Meeting Notes and Assignment 6/13/19

Paul Scanlon and Matt Murdoch

Colorado State University, Research for Dr. Sanjay Rajopadhye, Summer 2019

Matt and Paul alternated answering the examples from chapter 5, assigned last week, first.

We are confuces on the details of the math for the Bits in a Cache example, but this will be put on hold. Specifically, where does the additional -2 come from in what looks to be the tag field.

Email Waruna about eclipse AlphaZ problems.

We will continue to use this language, AlphaZ, for the research project.

AlphaZ cannot generate an eddy diagram - the logic goes the other way. We look at an Eddy diagram to generate code.

$Affine$ is a linear transformation plus a constant. $y = ax + b$ where y, b, and x are vectors. In $3d$ coordinates, or just $2d$ for the following example, a point i, j is mapped to $–i, j$. This is a linear transformation that reflects about jx. This concept applies to image transformations. The transformation itself can be represented by constant matrices.

At this point in the meeting, Sanjay gave an example of matrix transformation on the white board shown in *Figure* 1. The math going on here applies to image transformation by applying the matrix to all pixels in the image. The b allows for shift in any number of axes. The class of mathematical transformations that does all sorts of transformations like this is called $Affine$. In this class, the program is the equation or dependence function. Also, the question of i, j placement is the $Affineequation$ for defining program dependence. The propose of AlphaZ is to apply linear algebra to transform programs.

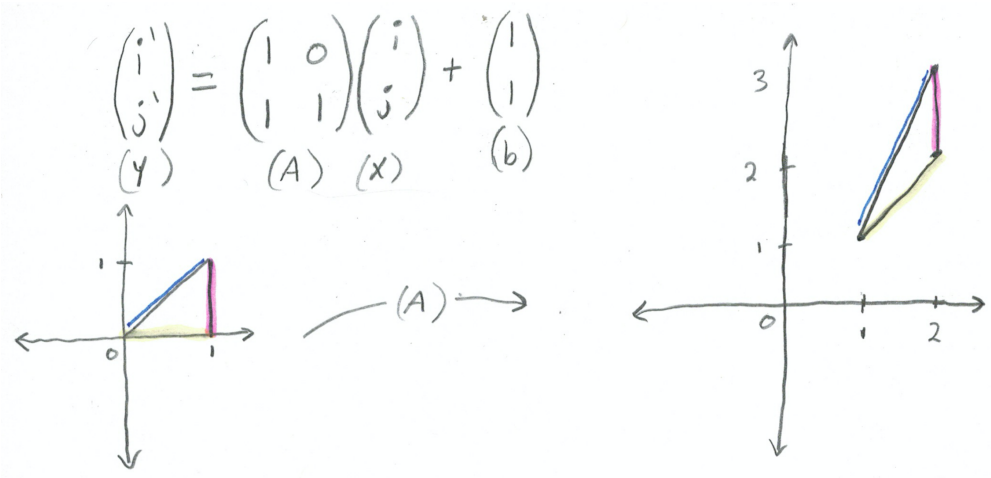

Figure 1: Graphical Example of Matrix Transformation. This is related to the recent matrix chain problem.

The first line of syntax in the LUD tutorial is:

$$affine\ LUD\ \{N|N > 0\} \tag{1}$$

The correct interpretation of this line is: "This program takes parameter N and requires $N > 0$."

The driving force of Eddy diagrams and AlphaZ is answering what the general idea of what an algorithm should compute is.

At this point in the meeting, the "Mountain in the Sun" AlphaZ assignment was given. Given an angel, θ, at which the sun hits a jagged mountain, and an array of elevations, Y, at discrete points along the surface on that mountain, considering two dimensions with respect to the angle of the sun, which points are in the sun, and which are shaded by another? An illustration is given in *Figure* 2. The assignment is to implement this in AlphaZ.

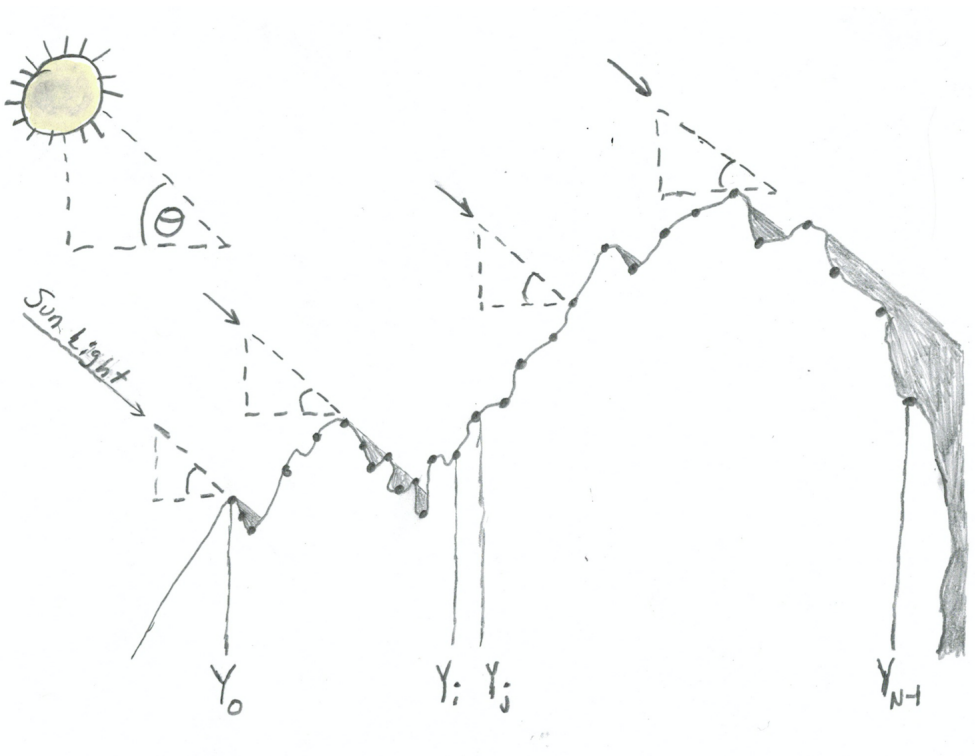

Figure 2: Concept of the Mountain in the Sun Assignment

There are N heights sub-scripted $0-(N-1)$. An array of booleans, S, corresponding to each height tells which points are in the sun and which are in the share after some calculation. Thus, the basic structure for the boolean array population is:

$$S[i] = \begin{cases} True\ if\ the\ point\ is\ in\ the\ sun \\ False\ if\ the\ point\ is\ not\ in\ the\ sun \end{cases}$$

From the illustration, we can see that a point depends on points to its left. Specifically, if the height of a point to its left is higher than the line of sight to the sun.

Therefore, populating the true values of the boolean output array has the form:

$$S[i] = \forall_{0 \le j < i} \ \frac{Y[j] - Y[i]}{(i-j)h} < tan(\theta) \tag{2}$$

A goal for this algorithm is to not rely on fixed size, N, and also be efficient. When implementing, think "and" and "or" to deduce the inclusion of "for all" in the algorithm. The *reduce* syntax will be required, but not the use of projections. In program development, we will not be able to use $make\ verify$ without constructing a verification program. Thus, we will debug manually by constructing our own corner cases and with *make check*

A self dependence run time error may be encountered as in last weeks assignment, the LUD tutorial. When self dependence occurs, it is likely due to defining a variable in terms of itself, such as $X[i, j]$ being a function of $X[i, j]$. Find this bug, try to fix it, and report the results of the process to Sanjay via email as soon as possible. Do this for both the LUD tutorial and this new assignment.

Recall that declaration of a triangular matrix such as in the matrix chain problem in arbitrary expression form is:

$$MCP = \{i, j \mid 1 \le i \le j \le N\} \tag{3}$$

The new meeting time to accommodate class schedules is 2:00 PM on Wednesdays.