common in compilers
- Good ways of doing this in dynamically-typed languages like Scheme... but not Haskell yet
- We've developed a generics-based pattern matching library for Haskell
  - Reusable pattern fragments
  - Fuzzy matching

Performance improvements

- When we presented Tock at CPA2007, we were a bit concerned about the compiler's performance
- It turns out that while lots of people have written compilers in Haskell...
  - ... not many had written compilers with 50+ passes and over a million data values in the AST of a large program
- So we had some scalability problems to solve

Making passes faster

- We've developed a new generics system for writing transformation/analysis passes
  - Also useful for other generic programming problems
- Dynamically prunes the AST search based on what it's looking for
  - e.g. you aren't going to find a PAR inside a type
- Avoids runtime type introspection by using typeclasses to do more work at compile time

Making Tock less lazy

- Haskell is a *lazy* language
  - Defer calculations until they're actually needed
- Allows some cool tricks (e.g. infinite lists), but has memory usage problems
- We found some common Haskell standard library features were excessively lazy...
  - ... but we've fixed the problems
  - Tock now uses less memory than occ21!

Where next?

- Full occam-pi support
  - The infrastructure is there now
- Bytecode backend for very small devices

That's all, folks

- All the above work has been written up...
  - ... although not all of it has been published yet
  - Ask us if you're interested in reading more!
- See: `http://offog.org/tock`
- Any questions?