SoSlang: SOurce-to-Source Clang

Mehdi Amini, Béatrice Creusillet, Ronan Keryell

SILKAN Wild Systems
Los Altos, CA, USA

November 8th 2012
Introduction

SILKAN: French company with 60 persons worldwide

- Designs, develops, integrates, delivers and supports high-performance, cost-effective simulation systems and solutions
- Use source-to-source compilation techniques with a team in Los Altos (<ad>we are hiring</ad>)

Question often asked

“Why using source-to-source instead of native compilation?”
Source-to-Source Was Good for Us

Worked since 1992 with the PIPS Fortran & C source-to-source interprocedural framework with polyhedra-based abstract interpretation

- From the team that introduced polyhedral compilation in the 80’s (Feautrier, Triolet, Irigoin) at MINES ParisTech
- Automatic parallelization from Fortran to Cray Fortran, Connection Machine Fortran (do you remember?)
- HPF (High Performance Fortran) compiler (was the fancy thing at that time)
- Virtual distributed shared memory with PVM
- Code obfuscation
- High-level hardware synthesis
- Code verification
- Vectorization inside OpenCL kernels
- OpenMP to MPI compilation
More recently at SILKAN

- Par4All open source automatic parallelizer based on PIPS
  - Fortran $\rightarrow$ OpenMP
  - C $\rightarrow$ OpenMP (SMP)
  - C $\rightarrow$ CUDA & OpenCL (GPU, MPSoC)
- Scilab/MATLAB $\rightarrow$ C $\rightarrow$ OpenMP/CUDA/OpenCL
- Parallelization of radar-oriented DSL
- Microcode generation for FPGA-based SIMD Ter@pix (hint: use intrinsics!)
- Code generation for dataflow-machine (CEA SCMP)
- Binary decompilation ($\text{objdump} \mid \text{python} \mid \text{PIPS}$)
- SME-C $\text{#pragma}$ to OpenCL & MCAPI for ST Microelectronics
  STHORM MPSoC, based on ROSE Compiler
Source-to-source *May Even Be Good for You*

- Model Driven Architecture: modeling is not enough... do the code transformations!
- Deep code refactoring
- Portable parallelization & vectorization
  - Conformance to newer/older/safer/... standards
  - Conversion of the apps in Google Play Store & Apple App Store to Windows Phone Store
    (PS for M$: send us a big check off-line on this subject 😊)
  - ...
- Easier debug of transformations
- Human readable output (exit the source LLVM backend)
- ...

**Use the Source, Luke**...

or in a higher order logic

**Use the Source-to-Source, Luke**...
Automated Transformations

Never send a human to do a machine’s job.

Agent SMITH.

In Matrix (Andy & Larry WACHOWSKI, 1999)
New Needs and Challenges

- More and more customers ask for C++ not dealt by PIPS
- Need tool with bigger backing community to provide stronger support
- Modern architectures are more and more parallel & heterogeneous
  - More open source and vendor tools to target
  - Generation of more different codes in interaction
Goals of SoSlang

1. Use C++ AST as THE Intermediate Representation
   - More and more tools generate C++ as a target
   - Fortran 2008 is basically C++, right? 😊
   - Use external translators to semantics-equivalent C++

2. Provide easy way to build complex source-to-source transformations over Clang

3. Link analyses to transformations

4. Interprocedural summarization of analyses and optimization

5. ...
Mutable Clang AST

- Currently, a Clang AST analysis is followed by a textual transformation
  → OK for renaming/refactoring, but what about:
    - Complex restructuring, like a loop fusion algorithm?
    - Chain of transformations → transformations $t_n$ requires analysis on the output of $t_{n-1}$
- Regenerate source code from the AST → requires bulletproof prettyprinter
- Clang misses an interface to provide easy mutation in the Clang AST
  → Ever-recurring request/question on `cf{e-dev}@` (...,3/11,4/11,11/11,12/11,1/12,4/12,...)
They were/are using this path:

- ROSE compiler framework (LLNL)
- PIPS (CRI/MINES ParisTech)
- Cetus (Purdue)
- DMS Software Reengineering Toolkit
- Parafrase (CSRD/UIUC)
- Polaris (CSRD/UIUC)
- SUIF (Stanford)
- ...

This can be done!
Convince (at least) a sub-community of Clang/LLVM there is need too for robust AST-only source-to-source tool 😊

Then, few simpler technical details 😊 to solve

- Clang AST designed to be immutable: is it a technical or a social issue?
- Design somehow stable API over Clang moving internals?
- Separation of concerns: what has to be done in Clang and what is part of a separate framework?

Yeah... more questions than answers right now!
Conclusion

- Source-to-source is an important cause
- Value of a program: its sources!
- In the good back-ends we trust 😊
- Clang can be a great tool for deeper source-to-source
- Creation of open community around Clang & source-to-source transformation tools
- We are hiring on this subject... join the scrum!

/~ ¿ SoSlang $%&^\$#@^!~*@ ?
Excuse my French!