# DEPENDENCE ANALYSIS

*A Book Series On*

# LOOP TRANSFORMATIONS FOR RESTRUCTURING COMPILERS

**Utpal Banerjee**

**Series Titles:**

*Loop Transformations for Restructuring Compilers: The Foundations*

*Loop Parallelization*

*Dependence Analysis*

# DEPENDENCE ANALYSIS

**Utpal Banerjee**
Intel Corporation

*A Book Series on*
**Loop Transformations for Restructuring Compilers**

**Kluwer Academic Publishers**
**Boston / Dordrecht / London**

**Distributors for North America:**
Kluwer Academic Publishers
101 Philip Drive
Assinippi Park
Norwell, Massachusetts 02061 USA

**Distributors for all other countries:**
Kluwer Academic Publishers Group
Distribution Centre
Post Office Box 322
3300 AH Dordrecht, THE NETHERLANDS

**Library of Congress Cataloging-in-Publication Data**

A C.I.P. Catalogue record for this book is available
from the Library of Congress.



*Printed on acid-free paper.*

Printed in the United States of America

*To my parents:*

*Late Santosh Kumar Banerjee*
*Santi Rani Banerjee*

# Contents

# List of Figures

# List of Tables

# List of Notations

In the following, $\mathbf{i} = (i_1, i_2, \ldots, i_m)$ and $\mathbf{j} = (j_1, j_2, \ldots, j_m)$ are two $m$-vectors (integer or real), and $1 \le \ell \le m$.

| | | |
|---|---|---|
| $\mathrm{sig}(i)$ | Sign of a number $i$ | 66 |
| $\mathbf{sig(i)}$ | $(\mathrm{sig}(i_1), \mathrm{sig}(i_2), \ldots, \mathrm{sig}(i_m))$ | 95 |
| lev(**i**) | If $\mathbf{i} = \mathbf{0}$, then $m + 1$; otherwise, $\ell$ where $i_1 = i_2 = \cdots = i_{\ell-1} = 0, i_\ell \neq 0$ | 66 |
| det(**P**) | Determinant of square matrix **P** | 83 |
| diag(**v**) | Diagonal matrix with main diagonal **v** | 83 |
| $\mathbf{P}^{(r)}$ | Column $r$ of matrix **P** | 84 |
| $\mathcal{I}_m$ | The $m \times m$ identity matrix | 85 |
| rank(**P**) | Rank of matrix **P** | 111 |
| (**i**;**j**) | The $2m$-vector obtained by concatenating the elements of **i** and **j** | 135 |
| $\binom{\mathbf{A}}{\mathbf{B}}$ | The matrix obtained by concatenating the rows of matrices **A** and **B** | 135 |
| (**A**;**B**) | The matrix obtained by concatenating the columns of matrices **A** and **B** | 205 |

# Preface

The book series *Loop Transformations for Restructuring Compilers* has been designed to provide a complete mathematical theory of transformations, that can be used to automatically change a sequential program containing FORTRAN-like **do** loops into an equivalent parallel form. Volume I of this series, subtitled *The Foundations*, provides the needed mathematical background, and introduces the concept of dependence and some of the major transformations. The model program used there is a perfect nest of loops with unit stride, whose body is a sequence of assignment statements. Volume II, *Loop Parallelization*, uses the same model and gives a detailed theory of iteration-level transformations. This is Volume III in the same series.

In this book, we extend the model to a program consisting of **do** loops and assignment statements, where the loops need not be sequentially nested and are allowed to have arbitrary strides. In the context of such a program, we study, in detail, dependence between statements of the program, caused by program variables that are elements of arrays. In the future, we will study transformations in this program model, and then extend our theory to a model that is generalized further to include conditional statements.

This book may be considered to be the second edition of the author's 1988 book *Dependence Analysis for Supercomputing.* It is, however, a completely new work that subsumes the material of the 1988 publication.

The books in this series are directed towards graduate and advanced undergraduate students, and professional writers of restructuring compilers. The recommended background for Volume I is some knowledge of programming languages, and familiarity with calculus and graph theory. We need a similar background for Volume III,

and it is understood that the reader has read Volume I. We have adopted a cyclical approach for coverage of material in this series, and many concepts that were first introduced in Volume I are again discussed here in more generality. However, we do need several results and algorithms that have already been covered in great detail in Volume I, and there is no need to discuss them again. This book is basically independent of Volume II.

Volume III is more mathematical in nature than the previous two volumes. That is to be expected since dependence analysis of array variables (with linear subscripts) is a special type of linear optimization problem. Readers who have studied linear programming will have an easier time with this book than those who have not. However, a knowledge of linear programming is not a prerequisite. We have tried to explain the required concepts and results from that discipline, as needed in this book. We have taken great pains to show the evolution of a concept or result as we pass from a single loop to a double loop to a perfect nest to a general program containing arbitrarily nested loops. It is our hope that the amount of redundancy introduced here is just right to make the material easily accessible to the beginning reader, while avoiding the risk of causing boredom to the experienced one.

As in Volume II, we refer to algorithms, theorems, etc. in Volume I (i.e., in [Bane 93]) by prefixing an "I" in front of their number. For example, Algorithm I.2.1 is Algorithm 2.1 in Volume I. Also, an exercise in the current book may be referenced in two different ways depending on the location of that reference. For example, Exercise 3 is the third exercise at the end of the current section, while Exercise 2.1.3 is the third exercise at the end of Section 2.1. As in the previous two volumes, a comment like (explain) or a question like (why?), enclosed in parentheses, is meant for the reader and not a comment for the author himself left over from a draft version of the manuscript.

In Chapter 1, the nature and the structure of the book are explained. Due to space and time constraints, we could not cover several important topics in this volume (e.g., pointer analysis, symbolic analysis, run-time dependence analysis, and nonlinear problems); we expect to return to them sometime in the future.

U.B.

*Santa Clara, California*

# Acknowledgments

The present structure of this book owes much to Constantine D. Polychronopoulos who read an early version of the manuscript and suggested major changes. His comments have led to a book that is much more readable than it otherwise would have been. He also provided a number of motivated reviewers from the ranks of his present and former students. The author would like to thank Constantine for his extremely valuable help.

Several versions of the manuscript were read by the following reviewers: Mohammad Haghighat, Hideki Saito, Nicholas Stavrakos, David Craig, George Dimitriou, Ramesh Subramonian, Milind Girkar, David Sehr, and Pohua Chang. They found many errors and suggested a large number of changes. The author would like to thank them all for their devoted work that did a lot to improve the quality of the manuscript.

Special thanks are due to Mohammad Haghighat who took a personal interest in this project. He was always available for consultation. He let the author use his unpublished paper for the preparation of Section 7.4. He also acted as the local LaTeX expert providing valuable advice on complicated typesetting problems.

The author is grateful to his colleague Kent Fielden for his help in setting up the system on which the book was written. Finally, the author wishes to thank Dr. Richard Wirt and the Intel management for giving him the opportunity to write this series of books.

# Chapter 1

# Introduction

Dependence analysis is the most important part of any program restructuring process. When a compiler tries to restructure a given program to satisfy a certain goal (e.g., parallelization or improved cache performance) without changing the meaning of the program, it must honor certain *dependence constraints*. These constraints are determined by the order in which each program variable (see the first footnote on page 22) is defined and used during the course of execution of the program. When we rearrange a program, the order of memory references can be changed only to the extent that wrong values are not used or stored when the rearranged program is executed. The main dependence problem is as follows: given two program variables, decide if they have instances that reference the same memory location during execution of the program. We may also want to know the exact set of such instance pairs and/or certain characteristic properties of that set. The aim of dependence analysis is to gather useful information about the underlying dependence structure of a program and present it in a suitable form. This analysis can be performed at various levels: we may study dependences between program statements, iterations of a loop nest, subroutines in the program, etc.

It may be very difficult or even impossible for a compiler to solve a particular dependence problem, if all the necessary information is not available at compile-time. This could happen, for example, when there are conditional statements in the program, or certain variables are to be assigned values at run-time, or the value of a loop bound can-

not be determined at compile-time. Also, variables of certain kinds may be difficult to deal with; pointer variables fall in this category.

In this book, we take as our model a sequential program consisting of **do** loops with arbitrary strides and assignment statements, and study dependence between those statements. We will only consider program variables that are elements of arrays, and assume that loop limits and array subscripts are affine functions of index variables of loops. (An *affine* function is a linear function plus a constant.) Dependence problems involving array elements are important for numerical programs, and they have been widely studied for twenty years.

Non-unit strides lead to *holes* in the index space of a loop nest. In the basic model, the index variable of a loop moved from one integer to the next larger integer as the loop iterations unfolded. That is no longer the case when the loop has a non-unit stride. To remedy the situation, we use an *iteration variable* for such a loop, that enumerates the iterations $0, 1, 2, \ldots$. The iteration variable of a loop with a non-unit stride behaves like the index variable of a loop with unit stride. When non-unit strides are present, dependence analysis and loop transformations are easier to handle in terms of iteration variables rather than in terms of index variables.

The passage from a perfect to an imperfect loop nest poses a more serious challenge. We handle that situation by utilizing two important points. First, an imperfect nest can be thought of as a perfect nest whose body is a sequence of other loops and assignment statements. Second, even in an imperfect loop nest, the loops containing a given statement are still sequentially nested. Thus, the sequence of loops determined by a statement behaves like a perfect nest.

If $a$ and $b$ denote two integers such that $b$ is not zero, then we use the notation $a/b$ to represent the rational number $\frac{a}{b}$. If $a/b$ is an integer (i.e., if $a \bmod b = 0$), then $b$ *(evenly) divides* $a$. We assume exact rational arithmetic when we deal with these expressions.

Consider the program segment

```
L1 :    do I1 = 10, 100, 3
L2 :       do I2 = 50, 5, −2
S  :          X(2I1 − 1, I1 + I2) = · · ·
T  :          · · · = · · · X(3I2 + 1, 2I1 + 2) · · ·
           enddo
        enddo
```

We explain the nature of the dependence problem by means of this example. This program is a nest of two loops containing two assignment statements. We show only the output variable of $S$ and one input variable of $T$, and they are both elements of a two-dimensional array $X$. We focus on the memory references (reads and writes) coming from the instances of these two variables only. The variables shown cause a dependence between $S$ and $T$, only if there is at least one memory location that both statements reference during the sequential execution of the program. In order to test for dependence, we first formulate the problem mathematically and then try to solve it.

Consider an iteration of the double loop $(L_1, L_2)$ defined by a set of values $I_1 = i_1, I_2 = i_2$ of the index variables. In this iteration, statement $S$ writes the memory location defined by $X(2i_1 - 1, i_1 + i_2)$. Consider another iteration defined by a set of values $I_1 = j_1, I_2 = j_2$. In that iteration, statement $T$ reads the memory location defined by $X(3j_2 + 1, 2j_1 + 2)$. These two locations are identical iff

$$\begin{aligned} 2i_1 - 1 &= 3j_2 + 1 \\ i_1 + i_2 &= 2j_1 + 2, \end{aligned}$$

that is, iff

$$2i_1 - 3j_2 = 2 \tag{1.1}$$

$$i_1 - 2j_1 + i_2 = 2. \tag{1.2}$$

Thus, to decide if there exists a memory location that is referenced by both $S$ and $T$, we need to decide if there exist values $i_1, j_1$ of $I_1$ and $i_2, j_2$ of $I_2$ that satisfy both equations (1.1) and (1.2). Now, $i_1$ and $j_1$ are integers characterized by the following conditions:

| | |
|---|---|
| $10 \le i_1 \le 100,$ | $10 \le j_1 \le 100,$ |
| $(i_1 - 10)/3$ is an integer, | $(j_1 - 10)/3$ is an integer. |

Similarly, $i_2$ and $j_2$ are integers satisfying the conditions:

| | |
|---|---|
| $5 \le i_2 \le 50,$ | $5 \le j_2 \le 50,$ |
| $(i_2 - 50)/(-2)$ is an integer, | $(j_2 - 50)/(-2)$ is an integer. |

Thus, we are looking for a simultaneous integer solution to equations (1.1)-(1.2) that satisfy all these conditions.

This problem can be simplified to a large extent if we can eliminate conditions of the type "$(i_1 - 10)/3$ is an integer." This could be done by labeling the loops differently as we shall now see. Note that the index variable $I_1$ takes the sequence of values: $10, 13, \ldots, 100$, and the index variable $I_2$ takes the sequence of values: $50, 48, \ldots, 6$. The first sequence is increasing, the second is decreasing, and neither is a sequence of *contiguous* integers. To standardize the labeling of loop iterations, we agree to number the iterations of each loop $0, 1, 2, \ldots$, in the order of sequential execution. This process, called *loop normalization*, is achieved by introducing a new variable for each loop. Since $(I_1 - 10)/3$ must be an integer, we may define an integer variable $\hat{I}_1$, called the *iteration variable* of $L_1$, by[1]

$$I_1 = 10 + 3\hat{I}_1. \tag{1.3}$$

Likewise, the *iteration variable* of $L_2$ is an integer variable denoted by $\hat{I}_2$ and defined by

$$I_2 = 50 - 2\hat{I}_2. \tag{1.4}$$

The sequence of iterations of $L_1$ corresponds to the values $0, 1, \ldots, 30$ of $\hat{I}_1$, and the sequence of iterations of $L_2$ corresponds to the values $0, 1, \ldots, 22$ of $\hat{I}_2$. Each iteration variable takes an increasing sequence of contiguous integer values.[2]

Let $\hat{\imath}_1, \hat{\jmath}_1$ denote the values of $\hat{I}_1$ corresponding to the values $i_1, j_1$ of $I_1$, and $\hat{\imath}_2, \hat{\jmath}_2$ the values of $\hat{I}_2$ corresponding to the values $i_2, j_2$ of $I_2$. From equations (1.3) and (1.4), we get

$$i_1 = 10 + 3\hat{\imath}_1 \qquad i_2 = 50 - 2\hat{\imath}_2 \tag{1.5}$$

$$j_1 = 10 + 3\hat{\jmath}_1 \qquad j_2 = 50 - 2\hat{\jmath}_2. \tag{1.6}$$

[1]The "hat" in $\hat{I}$ does not indicate a vector. We denote vectors and matrices by bold letters.

[2]Changing the description of a loop from the given one in terms of its index variable to the one in terms of its iteration variable may produce complicated array subscripts. It is not necessary for dependence analysis to completely rewrite a program in terms of iteration variables. We need to use the iteration variable of a loop only when the values of its index variable do not form an increasing sequence of contiguous integers (i.e., when the stride is not one). Whether the initial limit of the loop is zero or not is not important for our purpose.

Also, it follows from the ranges of $\hat{I}_1, \hat{I}_2$ given above that

$$0 \le \hat{\imath}_1 \le 30 \qquad 0 \le \hat{\imath}_2 \le 22 \tag{1.7}$$

$$0 \le \hat{\jmath}_1 \le 30 \qquad 0 \le \hat{\jmath}_2 \le 22. \tag{1.8}$$

Using (1.5) and (1.6), we rewrite equations (1.1) and (1.2) in terms of $\hat{\imath}_1, \hat{\jmath}_1, \hat{\imath}_2, \hat{\jmath}_2$:

$$6\hat{\imath}_1 + 6\hat{\jmath}_2 = 132 \tag{1.9}$$

$$3\hat{\imath}_1 - 6\hat{\jmath}_1 - 2\hat{\imath}_2 = -38. \tag{1.10}$$

Thus, there is dependence between $S$ and $T$ (i.e., there is a memory location referenced by both statements), iff there exists an integer solution $(\hat{\imath}_1, \hat{\jmath}_1, \hat{\imath}_2, \hat{\jmath}_2)$ to equations (1.9)-(1.10) satisfying the constraints (1.7)-(1.8). If we want the complete set of all instance[3] pairs $(S(i_1, i_2), T(j_1, j_2))$ involved in this dependence, then we need to find the set of *all* such solutions. If we want to know about a more specific dependence, then additional constraints will arise. For example, statement $T$ depends on statement $S$ and that dependence is *carried* by the outer loop $L_1$, iff there exists an integer solution to (1.9)-(1.10) that satisfies (1.7)-(1.8) and the condition $\hat{\imath}_1 < \hat{\jmath}_1$. (Note that since $\hat{\imath}_1$ and $\hat{\jmath}_1$ are integers, the inequality $\hat{\imath}_1 < \hat{\jmath}_1$ is equivalent to $\hat{\imath}_1 \le \hat{\jmath}_1 - 1$.)

We have demonstrated that a typical dependence problem (when loop bounds and subscripts are affine functions of index variables) leads to a system of linear diophantine equations and a system of linear inequalities. First, we have to decide if this combined system has an integer solution, and then find the set of all solutions in case it does. This ends the problem formulation stage. Next, we look at various approaches to solving the problem.

The general approach to a dependence problem consists of a few basic steps. They are stated below in the form of an informal algorithm.

**Algorithm 1.1** Given a dependence problem with a system of linear equations and a system of linear inequalities (each with rational coefficients), this algorithm decides if the particular dependence exists, and finds relevant information in case it does.

[3]We will label instances of a statement by values of index variables. Once $\hat{\imath}_1, \hat{\imath}_2, \hat{\jmath}_1, \hat{\jmath}_2$ are found, the corresponding index values $i_1, i_2, j_1, j_2$ can be computed from (1.5) and (1.6).

1. Decide if there is an integer solution to the system of equations (Algorithm I.2.1 and Theorem I.3.6).

2. If there is no solution, then terminate; the particular dependence does not exist.

3. Otherwise, find the general integer solution to the system of equations in terms of certain undetermined integer parameters (Theorem I.3.6).

4. Substitute the general solution into the inequalities to derive a new set of inequalities involving those integer parameters.

5. Decide if this new system of inequalities has an integer solution.

6. If there is no solution, then terminate; the particular dependence does not exist.

7. Otherwise, the dependence exists. Find the set of all integer solutions to the new inequalities, and compute information relevant to the dependence.

A system of linear diophantine equations can be solved in polynomial time. As pointed out above, we know how to test if such a system has an integer solution, and how to find the general solution in case it does. The problem of deciding if a system of linear inequalities has an integer solution is NP-complete [Schr 87]. There are several integer programming algorithms (e.g., the cutting plane method) available to solve this problem. However, there is no convenient way to represent the general integer solution to a system of linear inequalities (see [Fior 72] and [Will 83]).

For our example, we find that equations (1.9)-(1.10) have integer solutions. Indeed, starting with the coefficient matrix **A** for the system of equations

$$(\hat{\imath}_1, \hat{\jmath}_1, \hat{\imath}_2, \hat{\jmath}_2) \begin{pmatrix} 6 & 3 \\ 0 & -6 \\ 0 & -2 \\ 6 & 0 \end{pmatrix} = (132, -38),$$

we use Algorithm I.2.1 to find matrices

$$\mathbf{U} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & -1 \\ 2 & 0 & 3 & -2 \\ 0 & 1 & -3 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{S} = \begin{pmatrix} 6 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix},$$

such that $\mathbf{U}$ is unimodular, $\mathbf{S}$ is echelon, and $\mathbf{UA} = \mathbf{S}$. Next, we apply Theorem I.3.6. Any integer vector of the form $(22, -38, t_3, t_4)$ will satisfy the equation

$$(t_1, t_2, t_3, t_4)\mathbf{S} = (132, -38).$$

Thus, equations (1.9)–(1.10) have integer solutions, and the general solution is given by

$$\begin{aligned} (\hat{\imath}_1, \hat{\jmath}_1, \hat{\imath}_2, \hat{\jmath}_2) &= (22, -38, t_3, t_4)\mathbf{U} \\ &= (-38 + 2t_3, t_4, -38 + 3t_3 - 3t_4, 60 - 2t_3), \end{aligned} \tag{1.11}$$

where $t_3$ and $t_4$ are unknown integer parameters. Substituting from (1.11) into the conditions (1.7)–(1.8), we get the following system of inequalities (after rearrangement):

$$\left.\begin{array}{rrl} -2t_3 & & \le -38 \\ 2t_3 & & \le 68 \\ & -t_4 & \le 0 \\ & t_4 & \le 30 \\ -3t_3 & + 3t_4 & \le -38 \\ 3t_3 & - 3t_4 & \le 60 \\ 2t_3 & & \le 60 \\ -2t_3 & & \le -38. \end{array}\right\} \tag{1.12}$$

Note that while the number of inequalities is the same as the number in (1.7)–(1.8), the number of variables has been reduced from 4 to 2.

Although we can decide if this system has an integer solution by any integer programming method, it is commonly believed that using a general integer programming method to solve each and every dependence problem is not a good idea. The reasons usually cited are

1. A general integer programming method is expensive;

2. The system arising from a typical dependence problem is too small to justify the cost of such a method;

3. The compiler typically needs to solve a large number of dependence problems in the course of processing a typical program, so that the integer programming method will have to be used a large number of times.

This forces us to seek a hierarchical approach to solve the dependence problem. We will use several different techniques to handle different classes of problems. These classes may overlap and their boundaries are not strictly defined.

First, there is a class of *simple* problems. It turns out that in real programs, there is a large number of dependence problems where the system of new inequalities obtained in Step 4 of Algorithm 1.1, involving the unknown integer parameters, is trivial to solve. (This is system (1.12) for our example.) In this case, the dependence problem is completely settled: we can easily decide if the particular dependence is present, and compute all relevant information in case it does. A simple problem may arise in several different ways:

A. The system of new inequalities has no variables;

B. It has exactly one variable;

C. It can be broken up into subsystems each of which has exactly one variable.

See Example I.5.6 for a dependence problem that falls in Case A. The dependence problem for a one-dimensional array in a single loop belongs to Case B. A typical example is the following program:

**do** $I = 1, 100, 2$
  $X(2I) = \cdots$
  $\cdots = \cdots X(3I + 1) \cdots$
**enddo**

See also Example I.5.7 for a problem in Case B. There are problems in a perfect nest of loops with constant loop limits[4] that can be broken up into mutually independent single-loop problems. They fall in Case C. A typical example is the program

[4]Here, "constant" means loop-invariant.

```
do I1 = 0, 100, 1
   do I2 = 0, 50, 1
      X(3I1, 3I2) = ···
      ··· = ··· X(2I1 + 1, I2 + 2) ···
   enddo
enddo
```

Note that the range of $I_2$ is independent of $I_1$. Also, each subscript of each program variable has exactly one of $I_1$ and $I_2$, and the index variables in a pair of corresponding subscripts match.

Next, we describe an approximate algorithm for the solution to the dependence problem, that checks if there is an integer solution to the system of equations, and a *real* solution to the combined system of equations and inequalities.

**Algorithm 1.2** Given a dependence problem with a system of linear equations and a system of linear inequalities (each with rational coefficients), this algorithm either decides that the particular dependence does not exist, or terminates with the assumption that it may exist.

1. Decide if there is an integer solution to the system of equations. (We may use Algorithm I.2.1 and Theorem I.3.6.)

2. If there is no solution, then terminate; the particular dependence does not exist.

3. Decide if there is a real solution to the equations that satisfies the inequalities.

4. If there is no solution, then terminate; the particular dependence does not exist.

5. Otherwise, assume that the dependence exists. (The algorithm is inconclusive in this case.)

It is not very common to see a dependence problem in real programs where this algorithm predicts a false dependence. In fact, it takes some effort to construct a meaningful example where there are an integer solution to the equations and a real solution to the equations satisfying the inequalities, but no integer solution satisfying the

inequalities. Also, there are cases where the existence of such a real solution guarantees the existence of an integer solution.

The important new step in this algorithm is Step 3. Different ways to execute this step lead to different dependence testing methods; we describe two of these methods below.

The approximate method, which we call the *method of bounds*, has been successfully implemented in some form or another in many restructuring compilers, both in research and commercial environments. This method is an implementation of Algorithm 1.2, where Step 3 is based on finding extreme values of real-valued linear functions in certain sets. To get an idea about this method, consider Equation (1.10). Define a linear function $f : \mathbf{R}^4 \rightarrow \mathbf{R}$ by

$$f(x_1, y_1, x_2, y_2) = 3x_1 - 6y_1 - 2x_2.$$

Let $P$ denote the subset of $\mathbf{R}^4$ defined by the inequalities (1.7)-(1.8), where $\hat{\imath}_1, \hat{\jmath}_1, \hat{\imath}_2$, and $\hat{\jmath}_2$ are replaced by $x_1, y_1, x_2$, and $y_2$, respectively. The minimum and maximum values of $f$ in $P$ are as follows:

$$\min_P f = 3 \times 0 - 6 \times 30 - 2 \times 22 = -224$$
$$\max_P f = 3 \times 30 - 6 \times 0 - 2 \times 0 = 90.$$

It is clear then that for each point $(x_1, y_1, x_2, y_2) \in P$, we must have

$$-224 \leq f(x_1, y_1, x_2, y_2) \leq 90.$$

In other words, if $-224 \leq c \leq 90$ fails to hold, then the equation

$$f(x_1, y_1, x_2, y_2) = c$$

has no (real) solution in $P$. The converse also happens to be true: if $-224 \leq c \leq 90$ holds, then the above equation does have a real solution in $P$. Thus, (1.10) has a solution in $P$ since $-224 \leq -38 \leq 90$ is true. This technique can be extended to decide if the system of equations (1.9)-(1.10) has a simultaneous (real) solution in $P$ (i.e., a real solution satisfying the inequalities (1.7)-(1.8)).

Although the method of bounds is applicable, in principle, to any dependence problem, it is practical for only those problems where

the region $P$ is simple enough to allow *easy* computation of the extreme values. For such cases, it is normally a polynomial time method. This method is usually applied to a problem that involves a one-dimensional array in a loop nest with constant loop limits. It remains practical if we move to a two-dimensional array in such a loop nest; an example is the double loop we have been working on in this chapter. Computation of extreme values becomes harder as we move to higher dimensional arrays and/or non-constant loop limits. For multi-dimensional arrays, a relaxed version of the method of bounds may be used.

When it is hard to apply the method of bounds, we may apply another method which we call the *method of elimination.* This method is the following implementation of Algorithm 1.2.

**Algorithm 1.3** Given a dependence problem with a system of linear equations and a system of linear inequalities (each with rational coefficients), this algorithm either decides that the particular dependence does not exist, or terminates with the assumption that it may exist.

1. Decide if there is an integer solution to the system of equations (Algorithm I.2.1 and Theorem I.3.6).

2. If there is no solution, then terminate; the particular dependence does not exist.

3. Otherwise, find the general integer solution to the system of equations in terms of certain unknown integer parameters (Theorem I.3.6).

4. Substitute the general solution into the inequalities to derive a new set of inequalities involving those integer parameters.

5. Decide if this new system of inequalities has a real solution by Fourier's method of elimination (Algorithm I.3.2).

6. If there is no solution, then terminate; the particular dependence does not exist.

7. Otherwise, assume that the dependence exists. (The algorithm is inconclusive in this case.)

Since Algorithm I.3.2 (Fourier) is not polynomial [Schr 87], the method of elimination is not polynomial either. However, the number of unknown parameters is typically rather small, and an efficient implementation of Fourier elimination should not be too time consuming. An advantage of this method is that Algorithm I.3.2 gives the ranges of the variables in a hierarchical way. (See the comments after Algorithm I.3.2.) From that we get a description of all real solutions to the combined system of equations and inequalities. Enumeration may be used then, if desired, to decide if there is also an integer solution and find all integer solutions when they exist.

For example, applying it to the system (1.12), we see that there are real solutions, and the general integer solution can be represented by the following set of inequalities:

$$19 \le t_3 \le 30 \tag{1.13}$$

$$\lceil \max(0, t_3 - 20) \rceil \le t_4 \le \lfloor \min(30, t_3 - 38/3) \rfloor. \tag{1.14}$$

It is not immediately clear that there are integer vectors $(t_3, t_4)$ satisfying these constraints. To establish that, we take $t_3 = 19$ and get $0 \le t_4 \le 6$. Thus, $(19,0), (19,1), \ldots, (19,6)$ are integer solutions to (1.12). Now, going back to our original dependence problem, we can say that there is dependence between statements $S$ and $T$. The set of all pairs of instances $(S(i_1, i_2), T(j_1, j_2))$ of $S$ and $T$ involved in this dependence can be found as follows:

1. For each integral value of $t_3$ in (1.13), find all integral values of $t_4$ that satisfy (1.14).

2. For each integer vector $(t_3, t_4)$ satisfying (1.13)-(1.14), find the corresponding value of $(\hat{\imath}_1, \hat{\jmath}_1, \hat{\imath}_2, \hat{\jmath}_2)$ from (1.11).

3. For each vector $(\hat{\imath}_1, \hat{\jmath}_1, \hat{\imath}_2, \hat{\jmath}_2)$ found in the previous step, find the corresponding index values $(i_1, i_2)$ and $(j_1, j_2)$ from (1.5)-(1.6).

There could be a variation of the method of elimination where Algorithm I.3.2 is applied to the original equations and inequalities; we will not discuss it here.

Finally, established integer programming methods are the last resort. It is quite possible that for a very large and complicated dependence problem, the cost of an efficient integer programming method

will be less than the cost of the method of elimination (especially if we perform enumeration after elimination).

In this book, we give a precise formulation of the (linear) dependence problem, and discuss the solution of simple problems, the method of bounds, and the method of elimination. For integer programming methods, see [Schr 87] or any standard text on the subject.

The rest of the book is organized as follows. In Chapter 2, we study the dependence problem for a one-dimensional array in a single loop. Many of the important dependence concepts and techniques are introduced here, and we provide a large amount of detail to give the reader easy access to the subject. Chapter 3 deals with double loops; it acts as a bridge between Chapter 2 and the chapters after 3. The matrix notation introduced in Volume I is extended to perfect loop nests with arbitrary strides in Chapter 4. Chapter 5 deals with a general program where the loops are legally, but not necessarily perfectly, nested. We have tried to show the evolution of ideas and concepts as we move from a single loop to a general program. The emphasis in chapters 2 through 5 is on problem formulation. Simple problems are covered in these chapters as they occur naturally. The method of bounds is discussed in Chapter 6; the mathematical results needed there are described in the appendix. The method of elimination is covered in Chapter 7, and some conclusions are given in Chapter 8.

# Chapter 2

# Single Loops

## 2.1 Introduction

In this book, we introduce many important dependence concepts and techniques in terms of a single loop containing assignment statements, then generalize first to a double loop, then to a perfect loop nest, and finally to a program consisting of arbitrarily nested loops. This way the reader is exposed to complicated ideas and notation in steps that makes understanding of the material easier. This chapter is devoted to a study of the dependence problem in a single loop.

In Section 2.2, we discuss in detail the index and iteration spaces of a single loop. Dependence concepts are introduced in Section 2.3, and the dependence problem is stated in Section 2.4. In Section 2.5, we give a complete solution to the linear dependence problem for a one-dimensional array. This is one of the simple problems mentioned in Chapter 1. Finally, in Section 2.6, we introduce the method of bounds and apply it to solve the same problem.

## 2.2 Index and Iteration Spaces

If the loops in a perfect nest do not all have unit strides, it is important to distinguish between the index and iteration spaces of the nest. The two spaces are both finite subsets of $\mathbf{Z}^m$ (where $m$ is the number of loops in the nest) with the same number of points, but the index space is more *spread out* than the iteration space. In this section, we

introduce these concepts in their simplest form, that is, in terms of a single loop.

The model program for this chapter is a single **do** loop $L$ of the form

$L$: **do** $I = p, q, \theta$
  $H(I)$
**enddo**

where $I$ is the *index variable* of the loop, $p$ the *initial limit*, $q$ the *final limit*, and $\theta$ the *stride*. The loop limits are integer-valued and the stride is a nonzero integer. The loop body, $H(I)$, is a sequence of assignment statements.

The *index values* (or *index points*) of $L$ are the values of the index variable $I$. The *index space* of the loop is the set of all its index points. The following theorem characterizes the index values.

**Theorem 2.1** *An integer $I$ is an index value of the loop $L$ iff*

(a) $(I - p)/\theta$ *is an integer, and*

(b) $0 \le (I - p)/\theta \le (q - p)/\theta$.

PROOF. We have $\theta \neq 0$. The index variable $I$ of the loop $L$ is characterized by the following two conditions:

1. $I$ takes the sequence of values: $p, p + \theta, p + 2\theta, \ldots,$

2. $I$ lies between $p$ and $q$:

$$\left.\begin{array}{ll} p \le I \le q & \text{if } \theta > 0 \\ p \ge I \ge q & \text{if } \theta < 0. \end{array}\right\}$$

The two cases of Condition 2 can be combined into the statement

$$p/\theta \le I/\theta \le q/\theta,$$

that holds for all nonzero values of $\theta$. From this, we get

$$0 \le (I - p)/\theta \le (q - p)/\theta. \tag{2.1}$$

Thus, $I$ takes precisely those integral values for which $(I - p)/\theta$ is a nonnegative integer between 0 and $(q - p)/\theta$. □

We define an integer variable $\hat{I}$ by the equation

$$\hat{I} = (I - p)/\theta, \tag{2.2}$$

so that

$$I = p + \hat{I}\theta. \tag{2.3}$$

We call $\hat{I}$ the *iteration variable* of the loop $L$. The *iteration values* (or *iteration points*) of $L$ are the values of $\hat{I}$, and the *iteration space* of the loop is the set of all its iteration points. The following theorem characterizes the iteration values.

**Theorem 2.2** *An integer $\hat{I}$ is an iteration value of the loop $L$ iff*

$$0 \le \hat{I} \le (q - p)/\theta. \tag{2.4}$$

PROOF. The proof follows from the definition of $\hat{I}$ in (2.2) and the inequalities in (2.1). □

**Theorem 2.3** *The iteration space of $L$ is the set of integers $\{0, 1, \ldots, \hat{q}\}$ where $\hat{q} = \lfloor (q - p)/\theta \rfloor$.*

*The index space of $L$ is the set of integers $\{p + \hat{I}\theta : 0 \le \hat{I} \le \hat{q}\}$. In the sequential execution of $L$, the index space is traversed in the direction of increasing $\hat{I}$.*

PROOF. It follows from Theorem 2.2 that the iteration values are $0, 1, \ldots, \hat{q}$. The index values are given by (2.3) where $\hat{I}$ denotes an iteration point.

In the sequential execution of $L$, the iterations of $L$ are taken in the order $H(p), H(p + \theta), H(p + 2\theta), \ldots, H(p + \hat{q}\theta)$. Since the corresponding values of $\hat{I}$ are $0, 1, 2, \ldots, \hat{q}$, it follows that the index space is traversed in the direction of increasing $\hat{I}$. □

**Corollary 1** *A single loop with an arbitrary stride can be written as an equivalent loop with unit stride.*

PROOF. Consider the following loop with unit stride:

$\hat{L}$: **do** $\hat{I} = 0, \hat{q}, 1$
$H(p + \hat{I}\theta)$
**enddo**

The iterations of $\hat{L}$ are $H(p), H(p+\theta), H(p+2\theta), \ldots, H(p+\hat{q}\theta)$, and they are executed in this order. Hence, $\hat{L}$ is equivalent to $L$. □

We refer to $\hat{q}$ as the *normalized final limit* of $L$. This loop is not executed (i.e., the index and iteration spaces are empty) if $\hat{q} < 0$, which happens if either $\theta > 0$ and $p > q$, or $\theta < 0$ and $p < q$. Otherwise, the total number of iterations (i.e., the number of index or iteration points) is $\hat{q} + 1$, and the value of $I$ in the last iteration is $p + \hat{q}\theta$.

There is a one-to-one correspondence between the index and iteration spaces of $L$. The mapping $I \mapsto \hat{I}$ is called *loop normalization*. The iteration points are contiguous while the index points generally are not. The iteration variable $\hat{I}$ always increases from zero in steps of one as the loop is executed sequentially. The index variable $I$, on the other hand, may increase or decrease depending on the sign of $\theta$, and will not change in steps of one if $|\theta| > 1$. For this reason, it is easier to work with $\hat{I}$ rather than with $I$.

---

**Example 2.1** Consider a single loop of the form:

$L$: **do** $I = 120, 10, -3$
$H(I)$
**enddo**

The index variable $I$ decreases from 120 in steps of 3, and we write

$$I = 120 - 3\hat{I} \tag{2.5}$$

where $\hat{I}$ is an integer. Also, $I$ must lie between 120 and 10. From the inequalities

$$120 \geq 120 - 3\hat{I} \geq 10,$$

we get $0 \geq -3\hat{I} \geq -110$, so that $0 \leq \hat{I} \leq 110/3$. Here, $\hat{q} = \lfloor 110/3 \rfloor = 36$, and the iteration points of the loop are $0, 1, 2, \ldots, 36$. The iteration space is the set of these 37 integers. The index points of $L$ are found from (2.5) by plugging in the values of $\hat{I}$:

$$120, 120 - 3(1), 120 - 3(2), \ldots, 120 - 3(36).$$

Thus, the index space is the set $\{120, 117, 114, \ldots, 12\}$.

---

**EXERCISES 2.2**

1. What are the smallest and the largest values of $I$ in the model loop $L$ of this section?
2. Find a closed-form expression for the number of iterations of $L$, that holds for all values of $p, q$, and $\theta$ (including cases where the loop fails to execute).
3. Find the index and iteration spaces of the loop $L$, when
   (a) $p = 0, q = 9, \theta = 1$;
   (b) $p = 17, q = 39, \theta = 5$;
   (c) $p = -15, q = 20, \theta = 2$;
   (d) $p = 10, q = -13, \theta = -3$.
4. Find the number of iterations of $L$ and the value of $I$ in the last iteration, when
   (a) $p = 0, q = 1000, \theta = 1$;
   (b) $p = -17, q = 390, \theta = 4$;
   (c) $p = -15, q = -200, \theta = 3$;
   (d) $p = 1001, q = -137, \theta = -7$.

## 2.3 Dependence Concepts

Consider two, not necessarily distinct, assignment statements $S$ and $T$ in our model loop

```
L:   do I = p, q, θ
        H(I)
     enddo
```

If $S$ lexically precedes $T$ in the program, we write $S < T$. The meaning of the notation $S \leq T$ is then clear, and it is obvious that $\leq$ is a total order in the set of assignment statements in the program.

An index point (i.e., a value of the index variable $I$) determines an instance of each assignment statement in the loop. Since, by hypothesis, there are no conditionals in the program, all such instances are executed. Let $S(i)$ denote the instance of statement $S$ determined by an index point $i$, and $T(j)$ the instance of statement $T$ determined by

an index point $j$. The *distance* from $S(i)$ to $T(j)$ is defined to be the integer $(\hat{j} - \hat{\imath})$, where $\hat{\imath}$ and $\hat{j}$ are the iteration points corresponding to $i$ and $j$, respectively.[1]

We leave to the reader the proof of the following result.

**Lemma 2.4** *Consider two assignment statements $S$ and $T$ in the single loop $L$. Let $d$ denote the distance from an instance $S(i)$ of $S$ to an instance $T(j)$ of $T$. In the sequential execution of the program, $S(i)$ is executed before $T(j)$ iff either $d > 0$, or $d = 0$ and $S < T$.*

The concept of dependence can be introduced in many different contexts. We define dependence first between statement instances, and then between statements. An instance $T(j)$ of a statement $T$ *depends* on an instance $S(i)$ of a statement $S$, if there exists a memory location $\mathcal{M}$ such that

1. Both $S(i)$ and $T(j)$ reference (read or write) $\mathcal{M}$;
2. $S(i)$ is executed before $T(j)$ in the sequential execution of the program;
3. During sequential execution, the location $\mathcal{M}$ is not written in the time period from the end of execution of $S(i)$ to the beginning of execution of $T(j)$.

The statements $S$ and $T$ need not be distinct, but Condition 2 requires that the instances $S(i)$ and $T(j)$ be distinct. Let $\hat{\imath}$ and $\hat{j}$ denote the iteration values corresponding to $i$ and $j$, respectively. This dependence is *loop-carried* if $\hat{\imath} < \hat{j}$; it is *loop-independent* if $\hat{\imath} = \hat{j}$ (i.e., if $i = j$). In the loop-independent case, we necessarily have $S < T$ since $S(i)$ is to be executed before $T(i)$.

Since a memory reference is either a "read" or a "write," a pair of statement instances can reference the same memory location in four different ways. This leads to four different *types* of dependence:

1. $T(j)$ is *flow dependent* on $S(i)$, if $S(i)$ writes $\mathcal{M}$ and $T(j)$ reads it (the value computed by $S(i)$ is used by $T(j)$);

[1] See [Pugh 92b] for concerns expressed about the proper definition of dependence distance, and [Wolf 94] for related comments.

2. $T(j)$ is *anti-dependent* on $S(i)$, if $S(i)$ reads $\mathcal{M}$ and $T(j)$ writes it ($S(i)$ uses the value in $\mathcal{M}$ before it is changed by $T(j)$);

3. $T(j)$ is *output dependent* on $S(i)$, if $S(i)$ and $T(j)$ both write $\mathcal{M}$ (the value computed by $T(j)$ is stored after the value computed by $S(i)$ is stored);

4. $T(j)$ is *input dependent* on $S(i)$, if both $S(i)$ and $T(j)$ read $\mathcal{M}$ (the "read" by $T(j)$ comes after the "read" by $S(i)$).

Now, we consider dependence between two statements. A statement $T$ *depends* on a statement $S$, if there is at least one instance $S(i)$ of $S$ and one instance $T(j)$ of $T$, such that $T(j)$ depends on $S(i)$. We can be more specific. For example, $T$ is *flow dependent* on $S$ if there is an instance pair $(S(i), T(j))$ such that $T(j)$ is flow dependent on $S(i)$. One may similarly define what is meant by $T$ is *anti-dependent*, *output dependent*, or *input dependent* on $S$.

The *dependence* of $T$ on $S$ can be formally defined to be the set of all instance pairs $(S(i), T(j))$ such that $T(j)$ depends on $S(i)$. Thus, $T$ depends on $S$ iff the dependence of $T$ on $S$ is nonempty. Sometimes, it is convenient to say that there is dependence *between* $S$ and $T$ if either $T$ depends on $S$, or $S$ on $T$ (or both).

The dependence of $T$ on $S$ can be separated into two parts: the *loop-carried* part, consisting of all instance pairs $(S(i), T(j))$ such that $T(j)$ depends on $S(i)$ and $\hat{\imath} < \hat{\jmath}$; and the *loop-independent* part, consisting of all instance pairs $(S(i), T(i))$ such that $T(i)$ depends on $S(i)$. The loop-independent part is necessarily empty if $T \preceq S$.

We can also break up the dependence of $T$ on $S$ by the type of dependence between a pair of instances. This way we get four subsets:

1. The *flow dependence* of $T$ on $S$ consists of all instance pairs $(S(i), T(j))$ such that $T(j)$ is flow dependent on $S(i)$;

2. The *anti-dependence* of $T$ on $S$ consists of all instance pairs $(S(i), T(j))$ such that $T(j)$ is anti-dependent on $S(i)$;

3. The *output dependence* of $T$ on $S$ consists of all instance pairs $(S(i), T(j))$ such that $T(j)$ is output dependent on $S(i)$;

4. The *input dependence* of $T$ on $S$ consists of all instance pairs $(S(i), T(j))$ such that $T(j)$ is input dependent on $S(i)$.

These subsets are not necessarily pairwise disjoint (e.g., $T(j)$ may be both flow dependent and anti-dependent on $S(i)$).

The dependence between two statements can be described in terms of the program variables[2] involved. (See Section I.4.3.) The instances of the output variable of a statement $S$ determine the memory locations written by the instances of $S$. On the other hand, the instances of the input variables of $S$ determine the memory locations read by the instances of $S$. Let $u$ denote a variable of statement $S$ and $v$ a variable of statement $T$. The pair $(u, v)$ *causes* a dependence of $T$ on $S$, if there are statement instances $S(i)$ and $T(j)$, such that $T(j)$ depends on $S(i)$, and the corresponding memory location $\mathcal{M}$ is represented by both the instance of $u$ for $I = i$ and the instance of $v$ for $I = j$. If $u$ is the output variable of $S$ and $v$ an input variable of $T$, then $(u, v)$ can cause a flow dependence of $T$ on $S$, or an anti-dependence of $S$ on $T$, or both. The output variables of $S$ and $T$ can cause an output dependence of $T$ on $S$, and/or of $S$ on $T$. Two input variables of the statements can cause only an input dependence.

A *distance* for the dependence of $T$ on $S$ is the distance $(\hat{j} - \hat{\imath})$ from an instance $S(i)$ of $S$ to an instance $T(j)$ of $T$, where $T(j)$ depends on $S(i)$. A given dependence has at least one, but usually many distances. Since $S(i)$ must be executed before $T(j)$ by the definition of dependence, we have the following result as a direct consequence of Lemma 2.4.

**Theorem 2.5** *If a statement $T$ depends on a statement $S$ in the loop $L$, then each dependence distance satisfies $d \geq 0$. The equality $d = 0$ is possible only if $S < T$.*

The relation of dependence between statements is denoted by $\delta$, and we write $S \,\delta\, T$ to indicate that $T$ depends on $S$.[3] The *statement dependence graph* of the given program is the directed graph that represents the relation $\delta$. (In other words, it *is* the graph of $\delta$; see Section I.1.4.) We denote flow dependence, anti-dependence, output

[2] We often use the term "variable" somewhat loosely, although the exact meaning should always be clear from the context. In the current context, for "program variables," we may take the output variable $X(I)$ of $S$ and the input variable $X(I-2)$ of $T$ in Example 2.2 on the next page.

[3] Often, $\delta$ is used as a generic symbol for dependence. For example, one may write $S(i) \,\delta\, T(j)$ to denote dependence between statement instances.

dependence, and input dependence by the symbols $\delta^{\mathrm{f}}$, $\delta^{\mathrm{a}}$, $\delta^{\mathrm{o}}$, and $\delta^{\mathrm{i}}$, respectively. (Thus, for example, $S\,\delta^{\mathrm{f}}\,T$ means $T$ is flow dependent on $S$.) These relations may overlap, and they constitute the main relation of dependence: $\delta = \delta^{\mathrm{f}} \cup \delta^{\mathrm{a}} \cup \delta^{\mathrm{o}} \cup \delta^{\mathrm{i}}$. From this point, dependence will not include input dependence, unless otherwise stated,

The transitive closure $\overline{\delta}$ of the relation $\delta$ is the relation of *indirect dependence.* Thus, a statement $T$ is *indirectly dependent* on a statement $S$ if there is a (directed) path from $S$ to $T$ in the statement dependence graph. In symbols, we have $S\,\overline{\delta}\,T$ if there is a nonempty sequence of statements $S_1, S_2, \ldots, S_N$ such that

$$S = S_1,\ S_1\,\delta\,S_2, \ldots,\ S_{N-1}\,\delta\,S_N,\ S_N = T.$$

Indirect dependence between statement instances is defined similarly.

---

**Example 2.2** Consider the loop

```
L :        do I = 4, 100, 2
   S :        X(I) = X(I) + 1
   T :        Y(2I - 4) = X(I) + X(I + 1) + X(I + 2) + X(I - 2)
   U :        Y(I) = Z(I)
              enddo
```

The index values of the loop are $4, 6, 8, \ldots, 100$, and the corresponding iteration values are $0, 1, 2, \ldots, 48$. The first three iterations of $L$ along with the last one are shown below:

```
S(4) :      X(4) = X(4) + 1
T(4) :      Y(4) = X(4) + X(5) + X(6) + X(2)
U(4) :      Y(4) = Z(4)

S(6) :      X(6) = X(6) + 1
T(6) :      Y(8) = X(6) + X(7) + X(8) + X(4)
U(6) :      Y(6) = Z(6)

S(8) :      X(8) = X(8) + 1
T(8) :      Y(12) = X(8) + X(9) + X(10) + X(6)
U(8) :      Y(8) = Z(8)
 ⋮           ⋮
S(100) :    X(100) = X(100) + 1
T(100) :    Y(196) = X(100) + X(101) + X(102) + X(98)
U(100) :    Y(100) = Z(100)
```

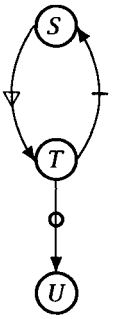


Figure 2.1: Statement dependence graph for Example 2.2.

(Statement instances are always labeled by index values.) From this pattern, it is easy to figure out the dependence structure of the program. Note the following facts:

1. $S$ does not depend on itself.
2. Fix the output variable $X(I)$ of $S$ and take the input variables of $T$, one at a time, in the order of their appearance. We can make these observations:
   (a) $X(I)$ of $S$ and $X(I)$ of $T$ cause a flow dependence of $T$ on $S$. This dependence is loop-independent; it has no loop-carried part. The only dependence distance is 0.
   (b) $X(I)$ of $S$ and $X(I+1)$ of $T$ do not cause a dependence between $S$ and $T$.
   (c) $X(I)$ of $S$ and $X(I+2)$ of $T$ cause an anti-dependence of $S$ on $T$. This dependence is loop-carried; it has no loop-independent part. The only dependence distance is 1.
   (d) $X(I)$ of $S$ and $X(I-2)$ of $T$ cause a flow dependence of $T$ on $S$ that is also loop-carried, without any loop-independent part. Again, the only dependence distance is 1.
3. There is an output dependence of $U$ on $T$. It has a loop-carried part and a loop-independent part. There are several dependence distances: $0, 1, 2, \ldots$.
4. $U$ does not depend on $S$, but $U$ depends on $S$ indirectly.

The statement dependence graph for the program is given in Figure 2.1. Strictly speaking, this figure shows the superposition of the graphs of the relations $\delta^{f}, \delta^{a}, \delta^{o}$. (We have ignored input dependences between statements, and will usually do so in future examples.) Note the different markings on the edges: the triangle represents flow dependence, the dash anti-dependence, and the circle output dependence. If we want to show more detailed dependence information, then a more detailed graph will be necessary. For example, to emphasize that there are two pairs of program variables each of which causes a flow dependence of $T$ on $S$, we would create a graph with two flow dependence edges from $S$ to $T$, each labeled with the appropriate variables.

---

**EXERCISES 2.3**

1. Take two iteration points $\hat{\imath}$ and $\hat{\jmath}$ of the model loop $L$. Let $i$ and $j$ denote the corresponding index points. Let $d = \hat{\jmath} - \hat{\imath}$ and $d' = j - i$. How is $d'$ related to $d$? Explain the difficulties that may arise if the distance from a statement instance $S(i)$ to a statement instance $T(j)$ is defined to be $d'$.
2. Write down the set of all pairs of statement instances that constitute the flow dependence of $T$ on $S$ in Example 2.2. Show the different subsets that represent the flow dependences caused by different pairs of variables.
3. Write down the set of all pairs of statement instances that constitute the output dependence of $U$ on $T$ in Example 2.2. Give all the distances for this dependence.
4. Are there any input dependences between statements in Example 2.2?
5. Give a complete description of the dependence structure of the loop in Example 2.2 (including a dependence graph) after changing the loop-header to

   (a) $L$: **do** $I = 4, 100, 1$

   (b) $L$: **do** $I = 4, 100, 3$

   (c) $L$: **do** $I = 100, 4, -2$

## 2.4 Dependence Problem

Consider again the model loop $L$ and two statements $S$ and $T$ in its body. Let $u$ denote a variable of $S$ and $v$ a variable of $T$. Now, we address the practical problem of determining whether $u$ and $v$ cause a dependence between the statements: a dependence of $T$ on $S$, or

of $S$ on $T$, or both. In this context, we are not interested in the type (flow, anti-, output, or input) of dependence.

This book focuses on array elements, and here we consider the case where $u$ and $v$ are both elements of a given array $X$. Throughout this chapter, we assume that $X$ is one-dimensional, unless otherwise specified. Let $u \equiv X(f(I))$ and $v \equiv X(g(I))$, where $f$ and $g$ are real-valued functions that return integer values for integer values of $I$. When $S < T$, $u$ is the output variable of $S$, and $v$ an input variable of $T$, we may partially represent the model program as follows:

$L$: **do** $I = p, q, \theta$

$\vdots$

$S$: $X(f(I)) = \cdots$

$\vdots$

$T$: $\cdots = \cdots X(g(I)) \cdots$

$\vdots$

**enddo**

Theorem 2.6 gives necessary conditions under which $u$ and $v$ will cause a dependence between $S$ and $T$.

**Theorem 2.6** *Consider any two statements $S$ and $T$ in the loop $L$. Let $X(f(I))$ denote a variable of $S$ and $X(g(I))$ a variable of $T$, where $X$ is a one-dimensional array. If these variables cause a dependence between $S$ and $T$, then the equation*

$$f(p + \theta\hat{\imath}) - g(p + \theta\hat{\jmath}) = 0 \tag{2.6}$$

*has an integer solution $(\hat{\imath}, \hat{\jmath})$ such that*

$$0 \le \hat{\imath} \le \hat{q} \quad \text{and} \quad 0 \le \hat{\jmath} \le \hat{q}, \tag{2.7}$$

*where $\hat{q} = \lfloor (q - p)/\theta \rfloor$. In each of the following special cases, the solution satisfies an additional condition:*

(a) *If $S < T$ and $S \,\delta\, T$, then $\hat{\imath} \le \hat{\jmath}$;*

(b) *If $S = T$ and $S \,\delta\, S$, then $\hat{\imath} < \hat{\jmath}$;*

(c) *If $S < T$ and $T \,\delta\, S$, then $\hat{\imath} > \hat{\jmath}$.*

PROOF. Suppose that the variables in question do cause a dependence between $S$ and $T$. Then, there is an instance $S(i)$ of $S$ and an instance $T(j)$ of $T$, such that either $T(j)$ depends on $S(i)$, or $S(i)$ depends on $T(j)$. In either case, there is a memory location $\mathcal{M}$ that is referenced by both $S(i)$ and $T(j)$, and that is represented by the instance $X(f(i))$ of $X(f(I))$ in $S(i)$ and also by the instance $X(g(j))$ of $X(g(I))$ in $T(j)$. This implies that we must have

$$f(i) = g(j).$$

Next, we restate the problem in terms of iteration values. Let $\hat{\imath}$ and $\hat{\jmath}$ denote the iteration values corresponding to $i$ and $j$, respectively. The index and iteration values are connected by the equations (see (2.3)):

$$i = p + \theta\hat{\imath} \qquad \text{and} \qquad j = p + \theta\hat{\jmath},$$

where $\hat{\imath}$ and $\hat{\jmath}$ satisfy (2.7) (Theorem 2.2). If we switch over to iteration values, the equation $f(i) = g(j)$ becomes

$$f(p + \theta\hat{\imath}) = g(p + \theta\hat{\jmath})$$

which is (2.6).

The additional condition (a), (b), or (c) in the theorem follows from the part of the definition of dependence that postulates the ordering of the statement instances. If $T$ depends on $S$, then the instance $S(i)$ must be executed before the instance $T(j)$ in the sequential execution of the loop. In that case, if $S < T$, we must have $\hat{\imath} \le \hat{\jmath}$; but if $S = T$, then the strict inequality $\hat{\imath} < \hat{\jmath}$ has to hold (since $S(i)$ cannot depend on itself). If $S$ depends on $T$, then the instance $S(i)$ is executed after the instance $T(j)$. In that case, if $S < T$, we must have $\hat{\imath} > \hat{\jmath}$. □

The equation $f(i) = g(j)$ is the *dependence equation* for the program variables $X(f(I))$ and $X(g(I))$ *in terms of index values.* The index values $i$ and $j$ satisfy the conditions (Theorem 2.1):

1. $(i - p)/\theta$ is an integer,
2. $0 \le (i - p)/\theta \le (q - p)/\theta$,
3. $(j - p)/\theta$ is an integer,
4. $0 \le (j - p)/\theta \le (q - p)/\theta$.

These conditions are called the *dependence constraints* for $X(f(I))$ and $X(g(I))$ *in terms of index values.* Equation (2.6) is the *dependence equation* for the program variables $X(f(I))$ and $X(g(I))$ *in terms of iteration values.* The inequalities in (2.7) are the *dependence constraints* for the same variables *in terms of iteration values.* In this book, dependence equations and constraints are expressed in terms of iteration values, unless otherwise stated.

Note that we cannot expect to have a strict converse to Theorem 2.6, since its proof does not take into account the third condition of dependence that precludes writing of the memory location $\mathcal{M}$ in the time period between execution of instances $S(i)$ and $T(j)$. Suppose there is a solution $(\hat{\imath}, \hat{\jmath})$ to Equation (2.6) that satisfies (2.7). This tells us that there are instances $S(i)$ of $S$ and $T(j)$ of $T$, both of which reference a certain memory location $\mathcal{M}$. The ordering of $\hat{\imath}$ and $\hat{\jmath}$ gives the order of execution of the two instances. However, we do not know if $\mathcal{M}$ will be written in the period between the two executions.

Theorem 2.7 is a weak converse to Theorem 2.6; we omit the proof.

**Theorem 2.7** *Suppose that Equation (2.6) has an integer solution* $(\hat{\imath}, \hat{\jmath})$ *that satisfies the conditions in (2.7).*

(a) *If* $\hat{\imath} < \hat{\jmath}$, *then* $S \overline{\delta} T$.

(b) *If* $\hat{\imath} > \hat{\jmath}$, *then* $T \overline{\delta} S$.

(c) *If* $\hat{\imath} = \hat{\jmath}$ *and* $S < T$, *then* $S \overline{\delta} T$.

When checking for dependence between two statements caused by a pair of program variables, we ignore all other variables and statements. Thus, we often abuse language and say "$T$ depends on $S$" when we know merely that $T$ depends on $S$ *indirectly.* (Note that most loop transformations are effectively guided by indirect and not ordinary dependence.) The *dependence problem* for the variable $X(f(I))$ of $S$ and the variable $X(g(I))$ of $T$ consists of finding all integer solutions $(\hat{\imath}, \hat{\jmath})$ to the dependence equation (2.6) that satisfy the dependence constraints (2.7), and then partitioning the solution set into three subsets based on whether $\hat{\imath} < \hat{\jmath}$, or $\hat{\imath} > \hat{\jmath}$, or $\hat{\imath} = \hat{\jmath}$.

The following example shows how Theorem 2.7 is applied to check for dependence between statements in a single loop, caused by elements of a one-dimensional array.

**Example 2.3** Consider a program segment of the form:

$L$: **do** $I = 1, 100, 2$

$\vdots$

$S$: $X(2I) = \cdots$

$\vdots$

$T$: $\cdots = \cdots X(3I + 1) \cdots$

$\vdots$

**enddo**

Let us test for dependence between the statements $S$ and $T$ caused by the two variables shown. Here, we have $p = 1, q = 100, \theta = 2$, $f(I) = 2I$ and $g(I) = 3I + 1$. Since

$$\begin{aligned} f(p + \theta\hat{\imath}) - g(p + \theta\hat{\jmath}) &= 2(p + \theta\hat{\imath}) - 3(p + \theta\hat{\jmath}) - 1 \\ &= 2(1 + 2\hat{\imath}) - 3(1 + 2\hat{\jmath}) - 1 \\ &= 4\hat{\imath} - 6\hat{\jmath} - 2, \end{aligned}$$

the dependence equation (2.6) becomes

$$4\hat{\imath} - 6\hat{\jmath} = 2. \tag{2.8}$$

Since $\hat{q} = \lfloor (100 - 1)/2 \rfloor = 49$, the constraints in (2.7) become

$$0 \le \hat{\imath} \le 49 \quad \text{and} \quad 0 \le \hat{\jmath} \le 49.$$

Thus, we are seeking integer solutions $(\hat{\imath}, \hat{\jmath})$ to (2.8) such that $\hat{\imath}$ and $\hat{\jmath}$ are nonnegative integers not larger than 49. One such solution is clearly $(2, 1)$, and there are many others. However, any such solution must necessarily satisfy $\hat{\imath} > \hat{\jmath}$ (Exercise 3). Thus, we conclude (by Theorem 2.7) that $T \,\overline{\delta}\, S$ holds and that $S \,\overline{\delta}\, T$ does not. Whether $T \,\delta\, S$ is true would depend on the part of the loop body not shown.

**EXERCISES 2.4**

1. In Theorem 2.6, provide the additional condition in the special case:
   (a) $T$ has a loop-carried dependence on $S$;
   (b) $S$ has a loop-carried dependence on $T$;

(c) $S < T$ and $T$ has a loop-independent dependence on $S$;

(d) $T < S$ and $S$ has a loop-independent dependence on $T$.

2. Prove Theorem 2.7.
3. Explain why each nonnegative solution $(\hat{\imath}, \hat{\jmath})$ to (2.8) must satisfy $\hat{\imath} > \hat{\jmath}$.
4. Find the dependence equation and dependence constraints in Example 2.3 after changing the loop-header to

   (a) $L$: **do** $I = 1, 100, -2$

   (b) $L$: **do** $I = 5, 100, 3$

   In each case, decide if $T$ depends indirectly on $S$, or $S$ on $T$, or both.
5. Let $X$ denote a two-dimensional array. Extend the concepts introduced so far to give a description of the dependence problem posed by the two variables in the following loop:

   $L$: **do** $I = 0, 100, 1$
   $S$: $X(2I, 3I + 1) = \cdots$
   $T$: $\cdots = \cdots X(3I + 1, 4I + 3) \cdots$
   **enddo**

   Can you solve this problem? What are your conclusions?

## 2.5 Solution to the Linear Problem

In a single loop, the dependence equation for two array elements of the form $X(aI + a_0)$ and $X(bI + b_0)$ turns out to be a linear diophantine equation in two variables. We saw this in Example 2.3. We start this section with a discussion on two-variable linear diophantine equations, and then we will return to the dependence problem.

A linear diophantine equation in any number of variables can be solved by Theorem I.3.5 and Algorithm I.2.1. We describe here a method that is customized for the two-variable case. Consider first the extended Euclid's algorithm (see [Knut 73] and [Knut 81]).

**Algorithm 2.1** (*Extended Euclid's algorithm*). Given two integers $a$ and $b$, this algorithm finds $g = \gcd(a, b)$, and two integers $x_0, y_0$ such that $ax_0 - by_0 = g$. The algorithm uses six auxiliary integer variables: $u, v, h, q, \alpha, \beta$.

$(u, v) \leftarrow (1, 0)$
$(g, h) \leftarrow (|a|, |b|)$

**do while** $h > 0$
  $q \leftarrow \lfloor g/h \rfloor$
  $(\alpha, \beta) \leftarrow (u - qv, g - qh)$
  $(u, v) \leftarrow (v, \alpha)$
  $(g, h) \leftarrow (h, \beta)$
**enddo**
$x_0 \leftarrow \text{sig}(a) \cdot u$
**if** $b = 0$
**then** $y_0 \leftarrow 0$
**else** $y_0 \leftarrow (ax_0 - g)/b$. □

The extended Euclid's algorithm is a special case of Algorithm I.2.1. Note that we have taken the minus sign in $ax_0 - by_0 = g$ only to conform to the notation of our dependence problem.

Although the gcd $g$ is unique, the integers $x_0$ and $y_0$ in the expression $g = ax_0 - by_0$ are not. However, *any* two integers with this property will work when we try to solve an equation of the form $ax - by = c$, as we see below.

**Theorem 2.8** *Let $a, b, c$ denote integers such that $a$ and $b$ are not both zero, and let $g = \gcd(a, b)$. The linear diophantine equation*

$$ax - by = c \tag{2.9}$$

*has a solution iff $g$ divides $c$. When a solution exists, the general solution is given by*

$$(x, y) = (cx_0/g + bt/g, cy_0/g + at/g),$$

*where $t$ is an arbitrary integer parameter, and $(x_0, y_0)$ is a pair of integers such that $g = ax_0 - by_0$.*

PROOF. The integer matrix $\mathbf{U} = \begin{pmatrix} x_0 & y_0 \\ b/g & a/g \end{pmatrix}$ is unimodular since $\det(\mathbf{U}) = (ax_0 - by_0)/g = 1$. Any 2-vector $(x, y)$ can be written as $(x, y) = (s, t)\mathbf{U}$, where $(s, t) = (x, y)\mathbf{U}^{-1}$. Since $\mathbf{U}$ is unimodular, $(x, y)$ is an integer vector iff $(s, t)$ is an integer vector. Now, $(x, y)$ is a solution

to Equation (2.9) iff

$$\begin{aligned} c &= ax - by \\ &= (x, y)\begin{pmatrix} a \\ -b \end{pmatrix} \\ &= (s, t)\mathbf{U}\begin{pmatrix} a \\ -b \end{pmatrix} \\ &= (s, t)\begin{pmatrix} x_0 & y_0 \\ b/g & a/g \end{pmatrix}\begin{pmatrix} a \\ -b \end{pmatrix} \\ &= (s, t)\begin{pmatrix} ax_0 - by_0 \\ 0 \end{pmatrix} \\ &= (s, t)\begin{pmatrix} g \\ 0 \end{pmatrix} \\ &= gs. \end{aligned}$$

If $g$ does not divide $c$, then $s$ is not an integer and hence $(x, y)$ cannot be an integer vector. Therefore, Equation (2.9) has no integer solution in this case. If $g$ does divide $c$, then all integer solutions to (2.9) are given by the formula $(x, y) = (c/g, t)\mathbf{U}$, where $t$ is an arbitrary integer parameter. This formula gives the expressions for $x$ and $y$ in the theorem. □

This theorem is a basic result of elementary number theory. It is a variation of Corollary 1 to Theorem I.3.5; here we gave a direct proof for the sake of completeness.

---

**Example 2.4** Let us solve the diophantine equation

$$21x - 34y = 2. \tag{2.10}$$

Here we have $a = 21, b = 34, c = 2$. First, apply Algorithm 2.1 to $a$ and $b$. The assignments in the **while** loop are shown in Table 2.1. Near the end of the algorithm, we get

$\gcd(21, 34) = g = 1,$
$x_0 = 13,$
$y_0 = (21 \times 13 - 1)/34 = 8,$

| $u$ | $v$ | $g$ | $h$ | $q$ | $\alpha$ | $\beta$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 21 | 34 | 0 | 1 | 21 |
| 0 | 1 | 34 | 21 | 1 | −1 | 13 |
| 1 | −1 | 21 | 13 | 1 | 2 | 8 |
| −1 | 2 | 13 | 8 | 1 | −3 | 5 |
| 2 | −3 | 8 | 5 | 1 | 5 | 3 |
| −3 | 5 | 5 | 3 | 1 | −8 | 2 |
| 5 | −8 | 3 | 2 | 1 | 13 | 1 |
| −8 | 13 | 2 | 1 | 2 | −34 | 0 |
| 13 | −34 | 1 | 0 | | | |

Table 2.1: Steps of Algorithm 2.1 for $a = 21$ and $b = 34$.

so that $ax_0 - by_0 = 21(13) - 34(8) = 1 = g$. By Theorem 2.8, Equation (2.10) has a solution, and the general solution is given by

$$(x, y) = (26 + 34t, 16 + 21t),$$

where $t$ is an integer parameter. Taking $t = 0, 1, -1$, we get the particular solutions: $(26, 16), (60, 37), (-8, -5)$.

---

We now return to the dependence problem between two statements $S$ and $T$ in a loop of the form

$L$: **do** $I = p, q, \theta$
  $H(I)$
**enddo**

posed by a variable $X(aI + a_0)$ of $S$ and a variable $X(bI + b_0)$ of $T$, where $a, a_0, b$, and $b_0$ are integer constants. Since here we have

$$f(I) = aI + a_0 \qquad \text{and} \qquad g(I) = bI + b_0,$$

the dependence equation (2.6), namely

$$f(p + \theta\hat{\imath}) - g(p + \theta\hat{\jmath}) = 0,$$

becomes

$$[a(p + \theta\hat{\imath}) + a_0] - [b(p + \theta\hat{\jmath}) + b_0] = 0$$

or

$$(a\theta)\hat{\imath} - (b\theta)\hat{\jmath} = b_0 - a_0 + (b-a)p. \tag{2.11}$$

We rewrite the dependence constraints (2.7):

$$\left.\begin{array}{l} 0 \le \hat{\imath} \le \hat{q} \\ 0 \le \hat{\jmath} \le \hat{q} \end{array}\right\} \tag{2.12}$$

where $\hat{q} = \lfloor (q-p)/\theta \rfloor$. Here, the dependence problem consists of finding all integer solutions $(\hat{\imath}, \hat{\jmath})$ to Equation (2.11) subject to the constraints (2.12), and partitioning the solution set into three subsets based on whether $\hat{\imath}$ is less than, greater than, or equal to $\hat{\jmath}$. A solution $(\hat{\imath}, \hat{\jmath})$ indicates dependence of $T$ on $S$ if $\hat{\imath} < \hat{\jmath}$, dependence of $S$ on $T$ if $\hat{\imath} > \hat{\jmath}$, and dependence of $T$ on $S$ if $\hat{\imath} = \hat{\jmath}$ in case $S < T$. (See Theorem 2.7 and the comment following it about not distinguishing between dependence and indirect dependence.)

The following theorem gives a necessary condition for dependence, that is not always sufficient.

**Theorem 2.9** (gcd test). *If a variable $X(aI + a_0)$ of a statement $S$ and a variable $X(bI + b_0)$ of a statement $T$ cause a dependence between $S$ and $T$ in the loop $L$, then the integer $(b_0 - a_0 + bp - ap)$ is an integral multiple of the integer $\theta \cdot \gcd(a, b)$.*

PROOF. We consider only the nontrivial case where $a$ and $b$ are not both zero. As we just saw, the dependence equation in this case is (2.11). By Theorem 2.6, the existence of dependence between statements $S$ and $T$ implies the existence of an integer solution to (2.11). By Theorem 2.8, that is possible iff $\gcd(a\theta, b\theta)$ divides the right-hand-side of (2.11). Since $\gcd(a\theta, b\theta) = |\theta| \cdot \gcd(a, b)$ (Lemma I.3.3(e)), it follows that $\theta \cdot \gcd(a, b)$ divides the integer $(b_0 - a_0 + bp - ap)$. □

W. L. Cohagan [Coha 73] first recognized the usefulness of the following result in dependence analysis: *Consider two mappings $f$ and $g$ of $\mathbf{Z}$ into $\mathbf{Z}$, defined by $f(I) = aI + a_0$ and $g(I) = bI + b_0$, where $a, a_0, b$, and $b_0$ are integers. The ranges of $f$ and $g$ are disjoint iff* $\gcd(a, b)$ *does not divide* $(b_0 - a_0)$. It follows immediately from Theorem 2.8, since these ranges intersect iff there is an integer solution

to the equation $ax - by = b_0 - a_0$. Application of this result to dependence analysis is contained in Theorem 2.9 (see Example 2.4).

Algorithm 2.2 given below solves the linear dependence problem in a single loop. It originated in [Bane 79]. (See also [WoBa 87].) Its current form is adapted from Algorithm I.3.1; see Section I.3.5 for more details. The basic idea of the algorithm is as follows. Check to see if the dependence equation (2.11) has an integer solution. If not, then there is no dependence. Otherwise, find the set of all solutions and partition it into three subsets, such that one subset consists of all solutions $(\hat{\imath}, \hat{\jmath})$ where $\hat{\imath} < \hat{\jmath}$, another consists of all solutions $(\hat{\imath}, \hat{\jmath})$ where $\hat{\imath} > \hat{\jmath}$, and the third consists of all solutions of the form $(\hat{\imath}, \hat{\imath})$.

To check the existence of a solution to (2.11), we apply the gcd test in two stages. First, see if $\theta$ divides the integer $[b_0 - a_0 + (b - a)p]$. If it does, then test whether $[b_0 - a_0 + (b - a)p]/\theta$ is an integral multiple of $\gcd(a, b)$. If it is (i.e., if the gcd test passes), then find all solutions to (2.11). The second stage of the gcd test and computation of the solution set are done separately for the three cases: $a = b = 0$, $a = b \neq 0$, and $a \neq b$. For the first two cases, it is trivial to find $\gcd(a, b)$ and then solutions to (2.11) if they exist. Euclid's algorithm and Theorem 2.8 need to be used explicitly only for the case $a \neq b$.

**Algorithm 2.2** Given integers $p, q, \theta(\neq 0), a, a_0, b$, and $b_0$, such that we have a single loop of the form

```
L:   do I = p, q, θ
        H(I)
     enddo
```

two assignment statements $S$ and $T$ in the loop body $H(I)$ with $S \leq T$, a variable $X(aI + a_0)$ of $S$ and a variable $X(bI + b_0)$ of $T$, where $X$ is a one-dimensional array, this algorithm

- Decides if the variables $X(aI + a_0)$ of $S$ and $X(bI + b_0)$ of $T$ cause a dependence of $T$ on $S$, or a dependence of $S$ on $T$;
- Finds the set $\Psi_1$ of all iteration value pairs $(\hat{\imath}, \hat{\jmath})$, such that $\hat{\imath} < \hat{\jmath}$ and the instance $T(j)$ of $T$ depends on the instance $S(i)$ of $S$ (the index values corresponding to $\hat{\imath}$ and $\hat{\jmath}$ are $i$ and $j$);
- Finds the set $\Psi_{-1}$ of all iteration value pairs $(\hat{\imath}, \hat{\jmath})$, such that $\hat{\jmath} < \hat{\imath}$ and the instance $S(i)$ depends on the instance $T(j)$;

- In case $S < T$, finds the set $\Psi_0$ of all iteration value pairs $(\hat{\imath}, \hat{\imath})$, such that the instance $T(i)$ depends on the instance $S(i)$;
- Finds the distances for each loop-carried dependence.

**1.** Set $\hat{q} \leftarrow \lfloor (q - p)/\theta \rfloor$.

If $\hat{q} < 0$, then terminate the algorithm; the loop is not executed.

**2.** Set $c_0 \leftarrow b_0 - a_0 + (b - a)p$.

[The dependence equation (2.11) becomes $\theta(a\hat{\imath} - b\hat{\jmath}) = c_0$.]

If $c_0 \bmod \theta \neq 0$, then there is no dependence between statements $S$ and $T$; terminate the algorithm.

**3.** [Here, $\theta$ does divide $c_0$.]

Set $c \leftarrow c_0/\theta$.

[The dependence equation is $a\hat{\imath} - b\hat{\jmath} = c$.]

**4.** Initialize $\Psi_1, \Psi_{-1}, \Psi_0$ to the empty set.

**5.** [There are three cases based on whether $a$ and $b$ are equal to each other or to zero.]

Select the proper case:

**Case** ($a = b = 0$): Go to Step 6.

**Case** ($a = b \neq 0$): Go to Step 7.

**Case** ($a \neq b$): Go to Step 9.

**6.** [Since here $a = b = 0$, the dependence equation degenerates to $0 = c$.]

If $c \neq 0$, then terminate the algorithm; there is no dependence between statements $S$ and $T$. Otherwise, set

$$\begin{aligned} \Psi_1 &\leftarrow \{(\hat{\imath}, \hat{\jmath}) : 0 \leq \hat{\imath} < \hat{\jmath} \leq \hat{q}\} \\ \Psi_{-1} &\leftarrow \{(\hat{\imath}, \hat{\jmath}) : 0 \leq \hat{\jmath} < \hat{\imath} \leq \hat{q}\} \\ \Psi_0 &\leftarrow \{(\hat{\imath}, \hat{\imath}) : 0 \leq \hat{\imath} \leq \hat{q}\}. \end{aligned}$$

Since $\hat{q} \geq 0$, the set $\Psi_0$ is always nonempty. So, if $S < T$, then $T$ depends on $S$ (loop-independent dependence). If $\hat{q} > 0$, then the

sets $\Psi_1$ and $\Psi_{-1}$ are both nonempty. In this case, $T$ depends on $S$ and $S$ depends on $T$. Also, the distance for each dependence is 1 (why?).

Terminate the algorithm.

**7.** [Here, $a = b \neq 0$. The dependence equation is $a(\hat{\imath} - \hat{\jmath}) = c$.]

If $c \bmod a \neq 0$, then there is no dependence between statements $S$ and $T$; terminate the algorithm. Otherwise, set $c_1 \leftarrow c/a$.

**8.** [The dependence equation is now $\hat{\imath} - \hat{\jmath} = c_1$ whose general solution is $(\hat{\imath}, \hat{\jmath}) = (t + c_1, t)$, where $t$ is an arbitrary integer parameter. The dependence constraints (2.12) reduce to the inequalities

$$\left.\begin{array}{ccccc} 0 & \leq & t + c_1 & \leq & \hat{q} \\ 0 & \leq & t & \leq & \hat{q}. \end{array}\right\}$$

These inequalities are consistent iff $\hat{q} \geq |c_1|$.]

If $\hat{q} < |c_1|$, then terminate the algorithm; there is no dependence between statements $S$ and $T$. Otherwise, select the proper case based on the sign of $c_1$:

**Case** ($c_1 < 0$): Set

$$\Psi_1 \leftarrow \{(t + c_1, t) : |c_1| \leq t \leq \hat{q}\}.$$

$T$ depends on $S$ with a unique distance $|c_1|$.

**Case** ($c_1 > 0$): Set

$$\Psi_{-1} \leftarrow \{(t + c_1, t) : 0 \leq t \leq \hat{q} - c_1\}.$$

$S$ depends on $T$ with a unique distance $c_1$.

**Case** ($c_1 = 0$): Set

$$\Psi_0 \leftarrow \{(t, t) : 0 \leq t \leq \hat{q}\}.$$

If $S < T$, then $T$ depends on $S$.

Terminate the algorithm.

**9.** [Here, $a \neq b$.]

By Algorithm 2.1, find the gcd $g$ of $a$ and $b$, and two integers $\hat{\imath}_0$ and $\hat{\jmath}_0$ such that $a\hat{\imath}_0 - b\hat{\jmath}_0 = g$.
If $c \bmod g \neq 0$, then there is no dependence between statements $S$ and $T$; terminate the algorithm.

**10.** [Now, $g$ does divide $c$.]

Set $(a_1, b_1, c_1) \leftarrow (a/g, b/g, c/g)$.

**11.** [The general solution to $a\hat{\imath} - b\hat{\jmath} = c$ is

$$\left.\begin{aligned} \hat{\imath} &= c_1\hat{\imath}_0 + b_1 t \\ \hat{\jmath} &= c_1\hat{\jmath}_0 + a_1 t \end{aligned}\right\} \tag{2.13}$$

where $t$ is an arbitrary integer parameter (Theorem 2.8). Since it must satisfy the constraints (2.12), the inequalities

$$\left.\begin{aligned} 0 \le c_1\hat{\imath}_0 + b_1 t \le \hat{q} \\ 0 \le c_1\hat{\jmath}_0 + a_1 t \le \hat{q} \end{aligned}\right\}$$

hold. We will find the range $\tau_1 \le t \le \tau_2$ of $t$.]

Select the proper case based on the sign of $b_1$:

**Case** ($b_1 = 0$): If $0 \le c_1\hat{\imath}_0 \le \hat{q}$ does not hold, then there is no dependence between statements $S$ and $T$; terminate the algorithm. Otherwise, set $\tau_1 \leftarrow -\infty$, $\tau_2 \leftarrow \infty$.

**Case** ($b_1 > 0$): Set

$$\tau_1 \leftarrow \lceil -c_1\hat{\imath}_0/b_1 \rceil \text{ and } \tau_2 \leftarrow \lfloor (\hat{q} - c_1\hat{\imath}_0)/b_1 \rfloor.$$

**Case** ($b_1 < 0$): Set

$$\tau_1 \leftarrow \lceil (\hat{q} - c_1\hat{\imath}_0)/b_1 \rceil \text{ and } \tau_2 \leftarrow \lfloor -c_1\hat{\imath}_0/b_1 \rfloor.$$

Select the proper case based on the sign of $a_1$:

**Case** ($a_1 = 0$): If $0 \le c_1\hat{\jmath}_0 \le \hat{q}$ does not hold, then there is no dependence between statements $S$ and $T$; terminate the algorithm.

**Case** ($a_1 > 0$): Set

$$\tau_1 \leftarrow \max(\tau_1, \lceil -c_1\hat{j}_0/a_1 \rceil) \text{ and } \tau_2 \leftarrow \min(\tau_2, \lfloor (\hat{q}-c_1\hat{j}_0)/a_1 \rfloor).$$

**Case** ($a_1 < 0$): Set

$$\tau_1 \leftarrow \max(\tau_1, \lceil (\hat{q}-c_1\hat{j}_0)/a_1 \rceil) \text{ and } \tau_2 \leftarrow \min(\tau_2, \lfloor -c_1\hat{j}_0/a_1 \rfloor).$$

If $\tau_1 > \tau_2$, then there is no dependence between statements $S$ and $T$; terminate the algorithm.

**12.** [Find the set $\Psi_0$. From (2.13) we get

$$\hat{j} - \hat{\imath} = (a_1 - b_1)t - c_1(\hat{\imath}_0 - \hat{j}_0). \tag{2.14}$$

Since $a \neq b$, we have $a_1 \neq b_1$. Let $\xi$ denote the value of $t$ for which $\hat{\imath} = \hat{j}$. If $\xi$ is an integer between $\tau_1$ and $\tau_2$, then it is an element (the only element in this case) of $\Psi_0$.]

Set

$$\xi \leftarrow c_1(\hat{\imath}_0 - \hat{j}_0)/(a_1 - b_1).$$

If $\xi$ is an integer such that $\tau_1 \leq \xi \leq \tau_2$, then set

$$\Psi_0 \leftarrow \{(c_1\hat{\imath}_0 + b_1\xi, c_1\hat{j}_0 + a_1\xi)\}.$$

**13.** [Find the sets $\Psi_1$ and $\Psi_{-1}$. The difference $(\hat{j} - \hat{\imath})$ given by (2.14) is either positive for all $t < \xi$ and negative for all $t > \xi$, or negative for all $t < \xi$ and positive for all $t > \xi$. We compute the intersections $[\tau_1, \tau_2] \cap (-\infty, \xi)$ and $[\tau_1, \tau_2] \cap (\xi, \infty)$. See Algorithm I.3.1 and figures I.3.4, I.3.5 for details.]

Set

$$\begin{aligned}
\tau_3 &\leftarrow \lceil \xi - 1 \rceil \\
\tau_4 &\leftarrow \lfloor \xi + 1 \rfloor \\
\tau_5 &\leftarrow \min(\tau_2, \tau_3) \\
\tau_6 &\leftarrow \max(\tau_1, \tau_4).
\end{aligned}$$

Select the proper case based on the sign of $(a_1 - b_1)$:

**Case** ($a_1 > b_1$): If $\tau_6 \leq \tau_2$, then set

$$\Psi_1 \leftarrow \{(c_1\hat{\imath}_0 + b_1 t, c_1\hat{\jmath}_0 + a_1 t) : \tau_6 \leq t \leq \tau_2\}.$$

If $\tau_1 \leq \tau_5$, then set

$$\Psi_{-1} \leftarrow \{(c_1\hat{\imath}_0 + b_1 t, c_1\hat{\jmath}_0 + a_1 t) : \tau_1 \leq t \leq \tau_5\}.$$

**Case** ($a_1 < b_1$): If $\tau_1 \leq \tau_5$, then set

$$\Psi_1 \leftarrow \{(c_1\hat{\imath}_0 + b_1 t, c_1\hat{\jmath}_0 + a_1 t) : \tau_1 \leq t \leq \tau_5\}.$$

If $\tau_6 \leq \tau_2$, then set

$$\Psi_{-1} \leftarrow \{(c_1\hat{\imath}_0 + b_1 t, c_1\hat{\jmath}_0 + a_1 t) : \tau_6 \leq t \leq \tau_2\}.$$

**14.** If $\Psi_1 = \emptyset$, then there is no loop-carried dependence of statement $T$ on statement $S$. Otherwise, there is such a dependence, and if $\Psi_1$ has the form

$$\Psi_1 = \{(c_1\hat{\imath}_0 + b_1 t, c_1\hat{\jmath}_0 + a_1 t) : \alpha \leq t \leq \beta\}$$

then the corresponding set of distances is

$$\{(a_1 - b_1)t + c_1(\hat{\jmath}_0 - \hat{\imath}_0) : \alpha \leq t \leq \beta\}.$$

If $\Psi_{-1} = \emptyset$, then there is no loop-carried dependence of statement $S$ on statement $T$. Otherwise, there is such a dependence, and if $\Psi_{-1}$ has the form

$$\Psi_{-1} = \{(c_1\hat{\imath}_0 + b_1 t, c_1\hat{\jmath}_0 + a_1 t) : \alpha \leq t \leq \beta\}$$

then the corresponding set of distances is

$$\{(b_1 - a_1)t + c_1(\hat{\imath}_0 - \hat{\jmath}_0) : \alpha \leq t \leq \beta\}.$$

If $S < T$ and $\Psi_0 \neq \emptyset$, then there is a loop-independent dependence of $T$ on $S$. Otherwise, there is no such dependence.

**15.** Terminate the algorithm. □

We would like to summarize here the major facts gleaned from the description of the above algorithm. First, note that the sets $\Psi_1, \Psi_{-1}, \Psi_0$ represent the following three dependences, respectively:

1. loop-carried dependence of $T$ on $S$,
2. loop-carried dependence of $S$ on $T$,
3. loop-independent dependence of $T$ on $S$ (provided $S < T$).

We cannot have a loop-independent dependence of $S$ on $T$, since $S \leq T$ by hypothesis.

Assume now that $a, b$ are not both zero. The values of $t$ defining a nonempty $\Psi$-set form a finite sequence of contiguous integers (steps 8, 12, 13). An element of a $\Psi$-set is a pair of iteration points $(\hat{\imath}, \hat{\jmath})$ of the loop $L$. We can compute the corresponding pair of index points $(i, j)$ by the formula

$$(i, j) = (p + \theta\hat{\imath}, p + \theta\hat{\jmath})$$

obtained from the relation $I = p + \theta\hat{I}$. This allows us to find the set of all instance pairs $(S(i), T(j))$ that are involved in the particular dependence. If $t$ has the range $\alpha \leq t \leq \beta$, where $\alpha$ and $\beta$ are integers such that $\alpha \leq \beta$, then the number of such instance pairs (i.e., the size of the $\Psi$-set) is $(\beta - \alpha + 1)$.

Next, consider the case $a = b \neq 0$. Let $\hat{q} \geq |c_1|$ (Step 8). Then, exactly two of the sets $\Psi_1, \Psi_{-1}$, and $\Psi_0$ are empty, and one is nonempty. Thus, exactly one of the following is true:

1. $T$ has a loop-carried dependence on $S$;
2. $S$ has a loop-carried dependence on $T$;
3. $T$ has a loop-independent dependence on $S$ (provided $S < T$).

In each case, the dependence distance is unique and independent of iteration points. It has the value $|c_1|$. This means, for example, that if $T$ depends on $S$, then an instance $T(j)$ of $T$ always depends on an instance $S(i)$ of $S$, as long as the corresponding iteration points $\hat{\jmath}$ and $\hat{\imath}$ satisfy $\hat{\jmath} - \hat{\imath} = |c_1|$. We characterize this situation by saying that the dependence is *uniform*, and that the distance is a *uniform* dependence distance. (See Example I.3.6.)

Finally, consider the case $a \neq b$. Any combination of the sets $\Psi_1, \Psi_{-1}$, and $\Psi_0$ may be nonempty. Thus, one or more of the three dependences listed above may be present at the same time. The loop-independent dependence, if present, holds only for one iteration ($\Psi_0$ is either empty or a singleton). There may be many distances for a loop-carried dependence, and they are not uniform. (See Example I.3.5.)

---

**Example 2.5** Let us test if there is a dependence between statements $S$ and $T$ in the following program (caused by the variables shown):

```
L:      do I = 10, 200, 5
  S:      X(7I + 2) = ···
  T:      ··· = ··· X(3I + 17) ···
        enddo
```

The steps of Algorithm 2.2 applied to this problem are shown below.

Input to the algorithm:
$p = 10, q = 200, \theta = 5,$
$a = 7, a_0 = 2, b = 3, b_0 = 17.$

1. $\hat{q} \leftarrow \lfloor (q - p)/\theta \rfloor = 38.$
   Since $\hat{q} > 0$, continue.

2. $c_0 \leftarrow b_0 - a_0 + (b - a)p = -25.$
   Since $c_0 \bmod \theta = 0$, continue.

3. $c \leftarrow c_0/\theta = -5.$

4. $\Psi_1 \leftarrow \emptyset, \Psi_{-1} \leftarrow \emptyset, \Psi_0 \leftarrow \emptyset.$

5. Since $a \neq b$, go to Step 9.

9. [By Algorithm 2.1, find $\gcd(7, 3)$ and two integers $\hat{\imath}_0, \hat{\jmath}_0$ such that
   $7\hat{\imath}_0 - 3\hat{\jmath}_0 = \gcd(7, 3).$]
   $g \leftarrow 1,$
   $(\hat{\imath}_0, \hat{\jmath}_0) \leftarrow (1, 2).$
   Since $c \bmod g = 0$, continue.

10. $(a_1, b_1, c_1) \leftarrow (a/g, b/g, c/g) = (7, 3, -5).$

**11.** Go to **Case** ($b_1 > 0$).
$\tau_1 \leftarrow \lceil -c_1\hat{\imath}_0/b_1 \rceil = 2$,
$\tau_2 \leftarrow \lfloor (\hat{q} - c_1\hat{\imath}_0)/b_1 \rfloor = 14$.
Go to **Case** ($a_1 > 0$).
$\tau_1 \leftarrow \max(\tau_1, \lceil -c_1\hat{\jmath}_0/a_1 \rceil) = 2$,
$\tau_2 \leftarrow \min(\tau_2, \lfloor (\hat{q} - c_1\hat{\jmath}_0)/a_1 \rfloor) = 6$.
Since $\tau_1 \le \tau_2$, continue.

**12.** $\xi \leftarrow c_1(\hat{\imath}_0 - \hat{\jmath}_0)/(a_1 - b_1) = 5/4$.
[Since $\xi$ is neither an integer, nor does it lie between $\tau_1$ and $\tau_2$, the set $\Psi_0$ remains empty.]

**13.** $\tau_3 \leftarrow \lceil \xi - 1 \rceil = 1$,
$\tau_4 \leftarrow \lfloor \xi + 1 \rfloor = 2$,
$\tau_5 \leftarrow \min(\tau_2, \tau_3) = 1$,
$\tau_6 \leftarrow \max(\tau_1, \tau_4) = 2$.
Go to **Case** ($a_1 > b_1$).
Since $\tau_6 \le \tau_2$, set
$\Psi_1 \leftarrow \{(-5 + 3t, -10 + 7t) : 2 \le t \le 6\}$.

**14.** Since $\Psi_1 \neq \emptyset$, there is a loop-carried dependence of statement $T$ on statement $S$.
The corresponding set of distances is $\{4t - 5 : 2 \le t \le 6\}$.
Since the sets $\Psi_{-1}$ and $\Psi_0$ are empty, there is no loop-carried dependence of $S$ on $T$, nor any loop-independent dependence between $S$ and $T$.

We see that the pairs of iteration points that cause the dependence of $T$ on $S$ form the set

$$\Psi_1 = \{(1,4), (4,11), (7,18), (10,25), (13,32)\}.$$

The corresponding set of pairs of index points is obtained by repeated applications of the relation $I = p + \hat{I}\theta$:

$$\{(15,30), (30,65), (45,100), (60,135), (75,170)\}.$$

Thus, $T(30)$ depends on $S(15)$, $T(65)$ depends on $S(30)$, etc. There are five such pairs of statement instances. The set of dependence

distances is $\{3, 7, 11, 15, 19\}$. (Note that dependence distances are computed from iteration points.)

---

**EXERCISES 2.5**

1. (a) By Algorithm 2.1, find $g = \gcd(10, 14)$ and two integers $x_0$ and $y_0$, such that $10x_0 - 14y_0 = g$. Show all steps. Apply Theorem 2.8 to find the general solution to the equation $10x - 14y = 6$, using these values of $x_0$ and $y_0$.

   (b) By trial and error, find a pair of integers $(x_1, y_1) \neq (x_0, y_0)$ with the same property: $10x_1 - 14y_1 = g$. Solve $10x - 14y = 6$ again, this time using $(x_1, y_1)$ instead of $(x_0, y_0)$ in Theorem 2.8. Show that the general solution obtained here generates the same set of ordered pairs as generated by the general solution in the previous problem.

   (c) Find a formula that will generate *all* integer pairs $(x, y)$ such that $10x - 14y = g$.

2. In Theorem 2.8, show that irrespective of whether $g$ divides $c$ or not, the set of all *real* solutions to Equation (2.9) is given by

$$x = (c/g)x_0 + (b/g)t$$
$$y = (c/g)y_0 + (a/g)t,$$

   where $t$ is a *real* parameter.

3. Write an algorithm such that given three integers $a, b$, and $c$, it decides if there is an integer solution $(x, y) \geq (0, 0)$ (i.e., $x \geq 0$ and $y \geq 0$) to the diophantine equation $ax - by = c$, and gives a formula to generate all such solutions when they exist. Apply your algorithm to three examples to demonstrate that the number of nonnegative solutions could be infinite, finite and positive, or zero.

4. In the context of Theorem 2.9, consider these three conditions:

   (a) $X(aI + a_0)$ and $X(bI + b_0)$ cause a dependence between $S$ and $T$;

   (b) $(b_0 - a_0 + bp - ap)$ is an integral multiple of $\theta \cdot \gcd(a, b)$;

   (c) $(b_0 - a_0)$ is an integral multiple of $\gcd(a, b)$.

   Show that (a) $\Rightarrow$ (b) $\Rightarrow$ (c). (We already know about the first implication.) Give an example where (c) does not imply (b). Find conditions on loop parameters such that (b) $\Leftrightarrow$ (c) for all integers $a, a_0, b$, and $b_0$.

   While (b) is the gcd test for the dependence equation in terms of iteration values, we can think of (c) as the gcd test for the equation $ai - bj = b_0 - a_0$, that is, for the dependence equation in terms of index values. Which is the stronger test for dependence in general: the gcd test in terms of iteration values, or the gcd test in terms of index values? When are they equivalent? (Note that (b) does not involve the final loop limit $q$, and (c) does not involve any loop parameters.)

5. Apply Algorithm 2.2 to the following examples:

(a) $L$: **do** $I = 3, 100, 1$
$S$: $X(9I + 22) = \cdots$
$T$: $\cdots = \cdots X(6I - 17) \cdots$
**enddo**

(b) $L$: **do** $I = 0, 100, 2$
$S$: $X(3I) = \cdots$
$T$: $\cdots = \cdots X(36) \cdots$
**enddo**

(c) $L$: **do** $I = 100, -10, -3$
$S$: $X(4I + 16) = \cdots$
$T$: $\cdots = \cdots X(4I - 4) \cdots$
**enddo**

(d) $L$: **do** $I = 100, -10, -2$
$S$: $X(4I + 16) = \cdots$
$T$: $\cdots = \cdots X(4I - 4) \cdots$
**enddo**

(e) $L$: **do** $I = 100, -10, -1$
$S$: $X(4I + 16) = \cdots$
$T$: $\cdots = \cdots X(4I - 4) \cdots$
**enddo**

(f) $L$: **do** $I = -50, 100, 1$
$S$: $X(-2I + 25) = \cdots$
$T$: $\cdots = \cdots X(3I + 4) \cdots$
**enddo**

(g) $L$: **do** $I = -300, 200, 3$
$S$: $X(6I - 17) = \cdots$
$T$: $\cdots = \cdots X(9I + 22) \cdots$
**enddo**

(h) $L$: **do** $I = 0, 100, 1$
$S$: $X(2I, 2I + 1) = \cdots$
$T$: $\cdots = \cdots X(3I + 1, 3I + 2) \cdots$
**enddo**

6. In Algorithm 2.2, let $a \neq b$. Show that if $\Psi_0$ is nonempty, then it contains the single point $(c/(a - b), c/(a - b))$.

7. In the context of Algorithm 2.2, give examples such that of the three sets $\Psi_1, \Psi_{-1}$, and $\Psi_0$,
   (a) only one is nonempty (three examples, one for each set);
   (b) only two are nonempty (three examples, one for each combination);
   (c) all three are nonempty (one example).

8. Write a simpler version of Algorithm 2.2 for the case where $b = -a \neq 0$.

9. Consider the program

```
L:    do I = p, q, θ
  S:      x = ···
  T:      ··· = ··· x ···
      enddo
```

where $x$ is a scalar. Can Algorithm 2.2 handle this dependence problem?

## 2.6 Method of Bounds

As we shall see, the dependence problem becomes more difficult when we move from a single loop to a more complicated program. The major difficulty arises from the fact that the general solution to the dependence equation usually contains more than one integer parameter, and therefore we get a system of inequalities with more than one integer variable when the solution is substituted in the dependence constraints (Step 4, Algorithm 1.1). Such a system of inequalities is not trivial to solve. We mentioned in Chapter 1 that for this reason, an approximate algorithm (Algorthm 1.2) is often used, where we test if there is an integer solution to the dependence equation *without* any constraints, and a real solution to the equation *with* the dependence constraints. We also discussed two implementations of that algorithm: the method of bounds and the method of elimination. Since the dependence problem involving a one-dimensional array in a single loop is a simple problem, an application of the method of elimination (Algorithm 1.3) to it is uninteresting (why?). On the other hand, by applying the method of bounds to this problem, we get results that can be applied to a more general program. In this section, we introduce the method of bounds in terms of a single loop. In Chapter 6, this method will be studied in detail in the context of a nest of several loops, where it is normally used in practice. The method of bounds originated in [Bane 76].

Consider again the dependence problem between two statements $S$ and $T$ in a loop of the form

```
L:    do I = p, q, θ
          H(I)
      enddo
```

posed by a variable $X(aI + a_0)$ of $S$ and a variable $X(bI + b_0)$ of $T$. Write the dependence equation (2.11)

$$(a\theta)\hat{\imath} - (b\theta)\hat{\jmath} = b_0 - a_0 + (b - a)p$$

in the form

$$a\hat{\imath} - b\hat{\jmath} = c, \tag{2.15}$$

where $c = [b_0 - a_0 + (b - a)p]/\theta$. Also, rewrite the dependence constraints (2.12):

$$\left.\begin{array}{l} 0 \le \hat{\imath} \le \hat{q} \\ 0 \le \hat{\jmath} \le \hat{q}, \end{array}\right\} \tag{2.16}$$

where $\hat{q} = \lfloor (q - p)/\theta \rfloor$.

We focus on Step 3 of Algorithm 1.2. In the method of bounds, this step is implemented by computing the extreme values of a particular linear function in a particular set. To motivate the following discussion, suppose $T < S$ and that we want to test if $T$ depends on $S$. Then, we want to know if there is a real solution to (2.15), that satisfies (2.16) and the condition $\hat{\imath} \le \hat{\jmath} - 1$. Note that (2.16) and this additional condition together define a right-angled triangle $P$ bounded by the lines: $x = 0, y = \hat{q}, x \le y - 1$. (This will be the triangle in Figure 2.2 if we take $p = 0, q = \hat{q}$, and $\varepsilon = 1$.) Also, the left-hand-side of (2.15) defines a linear function $f : \mathbf{R}^2 \to \mathbf{R}$ such that $f(x, y) = ax - by$. Our problem then is to decide if there is a real solution to the equation $f(x, y) = c$ in the triangle $P$. By Theorem A.6, there is such a solution iff

$$\min_{(x,y)\in P} f(x, y) \le c \le \max_{(x,y)\in P} f(x, y).$$

Thus, we need to compute the extreme values $\min_{(x,y)\in P} f(x, y)$ and $\max_{(x,y)\in P} f(x, y)$, and then apply the above test.

Minimum and maximum values of small sets of numbers can be conveniently expressed in terms of notations representing positive and negative parts of a number. In Section I.3.2, we defined the positive part $a^+$ and the negative part $a^-$ of a real number $a$ by

$$a^+ = \max(a, 0)$$
$$a^- = \max(-a, 0).$$

Thus, we have $3^+ = 3$, $3^- = 0$, $(-3)^+ = 0$, and $(-3)^- = 3$. It is clear that there are several equivalent expressions for $a^+$ and $a^-$:

$$a^+ = \max(0,a) = (|a| + a)/2 = \begin{cases} a & \text{if } a \geq 0 \\ 0 & \text{if } a < 0, \end{cases}$$

$$a^- = -\min(0,a) = (|a| - a)/2 = \begin{cases} 0 & \text{if } a \geq 0 \\ |a| & \text{if } a < 0. \end{cases}$$

A number of elementary facts on positive and negative parts are stated in Lemma I.3.1.

The following simple results are used in the proof of Theorem 2.11 on which the method of bounds is based.

**Lemma 2.10** *For any two real numbers $a$ and $b$, we have*

$$\min(a,b) = b - (a-b)^-$$
$$\max(a,b) = b + (a-b)^+.$$

PROOF. If $a \geq b$, then we have

$$b - (a-b)^- = b - 0 = b = \min(a,b),$$
$$b + (a-b)^+ = b + (a-b) = a = \max(a,b).$$

If $a < b$, then we have

$$b - (a-b)^- = b - (b-a) = a = \min(a,b),$$
$$b + (a-b)^+ = b + 0 = b = \max(a,b).$$ □

**Theorem 2.11** *Let $a, b, p, q$, and $\varepsilon$ denote real numbers such that $p \leq q$ and $0 \leq \varepsilon \leq q - p$. The minimum and maximum values of the function $f : (x,y) \mapsto ax + by$ in the right-angled triangle $P$, defined by the inequalities:*

$$\left.\begin{aligned} p \leq x \leq q \\ p \leq y \leq q \\ x \leq y - \varepsilon, \end{aligned}\right\} \tag{2.17}$$

*are given by*

$$\min_{(x,y)\in P} f(x,y) = (a+b)p + b\varepsilon - (q-p-\varepsilon)(a^- - b)^+,$$
$$\max_{(x,y)\in P} f(x,y) = (a+b)p + b\varepsilon + (q-p-\varepsilon)(a^+ + b)^+.$$

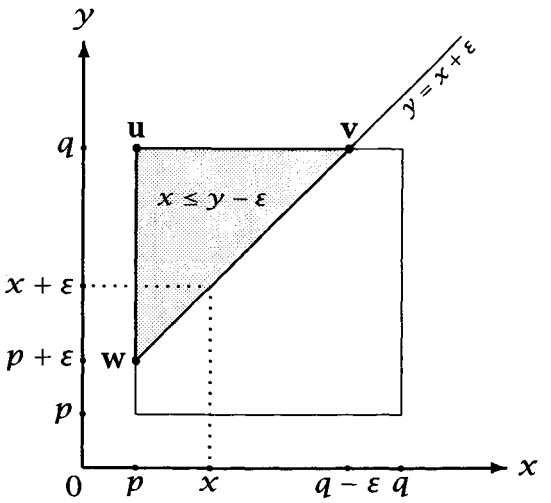


Figure 2.2: The triangle $P$ of Theorem 2.11.

PROOF. In Figure 2.2, denote the three vertices of the triangle under consideration by $\mathbf{u}, \mathbf{v}$, and $\mathbf{w}$ such that

$$\mathbf{u} = (p, q), \qquad \mathbf{v} = (q - \varepsilon, q), \qquad \mathbf{w} = (p, p + \varepsilon).$$

It follows from a well-known result in the theory of linear programming (Theorem A.5) that the extreme values of $f$ are among the values $f(\mathbf{u})$, $f(\mathbf{v})$, and $f(\mathbf{w})$. We can write

$$\begin{aligned} f(\mathbf{u}) &= ap + bq &&= (a+b)p + b\varepsilon + (q - p - \varepsilon) \cdot b, \\ f(\mathbf{v}) &= a(q - \varepsilon) + bq &&= (a+b)p + b\varepsilon + (q - p - \varepsilon) \cdot (a + b), \\ f(\mathbf{w}) &= ap + b(p + \varepsilon) &&= (a+b)p + b\varepsilon + (q - p - \varepsilon) \cdot 0. \end{aligned}$$

Hence, we have

$$\min_{(x,y)\in P} f(x, y) = (a+b)p + b\varepsilon + (q - p - \varepsilon) \min(b, a + b, 0),$$

$$\max_{(x,y)\in P} f(x, y) = (a+b)p + b\varepsilon + (q - p - \varepsilon) \max(b, a + b, 0).$$

The expressions in the theorem follow since

$$\begin{aligned} \min(b, a + b, 0) &= \min(\min(a + b, b), 0) \\ &= \min(b - a^-, 0) && \text{by Lemma 2.10} \\ &= -(b - a^-)^- && \text{by Lemma 2.10} \\ &= -(a^- - b)^+ && \text{by Lemma I.3.1(c),} \end{aligned}$$

and

$$\begin{aligned}\max(b, a+b, 0) &= \max(\max(a+b, b), 0) \\ &= \max(b + a^+, 0) && \text{by Lemma 2.10} \\ &= (b + a^+)^+. && \text{by Lemma 2.10.} \quad \square\end{aligned}$$

The minimum and maximum values of the function $f$ in $P$, studied in the above theorem, are frequently needed in the method of bounds. We denote them by $\mu(a, b, p, q, \varepsilon)$ and $\nu(a, b, p, q, \varepsilon)$, respectively. The expressions defining $\mu$ and $\nu$ are stated below.

$$\begin{aligned}&\mu(a, b, p, q, \varepsilon) \\ &\quad = (a+b)p + b\varepsilon + (q - p - \varepsilon)\min(b, a+b, 0) && (2.18) \\ &\quad = (a+b)p + b\varepsilon - (q - p - \varepsilon)(a^- - b)^+, && (2.19)\end{aligned}$$

$$\begin{aligned}&\nu(a, b, p, q, \varepsilon) \\ &\quad = (a+b)p + b\varepsilon + (q - p - \varepsilon)\max(b, a+b, 0) && (2.20) \\ &\quad = (a+b)p + b\varepsilon + (q - p - \varepsilon)(a^+ + b)^+. && (2.21)\end{aligned}$$

In Chapter 6, we will show how to express extreme values of functions of several variables in terms of $\mu$ and $\nu$.

**Corollary 1** *The minimum and maximum values of a function of the form $x \mapsto ax$ in a closed interval $p \le x \le q$ are $\mu(a, 0, p, q, 0)$ and $\nu(a, 0, p, q, 0)$, respectively.*

We can now derive necessary conditions for the existence of various kinds of dependence between two statements $S$ and $T$. The important cases are listed in Theorem 2.12.

**Theorem 2.12** (Bounds test) *Suppose that a variable $X(aI + a_0)$ of a statement $S$ and a variable $X(bI + b_0)$ of a statement $T$ cause a dependence between $S$ and $T$ in the loop $L$. Let $\hat{q} = \lfloor (q - p)/\theta \rfloor$ and $c = (b_0 - a_0 + bp - ap)/\theta$.*

(a) *If $S < T$ and $S \,\delta\, T$, then $\mu(a, -b, 0, \hat{q}, 0) \le c \le \nu(a, -b, 0, \hat{q}, 0)$;*

(b) *If $S = T$ and $S \,\delta\, S$, then $\mu(a, -b, 0, \hat{q}, 1) \le c \le \nu(a, -b, 0, \hat{q}, 1)$;*

(c) *If $S < T$ and $T \,\delta\, S$, then $\mu(-b, a, 0, \hat{q}, 1) \le c \le \nu(-b, a, 0, \hat{q}, 1)$.*

PROOF. We prove only Case (a) and leave the other two cases to the reader. Assume that $S < T$, and that the variables $X(aI + a_0)$ of $S$ and $X(bI + b_0)$ of $T$ cause a dependence of $T$ on $S$. As we saw at the beginning of this section, the dependence equation can be written as (2.15) which we restate:

$$a\hat{\imath} - b\hat{\jmath} = c.$$

Since $T$ depends on $S$, there exists a solution to this equation that satisfies the conditions (Theorem 2.6(a)):

$$\left.\begin{aligned} 0 \le \hat{\imath} \le \hat{q} \\ 0 \le \hat{\jmath} \le \hat{q} \\ \hat{\imath} \le \hat{\jmath}. \end{aligned}\right\}$$

Let $f$ denote the real function $(x, y) \mapsto ax - by$, and $P$ the right-angled triangle defined by the lines: $x = 0, y = \hat{q}$, and $x \le y$. Since $c = f(x, y)$ for some point $(x, y) \in P$, it follows that

$$\min_{(x,y)\in P} f(x, y) \le c \le \max_{(x,y)\in P} f(x, y).$$

By definition, $\mu(a, b, p, q, \varepsilon)$ and $\nu(a, b, p, q, \varepsilon)$ are the minimum and maximum values, respectively, of the function $(x, y) \mapsto ax + by$ in the triangle bounded by the lines: $x = p, y = q$, and $x \le y - \varepsilon$. Hence, we have

$$\min_{(x,y)\in P} f(x, y) = \mu(a, -b, 0, \hat{q}, 0)$$
$$\max_{(x,y)\in P} f(x, y) = \nu(a, -b, 0, \hat{q}, 0),$$

so that the following inequalities hold:

$$\mu(a, -b, 0, \hat{q}, 0) \le c \le \nu(a, -b, 0, \hat{q}, 0). \qquad \square$$

When we apply Theorem 2.12 to a given problem, the values of the corresponding $\mu$ and $\nu$ functions are computed by (2.18) and (2.20), or by (2.19) and (2.21). By Theorem A.6, in each case of the theorem, the set of inequalities is necessary *and* sufficient for the existence of a *real* solution to the dependence equation satisfying the dependence constraints ($0 \le \hat{\imath} \le \hat{q}$ and $0 \le \hat{\jmath} \le \hat{q}$) and the additional condition (e.g., $\hat{\imath} \le \hat{\jmath}$).

The major steps of the method of bounds (as applied to the model program of this chapter) are Theorem 2.9 (the gcd test) and Theorem 2.12 (the bounds test). We illustrate an application of this method by an example given below.

---

**Example 2.6** Consider again the loop of Example 2.5:

```
L:        do I = 10,200,5
     S:     X(7I + 2) = ···
     T:     ··· = ··· X(3I + 17) ···
          enddo
```

This time we use the gcd and bounds tests (theorems 2.9 and 2.12) to study possible dependence between statements $S$ and $T$. The parameters for this loop are $p = 10, q = 200$, and $\theta = 5$. Therefore, $\hat{q} = \lfloor (q - p)/\theta \rfloor = 38$.

For the variable $X(7I + 2)$ of $S$ and the variable $X(3I + 17)$ of $T$, we have $a = 7, a_0 = 2, b = 3$, and $b_0 = 17$. Using Algorithm 2.1, we find that $\gcd(7, 3) = 1$, and hence $\theta \cdot \gcd(a, b) = 5$. Compute

$$c_0 = b_0 - a_0 + (b - a)p = -25.$$

Since $c_0$ is an integral multiple of $\theta \cdot \gcd(a, b)$, the gcd test passes. Thus, dependence between $S$ and $T$ is not ruled out.

Consider now the possibility of dependence of $T$ on $S$. Compute $c = c_0/\theta = -5$. The inequalities

$$\mu(a, -b, 0, \hat{q}, 0) \le c \le \nu(a, -b, 0, \hat{q}, 0)$$

of Theorem 2.12(a) become

$$\mu(7, -3, 0, 38, 0) \le c \le \nu(7, -3, 0, 38, 0).$$

We compute the values of the $\mu$ and $\nu$ functions using definitions (2.19) and (2.21), and get the inequalities $-114 \le -5 \le 152$ which obviously hold. According to Theorem 2.12, a dependence of $T$ on $S$ may exist. We know from our work in Example 2.5 that this is indeed the case.

Next, study the possible dependence of $S$ on $T$. The inequalities of Theorem 2.12(c) become

$$\mu(-3, 7, 0, 38, 1) \le -5 \le \nu(-3, 7, 0, 38, 1),$$

or

$$7 \leq -5 \leq 266,$$

one of which is false. Thus, the bounds test implies that there is no dependence of $S$ on $T$ (reconfirming what we already found in Example 2.5).

---

**Example 2.7** Suppose we are checking dependence by the method of bounds. If either one of the two tests (gcd and bounds) fails, then there is definitely no dependence. If both tests pass, then we assume there is dependence. The situation does arise, albeit rarely, where both tests pass and dependence does not exist. We explore this below in the context of a particular example. Consider the loop

```
L(7):      do I = 0,7,1
      S:     X(7I + 2) = ···
      T:     ··· = ··· X(3I + 17) ···
           enddo
```

Here, the index and iteration spaces are identical and we have $\hat{q} = q = 7$. The dependence equation is

$$7i - 3j = 15, \tag{2.22}$$

where $i$ and $j$ denote index (or iteration) points. We study possible dependence of $T$ on $S$.

The gcd test (Theorem 2.9) is not conclusive, since gcd(7, 3) is 1. Consider the bounds test. The inequalities of Theorem 2.12(a) are

$$\mu(7, -3, 0, 7, 0) \leq 15 \leq \nu(7, -3, 0, 7, 0),$$

or

$$-21 \leq 15 \leq 28,$$

which clearly hold. Thus, the bounds test is also inconclusive. If we rely on these two tests alone, then we must assume dependence to be on the safe side. However, by unrolling the given loop, we get the sequence of statement instances shown in Figure 2.3. It is clear that there is no dependence of statement $T$ on statement $S$.

Let us now study the role of the size of the iteration space in this context. Consider the loop

$$
\begin{array}{ll}
S(0): & X(2) = \cdots \\
T(0): & \cdots = \cdots X(17) \cdots \\
S(1): & X(9) = \cdots \\
T(1): & \cdots = \cdots X(20) \cdots \\
S(2): & X(16) = \cdots \\
T(2): & \cdots = \cdots X(23) \cdots \\
S(3): & X(23) = \cdots \\
T(3): & \cdots = \cdots X(26) \cdots \\
S(4): & X(30) = \cdots \\
T(4): & \cdots = \cdots X(29) \cdots \\
S(5): & X(37) = \cdots \\
T(5): & \cdots = \cdots X(32) \cdots \\
S(6): & X(44) = \cdots \\
T(6): & \cdots = \cdots X(35) \cdots \\
S(7): & X(51) = \cdots \\
T(7): & \cdots = \cdots X(38) \cdots
\end{array}
$$

Figure 2.3: Loop nest of Example 2.7 after unrolling.

$$
\begin{array}{lll}
L(q): & & \textbf{do } I = 0, q, 1 \\
& S: & \quad X(7I + 2) = \cdots \\
& T: & \quad \cdots = \cdots X(3I + 17) \cdots \\
& & \textbf{enddo}
\end{array}
$$

where $q > 0$ is unspecified for the moment. The solution to Equation (2.22) (obtained by Algorithm 2.1 and Theorem 2.8) is

$$(i, j) = (15 + 3t, 30 + 7t),$$

where $t$ is an integer parameter. Since $i$ and $j$ must satisfy the loop bounds and the inequality $i \leq j$ for dependence of $T$ on $S$, we get the system of inequalities

$$
\left.
\begin{array}{l}
0 \leq 15 + 3t \leq q \\
0 \leq 30 + 7t \leq q \\
\quad 15 + 3t \leq 30 + 7t,
\end{array}
\right\}
$$

that reduces to

$$
\left.
\begin{array}{l}
-5 \leq t \leq (q - 15)/3 \\
-4 \leq t \leq (q - 30)/7 \\
-3 \leq t,
\end{array}
\right\}
$$

which is equivalent to

$$-3 \le t \le \lfloor \min((q-15)/3, (q-30)/7) \rfloor.$$

There is at least one integer $t$ within these bounds for each integral value of $q$ greater than or equal to 9, and there is no such $t$ for $q = 1, 2, \ldots, 8$.

The inequalities of Theorem 2.12(a) for the loop $L(q)$ are

$$\mu(7, -3, 0, q, 0) \le 15 \le \nu(7, -3, 0, q, 0),$$

or

$$-3q \le 15 \le 4q.$$

They hold for all values of $q$ greater than or equal to 4, and fail to hold for $q = 1, 2, 3$.

Thus, if the loop $L(q)$ has more than 9 iterations ($q \ge 9$) or fewer than 5 iterations ($q \le 3$), then the bounds test combined with the gcd test will accurately predict dependence or the lack thereof. Only for the values $q = 4, 5, 6, 7, 8$, the combination of these two tests will make a wrong prediction. See Exercise 16.

---

**EXERCISES 2.6**

1. Show that for any real $a$

   (a) $(\varepsilon a)^+ = \varepsilon a^+$ and $(\varepsilon a)^- = \varepsilon a^-$, when $\varepsilon \ge 0$;

   (b) $(\varepsilon a)^+ = |\varepsilon| a^-$ and $(\varepsilon a)^- = |\varepsilon| a^+$, when $\varepsilon < 0$.

2. Prove that $(a+b)^+ \le a^+ + b^+$. Under what conditions, do we get equality? State and prove similar results for $(a+b)^-$.

3. Show that the identity

   $$(a^- + b)^+ = b^+ + (a + b^-)^-$$

   holds for any two real numbers $a$ and $b$.

4. Express $a^+$ and $a^-$ in terms of the $\mu$ and $\nu$ functions.

5. Prove that $\nu(a, b, p, q, \varepsilon) = -\mu(-a, -b, p, q, \varepsilon)$.

6. Prove that $\mu(a, b, p, q, \varepsilon) = \nu(a, b, p, q, \varepsilon)$ iff either $q = p + \varepsilon$, or $a = b = 0$.

7. Instead of the functions $\mu$ and $\nu$ of five variables, we can use similar functions of three variables. Let $\mu_0(a, b, q)$ and $\nu_0(a, b, q)$ denote the minimum and maximum values of the function $ax + by$ in the triangle defined by the inequalities $0 \le x \le y \le q$. Express $\mu$ and $\nu$ in terms of $\mu_0$ and $\nu_0$.

8. In Theorem 2.11, give a direct proof (without using Theorem A.5) that the extreme values of $f$ in $P$ are attained at vertices of $P$.
9. Find the minimum and maximum values of the functions: $2x+3y$, $2x-3y$, $-2x+3y$, and $-2x-3y$ in each of the following sets:
   (a) $\{(x,y): 1 \le x \le y-2 \le 7\}$,
   (b) $\{(x,y): 1 \le x \le 9, 1 \le y \le 9\}$,
   (c) $\{(x,y): 1 \le y \le x \le 9\}$,
   (d) $\{(x,y): 1 \le x \le 9, 1 \le y \le 9, x \ge y-2\}$.
10. Derive Lemma I.3.2 from Corollary 1 to Theorem 2.11.
11. Prove cases (b) and (c) of Theorem 2.12.
12. As in Theorem 2.12, find a set of necessary conditions for the following cases:
    (a) $T$ has a loop-carried dependence on $S$;
    (b) $S$ has a loop-carried dependence on $T$;
    (c) $S < T$ and $T$ has a loop-independent dependence on $S$;
    (d) $T < S$ and $S$ has a loop-independent dependence on $T$.
13. In Theorem 2.12, assume $\theta > 0$. Show that the conditions in the three cases can be written as follows:
    (a) If $S < T$ and $S \,\delta\, T$, then
    $\mu(a,-b,p,p+\theta\hat{q},0) \le c \le \nu(a,-b,p,p+\theta\hat{q},0)$;
    (b) If $S = T$ and $S \,\delta\, S$, then
    $\mu(a,-b,p,p+\theta\hat{q},\theta) \le c \le \nu(a,-b,p,p+\theta\hat{q},\theta)$;
    (c) If $S < T$ and $T \,\delta\, S$, then
    $\mu(-b,a,p,p+\theta\hat{q},\theta) \le c \le \nu(-b,a,p,p+\theta\hat{q},\theta)$.

    Write down the conditions when $\theta = 1$.
    Next, assume $\theta < 0$. How do the conditions change?
14. Apply the method of bounds (gcd and bounds tests) to check for dependence of $T$ on $S$, and of $S$ on $T$ in loops (a)-(g) of Exercise 2.5.5.
15. Solve the dependence problem in the loop

```
L:    do I = 0,10,1
  S:      X(2I + 34) = ···
  T:      ··· = ··· X(-3I + 35) ···
        enddo
```

    in two different ways: by Algorithm 2.2, and by the method of bounds. Compare your findings.
16. In Example 2.7, find the values of $q$ for which the method of bounds makes a wrong prediction for the following dependences:
    (a) Loop-carried dependence of $T$ on $S$,
    (b) Loop-carried dependence of $S$ on $T$.

# Chapter 3

# Double Loops

## 3.1 Introduction

In Chapter 2, we introduced the main concepts and techniques of dependence analysis in terms of a single loop. In Chapter 4, we develop a formal structure for doing dependence analysis in a perfect loop nest, and then go on to study general programs and general methods in later chapters. The current chapter forms a bridge between Chapter 2 and the rest of the book. Here, we study the major changes that happen as we move from a single loop to a more general program: index and iteration variables become index and iteration *vectors*; dependence distances become dependence (distance) *vectors*; and there is usually more than one unknown integer parameter in the general solution of the dependence equation(s). The main difficulty we face now is that we are required to seek integer solutions to a system of linear inequalities in more than one variable (Step 5, Algorithm 1.1).

In Section 3.2, we study the index and iteration spaces of a double loop. In Section 3.3, we see how dependence concepts are generalized from a single to a double loop. Finally, dependence problems posed by elements of a one- or two-dimensional array are studied in Section 3.4. We do not present any formal results there, but through a series of examples, show how some of the general dependence testing methods work in a double loop. Concepts and results of this chapter will be subsumed in the next chapter on perfect loop nests. However, in Chapter 4, we use a compact matrix notation, while the treatment

in this chapter is still mostly in terms of scalars. The idea here is to postpone the new notation, to allow the reader some time to understand the difficulties that arise as we generalize our program model from a single loop.

## 3.2 Index and Iteration Spaces

The model program for this chapter is a double loop $(L_1, L_2)$ of the form

```
L1 :    do I1 = p1, q1, θ1
L2 :       do I2 = p2, q2, θ2
              H(I1, I2)
           enddo
        enddo
```

where $H(I_1, I_2)$ is a sequence of assignment statements. The *index vector* of $(L_1, L_2)$ is $(I_1, I_2)$, and the values of $(I_1, I_2)$ are the *index points* or *index values* of the loop nest. The set of all index points is the *index space.* Let $\hat{I}_1$ and $\hat{I}_2$ denote the iteration variables of the loops $L_1$ and $L_2$, respectively. Then, $(\hat{I}_1, \hat{I}_2)$ is the *iteration vector* of $(L_1, L_2)$. The values of $(\hat{I}_1, \hat{I}_2)$ are the *iteration points* or *iteration values*, and the set of all iteration points is the *iteration space.* The index and iteration spaces are finite subsets of $\mathbf{Z}^2$ with the same number of points.

We assume that the loop parameters are integer-valued; $p_1$, $q_1$, $\theta_1$, and $\theta_2$ are constants; $\theta_1$ and $\theta_2$ are nonzero; and $p_2$ and $q_2$ are affine functions of $I_1$. Let

$$p_2 = p_{20} + p_{21} I_1 \tag{3.1}$$

$$q_2 = q_{20} + q_{21} I_1, \tag{3.2}$$

where the coefficients are integer constants. The following theorem describes the structures of the index and iteration spaces.

**Theorem 3.1** *The iteration space of the double loop* $(L_1, L_2)$ *consists of all integer vectors* $(\hat{I}_1, \hat{I}_2)$ *such that*

$$0 \le \hat{I}_1 \le \hat{q}_1 \tag{3.3}$$

$$0 \le \hat{I}_2 \le \hat{q}_{20} + \hat{q}_{21} \hat{I}_1, \tag{3.4}$$

*where*

$$\hat{q}_1 = \lfloor (q_1 - p_1)/\theta_1 \rfloor \tag{3.5}$$

$$\hat{q}_{20} = [(q_{20} - p_{20}) + (q_{21} - p_{21})p_1]/\theta_2 \tag{3.6}$$

$$\hat{q}_{21} = (q_{21} - p_{21})\theta_1/\theta_2. \tag{3.7}$$

*The index space of* $(L_1, L_2)$ *is the set of all integer vectors* $(I_1, I_2)$ *given by*

$$I_1 = p_1 + \theta_1 \hat{I}_1 \tag{3.8}$$

$$I_2 = (p_{20} + p_{21}p_1) + (p_{21}\theta_1)\hat{I}_1 + \theta_2 \hat{I}_2, \tag{3.9}$$

*where* $(\hat{I}_1, \hat{I}_2)$ *satisfies (3.3)-(3.4). In the sequential execution of* $(L_1, L_2)$, *the index space is traversed in the direction of lexicographically increasing* $(\hat{I}_1, \hat{I}_2)$.

PROOF. The proof is derived from results on single loops described in Chapter 2. The index variables of $L_1$ and $L_2$ are given in terms of the corresponding iteration variables by (2.3):

$$I_1 = p_1 + \theta_1 \hat{I}_1 \tag{3.10}$$

$$I_2 = p_2 + \theta_2 \hat{I}_2. \tag{3.11}$$

By Theorem 2.2, we have

$$0 \le \hat{I}_1 \le (q_1 - p_1)/\theta_1 \tag{3.12}$$

$$0 \le \hat{I}_2 \le (q_2 - p_2)/\theta_2. \tag{3.13}$$

Since

$$\begin{aligned} q_2 - p_2 &= (q_{20} + q_{21}I_1) - (p_{20} + p_{21}I_1) && \text{by (3.1)-(3.2)} \\ &= (q_{20} - p_{20}) + (q_{21} - p_{21})I_1 \\ &= (q_{20} - p_{20}) + (q_{21} - p_{21})(p_1 + \theta_1 \hat{I}_1) && \text{by (3.10)} \\ &= [(q_{20} - p_{20}) + (q_{21} - p_{21})p_1] + (q_{21} - p_{21})\theta_1 \hat{I}_1, \end{aligned}$$

we can write (3.13) as

$$0 \le \hat{I}_2 \le \hat{q}_{20} + \hat{q}_{21}\hat{I}_1.$$

These inequalities give the range of $\hat{I}_2$ for a given value of $\hat{I}_1$, and the range of $\hat{I}_1$ is given by (3.12). This proves the first part of the theorem.

For each value of $(I_1, I_2)$, there is a unique value of $(\hat{I}_1, \hat{I}_2)$, and conversely. We can write

$$\begin{aligned} I_2 &= p_2 + \theta_2 \hat{I}_2 && \text{by (3.11)} \\ &= (p_{20} + p_{21} I_1) + \theta_2 \hat{I}_2 && \text{by (3.1)} \\ &= p_{20} + p_{21}(p_1 + \theta_1 \hat{I}_1) + \theta_2 \hat{I}_2 && \text{by (3.10)} \\ &= (p_{20} + p_{21} p_1) + (p_{21} \theta_1) \hat{I}_1 + \theta_2 \hat{I}_2. \end{aligned}$$

This result and (3.10) give the required description of the index space in terms of $\hat{I}_1$ and $\hat{I}_2$.

The last part of the theorem on the traversal order of the index space of $(L_1, L_2)$ follows from the fact that the index space of a single loop is traversed in the direction of increasing iteration value (Theorem 2.3). Consider any two distinct index points $(i_1, i_2)$ and $(j_1, j_2)$ of $(L_1, L_2)$, and let $(\hat{\imath}_1, \hat{\imath}_2)$ and $(\hat{\jmath}_1, \hat{\jmath}_2)$ denote the corresponding iteration points. In the sequential execution of the program, the iteration of $(L_1, L_2)$ defined by $(I_1, I_2) = (i_1, i_2)$ can come before the iteration defined by $(I_1, I_2) = (j_1, j_2)$ in exactly one of two ways:

1. The iteration of $L_1$ defined by $I_1 = i_1$ comes before the iteration defined by $I_1 = j_1$, that is, $\hat{\imath}_1 < \hat{\jmath}_1$ (Theorem 2.3).

2. The iterations of $L_1$ defined by $I_1 = i_1$ and $I_1 = j_1$ are identical, and the iteration of $L_2$ defined by $I_2 = i_2$ comes before the iteration defined by $I_2 = j_2$; that is, $\hat{\imath}_1 = \hat{\jmath}_1$ and $\hat{\imath}_2 < \hat{\jmath}_2$.

Combining the two sets of conditions, we can say that in the sequential execution of $(L_1, L_2)$, one moves from an index point $(i_1, i_2)$ to an index point $(j_1, j_2)$ iff $(\hat{\imath}_1, \hat{\imath}_2) \prec (\hat{\jmath}_1, \hat{\jmath}_2)$. This completes the proof of the theorem. □

It should be noted that the coefficients $\hat{q}_{20}$ and $\hat{q}_{21}$ are fractions, in general. As pointed out in Chapter 1, we are assuming exact rational arithmetic, so that the question of roundoff errors does not arise.

The double loop $(L_1, L_2)$ is *regular* if $p_{21} = q_{21}$; it is *rectangular* if $p_{21} = q_{21} = 0$. The following corollary deals with the description of the loop nest when both loops are normalized.

**Corollary 1** *A double loop $(L_1, L_2)$ with arbitrary strides can be written as an equivalent double loop $(\hat{L}_1, \hat{L}_2)$ with unit strides. If $(L_1, L_2)$ is regular, then $(\hat{L}_1, \hat{L}_2)$ is rectangular.*

PROOF. Let $\hat{H}(\hat{I}_1, \hat{I}_2)$ denote the image of $H(I_1, I_2)$ when $I_1$ and $I_2$ are replaced by their equivalent expressions in $\hat{I}_1$ and $\hat{I}_2$ from (3.8) and (3.9). Consider the double loop with unit strides

$\hat{L}_1$ : **do** $\hat{I}_1 = 0, \hat{q}_1, 1$
$\hat{L}_2$ : **do** $\hat{I}_2 = 0, \hat{q}_{20} + \hat{q}_{21}\hat{I}_1, 1$
$\hat{H}(\hat{I}_1, \hat{I}_2)$
**enddo**
**enddo**

The bodies of the loop nests $(L_1, L_2)$ and $(\hat{L}_1, \hat{L}_2)$ are labeled by different sets of variables. However, when all loops are unrolled, the two nests give exactly the same sequence of instances of $H(I_1, I_2)$. We leave the details to the reader. (See Exercise 1.)

When $(L_1, L_2)$ is regular, we have $p_{21} = q_{21}$, by definition. Then, $\hat{q}_{21} = 0$ by (3.7), so that the loop nest $(\hat{L}_1, \hat{L}_2)$ is rectangular. □

---

**Example 3.1** Consider the double loop

$L_1$ : **do** $I_1 = 20, 11, -2$
$L_2$ : **do** $I_2 = I_1 + 2, 2I_1 + 1, 3$
$H(I_1, I_2)$
**enddo**
**enddo**

It is clear that $I_1$ takes the values: 20, 18, 16, 14, 12, in that order. The values of $I_2$ for each value of $I_1$ are shown in Table 3.1. We can easily write down the corresponding table of iteration values, since the set of iteration values of any loop is at a one-to-one correspondence with the set of its index values, and the iteration values always come in the sequence: $0, 1, 2, \ldots$. This is Table 3.2 given next.

In Figure 3.1, we show the index space of $(L_1, L_2)$ on the left and the iteration space on the right.[1] The index space is bounded by the lines:

$$I_1 = 12, I_1 = 20, I_2 = I_1 + 2, I_2 = 2I_1 + 1.$$

[1]Different scales are used in the two diagrams.

| $I_1$ | $I_2$ → | | | | | | |
|---|---|---|---|---|---|---|---|
| ↓ 20 | 22 | 25 | 28 | 31 | 34 | 37 | 40 |
| 18 | 20 | 23 | 26 | 29 | 32 | 35 | |
| 16 | 18 | 21 | 24 | 27 | 30 | 33 | |
| 14 | 16 | 19 | 22 | 25 | 28 | | |
| 12 | 14 | 17 | 20 | 23 | | | |

Table 3.1: Index values for loops of Example 3.1.

| $\hat{I}_1$ | $\hat{I}_2$ → | | | | | | |
|---|---|---|---|---|---|---|---|
| ↓ 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | |
| 2 | 0 | 1 | 2 | 3 | 4 | 5 | |
| 3 | 0 | 1 | 2 | 3 | 4 | | |
| 4 | 0 | 1 | 2 | 3 | | | |

Table 3.2: Iteration values for loops of Example 3.1.

It is traversed in the direction:

$$(20,22),\ldots,(20,40),(18,20),\ldots,(18,35),\ldots,(12,14),\ldots,(12,23).$$

The corresponding sequence of iteration points is

$$(0,0),\ldots,(0,6),(1,0),\ldots,(1,5),\ldots,(4,0),\ldots,(4,3).$$

Next, apply Theorem 3.1 and its corollary to this example. We have

$$\begin{aligned} p_1 &= 20, & p_{20} &= 2, & q_{20} &= 1, \\ q_1 &= 11, & p_{21} &= 1, & q_{21} &= 2, \\ \theta_1 &= -2, & & & \theta_2 &= 3. \end{aligned}$$

Using (3.5)-(3.7), we compute:

$$\hat{q}_1 = 4, \quad \hat{q}_{20} = 19/3, \quad \hat{q}_{21} = -2/3.$$

By Theorem 3.1, the iteration space of $(L_1, L_2)$ is the set of points $(\hat{I}_1, \hat{I}_2)$ such that

$$\begin{aligned} 0 &\le \hat{I}_1 \le 4 \\ 0 &\le \hat{I}_2 \le 19/3 - 2\hat{I}_1/3. \end{aligned}$$

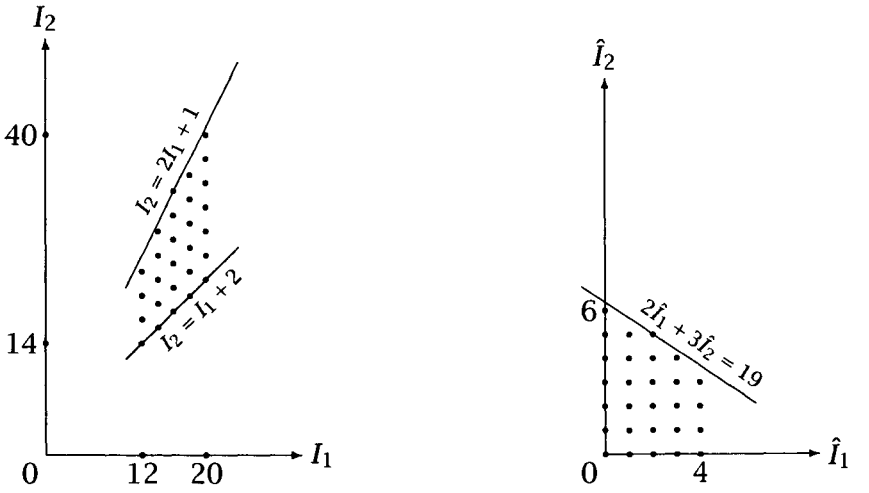


Figure 3.1: Index and iteration spaces for Example 3.1.

Thus, the iteration space is bounded by the lines:

$$\hat{I}_1 = 0, \hat{I}_1 = 4, \hat{I}_2 = 0, 2\hat{I}_1 + 3\hat{I}_2 = 19.$$

The index and iteration spaces are related by equations (3.8)–(3.9):

$$I_1 = 20 - 2\hat{I}_1$$
$$I_2 = 22 - 2\hat{I}_1 + 3\hat{I}_2.$$

When the loops are normalized, we get the equivalent program

$\hat{L}_1$ : **do** $\hat{I}_1 = 0, 4, 1$
$\hat{L}_2$ : **do** $\hat{I}_2 = 0, (19 - 2\hat{I}_1)/3, 1$
$H(20 - 2\hat{I}_1, 22 - 2\hat{I}_1 + 3\hat{I}_2)$
**enddo**
**enddo**

---

**EXERCISES 3.2**

1. Consider a double loop of the form

$L_1$ : **do** $I_1 = 10, 100, 5$
$L_2$ : **do** $I_2 = 10, I_1, 10$
$H(I_1, I_2)$
**enddo**
**enddo**

In the sequential execution of the program, show the order of the instances of $H(I_1, I_2)$, and the corresponding sequences of values of $(I_1, I_2)$ and $(\hat{I}_1, \hat{I}_2)$. Write the equivalent program after loop normalization. Show that in its sequential execution, the same sequence of instances of $H(I_1, I_2)$ is executed.

2. In Example 3.1, express $\hat{I}_1$ and $\hat{I}_2$ in terms of $I_1$ and $I_2$, and find the total number of iteration points.
3. For the double loop $(L_1, L_2)$ in each of the following programs, find
   - descriptions of the iteration and index spaces (as in Theorem 3.1),
   - the total number of iterations,
   - the first five index values and the last in sequential execution,
   - the corresponding iteration values,
   - expressions of the iteration variables in terms of the index variables,
   - an equivalent double loop with unit strides:

(a)
```
L1 :  do I1 = 1, 100, 2
L2 :     do I2 = 100, 1, -3
            H(I1, I2)
         enddo
      enddo
```

(b)
```
L1 :  do I1 = 10, 100, 1
L2 :     do I2 = 2I1 + 1, I1 + 13, 2
            H(I1, I2)
         enddo
      enddo
```

(c)
```
L1 :  do I1 = -110, -200, -5
L2 :     do I2 = 3I1, 20 - I1, 10
            H(I1, I2)
         enddo
      enddo
```

(d)
```
L1 :  do I1 = 10, 100, 2
L2 :     do I2 = 4I1 - 10, 4I1 + 93, 3
            H(I1, I2)
         enddo
      enddo
```

## 3.3 Dependence Concepts

Consider two, not necessarily distinct, assignment statements $S$ and $T$ in the double loop $(L_1, L_2)$ of the previous section. Let $S(i_1, i_2)$ denote the instance of $S$ for an index point $(i_1, i_2)$, and $T(j_1, j_2)$ the

instance of $T$ for an index point $(j_1, j_2)$. The *distance* from $S(i_1, i_2)$ to $T(j_1, j_2)$ is defined to be the vector $(\hat{j}_1 - \hat{i}_1, \hat{j}_2 - \hat{i}_2)$, where $(\hat{i}_1, \hat{i}_2)$ and $(\hat{j}_1, \hat{j}_2)$ are the iteration points corresponding to $(i_1, i_2)$ and $(j_1, j_2)$, respectively.

Lemma 2.4 (on the execution ordering of statement instances for a single loop) is generalized to a double loop by the following result, where the concept of a positive scalar distance is replaced by that of a lexicographically positive distance vector.

**Lemma 3.2** *Consider two assignment statements $S$ and $T$ in the loop nest $(L_1, L_2)$. Let $(d_1, d_2)$ denote the distance from an instance $S(i_1, i_2)$ of $S$ to an instance $T(j_1, j_2)$ of $T$. In the sequential execution of the program, $S(i_1, i_2)$ is executed before $T(j_1, j_2)$ iff one of the following two conditions holds:*

(a) $(d_1, d_2) \succ (0, 0)$,

(b) $(d_1, d_2) = (0, 0)$ *and* $S < T$.

PROOF. Let $(\hat{i}_1, \hat{i}_2)$ and $(\hat{j}_1, \hat{j}_2)$ denote the iteration points corresponding to the index points $(i_1, i_2)$ and $(j_1, j_2)$, respectively. Then, the distance $(d_1, d_2)$ from $S(i_1, i_2)$ to $T(j_1, j_2)$ is given by

$$(d_1, d_2) = (\hat{j}_1 - \hat{i}_1, \hat{j}_2 - \hat{i}_2).$$

Since in the sequential execution of $(L_1, L_2)$, the index space is traversed in the direction of increasing iteration value (Theorem 3.1), $H(i_1, i_2)$ is executed before $H(j_1, j_2)$ iff $(\hat{i}_1, \hat{i}_2) \prec (\hat{j}_1, \hat{j}_2)$, that is, iff $(d_1, d_2) \succ (0, 0)$.

To prove the *only if* part, assume that $S(i_1, i_2)$ is executed before $T(j_1, j_2)$. The instances $S(i_1, i_2)$ and $T(j_1, j_2)$ are in $H(i_1, i_2)$ and $H(j_1, j_2)$, respectively. If $(i_1, i_2) \neq (j_1, j_2)$, then $H(i_1, i_2)$ is executed before $H(j_1, j_2)$, and therefore $(d_1, d_2) \succ (0, 0)$. If $(i_1, i_2) = (j_1, j_2)$, then $(d_1, d_2) = (0, 0)$, and we must have $S < T$ for the instance of $S$ to precede the instance of $T$ in the same iteration of the double loop.

The proof of the *if* part is similar and is omitted. □

The relation of *dependence* between statements is denoted by $\delta$, and is defined as follows: For two statements $S$ and $T$, we have $S \, \delta \, T$ if there is an instance $S(i_1, i_2)$ of $S$, an instance $T(j_1, j_2)$ of $T$, and a memory location $\mathcal{M}$, such that

1. Both $S(i_1, i_2)$ and $T(j_1, j_2)$ reference (read or write) $\mathcal{M}$;
2. $S(i_1, i_2)$ is executed before $T(j_1, j_2)$ in the sequential execution of the program;
3. During sequential execution, the location $\mathcal{M}$ is not written in the time period from the end of execution of $S(i_1, i_2)$ to the beginning of the execution of $T(j_1, j_2)$.

The statements $S$ and $T$ need not be distinct, but Condition 2 requires that the instances $S(i_1, i_2)$ and $T(j_1, j_2)$ be distinct. The notation $S \, \delta \, T$ is read as "$T$ depends on $S$." The *dependence* of $T$ on $S$ is the set of all pairs $(S(i_1, i_2), T(j_1, j_2))$ that satisfy the above conditions. Thus, we have $S \, \delta \, T$ iff the dependence of $T$ on $S$ is nonempty. The graph of the relation $\delta$ is called the *statement dependence graph* of the given program.

Suppose that a statement $T$ depends on a statement $S$. Let $S(i_1, i_2)$ and $T(j_1, j_2)$ denote a pair of instances, such that they (together with some memory location $\mathcal{M}$) satisfy the above conditions. (There is at least one such pair.) We say that the instance $T(j_1, j_2)$ *depends* on the instance $S(i_1, i_2)$. Let $(d_1, d_2)$ denote the distance from $S(i_1, i_2)$ to $T(j_1, j_2)$. Let $(\sigma_1, \sigma_2) = (\text{sig}(d_1), \text{sig}(d_2))$ and $\ell = \text{lev}(d_1, d_2)$. (For the definitions of "sig" and "lev," see Section I.1.3.) In other words,

$$\sigma_1 = \begin{cases} 1 & \text{if } d_1 > 0 \\ -1 & \text{if } d_1 < 0 \\ 0 & \text{if } d_1 = 0, \end{cases}$$

$$\sigma_2 = \begin{cases} 1 & \text{if } d_2 > 0 \\ -1 & \text{if } d_2 < 0 \\ 0 & \text{if } d_2 = 0, \end{cases}$$

$$\ell = \begin{cases} 1 & \text{if } d_1 > 0 \\ 2 & \text{if } d_1 = 0 \text{ and } d_2 > 0 \\ 3 & \text{if } d_1 = 0 \text{ and } d_2 = 0. \end{cases}$$

Then, $(d_1, d_2)$ is a *distance vector*, $(\sigma_1, \sigma_2)$ a *direction vector*, and $\ell$ a *dependence level* for the dependence of $T$ on $S$. It is sometimes convenient to say that $T$ depends on $S$ *with* a distance vector $(d_1, d_2)$ or a direction vector $(\sigma_1, \sigma_2)$, and *at* a level $\ell$. When we do not want to specify the exact value of $\sigma_1$ or $\sigma_2$, we use the symbol "$*$." For

example, "there is a direction vector of the form $(1, *)$" means there is at least one direction vector from the set $\{(1,1),(1,0),(1,-1)\}$.

Since $S(i_1, i_2)$ must be executed before $T(j_1, j_2)$ if $T(j_1, j_2)$ depends on $S(i_1, i_2)$, we have the following theorem as a direct consequence of Lemma 3.2. The next two corollaries follow from the definitions just given.

**Theorem 3.3** *If $(d_1, d_2)$ is a distance vector for the dependence of statement $T$ on statement $S$, then $(d_1, d_2) \succeq (0,0)$. The equality $(d_1, d_2) = (0,0)$ is possible only if $S < T$.*

**Corollary 1** *If $(\sigma_1, \sigma_2)$ is a direction vector for the dependence of statement $T$ on statement $S$, then $(\sigma_1, \sigma_2) \succeq (0,0)$. The equality $(\sigma_1, \sigma_2) = (0,0)$ is possible only if $S < T$.*

**Corollary 2** *If $\ell$ is a dependence level for the dependence of statement $T$ on statement $S$, then $1 \leq \ell \leq 3$. The value $\ell = 3$ is possible only if $S < T$.*

The dependence of $T$ on $S$ can be partitioned into three parts: the parts carried by the two loops and the loop-independent part. The part *carried by* $L_1$ consists of all instance pairs $(S(i_1, i_2), T(j_1, j_2))$, such that $\hat{\imath}_1 < \hat{\jmath}_1$ and $T(j_1, j_2)$ depends on $S(i_1, i_2)$. The part *carried by* $L_2$ consists of all instance pairs $(S(i_1, i_2), T(j_1, j_2))$, such that $\hat{\imath}_1 = \hat{\jmath}_1$, $\hat{\imath}_2 < \hat{\jmath}_2$, and $T(j_1, j_2)$ depends on $S(i_1, i_2)$. The *loop-independent* part consists of all instance pairs $(S(i_1, i_2), T(i_1, i_2))$ such that $T(i_1, i_2)$ depends on $S(i_1, i_2)$. The following statements are equivalent in the context of dependence of a statement $T$ on a statement $S$:

1. The dependence of $T$ on $S$ is carried by the loop $L_1$;
2. There is a pair of instances $S(i_1, i_2)$ and $T(j_1, j_2)$, such that $\hat{\imath}_1 < \hat{\jmath}_1$ and $T(j_1, j_2)$ depends on $S(i_1, i_2)$;
3. There is a dependence distance vector $(d_1, d_2)$ with $d_1 > 0$;
4. There is a dependence direction vector of the form $(1, *)$;
5. $T$ depends on $S$ at level 1.

There are similar sets of equivalent statements for a dependence carried by $L_2$ and a loop-independent dependence (Exercise 1).

As in Section 2.3, we can define flow, anti-, output, and input dependences, indirect dependence, dependence caused by a pair of variables, etc. When needed, we also associate a distance or a direction vector, or a level of dependence with a particular type of dependence, or even with a particular pair of variables.

---

**Example 3.2** In this example, we illustrate some of the concepts introduced in the current section. Consider the program

$L_1$ : **do** $I_1 = 10, 100, 2$
$L_2$ : **do** $I_2 = 16, 11, -2$
$S$ : $X(I_1 + 2, I_2) = \cdots$
$T$ : $\cdots = \cdots X(I_1, I_2 - 4) \cdots$
$U$ : $\cdots = \cdots X(I_1 + 3, I_2 + 2) \cdots$
$V$ : $X(I_1, I_2) = \cdots$
**enddo**
**enddo**

A beginning part of the sequence of statement instances obtained by unrolling the two loops is shown in Table 3.3, where the first column gives iteration points.

It is clear that statement $T$ is flow dependent on statement $S$. For example, the instance $T(12, 16)$ depends on the instance $S(10, 12)$. The iteration points corresponding to the index points $(12, 16)$ and $(10, 12)$, are $(1, 0)$ and $(0, 2)$, respectively. Hence, the distance vector arising from these two instances is $(d_1, d_2) = (1, -2)$, and the direction vector is $(\sigma_1, \sigma_2) = (1, -1)$. It implies that $T$ depends on $S$ at level 1, since $d_1 > 0$. This dependence is carried by the outer loop $L_1$, it is not carried by the inner loop $L_2$, and there is no loop-independent part. The pattern is repeated several times. For example, $T(14, 16)$ depends on $S(12, 12)$, $T(16, 16)$ depends on $S(14, 12)$, and so on. We do not get any new distance or direction vectors.

Statement $V$ is anti-dependent on statement $T$. Indeed, $V(10, 12)$ depends on $T(10, 16)$, $V(12, 12)$ depends on $T(12, 16)$, etc. For this dependence, the unique distance vector, direction vector, and level are $(0, 2)$, $(0, 1)$, and 2, respectively. This dependence is carried by the inner loop $L_2$, but not by $L_1$.

| Iteration Point | Iteration of loop Nest | |
|---|---|---|
| (0,0) | $S(10,16)$ : | $X(12,16) = \cdots$ |
| | $T(10,16)$ : | $\cdots = \cdots X(10,12) \cdots$ |
| | $U(10,16)$ : | $\cdots = \cdots X(13,18) \cdots$ |
| | $V(10,16)$ : | $X(10,16) = \cdots$ |
| (0,1) | $S(10,14)$ : | $X(12,14) = \cdots$ |
| | $T(10,14)$ : | $\cdots = \cdots X(10,10) \cdots$ |
| | $U(10,14)$ : | $\cdots = \cdots X(13,16) \cdots$ |
| | $V(10,14)$ : | $X(10,14) = \cdots$ |
| (0,2) | $S(10,12)$ : | $X(12,12) = \cdots$ |
| | $T(10,12)$ : | $\cdots = \cdots X(10,8) \cdots$ |
| | $U(10,12)$ : | $\cdots = \cdots X(13,14) \cdots$ |
| | $V(10,12)$ : | $X(10,12) = \cdots$ |
| (1,0) | $S(12,16)$ : | $X(14,16) = \cdots$ |
| | $T(12,16)$ : | $\cdots = \cdots X(12,12) \cdots$ |
| | $U(12,16)$ : | $\cdots = \cdots X(15,18) \cdots$ |
| | $V(12,16)$ : | $X(12,16) = \cdots$ |
| (1,1) | $S(12,14)$ : | $X(14,14) = \cdots$ |
| | $T(12,14)$ : | $\cdots = \cdots X(12,10) \cdots$ |
| | $U(12,14)$ : | $\cdots = \cdots X(15,16) \cdots$ |
| | $V(12,14)$ : | $X(12,14) = \cdots$ |
| (1,2) | $S(12,12)$ : | $X(14,12) = \cdots$ |
| | $T(12,12)$ : | $\cdots = \cdots X(12,8) \cdots$ |
| | $U(12,12)$ : | $\cdots = \cdots X(15,14) \cdots$ |
| | $V(12,12)$ : | $X(12,12) = \cdots$ |
| (2,0) | $S(14,16)$ : | $X(16,16) = \cdots$ |
| | $T(14,16)$ : | $\cdots = \cdots X(14,12) \cdots$ |
| | $U(14,16)$ : | $\cdots = \cdots X(17,18) \cdots$ |
| | $V(14,16)$ : | $X(14,16) = \cdots$ |
| (2,1) | $S(14,14)$ : | $X(16,14) = \cdots$ |
| | $T(14,14)$ : | $\cdots = \cdots X(14,10) \cdots$ |
| | $U(14,14)$ : | $\cdots = \cdots X(17,16) \cdots$ |
| | $V(14,14)$ : | $X(14,14) = \cdots$ |
| (2,2) | $S(14,12)$ : | $X(16,12) = \cdots$ |
| | $T(14,12)$ : | $\cdots = \cdots X(14,8) \cdots$ |
| | $U(14,12)$ : | $\cdots = \cdots X(17,14) \cdots$ |
| | $V(14,12)$ : | $X(14,12) = \cdots$ |

Table 3.3: Some iterations of $(L_1, L_2)$ in Example 3.2.

Statement $V$ is output dependent on statement $S$, since $V(12,16)$ depends on $S(10,16)$, $V(12,14)$ depends on $S(10,14)$, etc. In this case, the distance vector is $(1,0)$, the direction vector is also $(1,0)$, and the level of dependence is 1. This dependence is carried by $L_1$, but not by $L_2$.

There is no dependence between statement $U$ and any other statement in the loop nest.

The subscripts for this problem were chosen so that we get simple repeating patterns that are clearly visible from an inspection of the first few statement instances. We do not always have such regularity, and the distance vectors, direction vectors, or levels of dependence need not be unique.

---

**EXERCISES 3.3**

1. Suppose that $T$ depends on $S$ in the model program and the dependence is carried by the loop $L_2$. Write a set of equivalent statements describing this fact in terms of statement instances, distance vectors, direction vectors, and dependence levels. Repeat for a loop-independent dependence of $T$ on $S$.
2. Give the complete dependence picture for the program of Example 3.2, after changing the loop headers to:

   (a) $L_1$: **do** $I_1 = 10,100,3$
   $L_2$: **do** $I_2 = 16,11,-2$

   (b) $L_1$: **do** $I_1 = 10,100,1$
   $L_2$: **do** $I_2 = 16,11,-1$

   (c) $L_1$: **do** $I_1 = 100,10,-2$
   $L_2$: **do** $I_2 = 11,16,2$

   (d) $L_1$: **do** $I_1 = 10,100,2$
   $L_2$: **do** $I_2 = I_1,100,1$
3. Study the dependence structure of the following programs (by inspection):

   (a)
   ```
   L1 :  do I1 = 0,10,1
   L2 :     do I2 = 0,10,1
   S  :        X(I1 + I2 + 2) = ···
   T  :        ··· = ··· X(I1 − I2 + 11) ···
            enddo
         enddo
   ```

   (b)
   ```
   L1 :  do I1 = 4,10,1
   L2 :     do I2 = 4,I1,1
   S  :        X(I1,I2) = X(I1 − 1,I2 − 1) + X(I1 + 1,I2 + 1)
            enddo
         enddo
   ```

## 3.4 Dependence Problems

The goal of this section is to give a taste of the complexity of dependence problems in double loops, and provide an informal introduction, through examples, to the various ways of solving such problems. These solution techniques are derived from general methods which will be discussed in detail in chapters to follow.

---

**Example 3.3** Consider the rectangular double loop

```
L1 :    do I1 = 0, 100, 1
L2 :       do I2 = 0, 50, 1
   S :        X(3I1, 3I2) = ···
   T :        ··· = ··· X(2I1 + 1, I2 + 2) ···
           enddo
        enddo
```

where the two program variables are elements of a two-dimensional array, each of their subscripts has only one index variable, and the corresponding subscripts have the same index variable. We study the dependence between statements $S$ and $T$ caused by $X(3I_1, 3I_2)$ and $X(2I_1+1, I_2+2)$. Note that the index and iteration spaces are identical in this case (why?).

The instance of the variable $X(3I_1, 3I_2)$ for an index point $(i_1, i_2)$ is $X(3i_1, 3i_2)$. The instance of the variable $X(2I_1 + 1, I_2 + 2)$ for an index point $(j_1, j_2)$ is $X(2j_1+1, j_2+2)$. These two instances represent the same memory location iff

$$3i_1 = 2j_1 + 1$$
$$3i_2 = j_2 + 2,$$

that is, iff

$$3i_1 - 2j_1 = 1 \tag{3.14}$$
$$3i_2 - j_2 = 2. \tag{3.15}$$

These are the *dependence equations* for the problem. Since $(i_1, i_2)$ and $(j_1, j_2)$ are index points for the given loop nest, they must satisfy

the following inequalities:

$$\left.\begin{array}{l} 0 \le i_1 \le 100 \\ 0 \le j_1 \le 100 \end{array}\right\} \tag{3.16}$$

$$\left.\begin{array}{l} 0 \le i_2 \le 50 \\ 0 \le j_2 \le 50. \end{array}\right\} \tag{3.17}$$

These are the *dependence constraints* for the problem.[2]

If the variables of $S$ and $T$ cause a dependence between those two statements, then there must exist integers $i_1, j_1, i_2$, and $j_2$ that satisfy the equations (3.14)-(3.15) and the inequalities (3.16)-(3.17).

This particular double-loop problem can be partitioned into two disjoint single-loop problems: one with dependence equation (3.14) and dependence constraints (3.16), the other with dependence equation (3.15) and dependence constraints (3.17). It is an example of what we called a simple problem in Chapter 1. We can solve each subproblem by Algorithm 2.2, and then combine the results to get the solution to the main problem. Instead of mechanically carrying out the steps of Algorithm 2.2, we show here the main steps.

Solving Equation (3.14) by Algorithm 2.1 and Theorem 2.8, we get

$$\left.\begin{array}{l} i_1 = 1 + 2t_1 \\ j_1 = 1 + 3t_1, \end{array}\right\} \tag{3.18}$$

where $t_1$ is an unknown integer parameter. Substituting for $i_1$ and $j_1$ in (3.16), we get the inequalities

$$\begin{array}{l} 0 \le 1 + 2t_1 \le 100 \\ 0 \le 1 + 3t_1 \le 100, \end{array}$$

which simplify to

$$\begin{array}{l} -1/2 \le t_1 \le 99/2 \\ -1/3 \le t_1 \le 33. \end{array}$$

Since $t_1$ is an integer, it will satisfy both sets of inequalities iff

$$0 \le t_1 \le 33. \tag{3.19}$$

[2]Since the stride of each loop is 1, conditions of the form "$(i_1 - 0)/1$ is an integer" are automatically satisfied.

Solving Equation (3.15) by Algorithm 2.1 and Theorem 2.8, we get

$$\left.\begin{aligned} i_2 &= t_2 \\ j_2 &= -2 + 3t_2, \end{aligned}\right\} \tag{3.20}$$

where $t_2$ is an unknown integer parameter. Substituting for $i_2, j_2$ in (3.17), we get the inequalities

$$\begin{aligned} 0 \leq \quad & t_2 \leq 50 \\ 0 \leq -2 + 3t_2 \leq 50, & \end{aligned}$$

which simplify to

$$\begin{aligned} 0 \quad \leq t_2 &\leq 50 \\ 2/3 \leq t_2 &\leq 52/3. \end{aligned}$$

Since $t_2$ is an integer, it will satisfy both sets of inequalities iff

$$1 \leq t_2 \leq 17. \tag{3.21}$$

If we take any integral value of $t_1$ from (3.19) and any integral value of $t_2$ from (3.21), then the corresponding integral values of $i_1, j_1, i_2$, and $j_2$, computed from (3.18) and (3.20), will satisfy the equations (3.14)-(3.15) and the constraints (3.16)-(3.17). Thus, there are integers that satisfy the dependence equations and the dependence constraints. Therefore, there is a dependence between $S$ and $T$.

We can restrict the dependences for which we test by requiring a specific direction vector, and still treat the dependence problem as a simple problem. Suppose we want to know if $T$ depends on $S$ with the direction vector $(1, 1)$. Then, we are looking for a solution $(i_1, j_1, i_2, j_2)$ to equations (3.14)-(3.15) that satisfies (3.16)-(3.17) *and* the conditions: $i_1 < j_1, i_2 < j_2$. From (3.18) we see that $i_1 < j_1$ is possible iff $t_1 > 0$. The range of $t_1$ as given by (3.19) is then further restricted to $1 \leq t_1 \leq 33$. Similarly, it is clear from (3.20) that $i_2 < j_2$ holds iff $t_2 > 1$. The range of $t_2$ given by (3.21) is then reduced to $2 \leq t_2 \leq 17$. Since the ranges of $t_1$ and $t_2$ under the additional conditions are nonempty, statement $T$ does depend on statement $S$ with the direction vector $(1, 1)$. This implies that $T$ depends on $S$ at level 1.

The set of instance pairs $(S(i_1, i_2), T(j_1, j_2))$, such that $T(j_1, j_2)$ depends on $S(i_1, i_2)$ and $i_1 < j_1, i_2 < j_2$, is given by (3.18) and (3.20),

where $1 \le t_1 \le 33$ and $2 \le t_2 \le 17$. The corresponding set of dependence distances is

$$\{(t_1, 2t_2 - 2) : 1 \le t_1 \le 33, 2 \le t_2 \le 17\}.$$

Next, suppose we want to find out if statement $S$ depends on statement $T$ with the direction vector $(1, -1)$. Then, since the index points $(i_1, i_2)$ and $(j_1, j_2)$ are associated with $S$ and $T$, respectively, the additional conditions are $j_1 < i_1$ and $j_2 > i_2$. From (3.18) we see that $j_1 < i_1$ holds iff $t_1 < 0$. However, as (3.19) shows, negative values of $t_1$ are not acceptable. Hence, although there are solutions to (3.15) satisfying (3.17) and $j_2 > i_2$ (as we saw above), there are no solutions to the system of equations (3.14)-(3.15) that satisfy (3.16)-(3.17) and the extra conditions. So, $S$ does not depend on $T$ with the direction vector $(1, -1)$.

---

**Example 3.4** Consider the rectangular double loop

```
L1 :    do I1 = 0, 100, 1
L2 :       do I2 = 0, 50, 1
  S :         X(3I1, 3I2) = ···
  T :         ··· = ··· X(2I2 + 1, I1 + 2) ···
           enddo
        enddo
```

obtained from the program of Example 3.3, by interchanging the index variables in the subscripts of the program variable of statement $T$. The dependence equations in this case are

$$3i_1 - 2j_2 = 1 \tag{3.22}$$

$$3i_2 - j_1 = 2. \tag{3.23}$$

The dependence constraints are unchanged:

$$\left.\begin{array}{l} 0 \le i_1 \le 100 \\ 0 \le j_1 \le 100 \\ 0 \le i_2 \le 50 \\ 0 \le j_2 \le 50. \end{array}\right\} \tag{3.24}$$

Solving Equation (3.22) by Algorithm 2.1 and Theorem 2.8, we get

$$\left.\begin{array}{l} i_1 = 1 + 2t_1 \\ j_2 = 1 + 3t_1, \end{array}\right\} \tag{3.25}$$

where $t_1$ is an unknown integer parameter. Substituting for $i_1$ and $j_2$ in the corresponding inequalities of (3.24), we get

$$\begin{aligned} 0 \le 1 + 2t_1 \le 100 \\ 0 \le 1 + 3t_1 \le 50, \end{aligned}$$

which simplify to

$$\begin{aligned} -1/2 \le t_1 \le 99/2 \\ -1/3 \le t_1 \le 49/3. \end{aligned}$$

Since $t_1$ is an integer, it will satisfy both sets of inequalities iff

$$0 \le t_1 \le 16. \tag{3.26}$$

Solving equation (3.23) by Algorithm 2.1 and Theorem 2.8, we get

$$\left.\begin{aligned} i_2 &= t_2 \\ j_1 &= -2 + 3t_2, \end{aligned}\right\} \tag{3.27}$$

where $t_2$ is an unknown integer parameter. Substituting for $i_2$ and $j_1$ in the corresponding inequalities of (3.24), we get

$$\begin{aligned} 0 \le \quad t_2 \le 50 \\ 0 \le -2 + 3t_2 \le 100, \end{aligned}$$

which give

$$1 \le t_2 \le 34. \tag{3.28}$$

Since the ranges of $t_1$ and $t_2$ are nonempty, there are integers that satisfy the dependence equations and the dependence constraints. Therefore, there is a dependence between $S$ and $T$.

The dependence problem we just solved was again a simple problem. However, when we try to be more specific, the problem gets more complicated. As in Example 3.3, suppose we want to know if statement $T$ depends on statement $S$ with the direction vector $(1, 1)$. Then, we are looking for a solution $(i_1, j_1, i_2, j_2)$ to equations (3.22)–(3.23) that satisfies (3.24) *and* the conditions: $i_1 < j_1$ and $i_2 < j_2$. Note that the last set of conditions is equivalent to $i_1 \le j_1 - 1$ and

$i_2 \le j_2 - 1$, since we are dealing with integers only. Using the expressions for $i_1, j_1, i_2$ and $j_2$ in (3.25) and (3.27), we see that $i_1 \le j_1 - 1$ holds iff

$$4/3 + 2t_1/3 \le t_2, \tag{3.29}$$

and $i_2 \le j_2 - 1$ holds iff

$$t_2 \le 3t_1. \tag{3.30}$$

Thus, statement $T$ depends on statement $S$ with the direction vector $(1,1)$, iff there are integers $t_1$ and $t_2$ such that (3.26), (3.28), (3.29), and (3.30) all hold simultaneously. Using Fourier elimination (Algorithm I.3.2) and remembering that $t_1$ and $t_2$ are integer variables, we get

$$1 \le t_1 \le 16 \tag{3.31}$$

$$\lceil \max(1, (4 + 2t_1)/3) \rceil \le t_2 \le \lfloor \min(34, 3t_1) \rfloor. \tag{3.32}$$

This shows that there are *real* values of $t_1$ and $t_2$ that satisfy the required conditions. To see if there are integers satisfying the same conditions, we may use enumeration. Indeed, there are. Taking $t_1 = 1$, we get $2 \le t_2 \le 3$, so that $(1,2)$ and $(1,3)$ are among the admissible values of $(t_1, t_2)$. The set of instance pairs $(S(i_1, i_2), T(j_1, j_2))$, such that $T(j_1, j_2)$ depends on $S(i_1, i_2)$, $i_1 < j_1$, and $i_2 < j_2$, can be described by (3.25) and (3.27), where $t_1$ and $t_2$ are integers satisfying (3.31) and (3.32).

This example was an illustration of the method of elimination for solving dependence problems (Algorithm 1.3).

---

**Example 3.5** Consider the dependence problem in the double loop

```
L1 :    do I1 = 10, 100, 2
L2 :       do I2 = 100, 10, -3
  S :         X(3I1 + 3I2) = ···
  T :         ··· = ··· X(I1 + 2I2 + 3) ···
           enddo
        enddo
```

involving a one-dimensional array. The instance of $X(3I_1 + 3I_2)$ for an index point $(i_1, i_2)$ is $X(3i_1 + 3i_2)$. The instance of $X(I_1 + 2I_2 + 3)$

for an index point $(j_1, j_2)$ is $X(j_1 + 2j_2 + 3)$. These two instances represent the same memory location iff

$$3i_1 - j_1 + 3i_2 - 2j_2 = 3. \tag{3.33}$$

This equation is the dependence equation in terms of index values. The variables $i_1, j_1, i_2$, and $j_2$ must satisfy the following conditions:

| | |
|---|---|
| $10 \le i_1 \le 100$, | $10 \le i_2 \le 100$, |
| $(i_1 - 10)/2$ is an integer, | $(i_2 - 100)/(-3)$ is an integer, |
| $10 \le j_1 \le 100$, | $10 \le j_2 \le 100$, |
| $(j_1 - 10)/2$ is an integer, | $(j_2 - 100)/(-3)$ is an integer. |

These inequalities are the dependence constraints in terms of index values.

We restate the problem in terms of iteration values using Theorem 3.1. The iteration variables $\hat{I}_1$ and $\hat{I}_2$ of $L_1$ and $L_2$ are given by

$$I_1 = 10 + 2\hat{I}_1$$
$$I_2 = 100 - 3\hat{I}_2,$$

and they satisfy the conditions

$$0 \le \hat{I}_1 \le 45$$
$$0 \le \hat{I}_2 \le 30.$$

Let $\hat{\imath}_1$ and $\hat{\jmath}_1$ denote the values of $\hat{I}_1$ corresponding to the values $i_1$ and $j_1$ of $I_1$, and $\hat{\imath}_2$ and $\hat{\jmath}_2$ the values of $\hat{I}_2$ corresponding to the values $i_2$ and $j_2$ of $I_2$. Then, we have

$$i_1 = 10 + 2\hat{\imath}_1, \qquad i_2 = 100 - 3\hat{\imath}_2,$$
$$j_1 = 10 + 2\hat{\jmath}_1, \qquad j_2 = 100 - 3\hat{\jmath}_2.$$

Substituting for $i_1, j_1, i_2$, and $j_2$ in Equation 3.33 and the associated constraints, we get the equation

$$6\hat{\imath}_1 - 2\hat{\jmath}_1 - 9\hat{\imath}_2 + 6\hat{\jmath}_2 = -117, \tag{3.34}$$

where

$$\left.\begin{array}{ll} 0 \le \hat{\imath}_1 \le 45, & 0 \le \hat{\imath}_2 \le 30, \\ 0 \le \hat{\jmath}_1 \le 45, & 0 \le \hat{\jmath}_2 \le 30. \end{array}\right\} \tag{3.35}$$

In terms of iteration variables, the *dependence equation* is (3.34) and the *dependence constraints* are given by (3.35).

Suppose we want to test if statement $T$ depends on statement $S$ with the direction vector $(1, -1)$. Then, we have to decide if there is an integer solution $(\hat{\imath}_1, \hat{\jmath}_1, \hat{\imath}_2, \hat{\jmath}_2)$ to Equation (3.34) that satisfies (3.35) and the conditions

$$\left.\begin{array}{l} \hat{\imath}_1 \le \hat{\jmath}_1 - 1 \\ \hat{\imath}_2 \ge \hat{\jmath}_2 + 1. \end{array}\right\} \tag{3.36}$$

We solve this problem by the method of bounds (Chapter 1). First, we check to see if (3.34) has any integer solutions at all. By Theorem I.3.5, there are integer solutions to this equation iff $\gcd(6, 2, 9, 6)$ divides 117. (This is called the *gcd test* for this problem.) This gcd can be found by Corollary 1 to Theorem I.3.4. Given the matrix

$$\mathbf{A} = \begin{pmatrix} 6 \\ 2 \\ 9 \\ 6 \end{pmatrix},$$

by Algorithm I.2.2, we can find a $4 \times 4$ unimodular matrix $\mathbf{V}$ and a $4 \times 1$ echelon matrix

$$\mathbf{S} = \begin{pmatrix} s_{11} \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

such that $\mathbf{A} = \mathbf{VS}$. Then, $\gcd(6, 2, 9, 6)$ is given by $|s_{11}|$ which happens to equal 1. For an alternative way, note that

$$\gcd(6, 2, 9, 6) = \gcd(6, \gcd(2, 9, 6)) = \gcd(6, \gcd(2, \gcd(9, 6))),$$

and we can evaluate the last expression by repeated applications of the classical Euclid's algorithm (see [Knut 73]). Algorithm 2.1 may also be used, but it computes more than what we need. In any case, since the gcd here is 1, Equation (3.34) does have integer solutions.

The next step is to see if there is a real solution to (3.34) that satisfies (3.35) and (3.36). Define a function $f : \mathbf{R}^4 \mapsto \mathbf{R}$ by

$$f(x_1, y_1, x_2, y_2) = 6x_1 - 2y_1 - 9x_2 + 6y_2.$$

Let $P$ denote the polytope in $\mathbf{R}^4$ defined by the inequalities (3.35)-(3.36) stated in terms of real variables:

$$\begin{array}{lll} 0 \le x_1 \le 45, & 0 \le x_2 \le 30, & x_1 \le y_1 - 1, \\ 0 \le y_1 \le 45, & 0 \le y_2 \le 30, & y_2 \le x_2 - 1. \end{array}$$

(A polytope is a bounded set described by linear inequalities; see Section A.2.) By Theorem A.6, there is a real solution to the equation

$$f(x_1, y_1, x_2, y_2) = -117$$

in $P$, iff

$$\min_P f(x_1, y_1, x_2, y_2) \le -117 \le \max_P f(x_1, y_1, x_2, y_2).$$

(This is the *bounds test* for this problem.)

We can compute these extreme values of $f$ by Theorem 2.11. We have

$$\begin{aligned} \min_P f(x_1, y_1, x_2, y_2) &= \min_{\substack{0 \le x_1 \le 45 \\ 0 \le y_1 \le 45 \\ x_1 \le y_1 - 1}} (6x_1 - 2y_1) + \min_{\substack{0 \le y_2 \le 30 \\ 0 \le x_2 \le 30 \\ y_2 \le x_2 - 1}} (6y_2 - 9x_2) \\ &= \mu(6, -2, 0, 45, 1) + \mu(6, -9, 0, 30, 1) \\ &= -360 \qquad \text{by (2.19),} \end{aligned}$$

and similarly,

$$\begin{aligned} \max_P f(x_1, y_1, x_2, y_2) &= \nu(6, -2, 0, 45, 1) + \nu(6, -9, 0, 30, 1) \\ &= 165 \qquad \text{by (2.21).} \end{aligned}$$

Since the condition $-360 \le -117 \le 165$ holds, there is a real solution to the equation $f(x_1, y_1, x_2, y_2) = -117$ in $P$, that is, a solution to Equation (3.34) satisfying the conditions (3.35)-(3.36), when the four variables are assumed to be real variables. At this point, we assume that there is probably an integer solution to the dependence equation satisfying all conditions, and therefore, there is probably a dependence of $T$ on $S$ with the direction vector $(1, -1)$.

**EXERCISES 3.4**

1. In the program of

   (a) Example 3.3,

   (b) Example 3.4,

   (c) Example 3.5,

   check for dependence of $T$ on $S$ and of $S$ on $T$, with each possible direction vector not already covered. Find the dependence levels and distance vectors in the first two programs.

# Chapter 4

# Perfect Loop Nests

## 4.1 Introduction

In Chapter 2, we gave a detailed introduction to the concepts and techniques of dependence analysis using a single loop as the model. In Chapter 3, we showed how things can become complicated very quickly when there are two loops to deal with. The model for this chapter is a perfect nest of an arbitrary number of loops. Our goal is to properly formulate the linear dependence problem in the context of such a program. We will not discuss methods of solution in this chapter except in some special cases (Section 4.6) and examples. Matrix notation becomes almost a necessity at this point; without that, we run the risk of drowning in a sea of subscripts. The reader should keep on hand a double loop of the form

$L_1$ : **do** $I_1 = p_{10}, q_{10}, \theta_1$
$L_2$ : **do** $I_2 = p_{20} + p_{21}I_1, q_{20} + q_{21}I_1, \theta_2$
$H(I_1, I_2)$
**enddo**
**enddo**

where $p_{10}, q_{10}, p_{20}, p_{21}, q_{20}$, and $q_{21}$ are integer constants, and try to understand the results and notation of this chapter first in terms of that particular perfect nest. Note that in Chapter 3, we wrote $p_1$ for $p_{10}$, and $q_1$ for $q_{10}$.

In Section 4.2, we study the index and iteration spaces of a perfect loop nest. The matrix notation mentioned above is introduced

in this section. Dependence concepts are explained in Section 4.3; representation of linear array subscripts in terms of matrices is presented in Section 4.4; and the linear dependence problem is described in Section 4.5. Three special cases of the problem in perfect nests are studied in Section 4.6: uniform dependence, two-variable problems, and a general case that includes the first two.

## 4.2 Index and Iteration Spaces

In this chapter, we consider a perfect nest of loops $\mathbf{L} = (L_1, L_2, \ldots, L_m)$ of the form

```
L1 :   do I1 = p1, q1, θ1
L2 :     do I2 = p2, q2, θ2
  ⋮        ⋮
Lm :       do Im = pm, qm, θm
             H(I1, I2, ..., Im)
           enddo
           ⋮
         enddo
       enddo
```

where $H(I_1, I_2, \ldots, I_m)$ is a sequence of assignment statements. We write $\mathbf{I} = (I_1, I_2, \ldots, I_m)$ and call $\mathbf{I}$ the *index vector* of $\mathbf{L}$. The values of $\mathbf{I}$ are the *index points* or *index values* of $\mathbf{L}$, and the set of all index points is the *index space*. Let $\hat{I}_r$ denote the iteration variable of the loop $L_r$, $1 \le r \le m$. We write $\hat{\mathbf{I}} = (\hat{I}_1, \hat{I}_2, \ldots, \hat{I}_m)$ and call $\hat{\mathbf{I}}$ the *iteration vector* of $\mathbf{L}$. The values of $\hat{\mathbf{I}}$ are the *iteration points* or *iteration values* of $\mathbf{L}$, and the set of all iteration points is the *iteration space*. The index and iteration spaces of a loop nest of length $m$ are subsets of $\mathbf{Z}^m$, with the same number of points.

For $1 \le r \le m$, the initial limit $p_r$ and the final limit $q_r$ are assumed to be integer-valued affine functions of $I_1, I_2, \ldots, I_{r-1}$. Let

$$p_r(I_1, I_2, \ldots, I_{r-1}) = p_{r0} + p_{r1}I_1 + p_{r2}I_2 + \cdots + p_{r(r-1)}I_{r-1}$$
$$q_r(I_1, I_2, \ldots, I_{r-1}) = q_{r0} + q_{r1}I_1 + q_{r2}I_2 + \cdots + q_{r(r-1)}I_{r-1},$$

where the coefficients are all integer constants. We define two $m$-vectors $\mathbf{p}_0$ and $\mathbf{q}_0$, and two $m \times m$ upper triangular matrices $\mathbf{P}$ and $\mathbf{Q}$

as follows:

$$\mathbf{p}_0 = (p_{10}, p_{20}, \ldots, p_{m0}),$$
$$\mathbf{q}_0 = (q_{10}, q_{20}, \ldots, q_{m0}),$$

$$\mathbf{P} = \begin{pmatrix} 1 & -p_{21} & -p_{31} & \cdots & -p_{m1} \\ 0 & 1 & -p_{32} & \cdots & -p_{m2} \\ 0 & 0 & 1 & \cdots & -p_{m3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix},$$

$$\mathbf{Q} = \begin{pmatrix} 1 & -q_{21} & -q_{31} & \cdots & -q_{m1} \\ 0 & 1 & -q_{32} & \cdots & -q_{m2} \\ 0 & 0 & 1 & \cdots & -q_{m3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}.$$

The vectors represent the constant parts of the loop limits, and the matrices represent the variable parts. Note that $\mathbf{P}$ and $\mathbf{Q}$ are unimodular matrices with $\det(\mathbf{P}) = \det(\mathbf{Q}) = 1$. For the loop nest $\mathbf{L}$, the *initial vector* is $\mathbf{p}_0$, the *initial matrix* is $\mathbf{P}$, the *final vector* is $\mathbf{q}_0$, and the *final matrix* is $\mathbf{Q}$.

The strides $\theta_r$ are assumed to be nonzero integer constants. The *stride matrix* of $\mathbf{L}$ is denoted by $\Theta$ and is defined by

$$\Theta = \operatorname{diag}(\theta_1, \theta_2, \ldots, \theta_m) \equiv \begin{pmatrix} \theta_1 & 0 & \cdots & 0 \\ 0 & \theta_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \theta_m \end{pmatrix}.$$

Since $\det(\Theta) = \prod_{r=1}^{m} \theta_r \neq 0$ by hypothesis, $\Theta$ is nonsingular.

**Theorem 4.1** *A point* $\mathbf{I} \in \mathbf{Z}^m$ *is an index point for the loop nest* $\mathbf{L}$ *iff*

(a) $(\mathbf{IP} - \mathbf{p}_0)\Theta^{-1}$ *is an integer vector, and*

(b) $\mathbf{I}$ *satisfies the inequalities:*

$$\left.\begin{aligned} \mathbf{p}_0\Theta^{-1} &\leq \mathbf{IP}\Theta^{-1} \\ \mathbf{IQ}\Theta^{-1} &\leq \mathbf{q}_0\Theta^{-1}. \end{aligned}\right\} \tag{4.1}$$

PROOF. By Theorem 2.1, an integer $I_r$ is an index value of the loop $L_r$ iff the following two conditions hold:

1. $(I_r - p_r)/\theta_r$ is an integer, and
2. $I_r$ satisfies the inequalities:

$$0 \le (I_r - p_r)/\theta_r \le (q_r - p_r)/\theta_r. \tag{4.2}$$

For $1 \le r \le m$, we have assumed the following form for the initial limit $p_r \equiv p_r(I_1, I_2, \ldots, I_{r-1})$ of $L_r$:

$$p_r = p_{r0} + p_{r1}I_1 + p_{r2}I_2 + \cdots + p_{r(r-1)}I_{r-1}.$$

Therefore, we can write

$$\begin{aligned} I_r - p_r &= I_r - (p_{r0} + p_{r1}I_1 + p_{r2}I_2 + \cdots + p_{r(r-1)}I_{r(r-1)}) \\ &= (-p_{r1}I_1 - p_{r2}I_2 - \cdots - p_{r(r-1)}I_{r(r-1)} + I_r) - p_{r0} \\ &= (I_1, I_2, \ldots, I_m) \cdot (-p_{r1}, -p_{r2}, \ldots, -p_{r(r-1)}, 1, 0, \ldots, 0) - p_{r0}, \end{aligned}$$

or

$$I_r - p_r = \mathbf{I} \cdot \mathbf{P}^{(r)} - p_{r0} \tag{4.3}$$

where $\mathbf{P}^{(r)}$ denotes column $r$ of the initial matrix $\mathbf{P}$.

Using (4.3), we see that the first condition on $I_r$ is equivalent to requiring that $(\mathbf{I} \cdot \mathbf{P}^{(r)} - p_{r0})/\theta_r$ be an integer. Since this condition has to hold for each $r$ in $1 \le r \le m$, an index point $\mathbf{I}$ must be such that $(\mathbf{IP} - \mathbf{p}_0)\mathbf{\Theta}^{-1}$ is an integer vector.

Again, substituting for $I_r - p_r$ from (4.3) in the left-hand-side inequality of (4.2), we get

$$0 \le (\mathbf{I} \cdot \mathbf{P}^{(r)} - p_{r0})/\theta_r$$

or

$$p_{r0}/\theta_r \le \mathbf{I} \cdot \mathbf{P}^{(r)}/\theta_r.$$

For each $r$ in $1 \le r \le m$, we have an inequality of this form. In matrix notation, the entire set of $m$ inequalities can be written as

$$\mathbf{p}_0\mathbf{\Theta}^{-1} \le \mathbf{IP}\mathbf{\Theta}^{-1}.$$

Next, the right-hand-side inequality of (4.2) is equivalent to

$$(I_r - q_r)/\theta_r \leq 0.$$

We leave it to the reader to show that in matrix notation, the set of $m$ inequalities of this form can be written as

$$\mathbf{IQ}\Theta^{-1} \leq \mathbf{q}_0\Theta^{-1}.$$

That will complete the proof of the theorem. □

Under certain conditions, the index space is completely characterized by the inequalities (4.1), as we see below.

**Corollary 1** *Assume that* $|\theta_r| = 1$ *for each loop* $L_r$. *Then, a point* $\mathbf{I} \in \mathbf{Z}^m$ *is an index point for* $\mathbf{L}$ *iff it satisfies the inequalities (4.1).*

PROOF. Condition (a) of the theorem is redundant in this case. Since $\Theta^{-1}$ is now an integer matrix, $(\mathbf{IP} - \mathbf{p}_0)\Theta^{-1}$ is necessarily an integer vector. □

Note that when the strides $\theta_r$ are all positive, (4.1) can be simplified to the system:

$$\left.\begin{aligned} \mathbf{p}_0 &\leq \mathbf{IP} \\ \mathbf{IQ} &\leq \mathbf{q}_0. \end{aligned}\right\}$$

We derived these inequalities in Section I.5.3 for the case where each loop had a unit stride.

We say that $\mathbf{L}$ is a *regular* nest if $\mathbf{P} = \mathbf{Q}$. A regular nest $\mathbf{L}$ is said to be *rectangular* if the limits have no variable parts, that is, if $p_1, p_2, \ldots, p_m, q_1, q_2, \ldots, q_m$ are constants. For a rectangular nest, we have (remember that $\mathcal{I}_m$ denotes the $m \times m$ identity matrix)

$$\mathbf{p}_0 = (p_1, p_2, \ldots, p_m), \quad \mathbf{q}_0 = (q_1, q_2, \ldots, q_m), \quad \mathbf{P} = \mathbf{Q} = \mathcal{I}_m.$$

The system of inequalities (4.1) simplifies to

$$\mathbf{p}_0\Theta^{-1} \leq \mathbf{IP}\Theta^{-1} \leq \mathbf{q}_0\Theta^{-1} \tag{4.4}$$

in the regular case, and further to

$$\mathbf{p}_0\Theta^{-1} \leq \mathbf{I}\Theta^{-1} \leq \mathbf{q}_0\Theta^{-1} \tag{4.5}$$

in the rectangular case.

**Corollary 2** *The index vector* $\mathbf{I}$ *and the iteration vector* $\hat{\mathbf{I}}$ *of the loop nest* $\mathbf{L}$ *are related by the equation*

$$\mathbf{IP} = \hat{\mathbf{I}}\Theta + \mathbf{p}_0. \tag{4.6}$$

PROOF. It is easy to see that the integer vector in the first condition of Theorem 4.1 is really the iteration vector of the loop nest. In fact, the index and iteration variables of the loop $L_r$ are connected by Equation (2.3):

$$I_r = p_r + \hat{I}_r \theta_r.$$

Using (4.3), we can write

$$\mathbf{I} \cdot \mathbf{P}^{(r)} - p_{r0} = \hat{I}_r \theta_r$$

or

$$\mathbf{I} \cdot \mathbf{P}^{(r)} = \hat{I}_r \theta_r + p_{r0}.$$

The matrix form of this set of $m$ scalar equations is (4.6). □

Since $\mathbf{P}$ and $\Theta$ are nonsingular, we can write

$$\mathbf{I} = (\hat{\mathbf{I}}\Theta + \mathbf{p}_0)\mathbf{P}^{-1} \tag{4.7}$$

and

$$\hat{\mathbf{I}} = (\mathbf{IP} - \mathbf{p}_0)\Theta^{-1}. \tag{4.8}$$

The mapping $\mathbf{I} \mapsto \hat{\mathbf{I}}$ of the index space into the iteration space is clearly bijective; this transformation is called *loop normalization.*

The following theorem gives a description of the iteration space:

**Theorem 4.2** *A point* $\hat{\mathbf{I}} \in \mathbf{Z}^m$ *is an iteration point for the loop nest* $\mathbf{L}$ *iff*

$$\left.\begin{aligned} \mathbf{0} &\le \hat{\mathbf{I}} \\ \hat{\mathbf{I}}\hat{\mathbf{Q}} &\le \hat{\mathbf{q}}_0, \end{aligned}\right\} \tag{4.9}$$

*where*

$$\left.\begin{aligned} \hat{\mathbf{Q}} &= \Theta\mathbf{P}^{-1}\mathbf{Q}\Theta^{-1} \\ \hat{\mathbf{q}}_0 &= (\mathbf{q}_0 - \mathbf{p}_0\mathbf{P}^{-1}\mathbf{Q})\Theta^{-1}. \end{aligned}\right\} \tag{4.10}$$

PROOF. The iteration points are precisely the integer $m$-vectors $\hat{\mathbf{I}}$ that satisfy an equation of the form

$$\mathbf{IP} = \hat{\mathbf{I}}\Theta + \mathbf{p}_0,$$

where $\mathbf{I}$ is an integer $m$-vector such that

$$\left.\begin{aligned} \mathbf{p}_0\Theta^{-1} &\le \mathbf{IP}\Theta^{-1} \\ \mathbf{IQ}\Theta^{-1} &\le \mathbf{q}_0\Theta^{-1}. \end{aligned}\right\}$$

Eliminating $\mathbf{I}$, we get the set of inequalities

$$\left.\begin{aligned} \mathbf{p}_0\Theta^{-1} &\le (\hat{\mathbf{I}}\Theta + \mathbf{p}_0)\Theta^{-1} \\ (\hat{\mathbf{I}}\Theta + \mathbf{p}_0)\mathbf{P}^{-1}\mathbf{Q}\Theta^{-1} &\le \mathbf{q}_0\Theta^{-1}. \end{aligned}\right\}$$

Further simplification yields the set of inequalities

$$\left.\begin{aligned} \mathbf{0} &\le \hat{\mathbf{I}} \\ \hat{\mathbf{I}}(\Theta\mathbf{P}^{-1}\mathbf{Q}\Theta^{-1}) &\le (\mathbf{q}_0 - \mathbf{p}_0\mathbf{P}^{-1}\mathbf{Q})\Theta^{-1}, \end{aligned}\right\}$$

which are exactly the inequalities in (4.9). □

The upper bound for $\hat{I}_1$ can be taken to be an integer to conform with the notation in the single and double-loop cases.

Note that while the iteration space is *equal to* the set of integer points in the polytope in $\mathbf{R}^m$ defined by (4.9), the index space is a *subset* of the set of integer points in the polytope defined by (4.1). An extra condition is needed to exactly identify the index space, namely that $(\mathbf{IP} - \mathbf{p}_0)\Theta^{-1}$ is an integer vector. For this reason, it is easier to deal with the iteration space rather than with the index space in dependence analysis, even though they are both finite sets with the same number of points.

The matrix $\hat{\mathbf{Q}}$ is an $m \times m$ upper-triangular rational matrix with $(1, 1, \ldots, 1)$ as the main diagonal (Exercise 2), and $\hat{\mathbf{q}}_0$ is a rational $m$-vector. We sometimes refer to $\hat{\mathbf{q}}_0$ and $\hat{\mathbf{Q}}$ as the *normalized* final vector and matrix, respectively.

**Theorem 4.3** *In the sequential execution of the loop nest* $\mathbf{L}$*, the index space is traversed in the direction of (lexicographically) increasing* $\hat{\mathbf{I}}$*.*

*Loop normalization produces a loop nest equivalent to* $\mathbf{L}$ *where the loops have zero initial limits and unit strides.*

PROOF. The first part states that an iteration $H(\mathbf{i})$ of the loop nest is executed before another iteration $H(\mathbf{j})$, iff the corresponding iteration values satisfy $\hat{\imath} \prec \hat{\jmath}$. We can prove it in the same way we proved the double-loop version of this fact in the proof of Theorem 3.1.

For the second part, let $\hat{\mathbf{q}}_0 = (\hat{q}_{10}, \hat{q}_{20}, \ldots, \hat{q}_{m0})$ and

$$\hat{\mathbf{Q}} = \begin{pmatrix} 1 & -\hat{q}_{21} & -\hat{q}_{31} & \cdots & -\hat{q}_{m1} \\ 0 & 1 & -\hat{q}_{32} & \cdots & -\hat{q}_{m2} \\ 0 & 0 & 1 & \cdots & -\hat{q}_{m3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}.$$

Then, the given loop nest is equivalent to the program

$\hat{L}_1$ : **do** $\hat{I}_1 = 0, \hat{q}_{10}, 1$
$\hat{L}_2$ : **do** $\hat{I}_2 = 0, \hat{q}_{20} + \hat{q}_{21}\hat{I}_1, 1$
$\vdots$ $\vdots$
$\hat{L}_m$ : **do** $\hat{I}_m = 0, \hat{q}_{m0} + \hat{q}_{m1}\hat{I}_1 + \cdots + \hat{q}_{m(m-1)}\hat{I}_{m-1}, 1$
$H((\hat{\mathbf{I}}\Theta + \mathbf{p}_0)\mathbf{P}^{-1})$
**enddo**
$\vdots$
**enddo**
**enddo**

since the sequence of iterations of $(\hat{L}_1, \hat{L}_2, \ldots, \hat{L}_m)$ is the same as that of $(L_1, L_2, \ldots, L_m)$ in the sequential execution of the two loop nests. We leave the details to the reader. □

**Corollary 1** *Loop normalization converts a regular loop nest into a rectangular loop nest.*

PROOF. We show that the loop nest $(\hat{L}_1, \hat{L}_2, \ldots, \hat{L}_m)$ is rectangular if the loop nest $\mathbf{L}$ is regular. Indeed, if $\mathbf{L}$ is regular, then we have $\mathbf{P} = \mathbf{Q}$. That implies

$$\hat{\mathbf{Q}} = \Theta\mathbf{P}^{-1}\mathbf{Q}\Theta^{-1} = \Theta\mathbf{P}^{-1}\mathbf{P}\Theta^{-1} = \Theta\mathcal{I}_m\Theta^{-1} = \mathcal{I}_m,$$

and

$$\begin{aligned} \hat{\mathbf{q}}_0 &= (\mathbf{q}_0 - \mathbf{p}_0\mathbf{P}^{-1}\mathbf{P})\Theta^{-1} \\ &= (\mathbf{q}_0 - \mathbf{p}_0)\Theta^{-1} \\ &= ((q_{10} - p_{10})/\theta_1, (q_{20} - p_{20})/\theta_2, \ldots, (q_{m0} - p_{m0})/\theta_m). \end{aligned} \quad (4.11)$$

The inequalities (4.9) then become

$$\mathbf{0} \le \hat{\mathbf{I}} \le ((q_{10} - p_{10})/\theta_1, (q_{20} - p_{20})/\theta_2, \ldots, (q_{m0} - p_{m0})/\theta_m),$$

or

$$0 \le \hat{I}_r \le (q_{r0} - p_{r0})/\theta_r \qquad (1 \le r \le m). \tag{4.12}$$

Thus, the loop $\hat{L}_r$ has constant limits 0 and $(q_{r0} - p_{r0})/\theta_r$. □

---

**Example 4.1** Consider the program

```
L1 :    do I1 = 6, 101, 2
L2 :       do I2 = I1 + 10, I1 + 217, 3
L3 :          do I3 = I1 + 3I2 − 2, 2I1 − I2 + 1, −2
                 H(I1, I2, I3)
              enddo
           enddo
        enddo
```

where $H$ denotes a sequence of assignment statements. Here we have

$$\begin{aligned}
p_1 &= p_{10} = 6,\\
q_1 &= q_{10} = 101,\\
p_2 &= p_{20} + p_{21}I_1 = 10 + 1 \times I_1,\\
q_2 &= q_{20} + q_{21}I_1 = 217 + 1 \times I_1,\\
p_3 &= p_{30} + p_{31}I_1 + p_{32}I_2 = -2 + 1 \times I_1 + 3I_2,\\
q_3 &= q_{30} + q_{31}I_1 + q_{32}I_2 = 1 + 2I_1 - 1 \times I_2,\\
\theta_1 &= 2, \theta_2 = 3, \theta_3 = -2.
\end{aligned}$$

Thus, for this loop nest, the initial and final vectors are

$$\begin{aligned}
\mathbf{p}_0 &= (p_{10}, p_{20}, p_{30}) = (6, 10, -2),\\
\mathbf{q}_0 &= (q_{10}, q_{20}, q_{30}) = (101, 217, 1);
\end{aligned}$$

and the initial, final, and stride matrices are

$$\mathbf{P} = \begin{pmatrix} 1 & -p_{21} & -p_{31} \\ 0 & 1 & -p_{32} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & -1 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\mathbf{Q} = \begin{pmatrix} 1 & -q_{21} & -q_{31} \\ 0 & 1 & -q_{32} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\Theta = \begin{pmatrix} \theta_1 & 0 & 0 \\ 0 & \theta_2 & 0 \\ 0 & 0 & \theta_3 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -2 \end{pmatrix}.$$

We compute the inverses of $\Theta$ and $\mathbf{P}$:

$$\Theta^{-1} = \begin{pmatrix} 1/2 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & -1/2 \end{pmatrix}, \qquad \mathbf{P}^{-1} = \begin{pmatrix} 1 & 1 & 4 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix},$$

the normalized final vector

$$\hat{\mathbf{q}}_0 = (\mathbf{q}_0 - \mathbf{p}_0\mathbf{P}^{-1}\mathbf{Q})\Theta^{-1} = (95/2, 207/3, 55/2),$$

and the normalized final matrix

$$\hat{\mathbf{Q}} = \Theta\mathbf{P}^{-1}\mathbf{Q}\Theta^{-1} = \begin{pmatrix} 1 & 0 & -3 \\ 0 & 1 & -6 \\ 0 & 0 & 1 \end{pmatrix}.$$

By Theorem 4.1, the index space of $(L_1, L_2, L_3)$ is *included in* the set of integer points of the polytope defined by the system

$$\left.\begin{aligned} \mathbf{p}_0\Theta^{-1} &\le \mathbf{I}\mathbf{P}\Theta^{-1} \\ \mathbf{I}\mathbf{Q}\Theta^{-1} &\le \mathbf{q}_0\Theta^{-1}, \end{aligned}\right\}$$

that is, of the polytope defined by the system

$$(3, 10/3, 1) \le \mathbf{I}\begin{pmatrix} 1/2 & -1/3 & 1/2 \\ 0 & 1/3 & 3/2 \\ 0 & 0 & -1/2 \end{pmatrix}$$

$$\mathbf{I}\begin{pmatrix} 1/2 & -1/3 & 1 \\ 0 & 1/3 & -1/2 \\ 0 & 0 & -1/2 \end{pmatrix} \le (101/2, 217/3, -1/2).$$

After simplification and regrouping, these inequalities may be written as

$$\left.\begin{array}{c} 6 \le I_1 \le 101 \\ I_1 + 10 \le I_2 \le I_1 + 217 \\ 2I_1 - I_2 + 1 \le I_3 \le I_1 + 3I_2 - 2. \end{array}\right\}$$

We can get this set directly from the given loop nest if we take into account the signs of the loop strides. Of all integer vectors $\mathbf{I}$ that satisfy these inequalities, only those for which $(\mathbf{IP} - \mathbf{p}_0)\mathbf{\Theta}^{-1}$ is also an integer vector, are index points for $(L_1, L_2, L_3)$.

By Theorem 4.2, the iteration space of $(L_1, L_2, L_3)$ is *equal to* the set of integer points in the polytope defined by the system

$$\left.\begin{array}{c} \mathbf{0} \le \hat{\mathbf{I}} \\ \hat{\mathbf{I}}\hat{\mathbf{Q}} \le \hat{\mathbf{q}}_0. \end{array}\right\}$$

It is the set of all integer points $\hat{\mathbf{I}} \ge \mathbf{0}$ such that

$$\hat{\mathbf{I}} \begin{pmatrix} 1 & 0 & -3 \\ 0 & 1 & -6 \\ 0 & 0 & 1 \end{pmatrix} \le (95/2, 207/3, 55/2),$$

that is, the set of all integer points that satisfy

$$\left.\begin{array}{l} 0 \le \hat{I}_1 \le 47 \\ 0 \le \hat{I}_2 \le 69 \\ 0 \le \hat{I}_3 \le 3\hat{I}_1 + 6\hat{I}_2 + 27. \end{array}\right\} \tag{4.13}$$

By Corollary 2 to Theorem 4.1, the mapping between the index and iteration spaces is defined by the equation

$$\mathbf{IP} = \hat{\mathbf{I}}\mathbf{\Theta} + \mathbf{p}_0,$$

that is, by the equation

$$\mathbf{I} \begin{pmatrix} 1 & -1 & -1 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix} = \hat{\mathbf{I}} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -2 \end{pmatrix} + (6, 10, -2).$$

We can express $\mathbf{I}$ in terms of $\hat{\mathbf{I}}$:

$$\begin{aligned}\mathbf{I} &= \hat{\mathbf{I}}\Theta\mathbf{P}^{-1} + \mathbf{p}_0\mathbf{P}^{-1} \\ &= \hat{\mathbf{I}}\begin{pmatrix} 2 & 2 & 8 \\ 0 & 3 & 9 \\ 0 & 0 & -2 \end{pmatrix} + (6, 16, 52) \\ &= (2\hat{I}_1 + 6, 2\hat{I}_1 + 3\hat{I}_2 + 16, 8\hat{I}_1 + 9\hat{I}_2 - 2\hat{I}_3 + 52), \end{aligned} \tag{4.14}$$

and express $\hat{\mathbf{I}}$ in terms of $\mathbf{I}$:

$$\begin{aligned}\hat{\mathbf{I}} &= \mathbf{I}\mathbf{P}\Theta^{-1} - \mathbf{p}_0\Theta^{-1} \\ &= \mathbf{I}\begin{pmatrix} 1/2 & -1/3 & 1/2 \\ 0 & 1/3 & 3/2 \\ 0 & 0 & -1/2 \end{pmatrix} - (3, 10/3, 1) \\ &= (I_1/2 - 3, (-I_1 + I_2 - 10)/3, (I_1 + 3I_2 - I_3 - 2)/2).\end{aligned}$$

It is not necessary to rewrite the program explicitly in terms of iteration variables for dependence analysis. However, we will do so to show how its form changes. To switch over from the index to the iteration variables, we replace $I_1, I_2$, and $I_3$ by the corresponding functions of $\hat{I}_1, \hat{I}_2$, and $\hat{I}_3$ from (4.14), and note the bounds on $\hat{I}_1, \hat{I}_2$, and $\hat{I}_3$ given in (4.13). The given program is then equivalent to the loop nest (Theorem 4.3)

```
L̂1 :   do Î1 = 0, 47, 1
L̂2 :     do Î2 = 0, 69, 1
L̂3 :       do Î3 = 0, 3Î1 + 6Î2 + 27, 1
             H(2Î1 + 6, 2Î1 + 3Î2 + 16, 8Î1 + 9Î2 − 2Î3 + 52)
           enddo
         enddo
       enddo
```

Note that the regular double loop $(L_1, L_2)$ has changed into the rectangular double loop $(\hat{L}_1, \hat{L}_2)$ (Corollary 1 to Theorem 4.3).

---

**EXERCISES 4.2**

1. What can you say in general about the matrices $\mathbf{P}^{-1}$ and $\mathbf{Q}^{-1}$?

2. Show that $\hat{\mathbf{Q}}$ is an upper triangular rational matrix with the main diagonal $(1, 1, \ldots, 1)$ and $\det(\hat{\mathbf{Q}}) = 1$. Show also that $\hat{\mathbf{Q}}$ is equal to the identity matrix iff the loop nest **L** is regular.
3. Let $\hat{\mathbf{L}}$ denote the loop nest obtained by normalizing the loops of **L**. Prove that $\hat{\mathbf{L}}$ is rectangular iff **L** is regular.
4. Find $\mathbf{p}_0, \mathbf{q}_0, \mathbf{P}, \mathbf{Q}, \hat{\mathbf{q}}_0$, and $\hat{\mathbf{Q}}$ for the loop nests of Exercise 3.2.3. Using (4.7) and (4.8), express the index variables in terms of the iteration variables and vice versa. Find a description of the iteration space from (4.9). Compare with results you already got from working on this exercise in Chapter 3.
5. In Example 4.1, find the total number of iteration (index) points of the loop nest, and the last iteration and index values in the sequential execution of the nest.
6. For the perfect loop nests in the following programs, find
   - $\mathbf{p}_0, \mathbf{q}_0, \mathbf{P}, \mathbf{Q}, \hat{\mathbf{q}}_0$, and $\hat{\mathbf{Q}}$;
   - the index and iteration spaces (theorems 4.1 and 4.2);
   - the total number of iterations;
   - the last index and iteration values in the sequential execution of the program;
   - expressions of the iteration variables in terms of the index variables;
   - an equivalent loop nest with unit strides:

(a)
```
L1 :  do I1 = 3, 100, 2
L2 :     do I2 = 3I1 − 30, 3I1 + 400, 5
L3 :        do I3 = 5I1 − 2I2 + 106, 5I1 − 2I2 + 1, −2
               H(I1, I2, I3)
            enddo
         enddo
      enddo
```

(b)
```
L1 :  do I1 = 3, 100, 1
L2 :     do I2 = 3, I1, 1
L3 :        do I3 = I2, 100, 1
               H(I1, I2, I3)
            enddo
         enddo
      enddo
```

(c)
```
L1 :  do I1 = 10, 200, 5
L2 :     do I2 = 2I1, I1, −1
L3 :        do I3 = I1 + I2, I1, −1
L4 :           do I4 = 3I2 + I3, 1000, 2
                  H(I1, I2, I3)
               enddo
            enddo
         enddo
      enddo
```

7. Show how you will compute the total number of iteration (index) points of the model loop nest L, and its last iteration and index values in sequential execution.
8. Find a matrix A such that the second vector inequality of (4.9) is equivalent to

$$\hat{\mathbf{I}}\hat{\mathbf{Q}}\mathbf{A} \leq \hat{\mathbf{q}}_0\mathbf{A},$$

where all coefficients are integral.

## 4.3 Dependence Concepts

Consider two arbitrary assignment statements $S$ and $T$ in the given loop nest **L**. The notations $S < T$ and $S \leq T$ are defined as before, and it is obvious that $\leq$ is a total order in the set of assignment statements in the program.

Let $S(\mathbf{i})$ and $T(\mathbf{j})$ denote the instances of $S$ and $T$ corresponding to two index points **i** and **j**, respectively. The *distance* from $S(\mathbf{i})$ to $T(\mathbf{j})$ is defined to be the $m$-vector $\hat{\jmath} - \hat{\imath}$, where $\hat{\imath}$ and $\hat{\jmath}$ are the iteration points corresponding to **i** and **j**.

The following lemma is a direct generalization of Lemma 3.2. It can easily be established by induction; we omit the proof.

**Lemma 4.4** *Consider two assignment statements $S$ and $T$ in the loop nest* **L**. *Let* **d** *denote the distance from an instance $S(\mathbf{i})$ of $S$ to an instance $T(\mathbf{j})$ of $T$. In the sequential execution of the program, $S(\mathbf{i})$ is executed before $T(\mathbf{j})$ iff one of the following two conditions holds:*

(a) $\mathbf{d} \succ \mathbf{0}$,

(b) $\mathbf{d} = \mathbf{0}$ *and* $S < T$.

The relation of *dependence* between statements is denoted by $\delta$ and is defined as follows: For two statements $S$ and $T$, we have $S\,\delta\,T$ if there is an instance $S(\mathbf{i})$ of $S$, an instance $T(\mathbf{j})$ of $T$, and a memory location $\mathcal{M}$, such that

1. Both $S(\mathbf{i})$ and $T(\mathbf{j})$ reference (read or write) $\mathcal{M}$;
2. $S(\mathbf{i})$ is executed before $T(\mathbf{j})$ in the sequential execution of the program;

3. During sequential execution, the location $\mathcal{M}$ is not written in the time period from the end of execution of $S(\mathbf{i})$ to the beginning of execution of $T(\mathbf{j})$.

The statements $S$ and $T$ need not be distinct, but Condition 2 requires that the instances $S(\mathbf{i})$ and $T(\mathbf{j})$ be distinct. The notation $S\,\delta\,T$ is read as "$T$ depends on $S$." The *dependence* of $T$ on $S$ is the set of all pairs $(S(\mathbf{i}), T(\mathbf{j}))$ that satisfy the above conditions. Thus, we have $S\,\delta\,T$ iff the dependence of $T$ on $S$ is nonempty. The graph of the relation $\delta$ is called the *statement dependence graph* of the given program.

Suppose that a statement $T$ depends on a statement $S$. Let $S(\mathbf{i})$ and $T(\mathbf{j})$ denote a pair of instances, such that they (together with some memory location $\mathcal{M}$) satisfy the above conditions. There is at least one such pair. We say that the instance $T(\mathbf{j})$ *depends* on the instance $S(\mathbf{i})$. Let $\mathbf{d}$ denote the distance from $S(\mathbf{i})$ to $T(\mathbf{j})$. Let $\boldsymbol{\sigma} = \mathbf{sig}(\mathbf{d})$ and $\ell = \text{lev}(\mathbf{d})$. (For the definitions of "**sig**" and "lev," see Section I.1.3.) Then, $\mathbf{d}$ is a *distance vector*, $\boldsymbol{\sigma}$ a *direction vector*, and $\ell$ a *dependence level* for the dependence of $T$ on $S$. It is sometimes convenient to say that $T$ depends on $S$ *with* a distance vector $\mathbf{d}$ or a direction vector $\boldsymbol{\sigma}$, and *at* a level $\ell$.

Since $S(\mathbf{i})$ must be executed before $T(\mathbf{j})$, we have the following theorem as a direct consequence of Lemma 4.4. The next two corollaries follow from the definitions just given.

**Theorem 4.5** *If* $\mathbf{d}$ *is a distance vector for the dependence of statement* $T$ *on statement* $S$, *then* $\mathbf{d} \succeq \mathbf{0}$. *The equality* $\mathbf{d} = \mathbf{0}$ *is possible only if* $S < T$.

**Corollary 1** *If* $\boldsymbol{\sigma}$ *is a direction vector for the dependence of statement* $T$ *on statement* $S$, *then* $\boldsymbol{\sigma} \succeq \mathbf{0}$. *The equality* $\boldsymbol{\sigma} = \mathbf{0}$ *is possible only if* $S < T$.

**Corollary 2** *If* $\ell$ *is a dependence level for the dependence of statement* $T$ *on statement* $S$, *then* $1 \leq \ell \leq m + 1$ *(where* $m$ *is the number of loops in the perfect nest* $\mathbf{L}$*). The value* $\ell = m + 1$ *is possible only if* $S < T$.

The dependence of $T$ on $S$ can be partitioned into $(m + 1)$ parts: the parts carried by the $m$ loops and the loop-independent part. The part *carried by* the loop $L_r$ consists of all instance pairs $(S(\mathbf{i}), T(\mathbf{j}))$

such that $\hat{\imath} \prec_r \hat{\jmath}$ and $T(\mathbf{j})$ depends on $S(\mathbf{i})$, $1 \leq r \leq m$. The *loop-independent* part consists of all instance pairs $(S(\mathbf{i}), T(\mathbf{i}))$ such that $T(\mathbf{i})$ depends on $S(\mathbf{i})$. The following statements are equivalent in the context of dependence of a statement $T$ on a statement $S$:

1. The dependence of $T$ on $S$ is carried by the loop $L_r$;
2. There is a pair of instances $S(\mathbf{i})$ and $T(\mathbf{j})$, such that $\hat{\imath} \prec_r \hat{\jmath}$ and $T(\mathbf{j})$ depends on $S(\mathbf{i})$;
3. There is a dependence distance vector $\mathbf{d}$ with $\mathbf{d} \succ_r \mathbf{0}$;
4. There is a dependence direction vector $\sigma$ with $\sigma \succ_r \mathbf{0}$;
5. $T$ depends on $S$ at level $r$.

As in Section 2.3, we can define flow, anti-, output, and input dependences; indirect dependence; dependence caused by a pair of variables; etc. When needed, we also associate a distance or a direction vector, or a level of dependence with a particular type of dependence, or even with a particular pair of variables.

---

**Example 4.2** In this example, we study dependence between statements and their instances in the triple loop

```
L1 :          do I1 = 1,4,2
L2 :             do I2 = 10,5,-3
L3 :                do I3 = 1,3,1
     S :               X(I1 + I2 + I3) = ···
     T :               ··· = ··· X(I2 + I3) ···
                    enddo
                 enddo
              enddo
```

In Table 4.1, we have shown the iterations of the loop nest obtained by unrolling the loops. From this table, the dependence structure between statement instances becomes clear; that structure is presented in Table 4.2. (This table does not include input dependence.) Note that the dependence distance from one statement instance to another must be computed from the corresponding pair of iteration points shown in Table 4.1. From Table 4.2, we can easily derive the dependence information between statements themselves. We see that

| Iteration Point | Iteration of loop Nest | |
|---|---|---|
| $(0,0,0)$ | $S(1,10,1):$ | $X(12) = \cdots$ |
| | $T(1,10,1):$ | $\cdots = \cdots X(11) \cdots$ |
| $(0,0,1)$ | $S(1,10,2):$ | $X(13) = \cdots$ |
| | $T(1,10,2):$ | $\cdots = \cdots X(12) \cdots$ |
| $(0,0,2)$ | $S(1,10,3):$ | $X(14) = \cdots$ |
| | $T(1,10,3):$ | $\cdots = \cdots X(13) \cdots$ |
| $(0,1,0)$ | $S(1,7,1):$ | $X(9) = \cdots$ |
| | $T(1,7,1):$ | $\cdots = \cdots X(8) \cdots$ |
| $(0,1,1)$ | $S(1,7,2):$ | $X(10) = \cdots$ |
| | $T(1,7,2):$ | $\cdots = \cdots X(9) \cdots$ |
| $(0,1,2)$ | $S(1,7,3):$ | $X(11) = \cdots$ |
| | $T(1,7,3):$ | $\cdots = \cdots X(10) \cdots$ |
| $(1,0,0)$ | $S(3,10,1):$ | $X(14) = \cdots$ |
| | $T(3,10,1):$ | $\cdots = \cdots X(11) \cdots$ |
| $(1,0,1)$ | $S(3,10,2):$ | $X(15) = \cdots$ |
| | $T(3,10,2):$ | $\cdots = \cdots X(12) \cdots$ |
| $(1,0,2)$ | $S(3,10,3):$ | $X(16) = \cdots$ |
| | $T(3,10,3):$ | $\cdots = \cdots X(13) \cdots$ |
| $(1,1,0)$ | $S(3,7,1):$ | $X(11) = \cdots$ |
| | $T(3,7,1):$ | $\cdots = \cdots X(8) \cdots$ |
| $(1,1,1)$ | $S(3,7,2):$ | $X(12) = \cdots$ |
| | $T(3,7,2):$ | $\cdots = \cdots X(9) \cdots$ |
| $(1,1,2)$ | $S(3,7,3):$ | $X(13) = \cdots$ |
| | $T(3,7,3):$ | $\cdots = \cdots X(10) \cdots$ |

Table 4.1: Loop nest of Example 4.2 after unrolling.

| From Instance | To Instance | Type | Distance | Direction | Level |
|---|---|---|---|---|---|
| $S(1,10,1)$ | $T(1,10,2)$ | Flow | $(0,0,1)$ | $(0,0,1)$ | 3 |
| $S(1,10,1)$ | $T(3,10,2)$ | Flow | $(1,0,1)$ | $(1,0,1)$ | 1 |
| $S(1,10,1)$ | $S(3,7,2)$ | Output | $(1,1,1)$ | $(1,1,1)$ | 1 |
| $T(1,10,1)$ | $S(1,7,3)$ | Anti- | $(0,1,2)$ | $(0,1,1)$ | 2 |
| $S(1,7,3)$ | $T(3,10,1)$ | Flow | $(1,-1,-2)$ | $(1,-1,-1)$ | 1 |
| $T(3,10,1)$ | $S(3,7,1)$ | Anti- | $(0,1,0)$ | $(0,1,0)$ | 2 |
| $S(1,7,3)$ | $S(3,7,1)$ | Output | $(1,0,-2)$ | $(1,0,-1)$ | 1 |
| $S(1,10,2)$ | $T(1,10,3)$ | Flow | $(0,0,1)$ | $(0,0,1)$ | 3 |
| $T(1,10,3)$ | $S(3,7,3)$ | Anti- | $(1,1,0)$ | $(1,1,0)$ | 1 |
| $S(1,10,2)$ | $S(3,7,3)$ | Output | $(1,1,1)$ | $(1,1,1)$ | 1 |
| $S(1,10,3)$ | $S(3,10,1)$ | Output | $(1,0,-2)$ | $(1,0,-1)$ | 1 |
| $S(1,7,1)$ | $T(1,7,2)$ | Flow | $(0,0,1)$ | $(0,0,1)$ | 3 |
| $S(1,7,1)$ | $T(3,7,2)$ | Flow | $(1,0,1)$ | $(1,0,1)$ | 1 |
| $S(1,7,2)$ | $T(1,7,3)$ | Flow | $(0,0,1)$ | $(0,0,1)$ | 3 |
| $S(1,7,2)$ | $T(3,7,3)$ | Flow | $(1,0,1)$ | $(1,0,1)$ | 1 |

Table 4.2: Dependence structure of $(L_1, L_2, L_3)$ in Example 4.2.

1. There is a flow dependence of $T$ on $S$. The distance vectors of this dependence are $(0,0,1)$, $(1,0,1)$, and $(1,-1,-2)$. The direction vectors of this dependence are $(0,0,1)$, $(1,0,1)$, and $(1,-1,-1)$. The dependence levels are 1 and 3. No part of this dependence is carried by the loop $L_2$.

2. There is an anti-dependence of $S$ on $T$. The distance vectors of this dependence are $(0,1,2)$, $(0,1,0)$, and $(1,1,0)$. The direction vectors are $(0,1,1)$, $(0,1,0)$, and $(1,1,0)$. The dependence levels are 1 and 2. No part of this dependence is carried by $L_3$.

3. There is an output-dependence of $S$ on itself. The distance vectors of this dependence are $(1,1,1)$ and $(1,0,-2)$. The direction vectors are $(1,1,1)$ and $(1,0,-1)$. The only dependence level is 1. No part of this dependence is carried by either $L_2$ or $L_3$.

4. There is no loop-independent dependence in this example.

**EXERCISES 4.3**

1. Study the dependence structure of the following programs:

(a)
$L_1$ : **do** $I_1 = 0,8,2$
$L_2$ : **do** $I_2 = 0,8,2$
$L_3$ : **do** $I_3 = 0,8,2$
$S$ : $X(I_1,I_2,I_3) = \cdots$
$T$ : $\cdots = \cdots X(I_1 - 1, I_2 + 1, I_3) \cdots$
**enddo**
**enddo**
**enddo**

(b)
$L_1$ : **do** $I_1 = 1,4,1$
$L_2$ : **do** $I_2 = 10, I_1, -3$
$L_3$ : **do** $I_3 = I_1, I_2, 1$
$S$ : $X(I_1 + I_2 + I_3) = \cdots$
$T$ : $\cdots = \cdots X(I_1 + I_2 + I_3) \cdots$
**enddo**
**enddo**
**enddo**

(c)
$L_1$ : **do** $I_1 = 0,10,3$
$L_2$ : **do** $I_2 = I_1, 10, 4$
$L_3$ : **do** $I_3 = I_2, I_1, -1$
$S$ : $X(I_1, I_2 + 1, I_3) = \cdots$
$T$ : $\cdots = \cdots X(I_3, I_2, I_1 + I_3) \cdots$
**enddo**
**enddo**
**enddo**

## 4.4 Subscript Representation

We have assumed that the input and output variables of statements in the given program are array-elements with subscripts affine in the index variables of appropriate loops. (Note that this description includes scalars.) Consider a variable of a statement $S$ in our model program **L**, that is an element of an $n$-dimensional array $X$. It has the form

$$X(a_{11}I_1 + \cdots + a_{m1}I_m + a_{01}, \ldots, a_{1n}I_1 + \cdots + a_{mn}I_m + a_{0n}),$$

where the coefficients are all integer constants. In matrix notation, this variable can be written as $X(\mathbf{IA} + \mathbf{a}_0)$, where

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

and $\mathbf{a}_0 = (a_{01}, a_{02}, \ldots, a_{0n})$. The *coefficient matrix* of this element of $X$ is **A**, and its *coefficient vector* is $\mathbf{a}_0$. We can write

$$\begin{aligned} \mathbf{IA} + \mathbf{a}_0 &= (\hat{\mathbf{I}}\boldsymbol{\Theta} + \mathbf{p}_0)\mathbf{P}^{-1}\mathbf{A} + \mathbf{a}_0 && \text{by (4.7)} \\ &= \hat{\mathbf{I}}(\boldsymbol{\Theta}\mathbf{P}^{-1}\mathbf{A}) + (\mathbf{p}_0\mathbf{P}^{-1}\mathbf{A} + \mathbf{a}_0) \\ &= \hat{\mathbf{I}}\hat{\mathbf{A}} + \hat{\mathbf{a}}_0, \end{aligned}$$

where

$$\left.\begin{aligned} \hat{\mathbf{A}} &= \boldsymbol{\Theta}\mathbf{P}^{-1}\mathbf{A} \\ \hat{\mathbf{a}}_0 &= \mathbf{p}_0\mathbf{P}^{-1}\mathbf{A} + \mathbf{a}_0. \end{aligned}\right\} \tag{4.15}$$

We refer to $\hat{\mathbf{A}}$ and $\hat{\mathbf{a}}_0$ as the *normalized* coefficient matrix and the *normalized* coefficient vector, respectively, and call $X(\hat{\mathbf{I}}\hat{\mathbf{A}} + \hat{\mathbf{a}}_0)$ the *normalized* form of the program variable $X(\mathbf{IA} + \mathbf{a}_0)$.

---

**Example 4.3** For the loop nest of Example 4.1, we already found that

$$\mathbf{P}^{-1} = \begin{pmatrix} 1 & 1 & 4 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Theta} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -2 \end{pmatrix}.$$

Consider now the same nest with one particular statement:

$L_1$ : **do** $I_1 = 6, 101, 2$
$L_2$ : **do** $I_2 = I_1 + 10, I_1 + 217, 3$
$L_3$ : **do** $I_3 = I_1 + 3I_2 - 2, 2I_1 - I_2 + 1, -2$
$S$ : $X(2I_1 + I_3 + 1, -I_1 + 3I_2 + I_3 - 2, -4I_2) = 1$
**enddo**
**enddo**
**enddo**

Since

$$(2I_1 + I_3 + 1, -I_1 + 3I_2 + I_3 - 2, -4I_2) = (I_1, I_2, I_3)\begin{pmatrix} 2 & -1 & 0 \\ 0 & 3 & -4 \\ 1 & 1 & 0 \end{pmatrix} + (1, -2, 0),$$

the coefficient matrix **A** and vector $\mathbf{a}_0$ for the array-element in this statement are given by

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 \\ 0 & 3 & -4 \\ 1 & 1 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{a}_0 = (1, -2, 0).$$

We compute the normalized coefficient matrix and the normalized coefficient vector:

$$\begin{aligned} \hat{\mathbf{A}} &= \Theta\mathbf{P}^{-1}\mathbf{A} \\ &= \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -2 \end{pmatrix}\begin{pmatrix} 1 & 1 & 4 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 2 & -1 & 0 \\ 0 & 3 & -4 \\ 1 & 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 12 & 12 & -8 \\ 9 & 18 & -12 \\ -2 & -2 & 0 \end{pmatrix}, \end{aligned}$$

and

$$\begin{aligned} \hat{\mathbf{a}}_0 &= \mathbf{p}_0\mathbf{P}^{-1}\mathbf{A} + \mathbf{a}_0 \\ &= (6, 10, -2)\begin{pmatrix} 1 & 1 & 4 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 2 & -1 & 0 \\ 0 & 3 & -4 \\ 1 & 1 & 0 \end{pmatrix} + (1, -2, 0) \\ &= (65, 92, -64). \end{aligned}$$

Thus, in terms of iteration variables, the array-element of $S$ becomes

$$X(12\hat{I}_1 + 9\hat{I}_2 - 2\hat{I}_3 + 65, 12\hat{I}_1 + 18\hat{I}_2 - 2\hat{I}_3 + 92, -8\hat{I}_1 - 12\hat{I}_2 - 64).$$

This is the normalized form of the program variable.

---

**EXERCISES 4.4**

1. Find the normalized forms of the program variables in the loop nests of Example 3.2, Exercise 3.3.3, Example 3.5, Exercise 4.3.1.

## 4.5 Dependence Problem

Consider two statements $S$ and $T$ in the model program **L**. Suppose $S$ has a program variable of the form $X(\mathbf{IA} + \mathbf{a}_0)$ and $T$ a program variable of the form $X(\mathbf{IB} + \mathbf{b}_0)$, where $X$ is an $n$-dimensional array, **A** and **B** are $m \times n$ integer matrices, and $\mathbf{a}_0$ and $\mathbf{b}_0$ are integer $n$-vectors.

First, we describe the dependence problem in terms of index variables. The instance of the variable $X(\mathbf{IA} + \mathbf{a}_0)$ for an index point **i** of the loop nest **L** is $X(\mathbf{iA} + \mathbf{a}_0)$. The instance of $X(\mathbf{IB} + \mathbf{b}_0)$ for an index point **j** is $X(\mathbf{jB} + \mathbf{b}_0)$. These two instances represent the same memory location iff $\mathbf{iA} + \mathbf{a}_0 = \mathbf{jB} + \mathbf{b}_0$, that is, iff

$$\mathbf{iA} - \mathbf{jB} = \mathbf{b}_0 - \mathbf{a}_0. \tag{4.16}$$

This matrix equation is the *dependence equation* for $X(\mathbf{IA} + \mathbf{a}_0)$ and $X(\mathbf{IB} + \mathbf{b}_0)$ *in terms of index values.* It consists of $n$ scalar equations, and there are $2m$ integer variables: the $m$ elements of **i** and the $m$ elements of **j**. Since **i** and **j** are index points of **L**, they must satisfy the following conditions (Theorem 4.1):

$$\left.\begin{aligned} \mathbf{p}_0\Theta^{-1} &\le \mathbf{iP}\Theta^{-1} \\ \mathbf{iQ}\Theta^{-1} &\le \mathbf{q}_0\Theta^{-1} \\ \mathbf{p}_0\Theta^{-1} &\le \mathbf{jP}\Theta^{-1} \\ \mathbf{jQ}\Theta^{-1} &\le \mathbf{q}_0\Theta^{-1}, \end{aligned}\right\} \tag{4.17}$$

and

$$\left.\begin{aligned} (\mathbf{iP} - \mathbf{p}_0)\Theta^{-1} &\text{ is an integer vector} \\ (\mathbf{jP} - \mathbf{p}_0)\Theta^{-1} &\text{ is an integer vector.} \end{aligned}\right\} \tag{4.18}$$

These conditions are the *dependence constraints in terms of index values.*

Next, we state the dependence problem in terms of iteration variables. Let $X(\hat{\mathbf{I}}\hat{\mathbf{A}} + \hat{\mathbf{a}}_0)$ and $X(\hat{\mathbf{I}}\hat{\mathbf{B}} + \hat{\mathbf{b}}_0)$ denote the normalized forms of the program variables $X(\mathbf{IA} + \mathbf{a}_0)$ and $X(\mathbf{IB} + \mathbf{b}_0)$, respectively (see Section 4.4). The matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$, and the vectors $\hat{\mathbf{a}}_0$ and $\hat{\mathbf{b}}_0$ are defined by equations (4.15):

$$\begin{aligned}\hat{\mathbf{A}} &= \Theta\mathbf{P}^{-1}\mathbf{A}\\ \hat{\mathbf{a}}_0 &= \mathbf{p}_0\mathbf{P}^{-1}\mathbf{A} + \mathbf{a}_0\\ \hat{\mathbf{B}} &= \Theta\mathbf{P}^{-1}\mathbf{B}\\ \hat{\mathbf{b}}_0 &= \mathbf{p}_0\mathbf{P}^{-1}\mathbf{B} + \mathbf{b}_0.\end{aligned}$$

The instance of the variable $X(\hat{\mathbf{I}}\hat{\mathbf{A}} + \hat{\mathbf{a}}_0)$ for an iteration point $\hat{\imath}$ of the loop nest $\mathbf{L}$ is $X(\hat{\imath}\hat{\mathbf{A}} + \hat{\mathbf{a}}_0)$. The instance of $X(\hat{\mathbf{I}}\hat{\mathbf{B}} + \hat{\mathbf{b}}_0)$ for an iteration point $\hat{\jmath}$ is $X(\hat{\jmath}\hat{\mathbf{B}}+\hat{\mathbf{b}}_0)$. These two instances represent the same memory location iff $\hat{\imath}\hat{\mathbf{A}} + \hat{\mathbf{a}}_0 = \hat{\jmath}\hat{\mathbf{B}} + \hat{\mathbf{b}}_0$, that is, iff

$$\hat{\imath}\hat{\mathbf{A}} - \hat{\jmath}\hat{\mathbf{B}} = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0. \tag{4.19}$$

The matrix equation (4.19) is the *dependence equation* for the variables $X(\hat{\mathbf{I}}\hat{\mathbf{A}} + \hat{\mathbf{a}}_0)$ and $X(\hat{\mathbf{I}}\hat{\mathbf{B}} + \hat{\mathbf{b}}_0)$ *in terms of iteration values.*[1] It consists of $n$ scalar equations, and there are $2m$ integer variables: the $m$ elements of $\hat{\imath}$ and the $m$ elements of $\hat{\jmath}$. Since $\hat{\imath}$ and $\hat{\jmath}$ are iteration points of $\mathbf{L}$, they must satisfy the inequalities (Theorem 4.2):

$$\left.\begin{aligned}\mathbf{0} &\le \hat{\imath}\\ \hat{\imath}\hat{\mathbf{Q}} &\le \hat{\mathbf{q}}_0\\ \mathbf{0} &\le \hat{\jmath}\\ \hat{\jmath}\hat{\mathbf{Q}} &\le \hat{\mathbf{q}}_0,\end{aligned}\right\} \tag{4.20}$$

where the matrix $\hat{\mathbf{Q}}$ and the vector $\hat{\mathbf{q}}_0$ are defined by (4.10). These inequalities are the *dependence constraints* for the variables $X(\hat{\mathbf{I}}\hat{\mathbf{A}}+\hat{\mathbf{a}}_0)$ and $X(\hat{\mathbf{I}}\hat{\mathbf{B}} + \hat{\mathbf{b}}_0)$ *in terms of iteration values.*

The main purpose of introducing iteration variables was to eliminate conditions like (4.18). When the stride of each loop is one, the

[1] In chapters 1-3, we first derived the dependence equation in index values, and then expressed the index values in terms of iteration values to get the equation in iteration values. Here also we can derive (4.19) from (4.16) by using (4.7); see Exercise 1.

stride matrix $\Theta$ is equal to the identity matrix, and (4.18) is satisfied automatically. In that case, we can approach the dependence problem through its description in terms of index variables. The conditions in (4.18) disappear also when the stride of each loop is $\pm 1$. However, we still prefer to work with iteration variables in this case, to avoid the fact that when the stride of a loop is $-1$, its index values form a *decreasing* sequence in the sequential execution. It is possible to do the analysis in a framework where we use index variables for loops with unit strides and iteration variables for all other loops. For the most part, we use either index variables or iteration variables for all loops in a given problem, and use iteration variables in formal statements.

The generalizations of theorems 2.6 and 2.7 to a perfect loop nest are stated below as theorems 4.6 and 4.7. We also add here two corollaries to Theorem 4.6, dealing with direction vectors and dependence levels. (Direction vectors and dependence levels were not discussed in Chapter 2, since they are not very significant for single loops.) The proofs are simple and left to the reader.

**Theorem 4.6** *Consider two statements $S$ and $T$ in the loop nest* **L**. *Let $X(\mathbf{IA}+\mathbf{a}_0)$ denote a variable of $S$ and $X(\mathbf{IB}+\mathbf{b}_0)$ a variable of $T$, where $X$ is an $n$-dimensional array. If these variables cause a dependence between $S$ and $T$, then Equation (4.19) has an integer solution $(\hat{\imath}, \hat{\jmath})$ that satisfies the constraints (4.20). In each of the following special cases, the solution satisfies an additional condition:*

(a) *If $S < T$ and $S \,\delta\, T$, then $\hat{\imath} \preceq \hat{\jmath}$;*

(b) *If $S = T$ and $S \,\delta\, S$, then $\hat{\imath} \prec \hat{\jmath}$;*

(c) *If $S < T$ and $T \,\delta\, S$, then $\hat{\imath} \succ \hat{\jmath}$.*

**Corollary 1** *If statement $T$ depends on statement $S$ with a direction vector $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$, then Equation (4.19) has an integer solution $(\hat{\imath}, \hat{\jmath})$ that satisfies (4.20) and the additional condition*

$$\operatorname{sig}(\hat{\jmath}_r - \hat{\imath}_r) = \sigma_r \qquad (1 \le r \le m), \tag{4.21}$$

*where $\hat{\imath} = (\hat{\imath}_1, \hat{\imath}_2, \ldots, \hat{\imath}_m)$ and $\hat{\jmath} = (\hat{\jmath}_1, \hat{\jmath}_2, \ldots, \hat{\jmath}_m)$.*

If $\sigma_r = 0$, then the extra equation $\hat{\imath}_r = \hat{\jmath}_r$ gets added to the system (4.19). We can use it to reduce the number of variables by one. If $\sigma_r = 1$, we get an extra inequality of the form $\hat{\imath}_r < \hat{\jmath}_r$, which is equivalent to $\hat{\imath}_r \leq \hat{\jmath}_r - 1$ since we are dealing with integer variables. If $\sigma_r = -1$, then the extra inequality is $\hat{\imath}_r \geq \hat{\jmath}_r + 1$.

By Corollary 1 to Theorem 4.5, a statement may depend on itself only with a positive ($\succ \mathbf{0}$) direction vector. Next, take two statements $S$ and $T$ such that $S < T$. A direction vector for a dependence of $S$ on $T$ must be positive, while a direction vector for a dependence of $T$ on $S$ can be nonnegative ($\succeq \mathbf{0}$). If we always associate the notations $\mathbf{i}$ and $\hat{\imath}$ with $S$, and $\mathbf{j}$ and $\hat{\jmath}$ with $T$, then a direction vector for a dependence of $T$ on $S$ has the form $(\mathrm{sig}(\hat{\jmath}_1 - \hat{\imath}_1), \ldots, \mathrm{sig}(\hat{\jmath}_m - \hat{\imath}_m))$, and a direction vector for a dependence of $S$ on $T$ has the form $(\mathrm{sig}(\hat{\imath}_1 - \hat{\jmath}_1), \ldots, \mathrm{sig}(\hat{\imath}_m - \hat{\jmath}_m))$.

The following result is an immediate consequence of Corollary 1, since dependence at a level $\ell$ is equivalent to dependence with a direction vector of the form $(\underbrace{0, 0, \ldots, 0}_{\ell-1}, 1, *, *, \ldots, *)$.

**Corollary 2** *If statement $T$ depends on statement $S$ at a level $\ell$, then Equation (4.19) has an integer solution $(\hat{\imath}, \hat{\jmath})$ that satisfies (4.20) and the additional conditions:*

$$\hat{\imath}_1 = \hat{\jmath}_1, \hat{\imath}_2 = \hat{\jmath}_2, \ldots, \hat{\imath}_{\ell-1} = \hat{\jmath}_{\ell-1}, \hat{\imath}_\ell \leq \hat{\jmath}_\ell - 1.$$

By Corollary 2 to Theorem 4.5, the possible levels are $1, 2, \ldots, m+1$ if $S < T$, and $1, 2, \ldots, m$ otherwise.

**Theorem 4.7** *Suppose that Equation (4.19) has an integer solution $(\hat{\imath}, \hat{\jmath})$ that satisfies the constraints (4.20).*

(a) *If $\hat{\imath} \prec \hat{\jmath}$, then $S \overline{\delta} T$.*

(b) *If $\hat{\imath} \succ \hat{\jmath}$, then $T \overline{\delta} S$.*

(c) *If $\hat{\imath} = \hat{\jmath}$ and $S < T$, then $S \overline{\delta} T$.*

Theorems 4.6 and 4.7 can be restated to deal with dependence between statement instances. We will not distinguish between dependence and indirect dependence. If there is an integer solution to

(4.19) satisfying (4.20) and additional conditions, if any, then we will assume that the corresponding dependence exists.

Given two statements $S$ and $T$ in the nest $\mathbf{L}$, a variable $X(\mathbf{IA} + \mathbf{a}_0)$ of $S$, and a variable $X(\mathbf{IB} + \mathbf{b}_0)$ of $T$, the *basic dependence problem* is to decide if there is a dependence between $S$ and $T$, that is, if there is an integer solution to Equation (4.19) that satisfies the constraints (4.20). The *extended dependence problem* is to find for each direction vector $\boldsymbol{\sigma}$, the set of all integer solutions to (4.19) satisfying (4.20) and the additional conditions (4.21). We also want to know the set of all distance vectors corresponding to each particular direction vector (the corresponding level is easily found). This information can be summarized and used to derive qualitative statements, as needed.

---

**Example 4.4** Consider the double loop

```
L1 :    do I1 = 5,200,3
L2 :       do I2 = I1,200,1
   S :        X(2I1 - 1,I2) = ···
   T :        ··· = ··· X(I1,3I1) ···
           enddo
        enddo
```

For this loop nest, the initial and final vectors are

$$\mathbf{p}_0 = (5,0), \qquad \mathbf{q}_0 = (200,200),$$

and the initial, final, and stride matrices are

$$\mathbf{P} = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}, \qquad \mathbf{Q} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \boldsymbol{\Theta} = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}.$$

We compute the inverses of $\boldsymbol{\Theta}$ and $\mathbf{P}$:

$$\boldsymbol{\Theta}^{-1} = \begin{pmatrix} 1/3 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{P}^{-1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix},$$

and the normalized final vector and matrix by (4.10):

$$\hat{\mathbf{q}}_0 = (\mathbf{q}_0 - \mathbf{p}_0\mathbf{P}^{-1}\mathbf{Q})\boldsymbol{\Theta}^{-1} = (65,195),$$
$$\hat{\mathbf{Q}} = \boldsymbol{\Theta}\mathbf{P}^{-1}\mathbf{Q}\boldsymbol{\Theta}^{-1} = \begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix}.$$

The coefficient matrix $\mathbf{A}$ and the coefficient vector $\mathbf{a}_0$ for the array element of statement $S$ are given by

$$\mathbf{A} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{a}_0 = (-1, 0).$$

The coefficient matrix $\mathbf{B}$ and the coefficient vector $\mathbf{b}_0$ for the array element of statement $T$ are given by

$$\mathbf{B} = \begin{pmatrix} 1 & 3 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{b}_0 = (0, 0).$$

We compute the normalized coefficient matrices and the normalized coefficient vectors by (4.15):

$$\hat{\mathbf{A}} = \Theta\mathbf{P}^{-1}\mathbf{A} = \begin{pmatrix} 6 & 3 \\ 0 & 1 \end{pmatrix},$$

$$\hat{\mathbf{B}} = \Theta\mathbf{P}^{-1}\mathbf{B} = \begin{pmatrix} 3 & 9 \\ 0 & 0 \end{pmatrix},$$

$$\hat{\mathbf{a}}_0 = \mathbf{p}_0\mathbf{P}^{-1}\mathbf{A} + \mathbf{a}_0 = (9, 5),$$
$$\hat{\mathbf{b}}_0 = \mathbf{p}_0\mathbf{P}^{-1}\mathbf{B} + \mathbf{b}_0 = (5, 15).$$

The dependence equation (4.19) is

$$(\hat{\imath}_1, \hat{\imath}_2)\begin{pmatrix} 6 & 3 \\ 0 & 1 \end{pmatrix} - (\hat{\jmath}_1, \hat{\jmath}_2)\begin{pmatrix} 3 & 9 \\ 0 & 0 \end{pmatrix} = (-4, 10),$$

which is equivalent to the system:

$$6\hat{\imath}_1 - 3\hat{\jmath}_1 = -4 \tag{4.22}$$
$$3\hat{\imath}_1 + \hat{\imath}_2 - 9\hat{\jmath}_1 = 10. \tag{4.23}$$

This system has two equations in three variables. The dependence constraints (4.20) become

$$\begin{aligned} (0,0) &\le (\hat{\imath}_1, \hat{\imath}_2) \\ &\quad (\hat{\imath}_1, \hat{\imath}_2)\begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix} \le (65, 195) \\ (0,0) &\le (\hat{\jmath}_1, \hat{\jmath}_2) \\ &\quad (\hat{\jmath}_1, \hat{\jmath}_2)\begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix} \le (65, 195), \end{aligned}$$

which reduce to the system:

$$\begin{cases} 0 \le \hat{\imath}_1 \le 65 \\ 0 \le \hat{\imath}_2 \le 195 - 3\hat{\imath}_1 \\ 0 \le \hat{\jmath}_1 \le 65 \\ 0 \le \hat{\jmath}_2 \le 195 - 3\hat{\jmath}_1. \end{cases}$$

It is clear that there is no integer solution to the single equation (4.22), since gcd(6, 3) does not divide 4 (Theorem 2.8). Hence, there is no integer solution to the system of equations (4.22)-(4.23). Therefore, there is no dependence between statements $S$ and $T$ (Theorem 4.6).

---

**EXERCISES 4.5**

1. Derive (4.19) from (4.16) by expressing $\mathbf{i}$ and $\mathbf{j}$ in terms of $\hat{\imath}$ and $\hat{\jmath}$.
2. Prove theorems 4.6 and 4.7. Restate them in terms of dependence between $S(\hat{\imath})$ and $T(\hat{\jmath})$.
3. In Example 4.4, state the dependence equation and constraints in terms of index values.
4. Rework Example 4.4 after changing the loop headers to

   $L_1$ : **do** $I_1 = 100, 1, -1$
   $L_2$ : **do** $I_2 = 2I_1 + 4, 10I_1 - 12, 2$

5. Find the dependence equation and constraints (in terms of iteration values) for the dependence problem between statements $S$ and $T$, $S$ and $S$, and $T$ and $T$ in the programs of Example 4.2, Exercise 4.3.1(a)-(c), and Example 4.3. (In Example 4.3, take $T = S$.)

## 4.6 Special Cases

In the last section, we saw that the dependence problem between two statements $S$ and $T$ is described by the equation

$$\hat{\imath}\hat{\mathbf{A}} - \hat{\jmath}\hat{\mathbf{B}} = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0 \tag{4.24}$$

and the constraints

$$\left.\begin{array}{l} \mathbf{0} \le \hat{\imath} \\ \qquad \hat{\imath}\hat{\mathbf{Q}} \le \hat{\mathbf{q}}_0 \\ \mathbf{0} \le \hat{\jmath} \\ \qquad \hat{\jmath}\hat{\mathbf{Q}} \le \hat{\mathbf{q}}_0, \end{array}\right\} \tag{4.25}$$

where $X(\hat{\mathbf{I}}\hat{\mathbf{A}} + \hat{\mathbf{a}}_0)$ and $X(\hat{\mathbf{I}}\hat{\mathbf{B}} + \hat{\mathbf{b}}_0)$ are the normalized forms of the program variables of $S$ and $T$ that cause the dependence. The normalized final matrix $\hat{\mathbf{Q}}$ and vector $\hat{\mathbf{q}}_0$ are defined in (4.10). In this section, we consider three special cases where the dependence problem becomes somewhat simpler. The first two cases are very common in real programs; they are included in the third case that deals with a more general situation.

### Special Case 1. Equal Coefficient Matrices

Let $\hat{\mathbf{A}} = \hat{\mathbf{B}}$. By (4.15), we have $\hat{\mathbf{A}} = \Theta\mathbf{P}^{-1}\mathbf{A}$ and $\hat{\mathbf{B}} = \Theta\mathbf{P}^{-1}\mathbf{B}$, so that this is the same as requiring $\mathbf{A} = \mathbf{B}$. The variables $X(3I_1+I_2+5, I_1+2I_2+3)$ and $X(3I_1 + I_2 + 1, I_1 + 2I_2)$ in a double loop provide an example for this case, since the coefficient matrix for either variable is $\left(\begin{smallmatrix} 3 & 1 \\ 1 & 2 \end{smallmatrix}\right)$.

In this case, any dependence caused by the two program variables $X(\hat{\mathbf{I}}\hat{\mathbf{A}}+\hat{\mathbf{a}}_0)$ and $X(\hat{\mathbf{I}}\hat{\mathbf{A}}+\hat{\mathbf{b}}_0)$ is said to be *uniform*, and each dependence distance of a uniform dependence is said to be a *uniform distance.* The "uniformity" here refers to the fact that a distance vector of a uniform dependence is independent of iteration (index) points. This is stated more formally in the following theorem.

**Theorem 4.8** *Assume that a variable $X(\hat{\mathbf{I}}\hat{\mathbf{A}}+\hat{\mathbf{a}}_0)$ of a statement $S$ and a variable $X(\hat{\mathbf{I}}\hat{\mathbf{B}} + \hat{\mathbf{b}}_0)$ of a statement $T$ cause a uniform dependence of $T$ on $S$ in the loop nest* **L**. *Let $\mathbf{d}_0$ denote a distance vector of this dependence. If $\hat{\mathbf{i}}_0$ and $\hat{\mathbf{j}}_0$ are two iteration points such that $\hat{\mathbf{j}}_0 - \hat{\mathbf{i}}_0 = \mathbf{d}_0$, then the instance $T(\mathbf{j}_0)$ corresponding to $\hat{\mathbf{j}}_0$ depends on the instance $S(\mathbf{i}_0)$ corresponding to $\hat{\mathbf{i}}_0$.*[2]

PROOF. Since the dependence caused by the two variables is uniform, we have $\hat{\mathbf{A}} = \hat{\mathbf{B}}$ by definition. The dependence equation for the problem, Equation (4.24), becomes

$$(\hat{\mathbf{j}} - \hat{\mathbf{i}})\hat{\mathbf{A}} = \hat{\mathbf{a}}_0 - \hat{\mathbf{b}}_0,$$

or

$$\mathbf{d}\hat{\mathbf{A}} = \hat{\mathbf{a}}_0 - \hat{\mathbf{b}}_0, \tag{4.26}$$

[2]Remember that we label statement instances by index values. Here, $\mathbf{i}_0$ and $\mathbf{j}_0$ denote the index points of **L** corresponding to the iteration points $\hat{\mathbf{i}}_0$ and $\hat{\mathbf{j}}_0$, respectively.

where $\mathbf{d} = \hat{\jmath} - \hat{\imath}$. Thus, all dependence distances in the uniform dependence case satisfy Equation (4.26). In particular, the distance $\mathbf{d}_0$ satisfies (4.26). This means the pair of iteration points $(\hat{\imath}_0, \hat{\jmath}_0)$ satisfies the dependence equation, since

$$(\hat{\jmath}_0 - \hat{\imath}_0)\hat{\mathbf{A}} = \mathbf{d}_0\hat{\mathbf{A}} = \hat{\mathbf{a}}_0 - \hat{\mathbf{b}}_0.$$

The same pair obviously satisfies the dependence constraints (4.25), since they *are* iteration points of the loop nest. Furthermore, we have $\hat{\imath}_0 \preceq \hat{\jmath}_0$ (with $\hat{\imath}_0 = \hat{\jmath}_0$ possible only if $S < T$), since $\mathbf{d}_0 \succeq \mathbf{0}$ (with $\mathbf{d}_0 = \mathbf{0}$ possible only if $S < T$), by Theorem 4.5. Hence, the instance $T(\mathbf{j}_0)$ of $T$ depends on the instance $S(\mathbf{i}_0)$ of $S$. (See Theorem 4.7 and the comments following it.) □

The converse to Theorem 4.8 is true in the ideal situation when the iteration space of $\mathbf{L}$ is the entire set $\{\hat{\imath} \in \mathbf{Z}^m : \hat{\imath} \succeq \mathbf{0}\}$.

**Theorem 4.9** *Assume that the iteration space of* $\mathbf{L}$ *consists of all integer* $m$*-vectors* $\hat{\imath} \succeq \mathbf{0}$. *Suppose that a variable* $X(\hat{\mathbf{I}}\hat{\mathbf{A}} + \hat{\mathbf{a}}_0)$ *of statement* $S$ *and a variable* $X(\hat{\mathbf{I}}\hat{\mathbf{B}} + \hat{\mathbf{b}}_0)$ *of statement* $T$ *cause a dependence of* $T$ *on* $S$. *Let* $\mathbf{d}$ *denote a distance vector of the dependence. If an instance* $T(\mathbf{j})$ *of* $T$ *depends on an instance* $S(\mathbf{i})$ *of* $S$ *whenever* $\hat{\jmath} = \hat{\imath} + \mathbf{d}$, *then this dependence is uniform.*

PROOF. Take any $m$-vector $\hat{\imath} \succeq \mathbf{0}$. Let $\hat{\jmath} = \hat{\imath} + \mathbf{d}$. Then, $\hat{\imath}$ and $\hat{\jmath}$ are iteration points of the loop nest $\mathbf{L}$. Let $\mathbf{i}$ and $\mathbf{j}$ denote the index points of $\mathbf{L}$ corresponding to the iteration points $\hat{\imath}$ and $\hat{\jmath}$, respectively. By hypothesis, $T(\mathbf{j})$ depends on $S(\mathbf{i})$. Hence, the pair $(\hat{\imath}, \hat{\jmath})$ satisfies the dependence equation (4.24), so that we have

$$\hat{\imath}\hat{\mathbf{A}} - (\hat{\imath} + \mathbf{d})\hat{\mathbf{B}} = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0$$

or

$$\hat{\imath}(\hat{\mathbf{A}} - \hat{\mathbf{B}}) = \mathbf{d}\hat{\mathbf{B}} + \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0.$$

This means $\hat{\imath}(\hat{\mathbf{A}} - \hat{\mathbf{B}})$ has the same value for all $\hat{\imath} \succeq \mathbf{0}$. That is possible only if $\hat{\mathbf{A}} = \hat{\mathbf{B}}$ (why?). Hence, the dependence of $T$ on $S$ is uniform. □

In the uniform dependence case, the dependence equation (4.24) reduces to Equation (4.26) where $\mathbf{d} = \hat{\jmath} - \hat{\imath}$. This is a major simplification, since the original equation is a system of $n$ equations in $2m$

variables, whereas (4.26) is a system of $n$ equations in $m$ variables. Letting $\hat{\jmath} = \hat{\imath} + \mathbf{d}$ in (4.25), we write the dependence constraints as

$$\left.\begin{aligned} \mathbf{0} &\le \hat{\imath} \\ \hat{\imath}\hat{\mathbf{Q}} &\le \hat{\mathbf{q}}_0 \\ \mathbf{0} &\le \hat{\imath} + \mathbf{d} \\ (\hat{\imath} + \mathbf{d})\hat{\mathbf{Q}} &\le \hat{\mathbf{q}}_0. \end{aligned}\right\} \tag{4.27}$$

If the loop nest $\mathbf{L}$ is regular, then $\hat{\mathbf{Q}}$ is equal to the identity matrix (see Exercise 4.2.2), and (4.27) simplifies to (explain)

$$\left.\begin{aligned} -\hat{\mathbf{q}}_0 \le \mathbf{d} &\le \hat{\mathbf{q}}_0 \\ \max(\mathbf{0}, -\mathbf{d}) \le \hat{\imath} &\le \min(\hat{\mathbf{q}}_0, \hat{\mathbf{q}}_0 - \mathbf{d}). \end{aligned}\right\} \tag{4.28}$$

There is a unique real solution to (4.26) if $\operatorname{rank}(\hat{\mathbf{A}}) = m$. (See sections I.5.4 and I.5.5.) If $n = m$ and $\hat{\mathbf{A}} = \hat{\mathbf{B}}$ is an $m \times m$ nonsingular matrix, then this unique solution is

$$\mathbf{d} = (\hat{\mathbf{a}}_0 - \hat{\mathbf{b}}_0)\hat{\mathbf{A}}^{-1}. \tag{4.29}$$

If, in addition, the loop nest $\mathbf{L}$ is regular, then the uniform dependence problem becomes a simple problem, since then the system of inequalities (4.28) is trivial to solve. There will be a dependence between statements $S$ and $T$, iff $\mathbf{d}$ given by (4.29) is an integer vector satisfying the first set of inequalities of (4.28), and there is an integer vector $\hat{\imath}$ satisfying the second set. If $\mathbf{d} \succ \mathbf{0}$, or if $\mathbf{d} = \mathbf{0}$ and $S < T$, then this dependence is a dependence of $T$ on $S$. If $\mathbf{d} \prec \mathbf{0}$, then it is a dependence of $S$ on $T$.

Remember that one may have a uniform dependence with many distance vectors (Example I.5.10), and a unique distance vector is not necessarily uniform (Exercise 3).

---

**Example 4.5** Consider the possibility of dependence between statements $S$ and $T$ in the loop nest:

```
L1 :    do I1 = 0, 200, 1
L2 :       do I2 = 0, 100 + I1, 1
   S :        X(3I1 + I2 + 5, I1 + 2I2 + 3) = ···
   T :        ··· = ··· X(3I1 + I2 + 1, I1 + 2I2) ···
           enddo
        enddo
```

(Here, the index and iteration variables are identical.) The coefficient matrix **A** of the array element of $S$ and the coefficient matrix **B** of the array element of $T$ are given by

$$\mathbf{A} = \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}.$$

Since $\mathbf{A} = \mathbf{B}$, if there is any dependence it will be uniform. The dependence equation (4.26) is

$$(d_1, d_2) \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix} = (4, 3).$$

There is a unique solution:

$$\begin{aligned} (d_1, d_2) &= (4, 3) \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}^{-1} \\ &= (4, 3) \begin{pmatrix} 2/5 & -1/5 \\ -1/5 & 3/5 \end{pmatrix} \\ &= (1, 1). \end{aligned}$$

Since it is an integer vector, we see next if the dependence constraints are satisfied. The constraints for this loop nest are

$$\left.\begin{array}{l} 0 \le i_1 \le 200 \\ 0 \le i_2 \le 100 + i_1 \\ 0 \le j_1 \le 200 \\ 0 \le j_2 \le 100 + j_1. \end{array}\right\}$$

Writing

$$(j_1, j_2) = (i_1, i_2) + (d_1, d_2) = (i_1 + 1, i_2 + 1),$$

we get the system

$$\left.\begin{array}{l} 0 \le i_1 \le 200 \\ 0 \le i_2 \le 100 + i_1 \\ 0 \le i_1 + 1 \le 200 \\ 0 \le i_2 + 1 \le 101 + i_1, \end{array}\right\}$$

which reduces to

$$\left.\begin{array}{l} 0 \le i_1 \le 199 \\ 0 \le i_2 \le 100 + i_1. \end{array}\right\}$$

Thus, $T(i_1+1, j_1+1)$ depends on $S(i_1, j_1)$ for each index point $(i_1, i_2)$ that satisfies the last two inequalities, and this dependence is uniform with a unique distance vector $(1, 1)$.

---

### Special Case 2. Diagonal (Normalized) Coefficient Matrices

Let $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ be $m \times m$ diagonal matrices. (Thus, $n = m$ and $X$ is an $m$-dimensional array.) Since $\hat{\mathbf{A}} = \Theta\mathbf{P}^{-1}\mathbf{A}$ and $\hat{\mathbf{B}} = \Theta\mathbf{P}^{-1}\mathbf{B}$, we get this case iff $\mathbf{P}^{-1}\mathbf{A}$ and $\mathbf{P}^{-1}\mathbf{B}$ are $m \times m$ diagonal matrices. In a rectangular loop nest, the condition is equivalent to requiring that $\mathbf{A}$ and $\mathbf{B}$ be diagonal (since then $\mathbf{P}$ is the identity matrix). An example of this case is provided by the variables $X(2I_1-1, 3I_2-1, I_3)$ and $X(2I_1+1, I_2+2, 5)$ in a rectangular triple loop, since their coefficient matrices are $\begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, respectively. Let

$$\hat{\mathbf{A}} = \mathbf{E} = \text{diag}(e_1, e_2, \ldots, e_m)$$
$$\hat{\mathbf{B}} = \mathbf{F} = \text{diag}(f_1, f_2, \ldots, f_m).$$

We assume that for each $r$ in $1 \le r \le m$, at least one of $e_r$ and $f_r$ is nonzero.[3] The dependence equation (4.24) can be written as

$$\hat{\imath}\mathbf{E} - \hat{\jmath}\mathbf{F} = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0. \tag{4.30}$$

It is equivalent to a system of $m$ mutually independent two-variable scalar equations:

$$e_r\hat{\imath}_r - f_r\hat{\jmath}_r = \hat{b}_{r0} - \hat{a}_{r0} \qquad (1 \le r \le m). \tag{4.31}$$

These equations can be solved easily by Algorithm 2.1 and Theorem 2.8. For each $r$, apply Algorithm 2.1 to find $g_r = \gcd(e_r, f_r)$, and two integers $\hat{\imath}_{r0}$ and $\hat{\jmath}_{r0}$ such that

$$e_r\hat{\imath}_{r0} - f_r\hat{\jmath}_{r0} = g_r.$$

By Theorem 2.8, the general solution $(\hat{\imath}_r, \hat{\jmath}_r)$ to (4.31) can be put in the form

$$(\hat{\imath}_r, \hat{\jmath}_r) = (\alpha_{r0} + \alpha_r t_r, \beta_{r0} + \beta_r t_r),$$

[3]This restriction is not really necessary; its only purpose is to avoid trivial equations of the form "0 = constant" in this discussion.

where

$$\alpha_{r0} = (\hat{b}_{r0} - \hat{a}_{r0})\hat{\imath}_{r0}/g_r, \qquad \alpha_r = f_r/g_r,$$
$$\beta_{r0} = (\hat{b}_{r0} - \hat{a}_{r0})\hat{\jmath}_{r0}/g_r, \qquad \beta_r = e_r/g_r,$$

and $t_r$ is an integer parameter. Then, the solution to (4.30) is

$$\hat{\imath} = (\alpha_{10} + \alpha_1 t_1, \alpha_{20} + \alpha_2 t_2, \ldots, \alpha_{m0} + \alpha_m t_m)$$
$$\hat{\jmath} = (\beta_{10} + \beta_1 t_1, \beta_{20} + \beta_2 t_2, \ldots, \beta_{m0} + \beta_m t_m).$$

After substitution in the dependence constraints (4.25), we get the following system of inequalities involving the parameters $t_r$:

$$\begin{aligned}
\mathbf{0} &\le (\alpha_{10} + \alpha_1 t_1, \alpha_{20} + \alpha_2 t_2, \ldots, \alpha_{m0} + \alpha_m t_m), \\
&\quad (\alpha_{10} + \alpha_1 t_1, \alpha_{20} + \alpha_2 t_2, \ldots, \alpha_{m0} + \alpha_m t_m)\hat{\mathbf{Q}} \le \hat{\mathbf{q}}_0, \\
\mathbf{0} &\le (\beta_{10} + \beta_1 t_1, \beta_{20} + \beta_2 t_2, \ldots, \beta_{m0} + \beta_m t_m), \\
&\quad (\beta_{10} + \beta_1 t_1, \beta_{20} + \beta_2 t_2, \ldots, \beta_{m0} + \beta_m t_m)\hat{\mathbf{Q}} \le \hat{\mathbf{q}}_0.
\end{aligned}$$

This can be written as

$$-\alpha_{r0} \le \alpha_r t_r \qquad (1 \le r \le m), \tag{4.32}$$
$$-\beta_{r0} \le \beta_r t_r \qquad (1 \le r \le m), \tag{4.33}$$
$$(\alpha_1 t_1, \ldots, \alpha_m t_m)\hat{\mathbf{Q}} \le \hat{\mathbf{q}}_0 - (\alpha_{10}, \ldots, \alpha_{m0})\hat{\mathbf{Q}} \tag{4.34}$$
$$(\beta_1 t_1, \ldots, \beta_m t_m)\hat{\mathbf{Q}} \le \hat{\mathbf{q}}_0 - (\beta_{10}, \ldots, \beta_{m0})\hat{\mathbf{Q}}. \tag{4.35}$$

Each inequality of (4.32) and (4.33) involves at most one variable, and provides a lower or upper bound for that variable in case its coefficient is nonzero. Since $\hat{\mathbf{Q}}$ is an upper triangular matrix, $t_1$ is the only variable that may appear in the first inequality of (4.34), $t_1$ and $t_2$ are the only variables that may appear in the second inequality, and so on. The system (4.35) also has similar properties. Thus, the system of inequalities (4.32)–(4.35) is easier to solve than a general system since partial elimination has already been done.

This dependence problem becomes a simple problem when the loop nest **L** is regular. Since then $\hat{\mathbf{Q}}$ is the identity matrix and

$$\hat{\mathbf{q}}_0 = ((q_{10} - p_{10})/\theta_1, (q_{20} - p_{20})/\theta_2, \cdots, (q_{m0} - p_{m0})/\theta_m)$$

(see (4.11)), the above system of inequalities simplifies to

$$\left.\begin{aligned} -\alpha_{r0} \le \alpha_r t_r &\le (q_{r0} - p_{r0})/\theta_r - \alpha_{r0} \\ -\beta_{r0} \le \beta_r t_r &\le (q_{r0} - p_{r0})/\theta_r - \beta_{r0} \end{aligned}\right\} \quad (1 \le r \le m). \tag{4.36}$$

Such a system is trivial to solve. Algorithm I.5.3 can be easily modified to provide a complete solution to this simple problem.

---

**Example 4.6** Consider the dependence problem posed by the two array elements shown in the following loop nest:

```
L1 :      do I1 = 0, 100, 1
L2 :         do I2 = 0, 100, 1
L3 :            do I3 = 0, 100, 1
   S :             X(2I1 - 1, 3I2 - 1, I3) = ···
   T :             ··· = ··· X(2I1 + 1, I2 + 2, 5) ···
                enddo
             enddo
          enddo
```

As in Example 4.5, we will work with index variables, since they are identical to the corresponding iteration variables. The dependence equations are

$$2i_1 - 2j_1 = 2 \tag{4.37}$$

$$3i_2 - j_2 = 3 \tag{4.38}$$

$$i_3 = 5, \tag{4.39}$$

and the dependence constraints are

$$0 \le i_1 \le 100 \quad \text{and} \quad 0 \le j_1 \le 100, \tag{4.40}$$

$$0 \le i_2 \le 100 \quad \text{and} \quad 0 \le j_2 \le 100, \tag{4.41}$$

$$0 \le i_3 \le 100 \quad \text{and} \quad 0 \le j_3 \le 100. \tag{4.42}$$

It is clear that the main dependence problem can be partitioned into three smaller problems, since the variables $i_1, j_1$ appear only in (4.37) and (4.40), variables $i_2, j_2$ only in (4.38) and (4.41), and variables $i_3, j_3$ only in (4.39) and (4.42). We can use Algorithm 2.2 to solve each of the three problems separately. For $r = 1, 2, 3$, we find the set

$\Psi_r$ consisting of all solutions $(i_r, j_r)$ to the corresponding equation that satisfy the corresponding set of constraints. Each set $\Psi_r$ will be partitioned into three subsets shown below:

$$\begin{aligned}\Psi_{r,1} &= \{(i_r, j_r) \in \Psi_r : i_r < j_r\} \\ \Psi_{r,-1} &= \{(i_r, j_r) \in \Psi_r : i_r > j_r\} \\ \Psi_{r,0} &= \{(i_r, j_r) \in \Psi_r : i_r = j_r\}.\end{aligned}$$

The general solution to (4.37) is $(i_1, j_1) = (t_1 + 1, t_1)$, where $t_1$ is an integer parameter. Using the constraints (4.40), we get

$$\begin{aligned}\Psi_1 &= \{(t_1 + 1, t_1) : 0 \le t_1 \le 99\} \\ \Psi_{1,1} &= \emptyset \\ \Psi_{1,-1} &= \{(t_1 + 1, t_1) : 0 \le t_1 \le 99\} \\ \Psi_{1,0} &= \emptyset.\end{aligned}$$

The general solution to (4.38) is $(i_2, j_2) = (t_2, 3t_2 - 3)$, where $t_2$ is an integer parameter. Using the constraints (4.41), we get

$$\begin{aligned}\Psi_2 &= \{(t_2, 3t_2 - 3) : 1 \le t_2 \le 34\} \\ \Psi_{2,1} &= \{(t_2, 3t_2 - 3) : 2 \le t_2 \le 34\} \\ \Psi_{2,-1} &= \{(t_2, 3t_2 - 3) : t_2 = 1\} = \{(1, 0)\} \\ \Psi_{2,0} &= \emptyset.\end{aligned}$$

The general solution to (4.39) is $(i_3, j_3) = (5, t_3)$, where $t_3$ is an integer parameter. Using the constraints (4.42), we get

$$\begin{aligned}\Psi_3 &= \{(5, t_3) : 0 \le t_3 \le 100\} \\ \Psi_{3,1} &= \{(5, t_3) : 6 \le t_3 \le 100\} \\ \Psi_{3,-1} &= \{(5, t_3) : 0 \le t_3 \le 4\} \\ \Psi_{3,0} &= \{(5, 5)\}.\end{aligned}$$

Suppose we want to know if statement $T$ depends on statement $S$ with the direction vector $(1, -1, 0)$. Then the question is whether there is an integer solution $(i_1, j_1, i_2, j_2, i_3, j_3)$ to the system of equations (4.37)-(4.39) satisfying the constraints (4.40)-(4.42), such that

$$i_1 < j_1, i_2 > j_2, i_3 = j_3.$$

The set of all such solutions is

$$\{(i_1, j_1, i_2, j_2, i_3, j_3) : (i_1, j_1) \in \Psi_{1,1}, (i_2, j_2) \in \Psi_{2,-1}, (i_3, j_3) \in \Psi_{3,0}\}$$

which is clearly empty. Hence, $T$ does not depend on $S$ with the direction vector $(1, -1, 0)$.

Next, suppose we want to know if statement $S$ depends on statement $T$ with the direction vector $(1, -1, 0)$. Then the question is whether there is an integer solution $(i_1, j_1, i_2, j_2, i_3, j_3)$ to the system of equations (4.37)-(4.39) satisfying the constraints (4.40)-(4.42), such that

$$j_1 < i_1, j_2 > i_2, j_3 = i_3.$$

(Note that $(i_1, i_2, i_3)$ goes with $S$ and $(j_1, j_2, j_3)$ with $T$, so that their roles are now reversed.) The set of all such solutions is

$$\{(i_1, j_1, i_2, j_2, i_3, j_3) : (i_1, j_1) \in \Psi_{1,-1}, (i_2, j_2) \in \Psi_{2,1}, (i_3, j_3) \in \Psi_{3,0}\}.$$

This set is nonempty so that $S$ depends on $T$ with the direction vector $(1, -1, 0)$. In fact, all instance pairs $(S(i_1, i_2, i_3), T(j_1, j_2, j_3))$, such that $i_1 > j_1, i_2 < j_2, i_3 = j_3$, and $S(i_1, i_2, i_3)$ depends on $T(j_1, j_2, j_3)$, form the set

$$\{(S(t_1 + 1, t_2, 5), T(t_1, 3t_2 - 3, 5)) : 0 \le t_1 \le 99, 2 \le t_2 \le 34\}.$$

The number of such pairs is $100 \times 33 = 3300$. The set of distance vectors for this dependence of $S$ on $T$ is

$$\{(1, 3 - 2t_2, 0) : 2 \le t_2 \le 34\} = \{(1, -1, 0), (1, -3, 0), \ldots, (1, -65, 0)\}.$$

---

## Special Case 3. A Generalization of Cases 1 and 2

Let $\hat{\mathbf{A}} = \mathbf{EC}$ and $\hat{\mathbf{B}} = \mathbf{FC}$, where $\mathbf{E}$ and $\mathbf{F}$ are $m \times m$ diagonal matrices, and $\mathbf{C}$ is an $m \times n$ matrix. It is clear that this last case includes the first two. We get $\hat{\mathbf{A}} = \hat{\mathbf{B}}$ when $\mathbf{E}$ and $\mathbf{F}$ are equal to the identity matrix, while $\hat{\mathbf{A}} = \mathbf{E}$ and $\hat{\mathbf{B}} = \mathbf{F}$ when $\mathbf{C}$ is the identity matrix. An example of this case is provided by the variables $X(I_3, 2I_1 - 1, 3I_2 - 1)$ and

$X(5, 2I_1 + 1, I_2 + 2)$ in a rectangular triple loop, since their coefficient matrices have the forms:

$$\begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & 3 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} = \text{diag}(2, 3, 1) \cdot \mathbf{C}$$

$$\begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} = \text{diag}(2, 1, 0) \cdot \mathbf{C}.$$

In general, the analysis of Special Case 2 can be extended to include variables whose coefficient matrices are diagonal matrices postmultiplied by a permutation matrix.

As in Special Case 2, let

$$\mathbf{E} = \text{diag}(e_1, e_2, \ldots, e_m)$$
$$\mathbf{F} = \text{diag}(f_1, f_2, \ldots, f_m),$$

and assume that for each $r$ in $1 \le r \le m$, at least one of $e_r$ and $f_r$ is nonzero. Then, we may choose the matrix $\mathbf{C}$ in such a way that $\gcd(e_r, f_r) = 1$ for each $r$. The dependence equation (4.24) has the form

$$\hat{\boldsymbol{\imath}}(\mathbf{EC}) - \hat{\boldsymbol{\jmath}}(\mathbf{FC}) = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0.$$

It can be broken up into two separate systems:

$$\hat{\boldsymbol{\imath}}\mathbf{E} - \hat{\boldsymbol{\jmath}}\mathbf{F} = \mathbf{k} \tag{4.43}$$

$$\mathbf{kC} = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0. \tag{4.44}$$

For a given $\mathbf{k} = (k_1, k_2, \ldots, k_m)$, the general solution to (4.43) can be written as

$$\hat{\boldsymbol{\imath}} = (\hat{\imath}_{10}k_1 + f_1t_1, \hat{\imath}_{20}k_2 + f_2t_2, \ldots, \hat{\imath}_{m0}k_m + f_mt_m)$$
$$\hat{\boldsymbol{\jmath}} = (\hat{\jmath}_{10}k_1 + e_1t_1, \hat{\jmath}_{20}k_2 + e_2t_2, \ldots, \hat{\jmath}_{m0}k_m + e_mt_m),$$

where $t_1, t_2, \ldots, t_m$ are integer parameters, and

$$e_r\hat{\imath}_{r0} - f_r\hat{\jmath}_{r0} = 1 \qquad (1 \le r \le m).$$

The constraints (4.25) can also be expressed in terms of $k_1, k_2, \ldots, k_m$ and $t_1, t_2, \ldots, t_m$.

This kind of problem in a regular loop nest (where $\hat{\mathbf{Q}}$ is the identity matrix) becomes a simple problem, if the matrix $\mathbf{C}$ is such that Equation (4.44) has no solution or a unique solution in $\mathbf{k}$. We omit the details; see Example I.5.12.

To see how to test whether the matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ satisfy the condition defining Special Case 3, see Exercise I.5.6.1.

**EXERCISES 4.6**

1. Consider the dependence equation

$$\hat{\imath}\hat{\mathbf{A}} - \hat{\jmath}\hat{\mathbf{B}} = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0.$$

If $\hat{\mathbf{A}} = \hat{\mathbf{B}}$, then for any given solution $(\hat{\imath}_0, \hat{\jmath}_0)$ and any given $m$-vector $\mathbf{h}$, $(\hat{\imath}_0 + \mathbf{h}, \hat{\jmath}_0 + \mathbf{h})$ is also a solution. Prove the converse: if the equation has this property, then $\hat{\mathbf{A}} = \hat{\mathbf{B}}$.

2. Find conditions on the iteration space of L such that it is finite and Theorem 4.9 is still valid.

3. Give an example of a nonuniform dependence with a unique distance vector.

4. Find the distance vectors, if any, of the dependence of $T$ on $S$, $S$ on $T$, and $S$ on $S$ in the following programs:

(a)
```
L1 :    do I1 = 0,200,2
L2 :       do I2 = 100 + I1,0,-3
S :           X(3I1 + I2 + 5,I1 + 2I2 + 3) = ···
T :           ··· = ··· X(3I1 + I2 + 1,I1 + 2I2) ···
           enddo
        enddo
```

(b)
```
L1 :    do I1 = 0,100,3
L2 :       do I2 = I1,50 + I1,2
S :           X(2I1 + 3I2 + 12) = ···
T :           ··· = ··· X(2I1 + 3I2 - 5) ···
           enddo
        enddo
```

5. Decide if $T$ depends on $S$ at level 2 in the program

```
L1 :    do I1 = 0,100,1
L2 :       do I2 = 0,50 + I1,1
S :           X(2I1 + 3I2 + 12) = ···
T :           ··· = ··· X(2I1 + 3I2 - 5) ···
           enddo
        enddo
```

6. In the following programs, study the dependence of $T$ on $S$, $S$ on $T$, $S$ on $S$, and $T$ on $T$. Find all distance and direction vectors, and levels of dependence. For each valid direction vector, find the set of all pairs of statement instances that are involved in the dependence.

(a)
```
L1 :   do I1 = 10, 100, 2
L2 :     do I2 = 100, 0, −2
L3 :       do I3 = 20, 100, 1
S :          X(2I1 − 1, 3I2 − 1, I3) = · · ·
T :          · · · = · · · X(2I1 + 1, I2 + 2, 5) · · ·
           enddo
         enddo
       enddo
```

(b)
```
L1 :   do I1 = 1, 100, 2
L2 :     do I2 = 100, 0, −3
S :        X(6I1 − 1, 2I2 − 1) = · · ·
T :        · · · = · · · X(4I1 + 9, 2I2 + 9) · · ·
         enddo
       enddo
```

7. Discuss the possibility of extending the concepts and results of this section to a program that is not a perfect nest.

# Chapter 5

# General Program

## 5.1 Introduction

A sequence of loops forms a *nest* if each loop (except the first) is totally included within the previous loop. So far in the book, we have studied the dependence problem in a *perfect* loop nest, where the body of each loop (except the last) is the next loop in the sequence. The parameters we used to characterize a perfect loop nest (initial and final vectors, initial and final matrices, and stride matrix) can also be used to characterize an ordinary loop nest.

The model for this chapter is a program that consists of loops and assignment statements, where the loops are arbitrarily (but legally) nested. If necessary, we create a trivial loop with one iteration so that the whole program is a loop. This does not alter the concepts of dependence in any way, but helps simplify the notation. Each given statement in the program *determines* a loop nest; it is the sequence of all loops in the program containing that statement (counted from the outermost loop inward). If the program is a perfect loop nest, then the nests determined by all statements are the same. When we compare two statements in the current program model to test for dependence between them, we have to deal with two possibly different loop nests. This is how the dependence problem is to be generalized as we move from a perfect nest of loops to an arbitrary program.

Much of the needed formalism is already in place. We show in Section 5.2 how the dependence concepts can be easily extended from a

perfect loop nest to a general program. The linear dependence problem in the general setting is formulated in Section 5.3. In Section 5.4, we discuss the generalized gcd test that is the basic test in dependence analysis.

## 5.2 Dependence Concepts

First, take any single assignment statement $S$ in the program. Let $m_S$ denote the number of loops containing $S$ and $\mathbf{L}_S$ the loop nest determined by $S$. For $\mathbf{L}_S$, let $\mathbf{I}_S$ denote the index vector, $\hat{\mathbf{I}}_S$ the iteration vector, $\mathbf{p}_{S0}$ and $\mathbf{q}_{S0}$ the initial and final vectors, $\mathbf{P}_S$ and $\mathbf{Q}_S$ the initial and final matrices, and $\Theta_S$ the stride matrix. The normalized final vector and the normalized final matrix of $\mathbf{L}_S$ are denoted by $\hat{\mathbf{q}}_{S0}$ and $\hat{\mathbf{Q}}_S$, respectively, and they are computed using (4.10):

$$\left.\begin{aligned} \hat{\mathbf{Q}}_S &= \Theta_S \mathbf{P}_S^{-1} \mathbf{Q}_S \Theta_S^{-1} \\ \hat{\mathbf{q}}_{S0} &= \left(\mathbf{q}_{S0} - \mathbf{p}_{S0} \mathbf{P}_S^{-1} \mathbf{Q}_S\right) \Theta_S^{-1}. \end{aligned}\right\} \tag{5.1}$$

Next, consider any two, not necessarily distinct, assignment statements $S$ and $T$ in the given program. If $S$ lexically precedes $T$ in the program, we write $S < T$. The meaning of the notation $S \leq T$ is then clear, and it is obvious that $\leq$ is a total order in the set of all assignment statements.

Let $m \equiv m_{ST}$ denote the number of loops that contain both statements $S$ and $T$, and $\mathbf{L}$ the nest formed by these loops. We have $m \leq \min(m_S, m_T)$. Since the outermost $m$ loops of $\mathbf{L}_S$ and $\mathbf{L}_T$ are the same, the first $m$ elements of $\mathbf{p}_{S0}$ and $\mathbf{p}_{T0}$ are identical, and so are the first $m$ elements of $\mathbf{q}_{S0}$ and $\mathbf{q}_{T0}$. Also, the $m \times m$ submatrices formed by the topmost $m$ rows and the leftmost $m$ columns of the matrices $\mathbf{P}_S$ and $\mathbf{P}_T$ are the same, and so are the similar submatrices of the matrices $\mathbf{Q}_S$ and $\mathbf{Q}_T$.

Let us label the loops of the program $L_1, L_2, \ldots, L_{m_S+m_T-m}$, such that

$$\begin{aligned} \mathbf{L} &= (L_1, L_2, \ldots, L_m) \\ \mathbf{L}_S &= (L_1, L_2, \ldots, L_m, L_{m+1}, \ldots, L_{m_S}) \\ \mathbf{L}_T &= (L_1, L_2, \ldots, L_m, L_{m_S+1}, \ldots, L_{m_S+m_T-m}). \end{aligned}$$

A value $\mathbf{i} = (i_1, i_2, \ldots, i_m, i_{m+1}, \ldots, i_{m_S})$ of the index vector $\mathbf{I}_S$ determines an instance of statement $S$ that is denoted by $S(\mathbf{i})$, and a

value $\mathbf{j} = (j_1, j_2, \ldots, j_m, j_{m_S+1}, \ldots, j_{m_S+m_T-m})$ of the index vector $\mathbf{I}_T$ determines an instance of statement $T$ that is denoted by $T(\mathbf{j})$. The *distance* from the instance $S(\mathbf{i})$ to the instance $T(\mathbf{j})$ is defined to be the $m$-vector $(\hat{j}_1 - \hat{\imath}_1, \hat{j}_2 - \hat{\imath}_2, \ldots, \hat{j}_m - \hat{\imath}_m)$, where

$$\hat{\imath} = (\hat{\imath}_1, \ldots, \hat{\imath}_m, \hat{\imath}_{m+1}, \ldots, \hat{\imath}_{m_S})$$
$$\hat{\jmath} = (\hat{\jmath}_1, \ldots, \hat{\jmath}_m, \hat{\jmath}_{m_S+1}, \ldots, \hat{\jmath}_{m_S+m_T-m})$$

are the iteration vectors corresponding to $\mathbf{i}$ and $\mathbf{j}$, respectively. Thus, the distance is defined in terms of the first $m$ elements of the iteration vectors of the loop nests $\mathbf{L}_S$ and $\mathbf{L}_T$. In other words, the distance is defined[1] by the iterations of the common nest $\mathbf{L}$. An extension of Lemma 4.4 to the general case is stated below without proof.

**Lemma 5.1** *Consider two assignment statements $S$ and $T$ in the model program. Let $\mathbf{d}$ denote the distance from an instance $S(\mathbf{i})$ of $S$ to an instance $T(\mathbf{j})$ of $T$. In the sequential execution of the program, $S(\mathbf{i})$ is executed before $T(\mathbf{j})$ iff one of the following two conditions holds:*

(a) $\mathbf{d} \succ \mathbf{0}$,

(b) $\mathbf{d} = \mathbf{0}$ *and* $S < T$.

The dependence relation $\delta$ between statements in the general program can be defined as in Section 4.3. Descriptions of other associated concepts can be extended almost word for word to the general case. We need only remember, for example, that a distance vector $\mathbf{d}$ of dependence of $T$ on $S$, defined as the distance from an instance $S(\mathbf{i})$ to an instance $T(\mathbf{j})$, is computed as the difference between those parts of $\hat{\jmath}$ and $\hat{\imath}$ that relate to the loop nest $\mathbf{L}$ common to $S$ and $T$. The corresponding direction vector $\boldsymbol{\sigma}$ is the sign of $\mathbf{d}$, and the dependence level $\ell$ is the level of $\mathbf{d}$ (or of $\boldsymbol{\sigma}$). We state here the generalizations of Theorem 4.5 and the two corollaries, for the sake of completeness.

**Theorem 5.2** *If $\mathbf{d}$ is a distance vector for the dependence of statement $T$ on statement $S$ in the general program, then $\mathbf{d} \succeq \mathbf{0}$. The equality $\mathbf{d} = \mathbf{0}$ is possible only if $S < T$.*

[1]Note that in a general program, the iteration points $\hat{\imath}$ and $\hat{\jmath}$ need not have the same number of elements, so that the difference $\hat{\jmath} - \hat{\imath}$ may not even make sense.

**Corollary 1** *If $\boldsymbol{\sigma}$ is a direction vector for the dependence of statement $T$ on statement $S$ in the general program, then $\boldsymbol{\sigma} \succeq \mathbf{0}$. The equality $\boldsymbol{\sigma} = \mathbf{0}$ is possible only if $S < T$.*

**Corollary 2** *If $\ell$ is a dependence level for the dependence of statement $T$ on statement $S$ in the general program, then $1 \le \ell \le m+1$ (where $m$ is the number of loops that contain both $S$ and $T$). The value $\ell = m+1$ is possible only if $S < T$.*

---

**Example 5.1** Consider the program

```
L1 :      do I1 = 1, 100, 1
L2 :        do I2 = 1, I1, 2
L3 :          do I3 = I1, I1 + I2, 1
   S :          X(2I1 − 1, 3I2 + 1, 2I3) = · · ·
              enddo
L4 :          do I4 = 100, 0, −3
L5 :            do I5 = I1, I4, 1
   T :            · · · = · · · X(2I1 + 1, 4I2 + 6, I4 + I5) · · ·
                enddo
              enddo
            enddo
          enddo
```

Statement $S$ determines the loop nest $\mathbf{L}_S = (L_1, L_2, L_3)$ with $m_S = 3$ loops. For this nest, we have:

$$
\begin{aligned}
&p_1 = p_{10} = 1,\\
&q_1 = q_{10} = 100,\\
&p_2 = p_{20} + p_{21} I_1 = 1 + 0 \times I_1,\\
&q_2 = q_{20} + q_{21} I_1 = 0 + 1 \times I_1,\\
&p_3 = p_{30} + p_{31} I_1 + p_{32} I_2 = 0 + 1 \times I_1 + 0 \times I_2,\\
&q_3 = q_{30} + q_{31} I_1 + q_{32} I_2 = 0 + 1 \times I_1 + 1 \times I_2,\\
&\theta_1 = 1, \theta_2 = 2, \theta_3 = 1.
\end{aligned}
$$

Hence, the initial and final vectors are

$$
\begin{aligned}
\mathbf{p}_{S0} &= (p_{10}, p_{20}, p_{30}) = (1, 1, 0),\\
\mathbf{q}_{S0} &= (q_{10}, q_{20}, q_{30}) = (100, 0, 0);
\end{aligned}
$$

and the initial, final, and stride matrices are

$$\mathbf{P}_S = \begin{pmatrix} 1 & -p_{21} & -p_{31} \\ 0 & 1 & -p_{32} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\mathbf{Q}_S = \begin{pmatrix} 1 & -q_{21} & -q_{31} \\ 0 & 1 & -q_{32} \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\mathbf{\Theta}_S = \begin{pmatrix} \theta_1 & 0 & 0 \\ 0 & \theta_2 & 0 \\ 0 & 0 & \theta_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

We compute the normalized final vector and matrix:

$$\hat{\mathbf{q}}_{S0} = \left(\mathbf{q}_{S0} - \mathbf{p}_{S0}\mathbf{P}_S^{-1}\mathbf{Q}_S\right)\mathbf{\Theta}_S^{-1} = (99, 0, 1),$$

$$\hat{\mathbf{Q}}_S = \mathbf{\Theta}_S\mathbf{P}_S^{-1}\mathbf{Q}_S\mathbf{\Theta}_S^{-1} = \begin{pmatrix} 1 & -1/2 & 0 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{pmatrix}.$$

The iteration vector $\hat{\mathbf{I}}_S = (\hat{I}_1, \hat{I}_2, \hat{I}_3)$ satisfies the constraints:

$$\left.\begin{aligned} \mathbf{0} &\le \hat{\mathbf{I}}_S \\ \hat{\mathbf{I}}_S\hat{\mathbf{Q}}_S &\le \hat{\mathbf{q}}_{S0}. \end{aligned}\right\}$$

Thus, the iteration space of the loop nest $\mathbf{L}_S$ is the set of all integer vectors $(\hat{\imath}_1, \hat{\imath}_2, \hat{\imath}_3) \ge (0,0,0)$ such that

$$\left.\begin{aligned} \hat{\imath}_1 &\le 99 \\ -\hat{\imath}_1/2 + \hat{\imath}_2 &\le 0 \\ -2\hat{\imath}_2 + \hat{\imath}_3 &\le 1, \end{aligned}\right\}$$

that is, the set of all integer 3-vectors $(\hat{\imath}_1, \hat{\imath}_2, \hat{\imath}_3)$ such that

$$\left.\begin{aligned} 0 \le \hat{\imath}_1 &\le 99 \\ 0 \le \hat{\imath}_2 &\le \hat{\imath}_1/2 \\ 0 \le \hat{\imath}_3 &\le 2\hat{\imath}_2 + 1. \end{aligned}\right\}$$

Given an index point $(i_1, i_2, i_3)$ of $\mathbf{L}_S$, the corresponding iteration point $(\hat{\imath}_1, \hat{\imath}_2, \hat{\imath}_3)$ is computed using the equation

$$\hat{\mathbf{I}} = (\mathbf{IP} - \mathbf{p}_0)\mathbf{\Theta}^{-1},$$

so that

$$
\begin{aligned}
(\hat{i}_1, \hat{i}_2, \hat{i}_3) &= [(i_1, i_2, i_3)\mathbf{P}_S - \mathbf{p}_{S0}]\,\boldsymbol{\Theta}_S^{-1} \\
&= \left[ (i_1, i_2, i_3) \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - (1,1,0) \right] \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= (i_1 - 1, (i_2 - 1)/2, i_3 - i_1)\,.
\end{aligned}
$$

Statement $T$ determines the loop nest $\mathbf{L}_T = (L_1, L_2, L_4, L_5)$ with $m_T = 4$ loops. For this nest, the initial and final vectors are

$$\mathbf{p}_{T0} = (1, 1, 100, 0), \qquad \mathbf{q}_{T0} = (100, 0, 0, 0);$$

and the initial, final, and stride matrices are

$$
\mathbf{P}_T = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{Q}_T = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix},
$$

$$
\boldsymbol{\Theta}_T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.
$$

Note that the first two elements of $\mathbf{p}_{S0}$ and $\mathbf{p}_{T0}$ are identical, and the same is true for $\mathbf{q}_{S0}$ and $\mathbf{q}_{T0}$. Also, the $2 \times 2$ submatrix of $\mathbf{P}_{S0}$ at the northwest corner is identical to the similar submatrix of $\mathbf{P}_{T0}$, and the same holds for $\mathbf{Q}_{S0}$ and $\mathbf{Q}_{T0}$. We compute the normalized final vector and matrix:

$$
\hat{\mathbf{q}}_{T0} = \left(\mathbf{q}_{T0} - \mathbf{p}_{T0}\mathbf{P}_T^{-1}\mathbf{Q}_T\right)\boldsymbol{\Theta}_T^{-1} = (99, 0, 100/3, 99),
$$

$$
\hat{\mathbf{Q}}_T = \boldsymbol{\Theta}_T\mathbf{P}_T^{-1}\mathbf{Q}_T\boldsymbol{\Theta}_T^{-1} = \begin{pmatrix} 1 & -1/2 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix}.
$$

| Statement Instance | Iteration Point of $\mathbf{L}_S$ | Iteration Point of $\mathbf{L}_T$ | Iteration Point of $\mathbf{L}$ |
|---|---|---|---|
| $S(\mathbf{i}_1) = S(31,29,58)$ | $(30,14,27)$ | — | $(30,14)$ |
| $T(\mathbf{j}_1) = T(36,5,70,70)$ | — | $(35,2,10,34)$ | $(35,2)$ |
| $S(\mathbf{i}_2) = S(41,25,61)$ | $(40,12,20)$ | — | $(40,12)$ |

Table 5.1: Three statement instances for Example 5.1.

The iteration space of the loop nest $\mathbf{L}_T$ is the set of all integer 4-vectors $(\hat{\jmath}_1, \hat{\jmath}_2, \hat{\jmath}_4, \hat{\jmath}_5)$ such that

$$\left.\begin{array}{l} 0 \le \hat{\jmath}_1 \le 99 \\ 0 \le \hat{\jmath}_2 \le \hat{\jmath}_1/2 \\ 0 \le \hat{\jmath}_4 \le 33 \\ 0 \le \hat{\jmath}_5 \le 99 - \hat{\jmath}_1 - 3\hat{\jmath}_4. \end{array}\right\}$$

The iteration value $(\hat{\jmath}_1, \hat{\jmath}_2, \hat{\jmath}_4, \hat{\jmath}_5)$ corresponding to an index value $(j_1, j_2, j_4, j_5)$ of $\mathbf{L}_T$ is given by

$$\begin{aligned}(\hat{\jmath}_1, \hat{\jmath}_2, \hat{\jmath}_4, \hat{\jmath}_5) &= [(j_1, j_2, j_4, j_5)\mathbf{P}_T - \mathbf{p}_{T0}]\,\Theta_T^{-1} \\ &= (j_1 - 1, (j_2 - 1)/2, (100 - j_4)/3, j_5 - j_1).\end{aligned}$$

We take three statement instances $S(\mathbf{i}_1), S(\mathbf{i}_2)$, and $T(\mathbf{j}_1)$, where

$$\mathbf{i}_1 = (31,29,58), \quad \mathbf{i}_2 = (41,25,61), \quad \mathbf{j}_1 = (36,5,70,70).$$

The iteration points of different loop nests for these instances are given in Table 5.1. The distance from $S(\mathbf{i}_1)$ to $T(\mathbf{j}_1)$ is $(5,-12) = (35,2) - (30,14)$, and the distance from $T(\mathbf{j}_1)$ to $S(\mathbf{i}_2)$ is $(5,10) = (40,12) - (35,2)$. Since $(5,-12)$ and $(5,10)$ are both lexicographically positive, Lemma 5.1 implies that in the sequential execution of the given program, $S(\mathbf{i}_1)$ is executed before $T(\mathbf{j}_1)$, and $T(\mathbf{j}_1)$ before $S(\mathbf{i}_2)$. That is, the instances $S(\mathbf{i}_1)$, $T(\mathbf{j}_1)$, and $S(\mathbf{i}_2)$ are executed in this order since

$$(30,14) \prec (35,2) \prec (40,12).$$

Next, note that the instance $S(24,19,40)$ of $S$ writes the memory location $X(47,58,80)$, and the instance $T(23,13,52,28)$ of $T$ reads the same location. Thus, there is a dependence between $S$ and $T$. Now,

the iteration point of the loop nest $\mathbf{L}_S$ for the index point $(24, 19, 40)$ is $(23, 9, 16)$. Also, the iteration point of the loop nest $\mathbf{L}_T$ for the index point $(23, 13, 52, 28)$ is $(22, 6, 16, 5)$. Since $(22, 6) \prec (23, 9)$, $T$ reads $X(47, 58, 80)$ before $S$ writes it. Hence, $S$ is anti-dependent on $T$ with a distance vector $(1, 3) = (23, 9) - (22, 6)$. The corresponding direction vector is $(1, 1)$ and the level of dependence is 1.

---

**EXERCISES 5.2**

1. In Example 5.1, express the index values $(i_1, i_2, i_3)$ and $(j_1, j_2, j_4, j_5)$ in terms of the corresponding iteration values. Find descriptions of the index spaces of the loop nests $\mathbf{L}_S$ and $\mathbf{L}_T$.
2. In Example 5.1, list all possible direction vectors and levels that a dependence of $T$ on $S$ may theoretically have. Without doing any heavy calculations, can you rule out any items from either list? Repeat for a dependence of $S$ on $T$.
3. In Example 5.1, let $\mathbf{d} = (\hat{j}_1 - \hat{i}_1, \hat{j}_2 - \hat{i}_2)$ and $\mathbf{d}' = (j_1 - i_1, j_2 - i_2)$. Express $\mathbf{d}'$ in terms of $\mathbf{d}$. How does $\mathbf{d}'$ change if $\mathbf{d}$ is increased by $(0, 1)$?
4. Let $\hat{\mathbf{i}}$ denote an iteration point of the loop nest $\mathbf{L}_S$ and $\hat{\mathbf{j}}$ an iteration point of the loop nest $\mathbf{L}_T$. Let $\mathbf{i}$ denote the index point of $\mathbf{L}_S$ corresponding to $\hat{\mathbf{i}}$, and $\mathbf{j}$ the index point of $\mathbf{L}_T$ corresponding to $\hat{\mathbf{j}}$. Let

$$\mathbf{d} = (\hat{j}_1 - \hat{i}_1, \hat{j}_2 - \hat{i}_2, \ldots, \hat{j}_m - \hat{i}_m)$$
$$\mathbf{d}' = (j_1 - i_1, j_2 - i_2, \ldots, j_m - i_m).$$

   How is $\mathbf{d}'$ related to $\mathbf{d}$? How does $\mathbf{d}'$ change if $\mathbf{d}$ is increased by $(0, \ldots, 0, 1)$?

5. Study the dependence structure of the following program. For each dependence between two statements, find all distance vectors, direction vectors, and dependence levels. Draw a statement dependence graph.

```
L1 :        do I1 = 1, 10, 1
   S1 :        X(I1) = ···
L2 :           do I2 = 10, 1, -1
   S2 :           X(I1 + I2) = ···
               enddo
   S3 :        X(I1 + 1) = ···
L3 :           do I3 = I1, 10, 1
   S4 :           X(I1 + I3) = ···
L4 :              do I4 = I3, I1, -1
   S5 :              X(I1 + I3 + I4) = ···
                  enddo
   S6 :           X(I3) = ···
               enddo
            enddo
```

## 5.3 Dependence Problem

In this section, we state the dependence problem posed by two array-elements in a general program. The problem is described in terms of iteration variables; the description in terms of index variables is left to the reader.

Consider two assignment statements $S$ and $T$ in the program. Let $X$ denote an $n$-dimensional array. Suppose that each statement has a variable that is an element of $X$ with linear subscripts. Since the index vector of the loop nest determined by $S$ is denoted by $\mathbf{I}_S$, this variable of $S$ can be written in the form $X(\mathbf{I}_S\mathbf{A} + \mathbf{a}_0)$, where $\mathbf{A}$ is an $m_S \times n$ integer matrix and $\mathbf{a}_0$ is an integer $n$-vector (see Section 4.4). Similarly, the variable of $T$ can be written as $X(\mathbf{I}_T\mathbf{B} + \mathbf{b}_0)$, where $\mathbf{B}$ is an $m_T \times n$ integer matrix and $\mathbf{b}_0$ is an integer $n$-vector. Let $X\left(\hat{\mathbf{I}}_S\hat{\mathbf{A}} + \hat{\mathbf{a}}_0\right)$ and $X\left(\hat{\mathbf{I}}_T\hat{\mathbf{B}} + \hat{\mathbf{b}}_0\right)$ denote the normalized forms of the two variables. The $m_S \times n$ integer matrix $\hat{\mathbf{A}}$, the $m_T \times n$ integer matrix $\hat{\mathbf{B}}$, and the integer $n$-vectors $\hat{\mathbf{a}}_0$ and $\hat{\mathbf{b}}_0$ are defined by the equations in (4.15):

$$\left.\begin{aligned} \hat{\mathbf{A}} &= \Theta_S \mathbf{P}_S^{-1}\mathbf{A} \\ \hat{\mathbf{a}}_0 &= \mathbf{p}_{S0}\mathbf{P}_S^{-1}\mathbf{A} + \mathbf{a}_0 \\ \hat{\mathbf{B}} &= \Theta_T \mathbf{P}_T^{-1}\mathbf{B} \\ \hat{\mathbf{b}}_0 &= \mathbf{p}_{T0}\mathbf{P}_T^{-1}\mathbf{B} + \mathbf{b}_0. \end{aligned}\right\} \tag{5.2}$$

The instance of the variable $X\left(\hat{\mathbf{I}}_S\hat{\mathbf{A}} + \hat{\mathbf{a}}_0\right)$ for an iteration point

$$\hat{\boldsymbol{\imath}} = (\hat{\imath}_1, \hat{\imath}_2, \ldots, \hat{\imath}_m, \hat{\imath}_{m+1}, \ldots, \hat{\imath}_{m_S})$$

of the loop nest $\mathbf{L}_S$ is $X\left(\hat{\boldsymbol{\imath}}\hat{\mathbf{A}} + \hat{\mathbf{a}}_0\right)$. The instance of $X\left(\hat{\mathbf{I}}_T\hat{\mathbf{B}} + \hat{\mathbf{b}}_0\right)$ for an iteration point

$$\hat{\boldsymbol{\jmath}} = (\hat{\jmath}_1, \hat{\jmath}_2, \ldots, \hat{\jmath}_m, \hat{\jmath}_{m_S+1}, \ldots, \hat{\jmath}_{m_S+m_T-m})$$

of the loop nest $\mathbf{L}_T$ is $X\left(\hat{\boldsymbol{\jmath}}\hat{\mathbf{B}} + \hat{\mathbf{b}}_0\right)$. These two instances represent the same memory location iff $\hat{\boldsymbol{\imath}}\hat{\mathbf{A}} + \hat{\mathbf{a}}_0 = \hat{\boldsymbol{\jmath}}\hat{\mathbf{B}} + \hat{\mathbf{b}}_0$, that is, iff

$$\hat{\boldsymbol{\imath}}\hat{\mathbf{A}} - \hat{\boldsymbol{\jmath}}\hat{\mathbf{B}} = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0. \tag{5.3}$$

The matrix equation (5.3) is the *dependence equation* for the program variables $X\left(\hat{\mathbf{I}}_S\hat{\mathbf{A}} + \hat{\mathbf{a}}_0\right)$ and $X\left(\hat{\mathbf{I}}_T\hat{\mathbf{B}} + \hat{\mathbf{b}}_0\right)$. It consists of $n$ scalar equations, and there are $(m_S + m_T)$ integer variables: the $m_S$ elements of

| Parameter | Type | Size |
|---|---|---|
| $\hat{\mathbf{q}}_{S0}$ | integer vector | $m_S$ |
| $\hat{\mathbf{Q}}_S$ | integer matrix | $m_S \times m_S$ |
| $\hat{\mathbf{q}}_{T0}$ | integer vector | $m_T$ |
| $\hat{\mathbf{Q}}_T$ | integer matrix | $m_T \times m_T$ |
| $\hat{\mathbf{a}}_0$ | integer vector | $n$ |
| $\hat{\mathbf{A}}$ | integer matrix | $m_S \times n$ |
| $\hat{\mathbf{b}}_0$ | integer vector | $n$ |
| $\hat{\mathbf{B}}$ | integer matrix | $m_T \times n$ |

Table 5.2: Parameters for the Dependence Problem.

$\hat{\imath}$ and the $m_T$ elements of $\hat{\jmath}$. Since $\hat{\imath}$ is a value of $\hat{\mathbf{I}}_S$, it must satisfy the inequalities (4.9) for $\mathbf{L}_S$:

$$\left.\begin{array}{l} \mathbf{0} \le \hat{\imath} \\ \hat{\imath}\hat{\mathbf{Q}}_S \le \hat{\mathbf{q}}_{S0}, \end{array}\right\} \tag{5.4}$$

where $\hat{\mathbf{Q}}_S$ is the normalized final matrix and $\hat{\mathbf{q}}_{S0}$ the normalized final vector of $\mathbf{L}_S$. Since $\hat{\jmath}$ is a value of $\hat{\mathbf{I}}_T$, it must satisfy the inequalities (4.9) for $\mathbf{L}_T$:

$$\left.\begin{array}{l} \mathbf{0} \le \hat{\jmath} \\ \hat{\jmath}\hat{\mathbf{Q}}_T \le \hat{\mathbf{q}}_{T0}, \end{array}\right\} \tag{5.5}$$

where $\hat{\mathbf{Q}}_T$ is the normalized final matrix and $\hat{\mathbf{q}}_{T0}$ the normalized final vector of $\mathbf{L}_T$. The inequalities in (5.4) and (5.5) constitute the *dependence constraints* for the program variables $X\left(\hat{\mathbf{I}}_S\hat{\mathbf{A}} + \hat{\mathbf{a}}_0\right)$ and $X\left(\hat{\mathbf{I}}_T\hat{\mathbf{B}} + \hat{\mathbf{b}}_0\right)$ of statements $S$ and $T$. Descriptions of the different parameters for the dependence problem are collected in Table 5.2.

The extensions of Theorem 4.6, its corollaries, and Theorem 4.7 to a general program are stated below without proof.

**Theorem 5.3** *Consider two statements $S$ and $T$ in the general program* **L**. *Let $X\left(\mathbf{I}_S\mathbf{A} + \mathbf{a}_0\right)$ denote a variable of $S$ and $X\left(\mathbf{I}_T\mathbf{B} + \mathbf{b}_0\right)$ a variable of $T$, where $X$ is an $n$-dimensional array. If these variables cause a dependence between $S$ and $T$, then Equation (5.3) has an integer solution $(\hat{\imath}, \hat{\jmath})$ that satisfies (5.4)-(5.5). In each of the following special cases, the solution satisfies an additional condition:*

(a) *If* $S < T$ *and* $S \,\delta\, T$, *then* $(\hat{\imath}_1, \hat{\imath}_2, \ldots, \hat{\imath}_m) \preceq (\hat{\jmath}_1, \hat{\jmath}_2, \ldots, \hat{\jmath}_m)$;

(b) *If* $S = T$ *and* $S \,\delta\, S$, *then* $(\hat{\imath}_1, \hat{\imath}_2, \ldots, \hat{\imath}_m) \prec (\hat{\jmath}_1, \hat{\jmath}_2, \ldots, \hat{\jmath}_m)$;

(c) *If* $S < T$ *and* $T \,\delta\, S$, *then* $(\hat{\imath}_1, \hat{\imath}_2, \ldots, \hat{\imath}_m) \succ (\hat{\jmath}_1, \hat{\jmath}_2, \ldots, \hat{\jmath}_m)$;

*where*

$$\hat{\imath} = (\hat{\imath}_1, \ldots, \hat{\imath}_m, \hat{\imath}_{m+1}, \ldots, \hat{\imath}_{m_S})$$
$$\hat{\jmath} = (\hat{\jmath}_1, \ldots, \hat{\jmath}_m, \hat{\jmath}_{m_S+1}, \ldots, \hat{\jmath}_{m_S+m_T-m}).$$

**Corollary 1** *If statement* $T$ *depends on statement* $S$ *with a direction vector* $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_m)$, *then Equation (5.3) has an integer solution* $(\hat{\imath}, \hat{\jmath})$ *that satisfies (5.4)-(5.5) and the additional condition*

$$\operatorname{sig}(\hat{\jmath}_r - \hat{\imath}_r) = \sigma_r \qquad (1 \le r \le m). \tag{5.6}$$

**Corollary 2** *If statement* $T$ *depends on statement* $S$ *at a level* $\ell$, *then Equation (5.3) has an integer solution that satisfies (5.4)-(5.5) and the additional conditions:*

$$\hat{\imath}_1 = \hat{\jmath}_1, \hat{\imath}_2 = \hat{\jmath}_2, \ldots, \hat{\imath}_{\ell-1} = \hat{\jmath}_{\ell-1}, \hat{\imath}_\ell \le \hat{\jmath}_\ell - 1.$$

By Corollary 1 to Theorem 5.2, we have $\boldsymbol{\sigma} \succeq \mathbf{0}$ if $S < T$, and $\boldsymbol{\sigma} \succ \mathbf{0}$ otherwise. Also, by Corollary 2 to the same theorem, we have $1 \le \ell \le m + 1$ if $S < T$, and $1 \le \ell \le m$ otherwise.

**Theorem 5.4** *Suppose that Equation (5.3) has an integer solution* $(\hat{\imath}, \hat{\jmath})$ *that satisfies the constraints (5.4)-(5.5). Let* $\hat{\imath} = (\hat{\imath}_1, \hat{\imath}_2, \ldots, \hat{\imath}_{m_S})$ *and* $\hat{\jmath} = (\hat{\jmath}_1, \ldots, \hat{\jmath}_m, \hat{\jmath}_{m_S+1}, \ldots, \hat{\jmath}_{m_S+m_T-m})$.

(a) *If* $(\hat{\imath}_1, \hat{\imath}_2, \ldots, \hat{\imath}_m) \prec (\hat{\jmath}_1, \hat{\jmath}_2, \ldots, \hat{\jmath}_m)$, *then* $S \,\overline{\delta}\, T$.

(b) *If* $(\hat{\imath}_1, \hat{\imath}_2, \ldots, \hat{\imath}_m) \succ (\hat{\jmath}_1, \hat{\jmath}_2, \ldots, \hat{\jmath}_m)$, *then* $T \,\overline{\delta}\, S$.

(c) *If* $(\hat{\imath}_1, \hat{\imath}_2, \ldots, \hat{\imath}_m) = (\hat{\jmath}_1, \hat{\jmath}_2, \ldots, \hat{\jmath}_m)$ *and* $S < T$, *then* $S \,\overline{\delta}\, T$.

As always, we do not distinguish between dependence and indirect dependence. If there is an integer solution to (5.3), satisfying (5.4)-(5.5) and additional conditions, if any, then we assume that the

corresponding dependence exists. The dependence problem for the general program is defined as in Section 4.5.

---

**Example 5.2** In this example, we state the dependence problem for the program of Example 5.1:

```
L1 :      do I1 = 1, 100, 1
L2 :         do I2 = 1, I1, 2
L3 :            do I3 = I1, I1 + I2, 1
   S :             X(2I1 − 1, 3I2 + 1, 2I3) = · · ·
                enddo
L4 :            do I4 = 100, 0, −3
L5 :               do I5 = I1, I4, 1
   T :                · · · = · · · X(2I1 + 1, 4I2 + 6, I4 + I5) · · ·
                enddo
             enddo
          enddo
```

Much of the computation has been done already. The variable of statement $S$ can be written as $X((I_1, I_2, I_3)\mathbf{A} + \mathbf{a}_0)$, where

$$\mathbf{A} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{pmatrix} \quad \text{and} \quad \mathbf{a}_0 = (-1, 1, 0).$$

Its normalized form is $X\left((\hat{I}_1, \hat{I}_2, \hat{I}_3)\hat{\mathbf{A}} + \hat{\mathbf{a}}_0\right)$, where

$$\hat{\mathbf{A}} = \Theta_S \mathbf{P}_S^{-1}\mathbf{A} = \begin{pmatrix} 2 & 0 & 2 \\ 0 & 6 & 0 \\ 0 & 0 & 2 \end{pmatrix},$$

$$\hat{\mathbf{a}}_0 = \mathbf{p}_{S0}\mathbf{P}_S^{-1}\mathbf{A} + \mathbf{a}_0 = (1, 4, 2).$$

The variable of $T$ can be written as $X((I_1, I_2, I_4, I_5)\mathbf{B} + \mathbf{b}_0)$, where

$$\mathbf{B} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\mathbf{b}_0 = (1, 6, 0).$$

Its normalized form is $X\left((\hat{I}_1, \hat{I}_2, \hat{I}_4, \hat{I}_5)\hat{\mathbf{B}} + \hat{\mathbf{b}}_0\right)$, where

$$\hat{\mathbf{B}} = \Theta_T \mathbf{P}_T^{-1}\mathbf{B} = \begin{pmatrix} 2 & 0 & 1 \\ 0 & 8 & 0 \\ 0 & 0 & -3 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\hat{\mathbf{b}}_0 = \mathbf{p}_{T0}\mathbf{P}_T^{-1}\mathbf{B} + \mathbf{b}_0 = (3, 10, 101).$$

The dependence equation (5.3) for this problem is

$$(\hat{\imath}_1, \hat{\imath}_2, \hat{\imath}_3)\begin{pmatrix} 2 & 0 & 2 \\ 0 & 6 & 0 \\ 0 & 0 & 2 \end{pmatrix} - (\hat{\jmath}_1, \hat{\jmath}_2, \hat{\jmath}_4, \hat{\jmath}_5)\begin{pmatrix} 2 & 0 & 1 \\ 0 & 8 & 0 \\ 0 & 0 & -3 \\ 0 & 0 & 1 \end{pmatrix} = (2, 6, 99). \tag{5.7}$$

It consists of three scalar equations in 7 integer variables:

$$\left.\begin{aligned} 2\hat{\imath}_1 - 2\hat{\jmath}_1 &= 2 \\ 6\hat{\imath}_2 - 8\hat{\jmath}_2 &= 6 \\ 2\hat{\imath}_1 + 2\hat{\imath}_3 - \hat{\jmath}_1 + 3\hat{\jmath}_4 - \hat{\jmath}_5 &= 99. \end{aligned}\right\}$$

In Example 5.1, we found the dependence constraints (5.4)–(5.5) for this problem:

$$\left\{\begin{aligned} &0 \le \hat{\imath}_1 \le 99 \\ &0 \le \hat{\imath}_2 \le \hat{\imath}_1/2 \\ &0 \le \hat{\imath}_3 \le 2\hat{\imath}_2 + 1 \\ &0 \le \hat{\jmath}_1 \le 99 \\ &0 \le \hat{\jmath}_2 \le \hat{\jmath}_1/2 \\ &0 \le \hat{\jmath}_4 \le 33 \\ &0 \le \hat{\jmath}_5 \le 99 - \hat{\jmath}_1 - 3\hat{\jmath}_4. \end{aligned}\right.$$

If we want to test if there is a dependence of $T$ on $S$ with a particular direction vector, say $(1, -1)$, then we will get two additional conditions: $i_1 \le j_1 - 1$ and $j_2 \le i_2 - 1$. Similarly, for dependence at a given level, say 2, the additional conditions are $i_1 = j_1$ and $i_2 \le j_2 - 1$.

**EXERCISES 5.3**

1. Consider the program

```
L1 :      do I1 = 10, 100, 2
L2 :         do I2 = 2I1, I1 + 1, −1
     S1 :       X(7I1 − I2 + 6, I2) = · · ·
             enddo
     S2 :    Y(3I1 + 1) = · · ·
L3 :         do I3 = I1 + 3, 100, 3
     S3 :       X(4I1 + 12I3 − 5, I1 + I3 + 3) = · · ·
L4 :            do I4 = I1 + 2I3, 5I1 − I3, 1
     S4 :          Y(2I1 − I3 + I4) = · · ·
                enddo
             enddo
          enddo
```

   Find the dependence equations and constraints for each possible dependence between statements. (Include dependence of a statement on itself.)

## 5.4 Generalized gcd Test

After formalizing the dependence problem in a sequence of programs with increasing generality, we have reached a point where problem formulation is complete and problem solution is to begin. The basic idea is to find integer solutions to a combined system of linear equations and inequalities. Over the course of chapters 1–4, we have pointed out and completely solved certain problems that we called simple problems in Chapter 1. In chapters 6 and 7, we will explain the two approximate methods (method of bounds and method of elimination) that may be used for problems not treatable as simple problems. The first major step of any dependence testing algorithm is to decide if the system of equations alone has an integer solution. In Chapter 1, we showed in detail how to solve a single equation in two variables. That technique can be used repeatedly to solve a system of two-variable equations such that no two equations have a common variable. In occasional examples, we have shown how to decide the existence of a solution to a more general system, or even to find the general solution. The basic machinery for solving systems of linear diophantine equations is available in Chapter I.3. We are adapting it here to the dependence problem.

A single equation of the form

$$a_1x_1 + a_2x_2 + \cdots + a_mx_m = c$$

with integer coefficients, has an integer solution in $x_1, x_2, \ldots, x_m$, iff $\gcd(a_1, a_2, \ldots, a_m)$ evenly divides $c$ (Theorem I.3.5). When this well-known result is used in dependence analysis, it is called the *gcd test.* The gcd test for a single loop, that is, the test for an equation in two variables, can be traced to [Coha 73]. The gcd test for a one-dimensional array in a nest of loops, that is, for a single equation in many variables, appeared in [Towl 76].

Consider now the dependence problem for an array with one or more dimensions in a general program. In this case, we get a general system of linear diophantine equations. The test that is applicable to any linear dependence problem was introduced in [Bane 88a]. It is called the *generalized gcd test.* As the name indicates, it generalizes the gcd test for a one-dimensional array. (See Section 6.4 of [Bane 88a], Theorem I.3.6, and Section I.5.2.) It is stated below in the context of the dependence problem in a general program. Note that the dependence equation (5.3) can be written in the form

$$(\hat{\mathbf{i}}; \hat{\mathbf{j}}) \begin{pmatrix} \hat{\mathbf{A}} \\ -\hat{\mathbf{B}} \end{pmatrix} = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0, \tag{5.8}$$

where $(\hat{\mathbf{i}}; \hat{\mathbf{j}})$ is the vector of size $(m_S + m_T)$ obtained by concatenating the elements of $\hat{\imath}$ and $\hat{\imath}$, and the coefficient matrix of the equation is the $(m_S + m_T) \times n$ matrix obtained by concatenating the rows of matrices $\hat{\mathbf{A}}$ and $-\hat{\mathbf{B}}$.

**Generalized gcd test.** *Find an* $(m_S + m_T) \times (m_S + m_T)$ *unimodular matrix* $\mathbf{U}$ *and an* $(m_S + m_T) \times n$ *echelon matrix* $\mathbf{S}$, *such that*

$$\mathbf{U} \cdot \begin{pmatrix} \hat{\mathbf{A}} \\ -\hat{\mathbf{B}} \end{pmatrix} = \mathbf{S}.$$

*If the variables* $X\left(\hat{\mathbf{I}}_S\hat{\mathbf{A}} + \hat{\mathbf{a}}_0\right)$ *of statement* $S$ *and* $X\left(\hat{\mathbf{I}}_T\hat{\mathbf{B}} + \hat{\mathbf{b}}_0\right)$ *of statement* $T$ *cause a dependence between* $S$ *and* $T$, *then there exists an integer vector* $\mathbf{t}$ *of size* $(m_S + m_T)$ *that satisfies the equation*

$$\mathbf{tS} = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0. \tag{5.9}$$

The matrices **U** and **S** can be found by Algorithm I.2.1. The test follows from Theorem I.3.6. That theorem says that there is an integer solution to the dependence equation (5.8) iff there is an integer solution **t** to Equation (5.9). The whole point here is that (5.9) is much easier to solve than (5.8), since **S** is an echelon matrix. If there is no integer solution to the dependence equation, then dependence cannot exist. On the other hand, if the dependence equation does have an integer solution, then that fact alone cannot decide the existence of dependence. We must have an integer solution that also satisfies the constraints. Thus, the generalized gcd test provides only a necessary condition for dependence between statements $S$ and $T$.

---

**Example 5.3** Consider again the program of examples 5.1–5.2. In this example, we apply the generalized gcd test to the dependence problem between statements $S$ and $T$. The dependence equation (5.7) in Example 5.2 can be written as

$$(\hat{\imath}_1, \hat{\imath}_2, \hat{\imath}_3, \hat{\jmath}_1, \hat{\jmath}_2, \hat{\jmath}_4, \hat{\jmath}_5) \begin{pmatrix} 2 & 0 & 2 \\ 0 & 6 & 0 \\ 0 & 0 & 2 \\ -2 & 0 & -1 \\ 0 & -8 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & -1 \end{pmatrix} = (2, 6, 99).$$

We apply Algorithm I.2.1 to the coefficient matrix of this system to get the decomposition

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 4 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 \end{pmatrix} \begin{pmatrix} 2 & 0 & 2 \\ 0 & 6 & 0 \\ 0 & 0 & 2 \\ -2 & 0 & -1 \\ 0 & -8 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} -2 & 0 & -1 \\ 0 & -2 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

where the $7 \times 7$ matrix on the left (the **U** matrix) is unimodular and the $7 \times 3$ matrix on the right (the **S** matrix) is echelon. The equation

$$(t_1, t_2, t_3, t_4, t_5, t_6, t_7) \cdot \mathbf{S} = (2, 6, 99)$$

is equivalent to the equation

$$(-2t_1, -2t_2, -t_1 - t_3) = (2, 6, 99)$$

that clearly has integer solutions. In fact, any 7-vector of the form $(-1, -3, -98, t_4, t_5, t_6, t_7)$ is an integer solution, where $t_4, \ldots, t_7$ are arbitrary integers. Thus, the generalized gcd test passes. This means for the current dependence problem, this test does not rule out dependence, nor does it confirm it.

---

It is clear that there is a version of the generalized gcd test for each special dependence problem where a direction vector is specified. For the dependence of $T$ on $S$ with a given direction vector $(\sigma_1, \sigma_2, \ldots, \sigma_m)$, we get $m$ extra conditions. The condition that corresponds to an element $\sigma_r = 0$ is an equality: $\hat{\imath}_r = \hat{\jmath}_r$. One approach is to attach all those equalities to the system (5.8), and then apply the generalized gcd test to the extended system. Another approach is to reduce the number of variables from the equations and constraints using these equalities, right at the beginning. Yet another approach is to perform the generalized gcd test once (as described here), and then apply the additional conditions imposed by the given direction vector. We will return to this point in Section 7.2.

**EXERCISES 5.4**

1. Apply the generalized gcd test to each dependence problem (between two statements) of Exercise 5.3.1. In each case, apply the test also to the problem of checking dependence (of one statement on another) at level 2 (if such a level is permissible).
2. To apply the generalized gcd test, we need not know the final vectors $\mathbf{q}_{S0}$ and $\mathbf{q}_{T0}$, or the final matrices $Q_{S0}$ and $Q_{T0}$. Is there a situation where we will be able to apply this test without knowing even the initial vectors and matrices?

# Chapter 6

# Method of Bounds

## 6.1 Introduction

Algorithm 1.2 is an approximate dependence algorithm that checks if (1) the system of dependence equations has an integer solution, and (2) the combined system of equations and inequalities has a real solution. The *method of bounds* is an implementation of Algorithm 1.2 that consists of two major tests: the generalized gcd test to check for an integer solution to the equations, and the bounds test to check for a real solution to the equations and the inequalities. We covered the first test in Section 5.4; this chapter is devoted to the second. For this chapter, we need several mathematical definitions and results that have been collected in the appendix.

The *bounds test* works by testing if a certain real number lies between the extreme values of a certain real-valued function in a certain set. For a linear dependence problem involving an $n$-dimensional array $X$, there are $n$ scalar dependence equations that can be written in the form

$$f_k(\mathbf{x}) = c_k \qquad (1 \le k \le n), \tag{6.1}$$

where $f_1, f_2, \ldots, f_n$ are real-valued linear functions on some Euclidean space $\mathbf{R}^N$, and $c_1, c_2, \ldots, c_n$ are integers. The inequalities of the problem define a subset $P$ of $\mathbf{R}^N$, where $P$ is a polytope (see Section A.2). First, consider the case $n = 1$. Here, we have a single dependence

equation; denote it by

$$f(\mathbf{x}) = c. \tag{6.2}$$

By Theorem A.6, there is a real solution to Equation (6.2) in $P$ iff

$$\min_{\mathbf{x}\in P} f(\mathbf{x}) \le c \le \max_{\mathbf{x}\in P} f(\mathbf{x}). \tag{6.3}$$

The bounds test for a one-dimensional array consists of testing this inequality. When $n > 1$, the situation gets more complicated. By Theorem A.11, the system of equations (6.1) has a real solution in $P$ iff

$$\max_{\boldsymbol{\lambda}\in\mathbf{R}^{n-1}} \min_{\mathbf{x}\in P} \left[ f_1(\mathbf{x}) + \sum_{k=2}^{n} \lambda_k (f_k(\mathbf{x}) - c_k) \right] \le c_1$$
$$\le \min_{\boldsymbol{\lambda}\in\mathbf{R}^{n-1}} \max_{\mathbf{x}\in P} \left[ f_1(\mathbf{x}) + \sum_{k=2}^{n} \lambda_k (f_k(\mathbf{x}) - c_k) \right]. \tag{6.4}$$

This is the general inequality for the bounds test; it includes (6.3) as a special case ($n = 1$). If (6.3) or (6.4) fails to hold, then there is no dependence. The bounds test originated in [Bane 76].

The bounds test is practical when the extreme values in (6.3) or (6.4) can be computed easily. The extreme values of a linear function $f$ in a polytope $P$ are attained at extreme points of $P$ (Theorem A.5). There are dependence problems where the structure of $P$ is so regular, that we can easily find the extreme values of our function by repeated applications of Theorem 2.11. Also, taking the minimum and maximum values over $\boldsymbol{\lambda} \in \mathbf{R}^{n-1}$ in (6.4) may not be too hard if $n$ is small.

The bounds test can be diluted in two ways. First, instead of using the minimum and maximum values of the function in $P$, we can use a lower bound and an upper bound in $P$. Then, the condition we would get will be only a necessary condition for the existence of a real solution. This fact can be stated in the form of the following lemma whose proof is trivial.

**Lemma 6.1** *Consider a bounded real-valued function $f : D \to \mathbf{R}$ defined on a nonempty set $D$. Let $\mu$ denote a lower and $\nu$ an upper bound for $f$. If the equation $f(x) = c$ has a solution in $D$, then $\mu \le c \le \nu$.*

Next, in case of a multi-dimensional array, instead of seeking a *simultaneous* real solution to the system of $n$ equations (6.1) in $P$, we may simply test for $n$ *individual* real solutions to the $n$ individual equations in $P$. Then, we would test the following sequence of $n$ inequalities:

$$\min_{\mathbf{x}\in P} f_k(\mathbf{x}) \le c_k \le \max_{\mathbf{x}\in P} f_k(\mathbf{x}) \qquad (1 \le k \le n). \tag{6.5}$$

If at least one of these inequalities fails to hold, then there is no dependence.

In Section 6.2, we start with results for a one-dimensional array in a perfect rectangular loop nest with unit strides, and then generalize them by allowing arbitrary nesting of loops in Section 6.3. In Section 6.4, we consider tests for multi-dimensional arrays in a nest of rectangular loops with unit strides: the subscript by subscript version of the bounds test as stated in (6.5), and the general test (6.4). Finally, in Section 6.5, we discuss the general bounds test (6.4) in a general program. We should note here that unit strides have been assumed solely for the purpose of simplifying expressions. For a loop with non-unit stride, we can use its iteration variable instead of its index variable, and thus effectively change the stride to 1. Indeed, by the methods of sections 6.2–6.5, we can handle *regular* loops with arbitrary strides (see Theorem 4.3 and its corollary). We return to this point in some of the examples and exercises.

## 6.2 Perfect Nest, One-Dimensional Array

Consider a perfect loop nest **L** of the form:

```
L1 :   do I1 = p1, q1, 1
L2 :      do I2 = p2, q2, 1
  ⋮          ⋮
Lm :         do Im = pm, qm, 1
                H(I1, I2, ..., Im)
             enddo
             ⋮
          enddo
       enddo
```

where each loop has constant limits and a stride of 1, and $H$ is a sequence of assignment statements. Let $S$ and $T$ denote two statements in $H$. Suppose $X(a_0 + a_1 I_1 + a_2 I_2 + \cdots + a_m I_m)$ is a program variable of $S$ and $X(b_0 + b_1 I_1 + b_2 I_2 + \cdots + b_m I_m)$ a variable of $T$, where $X$ is a one-dimensional array, and all coefficients are integer constants. We study the problem of dependence between $S$ and $T$ posed by these two variables. Since the loop strides are all equal to 1, we work with the loop index variables: $I_1, I_2, \ldots, I_m$.

The first theorem gives a set of necessary conditions under which $T$ depends on $S$ at a given level $\ell$.

**Theorem 6.2** *If in the loop nest* **L**, *a variable* $X\ (a_0 + \sum_{r=1}^{m} a_r I_r)$ *of* $S$ *and a variable* $X\ (b_0 + \sum_{r=1}^{m} b_r I_r)$ *of* $T$ *cause a dependence of* $T$ *on* $S$ *at a level* $\ell$, *then the following two conditions hold:*

(a) *The gcd of*

$$a_1 - b_1, a_2 - b_2, \ldots, a_{\ell-1} - b_{\ell-1}, a_\ell, \ldots, a_m, b_\ell, \ldots, b_m$$

*divides* $(b_0 - a_0)$; *and*

(b) $\overline{\mu} \le b_0 - a_0 \le \overline{\nu}$,

*where*

$$\overline{\mu} = \sum_{r=1}^{\ell-1} \mu(a_r - b_r, 0, p_r, q_r, 0) + \mu(a_\ell, -b_\ell, p_\ell, q_\ell, 1) + \sum_{r=\ell+1}^{m} [\mu(a_r, 0, p_r, q_r, 0) + \mu(-b_r, 0, p_r, q_r, 0)]$$

*and*

$$\overline{\nu} = \sum_{r=1}^{\ell-1} \nu(a_r - b_r, 0, p_r, q_r, 0) + \nu(a_\ell, -b_\ell, p_\ell, q_\ell, 1) + \sum_{r=\ell+1}^{m} [\nu(a_r, 0, p_r, q_r, 0) + \nu(-b_r, 0, p_r, q_r, 0)].$$

PROOF. The dependence equation for these two variables is

$$\sum_{r=1}^{m} (a_r i_r - b_r j_r) = b_0 - a_0, \tag{6.6}$$

and the dependence constraints are

$$\left.\begin{array}{l} p_r \le i_r \le q_r \\ p_r \le j_r \le q_r \end{array}\right\} \qquad (1 \le r \le m). \tag{6.7}$$

Since there is a dependence of statement $T$ on statement $S$ at level $\ell$, there are integer vectors $(i_1, i_2, \ldots, i_m)$ and $(j_1, j_2, \ldots, j_m)$ that satisfy Equation (6.6), the constraints (6.7), and the following additional conditions[1]:

$$i_1 = j_1, i_2 = j_2, \ldots, i_{\ell-1} = j_{\ell-1}, i_\ell < j_\ell. \tag{6.8}$$

Using the $\ell - 1$ equalities in (6.8), we can reduce the number of variables from $2m$ to $(2m - \ell + 1)$. Thus, there are integers $i_1, i_2, \ldots, i_{\ell-1}$, $i_\ell, j_\ell, i_{\ell+1}, j_{\ell+1}, \ldots, i_m, j_m$ that satisfy the equation

$$\sum_{r=1}^{\ell-1} (a_r - b_r) i_r + (a_\ell i_\ell - b_\ell j_\ell) + \sum_{r=\ell+1}^{m} (a_r i_r - b_r j_r) = b_0 - a_0, \tag{6.9}$$

and the conditions:

$$\left.\begin{array}{ll} p_r \le i_r \le q_r & (1 \le r \le \ell - 1) \\ p_\ell \le i_\ell \le j_\ell - 1 \le q_\ell - 1 & \\ p_r \le i_r \le q_r & (\ell + 1 \le r \le m) \\ p_r \le j_r \le q_r & (\ell + 1 \le r \le m). \end{array}\right\} \tag{6.10}$$

Note that the first sum in (6.9) is empty if $\ell = 1$, the term $(a_\ell i_\ell - b_\ell j_\ell)$ is absent if $\ell = m + 1$, and the last sum is empty if $\ell$ equals $m$ or $m+1$. (These assumptions are implicit in the interpretation of similar expressions throughout this chapter.)

Since Equation (6.9) has an integer solution, the gcd of the coefficients on the left-hand-side must divide the right-hand-side (Theorem I.3.5). That gives Condition (a) of the theorem.

To derive Condition (b), consider the Euclidean space $\mathbf{R}^{2m-\ell+1}$ and denote an arbitrary vector in it by

$$\mathbf{x} = (x_1, \ldots, x_{\ell-1}, x_\ell, y_\ell, x_{\ell+1}, y_{\ell+1}, \ldots, x_m, y_m).$$

[1]See Corollary 2 to Theorem 4.5. If $\ell = m + 1$, the inequality $i_\ell < j_\ell$ is absent.

(The symbols $x_r, y_r$ for the real variables have been so chosen that their correspondence to the integer variables of our problem is immediately clear.) Define a real-valued linear function $f$ on $\mathbf{R}^{2m-\ell+1}$ by

$$f(x_1, \ldots, x_{\ell-1}, x_\ell, y_\ell, x_{\ell+1}, y_{\ell+1}, \ldots, x_m, y_m)$$
$$= \sum_{r=1}^{\ell-1} (a_r - b_r)x_r + (a_\ell x_\ell - b_\ell y_\ell) + \sum_{r=\ell+1}^{m} (a_r x_r - b_r y_r). \quad (6.11)$$

Let $P$ denote the polytope in $\mathbf{R}^{2m-\ell+1}$ defined by the conditions in (6.10), when the integer variables are replaced by their real counterparts. In the real domain, Equation (6.9) may be written as

$$f(\mathbf{x}) = b_0 - a_0.$$

Since it has a solution in $P$ by hypothesis, Theorem A.6 implies

$$\min_{\mathbf{x}\in P} f(\mathbf{x}) \le b_0 - a_0 \le \max_{\mathbf{x}\in P} f(\mathbf{x}).$$

To complete the proof, we need to show that $\min_{\mathbf{x}\in P} f(\mathbf{x}) = \overline{\mu}$ and $\max_{\mathbf{x}\in P} f(\mathbf{x}) = \overline{\nu}$. The function $f$ defined by (6.11) and the polytope $P$ defined by (6.10) are such that the problem of finding the extreme values of $f$ in $P$ can be broken up into a number of two-variable problems, so that we may use Theorem 2.11 and its corollary. Let $P_\ell$ denote the polytope

$$\left\{(x, y) \in \mathbf{R}^2 : p_\ell \le x \le q_\ell, p_\ell \le y \le q_\ell, x \le y - 1\right\}.$$

Then, we have

$$\begin{aligned}
\min_{\mathbf{x}\in P} f(\mathbf{x}) &= \sum_{r=1}^{\ell-1} \min_{p_r \le x_r \le q_r} (a_r - b_r)x_r + \min_{(x_\ell, y_\ell)\in P_\ell} (a_\ell x_\ell - b_\ell y_\ell) + \\
&\qquad \sum_{r=\ell+1}^{m} \left[\min_{p_r \le x_r \le q_r} a_r x_r + \min_{p_r \le y_r \le q_r} (-b_r)y_r)\right] \\
&= \sum_{r=1}^{\ell-1} \mu(a_r - b_r, 0, p_r, q_r, 0) + \mu(a_\ell, -b_\ell, p_\ell, q_\ell, 1) + \\
&\qquad \sum_{r=\ell+1}^{m} [\mu(a_r, 0, p_r, q_r, 0) + \mu(-b_r, 0, p_r, q_r, 0)] \\
&= \overline{\mu},
\end{aligned}$$

$$\begin{aligned}\max_{\mathbf{x}\in P} f(\mathbf{x}) &= \sum_{r=1}^{\ell-1} \max_{p_r\le x_r\le q_r} (a_r - b_r)x_r + \max_{(x_\ell,y_\ell)\in P_\ell} (a_\ell x_\ell - b_\ell y_\ell) + \\ &\qquad \sum_{r=\ell+1}^{m} \left[\max_{p_r\le x_r\le q_r} a_r x_r + \max_{p_r\le y_r\le q_r} (-b_r)y_r)\right] \\ &= \sum_{r=1}^{\ell-1} \nu(a_r - b_r, 0, p_r, q_r, 0) + \nu(a_\ell, -b_\ell, p_\ell, q_\ell, 1) + \\ &\qquad \sum_{r=\ell+1}^{m} [\nu(a_r, 0, p_r, q_r, 0) + \nu(-b_r, 0, p_r, q_r, 0)] \\ &= \overline{\nu}. \end{aligned}$$

□

**Corollary 1** *If statement $T$ depends on statement $S$ in the loop nest* **L**, *then the conditions of Theorem 6.2 hold for some dependence level $\ell$ such that $1 \le \ell \le m+1$ (or $1 \le \ell \le m$, if $T \le S$).*

Theorem 6.2 can be easily generalized to test for dependence with a specified direction vector. To simplify expressions, we agree that when the range of $r$ is not explicitly stated, it is $\{1, 2, \ldots, m\}$.

**Theorem 6.3** *Let $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ denote a vector whose elements are members of the set $\{0, 1, -1, *\}$. If in the loop nest* **L**, *the variables $X(a_0 + \sum_{r=1}^{m} a_r I_r)$ of $S$ and $X(b_0 + \sum_{r=1}^{m} b_r I_r)$ of $T$ cause a dependence of $T$ on $S$ with the direction vector $\boldsymbol{\sigma}$, then the following two conditions hold:*

(a) *The gcd of all integers in the three lists:*

$$\{(a_r - b_r) : \sigma_r = 0\}, \{a_r : \sigma_r \ne 0\}, \{b_r : \sigma_r \ne 0\}$$

*divides $(b_0 - a_0)$; and*

(b) $\overline{\mu} \le b_0 - a_0 \le \overline{\nu}$,

*where*

$$\begin{aligned}\overline{\mu} = &\sum_{\sigma_r=0} \mu(a_r - b_r, 0, p_r, q_r, 0) + \sum_{\sigma_r=1} \mu(a_r, -b_r, p_r, q_r, 1) + \\ &\sum_{\sigma_r=-1} \mu(-b_r, a_r, p_r, q_r, 1) + \\ &\sum_{\sigma_r=*} [\mu(a_r, 0, p_r, q_r, 0) + \mu(-b_r, 0, p_r, q_r, 0)]\end{aligned}$$

*and*

$$\overline{\nu} = \sum_{\sigma_r=0} \nu(a_r - b_r, 0, p_r, q_r, 0) + \sum_{\sigma_r=1} \nu(a_r, -b_r, p_r, q_r, 1) + \sum_{\sigma_r=-1} \nu(-b_r, a_r, p_r, q_r, 1) + \sum_{\sigma_r=*} [\nu(a_r, 0, p_r, q_r, 0) + \nu(-b_r, 0, p_r, q_r, 0)].$$

PROOF. The proof is similar to the proof of Theorem 6.2. We have the same dependence equation:

$$\sum_{r=1}^{m} (a_r i_r - b_r j_r) = b_0 - a_0, \tag{6.12}$$

and the same set of dependence constraints:

$$\left.\begin{aligned} p_r \le i_r \le q_r \\ p_r \le j_r \le q_r \end{aligned}\right\} \quad (1 \le r \le m). \tag{6.13}$$

Since $T$ depends on $S$ with the direction vector $(\sigma_1, \sigma_2, \ldots, \sigma_m)$, there exist integers $i_1, i_2, \ldots, i_m, j_1, j_2, \ldots, j_m$ that satisfy Equation (6.12), the constraints (6.13), and the additional conditions:

$$i_r = j_r \qquad \text{if } \sigma_r = 0, \tag{6.14}$$

$$i_r \le j_r - 1 \qquad \text{if } \sigma_r = 1, \tag{6.15}$$

$$j_r \le i_r - 1 \qquad \text{if } \sigma_r = -1. \tag{6.16}$$

Define an integer $N$ such that the number of zero elements of $\boldsymbol{\sigma}$ is $2m - N$. Then, we can reduce the number of variables from $2m$ to $N$ by writing $j_r = i_r$ for each $r$ such that $\sigma_r = 0$.

After regrouping of terms, Equation (6.12) can be written as

$$\sum_{\sigma_r=0} (a_r - b_r) i_r + \sum_{\sigma_r=1} (a_r i_r - b_r j_r) + \sum_{\sigma_r=-1} (a_r i_r - b_r j_r) + \sum_{\sigma_r=*} (a_r i_r - b_r j_r) = b_0 - a_0.$$

Since this equation has an integer solution, the gcd of the coefficients on the left-hand-side must divide the right-hand-side (Theorem I.3.5). That gives Condition (a) of the theorem.

To prove Condition (b), we define a real-valued linear function $f$ on $\mathbf{R}^N$ by

$$f(\mathbf{x}) = \sum_{\sigma_r=0} (a_r - b_r)x_r + \sum_{\sigma_r=1} (a_r x_r - b_r y_r) + \sum_{\sigma_r=-1} (a_r x_r - b_r y_r) + \sum_{\sigma_r=*} (a_r x_r - b_r y_r).$$

Let $P$ denote the polytope in $\mathbf{R}^N$ defined by the inequalities in (6.13), (6.15), and (6.16), when integer variables have been replaced by the corresponding real variables. Existence of dependence implies that the equation $f(\mathbf{x}) = b_0 - a_0$ has a real solution in $P$. Hence, by Theorem A.6, we have

$$\min_{\mathbf{x}\in P} f(\mathbf{x}) \le b_0 - a_0 \le \max_{\mathbf{x}\in P} f(\mathbf{x}).$$

To compute these extreme values, we note that for $\sigma_r = 1$, the variables $x_r$ and $y_r$ satisfy (6.13) and (6.15) iff $(x_r, y_r) \in P_r$, where

$$P_r = \{(x, y) \in \mathbf{R}^2 : p_r \le x \le q_r, p_r \le y \le q_r, x \le y - 1\}$$

is a polytope in $\mathbf{R}^2$. Also, for $\sigma_r = -1$, the variables $x_r$ and $y_r$ satisfy (6.13) and (6.16) iff $(y_r, x_r) \in P_r$. The extreme values are then found by repeated applications of Theorem 2.11 and its corollary:

$$\begin{aligned}\min_{\mathbf{x}\in P} f(\mathbf{x}) &= \sum_{\sigma_r=0} \min_{p_r \le x_r \le q_r} (a_r - b_r)x_r + \sum_{\sigma_r=1} \min_{(x_r,y_r)\in P_r} (a_r x_r - b_r y_r) + \\ &\quad \sum_{\sigma_r=-1} \min_{(y_r,x_r)\in P_r} (-b_r y_r + a_r x_r) + \\ &\quad \sum_{\sigma_r=*} \left[ \min_{p_r \le x_r \le q_r} a_r x_r + \min_{p_r \le y_r \le q_r} (-b_r) y_r \right] \\ &= \sum_{\sigma_r=0} \mu(a_r - b_r, 0, p_r, q_r, 0) + \sum_{\sigma_r=1} \mu(a_r, -b_r, p_r, q_r, 1) + \\ &\quad \sum_{\sigma_r=-1} \mu(-b_r, a_r, p_r, q_r, 1) + \\ &\quad \sum_{\sigma_r=*} [\mu(a_r, 0, p_r, q_r, 0) + \mu(-b_r, 0, p_r, q_r, 0)] \\ &= \overline{\mu},\end{aligned}$$

$$\max_{\mathbf{x}\in P} f(\mathbf{x}) = \sum_{\sigma_r=0} \max_{p_r \le x_r \le q_r} (a_r - b_r)x_r + \sum_{\sigma_r=1} \max_{(x_r,y_r)\in P_r} (a_r x_r - b_r y_r) +$$
$$\sum_{\sigma_r=-1} \max_{(y_r,x_r)\in P_r} (-b_r y_r + a_r x_r) +$$
$$\sum_{\sigma_r=*} \left[ \max_{p_r \le x_r \le q_r} a_r x_r + \max_{p_r \le y_r \le q_r} (-b_r) y_r \right]$$
$$= \sum_{\sigma_r=0} v(a_r - b_r, 0, p_r, q_r, 0) + \sum_{\sigma_r=1} v(a_r, -b_r, p_r, q_r, 1) +$$
$$\sum_{\sigma_r=-1} v(-b_r, a_r, p_r, q_r, 1) +$$
$$\sum_{\sigma_r=*} [v(a_r, 0, p_r, q_r, 0) + v(-b_r, 0, p_r, q_r, 0)]$$
$$= \overline{v}. \qquad \square$$

**Corollary 1** *If statement $T$ depends on statement $S$ in the loop nest* **L**, *then the conditions of Theorem 6.3 hold for some direction vector* $\boldsymbol{\sigma} \succeq \mathbf{0}$ *(or* $\boldsymbol{\sigma} \succ \mathbf{0}$, *if* $T \le S$*).*

The first bounds test, a version of the test in Theorem 6.2(b), appeared in [Bane 76]. An extension of that inequality was given in [Kenn 80]. That extension is a special case of the test in Theorem 6.3(b) for a $\boldsymbol{\sigma}$ of the form $(0,\ldots,0,1,-1,*,\ldots,*)$. (Presence of dependence with such a direction vector prevents loop interchange.) Bounds tests of various types were given in [Bane 88a].

We would like to note here that the loop nest **L** need not be perfect. The results of this section apply when the loop nests determined by the statements $S$ and $T$ are the same.

---

**Example 6.1** Consider the double loop

```
L1 :    do I1 = 10,100,1
L2 :       do I2 = 2,50,1
   S :        X(I1 + I2 - 10) = ···
   T :        ··· = ··· X(2I1 + I2 + 31) ···
           enddo
        enddo
```

We check for dependence of $T$ on $S$ with the direction vector $(1, -1)$. Instead of applying Theorem 6.3 directly, we work out the proof of that theorem for this particular example.

The dependence equation is

$$(i_1 - 2j_1) + (i_2 - j_2) = 41.$$

The gcd of the coefficients is 1, so that this equation has integer solutions. Thus, the gcd test is not effective in this case.

The dependence constraints are

$$\begin{aligned} 10 \le i_1 \le 100, \qquad & 2 \le i_2 \le 50, \\ 10 \le j_1 \le 100, \qquad & 2 \le j_2 \le 50. \end{aligned}$$

The direction vector $(1, -1)$ imposes the additional conditions:

$$i_1 \le j_1 - 1 \qquad \text{and} \qquad j_2 \le i_2 - 1.$$

All these inequalities together define a polytope $P$ in $\mathbf{R}^4$. In terms of real variables, the dependence equation can be written as

$$f(\mathbf{x}) = 41,$$

where $\mathbf{x} = (x_1, y_1, x_2, y_2)$, and $f : \mathbf{R}^4 \to \mathbf{R}$ is a linear function defined by

$$f(\mathbf{x}) = (x_1 - 2y_1) + (x_2 - y_2).$$

The bounds test says that dependence cannot exist unless

$$\min_{\mathbf{x} \in P} f(\mathbf{x}) \le 41 \le \max_{\mathbf{x} \in P} f(\mathbf{x}).$$

Now, we have

$$\begin{aligned} \min_{\mathbf{x} \in P} f(\mathbf{x}) &= \min_{\substack{10 \le x_1 \le 100 \\ 10 \le y_1 \le 100 \\ x_1 \le y_1 - 1}} (x_1 - 2y_1) + \min_{\substack{2 \le y_2 \le 50 \\ 2 \le x_2 \le 50 \\ y_2 \le x_2 - 1}} (-y_2 + x_2) \\ &= \mu(1, -2, 10, 100, 1) + \mu(-1, 1, 2, 50, 1) \\ &= -189 \qquad \text{by (2.19)}, \end{aligned}$$

and similarly,

$$\begin{aligned}\max_{\mathbf{x}\in P} f(\mathbf{x}) &= \max_{\substack{10\le x_1\le 100\\ 10\le y_1\le 100\\ x_1\le y_1-1}} (x_1 - 2y_1) + \max_{\substack{2\le y_2\le 50\\ 2\le x_2\le 50\\ y_2\le x_2-1}} (-y_2 + x_2)\\ &= \nu(1,-2,10,100,1) + \nu(-1,1,2,50,1)\\ &= 36 \qquad \text{by (2.21).}\end{aligned}$$

Since the condition $-189 \le 41 \le 36$ does not hold, there is no dependence of $T$ on $S$ with the direction vector $(1,-1)$.

---

**EXERCISES 6.2**

1. In the notation of Theorem 6.2, what are the necessary conditions for the dependence of $S$ on $T$ at a level $\ell$? In the notation of Theorem 6.3, what are the necessary conditions for the dependence of $S$ on $T$ with a direction vector $\sigma$?
2. In the following programs, test for the dependence of $T$ on $S$, $S$ on $T$, $S$ on $S$, and $T$ on $T$ at each possible level and with each possible direction vector (use theorems 6.2 and 6.3):

   (a) The program of Example 6.1;

   (b) The program of Example 6.1 after changing the loop headers to

```
L1 :    do I1 = 10, 100, 1
L2 :       do I2 = 2I1 + 2, 2I1 + 50, 3
```

   (c)
```
L1 :    do I1 = 0, 10, 1
L2 :       do I2 = 0, 5, 1
L3 :          do I3 = 0, 8, 1
S :              X(4I1 − 8I2 + 1) = · · ·
T :              · · · = · · · X(−6I1 + 4I2 + 16I3 + 89) · · ·
              enddo
           enddo
        enddo
```

   (d)
```
L1 :    do I1 = 0, 100, 1
L2 :       do I2 = I1, I1 + 25, 2
L3 :          do I3 = I1 + I2, I1 + I2 + 48, 1
S :              X(4I1 − 8I2 + 1) = · · ·
T :              · · · = · · · X(−6I1 + 4I2 + 16I3 + 89) · · ·
              enddo
           enddo
        enddo
```

## 6.3 Rectangular Loops, One-Dimensional Array

In this section, we consider a general program and use the notation developed in Chapter 5. Let $S$ and $T$ denote any two statements in the program. Let $m_S$ denote the number of loops in the nest $\mathbf{L}_S$ determined by $S$, $m_T$ the number of loops in the nest $\mathbf{L}_T$ determined by $T$, and $m$ the number of loops in the nest determined by both. We label the loops in the program $L_1, L_2, \ldots, L_{m_S+m_T-m}$, such that

$$\begin{aligned}\mathbf{L} &= (L_1, L_2, \ldots, L_m)\\ \mathbf{L}_S &= (L_1, L_2, \ldots, L_m, L_{m+1}, \ldots, L_{m_S})\\ \mathbf{L}_T &= (L_1, L_2, \ldots, L_m, L_{m_S+1}, \ldots, L_{m_S+m_T-m}).\end{aligned}$$

We assume that the limits of each loop in the program are constant integers, and that the stride of each loop is 1. However, as pointed out in Section 6.1, it would suffice simply to assume that the loop nests determined by the two statements are regular. The index variable of the loop $L_r$ is denoted by $I_r$, its initial limit by $p_r$, and its final limit by $q_r$, where $1 \le r \le m_S + m_T - m$.

Assume that both assignment statements have program variables that are elements of a one-dimensional array $X$. Let

$$\begin{aligned}&X(a_0 + a_1 I_1 + \cdots + a_m I_m + a_{m+1} I_{m+1} + \cdots + a_{m_S} I_{m_S}) \text{ and}\\ &X(b_0 + b_1 I_1 + \cdots + b_m I_m + b_{m+1} I_{m_S+1} + \cdots + b_{m_T} I_{m_S+m_T-m})\end{aligned}$$

denote the variables of $S$ and $T$, respectively. The dependence equation is

$$a_0 + \sum_{r=1}^{m} a_r i_r + \sum_{r=m+1}^{m_S} a_r i_r = b_0 + \sum_{r=1}^{m} b_r j_r + \sum_{r=m+1}^{m_T} b_r j_{m_S+r-m}$$

or

$$\sum_{r=1}^{m} (a_r i_r - b_r j_r) + \sum_{r=m+1}^{m_S} a_r i_r - \sum_{r=m+1}^{m_T} b_r j_{m_S+r-m} = b_0 - a_0. \tag{6.17}$$

The dependence constraints are

$$\begin{cases} p_r \le i_r \le q_r & (1 \le r \le m_S)\\ p_r \le j_r \le q_r & (1 \le r \le m, m_S + 1 \le r \le m_S + m_T - m). \end{cases}$$

We can now generalize the results of the previous section for a perfect nest to results for a program consisting of arbitrarily nested loops. We state those results with just hints for proofs, omitting details that can be constructed by emulating the proofs given for theorems 6.2 and 6.3. The first theorem corresponds to Theorem 6.2.

**Theorem 6.4** *If in the model program of this section, two variables*

$$X\left(a_0 + \sum_{r=1}^{m_S} a_r I_r\right) \text{ and } X\left(b_0 + \sum_{r=1}^{m} b_r I_r + \sum_{r=m+1}^{m_T} b_r I_{m_S+r-m}\right)$$

*of statements S and T, respectively, cause a dependence of T on S at a level $\ell$, then the following two conditions hold:*

(a) *The gcd of*

$$a_1 - b_1, a_2 - b_2, \ldots, a_{\ell-1} - b_{\ell-1}, a_\ell, \ldots, a_{m_S}, b_\ell, \ldots, b_{m_T}$$

*divides $(b_0 - a_0)$; and*

(b) $\overline{\mu} \le b_0 - a_0 \le \overline{\nu}$,

*where*

$$\overline{\mu} = \sum_{r=1}^{\ell-1} \mu(a_r - b_r, 0, p_r, q_r, 0) + \mu(a_\ell, -b_\ell, p_\ell, q_\ell, 1) + \sum_{r=\ell+1}^{m_S} \mu(a_r, 0, p_r, q_r, 0) + \sum_{r=\ell+1}^{m} \mu(-b_r, 0, p_r, q_r, 0) + \sum_{r=m+1}^{m_T} \mu(-b_r, 0, p_{m_S+r-m}, q_{m_S+r-m}, 0)$$

*and*

$$\overline{\nu} = \sum_{r=1}^{\ell-1} \nu(a_r - b_r, 0, p_r, q_r, 0) + \nu(a_\ell, -b_\ell, p_\ell, q_\ell, 1) + \sum_{r=\ell+1}^{m_S} \nu(a_r, 0, p_r, q_r, 0) + \sum_{r=\ell+1}^{m} \nu(-b_r, 0, p_r, q_r, 0) + \sum_{r=m+1}^{m_T} \nu(-b_r, 0, p_{m_S+r-m}, q_{m_S+r-m}, 0).$$

PROOF. The dependence equation (6.17) can be written as

$$\sum_{r=1}^{\ell-1}(a_r - b_r)i_r + (a_\ell i_\ell - b_\ell j_\ell) + \sum_{r=\ell+1}^{m_S} a_r i_r + \sum_{r=\ell+1}^{m}(-b_r)j_r + \sum_{r=m+1}^{m_T}(-b_r)j_{m_S+r-m} = b_0 - a_0.$$

We can now emulate the proof of Theorem 6.2 and create a detailed proof for the current theorem. □

**Corollary 1** *If statement T depends on statement S in the model program, then the conditions of Theorem 6.4 hold for some dependence level $\ell$ such that $1 \le \ell \le m + 1$ (or $1 \le \ell \le m$, if $T \le S$).*

The next result is the generalized version of Theorem 6.3.

**Theorem 6.5** *Let $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ denote a vector whose elements are members of the set $\{0, 1, -1, *\}$. If in the model program of this section, two variables*

$$X\left(a_0 + \sum_{r=1}^{m_S} a_r I_r\right) \text{ and } X\left(b_0 + \sum_{r=1}^{m} b_r I_r + \sum_{r=m+1}^{m_T} b_r I_{m_S+r-m}\right)$$

*of statements S and T, respectively, cause a dependence of T on S with the direction vector $\boldsymbol{\sigma}$, then the following two conditions hold:*

(a) *The gcd of all integers in the three lists:*

$$\{a_r - b_r : 1 \le r \le m \text{ and } \sigma_r = 0\},$$
$$\{a_r : 1 \le r \le m \text{ and } \sigma_r \ne 0, \text{ or } m + 1 \le r \le m_S\},$$
$$\{b_r : 1 \le r \le m \text{ and } \sigma_r \ne 0, \text{ or } m + 1 \le r \le m_T\}$$

*divides $(b_0 - a_0)$; and*

(b) $\overline{\mu} \le b_0 - a_0 \le \overline{\nu}$,

*where*

$$\overline{\mu} = \sum_{\substack{\sigma_r=0 \\ 1\le r\le m}} \mu(a_r - b_r, 0, p_r, q_r, 0) +$$
$$\sum_{\substack{\sigma_r=1 \\ 1\le r\le m}} \mu(a_r, -b_r, p_r, q_r, 1) + \sum_{\substack{\sigma_r=-1 \\ 1\le r\le m}} \mu(-b_r, a_r, p_r, q_r, 1) +$$
$$\sum_{\substack{\sigma_r=* \\ 1\le r\le m}} [\mu(a_r, 0, p_r, q_r, 0) + \mu(-b_r, 0, p_r, q_r, 0)] +$$
$$\sum_{r=m+1}^{m_S} \mu(a_r, 0, p_r, q_r, 0) +$$
$$\sum_{r=m+1}^{m_T} \mu(-b_r, 0, p_{m_S+r-m}, q_{m_S+r-m}, 0)$$

*and*

$$\overline{\nu} = \sum_{\substack{\sigma_r=0 \\ 1\le r\le m}} \nu(a_r - b_r, 0, p_r, q_r, 0) +$$
$$\sum_{\substack{\sigma_r=1 \\ 1\le r\le m}} \nu(a_r, -b_r, p_r, q_r, 1) + \sum_{\substack{\sigma_r=-1 \\ 1\le r\le m}} \nu(-b_r, a_r, p_r, q_r, 1) +$$
$$\sum_{\substack{\sigma_r=* \\ 1\le r\le m}} [\nu(a_r, 0, p_r, q_r, 0) + \nu(-b_r, 0, p_r, q_r, 0)] +$$
$$\sum_{r=m+1}^{m_S} \nu(a_r, 0, p_r, q_r, 0) +$$
$$\sum_{r=m+1}^{m_T} \nu(-b_r, 0, p_{m_S+r-m}, q_{m_S+r-m}, 0).$$

PROOF. We can construct a proof by rewriting (6.17) in the form

$$\sum_{\substack{\sigma_r=0 \\ 1\le r\le m}} (a_r - b_r)i_r + \sum_{\substack{\sigma_r=1 \\ 1\le r\le m}} (a_r i_r - b_r j_r) +$$
$$\sum_{\substack{\sigma_r=-1 \\ 1\le r\le m}} (a_r i_r - b_r j_r) + \sum_{\substack{\sigma_r=* \\ 1\le r\le m}} (a_r i_r - b_r j_r) +$$
$$\sum_{r=m+1}^{m_S} a_r i_r + \sum_{r=m+1}^{m_T} (-b_r) j_{m_S+r-m} = b_0 - a_0. \quad \square$$

**Corollary 1** *If statement $T$ depends on statement $S$ in the model program, then the conditions of Theorem 6.5 hold for some direction vector $\sigma \geq \mathbf{0}$ (or $\sigma \succ \mathbf{0}$, if $T \leq S$).*

---

**Example 6.2** Consider the program

```
L1 :    do I1 = 10, 100, 1
L2 :       do I2 = 0, 100, 2
L3 :          do I3 = I1, I1 + 10, 1
   S :           X(2I1 + 2I2 − 2I3 + 5) = ···
              enddo
L4 :          do I4 = I2 + 10, I2, −1
   T :           ··· = ··· X(2I2 + 2I4 + 9) ···
              enddo
           enddo
        enddo
```

Suppose we want to test if statement $T$ depends on statement $S$ at level 2. The loop nests determined by the two statements are regular, since the coefficients of any given index variable in initial and final limits of each loop are equal. We can use the technique of Chapter 5 to state the dependence problem in terms of iteration variables, so that it will become a problem for rectangular loops with unit strides. Then, we would apply Theorem 6.4 to check for the dependence. In the following, we show the details of how this is done. Since $L_1$ has a stride of 1, we choose not to change from its index variable $I_1$ to its iteration variable $\hat{I}_1$.

The dependence equation in terms of index values is

$$(2i_1 + 2i_2 - 2i_3) - (2j_2 + 2j_4) = 4.$$

Noting that the index variables $I_2, I_3$, and $I_4$ are related to the corresponding iteration variables $\hat{I}_2, \hat{I}_3$, and $\hat{I}_4$ by the following equations:

$$\begin{cases} I_2 = 2\hat{I}_2 \\ I_3 = I_1 + \hat{I}_3 \\ I_4 = (I_2 + 10) - \hat{I}_4 = 2\hat{I}_2 - \hat{I}_4 + 10, \end{cases}$$

we can rewrite the dependence equation in the form

$$4\hat{i}_2 - 8\hat{j}_2 - 2\hat{i}_3 + 2\hat{j}_4 = 24. \tag{6.18}$$

The ranges of $I_1, \hat{I}_2, \hat{I}_3$, and $\hat{I}_4$ are given by

$$10 \le I_1 \le 100, \qquad 0 \le \hat{I}_3 \le 10,$$
$$0 \le \hat{I}_2 \le 50, \qquad 0 \le \hat{I}_4 \le 10.$$

From these inequalities, we get the dependence constraints in terms of $i_1, j_1, \hat{\imath}_2, \hat{\jmath}_2, \hat{\imath}_3$, and $\hat{\jmath}_4$:

$$10 \le i_1 \le 100, \quad 0 \le \hat{\imath}_2 \le 50, \quad 0 \le \hat{\imath}_3 \le 10,$$
$$10 \le j_1 \le 100, \quad 0 \le \hat{\jmath}_2 \le 50, \quad 0 \le \hat{\jmath}_4 \le 10.$$

Dependence at level 2 imposes two extra conditions: $i_1 = j_1$ and $\hat{\imath}_2 \le \hat{\jmath}_2 - 1$. We use the equality to eliminate the variable $j_1$. (For this problem, we effectively deal with four variables: $\hat{\imath}_2, \hat{\jmath}_2, \hat{\imath}_3, \hat{\jmath}_4$.)

The gcd of the coefficients on the left-hand-side of (6.18) is 2. It divides the right-hand-side, and therefore the gcd test is inconclusive. Next, we consider the bounds test. To simplify computation, we rewrite (6.18) as the equation

$$2\hat{\imath}_2 - 4\hat{\jmath}_2 - \hat{\imath}_3 + \hat{\jmath}_4 = 12.$$

Denote a vector in $\mathbf{R}^5$ by $\mathbf{x} = (x_1, x_2, y_2, x_3, y_4)$, and a linear function $f : \mathbf{R}^5 \to \mathbf{R}$ by

$$f(\mathbf{x}) = 2x_2 - 4y_2 - x_3 + y_4.$$

In the real domain, the dependence equation has the form $f(\mathbf{x}) = 12$. By the bounds test, the specified dependence cannot exist unless

$$\min_{\mathbf{x} \in P} f(\mathbf{x}) \le 12 \le \max_{\mathbf{x} \in P} f(\mathbf{x}),$$

where $P$ is the polytope in $\mathbf{R}^5$ defined by

$$10 \le x_1 \le 100, \qquad 0 \le x_2 \le 50, \qquad 0 \le x_3 \le 10,$$
$$0 \le y_2 \le 50, \qquad 0 \le y_4 \le 10.$$
$$x_2 \le y_2 - 1,$$

We have

$$\begin{aligned}\min_{\mathbf{x} \in P} f(\mathbf{x}) &= \min_{\substack{0 \le x_2 \le 50 \\ 0 \le y_2 \le 50 \\ x_2 \le y_2 - 1}} (2x_2 - 4y_2) + \min_{0 \le x_3 \le 10} (-x_3) + \min_{0 \le y_4 \le 10} y_4 \\ &= \mu(2, -4, 0, 50, 1) + \mu(-1, 0, 0, 10, 0) + \mu(1, 0, 0, 10, 0) \\ &= -210 \qquad \text{by (2.19)},\end{aligned}$$

and similarly,

$$\begin{aligned}\max_{\mathbf{x}\in P} f(\mathbf{x}) &= \max_{\substack{0\le x_2\le 50\\ 0\le y_2\le 50\\ x_2\le y_2-1}} (2x_2 - 4y_2) + \max_{0\le x_3\le 10} (-x_3) + \max_{0\le y_4\le 10} y_4 \\ &= \nu(2,-4,0,50,1) + \nu(-1,0,0,10,0) + \nu(1,0,0,10,0) \\ &= 6 \qquad \text{by (2.21).}\end{aligned}$$

Since the condition $-210 \le 12 \le 6$ does not hold, the bounds test rules out the dependence of statement $T$ on statement $S$ at level 2.

---

**EXERCISES 6.3**

1. In the notation of Theorem 6.4, what are the necessary conditions for the dependence of $S$ on $T$ at a level $\ell$? In the notation of Theorem 6.5, what are the necessary conditions for the dependence of $S$ on $T$ with a direction vector $\sigma$?

2. In the following programs, test for the dependence of $T$ on $S$, $S$ on $T$, $S$ on $S$, and $T$ on $T$ at each possible level and with each possible direction vector (use theorems 6.4 and 6.5):

   (a) The program of Example 6.2;

   (b)
   ```
   L1 :    do I1 = 10, 100, 1
   L2 :       do I2 = I1, I1 + 100, 2
   L3 :          do I3 = I1 + I2, I1 + I2 + 10, 1
   S :              X(2I1 + 2I2 − 2I3 + 5) = ···
                 enddo
   L4 :          do I4 = 3I1 + 10, 3I1, −1
   T :              ··· = ··· X(2I2 + 2I4 + 9) ···
                 enddo
              enddo
   ```

   (c)
   ```
   L1 :    do I1 = 10, 100, 1
   L2 :       do I2 = 0, 100, 2
   L3 :          do I3 = I1, I1 + 10, 1
   S :              X(2I1 + 2I2 − 2I3 + 5) = ···
                 enddo
   T :              ··· = ··· X(2I1 + 2I2 + 9) ···
              enddo
           enddo
   ```

## 6.4 Rectangular Loops, Multi-Dimensional Array

In this section, we consider dependence tests for multi-dimensional arrays in rectangular loops. First, we describe subscript-by-subscript tests. Suppose we have a method of testing for dependence caused by two program variables that are elements of a one-dimensional array. A dependence problem posed by elements of an $n$-dimensional array can be treated as a sequence of $n$ one-dimensional problems. If our method detects a lack of dependence in any one of those $n$ problems, then there is no dependence in the $n$-dimensional problem. However, the converse is false. We describe below the subscript-by-subscript version of the bounds method that consists of a gcd test and a bounds test in each dimension. (These bounds tests are given by (6.5).)

For each of the results in sections 6.2 and 6.3, there is a corresponding result for $n$-dimensional arrays ($n \geq 1$). We only show here how to generalize Theorem 6.2; the rest is left to the reader.

**Theorem 6.6** *Consider the perfect loop nest* **L** *of Section 6.2, and two statements $S$ and $T$ in its body. Let $X(\mathbf{IA} + \mathbf{a}_0)$ denote a program variable of $S$ and $X(\mathbf{IB} + \mathbf{b}_0)$ a variable of $T$, where $X$ is an $n$-dimensional array, $\mathbf{a}_0 = (a_{01}, a_{02}, \ldots, a_{0n})$ and $\mathbf{b}_0 = (b_{01}, b_{02}, \ldots, b_{0n})$ are integer $n$-vectors, and $\mathbf{A} = (a_{rk})$ and $\mathbf{B} = (b_{rk})$ are $m \times n$ integer matrices. If these variables cause a dependence of $T$ on $S$ at a level $\ell$, then the following conditions hold:*

(a) *For each $k$ in $1 \leq k \leq n$, the gcd of*

$$a_{1k} - b_{1k}, \ldots, a_{(\ell-1)k} - b_{(\ell-1)k}, a_{\ell k}, \ldots, a_{mk}, b_{\ell k}, \ldots, b_{mk}$$

*divides $(b_{0k} - a_{0k})$; and*

(b) *For each $k$ in $1 \leq k \leq n$, we have*

$$\overline{\mu}_k \leq b_{0k} - a_{0k} \leq \overline{\nu}_k,$$

*where*

$$\overline{\mu}_k = \sum_{r=1}^{\ell-1} \mu(a_{rk} - b_{rk}, 0, p_r, q_r, 0) + \mu(a_{\ell k}, -b_{\ell k}, p_\ell, q_\ell, 1) + \sum_{r=\ell+1}^{m} [\mu(a_{rk}, 0, p_r, q_r, 0) + \mu(-b_{rk}, 0, p_r, q_r, 0)]$$

*and*

$$\bar{v}_k = \sum_{r=1}^{\ell-1} v(a_{rk} - b_{rk}, 0, p_r, q_r, 0) + v(a_{\ell k}, -b_{\ell k}, p_\ell, q_\ell, 1) + \sum_{r=\ell+1}^{m} [v(a_{rk}, 0, p_r, q_r, 0) + v(-b_{rk}, 0, p_r, q_r, 0)].$$

PROOF. Instead of a single dependence equation (6.6) as in Theorem 6.2, we get a system of $n$ dependence equations:

$$\sum_{r=1}^{m} (a_{rk} i_r - b_{rk} j_r) = b_{0k} - a_{0k}, \qquad (1 \le k \le n). \tag{6.19}$$

The dependence constraints (6.7) remain the same:

$$\left.\begin{array}{l} p_r \le i_r \le q_r \\ p_r \le j_r \le q_r \end{array}\right\} \qquad (1 \le r \le m), \tag{6.20}$$

and we have the same additional conditions:

$$i_1 = j_1, i_2 = j_2, \ldots, i_{\ell-1} = j_{\ell-1}, i_\ell < j_\ell. \tag{6.21}$$

Existence of the dependence implies the existence of a simultaneous integer solution to the system (6.19) that satisfies (6.20) and (6.21). Then, there is an integer solution to each individual equation in (6.19) that satisfies (6.20) and (6.21). The conditions in (a) and (b) will follow, if the technique in the proof of Theorem 6.2 is applied separately to each equation. □

It is trivial to translate this theorem into an algorithm. We can first perform the $n$ gcd tests in sequence. If one of them fails, then we terminate; there is no dependence. If all the gcd tests pass, then we do the $n$ bounds tests in sequence. If one of them fails, then we terminate; there is no dependence. If all bounds tests pass, then there is no definite conclusion; we assume dependence in this case.

Next, we consider the general bounds test. We state a result for elements of a two-dimensional array, that generalizes Theorem 6.2. Similar generalization of the results in sections 6.2–6.3 to $n$ dimensions is left to the reader.

**Theorem 6.7** *Consider the perfect loop nest* **L** *of Section 6.2, and two statements S and T in its body. Let* $X(\mathbf{IA} + \mathbf{a}_0)$ *denote a program variable of S and* $X(\mathbf{IB} + \mathbf{b}_0)$ *a variable of T, where X is a two-dimensional array,* $\mathbf{a}_0 = (a_{01}, a_{02})$ *and* $\mathbf{b}_0 = (b_{01}, b_{02})$ *are integer 2-vectors, and* $\mathbf{A} = (a_{rk})$ *and* $\mathbf{B} = (b_{rk})$ *are* $m \times 2$ *integer matrices. If these variables cause a dependence of T on S at a level* $\ell$*, then the following conditions hold:*

(a) *The system of two equations*

$$\sum_{r=1}^{\ell-1} (a_{rk} - b_{rk}) i_r + (a_{\ell k} i_\ell - b_{\ell k} j_\ell) + \sum_{r=\ell+1}^{m} (a_{rk} i_r - b_{rk} j_r) = b_{0k} - a_{0k}, \qquad (k = 1, 2) \quad (6.22)$$

*has an integer solution; and*

(b) $\max_{\lambda} \overline{\mu}(\lambda) \le b_{01} - a_{01} \le \min_{\lambda} \overline{\nu}(\lambda)$,

*where*

$$\overline{\mu}(\lambda) = \sum_{r=1}^{\ell-1} \mu(a_{r1} - b_{r1} + \lambda(a_{r2} - b_{r2}), 0, p_r, q_r, 0) + \mu(a_{\ell 1} + \lambda a_{\ell 2}, -b_{\ell 1} - \lambda b_{\ell 2}, p_\ell, q_\ell, 1) + \sum_{r=\ell+1}^{m} [\mu(a_{r1} + \lambda a_{r2}, 0, p_r, q_r, 0) + \mu(-b_{r1} - \lambda b_{r2}, 0, p_r, q_r, 0)]$$

*and*

$$\overline{\nu}(\lambda) = \sum_{r=1}^{\ell-1} \nu(a_{r1} - b_{r1} + \lambda(a_{r2} - b_{r2}), 0, p_r, q_r, 0) + \nu(a_{\ell 1} + \lambda a_{\ell 2}, -b_{\ell 1} - \lambda b_{\ell 2}, p_\ell, q_\ell, 1) + \sum_{r=\ell+1}^{m} [\nu(a_{r1} + \lambda a_{r2}, 0, p_r, q_r, 0) + \nu(-b_{r1} - \lambda b_{r2}, 0, p_r, q_r, 0)].$$

PROOF. The proof of this theorem is also based on that of Theorem 6.2. Instead of the single dependence equation (6.9), we get the two equations of (6.22). For the dependence to exist, there must be

an integer solution to this system. (That is to be checked by the generalized gcd test of Section 5.4.) This is Condition (a).

To derive Condition (b), consider the Euclidean space $\mathbf{R}^{2m-\ell+1}$ and denote an arbitrary vector in it by

$$\mathbf{x} = (x_1, \ldots, x_{\ell-1}, x_\ell, y_\ell, x_{\ell+1}, y_{\ell+1}, \ldots, x_m, y_m).$$

Define two real-valued linear functions $f_1$ and $f_2$ on $\mathbf{R}^{2m-\ell+1}$ by

$$f_k(\mathbf{x}) = \sum_{r=1}^{\ell-1} (a_{rk} - b_{rk})x_r + (a_{\ell k}x_\ell - b_{\ell k}y_\ell) + \sum_{r=\ell+1}^{m} (a_{rk}x_r - b_{rk}y_r)$$

for $k = 1, 2$. Let $P$ denote the polytope in $\mathbf{R}^{2m-\ell+1}$ defined by the conditions in (6.10), when the integer variables are replaced by their real counterparts. In the real domain, the dependence equations (6.22) may be written in the form

$$f_1(\mathbf{x}) = b_{01} - a_{01}$$
$$f_2(\mathbf{x}) = b_{02} - a_{02}.$$

Since this system has a solution in $P$ by hypothesis, by Theorem A.10 we get

$$\max_{\lambda} \min_{\mathbf{x}\in P} [f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2)] \le c_1 \le \min_{\lambda} \max_{\mathbf{x}\in P} [f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2)].$$

The definitions of $f_1$ and $f_2$ imply

$$f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2) = \sum_{r=1}^{\ell-1} [(a_{r1} - b_{r1}) + \lambda(a_{r2} - b_{r2})]x_r + [(a_{\ell 1} + \lambda a_{\ell 2})x_\ell - (b_{\ell 1} + \lambda b_{\ell 2})y_\ell] + \sum_{r=\ell+1}^{m} [(a_{r1} + \lambda a_{r2})x_r - (b_{r1} + \lambda b_{r2})y_r].$$

Now, we can show that

$$\min_{\mathbf{x}\in P} [f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2)] = \overline{\mu}(\lambda)$$
$$\max_{\mathbf{x}\in P} [f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2)] = \overline{\nu}(\lambda)$$

as in the proof of Theorem 6.2. Condition (b) then follows and the proof is complete. □

In Example 6.3, we apply the bounds test to check for dependence caused by elements of a two-dimensional array. We consider dependence with a given direction vector to show how Theorem 6.3 can be extended to the two-dimensional case. The general bounds test and its subscript-by-subscript version are both illustrated.

---

**Example 6.3** Consider the program (from Example 3.1 in [Li 89])

```
L1 :    do I1 = 1, 50, 1
L2 :       do I2 = 2, 50, 1
   S :        X(2I1 + 3I2 + 50, 3I1 + I2 + 49) = ···
   T :        ··· = ··· X(I1 − I2 + 51, 2I1 − I2 + 48) ···
           enddo
        enddo
```

Let us study the dependence of $T$ on $S$ with the direction vector $(1, 1)$. The dependence equations are

$$2i_1 - j_1 + 3i_2 + j_2 = 1 \tag{6.23}$$

$$3i_1 - 2j_1 + i_2 + j_2 = -1. \tag{6.24}$$

First, we apply Theorem 6.3 separately to each individual dimension. The gcd test in each case is inconclusive. The inequalities for the bounds test applied separately to the two dimensions are

$$\begin{aligned}\mu(2,-1,1,50,1) + \mu(3,1,2,50,1) &\le 1 \\ &\le \nu(2,-1,1,50,1) + \nu(3,1,2,50,1),\\ \mu(3,-2,1,50,1) + \mu(1,1,2,50,1) &\le -1 \\ &\le \nu(3,-2,1,50,1) + \nu(1,1,2,50,1),\end{aligned}$$

or

$$-39 \le 1 \le 245 \quad \text{and} \quad -92 \le -1 \le 146,$$

both of which hold. Thus, if we rely on this subscript-by-subscript test, we have to assume dependence.

Next, we apply the bounds test in its generality. In the real domain, equations (6.23)-(6.24) can be written in the form

$$\left.\begin{aligned} f_1(\mathbf{x}) &= 1 \\ f_2(\mathbf{x}) &= -1, \end{aligned}\right\} \tag{6.25}$$

where

$$\begin{aligned} \mathbf{x} &= (x_1, y_1, x_2, y_2) \\ f_1(\mathbf{x}) &= 2x_1 - y_1 + 3x_2 + y_2 \\ f_2(\mathbf{x}) &= 3x_1 - 2y_1 + x_2 + y_2. \end{aligned}$$

Let $P$ denote the polytope in $\mathbf{R}^4$ defined by the inequalities

$$\begin{aligned} 1 \le x_1 \le 50, \qquad & 2 \le x_2 \le 50, \\ 1 \le y_1 \le 50, \qquad & 2 \le y_2 \le 50, \\ x_1 \le y_1 - 1, \qquad & x_2 \le y_2 - 1. \end{aligned}$$

By Theorem A.10, the system (6.25) has a real solution in $P$ iff

$$\max_{\lambda} \min_{\mathbf{x}\in P} [f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) + 1)] \le 1 \le \min_{\lambda} \max_{\mathbf{x}\in P} [f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) + 1)]. \quad (6.26)$$

Since

$$\begin{aligned} & f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) + 1) \\ &\quad = 2x_1 - y_1 + 3x_2 + y_2 + \lambda(3x_1 - 2y_1 + x_2 + y_2 + 1) \\ &\quad = (2 + 3\lambda)x_1 - (1 + 2\lambda)y_1 + (3 + \lambda)x_2 + (1 + \lambda)y_2 + \lambda, \end{aligned}$$

by Theorem 2.11 and Definition (2.19), we get

$$\begin{aligned} & \min_{\mathbf{x}\in P} [f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) + 1)] \\ &= \min_{\substack{1\le x_1\le 50 \\ 1\le y_1\le 50 \\ x_1\le y_1-1}} [(2 + 3\lambda)x_1 - (1 + 2\lambda)y_1] + \\ &\qquad \min_{\substack{2\le x_2\le 50 \\ 2\le y_2\le 50 \\ x_2\le y_2-1}} [(3 + \lambda)x_2 + (1 + \lambda)y_2] + \lambda \\ &= \mu(2 + 3\lambda, -1 - 2\lambda, 1, 50, 1) + \mu(3 + \lambda, 1 + \lambda, 2, 50, 1) + \lambda \\ &= [(1 + \lambda) - (1 + 2\lambda) + 48 \min(-1 - 2\lambda, 1 + \lambda, 0)] + \\ &\qquad [(4 + 2\lambda)2 + (1 + \lambda) + 47 \min(1 + \lambda, 4 + 2\lambda, 0)] + \lambda \\ &= 5\lambda + 9 + 48 \min(-1 - 2\lambda, 1 + \lambda, 0) + 47 \min(1 + \lambda, 4 + 2\lambda, 0). \end{aligned}$$

Since

$$\min(-1-2\lambda, 1+\lambda, 0) = \begin{cases} 1+\lambda & \text{if } \lambda \le -1 \\ 0 & \text{if } -1 \le \lambda \le -1/2 \\ -1-2\lambda & \text{if } -1/2 \le \lambda, \end{cases}$$

and

$$\min(1+\lambda, 4+2\lambda, 0) = \begin{cases} 4+2\lambda & \text{if } \lambda \le -3 \\ 1+\lambda & \text{if } -3 \le \lambda \le -1 \\ 0 & \text{if } -1 \le \lambda, \end{cases}$$

we have

$$\min_{\mathbf{x}\in P}[f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x})+1)] = \begin{cases} 8\lambda+14 & \text{if } \lambda \le -3 \\ 7\lambda+11 & \text{if } -3 \le \lambda \le -1 \\ 5\lambda+9 & \text{if } -1 \le \lambda \le -1/2 \\ 3\lambda+8 & \text{if } -1/2 \le \lambda. \end{cases}$$

Thus, this minimum is a piecewise-linear function of $\lambda$. In the interval $-\infty < \lambda \le -3$, its maximum value is equal to

$$\max\left(\lim_{\lambda\to-\infty}(8\lambda+14), 8(-3)+14\right) = -10.$$

There are similar results for the other three intervals. Hence, we have

$$\begin{aligned}\max_{\lambda}\min_{\mathbf{x}\in P}[f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x})+1)] &= \max(-10, 4, 13/2, \infty) \\ &= \infty.\end{aligned}$$

This shows that (6.26) cannot hold. Hence, $T$ does not depend on $S$ with the direction vector $(1,1)$.

---

Zhiyuan Li [Li 89] was the first to do a serious study of dependence analysis for multi-dimensional array elements, that did not involve subscript-by-subscript testing, nor array linearization. (See Exercise 2.) The general bounds method of this chapter, based on the theory presented in the appendix, includes Li's $\lambda$-test. His method of development, however, is completely different from ours, and we have not covered all his results. See [Li 89], [LiYe 89], and [LiYZ 90] for full details.

**EXERCISES 6.4**

1. (a) Generalize Theorem 6.6 to give necessary conditions for dependence with a given direction vector $\sigma$.
   (b) Generalize Theorem 6.7 to give necessary conditions for dependence with a given direction vector $\sigma$.
   (c) State the $n$-dimensional version of Theorem 6.7 and that of the theorem for a given direction vector obtained in the previous exercise.
   (d) Extend all results from a perfect loop nest to a general program.
2. Discuss the effect of array linearization in the context of dependence testing for multi-dimensional arrays. Explain the difficulties that may arise due to linearization. (See [WoBa 87] and [Bane 88a].)
3. In the following programs, test for the dependence of $T$ on $S$, $S$ on $T$, $S$ on $S$, and $T$ on $T$ at all possible levels and with all possible direction vectors:
   (a) The program of the example in Chapter 1;
   (b) The program of Example 3.3;
   (c) The program of Example 3.3 after changing the loop headers to

   $L_1$ : **do** $I_1 = 0, 100, 2$
   $L_2$ : **do** $I_2 = 2I_1 + 4, 2I_1 - 12, -1$

   (d) The program of Example 3.4;
   (e) The program of Example 6.3.

   Use both the subscript-by-subscript test and the general method of bounds. Compare your results.

## 6.5 General Method

In theory, the bounds test can be applied to elements of a multi-dimensional array in a general program, where loop limits are not necessarily constant. However, the complexity of the process increases rapidly with the number of the parameters (the $\lambda$'s and the $\mu$'s). In Example 6.4, we illustrate an application of the method of bounds to elements of a one-dimensional array, such that we have to deal with two $\lambda$ parameters. In this problem, we use Lagrangean relaxation to compute the minimum and maximum values of a linear function in a polytope (see Section A.4).

As pointed out in Section 6.1, arbitrary strides and regular loop nests do not add to the essential complexity of the bounds method; we simply change over to iteration variables in that case. In Exercise 3,

the reader is asked to generalize the results of this chapter along this line.

---

**Example 6.4** Consider the double loop

```
L1 :    do I1 = 0, 100, 1
L2 :       do I2 = 0, I1, 1
  S :         X(2I1 + 3I2) = ···
  T :         ··· = ··· X(4I1 − I2 + 5) ···
           enddo
        enddo
```

Suppose we want to test for the dependence of $T$ on $S$ with the direction vector $(1, 1)$. The dependence equation is

$$2i_1 - 4j_1 + 3i_2 + j_2 = 5. \tag{6.27}$$

Since $\gcd(2, 4, 3, 1) = 1$, the gcd test is inconclusive. The dependence constraints are

$$\begin{aligned} 0 \le i_1 \le 100, \qquad & 0 \le i_2 \le i_1, \\ 0 \le j_1 \le 100, \qquad & 0 \le j_2 \le j_1. \end{aligned}$$

The additional conditions imposed by the direction vector are

$$\begin{aligned} i_1 &\le j_1 - 1 \\ i_2 &\le j_2 - 1. \end{aligned}$$

Let $P$ denote the polytope in $\mathbf{R}^4$ defined by all these inequalities, when the integer variables $i_1, i_2, j_1$, and $j_2$ are replaced by real variables $x_1, x_2, y_1$, and $y_2$, respectively. Let $Q$ denote the polytope in $\mathbf{R}^4$ defined by the following inequalities:

$$\begin{aligned} 0 \le x_1 \le 100, \qquad & 0 \le x_2 \le 100, \\ 0 \le y_1 \le 100, \qquad & 0 \le y_2 \le 100, \\ x_1 \le y_1 - 1, \qquad & x_2 \le y_2 - 1. \end{aligned}$$

Then, $P$ is a subset of $Q$ specified by the additional conditions $x_2 \le x_1$ and $y_2 \le y_1$.

The dependence equation (6.27) can be written as $f(\mathbf{x}) = 5$, where

$$\mathbf{x} = (x_1, x_2, y_1, y_2)$$
$$f(\mathbf{x}) = 2x_1 - 4y_1 + 3x_2 + y_2.$$

The bounds test states that statement $T$ cannot depend on statement $S$ with the direction vector $(1, 1)$, unless

$$\min_{\mathbf{x}\in P} f(\mathbf{x}) \le 5 \le \max_{\mathbf{x}\in P} f(\mathbf{x}). \tag{6.28}$$

Now, we have

$$\begin{aligned}
&\min_{\mathbf{x}\in P} f(\mathbf{x}) \\
&= \min_{\substack{\mathbf{x}\in Q \\ x_2 \le x_1 \\ y_2 \le y_1}} f(\mathbf{x}) \\
&= \max_{\substack{\mu_1 \le 0 \\ \mu_2 \le 0}} \min_{\mathbf{x}\in Q} [f(\mathbf{x}) + \mu_1(x_1 - x_2) + \mu_2(y_1 - y_2)] \qquad \text{by Theorem A.9} \\
&= \max_{\substack{\mu_1 \le 0 \\ \mu_2 \le 0}} \min_{\mathbf{x}\in Q} [(\mu_1 + 2)x_1 + (\mu_2 - 4)y_1 + (3 - \mu_1)x_2 + (1 - \mu_2)y_2] \\
&= \max_{\substack{\mu_1 \le 0 \\ \mu_2 \le 0}} \left[ \begin{array}{l} \min\limits_{\substack{0 \le x_1 \le 100 \\ 0 \le y_1 \le 100 \\ x_1 \le y_1 - 1}} \{(\mu_1 + 2)x_1 + (\mu_2 - 4)y_1\} + \\ \qquad\qquad \min\limits_{\substack{0 \le x_2 \le 100 \\ 0 \le y_2 \le 100 \\ x_2 \le y_2 - 1}} \{(3 - \mu_1)x_2 + (1 - \mu_2)y_2\} \end{array} \right] \\
&= \max_{\substack{\mu_1 \le 0 \\ \mu_2 \le 0}} [(\mu_2 - 4) + 99 \min(\mu_2 - 4, \mu_1 + \mu_2 - 2, 0) + (1 - \mu_2) + \\
&\qquad\qquad 99 \min(1 - \mu_2, 4 - \mu_1 - \mu_2, 0)] \qquad \text{by Theorem 2.11} \\
&= -3 + 99 \max_{\substack{\mu_1 \le 0 \\ \mu_2 \le 0}} [\min(\mu_2 - 4, \mu_1 + \mu_2 - 2, 0) + \\
&\qquad\qquad\qquad\qquad \min(1 - \mu_2, 4 - \mu_1 - \mu_2, 0)].
\end{aligned}$$

Denoting the sum of these two minimum values by $g(\mu_1, \mu_2)$, we get

$$\min_{\mathbf{x}\in P} f(\mathbf{x}) = -3 + 99 \max_{\substack{\mu_1 \le 0 \\ \mu_2 \le 0}} g(\mu_1, \mu_2). \tag{6.29}$$

In the quadrant of the $\mu_1\mu_2$-plane where $\mu_1 \le 0$ and $\mu_2 \le 0$, the minimum values defining $g$ are given by

$$\min(\mu_2 - 4, \mu_1 + \mu_2 - 2, 0) = \begin{cases} \mu_2 - 4 & \text{if } -2 \le \mu_1, \mu_2 \le 4 \\ \mu_1 + \mu_2 - 2 & \text{if } \mu_1 + \mu_2 \le 2, \mu_1 \le -2 \\ 0 & \text{if } 4 \le \mu_2, 2 \le \mu_1 + \mu_2 \end{cases}$$

$$= \begin{cases} \mu_1 + \mu_2 - 2 & \text{if } \mu_1 \le -2 \\ \mu_2 - 4 & \text{if } \mu_1 \ge -2, \end{cases}$$

and

$$\min(1 - \mu_2, 4 - \mu_1 - \mu_2, 0) = \begin{cases} 1 - \mu_2 & \text{if } \mu_1 \le 3, 1 \le \mu_2 \\ 4 - \mu_1 - \mu_2 & \text{if } 3 \le \mu_1, 4 \le \mu_1 + \mu_2 \\ 0 & \text{if } \mu_2 \le 1, \mu_1 + \mu_2 \le 4 \end{cases}$$

$$= 0.$$

Then, we have

$$\max_{\substack{\mu_1 \le 0 \\ \mu_2 \le 0}} g(\mu_1, \mu_2) = \max_{\substack{\mu_1 \le 0 \\ \mu_2 \le 0}} [\min(\mu_2 - 4, \mu_1 + \mu_2 - 2, 0) + 0]$$

$$= \max\left( \max_{\substack{\mu_1 \le -2 \\ \mu_2 \le 0}} (\mu_1 + \mu_2 - 2), \max_{\substack{-2 \le \mu_1 \le 0 \\ \mu_2 \le 0}} (\mu_2 - 4) \right)$$

$$= \max(-4, -4)$$

$$= -4,$$

so that

$$\min_{\mathbf{x} \in P} f(\mathbf{x}) = -399$$

from (6.29).

We can compute $\max_{\mathbf{x} \in P} f(\mathbf{x})$ in a similar way, and then test Condition (6.28) to decide the existence of the dependence. The details are left to the reader.

**EXERCISES 6.5**

1. Complete Example 6.4.
2. In Example 6.4, check for dependences with other direction vectors and at different dependence levels.
3. State the analogs of the theorems in this chapter (and their generalizations asked for in Exercise 6.4.1), when loops are allowed arbitrary strides and the loop nest(s) involved are regular.

# Chapter 7

# Method of Elimination

## 7.1 Introduction

In this chapter, we discuss the method of elimination as outlined in Algorithm 1.3. We have already used elimination at several places in the book. A simple dependence problem, by definition, is one where elimination is trivial. For a simple problem, the method of elimination lets us find integer solutions almost as easily as real solutions. The prime example of that was seen in Chapter 2 for dependence problems involving one-dimensional arrays in single loops. Our focus here is on problems where that is not the case. We have already seen dependence problems where elimination is nontrivial, for instance, in the example of Chapter 1 and in Example 3.4.

In the first part of the method of elimination (steps 1–3, Algorithm 1.3), we decide if the system of dependence equations has an integer solution, and find the general solution in case it does. Deciding the existence of an integer solution constitutes the generalized gcd test that was covered in Section 5.4. We pick up the thread in Section 7.2 at the point where we left off in Chapter 5, and explain in detail how the method of elimination works. Section 7.3 is devoted to a study of two-variable dependence problems; they have some special properties. Finally, in Section 7.4, we look into other elimination approaches one might take.

## 7.2 Dependence Testing by Elimination

Consider the dependence problem in a general program as developed in Chapter 5. We are focusing on two statements $S$ and $T$ in the program, a program variable $X\left(\hat{\mathbf{I}}_S\hat{\mathbf{A}} + \hat{\mathbf{a}}_0\right)$ of $S$, and a program variable $X\left(\hat{\mathbf{I}}_T\hat{\mathbf{B}} + \hat{\mathbf{b}}_0\right)$ of $T$ (in their normalized forms). The dependence equation (5.3) for the problem posed by these variables is written again:

$$\hat{\imath}\hat{\mathbf{A}} - \hat{\jmath}\hat{\mathbf{B}} = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0.$$

It consists of $n$ scalar equations in $(m_S + m_T)$ variables: the $m_S$ elements of $\hat{\imath}$ and the $m_T$ elements of $\hat{\jmath}$. We can put this equation in the form

$$(\hat{\imath};\hat{\jmath})\begin{pmatrix}\hat{\mathbf{A}}\\ -\hat{\mathbf{B}}\end{pmatrix} = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0, \tag{7.1}$$

where $(\hat{\imath};\hat{\jmath})$ is the vector of size $(m_S + m_T)$ obtained by concatenating the elements of $\hat{\imath}$ with the elements of $\hat{\jmath}$, and the coefficient matrix of the equation is the $(m_S + m_T) \times n$ matrix obtained by concatenating the rows of $\hat{\mathbf{A}}$ with the rows of $-\hat{\mathbf{B}}$.

Use Algorithm I.2.1 to find an $(m_S + m_T) \times (m_S + m_T)$ unimodular matrix $\mathbf{U}$ and an $(m_S + m_T) \times n$ echelon matrix $\mathbf{S}$, such that

$$\mathbf{U} \cdot \begin{pmatrix}\hat{\mathbf{A}}\\ -\hat{\mathbf{B}}\end{pmatrix} = \mathbf{S}.$$

By Theorem I.3.6, there is an integer solution $(\hat{\imath};\hat{\jmath})$ to (7.1), iff there is an integer vector $\mathbf{t}$ of size $(m_S + m_T)$ satisfying the equation

$$\mathbf{tS} = \hat{\mathbf{b}}_0 - \hat{\mathbf{a}}_0. \tag{7.2}$$

If there is no such $\mathbf{t}$, then there is no dependence between $S$ and $T$. So, assume there is an integer solution $\mathbf{t}$ to (7.2). Then, by the same theorem, the general solution to (7.1) is given by

$$(\hat{\imath};\hat{\jmath}) = \mathbf{tU}, \tag{7.3}$$

where $\mathbf{t}$ is the general solution to (7.2). The number of elements of $\mathbf{t}$ completely determined by (7.2) equals rank($\mathbf{S}$), that is, rank $\begin{pmatrix}\hat{\mathbf{A}}\\ -\hat{\mathbf{B}}\end{pmatrix}$.

The number of undetermined elements of $\mathbf{t}$ is $(m_S + m_T - \text{rank}(\mathbf{S}))$. These latter elements appear as parameters in the general solution (7.3) to Equation (7.1). We can express $\hat{\imath}$ and $\hat{\jmath}$ separately in the form

$$\left.\begin{aligned} \hat{\imath} &= \mathbf{t}\mathbf{U}_1 \\ \hat{\jmath} &= \mathbf{t}\mathbf{U}_2, \end{aligned}\right\} \tag{7.4}$$

where $\mathbf{U}_1$ is the submatrix of $\mathbf{U}$ consisting of the leftmost $m_S$ columns, and $\mathbf{U}_2$ the submatrix consisting of the rightmost $m_T$ columns.

We rewrite the dependence constraints (5.4)–(5.5) for this problem:

$$\left.\begin{aligned} \mathbf{0} &\le \hat{\imath} \\ \hat{\imath}\hat{\mathbf{Q}}_S &\le \hat{\mathbf{q}}_{S0} \\ \mathbf{0} &\le \hat{\jmath} \\ \hat{\jmath}\hat{\mathbf{Q}}_T &\le \hat{\mathbf{q}}_{T0}, \end{aligned}\right\} \tag{7.5}$$

where $\hat{\mathbf{Q}}_S$ is the normalized final matrix and $\hat{\mathbf{q}}_{S0}$ the normalized final vector of $\mathbf{L}_S$, and $\hat{\mathbf{Q}}_T$ is the normalized final matrix and $\hat{\mathbf{q}}_{T0}$ the normalized final vector of $\mathbf{L}_T$. Substituting for $\hat{\imath}$ and $\hat{\jmath}$ from (7.4) into (7.5), we get

$$\left.\begin{aligned} \mathbf{0} &\le \mathbf{t}\mathbf{U}_1 \\ \mathbf{t}\left(\mathbf{U}_1\hat{\mathbf{Q}}_S\right) &\le \hat{\mathbf{q}}_{S0} \\ \mathbf{0} &\le \mathbf{t}\mathbf{U}_2 \\ \mathbf{t}\left(\mathbf{U}_2\hat{\mathbf{Q}}_T\right) &\le \hat{\mathbf{q}}_{T0}. \end{aligned}\right\} \tag{7.6}$$

There are $(m_S + m_T - \text{rank}(\mathbf{S}))$ variables in this system: those undetermined elements of $\mathbf{t}$. If these inequalities have an integer solution, then there is a dependence between $S$ and $T$. The second part of the method of elimination (steps 5–7, Algorithm 1.3) consists of finding real solutions to this system of inequalities by Fourier elimination, and then extrapolating the existence of integer solutions. This kind of elimination was used in [Bane 79]. Example 7.1 given below is Example 3 in Chapter 3 of that thesis, changed to suit our current notation. A similar approach is taken for the Power test in [WoTs 92].

Suppose $(\hat{\imath}, \hat{\jmath}) = (\mathbf{t}\mathbf{U}_1, \mathbf{t}\mathbf{U}_2)$ is a solution to Equation (7.1) such that $\mathbf{t}$ satisfies the inequalities (7.6). Let $\mathbf{i}$ denote the index point of the loop nest $\mathbf{L}_S$ corresponding to the iteration point $\hat{\imath}$, and $\mathbf{j}$ the index point of the loop nest $\mathbf{L}_T$ corresponding to the iteration point $\hat{\jmath}$. Let $\mathbf{d}$ denote the distance from the instance $S(\mathbf{i})$ of $S$ to the instance $T(\mathbf{j})$ of $T$.

Let $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(m_S+m_T)}$ denote the columns of $\mathbf{U}$. We can write

$$\hat{\imath} = \left(\mathbf{t}\mathbf{U}^{(1)}, \mathbf{t}\mathbf{U}^{(2)}, \ldots, \mathbf{t}\mathbf{U}^{(m_S)}\right)$$
$$\hat{\jmath} = \left(\mathbf{t}\mathbf{U}^{(m_S+1)}, \mathbf{t}\mathbf{U}^{(m_S+2)}, \ldots, \mathbf{t}\mathbf{U}^{(m_S+m_T)}\right)$$
$$\mathbf{d} = \left(\mathbf{t}\mathbf{U}^{(m_S+1)} - \mathbf{t}\mathbf{U}^{(1)}, \mathbf{t}\mathbf{U}^{(m_S+2)} - \mathbf{t}\mathbf{U}^{(2)}, \ldots, \mathbf{t}\mathbf{U}^{(m_S+m)} - \mathbf{t}\mathbf{U}^{(m)}\right).$$

If $\mathbf{d} \succ \mathbf{0}$, then $T$ depends on $S$ with the distance vector $\mathbf{d}$. If $\mathbf{d} \prec \mathbf{0}$, then $S$ depends on $T$ with the distance vector $-\mathbf{d}$. If $\mathbf{d} = \mathbf{0}$ and $S < T$, then $T$ depends on $S$ with the distance vector $\mathbf{0}$.

Consider next the dependence problem with a given direction vector $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_m)$. There are now $m$ additional conditions:

$$\left.\begin{array}{ll} \hat{\imath}_r = \hat{\jmath}_r & \text{if } \sigma_r = 0 \\ \hat{\imath}_r \le \hat{\jmath}_r - 1 & \text{if } \sigma_r = 1 \\ \hat{\jmath}_r \le \hat{\imath}_r - 1 & \text{if } \sigma_r = -1. \end{array}\right\} \tag{7.7}$$

As mentioned at the end of Section 5.4, we can handle these additional conditions in several different ways:

1. Create a larger system of equations and inequalities by attaching the equalities of (7.7) to (7.1), and the inequalities in (7.7) to (7.5). Process that new system by Algorithm 1.3.

2. Use the equalities of (7.7) to eliminate variables in (7.1) and (7.5), and create a system of equations and inequalities with fewer variables. Process that new system by Algorithm 1.3.

3. Solve the system of equations (7.1) and derive the system of inequalities (7.6). Restate (7.7) in terms of the unknown parameters:

$$\left.\begin{array}{rll} \mathbf{t}\mathbf{U}^{(r)} = \mathbf{t}\mathbf{U}^{(m_S+r)} & & \text{if } \sigma_r = 0 \\ \mathbf{t}\mathbf{U}^{(r)} \le \mathbf{t}\mathbf{U}^{(m_S+r)} - 1 & & \text{if } \sigma_r = 1 \\ \mathbf{t}\mathbf{U}^{(m_S+r)} \le \mathbf{t}\mathbf{U}^{(r)} - 1 & & \text{if } \sigma_r = -1. \end{array}\right\} \tag{7.8}$$

Attach these conditions to (7.6) and solve the larger system of inequalities.

---

**Example 7.1** Consider the following program (from Example 3 in Chapter 3 of [Bane 79]):

```
L1 :      do I1 = 0, 20, 1
L2 :        do I2 = 0, 10, 1
  S :         X(2I1 + 8I2 − 50) = ···
            enddo
L3 :        do I3 = 0, 20, 1
  T :         ··· = ··· X(12I1 + 16I3 + 46) ···
            enddo
          enddo
```

The dependence equation is

$$2i_1 - 12j_1 + 8i_2 - 16j_3 = 96.$$

Its general solution (Theorem I.3.5) can be written as

$$(i_1, j_1, i_2, j_3) = (48 + 2t_2, t_2 + 2t_3, t_2 + 3t_3 + 2t_4, t_4),$$

where $t_2, t_3$, and $t_4$ are arbitrary integer parameters. Suppose we want to check if there is a dependence of $T$ on $S$. Then, the conditions on $i_1$, $j_1$, $i_2$, and $j_3$ are

$$0 \le i_1 \le 20, \quad 0 \le i_2 \le 10, \quad i_1 \le j_1.$$
$$0 \le j_1 \le 20, \quad 0 \le j_3 \le 20,$$

Substituting the general solution into these conditions, we get

$$\begin{aligned} 0 &\le 48 + 2t_2 &&\le 20 \\ 0 &\le t_2 + 2t_3 &&\le 20 \\ 0 &\le t_2 + 3t_3 + 2t_4 &&\le 10 \\ 0 &\le t_4 &&\le 20 \\ 48 &+ 2t_2 &&\le t_2 + 2t_3. \end{aligned}$$

Algorithm I.3.2 (Fourier) applied to this system will show that it has no real solution. Therefore, the specified dependence does not exist.

---

**EXERCISES 7.2**

1. Use the method of elimination (as explained in this section) to test for the dependence of $T$ on $S$, $S$ on $T$, $S$ on $S$, and $T$ on $T$ in the following programs. Find the levels and direction vectors for each dependence, and the distance vectors corresponding to each valid direction vector:
   (a) The program of the example in Chapter 1;
   (b) The program of Example 3.4;
   (c) The program of Example 3.5;
   (d) The program used in examples 5.1–5.3.

## 7.3 Two-variable Problems

In dependence problems for real programs, it is commonly observed that each dependence equation or constraint has no more than two variables. On the other hand, each of the additional conditions imposed by a given direction vector always involves two variables (e.g., $\hat{\imath}_1 < \hat{\jmath}_1$). Thus, we often see that a linear dependence problem consists of a system of two-variable linear equations and a system of two-variable linear inequalities. Such systems have certain special properties that make them interesting and easier to solve than a more general system. We have already considered several two-variable problems in the book. In this section, we will focus on general methods for handling such problems. Note that some symbols used here may have different meanings in the rest of the book.

Consider a system of $n$ linear diophantine equations

$$\mathbf{xA} = \mathbf{c} \tag{7.9}$$

in $m$ variables: $x_1, x_2, \ldots, x_m$, where $\mathbf{A}$ is an $m \times n$ integer matrix and $\mathbf{c}$ an integer $n$-vector. If the coefficient of a variable in a given equation is nonzero, we may express that fact in many different ways. For example, we may say that the variable is *present* in the equation, or that the equation *has* the variable. Formally, a *two-variable system of linear equations* is defined to be a system of the form (7.9) where each equation has at least one but no more than two variables. In terms of the matrix $\mathbf{A}$, this means that each of its columns has either one or two nonzero elements.

To solve a system of linear diophantine equations, we use Algorithm I.2.1 (echelon reduction) and Theorem I.3.6. In the case of a two-variable system of the form (7.9), we may reduce the number of searches for nonzero elements during the echelon reduction process, by taking into account the sparseness of the coefficient matrix $\mathbf{A}$. Thus, for each column of $\mathbf{A}$, we keep track of the (one or two) rows that contain the nonzero elements on that column.

Aspvall and Shiloach [AsSh 80] have given a fast algorithm for solving a two-variable system of linear equations in real variables. It is possible to extend their method so that we may solve a two-variable system of linear equations in integer variables by repeated applications of Algorithm 2.1 (extended Euclid) and Theorem 2.8. Since a

dependence problem typically has very few equations, the start-up cost of such an approach may not be justifiable. However, we can easily carry over from [AsSh 80] certain concepts and results that will help us better understand the process of solving two-variable integer systems. (Note that we use a different notation.)

For a two-variable system of the form (7.9), define an undirected multigraph $G$ with $m$ vertices and $n$ edges as follows. For each of the $m$ variables $x_1, x_2, \ldots, x_m$, there is a vertex in $G$. For each equation, there is an edge in $G$ such that if the equation has variables $x_i$ and $x_j$, then the edge joins the vertices corresponding to $x_i$ and $x_j$. We say that $G$ is the *graph associated with* the system (7.9). The two-variable system is *connected* if its graph is connected.[1]

In [AsSh 80], the following results were proved for a two-variable system of linear equations in real variables, but they apply equally well to a similar system in integer variables.

**Lemma 7.1** *Let $G$ denote the graph associated with a two-variable system $\mathbf{xA} = \mathbf{c}$ of $n$ linear diophantine equations in $m$ variables.*

(a) *If $G$ is a tree, then* $\text{rank}(\mathbf{A}) = \text{m} - 1$.

(b) *If $G$ is connected, then* $\text{rank}(\mathbf{A}) \geq \text{m} - 1$.

From this lemma, we can derive important conclusions about the occurrence of undetermined parameters in the general solution to a two-variable system of linear equations.

**Theorem 7.2** *If a two-variable system $\mathbf{xA} = \mathbf{c}$ of linear diophantine equations has an integer solution, then the general solution can be written in a form where the expression for each variable has at most one undetermined integer parameter.*

*If the system is connected, then the entire general solution has no more than one integer parameter.*

PROOF. Let $G$ denote the graph associated with the system $\mathbf{xA} = \mathbf{c}$. Assume that $G$ has been broken up into its (weakly) connected components. This leads to a decomposition of the original system $\mathbf{xA} = \mathbf{c}$

[1] A *multigraph* is a graph that allows multiple edges between a pair of vertices. The concept of connectedness in an undirected graph or multigraph can be defined in the same way we defined weak connectedness for a digraph in Section I.1.4.

into disjoint subsystems. Each component of $G$ represents a connected subsystem; the subsystems corresponding to two different components have no variables or equations in common; and the subsystems formed this way make up the entire system of equations. In view of this, it suffices to prove the second part of the theorem.

We assume that the system $\mathbf{xA} = \mathbf{c}$ is connected and has a solution. We will show that the general solution has no more than one integer parameter. By Algorithm I.2.1, we find an $m \times m$ unimodular matrix $\mathbf{U}$ and an $m \times n$ echelon matrix $\mathbf{S}$ such that $\mathbf{UA} = \mathbf{S}$. Since the system $\mathbf{xA} = \mathbf{c}$ has an integer solution, so does the system $\mathbf{tS} = \mathbf{c}$ (Theorem I.3.6). Some of the elements of $\mathbf{t}$ get unique values and some remain undetermined. These undetermined elements of $\mathbf{t}$ become the (integer) parameters in the general solution $\mathbf{x} = \mathbf{tU}$ to $\mathbf{xA} = \mathbf{c}$. We have

$$\begin{aligned}\text{Number of parameters} &= \text{Number of zero rows of } \mathbf{S} \\ &= m - \text{rank}(\mathbf{S}) \\ &= m - \text{rank}(\mathbf{A}) \\ &\leq 1,\end{aligned}$$

since $\text{rank}(\mathbf{A}) \geq m - 1$ by Lemma 7.1. □

A *two-variable system of linear inequalities* is defined to be a system of the form $\mathbf{xA} \leq \mathbf{c}$ where each inequality has at least one but no more than two variables. Consider a dependence problem with a two-variable system of equations and a two-variable system of inequalities. Suppose the generalized gcd test passes, so that the equations have an integer solution. By Theorem 7.2, the general solution is such that at most one integer parameter appears in the expression for each iteration value. When this solution is substituted in the dependence constraints, we again get a two-variable system of inequalities in those parameters. The question then boils down to solving a system of two-variable inequalities.

It may happen that the final set of inequalities can be partitioned into disjoint subsystems, such that each subsystem has no more than one parameter. Then the dependence problem becomes simple. We saw this in Special Case 2 of Section 4.6 with regular loop nests. When the inequalities cannot be partitioned this way, we have to use some general algorithm that decides if a given two-variable system of lin-

ear inequalities has a real solution. A lot of work has been done on finding such an algorithm over the years. Shostak's paper [Shos 81] laid the foundation for a line of research that has produced a number of articles. The algorithms in [HoNa 94] are particularly interesting in that they are up-to-date and easy to follow. See the same paper for a list of other references.

We end this section by pointing out that a system of inequalities of the form $x \leq y + c$ has an integer solution iff it has a real solution; see Exercise I.3.6.2 and Section 4 of Shostak's paper.

---

**Example 7.2** Consider the problem of testing for dependence between statements $S$ and $T$ in the double loop

```
L1 :    do I1 = 1,100,1
L2 :       do I2 = 1,I1 - 1,1
   S :        X(I1,I2) = ···
   T :        ··· = ··· X(I2,I1) ···
           enddo
        enddo
```

The dependence equations are

$$i_1 - j_2 = 0$$
$$i_2 - j_1 = 0.$$

The general solution to these equations can be written as

$$(i_1, j_1, i_2, j_2) = (t_1, t_2, t_2, t_1),$$

where $t_1$ and $t_2$ are integer parameters. The dependence constraints are

$$1 \leq i_1 \leq 100$$
$$1 \leq j_1 \leq 100$$
$$1 \leq i_2 \leq i_1 - 1$$
$$1 \leq j_2 \leq j_1 - 1.$$

Substituting the general solution into the constraints, we get the following system of inequalities:

$$1 \leq t_1 \leq 100$$
$$1 \leq t_2 \leq 100$$
$$1 \leq t_2 \leq t_1 - 1$$
$$1 \leq t_1 \leq t_2 - 1.$$

It is easy to see that this system is inconsistent, since the right-hand inequalities of the last two lines give

$$\begin{aligned} 1 &\le t_1 - t_2 \\ t_1 - t_2 &\le -1. \end{aligned}$$

This inconsistency will be caught by Algorithm I.3.2. Thus, there is no dependence between statements $S$ and $T$.

---

**EXERCISES 7.3**

1. Give examples of programs where a dependence problem consists of a system of two-variable equations and a system of two-variable inequalities. Include programs where loops do not necessarily have constant limits and unit strides. Discuss the change in the two-variable nature of a problem as we pass from index values to iteration values.
2. Modify Algorithm I.2.1 (echelon reduction) to make it run efficiently on matrices that have no more than two nonzero elements on each column.
3. If a two-variable system of $n$ linear equations in $m$ variables is connected, then show that $n \ge m - 1$.

## 7.4 Other Methods

In Algorithm 1.2, we first decide if there is an integer solution to the dependence equations. If there is such a solution, then we decide if there is a real solution to the equations that satisfies the dependence constraints. The method of bounds and the method of elimination are both implementations of Algorithm 1.2. Each method uses the generalized gcd test for the first step. In the method of bounds, the second step consists of the bounds test which is based on computation of bounds (extreme values) of linear functions in certain polytopes. In the method of elimination, the general solution to the equations (that is easily obtained during application of the generalized gcd test) is substituted in the dependence constraints, and then Fourier's algorithm is applied to the resulting system of inequalities.

We can slightly improve the method of elimination if we extend Algorithm I.3.2 along the lines suggested in Section I.3.6. For example, suppose that traditional Fourier elimination (of a set of linear inequalities in integer variables $t_1, t_2, \ldots$) has yielded a system of the

form:

$$\begin{aligned} 1/3 \le t_1 &\le 1/2 \\ b_2(t_1) \le t_2 &\le B_2(t_1) \\ b_3(t_1, t_2) \le t_3 &\le B_3(t_1, t_2) \\ &\vdots \end{aligned}$$

Since the $t$'s are integers, we can get the more restricted set of inequalities, where each lower bound is replaced by its ceiling and each upper bound by its floor. For this particular example, that step will clearly show that there are no integer vectors satisfying the given set of inequalities. (The range of $t_1$ is $\lceil 1/3 \rceil \le t_1 \le \lfloor 1/2 \rfloor$, which is empty.) We have used this extra step after elimination throughout the book.

There is yet another way to decide the existence of a real solution to equations and inequalities. Instead of substituting the general solution in the dependence constraints and then eliminating the unknown parameters, we eliminate the original variables directly from the original system of dependence equations and constraints. (We can actually find the extreme values of the bounds method this way [Duff 74]. However, finding extreme values is only a means, not the goal for us.) If we do not use the generalized gcd test with this method, then we are losing a vital test for the existence of integer solutions. On the other hand, if we do use the two tests together, then we are not fully utilizing the work already done for the generalized gcd test, and ending up eliminating more variables than in the method of elimination.[2]

For elimination of variables from linear inequalities, Fourier's algorithm (Algorithm I.3.2) is the standard method of choice. This algorithm has been applied to problems in dependence analysis by many authors ([Bane 79], [Kuhn 80], [TrIf 86], [WoTs 92], and [Pugh 92a]).

In the remainder of this section, we briefly discuss two other tests for dependence: the $I$ test and the Omega test.

### The $I$ Test

The $I$ test ([KoKP 91], [PsKK 93]) tries to extend the range of applicability and the accuracy of the method of bounds in certain situations:

[2]Once we have found the general integer solution to a system of linear diophantine equations, the general *real* solution is obtained simply by changing the undetermined integer parameters into undetermined real parameters.

The array involved is one-dimensional, loops have constant limits, and the coefficients of the dependence equation are sufficiently small (at least one coefficient is $\pm 1$). This test is based on the following observation: Given integers $a_1, a_2, \ldots, a_m$, not all zero, and integers $\mu$ and $\nu$, there are integers $x_1, x_2, \ldots, x_m$ such that

$$\mu \leq a_1 x_1 + a_2 x_2 + \cdots + a_m x_m \leq \nu$$

iff $\lceil \mu/g \rceil \leq \lfloor \nu/g \rfloor$, where $g = \gcd(a_1, a_2, \ldots, a_m)$. We consider below an example where the $I$ test makes a correct prediction and the bounds test is inconclusive.

---

**Example 7.3** The following dependence problem appears in [KoKP 91]: The dependence equation is

$$i_1 - 3i_2 + 7i_3 = 8, \tag{7.10}$$

and the dependence constraints are

$$1 \leq i_1 \leq 3, 1 \leq i_2 \leq 2, 1 \leq i_3 \leq 4. \tag{7.11}$$

Since $\gcd(1, 3, 7) = 1$, the gcd test says that there are integer solutions to (7.10). In the polytope defined by (7.11), the extreme values of the left-hand-side of the equation are 2 and 28. Since 8 lies between 2 and 28, the bounds test says that there is a real solution to the equation that satisfies the constraints. Relying on the method of bounds, we would assume a dependence in this case. However, a simple enumeration shows that there is no integer solution $(i_1, i_2, i_3)$ to (7.10) that satisfies (7.11). For this example, the $I$ test will correctly conclude that there is no dependence.

The method of elimination will also give the correct conclusion in this case. The general solution to Equation (7.10) is

$$\begin{aligned} i_1 &= 3t_2 - 7t_3 + 8 \\ i_2 &= t_2 \\ i_3 &= t_3, \end{aligned}$$

where $t_2$ and $t_3$ are undetermined integer parameters. Substituting this solution in the constraints (7.11), we get

$$\left.\begin{array}{rrr} -7 \leq & 3t_2 - 7t_3 & \leq -5 \\ 1 \leq & t_2 & \leq 2 \\ 1 \leq & & t_3 \leq 4. \end{array}\right\} \tag{7.12}$$

Elimination of the parameter $t_2$ yields the inequalities

$$8/7 \leq t_3 \leq 13/7 \quad \text{and} \quad 1 \leq t_3 \leq 4.$$

Since the range $\lceil 8/7 \rceil \leq t_3 \leq \lfloor 13/7 \rfloor$ is empty, there is no integer $t_3$ that satisfies this system . Hence, there is no dependence.

We can get the same answer if we compute the extreme values of the function

$$i_3 = (8 - i_1 + 3i_2)/7$$

in the polytope defined by (7.11). They turn out to be 8/7 and 13/7. Since $i_3$ is an integer, its range is $\lceil 8/7 \rceil \leq i_3 \leq \lfloor 13/7 \rfloor$, which is empty.

Note that the limits 1 and 4 on $i_3$ in (7.11) were not used in the above two methods; they are not needed for an application of the *I* test either. Note also that the order in which the variables are eliminated is important in the *I* test and in the elimination process.

---

## The Omega Test

The Omega test [Pugh 92a] is based on the least remainder algorithm [Knut 81] (which is a variation of Euclid's algorithm) and Fourier's method of elimination. The following discussion is a summary of the description of the test given in [Hagh 96]. We start with a linear dependence problem defined by a set of dependence equations and a set of dependence constraints.

The first part of the Omega test deals with the elimination of variables from equality constraints by a known method based on Euclid's algorithm. The general method ([Knut 81], [Malm 80]) uses the following procedure. At each step, a nonzero coefficient of smallest absolute value in the system of equations is chosen. Without loss of generality, suppose that $a_1 (> 0)$ is such a coefficient that appears in an equation of the form

$$a_1 x_1 + a_2 x_2 + \cdots + a_m x_m = a_0. \tag{7.13}$$

If $a_1 = 1$, use this equation to eliminate the variable $x_1$ from other equations in the system. If no more equations remain, a general solution has essentially been obtained. Suppose next that $a_1 > 1$. Then,

if $a_1$ divides $a_2, a_3, \ldots, a_m$, check that $a_1$ also divides $a_0$. If $a_1$ does not divide $a_0$, then there is no integer solution to the system. Otherwise, divide both sides of Equation (7.13) by $a_1$ and eliminate $x_1$ as in the previous case. Finally, if $a_1$ fails to divide all the coefficients $a_2, a_3, \ldots, a_m$, then introduce a new variable $x_1'$ by

$$\lfloor a_1/a_1 \rfloor x_1 + \lfloor a_2/a_1 \rfloor x_2 + \cdots + \lfloor a_m/a_1 \rfloor x_m = x_1'. \tag{7.14}$$

Solve this equation for $x_1$ and replace (7.13) by the equation

$$a_1 x_1' + (a_2 \bmod a_1) x_2 + \cdots + (a_m \bmod a_1) x_m = a_0. \tag{7.15}$$

Since at each step either the number of equations or the absolute value of the smallest coefficient in the system is reduced, the above procedure has to terminate. The method is intimately connected to Euclid's algorithm and is easy to understand. It is not, however, the best method for solving the problem even though substantial refinements are possible [Knut 81].

One refinement is to base the procedure on the *least remainder* algorithm ([Knut 81], Section 4.5.3, exercises 30 and 31). This modification reduces the average number of division steps required by Euclid's algorithm and allows the system to reduce faster. The idea of the algorithm is to choose the *remainder of the smallest magnitude* for the next division step. Thus, instead of replacing $b$ by $(a \bmod b)$ during the division step of Euclid's algorithm, we replace it by $(a \bmod b - b)$ if $|a \bmod b| > \frac{1}{2}|b|$. This approach has been used in the Omega test to eliminate variables from equality constraints. An operator $\widehat{\bmod}$ is defined by

$$a \mathbin{\widehat{\bmod}} b = a - b\lfloor a/b + 1/2 \rfloor.$$

It is easy to verify that if $|a \bmod b| < \frac{1}{2}|b|$, then $a \mathbin{\widehat{\bmod}} b = a \bmod b$; otherwise, $a \mathbin{\widehat{\bmod}} b = a \bmod b - b$. At each division step, we replace $(a, b)$ by $(b, a \mathbin{\widehat{\bmod}} b)$.

The above modification affects the calculation iff the following division step in the original Euclid's algorithm would have a quotient of one. The analysis of this algorithm is related to the Fibonacci numbers and the golden ratio, and approximately 30.6% of the division steps

are saved[3] on the average [Knut 81]. The modified algorithm, however, makes each division step longer on many computers [Knut 81].

Knuth suggests an alternative approach that saves *all* division steps when the quotient is unity. Additional information related to the least remainder algorithm can be found in [BrDu 71], [GoZa 52], and [Moor 92]. For further study of efficient algorithms for solving the greatest common divisor problem, as well as other problems from the theory of numbers, the interested reader is referred to [BaSh 96].

After variables have been eliminated from all the equations, we are left with a system of linear inequalities. The Omega test employs Fourier's method repeatedly to eliminate variables from linear inequalities. Since classical Fourier elimination can only detect real solutions, certain steps are added to isolate integer solutions. To understand the process of finding integer solutions by elimination, consider a set of two inequalities of the form

$$\left.\begin{aligned} ax &\le \alpha \\ \beta &\le bx, \end{aligned}\right\} \tag{7.16}$$

where $x$ is a variable, $a$ and $b$ are positive integers, and $\alpha$ and $\beta$ are integer-valued expressions. Elimination of $x$ from (7.16) by Fourier Elimination (Algorithm I.3.2) results in the inequality

$$a\beta \le b\alpha. \tag{7.17}$$

The systems (7.16) and (7.17) are equivalent in the real domain, but they are not necessarily equivalent in the integer sense. An integer solution to (7.16) will also satisfy (7.17), but an integer solution to (7.17) may not always lead to an integer solution to (7.16). A sufficient (but not necessary) condition for the existence of an integer $x$ satisfying (7.16) is the inequality

$$b\alpha - a\beta \ge ab - a - b + 1 = (a-1)(b-1). \tag{7.18}$$

(See Exercise 4.) Note that when $a = 1$ or $b = 1$, (7.18) is trivially implied by (7.17).

[3]This choice of least remainder in absolute value, that results in the smallest number of iterations over all gcd algorithms that replace $(a, b)$ by $(b, (\pm a) \bmod b)$, has been observed by K. Vahlen in 1895 [Knut 81].

The Omega test uses Fourier's method to eliminate the variable $x$ from the system (7.16) as follows. The real solution space to this system (i.e., the solution space of (7.17)) is called the *real* shadow of (7.16). The subset of the real shadow that further satisfies the constraint (7.18) is called the *dark* shadow of (7.16). First, the real and the dark shadows are both computed. (In the general case, that means two applications of the Fourier elimination algorithm.) If the real shadow is empty, then there is no real, and hence no integer, solution to the system (7.16). Otherwise, if the dark shadow is nonempty, then there is an integer solution to (7.16). Finally, if the real shadow is nonempty but the dark shadow is empty, then an exhaustive search is performed on all integers $i$ in the range $0 \leq i \leq (ab - a - b)/a$, to see whether any of the equations of the form $bx = \beta + i$ has an integer solution. The search may be terminated as soon as the first affirmative answer is found; otherwise, the whole range has to be searched.

Observe that, from the above integer programming problem, the Omega test may create $\lfloor b - b/a \rfloor$ integer programming subproblems. This number is proportional to the absolute value of the coefficient $b$ (let $a = b$, then $\lfloor b - b/a \rfloor = b - 1$), and, thus, is exponential with respect to the input data. Each subproblem, in turn, is solved by the modified Fourier elimination method, a method that itself is an exponential algorithm. It has been claimed [Pugh 92] that in practice the Omega test is fast and its execution time rarely exceeds twice the time required to scan the problem input (i.e., the array subscripts and loop bounds). Several implementation details have also been used to improve the efficiency of the algorithms.

---

**Example 7.4** Let us use the Omega test approach, described in this section, to decide if the system

$$\left.\begin{aligned} 1000x &\leq 999 \\ 1 &\leq 1000x \end{aligned}\right\} \tag{7.19}$$

has an integer solution. The parameters of this system, using the notation of our discussion on the system (7.16), are

$$a = 1000, \alpha = 999, b = 1000, \beta = 1.$$

It is clear that (7.19) has real solutions since $1 \leq 999$. An equivalent reason, in the Omega test terminology, is that the real shadow of the system is not empty (i.e., $1000 \times 999 - 1000 \times 1 \geq 0$). Thus, the system may have integer solutions. The dark shadow of the system, however, is empty, since

$$1000 \times 999 - 1000 \times 1 \geq (1000 - 1)(1000 - 1)$$

does not hold. Thus, if the system (7.19) has an integer solution $x$, then the relation

$$1000x = i + 1 \tag{7.20}$$

holds for an integer $i$ in the range

$$0 \leq i \leq (1000 \times 1000 - 1000 - 1000)/1000.$$

Following the Omega test approach, we must perform an exhaustive search on all integers $i$ in the range of zero to 998 to see if the system (7.20) has a solution. All the 999 integers in that range must be searched, since none of the corresponding equations has a solution.

If both occurrences of 1000 in the problem statement of this exercise are replaced by 10000, then solving the problem by this approach involves enumeration of 9999 integers and solving the same number of diophantine equations. In the general case, each of these equalities will be solved (hence, eliminated) and the solution will be substituted in the augmented system. Then, the augmented system is solved by the above variant of Fourier elimination method.

---

**EXERCISES 7.4**

1. In the method of elimination used in Example 7.3, first eliminate the parameter $t_3$ from the inequalities (7.12). Compare your result with that obtained in the example.
2. Show by the method of elimination that there is no dependence in the problem represented by the dependence equation

   $$i_1 - 2i_2 + 8i_3 + 8j_3 = 26$$

   and the dependence constraints

   $$1 \leq i_1 \leq 3, 1 \leq i_2 \leq 3, 1 \leq i_3 < j_3 \leq 10.$$

   (This problem appears in [PsKK 93].)

3. Find several examples of dependence problems where the $I$ test ([KoKP 91], [PsKK 93]) makes a correct prediction while the method of bounds is inconclusive. (Use loops with at least 25 iterations each.)
4. A necessary and sufficient condition for the existence of an integer $x$ satisfying the inequalities (7.16) is the following:

$$\lceil \beta/b \rceil \leq \lfloor \alpha/a \rfloor.$$

Show that (7.18) implies this condition, and hence (7.18) is a sufficient condition for the existence of an integer $x$. Show also that (7.18) is not a necessary condition.

# Chapter 8

# Conclusions

In Chapters 1 through 7, we have tried to formulate a precise theory of dependence analysis in a program consisting of loops and assignment statements. We also discussed several methods for solving the dependence problem. The dependence problem consists of finding integer solutions to a system of linear equations and inequalities. Since a system of linear diophantine equations can be solved in polynomial time, it is always a good idea to solve the equations first. If there is no integer solution to the equations, there is no solution to the combined system, and therefore the particular dependence does not exist (generalized gcd test). If the equations admit integer solutions, then there are several options based on the nature of the problem.

If the dependence problem falls in the category of simple problems, we can easily find a complete solution. For a non-simple problem, the method of bounds may be used, if those bounds are easy to compute and only the existence of the dependence needs to be settled. When the polytope defined by the dependence constraints is such that its extreme points are hard to find, or that there are too many extreme points, then the computation of the bounds (minimum and maximum values in most cases) becomes hard.[1] When the method of bounds is not applicable, or when detailed information about iterations (involved in the dependence) is needed, we may use the method of elimination, perhaps with some limited enumeration.

[1]In fact, the efficiency of a linear programming algorithm resides in the cleverness of the way it navigates the set of extreme points.

For a general dependence problem, neither the method of bounds nor the method of elimination (without complete enumeration) will always be accurate. As a last resort, we can use an established integer programming algorithm, for example, Gomory's cutting plane method [Schr 87]. (See also [Wall 88] and [Feau 88].) But such an algorithm may not produce the set of all iteration pairs for the dependence.

Dependence problems found in real programs are usually simple enough to make the hierarchical approach described above quite useful. As a practical matter, one may also add several heuristics to a dependence test suite in a real compiler. For instance, pattern matching (e.g., checking for subscripts like $i, i+1, i-1$) can take care of many dependence problems. Also, limited enumeration can sometimes be used to decide if linear diophantine equations have nonnegative integer solutions [BoTr 76]. If there is no nonnegative integer solution at all, then there cannot be any solution satisfying the dependence constraints. (When a dependence problem is stated in terms of iteration variables of loops, all variables in the problem are nonnegative.)

When we infer the existence of an integer solution from the existence of a real solution (in the method of bounds or elimination), we may sometimes end up with a spurious dependence that should not be there. But, that is the price we pay for saving time on a more expensive algorithm. It should be noted that the bounds test is a sufficient condition for existence of integer solutions in certain situations; see [Bane 76], [Li 89], and [PsKK 91]. Furthermore, while a spurious dependence may result in preventing some transformations of the given program, it will not cause any *illegal* transformations. Thus, an approximate dependence test is conservative in the sense that it will preserve the integrity of the program.

We have tried to include a large number of references in the bibliography, not all of which have been cited in the text. However, this is not a complete list; more references can be found in the sources we have listed.

# Appendix A

# Linear Equations on Polytopes

## A.1 Introduction

Our main goal in this appendix is to study the existence of real solutions to a system of linear equations in a polytope. We give a minimal sequence of definitions and results (many without proof) that should prepare the reader to solve dependence problems by the method of bounds. However, the body of knowledge on polytopes is vast. For an in-depth study, a suitable book should be used; many books on the theory of linear and integer programming cover polytopes in detail.

The general reference for the appendix is [Schr 87]. However, our approach and notation are sometimes different from Schrijver's. Note also that some symbols used here could have meanings different from their meanings in the context of dependence analysis.

In Section A.2, we discuss some basic properties of polytopes. Section A.3 gives the necessary and sufficient condition under which a single linear equation has a real solution in a polytope. The method of Lagrangean relaxation is sketched in Section A.4. This method can often simplify computation of extreme values of linear functions in polytopes. It is applied in Section A.5 to the study of existence of real solutions to systems of real equations in a polytope. Finally, in Section A.6, we describe a situation where existence of a real solution (to a system of linear equations in a polytope) implies the existence of an integer solution. Note that some of the results stated here have more general versions.

## A.2 Polytopes

We consider vectors in $\mathbf{R}^m$ for some fixed $m \geq 1$. A set $X \subset \mathbf{R}^m$ is *convex* if the vector $\lambda\mathbf{x} + (1-\lambda)\mathbf{y}$ is always in $X$, whenever $\mathbf{x}$ and $\mathbf{y}$ are in $X$, and $\lambda$ is a real number such that $0 \leq \lambda \leq 1$. In other words, whenever the end points of a line segment are in a convex set, the line segment itself is included in the set.

If a set is not convex, we may want to know the smallest convex set that includes it. A *convex combination* of a finite set of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$ is an expression of the form

$$\mathbf{y} = \lambda_1\mathbf{x}_1 + \lambda_2\mathbf{x}_2 + \cdots + \lambda_p\mathbf{x}_p,$$

where $\lambda_1, \lambda_2, \ldots, \lambda_p$ are nonnegative real numbers such that

$$\lambda_1 + \lambda_2 + \cdots + \lambda_p = 1.$$

The *convex hull* of a set $X \subset \mathbf{R}^m$ is the set of all convex combinations of elements of $X$. The convex hull of $X$ is the smallest convex set that includes $X$ (Exercise 1). If $X$ itself is convex, it is closed under taking convex combinations of its elements (Exercise 2), and is therefore its own convex hull.

A set $X \subset \mathbf{R}^m$ is *bounded* if there is a positive number $\delta$, such that

$$-\delta \leq x_r \leq \delta \qquad (1 \leq r \leq m)$$

for each vector $(x_1, x_2, \ldots, x_m) \in X$. A *polyhedron* in $\mathbf{R}^m$ is a set of the form

$$P = \{\mathbf{x} \in \mathbf{R}^m : \mathbf{x}\mathbf{A} \leq \mathbf{b}\},$$

where $\mathbf{A}$ is an $m \times l$ real matrix and $\mathbf{b}$ a real $l$-vector ($l \geq 1$). A *polytope* is a bounded polyhedron.

**Lemma A.1** *Each polytope is a convex set.*

PROOF. Take any polytope $P = \{\mathbf{x} \in \mathbf{R}^m : \mathbf{x}\mathbf{A} \leq \mathbf{b}\}$. Let $\mathbf{x} \in P$, $\mathbf{y} \in P$, and $0 \leq \lambda \leq 1$. The vector $\mathbf{z} = \lambda\mathbf{x} + (1-\lambda)\mathbf{y}$ is in $P$, because

$$\begin{aligned}
\mathbf{z}\mathbf{A} &= (\lambda\mathbf{x} + (1-\lambda)\mathbf{y})A \\
&= \lambda\mathbf{x}\mathbf{A} + (1-\lambda)\mathbf{y}\mathbf{A} \\
&\leq \lambda\mathbf{b} + (1-\lambda)\mathbf{b}, \qquad \text{since } \mathbf{x}\mathbf{A} \leq \mathbf{b},\ \lambda \geq 0,\ \mathbf{y}\mathbf{A} \leq \mathbf{b},\ 1-\lambda \geq 0 \\
&= \mathbf{b}.
\end{aligned}$$

Hence, $P$ is convex. □

An *extreme point* (or *vertex*) of a polytope $P$ is a point $\mathbf{v} \in P$ such that $\mathbf{v}$ is not the convex combination of two distinct points in $P$. In other words, an extreme point is such that no nonempty line segment containing it is included in the polytope. For example, the vertices of a triangle are its extreme points.

**Theorem A.2** *Each extreme point of a polytope* $\{\mathbf{x} \in \mathbf{R}^m : \mathbf{x}\mathbf{A} \leq \mathbf{b}\}$, *where* $\mathbf{A}$ *is an* $m \times l$ *matrix, is the unique solution to a set of* $m$ *linearly independent equations from the system* $\mathbf{x}\mathbf{A} = \mathbf{b}$.

**Theorem A.3** *A polytope has a finite number of extreme points.*

**Theorem A.4** *Each polytope is the convex hull of the set of its extreme points.*

EXERCISES A.2

1. Prove that the convex hull of a set $X$ of vectors is the smallest convex set that includes $X$.
2. Show that a convex set is closed under taking convex combinations of its elements.
3. Let $\mathbf{x}, \mathbf{z}, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_p$ denote points in $\mathbf{R}^m$, such that $\mathbf{x}$ is a convex combination of $\mathbf{y}_1$ and $\mathbf{z}$, and $\mathbf{z}$ is a convex combination of $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_p$. Prove that $\mathbf{x}$ is a convex combination of $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_p$.

## A.3 Real Solutions to a Single Equation

A real-valued function $f$ on $\mathbf{R}^m$ is *linear* if

$$f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$
$$f(\lambda\mathbf{x}) = \lambda f(\mathbf{x})$$

for all $\mathbf{x} \in \mathbf{R}^m, \mathbf{y} \in \mathbf{R}^m$, and $\lambda \in \mathbf{R}$. Such a function has the form

$$f(\mathbf{x}) = \mathbf{a}\mathbf{x} = \mathbf{x}\mathbf{a}$$

for some fixed vector $\mathbf{a}$ in $\mathbf{R}^m$.

The following theorem is a basic result in linear programming. It says that in a polytope, a real-valued linear function has a minimum and a maximum value, and those values are attained at extreme points of that polytope.

**Theorem A.5** *Consider a real-valued linear function $f$ on $\mathbf{R}^m$ and a nonempty polytope $P \subset \mathbf{R}^m$. Then, $f$ has a minimum and a maximum value in $P$, and there are two extreme points $\mathbf{u}$ and $\mathbf{v}$ of $P$ such that*

$$f(\mathbf{u}) = \min_{\mathbf{x} \in P} f(\mathbf{x}) \qquad \text{and} \qquad f(\mathbf{v}) = \max_{\mathbf{x} \in P} f(\mathbf{x}).$$

PROOF. Let $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p\}$ denote the set of extreme points of $P$. Without any loss of generality, we may assume that

$$f(\mathbf{v}_1) \leq f(\mathbf{v}_2) \leq \cdots \leq f(\mathbf{v}_p).$$

The value of $f$ at any point $\mathbf{x} \in P$ can be expressed in terms of its values at the extreme points. Indeed, by Theorem A.4, $\mathbf{x}$ is a convex combination of the extreme points, so that we can write

$$\mathbf{x} = \lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2 + \cdots + \lambda_p \mathbf{v}_p,$$

where the $\lambda$'s are nonnegative reals satisfying $\lambda_1 + \lambda_2 + \cdots + \lambda_p = 1$. Then, the linearity of $f$ implies that

$$f(\mathbf{x}) = \lambda_1 f(\mathbf{v}_1) + \lambda_2 f(\mathbf{v}_2) + \cdots + \lambda_p f(\mathbf{v}_p).$$

Since

$$f(\mathbf{v}_1) \leq f(\mathbf{v}_i) \leq f(\mathbf{v}_p) \qquad (1 \leq i \leq p),$$

it follows that

$$\sum_{i=1}^{p} \lambda_i f(\mathbf{v}_1) \leq \sum_{i=1}^{p} \lambda_i f(\mathbf{v}_i) \leq \sum_{i=1}^{p} \lambda_i f(\mathbf{v}_p)$$

that is,

$$f(\mathbf{v}_1) \sum_{i=1}^{p} \lambda_i \leq \sum_{i=1}^{p} \lambda_i f(\mathbf{v}_i) \leq f(\mathbf{v}_p) \sum_{i=1}^{p} \lambda_i$$

or

$$f(\mathbf{v}_1) \leq f(\mathbf{x}) \leq f(\mathbf{v}_p).$$

This holds for all points $\mathbf{x} \in P$. Therefore, $f(\mathbf{v}_1)$ is the minimum value of $f$ in $P$, and $f(\mathbf{v}_p)$ is the maximum value. To complete the proof, we need only take $\mathbf{u} = \mathbf{v}_1$ and $\mathbf{v} = \mathbf{v}_p$. □

Using the above theorem, we establish the following result that gives a necessary and sufficient condition for the existence of a real solution to a linear equation.

**Theorem A.6** *Consider a real-valued linear function $f$ on $\mathbf{R}^m$ and a nonempty polytope $P \subset \mathbf{R}^m$. For a given real number $c$, the equation*

$$f(\mathbf{x}) = c \tag{A.1}$$

*has a solution $\mathbf{x} \in P$ iff*

$$\min_{\mathbf{x}\in P} f(\mathbf{x}) \le c \le \max_{\mathbf{x}\in P} f(\mathbf{x}). \tag{A.2}$$

PROOF. The *only if* part is trivial. If there is a point $\mathbf{x} \in P$ such that $f(\mathbf{x}) = c$, then (A.2) has to hold since each value of $f$ lies between the minimum and the maximum.

To prove the *if* part, assume that $c$ is a real number satisfying (A.2). By Theorem A.5, there are extreme points $\mathbf{u}$ and $\mathbf{v}$ of $P$, such that

$$f(\mathbf{u}) = \min_{\mathbf{x}\in P} f(\mathbf{x}) \qquad \text{and} \qquad f(\mathbf{v}) = \max_{\mathbf{x}\in P} f(\mathbf{x}).$$

Hence, we have $f(\mathbf{u}) \le c \le f(\mathbf{v})$. We will find a point $\mathbf{x} \in P$ such that $f(\mathbf{x}) = c$. We assume that $f$ is not constant on $P$, since there is not much to prove otherwise (why?). Then $f(\mathbf{v}) - f(\mathbf{u}) > 0$, and we can define

$$\lambda = \frac{f(\mathbf{v}) - c}{f(\mathbf{v}) - f(\mathbf{u})}.$$

Since

$$0 \le f(\mathbf{v}) - c \le f(\mathbf{v}) - f(\mathbf{u}),$$

it follows that $0 \le \lambda \le 1$. Since $P$ is convex and $\mathbf{u}$ and $\mathbf{v}$ are points in $P$, the convex combination $\mathbf{x} = \lambda\mathbf{u} + (1 - \lambda)\mathbf{v}$ of $\mathbf{u}$ and $\mathbf{v}$ is also a point in $P$. This point $\mathbf{x}$ satisfies $f(\mathbf{x}) = c$, since

$$\begin{aligned} f(\mathbf{x}) &= \lambda f(\mathbf{u}) + (1 - \lambda) f(\mathbf{v}) \\ &= \lambda(f(\mathbf{u}) - f(\mathbf{v})) + f(\mathbf{v}) \\ &= (c - f(\mathbf{v})) + f(\mathbf{v}) \\ &= c. \end{aligned}$$

The proof of the theorem is now complete. □

In view of Theorem A.5, the extreme values of $f$ in $P$ are equal to its extreme values in the set of vertices of $P$. This is formally stated below. (Thus, it is enough to evaluate $f$ at the vertices, which form a finite set.)

**Corollary 1** *Let $V$ denote the set of extreme points of $P$. Given a real number $c$, the equation $f(\mathbf{x}) = c$ has a solution in $P$ iff*

$$\min_{\mathbf{x}\in V} f(\mathbf{x}) \leq c \leq \max_{\mathbf{x}\in V} f(\mathbf{x}). \tag{A.3}$$

Theorems A.5 and A.6 are valid in a much more general setting. That generalization involves topological concepts and is beyond our scope. The interested reader is referred to sections 3.17 and 3.18 in [Dieu 69], or the description of the Intermediate Value Theorem in any advanced calculus text.

## A.4 Lagrangean Relaxation

The Lagrangean relaxation method is used to obtain upper bounds for certain integer programming problems (see Section 24.3 of [Schr 87].) In this section, we drop the integrality requirement and derive some results for linear programming.

First, we state a basic result in the theory of linear programming; for a proof, see Section 7.4 of [Schr 87].

**Theorem A.7** (Duality Theorem of Linear Programming). *Consider an $m \times l$ real matrix $\mathbf{A}$ and two vectors $\mathbf{a} \in \mathbf{R}^m$ and $\mathbf{b} \in \mathbf{R}^l$, such that*

$$\{\mathbf{x} \in \mathbf{R}^m : \mathbf{xA} \leq \mathbf{b}\} \quad \textit{and} \quad \{\mathbf{y} \in \mathbf{R}^l : \mathbf{Ay} = \mathbf{a}, \mathbf{y} \geq \mathbf{0}\}$$

*are nonempty sets. Then, we have*

$$\max_{\mathbf{xA}\leq\mathbf{b}} \mathbf{xa} = \min_{\substack{\mathbf{y}\geq\mathbf{0} \\ \mathbf{Ay}=\mathbf{a}}} \mathbf{by}.$$

There are several equivalent forms of this theorem. We will need the following form that gives the minimum of $\mathbf{xa}$ in the polyhedron defined by the inequalities $\mathbf{xA} \leq \mathbf{b}$.

**Corollary 1** *In the notation of Theorem A.7, the following holds:*

$$\min_{\mathbf{xA}\le\mathbf{b}} \mathbf{xa} = \max_{\substack{\mathbf{z}\le\mathbf{0}\\ \mathbf{Az}=\mathbf{a}}} \mathbf{bz}.$$

PROOF. We have

$$\begin{aligned}
\min_{\mathbf{xA}\le\mathbf{b}} \mathbf{xa} &= -\max_{\mathbf{xA}\le\mathbf{b}} \mathbf{x}(-\mathbf{a}) \\
&= -\min_{\substack{\mathbf{y}\ge\mathbf{0}\\ \mathbf{Ay}=-\mathbf{a}}} \mathbf{by} && \text{by Theorem A.7} \\
&= -\min_{\substack{\mathbf{z}\le\mathbf{0}\\ \mathbf{Az}=\mathbf{a}}} \mathbf{b}(-\mathbf{z}) && \text{where } \mathbf{z} = -\mathbf{y} \\
&= \max_{\substack{\mathbf{z}\le\mathbf{0}\\ \mathbf{Az}=\mathbf{a}}} \mathbf{bz}.
\end{aligned}$$

□

Theorem A.7 and its corollary can be very useful in the computation of extreme values of a linear function $f$ subject to a set of linear constraints. Suppose the set of constraints can be broken up into an *easy* part and a *hard* part. Attaching the hard constraints to $f$, we create a more complicated linear function, and then find its extreme values subject to the easy constraints. This is stated more formally in Theorem A.8 and its corollary. We assume that the polytopes in the results of this section are nonempty.

**Theorem A.8** *Consider a real-valued linear function $f : \mathbf{x} \mapsto \mathbf{xa}$ defined on $\mathbf{R}^m$, and a polytope $P = \{\mathbf{x} : \mathbf{xA} \le \mathbf{b}\}$ in $\mathbf{R}^m$, where $\mathbf{A}$ is an $m \times l$ matrix. Let $\mathbf{A}_1$ denote an $m \times l_1$ matrix and $\mathbf{b}_1$ an $l_1$-vector. The extreme values of $f$ in $P$, subject to the set of additional constraints $\mathbf{xA}_1 \le \mathbf{b}_1$, are given by the formulas:*

$$\max_{\substack{\mathbf{xA}\le\mathbf{b}\\ \mathbf{xA}_1\le\mathbf{b}_1}} \mathbf{xa} = \min_{\mathbf{y}\ge\mathbf{0}} \max_{\mathbf{xA}\le\mathbf{b}} [\mathbf{xa} + (\mathbf{b}_1 - \mathbf{xA}_1)\mathbf{y}] \tag{A.4}$$

$$\min_{\substack{\mathbf{xA}\le\mathbf{b}\\ \mathbf{xA}_1\le\mathbf{b}_1}} \mathbf{xa} = \max_{\mathbf{z}\le\mathbf{0}} \min_{\mathbf{xA}\le\mathbf{b}} [\mathbf{xa} + (\mathbf{b}_1 - \mathbf{xA}_1)\mathbf{z}]. \tag{A.5}$$

PROOF. Note that the two sets of constraints $\mathbf{xA} \le \mathbf{b}$ and $\mathbf{xA}_1 \le \mathbf{b}_1$ can be combined as

$$\mathbf{x}(\mathbf{A}; \mathbf{A}_1) \le (\mathbf{b}; \mathbf{b}_1).$$

We have

$$\begin{aligned}
\max_{\substack{\mathbf{xA}\le\mathbf{b}\\ \mathbf{xA}_1\le\mathbf{b}_1}} \mathbf{xa} &= \min_{\substack{\mathbf{u}\ge\mathbf{0},\mathbf{y}\ge\mathbf{0}\\ \mathbf{Au}+\mathbf{A}_1\mathbf{y}=\mathbf{a}}} (\mathbf{bu}+\mathbf{b}_1\mathbf{y}) && \text{by Theorem A.7}\\
&= \min_{\mathbf{y}\ge\mathbf{0}} \min_{\substack{\mathbf{u}\ge\mathbf{0}\\ \mathbf{Au}+\mathbf{A}_1\mathbf{y}=\mathbf{a}}} (\mathbf{b}_1\mathbf{y}+\mathbf{bu})\\
&= \min_{\mathbf{y}\ge\mathbf{0}} \left[\mathbf{b}_1\mathbf{y} + \min_{\substack{\mathbf{u}\ge\mathbf{0}\\ \mathbf{Au}=\mathbf{a}-\mathbf{A}_1\mathbf{y}}} \mathbf{bu}\right]\\
&= \min_{\mathbf{y}\ge\mathbf{0}} \left[\mathbf{b}_1\mathbf{y} + \max_{\mathbf{xA}\le\mathbf{b}} \mathbf{x}(\mathbf{a}-\mathbf{A}_1\mathbf{y})\right] && \text{by Theorem A.7}\\
&= \min_{\mathbf{y}\ge\mathbf{0}} \max_{\mathbf{xA}\le\mathbf{b}} [\mathbf{xa}+(\mathbf{b}_1-\mathbf{xA}_1)\mathbf{y}].
\end{aligned}$$

The second formula can be derived from the Corollary to Theorem A.7 in the same way. □

As we see below, when the extra constraints are equalities, there are no restrictions on $\mathbf{y}$ or $\mathbf{z}$.

**Corollary 1** *In the notation of Theorem A.8, the following formulas hold:*

$$\max_{\substack{\mathbf{xA}\le\mathbf{b}\\ \mathbf{xA}_1=\mathbf{b}_1}} \mathbf{xa} = \min_{\mathbf{y}\in R^{l_1}} \max_{\mathbf{xA}\le\mathbf{b}} [\mathbf{xa}+(\mathbf{b}_1-\mathbf{xA}_1)\mathbf{y}] \tag{A.6}$$

$$\min_{\substack{\mathbf{xA}\le\mathbf{b}\\ \mathbf{xA}_1=\mathbf{b}_1}} \mathbf{xa} = \max_{\mathbf{y}\in R^{l_1}} \min_{\mathbf{xA}\le\mathbf{b}} [\mathbf{xa}+(\mathbf{b}_1-\mathbf{xA}_1)\mathbf{y}]. \tag{A.7}$$

PROOF. These results follow directly from Theorem A.8, if we note that the equation $\mathbf{xA}_1 = \mathbf{b}_1$ is equivalent to the combined set of inequalities: $\mathbf{xA}_1 \le \mathbf{b}_1$ and $-\mathbf{xA}_1 \le -\mathbf{b}_1$. Indeed, we have

$$\begin{aligned}
\max_{\substack{\mathbf{xA}\le\mathbf{b}\\ \mathbf{xA}_1=\mathbf{b}_1}} \mathbf{xa} &= \min_{\mathbf{u}\ge\mathbf{0}} \min_{\mathbf{v}\ge\mathbf{0}} \max_{\mathbf{xA}\le\mathbf{b}} [\mathbf{xa}+(\mathbf{b}_1-\mathbf{xA}_1)\mathbf{u}+(-\mathbf{b}_1+\mathbf{xA}_1)\mathbf{v}]\\
&= \min_{\mathbf{u}\ge\mathbf{0}} \min_{\mathbf{v}\ge\mathbf{0}} \max_{\mathbf{xA}\le\mathbf{b}} [\mathbf{xa}+(\mathbf{b}_1-\mathbf{xA}_1)(\mathbf{u}-\mathbf{v})]\\
&= \min_{\mathbf{y}\in R^{l_1}} \max_{\mathbf{xA}\le\mathbf{b}} [\mathbf{xa}+(\mathbf{b}_1-\mathbf{xA}_1)\mathbf{y}]
\end{aligned}$$

where we have written $\mathbf{y} = \mathbf{u} - \mathbf{v}$. There is no sign restriction on $\mathbf{y}$, since it is the difference between two nonnegative vectors.

The other formula can be proved similarly. □

We now recast Theorem A.8 and its corollary in a slightly different form that will be useful for the discussion in the next section.

**Theorem A.9** *Let $f$ denote a real-valued linear function on $\mathbf{R}^m$ and $P$ a nonempty polytope in $\mathbf{R}^m$. Consider an inequality of the form $g(\mathbf{x}) \leq c$, where $g$ is also a real-valued linear function on $\mathbf{R}^m$. The extreme values of $f$ in $P$, subject to the inequality constraint $g(\mathbf{x}) \leq c$, are given by*

$$\max_{\substack{\mathbf{x}\in P \\ g(\mathbf{x})\leq c}} f(\mathbf{x}) = \min_{\lambda\geq 0} \max_{\mathbf{x}\in P} [f(\mathbf{x}) + \lambda(c - g(\mathbf{x}))] \tag{A.8}$$

$$\min_{\substack{\mathbf{x}\in P \\ g(\mathbf{x})\leq c}} f(\mathbf{x}) = \max_{\mu\leq 0} \min_{\mathbf{x}\in P} [f(\mathbf{x}) + \mu(c - g(\mathbf{x}))]. \tag{A.9}$$

PROOF. Here, we have $l_1 = 1$. The above formulas follow from similar formulas in Theorem A.8, if we write $f(\mathbf{x})$ for $\mathbf{xa}$, "$\mathbf{x} \in P$" for "$\mathbf{xA} \leq \mathbf{b}$," $\lambda$ for $\mathbf{y}$, $\mu$ for $\mathbf{z}$, $c$ for $\mathbf{b}_1$, and $g(\mathbf{x})$ for $\mathbf{xA}_1$. □

**Corollary 1** *In Theorem A.9, the extreme values of $f$ in $P$ subject to the equality constraint $g(\mathbf{x}) = c$ are given by*

$$\max_{\substack{\mathbf{x}\in P \\ g(\mathbf{x})=c}} f(\mathbf{x}) = \min_{\lambda} \max_{\mathbf{x}\in P} [f(\mathbf{x}) + \lambda(c - g(\mathbf{x}))] \tag{A.10}$$

$$\min_{\substack{\mathbf{x}\in P \\ g(\mathbf{x})=c}} f(\mathbf{x}) = \max_{\lambda} \min_{\mathbf{x}\in P} [f(\mathbf{x}) + \lambda(c - g(\mathbf{x}))]. \tag{A.11}$$

PROOF. The proof follows from Corollary 1 to Theorem A.8. □

Since there is no sign restriction on $\lambda$, we may write $\lambda(g(\mathbf{x}) - c)$ instead of $\lambda(c - g(\mathbf{x}))$ in the corollary.

Suppose we have two constraints of the form $g_1(\mathbf{x}) \leq c_1$ and $g_2(\mathbf{x}) = c_2$. Then, Theorem A.9 and its corollary can be combined to yield the following expressions:

$$\max_{\substack{\mathbf{x}\in P \\ g_1(\mathbf{x})\leq c_1 \\ g_2(\mathbf{x})=c_2}} f(\mathbf{x}) = \min_{\lambda_1\geq 0} \min_{\lambda_2} \max_{\mathbf{x}\in P} [f(\mathbf{x}) + \lambda_1(c_1 - g_1(\mathbf{x})) + \lambda_2(c_2 - g_2(\mathbf{x}))]$$

$$\min_{\substack{\mathbf{x}\in P \\ g_1(\mathbf{x})\leq c_1 \\ g_2(\mathbf{x})=c_2}} f(\mathbf{x}) = \max_{\mu_1\leq 0} \max_{\lambda_2} \min_{\mathbf{x}\in P} [f(\mathbf{x}) + \mu_1(c_1 - g_1(\mathbf{x})) + \lambda_2(c_2 - g_2(\mathbf{x}))].$$

Similarly, we can write down formulas for extreme values of $f$ subject to any number of linear constraints that are equations or inequalities.

**EXERCISES A.4**

1. Let $f_1, f_2, \ldots, f_n$ denote real-valued linear functions on $\mathbf{R}^m$ and $P \subset \mathbf{R}^m$ a nonempty polytope. Write down a pair of expressions giving the extreme values of $f_1$ in $P$ under a set of additional constraints of the form:
   (a) $f_2(\mathbf{x}) \leq c_2, f_3(\mathbf{x}) \leq c_3, \ldots, f_n(\mathbf{x}) \leq c_n$;
   (b) $f_2(\mathbf{x}) = c_2, f_3(\mathbf{x}) = c_3, \ldots, f_n(\mathbf{x}) = c_n$;
   (c) $f_2(\mathbf{x}) \geq c_2, f_3(\mathbf{x}) \geq c_3, \ldots, f_n(\mathbf{x}) \geq c_n$.

## A.5 Real Solutions to a System of Equations

Consider a system of $n$ scalar equations:

$$\left.\begin{aligned} f_1(\mathbf{x}) &= c_1 \\ f_2(\mathbf{x}) &= c_2 \\ &\vdots \\ f_n(\mathbf{x}) &= c_n, \end{aligned}\right\}$$

where $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})$ are real-valued linear functions on $\mathbf{R}^m$. In this section, we present a necessary and sufficient condition for the existence of a real solution to this system in a polytope. This result generalizes Theorem A.6 that deals with a single equation. We first establish the condition for a system of two equations, and then state the general result without proof. The proof in the general case is similar to the proof in the two-variable case, and is left to the reader.

**Theorem A.10** *Consider two real-valued linear functions $f_1$ and $f_2$ on $\mathbf{R}^m$, and a nonempty polytope $P \subset \mathbf{R}^m$. For a given pair of real numbers $c_1$ and $c_2$, the system of equations*

$$\left.\begin{aligned} f_1(\mathbf{x}) &= c_1 \\ f_2(\mathbf{x}) &= c_2 \end{aligned}\right\} \tag{A.12}$$

*has a solution $\mathbf{x} \in P$ iff*

$$\max_{\lambda} \min_{\mathbf{x} \in P} [f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2)] \leq c_1 \leq \min_{\lambda} \max_{\mathbf{x} \in P} [f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2)]. \tag{A.13}$$

PROOF. Define a polytope $Q \subset \mathbf{R}^m$ by

$$Q = \{\mathbf{x} \in P : f_2(\mathbf{x}) = c_2\}.$$

First, assume that the system (A.12) has a solution $\mathbf{x} \in P$. Then, $Q$ is nonempty and the equation $f_1(\mathbf{x}) = c_1$ has a solution $\mathbf{x} \in Q$. Theorem A.6 implies that

$$\min_{\mathbf{x}\in Q} f_1(\mathbf{x}) \le c_1 \le \max_{\mathbf{x}\in Q} f_1(\mathbf{x}). \tag{A.14}$$

But, we have

$$\begin{aligned}\min_{\mathbf{x}\in Q} f_1(\mathbf{x}) &\equiv \min_{\substack{\mathbf{x}\in P \\ f_2(\mathbf{x})=c_2}} f_1(\mathbf{x}) \\ &= \max_{\lambda} \min_{\mathbf{x}\in P} [f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2)] \qquad \text{by (A.11)}\end{aligned}$$

and also

$$\begin{aligned}\max_{\mathbf{x}\in Q} f_1(\mathbf{x}) &\equiv \max_{\substack{\mathbf{x}\in P \\ f_2(\mathbf{x})=c_2}} f_1(\mathbf{x}) \\ &= \min_{\lambda} \max_{\mathbf{x}\in P} [f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2)] \qquad \text{by (A.10).}\end{aligned}$$

Hence, Condition (A.14) is the same as Condition (A.13) in the theorem.

Next, assume that (A.13) holds. It can be shown that the polytope $Q$ is nonempty (Exercise 2). As in the previous paragraph, we can derive (A.14) from (A.13), and then use Theorem A.6 to conclude that the equation $f_1(\mathbf{x}) = c_1$ has a solution $Q$. That is the same as saying that the system (A.12) has a solution in $P$. □

**Corollary 1** *The system of equations (A.12) has a solution $\mathbf{x} \in P$, iff the single equation*

$$f_1(\mathbf{x}) + \lambda f_2(\mathbf{x}) = c_1 + \lambda c_2 \tag{A.15}$$

*has a solution in $P$ for each real $\lambda$.*

PROOF. The *only if* part is trivial. If $\mathbf{y}$ is a solution to the system (A.12), then $\mathbf{y}$ is a solution to (A.15) for each $\lambda$. Indeed, we have $f_1(\mathbf{y}) = c_1$ and $f_2(\mathbf{y}) = c_2$, so that

$$f_1(\mathbf{y}) + \lambda f_2(\mathbf{y}) = c_1 + \lambda c_2$$

for each $\lambda$.

The interesting feature of the *if* part is that for each real $\lambda$, we are assuming the existence of a solution $\mathbf{y}_\lambda$ to (A.15). From that the existence of a solution $\mathbf{y}$ to (A.12) is to follow, where $\mathbf{y}$ is independent of $\lambda$. Equation (A.15) can be written as

$$f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2) = c_1.$$

By Theorem A.6, it has a solution in $P$ iff

$$\min_{\mathbf{x}\in P}[f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2)] \le c_1 \le \max_{\mathbf{x}\in P}[f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2)].$$

Since this must hold for each $\lambda$ and $c_1$ is independent of $\lambda$, it follows that

$$\max_{\lambda}\min_{\mathbf{x}\in P}[f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2)] \le c_1 \le \min_{\lambda}\max_{\mathbf{x}\in P}[f_1(\mathbf{x}) + \lambda(f_2(\mathbf{x}) - c_2)].$$

By Theorem A.10, the system (A.12) then has a solution in $P$. □

Next, we state the extensions of Theorem A.10 and its corollary to the general case of $n$ equations. The proofs are quite similar and are omitted. The main difference is that now we need $n-1$ real numbers $\lambda_2, \lambda_3, \ldots, \lambda_{n-1}$ for the $n-1$ equations in the system in addition to the first. We write $\boldsymbol{\lambda} = (\lambda_2, \lambda_3, \ldots, \lambda_n)$. Note also that the ordering of the equations is irrelevant; in particular, any of the $n$ equations can be labeled as $f_1(\mathbf{x}) = c_1$.

**Theorem A.11** *Let $f_1, f_2, \ldots, f_n$ denote a sequence of real-valued linear functions on $\mathbf{R}^m$, and $P$ a nonempty polytope in $\mathbf{R}^m$. For a given set of real numbers $c_1, c_2, \ldots, c_n$, the system of equations*

$$f_k(\mathbf{x}) = c_k \qquad (1 \le k \le n) \tag{A.16}$$

*has a solution $\mathbf{x} \in P$ iff*

$$\max_{\boldsymbol{\lambda}\in\mathbf{R}^{n-1}}\min_{\mathbf{x}\in P}\left[f_1(\mathbf{x}) + \sum_{k=2}^{n}\lambda_k(f_k(\mathbf{x}) - c_k)\right] \le c_1 \le \min_{\boldsymbol{\lambda}\in\mathbf{R}^{n-1}}\max_{\mathbf{x}\in P}\left[f_1(\mathbf{x}) + \sum_{k=2}^{n}\lambda_k(f_k(\mathbf{x}) - c_k)\right]. \tag{A.17}$$

**Corollary 1** *The system of linear equations (A.16) has a real solution in a nonempty polytope $P$, iff the single equation*

$$f_1(\mathbf{x}) + \sum_{k=2}^{n} \lambda_k f_k(\mathbf{x}) = c_1 + \sum_{k=2}^{n} \lambda_k c_k \tag{A.18}$$

*has a real solution in $P$ for each $(\lambda_2, \lambda_3, \ldots, \lambda_n) \in \mathbf{R}^{n-1}$.*

Using Theorem A.11, we can write a simple algorithm that decides if a given system of linear equations has a solution in a given polytope, by first checking a single equation, then a pair of equations, etc. In the worst case, we end up testing $n$ systems of equations. But, in practice, we can get an approximate test by quitting after a small number of steps. That may be desirable since the complexity of handling a system of $n$ equations increases rapidly with the size of $n$.

**Algorithm A.1** Given a set of real-valued linear functions $f_1, f_2, \ldots, f_n$ on $\mathbf{R}^m$, a nonempty polytope $P \subset \mathbf{R}^m$, and a set of real numbers $c_1, c_2, \ldots, c_n$, this algorithm decides if the system of equations

$$f_k(\mathbf{x}) = c_k \qquad (1 \le k \le n) \tag{A.19}$$

has a solution in $P$.

For $1 \le k \le n$, let Cond($k$) denote the condition

$$\max_{\boldsymbol{\lambda} \in \mathbf{R}^{k-1}} \min_{\mathbf{x} \in P} \left[ f_1(\mathbf{x}) + \sum_{i=2}^{k} \lambda_i (f_i(\mathbf{x}) - c_i) \right] \le c_1$$

$$\le \min_{\boldsymbol{\lambda} \in \mathbf{R}^{k-1}} \max_{\mathbf{x} \in P} \left[ f_1(\mathbf{x}) + \sum_{i=2}^{k} \lambda_i (f_i(\mathbf{x}) - c_i) \right].$$

When $k = 1$ the condition becomes simply

$$\min_{\mathbf{x} \in P} f_1(\mathbf{x}) \le c_1 \le \max_{\mathbf{x} \in P} f_1(\mathbf{x}).$$

**do** $k = 1, n, 1$
  **if** Cond($k$) is false
  **then**
    **return** (the system (A.19) has no solution in $P$)
**enddo**
**return** (the system (A.19) has a solution in $P$)

PROOF. The algorithm is self-explanatory. We first test if $f_1(\mathbf{x}) = c_1$ has a solution in $P$. If not, then the system cannot have a solution in $P$. If yes, then we test if the subsystem $f_1(\mathbf{x}) = c_1, f_2(\mathbf{x}) = c_2$ has a solution in $P$, and so on. □

There could be variations of this algorithm. For example, we may first test if the single equations $f_k(\mathbf{x}) = c_k$ have individual solutions.

**EXERCISES A.5**

1. Consider a linear function $F : \mathbf{R}^3 \to \mathbf{R}^2$. It defines two real-valued functions $f_1$ and $f_2$ on $\mathbf{R}^3$ by the equation

$$F(x_1, x_2, x_3) = (f_1(x_1, x_2, x_3), f_2(x_1, x_2, x_3)) \qquad ((x_1, x_2, x_3) \in \mathbf{R}^3).$$

   Show that $f_1$ and $f_2$ are both linear. (This result can be easily generalized to linear functions $F$ from any space $\mathbf{R}^m$ to any space $\mathbf{R}^n$.

2. In Theorem A.10, show that if the equation $f_2(\mathbf{x}) = c_2$ has no solution in $P$, that is, if $Q$ is empty, then Condition A.13 does not hold.

## A.6 Integer Solutions to Linear Equations

In this appendix, we have dealt with real solutions to linear equations with real coefficients in polytopes. Consider now a system of linear equations with integer coefficients, and a given polytope. Under what conditions, does the existence of a real solution to that system in that polytope imply the existence of an integer solution in the same polytope? In this context, the concept of a totally unimodular matrix becomes useful. An integer matrix $\mathbf{A}$ (of any size) is *totally unimodular* if the determinant of each square submatrix is 0, 1, or $-1$ (see Chapter 19 of [Schr 87]). Thus, in particular, each element of a totally unimodular matrix is 0, 1, or $-1$. A square totally unimodular matrix is necessarily unimodular.

We state a basic fact about extreme points of a polytope given by a totally unimodular matrix, and then prove a result on integer solutions to a system of equations.

**Lemma A.12** *The extreme points of a polytope* $P = \{\mathbf{x} : \mathbf{xA} \leq \mathbf{b}\}$ *are integer vectors, if* $\mathbf{A}$ *is a totally unimodular matrix and* $\mathbf{b}$ *is an integer vector.*

**Theorem A.13** *Consider a system*

$$\mathbf{xB} = \mathbf{c}$$

*of $n$ linear equations in $m$ variables, where $\mathbf{B}$ is an $m \times n$ integer matrix and $\mathbf{c} \in \mathbf{R}^n$ is an integer vector. Suppose it has a real solution in a polytope $P = \{\mathbf{x} \in \mathbf{R}^m : \mathbf{xA} \leq \mathbf{b}\}$, where $\mathbf{A}$ is a totally unimodular matrix and $\mathbf{b}$ is an integer vector. If the concatenated matrix $(\mathbf{A};\mathbf{B})$ is totally unimodular, then this system has an integer solution in $P$.*

PROOF. The set of solutions to $\mathbf{xB} = \mathbf{c}$ in $P$ is the polytope $Q$ defined by the following system of inequalities:

$$\left.\begin{aligned} \mathbf{xA} &\leq \mathbf{b} \\ \mathbf{xB} &\leq \mathbf{c} \\ \mathbf{x}(-\mathbf{B}) &\leq -\mathbf{c}. \end{aligned}\right\} \tag{A.20}$$

By hypothesis, this polytope is nonempty. Since the matrix $(\mathbf{A};\mathbf{B})$ is totally unimodular, it can be shown (Exercise 1.) that so is the matrix $(\mathbf{A};\mathbf{B};-\mathbf{B})$. Then, by Lemma A.12, each extreme point of $Q$ is an integer point. This means the system of equations $\mathbf{xB} = \mathbf{c}$ has an integer solution in $P$. □

The condition of this theorem is sufficient, but not necessary for the existence of an integer solution. Not *each* extreme point of $Q$ needs to be an integer point; it is enough to have *one* extreme point that is an integer point. In fact, it suffices to have just one integer point somewhere in $Q$; it does not even have to be an extreme point.

The conditions under which the existence of a real solution implies the existence of an integer solution are important in dependence analysis, since they describe situations where an approximate dependence test would make accurate predictions. For example, we would know when the passing of the bounds test definitely implies the existence of dependence. We will not study this problem in detail; see [Bane 76], [Li 89], and [PSKK 91] for results in this area.

**EXERCISES A.6**

1. Let $\mathbf{A}$ and $\mathbf{B}$ denote two matrices with the same number of rows. Show that if $(\mathbf{A};\mathbf{B})$ is a totally unimodular matrix, then so is the matrix $(\mathbf{A};\mathbf{B};-\mathbf{B})$.

2. Explain how Theorem A.13 may be used in dependence problems to show that the existence of a real solution to the dependence equations satisfying all conditions implies the existence of an integer solution with the same property. Describe the kinds of problems that can be covered this way; give examples.
3. Describe some dependence problems where Theorem A.13 is not applicable, but the same conclusion (real solution implies integer solution) can be derived by finding one integral extreme point. (Extreme points can be found by Theorem A.2.)

# Bibliography


[AlKe 84] John Randal Allen & Ken Kennedy. Automatic Loop Interchange. *Proceedings of the SIGPLAN '84 Symposium on Compiler Construction*, Montreal, Canada, June 17-22, 1984. Available as *SIGPLAN Notices* **19** (1984), 233-246.

[AlKe 87] John Randal Allen & Ken Kennedy. Automatic Translation of FORTRAN Programs to Vector Form. *ACM Transactions on Programming Languages and Systems* **9** (1987), 491-542.

[Alle 83] John Randal Allen. Dependence Analysis for Subscripted Variables and its Application to Program Transformations. PhD Thesis, Department of Mathematical Sciences, Rice University, Houston, Texas, April 1983. Available as *Document 83-14916* from University Microfilms, Ann Arbor, Michigan.

[AsSh 80] Bengt Aspvall & Yossi Shiloach. A Fast Algorithm for Solving Systems of Linear Equations with Two Variables per Equation. *Linear Algebra and its Applications* **34** (1980), 117-124.

[Bane 76] Utpal Banerjee. Data Dependence in Ordinary Programs. MS Thesis, *Report 76-837*, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, November 1976.

[Bane 79] Utpal Banerjee. Speedup of Ordinary Programs. PhD Thesis, *Report 79-989*, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, October 1979. Available as *Document 80-08967* from University Microfilms, Ann Arbor, Michigan.

[Bane 88a] Utpal Banerjee. *Dependence Analysis for Supercomputing.* Kluwer Academic Publishers, Norwell, Massachusetts, 1988.

[Bane 88b] Utpal Banerjee. An Introduction to a Formal Theory of Dependence Analysis. *The Journal of Supercomputing* **2** (1988), 133-149.

[Bane 93] Utpal Banerjee. *Loop Transformations for Restructuring Compilers: The Foundations.* Kluwer Academic Publishers, Norwell, Massachusetts, 1993.

[Bane 94] Utpal Banerjee. *Loop Transformations for Restructuring Compilers: Loop Parallelization.* Kluwer Academic Publishers, Norwell, Massachusetts, 1994.

[BaSh 96] E. Bach & J. Shallit. *Algorithmic Number Theory, Volume I: Efficient Algorithms.* MIT Press, Cambridge, Massachusetts, 1996.

[BCKT 79] Utpal Banerjee, Shyh-Ching Chen, David J. Kuck & Ross A. Towle. Time and Parallel Processor Bounds for Fortran-Like Loops. *IEEE Transactions on Computers* **C-28** (1979), 660-670.

[BENP 93] Utpal Banerjee, Rudolf Eigenmann, Alexandru Nicolau & David A. Padua. Automatic Program Parallelization. *IEEE Proceedings* **81** (1993), 211-243.

[BoTr 76] I. Borosh & L. B. Treybig. Bounds on Positive Integral Solutions of Linear Diophantine Equations. *Proceedings of the American Mathematical Society* **55** (1976), 299-304.

[BrDu 71] J. L. Brown, Jr. & R. L. Duncan. The Least Remainder Algorithm. *Fibonacci Quarterly* **9** (1971), 347-350,401.

[BuCy 86] Michael Burke & Ron Cytron. Interprocedural Dependence Analysis and Parallelization. *Proceedings of the ACM SIGPLAN '86 Symposium on Compiler Construction,* Palo Alto, California, June 25-27, 1986. Available as *SIGPLAN Notices* **21** (1986), 162-175.

[Coha 73] William L. Cohagan. Vector Optimization for the ASC. *Proceedings of the 7th Annual Princeton Conference on Information Sciences and Systems,* March 22-23, 1973. Princeton University Press, Princeton, New Jersey, 1973, 169-174.

[DaEa 73] G. B. Dantzig & B. C. Eaves. Fourier-Motzkin Elimination and its Dual. *Journal of Combinatorial Theory (A)* **14** (1973), 288-297.

[Dieu 69] Jean A. Dieudonné. *Foundations of Modern Analysis.* Academic Press, New York, New York, 1969.

[Duff 74] Richard J. Duffin. On Fourier's Analysis of Linear Inequality Systems. *Mathematical Programming Study* **1** (1974), 71-95.

[Feau 88] Paul Feautrier. Parametric Integer Programming. *RAIRO Recherche Operationnelle* **22** (1988), 243-268.

[Fior 72] J. Ch. Fiorot. Generation of all Integer Points for Given Sets of Linear Inequalities. *Mathematical Programming* 3 (1972), 276-295.

[GaJo 79] M. R. Garey & D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco, California, 1979.

[GoKT 91] Gina Goff, Ken Kennedy & Chau-Wen Tseng. Practical Dependence Testing. *Proceedings of the ACM SIGPLAN '91 Conference on Programming Language Design and Implementation*, Toronto, Canada, June 26-28, 1991. Available as *SIGPLAN Notices* **26** (1991), 15-29.

[GoZa 52] A. W. Goodman & W. M. Zaring. Euclid's Algorithm and the Least Remainder Algorithm. *American Mathematical Monthly* **59** (1952), 156-159.

[Hagh 95] Mohammad R. Haghighat. *Symbolic Analysis for Parallelizing Compilers.* Kluwer Academic Publishers, Norwell, Massachusetts, 1995.

[Hagh 96] Mohammad R. Haghighat. A Note on the Omega Test. *To be published.*

[HoNa 94] Dorit S. Hochbaum & Joseph (Seffi) Naor. Simple and Fast Algorithms for Linear and Integer Programs with Two Variables per Inequality. *SIAM Journal on Computing* **23** (1994), 1179-1192.

[IrTr 89] François Irigoin & Rémi Triolet. Dependence Approximation and Global Parallel Code Generation for Nested Loops. *Parallel and Distributed Algorithms*, (eds. M. Cosnard et al.), Elsivier (North-Holland), New York, New York, 1989, 297-308.

[Kenn 80] Ken Kennedy. Automatic Translation of FORTRAN Programs to Vector Form. *Rice Technical Report 476-029-4*, Department of Mathematical Sciences, Rice University, Houston, Texas, October 1980.

[Knut 73] Donald E. Knuth. *The Art of Computer Programming, Volume 1/ Fundamental Algorithms.* Addison-Wesley, Reading, Massachusetts, Second Edition, 1973.

[Knut 81] Donald E. Knuth. *The Art of Computer Programming, Volume 2/ Seminumerical Algorithms.* Addison-Wesley, Reading, Massachusetts, Second Edition, 1981.

[KoKP 91] Xiangyun Kong, David Klappholz & Kleanthis Psarris. The I Test: An Improved Dependence Test for Automatic Parallelization and Vectorization. *IEEE Transactions on Parallel and Distributed Systems* **2** (1991), 342–349.

[Kuhn 80] Robert Henry Kuhn. Optimization and Interconnection Complexity for: Parallel Processors, Single-Stage Networks, and Decision Trees. PhD Thesis, *Report 80–1009*, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, February 1980. Available as *Document 80–26541* from University Microfilms, Ann Arbor, Michigan.

[Li 89] Zhiyuan Li. Intraprocedural and Interprocedural Data Dependence Analysis for Parallel Computing, PhD Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, August 1989. Available as *Report 910*, Center for Supercomputing Research and Development.

[LiYe 89] Zhiyuan Li & Pen-Chung Yew. Some Results on Exact Data Dependence Analysis. *Proceedings of the Second Workshop on Languages and Compilers for Parallel Computing*, Urbana, Illinois, August 1989. Available as *Languages and Compilers for Parallel Computing* (eds. D. Gelernter, A. Nicolau & D. Padua), The MIT Press, Cambridge, Massachusetts 1990, 374–401.

[LiYZ 90] Zhiyuan Li, Pen-Chung Yew & Chuan-Qi Zhu. An Efficient Data Dependence Analysis for Parallelizing Compilers. *IEEE Transactions on Parallel and Distributed Systems* **1** (1990), 26–34.

[MaHL 91] Dror E. Maydan, John L. Hennessy & Monica S. Lam. Efficient and Exact Data Dependence Analysis. *Proceedings of the ACM SIGPLAN '91 Conference on Programming Language Design and Implementation*, Toronto, Canada, June 26–28, 1991. Available as *SIGPLAN Notices* **26** (1991), 1–14.

[Malm 80] Donald G. Malm. *A Computer Laboratory Manual for Number Theory*. COMPress, Inc., Wentworth, New Hampshire, 1980.

[Moor 92] T. E. Moore. On the Least Absolute Remainder Euclidean Algorithm. *Fibonacci Quarterly* **30** (1992), 161–165.

[PsKK 91] Kleanthis Psarris, David Klappholz & Xiangyun Kong. On the Accuracy of the Banerjee Test. *Journal of Parallel and Distributed Computing* **12** (1991), 152–158.

[PsKK 93] Kleanthis Psarris, David Klappholz & Xiangyun Kong. The Direction Vector *I* Test. *IEEE Transactions on Parallel and Distributed Systems* **4** (1993), 1280–1290.

[Pugh 92a] William Pugh. A Practical Algorithm for Exact Array Dependence Analysis. *Communications of the ACM* **35** (1992), 102–114.

[Pugh 92b] William Pugh. Definitions of Dependence Distance. *ACM Letters on Programming Languages and Systems* **1** (1992), 261–265.

[Schr 87] Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, New York, New York, 1987.

[Shos 81] Robert Shostak. Deciding Linear Inequalities by Computing Loop Residues. *Journal of the Association for Computing Machinery* **28** (1981), 769–779.

[Towl 76] Ross Albert Towle. Control and Data Dependence for Program Transformations. PhD Thesis, *Report 76–788*, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, March 1976.

[TrIF 86] Rémi Triolet, François Irigoin & Paul Feautrier. Direct Parallelization of Call Statements. *Proceedings of the ACM SIGPLAN '86 Symposium on Compiler Construction*, Palo Alto, California, June 1986, 176–185.

[Wall 88] David R. Wallace. Dependence of Multi-Dimensional Array References. *Proceedings of the 1988 International Conference on Supercomputing*, St. Malo, France, July 1988, 418–428. The ACM Press, New York, New York.

[Will 83] H. P. Williams. A Characterization of all Feasible Solutions to an Integer Program. *Discrete Applied Mathematics* **5** (1983), 147–155.

[Will 86] H. P. Williams. Fourier's Method of Linear Programming and its Dual. *The American Mathematical Monthly* **93** (1986), 681–695.

[Wolf 82] Michael J. Wolfe. Optimizing Compilers for Supercomputers. PhD Thesis, *Report 82–1105*, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, October 1982. Available as *Report 329*, Center for Supercomputing Research and Development, and also as *Document 83-03027* from University Microfilms, Ann Arbor, Michigan.

[WoBa 87] Michael Wolfe & Utpal Banerjee. Data Dependence and its Application to Parallel Processing. *International Journal of Parallel Programming* **16** (1987), 137–178.

[Wolf 94] Michael Wolfe. The Definition of Dependence Distance. *ACM Transactions on Programming Languages and Systems* **16** (1994), 1114–1116.

[WoTs 92] Michael Wolfe & Chau-Wen Tseng. The Power Test for Data Dependence. *IEEE Transactions on Parallel and Distributed Systems* **3** (1992), 591–601.

# Index