The AlphaZ Automation Tool

Announcements

- Project proposal is due THIS Friday
- HW5 is due Wednesday February 29th

Today

- AlphaZ overview
- Some Alphabets syntax with examples
- Some AlphaZ scripting syntax
- Equations to Loops

Recall the Protein String Matching Example (smithWaterman.c)

```
for (i=1;i<=a[0];i++) {
  for (j=1;j<=b[0];j++) {
    diag    = h[i-1][j-1] + sim[a[i]][b[j]];
    down    = h[i-1][j] + DELTA;
    right   = h[i][j-1] + DELTA;
    max=MAX3(diag,down,right);
    if (max <= 0)  {
      h[i][j]=0;
    } else if (max == diag) {
      h[i][j]=diag;
    } else if (max == down) {
      h[i][j]=down;
    } else  {
      h[i][j]=right; xTraceback[i][j]=i; yTraceback[i][j]=j-1;
    }
    if (max > Max){
       Max=max; xMax=i; yMax=j;
    }
 }}  // end for loops
```

System of Recurrence Equations

Recurrence Equation

- An equation that describes the value of a function when applied to a parameter as the same function applied to smaller instances of that same parameter.
- Used to specify recursive computation.

System of Recurrence Equations

- A set of equations where each equation defines a different function.
- The functions can be mutually recursive.

Recall the Protein String Matching Example (smithWaterman.c)

```
for (i=1;i<=a[0];i++) {
  for (j=1;j<=b[0];j++) {
    diag    = h[i-1][j-1] + sim[a[i]][b[j]];
    down    = h[i-1][j] + DELTA;
    right   = h[i][j-1] + DELTA;
    max=MAX3(diag,down,right);
    if (max <= 0)  {
      h[i][j]=0;
    } else if (max == diag) {
      h[i][j]=diag;
    } else if (max == down) {
      h[i][j]=down;
    } else  {
      h[i][j]=right;
    }
    if (max > Max){
       Max=max; xMax=i; yMax=j;
    }
 }}  // end for loops
```

$$H_{i,j} = max \begin{cases} 0 \\ H_{i-1,j-1} + sim_{a_i,b_j} \\ H_{i-1,j} + \delta \\ H_{i,j-1} + \delta \end{cases}$$

AlphaZ

Recall the automation process

- Translate the code to a model
- Select a transformation/schedule
  - Determination of partial order on E, data dependence analysis
  - Ensure that the loop transformation/schedule is legal
- Transform the model and generate the transformed code

AlphaZ

- Uses an equation language Alphabets to specify the computation directly into the model.
- User can write scripts that have AlphaZ automate schedule selection or let the user specify a schedule and a storage mapping.
- Code generators use the schedule and storage mapping to generate code.

Some Alphabets Syntax

Overall structure

```
affine systemname <input parameter set constraints>
  given <input var domain list>;
  returns <output var domain>;
  using <temp var domain list>;
  through
    <system of affine recurrence equations>
.
```

Input parameter set constraints

```
{P, Q, R|P>1 && Q>1 && R>1}
```

Variable domain examples

```
float A {i,k | 0<=i<P && 0<=k<Q};
float C {i,j,k| 0<=i<P && 0<=j<R && k==Q+1};
```

Recurrence equation

```
A[i,k] = C[i,k];
```

Some Alphabets Examples

Examples can be found on resources web page

- under CodeExamples
- AlphaZstart.tar

Examples

- smith_waterman.ab (in AlphaZstart.tar)
- MM.ab (in Eclipse example files, in AlphaZstart.tar as matrix_product.ab)

Key Concepts AlphaZ scripting

Space-time mapping concept

- Space-time mapping is same as the schedule mapping.
- The space-time adjective refers to some of the elements in the map indicating which processor (space) and some indicating time.
- In AlphaZ scripting there is also the concept of an element indicating statement order.

Storage Mapping

- Recurrence equations are analogous to array assignments where all of the arrays are fully expanded.
- This leads to single assignment, or each location is only written once.
- Efficient implementations share memory locations between writes.
- It is important to specify the storage mapping in AlphaZ.

Some AlphaZ Examples

Examples can be found on resources web page

- under CodeExamples
- AlphaZstart.tar

Examples

- basicTest.cs (in AlphaZstart.tar)
- wavefrontScheduleC.cs (in AlphaZstart.tar)

Equations to Loops

Now we HAVE to specify a schedule!

Recommended procedure

- Brainstorm straightforward sequential schedules.
- Verify the schedule with the tool.
- Try out some storage mappings that reduce memory usage.
- Consider how those storage mappings require the schedule to change.
- Verify the resulting schedule and storage mapping with AlphaZ.

Onto Parallelism!

Parallelizing code using AlphaZ

- The exact flow dependences can be read from the system of recurrence equations.
  - <Create a dependence relation for an example>
  - <Create a dependence vector from the dependence relation>
- Apply the schedule mapping to the dependence vectors to determine the new dependence vectors.
- Any dimension in the new dependence vectors that do not carry a dependence can be made parallel.
- Currently have to insert the OpenMP pragma in the generated code, but the schedule verifier should be able to verify if parallelism is possible.

Concepts

Systems of recurrence equations

- Some AlphaZ syntax for specifying them
- How they specify a computation without implementation details
- Why an equation specification simplifies dependence analysis

How to use AlphaZ scripts

- HAVE to specify a schedule in AlphaZ (later will show automation of this step)
- How to determine if parallelism is possible

Scheduling in AlphaZ

- Space-time mapping as a schedule
- Dimensions in that mapping being of type serial, parallel, or statement order

Next Time

Reading

- AlphaZ wiki, AlphaZ LUD tutorial, and Alphabets grammar

Homework

- Project proposal due Friday 2/24/12
- HW5 is due Wednesday 2/29/12

Lecture

- Tools for specifying and transforming polyhedra: AlphaZ continued