Jiangfeng Wang, David Rincon-Cruz, Wode ‘Nimo” Ni, Chi Zhang
Overview
What is MPL?

**LLVM**
- MPL compiles to LLVM IR
- LLVM is flexible and works across multiple platforms

**Motivation**
- C/Java/Matlab - like Syntax
- Programmable Matrix Operations
- Lightweight and intuitive without math background

**Matrices**
- Matrix Arithmetic
- Apply Function

**Images**
- Reading in images
- Manipulating Pixels
- Writing images
Language Syntax
Programming in MPL

Comments

/* This is a comment*/

Primitives

int, float, bool, void, string, Mat

Control Flow

if, else, while, return

Arithmetic Operator

+ - * / = ++ --

Conditional Operator

== != > < >= <=

Logical Operator

!, &&, ||

Matrix

\[1,2;3,4\] \[1.5,2.5;3.5,4.5\]

Entry function

int entryf() {
    return 1;
}

   _____
Sample MPL program

Calculating GCD

```c
int gcd()
{
    if (#C > #W) {
        return #C - #W;
    } else {
        return #C;
    }
}

int main() {
    int h;
    Mat<int> [1][2] m;
    m = [50, 40];
    while (m[0][0] != m[0][1]) {
        gcd @ m;
    }
    h = m[0][0];
    print(h);
}
```

```c
int entry() {
    int sum;
    sum = #NW + #N + #NE + #W + #S + #E + #SW + #SE;
    sum = #C * 8 - sum;
    if (sum < 0)
        sum = 0;
    return sum;
}

int main() {
    Mat<int> [512][512] img;
    pgmread("lena.pgm", img);
    entry @ img;
    pgmwrite("lena-out.pgm", img);
}
```
Architecture
Architecture

- Added SAST for matrix dimensional information inferred by Semant
- C functions for image and console IO
- Not too different from MicroC
- Generating code for the Apply operator
Generating Code for Entry functions

- `<function> @ <Mat>`
- Generate while loops over the target matrix
- neighbors passed in by value
- Moore neighborhood
- Edge problem: a torus!

<table>
<thead>
<tr>
<th>NW</th>
<th>N</th>
<th>NE</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>C</td>
<td>E</td>
</tr>
<tr>
<td>SW</td>
<td>S</td>
<td>SE</td>
</tr>
</tbody>
</table>
Testing

Blaming the User
Pocket Reference

You're a 10x hacker and it must be someone else's fault.

O RLY?
@ThePracticalDev
Testing

➔ Scanner test and Program test
➔ MicroC’s style of test is efficient.
For our language, printm() is the most useful function for testing.

Example: @ Apply test

```c
int entryf() {
    return 1;
}

int main() {
    Mat<int>[3][3] m;
    int p;
    m = [1,2,3;4,5,6;7,8,9];
    entryf @ m;
    printm(m);
    return 0;
}

int entry(){
    return 1;
}

int main(){
    int k;
    k = 0;
    entry @ k;
    return 0;
}
```

Fatal error: exception Failure("k must be a matrix type")
Project Management
Project Timeline

2/8
Project Proposal

3/5
Scanner, parser, ast

3/28-5/8
Semantic checker, Codegen

2/22
LRM

3/28
Hello World
Project Management

- 3-4 weekly meetings
- TA advising meetings
- Dividing tasks and pair programming
- Multiple branches
Contribution

Jiangfeng and David: Design, scanner, parser, ast, semantic checker, sast

Nimo and Chi: Skeleton of Scanner and Parser, Codegen, example programs, test suite, game of life
Lessons Learned

Getting it to Work
And Having No Idea How
Lessons Learned

Jiangfeng: Start early. Micro C and previous projects are extremely helpful as sources of instruction.

David: It’s better to argue out the features of the language so that everyone is on board. Pair programming keeps everyone on board and provides sanity checks.

Chi: Understanding of code is important. Especially when you try to learn from previous project.

Nimo: Frequency of the meetings is important. Incremental development is always better than merging big chunks of code.
Conway’s Game of Life

➔ Any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
➔ Any live cell with two or three live neighbours lives on to the next generation.
➔ Any live cell with more than three live neighbours dies, as if by overpopulation.
➔ Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
➔ There are known patterns

CONWAY'S GAME OF DEATH

Each cell has three possible states: human, dead, or zombie.
At each step in time, the following transitions occur:
1. Any human cell dies.
2. Any dead cell becomes a zombie cell.
3. Any zombie cell converts to a live human iff it has exactly two human neighbour cells.

spikedmath.com
© 2010

"Based on "Brian's Brain" attributed to Brian Silverman."
Demo

→ Image Convolution
→ Game of Life Simulation