Gamma Ray

By Matthew Maycock, Weiyuan Li, Ben Caimano and Arthy Sundaram
Agenda

Gamma Language
Features

Ray Compiler
Architecture and design.
Implementation details.

Test-suite and Toolchain
Challenges and Lessons learnt
Demo
The Gamma Language

Elegant and Fully object oriented
  Primitive types are classes and variables are instances.
  IO wrappers encapsulated within objects.

Secure
  Private (protected) members are private (protected) to instances.
  Subclasses cannot override superclass behaviors.

Refinement
  Extend superclass behavior by refinement.
  Superclass provides hooks to refining types.
  Dynamic dispatch.

Anonymous class
  Create classes on the fly
  Akin to Lambda definitions and Java’s anonymous instantiations.

A language that has it all...!
class Account:

public:
    Integer bal
    Integer interest

Integer getBalance():

    if (refinable(bonus)) {
        bal := bal + refine bonus(interest) to Integer;
    }
    return bal

class NewAccount extends Account:

    refinement:

        Integer getBalance.bonus(Integer norm):
            return norm * rewards

    Public:
        Integer rewards
Anonymous classes

class Person:
    protected:
        String name
    public:
        init(String name):
            super()
            this.name := name
        void introduce():
            Printer p := system.out
            p.printString(name)
            p.printString(refine origin() to String)
            p.printInteger(refine age() to Integer)

main(System sys, String[] args):
    (new Person("Matthew") {
        String introduce.origin() { return "New Jersey"; }
        Integer introduce.age() { return 33; }
    }).introduce()
class IOTest:
public:
    init():
        super()

void interact():
    Printer p := system.out
    Integer i := promptInteger("Please enter an integer")
    p.printString("Integer converted to Float = ")
    p.printFloat(i.toF() )
    p.printString("\n")

    Integer promptInteger(String msg):
        prompt(msg)
        return system.in.scanInteger()

    main(System system, String[] args):
        IOTest test := new IOTest()
        test.interact()
From gamma to C

Gamma source

Scan and Parse

Abstract Syntax Tree

Semantic Analysis

SAST

Intermediate Representation

C-AST

Code Generation

Target C

Build Environment
Access & scope checks
Type checking
Tag types

Deanonymization
Refinement dispatch
Methods -> mangled functions

Preprocess the MACROs
Link the builtin functions.
Objects and dynamic dispatch in C
Tool chains and Test suites

Tools to inspect what is going on in the compiler: streams shows scanner results, canonize takes space delimited input and produces braced input, inspect/prettify takes input and shows the initial AST, classinfo shows metainfo about all classes (methods, variables, etc) including built ins.

We have automated testing from earlier in our development to make sure scanner / parser input remained consistent. We have additional testing facilities via a script to automatically compile and run any gamma source -- showing both the source and the output.
Challenges and Lessons

Translating an object-oriented program to a structural language using functional programming language!

Design choices - don’t do early optimization (arrays, null, this)

Feature subset

Prioritizing tasks

Scheduling weekly team meetings

Most of all: Don’t take too many other classes while taking PLT
DEMO

BANK SIMULATION AND N-QUEENS PUZZLE