Dimensional Analysis for Multidimensional Dataflow Programming

Bill Wadge & Monem Shennat
wwadge@uvic.ca  ashennat@uvic.ca
Introduction
Multidimensional Data
Dimensional Analysis Example
Main Objective
Data Flow Operators
Dimensional Analysis Algorithms
Conclusion
Significance
INTRODUCTION

Dataflow Programming

Is programming in functional language in which every data object is a stream. Lucid is our prime example.

Statements in Lucid

The statement of Lucid program are equations specifying the transformations to be applied to streams, e.g. $S = \text{rsum}(X)$. 
Multidimensional Data

There are numerous Multidimensional data base (MDB) systems as well as OLAP systems, which are also based on a multidimensional model.

GLU and Dimensionality

GLU was a dialect of Lucid that allowed multiple dimensions. It’s DA was primitive.
GENERAL EXAMPLE OF DIMENSIONAL ANALYSIS

The measurements are represented in one to four possible values start with the first dataset.

X-Ray Dimensions

6 Dimensions
- Detector dimension & Field of View
- Bit depth
- Time
- Patient
- Date
MAIN OBJECTIVE

Our goal is to create an algorithm for Dimensional Analysis of a multidimensional dialect of the data flow language Lucid that allows us to calculate the dimensionality of program variables, and to exclude irrelevant dimensions.
SIMPLE DATA FLOW NETWORK
LET’S REVIEW SOME CONCEPTS

First & Next

**First** send on the first item in its input stream and surpasses the rest.

**Next** skips the first and passes on the rest

FBY

The output of this operator is the first component of its first input followed by its entire second input

ASA

The value of the first argument has when the second argument first becomes true
DIMENSIONAL ANALYSIS
ALGORITHMS

Time Dimension \{t\}

Space and Time Dimensions \{s, t\}

Multiple Dimensions \{s, t, h, v, z, \ldots\}
ALGORITHM FOR ONE DIMENSION “TIME”

\[
\begin{align*}
a & = b + c \\
m & = i + 1 \\
d & = 3 \\
i & = 1 \text{ fby } m \\
b & = \text{first } j \\
j & = 1 \text{ fby } k \\
c & = \text{next } i
\end{align*}
\]
d = 3

b = first j

a = b + c

c = next i

m = i + 1

j = 1 fby k

i = 1 fby m
ALGORITHM FOR TWO DIMENSIONS “TIME & SPACE”

The algorithm for time and space is slightly more complex. Instead of accumulating a single set, we accumulate a table that assigns to each variable a subset of \{s, t\}. Subset is the estimated dimensionality of the variable, i.e. the set of dimensions relevant to the value of the variable.
CONTINUED...

P = 8
Q = X sby T
T = P fby Q
V = T + Q
X = first Q
ALGORITHM FOR MULTIPLE DIMENSIONS

Unlimited sets can be accumulated in a table which assigns to each variable a set of dimensions based on how many times this variable will be re-evaluated.
Streams “set of atomic equations”

Dimensionality Re-evaluation

First approximation of the re-evaluate dimensions

Does the equation have more approximations?

2 to N number of dimensionalities re-evaluations

Stop
<table>
<thead>
<tr>
<th>Set of ‘atomic’ Eqs</th>
<th>Dimensionalities</th>
<th>First Appro</th>
<th>Second</th>
<th>Third</th>
<th>Forth</th>
<th>Fifth</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Y = H ) fby.c ( V )</td>
<td>( Y )'s dimensionality</td>
<td>{ }</td>
<td>{c}</td>
<td>{c, m}</td>
<td>{c, m, k}</td>
<td></td>
</tr>
<tr>
<td>( X = ) next.c ( Y )</td>
<td>( X )'s dimensionality</td>
<td>{ }</td>
<td>{ }</td>
<td>{c}</td>
<td>{c, m}</td>
<td>{c, m, k}</td>
</tr>
<tr>
<td>( H = X ) fby.m ( Y )</td>
<td>( V )'s dimensionality</td>
<td>{ }</td>
<td>{m}</td>
<td>{c, m}</td>
<td>{c, m}</td>
<td>{c, m, k}</td>
</tr>
<tr>
<td>( W = X ) asa.c ( Y )</td>
<td>( W )'s dimensionality</td>
<td>{ }</td>
<td>{ }</td>
<td>{ }</td>
<td>{m}</td>
<td>{m, k}</td>
</tr>
<tr>
<td>( R = Y ) fby.k ( X )</td>
<td>( R )'s dimensionality</td>
<td>{ }</td>
<td>{k}</td>
<td>{k, c}</td>
<td>{k, c, m}</td>
<td></td>
</tr>
</tbody>
</table>

Continued...
CONCLUSION

We have algorithms to estimate the dimensionality of programs with one, two, and multiple dimensions. What remains is to analyze programs with user-defined transformations, e.g. 
\[ rsum(X) = S \text{ where } S = \text{first } X \text{ fby } S + \text{next } X \]
SIGNIFICANCE

Our DA algorithm will make possible the efficient implementation of multidimensional Lucid. Also, tools or platforms such as TensorFlow needs to know which dimensions are relevant, and currently this is the user’s responsibility.
THANK YOU!