LLVM @ RaincodeLabs

Johan Fabry

Senior Software Engineer

johan@raincode.com - @johanfabry

raincode LABS

compiler experts

<Prologue>

raincode LABS
compiler experts

Dragons in the room

raincode LABS

compiler experts

What we do

- Raincode Labs provides bespoke compiler services
- The first independent compiler expertise company in the world
- Consultancy services
- Technical implementation of them
- Full scope: consulting + delivery

Expertise

- Grammar, Languages, Formal Logic and COMPILERS
- .NET
- JVM
- LLVM
- DSLs
- GCC toolchain
- Visual Studio Plugins
- Micro-controllers

raincode LABS
compiler experts

Some very smart (re)engineering

Remove Technical Dependencies

- Datakom & Ideal COBOL
- PACBASE
- EGL
- APPBUILDER
- CA Gen/CoolGen

Bespoke Compilers

- PL/I LLVM for LzLABS
- COBOL-IT
- SAGE

Language migration

- Jbasic

raincode LABS
compiler experts

Clients

We are under NDA, sorry.

Raincode Labs & Academia

- We cherish academic partnership (McGill,ULB,VUB,Koblenz,UvA, …)
- We sponsor international research events
  - Software Language Engineering conference (SLE 2016)
  - The Compiler Construction conference (CC 2017, 2018)
  - Domain-Specific Modelling summer school (DSM-TP 2017)
  - SPLASH conference in 2017, 2018(?) (including SLE)
- We do tutorials and teach
  - Sponsored coding dojo at the <Programming> 2017 (VUB) 2018 (U. de Nice)
  - Classes on Software Construction, Evolution, … at UvA (2016-2018)

raincode LABS
compiler experts

raincode LABS
compiler experts

Compilers, Languages and Grammar

We dare to do
what others won't dare to think.

raincode LABS
compiler experts

</Prologue>

raincode LABS
compiler experts

Background

Raincode: Mainframe to .NET

- PL/I compiler, COBOL compiler, ASM 370 compiler
  - The three are used together
- Stability and backward compatibility is key!
  - External dependencies are of the devil
- We have our own compiler builder infra: YAFL
  - Only requirement: C compiler

raincode LABS
compiler experts

COBOL code example

```
*****************************************************************
 W-PAD SECTION.
*****************************************************************
     MOVE SPACES TO W-PAD-RETURN
     MOVE ZERO TO WRK-LENGTH
     MOVE SPACES TO WRK-FIELD
     MOVE ZERO TO WRK-TRAILING-SPACES
* actual length of W-PAD-VALUE -> WRK-LENGTH
     INSPECT FUNCTION REVERSE (W-PAD-VALUE)
         TALLYING WRK-TRAILING-SPACES FOR LEADING SPACE
     SUBTRACT WRK-TRAILING-SPACES FROM LENGTH OF W-PAD-VALUE
         GIVING WRK-LENGTH

     IF W-PAD-LENGTH <= WRK-LENGTH
         MOVE W-PAD-VALUE TO W-PAD-RETURN
     ELSE
 (...)
```

raincode LABS
compiler experts

PL/I code example

```
PROCESS_INPUT_FILE: PROC;
   RECORD = '';
   OPEN FILE (FILENAME);
   CALL X500_READ_FILE;
   IF IF = THEN                          /* OH YES THIS IS VALID  */
   THEN DO;
        ELSE = ELSE + 1;
        END;

   DO WHILE (SQLCODE = OK);
       CALL X100_MAKE_NEXT_RECORD;
       CALL X200_WRITE_RECORD_TO_DB(RECORD);
   END;
   (...)
```

raincode LABS
compiler experts

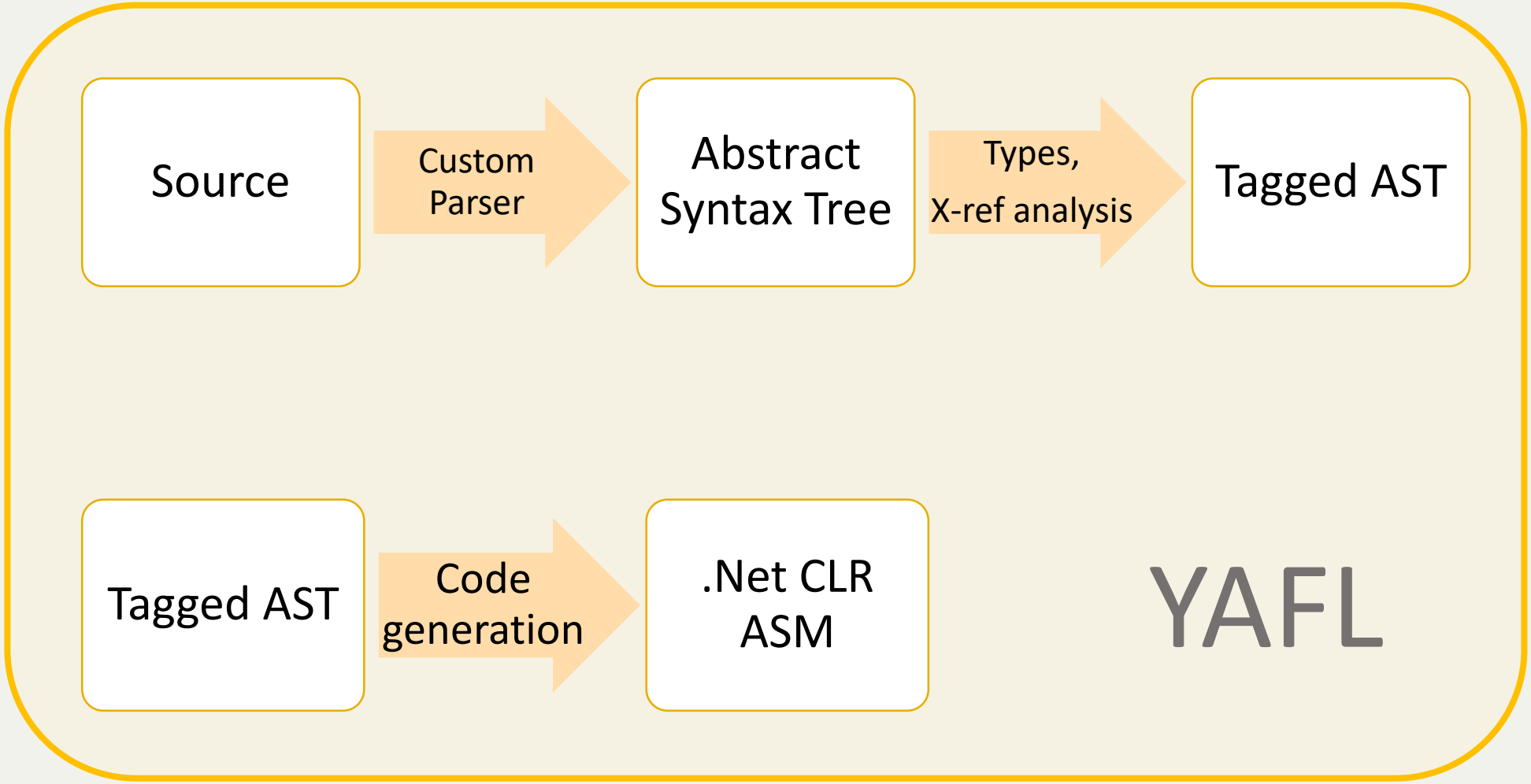
LLVM Work

raincode LABS
compiler experts

What do we have right now?

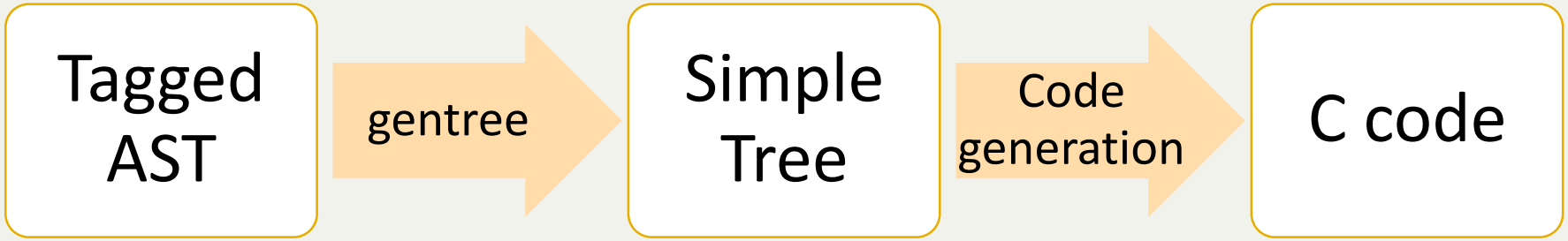
- PL/I compiler
  - 3.5 MY work in total
  - ± 75% coverage of the IBM specs (750 pages doc)
- COBOL compiler
  - 2 weeks work
  - We can do “Hello, World!”
- Quite a lot of shared infrastructure

Version 0 : PL/I .NET compiler



raincode LABS
compiler experts

Version 1: C generation (± 1MY)



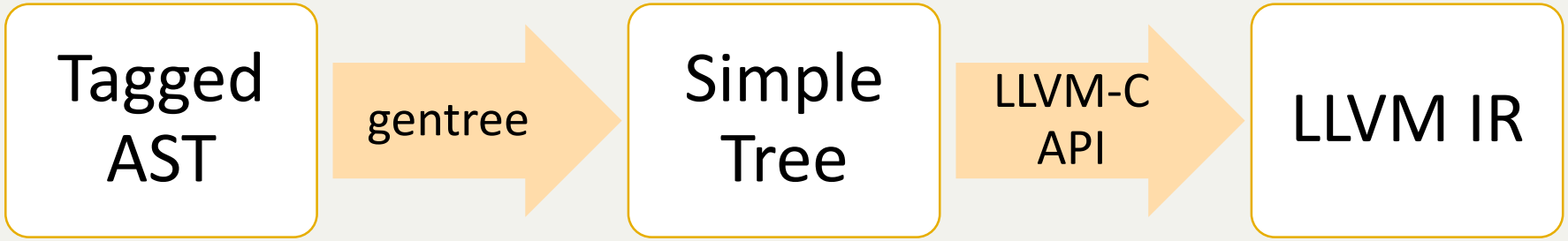
gentree

- Flatten control flow
- Var resolution (nested scopes)
- …

Issues

- No debug info
- Unclear semantics, e.g. names
- Slow executables
- Too complex

Version 2: LLVM IR (±2.5 MY)



- Client requested LLVM
- Thorough rewrite
- + lessons learned from V1

LLVM-C API

- Stability! Compatibility!
- C++ API impendence mismatch
- But C API is a second class citizen

LLVM-C API misses (LLVM 5)

- Debugging info generation: variable metadata
  - LLVM-C patches are under review for 6M+ (dead thread)
  - Yet C++ API has it
  - Go patches were first. They do not compile, so we adapted them
- Mainframe things missing
  - Packed decimal (yet DWARF standard: all PL/I & COBOL types)
  - Mainframe endianness, IBM floats
- BUT: Character encoding (EBCDIC) works!

Notable

- The tough part is mapping PL/I to LLVM IR
- We use plain vanilla features only
  - Stability! Compatibility! No dependencies!
- Upgrade LLVM V4 to V5: Only 3-4 days
  - Regenerate our YAFL to LLVM-C API bindings
  - Reapply debugging metadata patches

A fun story

Compilation time of a test program

- On Win: 30 seconds. On Linux: 12 hours.

Cause: basic block of ± 4.000.000 IR instructions (inlining!)

Origin: calculation of offset of instructions is in linear time

- But! Done for all instructions in the block at code generation time

Fix: limit number of instructions in the basic block

- But! C API does not provide a count (C++ does)
- Solution: generate IR for max 100 nodes in the gentree simple tree

Why difference Win vs Linux? Unknown (2 days work already)

LLVM Coolness

- It just works
- We like the IR: documented, clean, focused
- The ecosystem is broad and very active

LLVM Uncoolness

- LLVM-C API is badly documented
  - E.g. who is responsible to `free()` a string?
  - First approach: generate bunch of test programs through the API (3 weeks)
  - Now: look at the source code of the API implementation
- Assert fail in the backend: traceback to error in source code is hard
  - Essentially a YAFL issue: the mapping is not trivial
- LLVM itself is hard to understand and debug
  - As a client, we only look inside when we messed something up
  - In the end, complexity and difficulty is to be expected

Conclusions

- We are happy customers of LLVM
  - But use just plain vanilla, by design
- LLVM-C could be improved
- But we admit to not submit patches
  - Rare in any case
  - Process is too heavyweight
  - Difficult to justify investment (?)

Future Work

Future work for LzLABS

- PL/I compiler
- COBOL compiler
- Start on the ASM 370 compiler (?)

raincode LABS

compiler experts

RAINCODELABS HQ

Rue de la Caserne 45
B-1000 Brussels
Belgium

+32 2 522 06 63

RAINCODELABS USA

13245 Atlantic Boulevard.
Suite 4-263
Jacksonville, FL 32225
USA

+1 412.552.8207

info@raincodelabs.com

www.raincodelabs.com

raincode LABS
compiler experts