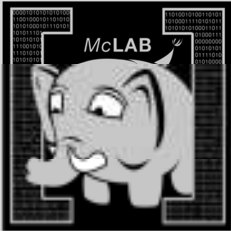


McLab Tutorial

www.sable.mcgill.ca/mclab



Part 6 – Introduction to the McLab Backends

- MATLAB-to-MATLAB
- MATLAB-to-Fortran90 (McFor)
- McVM with JIT

6/4/2011

McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed

Backends- 1

So far in the tutorial we have concentrated on the front-end and the analysis framework. Now we turn our attention to possible backends. We support three kinds of backends, first just producing MATLAB, second a more static compiler that translates MATLAB to Fortran90, and third a Virtual machine for executing MATLAB which contains a JIT compiler.

MATLAB-to-MATLAB

- We wish to support high-level transformations, as well as refactoring tools.
- Keep comments in the AST.
- Can produce .xml or .m files from McAST or McLAST.
- Design of McLAST such that it remains valid MATLAB, although simplified.

6/4/2011

McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed

Backends- 2

We think that one important use of our front-end and analysis framework is for supporting high-level transformations and refactoring tools. Keeping this in mind, our AST keeps the comments from the original input program, so that we can pretty-print the transformed source, as well as the comments. McLab can generate both .xml and .m files from McAST or McLAST. We use the .xml format as a way of conveying the AST to McVM. Since we wanted to be able to generate valid MATLAB from our ASTs, we have designed the ASTs to use only valid MATLAB constructs.

MATLAB-to-Fortran90

- MATLAB programmers often want to develop their prototype in MATLAB and then develop a FORTRAN implementation based on the prototype.
- 1st version of McFOR implemented by Jun Li as M.Sc. thesis.
 - handled a smallish subset of MATLAB
 - gave excellent performance for the benchmarks handled
 - provided good insights into the problems needed to be solved, and some good initial solutions.
- 2nd version of McFOR currently under development.
 - fairly large subset of MATLAB, more complete solutions
 - provide a set of analyses, transformations and IR simplifications that will likely be suitable for both the FORTRAN generator, as well as other HLL.
- e-mail hendren@cs.mcgill.ca to be put on the list of those interested in McFor.

6/4/2011

McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed

Backends- 3

McVM-McJIT

- Whereas the other back-ends are based on static analyses and ahead-of-time compilation, the dynamic nature of MATLAB makes it more suitable for a VM/JIT.
- MathWorks' implementation does have a JIT, although technical details are not known.
- McVM/McJIT is an open implementation aimed at supporting research into dynamic optimization techniques for MATLAB.

6/4/2011

McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed

Backends- 4

McVM is designed for flexibility and high performance.

McVM Design

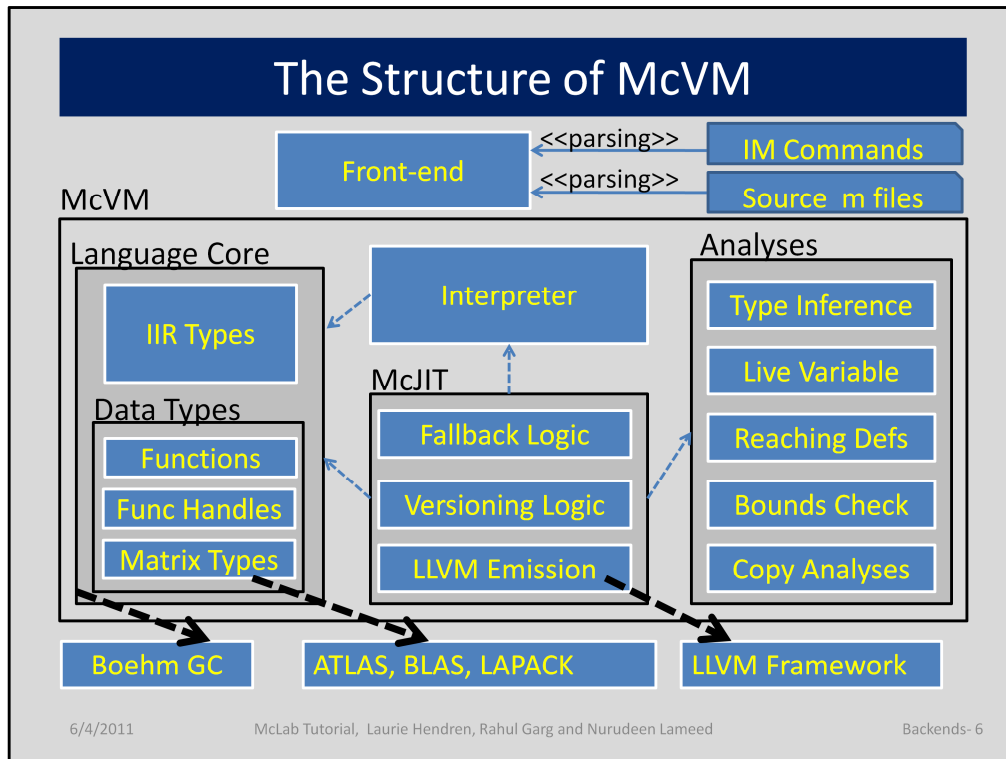
- A basic but fast interpreter for the MATLAB language
- A garbage-collected JIT Compiler as an extension to the interpreter
- Easy to add new data types and statements by modifying only the interpreter.
- Supported by the LLVM compiler framework and some numerical computing libraries.
- Written entirely in C++; interface with the McLab front-end via a network port.

6/4/2011

McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed

Backends- 5

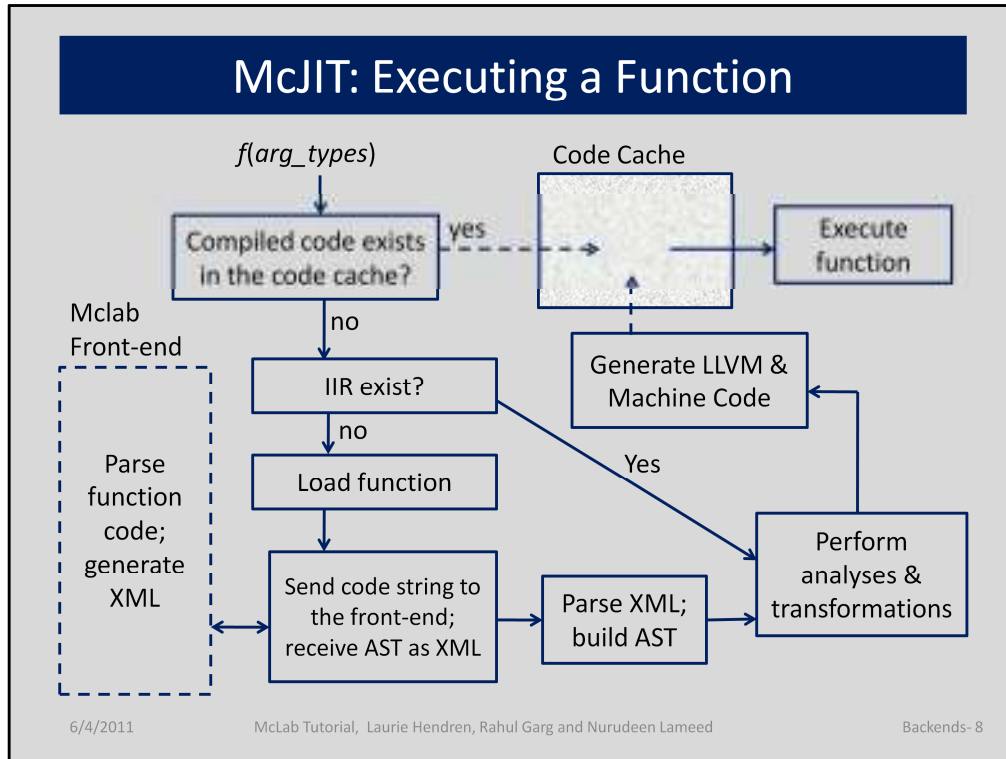
McVM is designed for flexibility and high performance.



The front-end is responsible for parsing matlab commands and m files. The language core is supported by Boehm GC, ensuring an automatic garbage collection of IIR nodes. The operations on Matrix-type data are supported by ATLAS, BLAS and LAPACK library. McJIT is supported for efficient code compilation by several analyses.

Supported Types	
Logical Arrays	
Character Arrays	
Double-precision floating points	
Double-precision complex number matrices	
Cell arrays	
Function Handles	
6/4/2011	McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed
Backends- 7	

These are the currently supported types in McVM.



How does the JIT compiler execute a function? Given a function called with some argument types; McVM checks if a compiled code exists that match the call, in terms of the types of the arguments and proceed as shown.

Type Inference

- It is a key performance driver for the JIT Compiler:
 - the type information provided are used by the JIT compiler for function specialization.

6/4/2011

McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed

Backends-9

A key analysis performed McVM is the type inference analysis ...

Type Inference

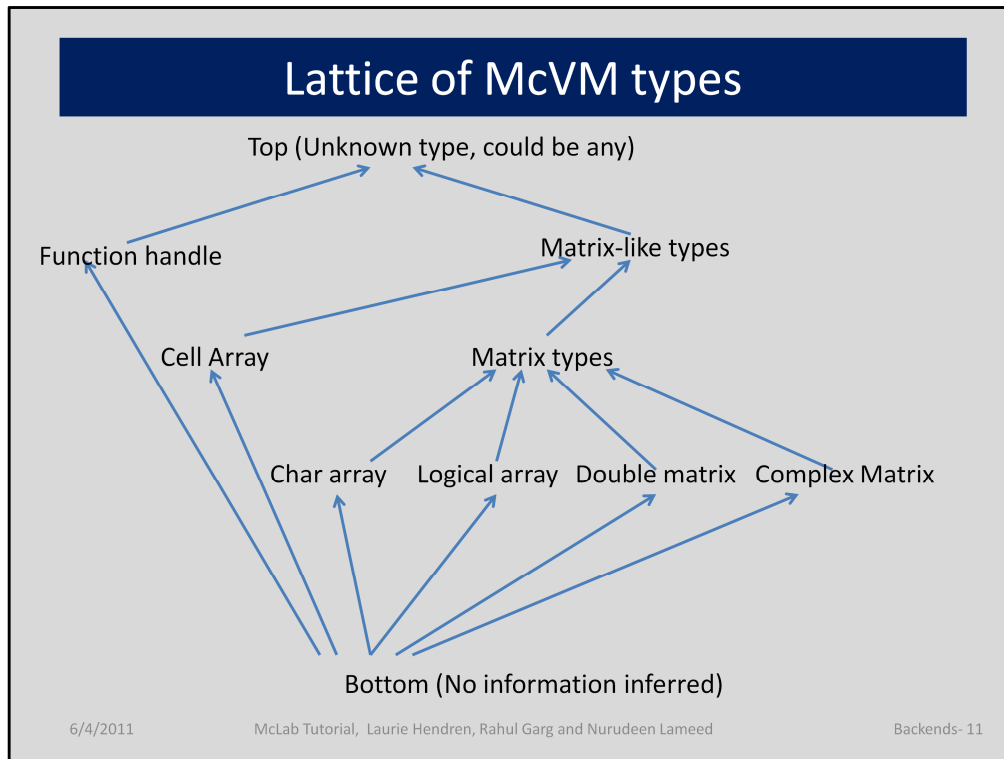
- It is a forward flow analysis: propagates the set of possible types through every possible branch of a function.
- Assumes that:
 - for each input argument *arg*, there exist some possible types
- At every program point *p*, infers the set of possible types for each variable
- May generate different results for the same function at different times depending on the types of the input arguments

6/4/2011

McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed

Backends- 10

Unlike other type-inference analysis that assumes that all program components can be loaded at once and performs a whole-program analysis, our type inference analysis is intra-procedural since dynamic loading suggests that not all program components may be loaded at once. Variables can also have different types at different points in a function.



A key analysis performed McVM is the type inference analysis ...

Internal Intermediate Representation

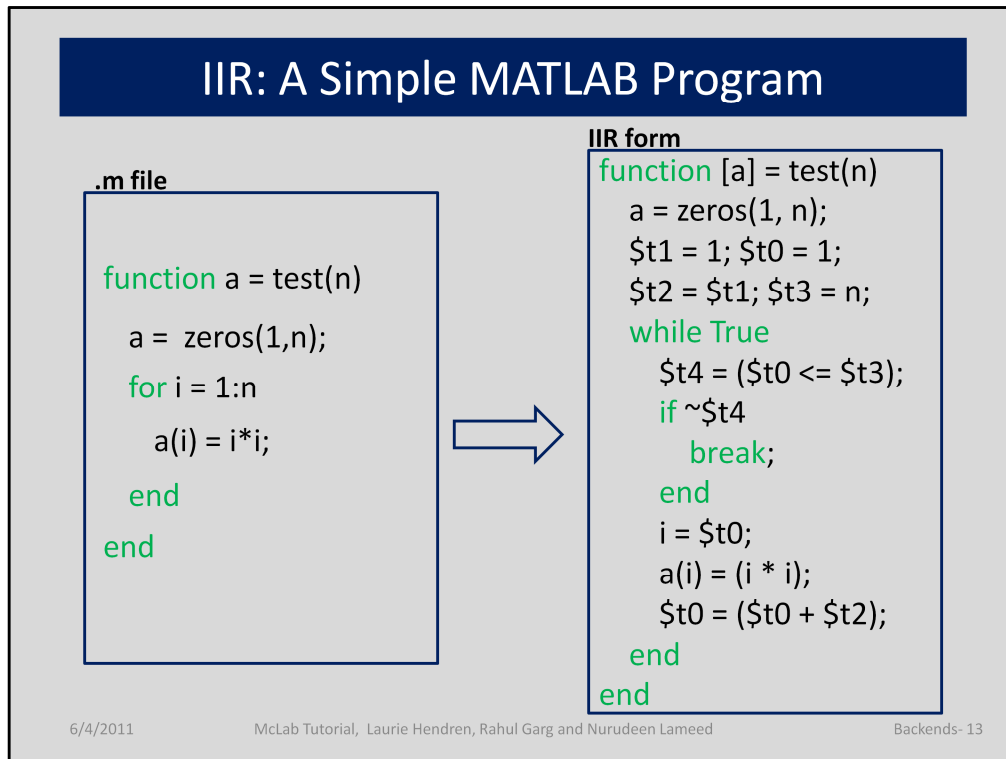
- A simplified form of the Abstract Syntax Tree (AST) of the original source program
- It is machine independent
- All IIR nodes are garbage collected

6/4/2011

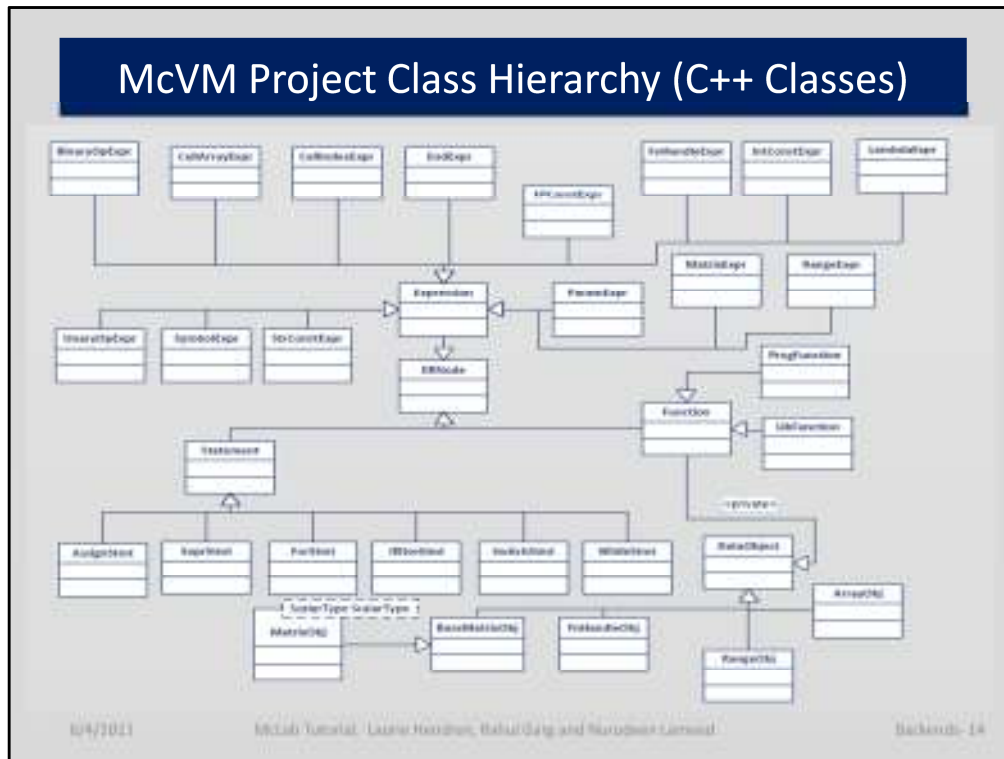
McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed

Backends- 12

McVM converts source code into a form more amenable to analyses. It is similar in form to a three address code.



The box on the left-hand side shows the code of an .m file. This is transformed into the box on the right : the corresponding function in internal IR form.



The figure shows a simplified UML class diagram for the McVM project. At the root of class hierarchy is the IIRNode. There are a number of statements and expressions as well.

```

Running McVM

$ cd ~/mcvm2.8/mclab/mcvm-llvm2.8/debug> ./mcvm -jit_enable true -start_dir ~/pldill_mclabtutorial/

McVM - The McLab Virtual Machine v1.0
Visit http://www.sabio.mcgill.ca for more information,
=====

>: c = test(10);
Compiling function: "test"
>: c
ans =
matrix of size 1x10
  1    4    9   16   25   36   49   64   81  100
>:

```

6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed Backends- 15

Here, I show how to start and execute functions McVM/McJIT.