CPSC 474 Project 1: Data Dependencies
Distributed computing - Bernstein's conditions for Data Dependencies

Prof. Doina Bein, CSU Fullerton

dbein@fullerton.edu

40 points

Introduction

In this project you will have to analyze the data dependencies between simple instructions. You will consider only simple arithmetic instructions that involve variables whose names are single letter of the English alphabet, the arithmetic operators (=, -, /, *), open and close parentheses, and the assignment operator. For example: "a = b + (c – d / e)". There will be a space between the variables, the operators and the open and close parentheses. Each simple instruction must have at least three distinct variables. You will design two separate algorithms, describe the algorithms using clear pseudocode and implement your algorithms in C/C++/Java, compile, test it, and submit BOTH the report (as a PDF file) and the programs. We do not care about the efficiency of the algorithms **but each run of a program must terminate within 60 minutes if N=3 and M=4**.

Algorithm Calculate

Problem: Given a simple instruction and a block of n instructions (n < 11), identify which instructions in the block can be executed in parallel to the simple instruction.

Example 1
INPUT:
Instruction:
d = b + (c – d / e)

Block:
b = b * c
c = c – a
a = a + b * c

OUTPUT:
Then the output should be:
c = c – a

Algorithm Verify

Problem: You are given a number N < 11 instructions, identify all the pairs of instructions that can be executed in parallel.

Example 1

INPUT:
Block:
b = b * c
d = c – a
a = a + b * c

OUTPUT:
The pairs of instructions that can be executed in parallel are:

(b = b * c, d = c - a)

Example 2

Block
b = a * b * c
c = c – a
a = a + b * c

OUTPUT
The pairs of instructions that can be executed in parallel are:
NONE

Explanation (not part of the output): None of them can be executed in parallel because:

- Instructions I1 and I2 are flow dependent because variable a is generated by I1 as output and used by I2 as input.
- Instructions I2 and I3 are anti-dependent because variable a is generated by I3 but used by I2 and in sequence I2 comes first.
- I3 is flow dependent on I2 because of variable c.
- Instructions I3 and I1 are output dependent because variable a is generated by both instructions.

What to do

1. Write clear pseudocode for each algorithm and submit it as a PDF report.

2. Implement your algorithm in C/C++/Java.

3. Compile and execute the program using the examples provided.

4. Create a file with the output of the program for an input value and submit it together with the program. Note, the output can be redirected to a file (for easy printing). For example, the following command line

will create an output file in Linux-based operating system called `a1out.txt` by re-directing the output from the screen (display) to the file a1out.txt:

```
K:\cs474> a.out > a1out.txt
```

Grading rubric

The total grade is 40 points. Your grade will be comprised of three parts, Form, Function, and Report:

- Function refers to whether your code works properly (30 points), 10 points for Calculate and 20 points for Verify.
- Form refers to the design, organization, and presentation of your code. The instructor will read your code and evaluate these aspects of your submission (3 points):
    - README.md completed clearly (1 points)
    - Style (whitespace, variable names, comments, helper functions, etc.) (2 points)
- Report (7 points) divided as follows:
    - Summary of report document (2 points)
    - Pseudocode of the chosen algorithm (2 points)
    - Three screenshots: one for the group members and two snapshots of code executing for some two distinct values of N (1 point each, total 3 points)

Obtaining and Submitting Code

This document explains how to obtain and submit your work:
GitHub Education / Tuffix Instructions

Here is the invitation link for this project:

https://classroom.github.com/a/PtCWBHGr