Loop parallelization using compiler analysis

Which of these loops is parallel?

- Examples

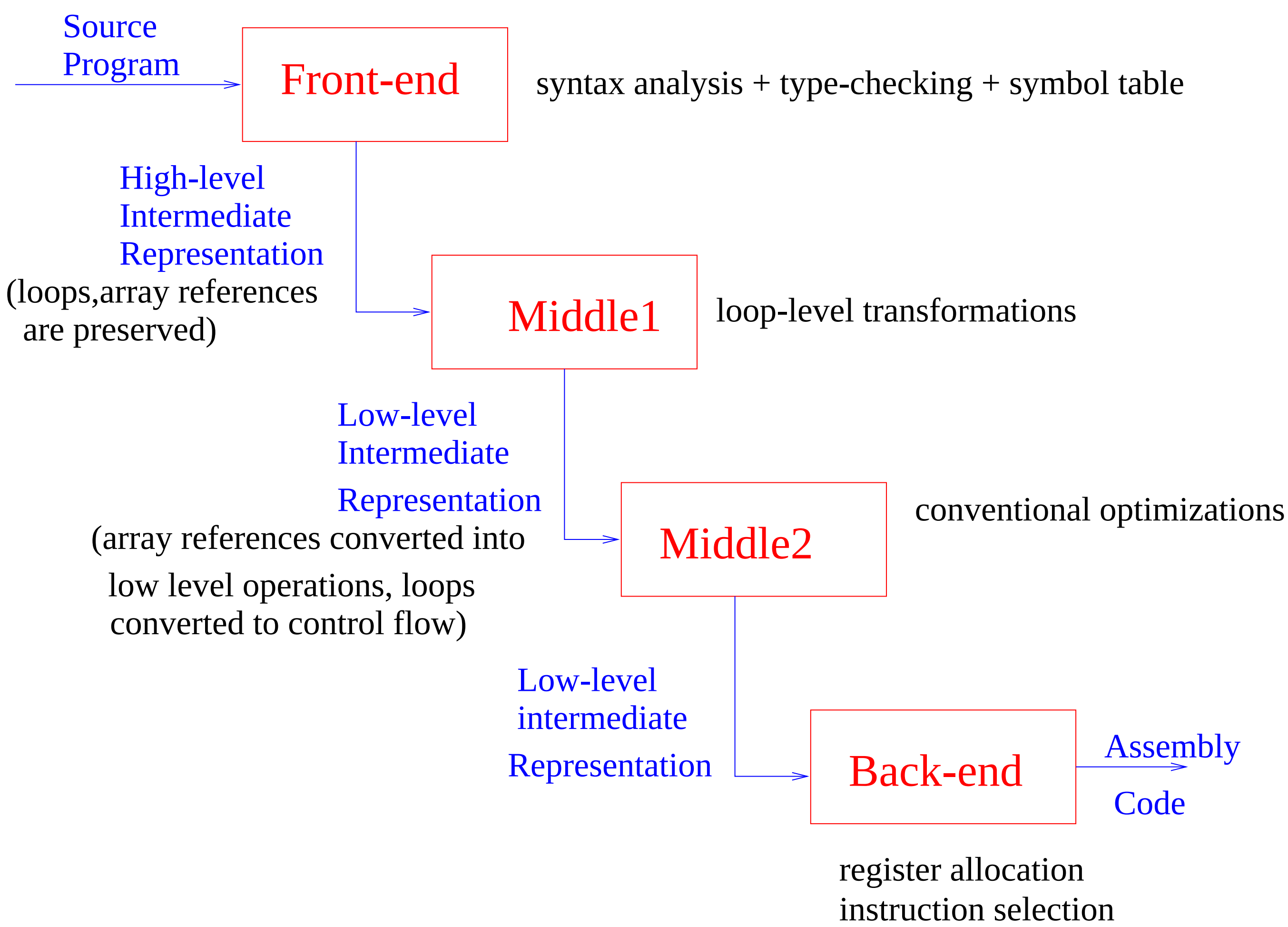
```
FOR I = 0 to 5
   A[I+1] = A[I] + 1

FOR I = 0 to 5
   A[I] = A[I+6] + 1

For I = 0 to 5
   A[2*I] = A[2*I + 1] + 1
```

How can we determine this automatically using compiler analysis?

Organization of a Modern Compiler



Key concepts:

Perfectly-nested loop: Loop nest in which all assignment statements occur in body of innermost loop.

```
for J = 1, N
  for I = 1, N
    Y(I) = Y(I) + A(I,J)*X(J)
```

Imperfectly-nested loop: Loop nest in which some assignment statements occur within some but not all loops of loop nest

```
  for k = 1, N
    a(k,k) = sqrt (a(k,k))
    for i = k+1, N
      a(i,k) = a(i,k) / a(k,k)
    for i = k+1, N
      for j = k+1, i
        a(i,j) -= a(i,k) * a(j,k)
```

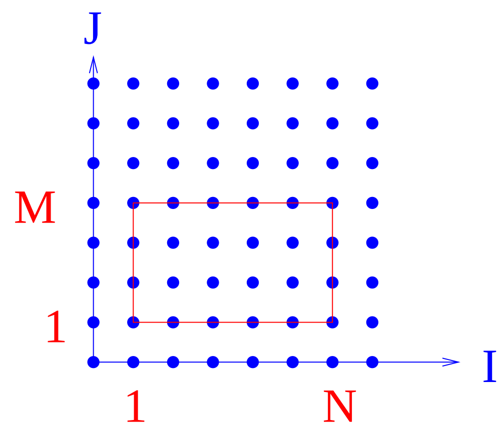
Our focus for now: perfectly-nested loops

Iteration Space of a Perfectly-nested Loop

Each iteration of a loop nest with n loops can be viewed as an integer point in an n-dimensional space.

Iteration space of loop: all points in n-dimensional space corresponding to loop iterations

```
DO  I = 1, N
  DO  J  = 1,M
     S
```



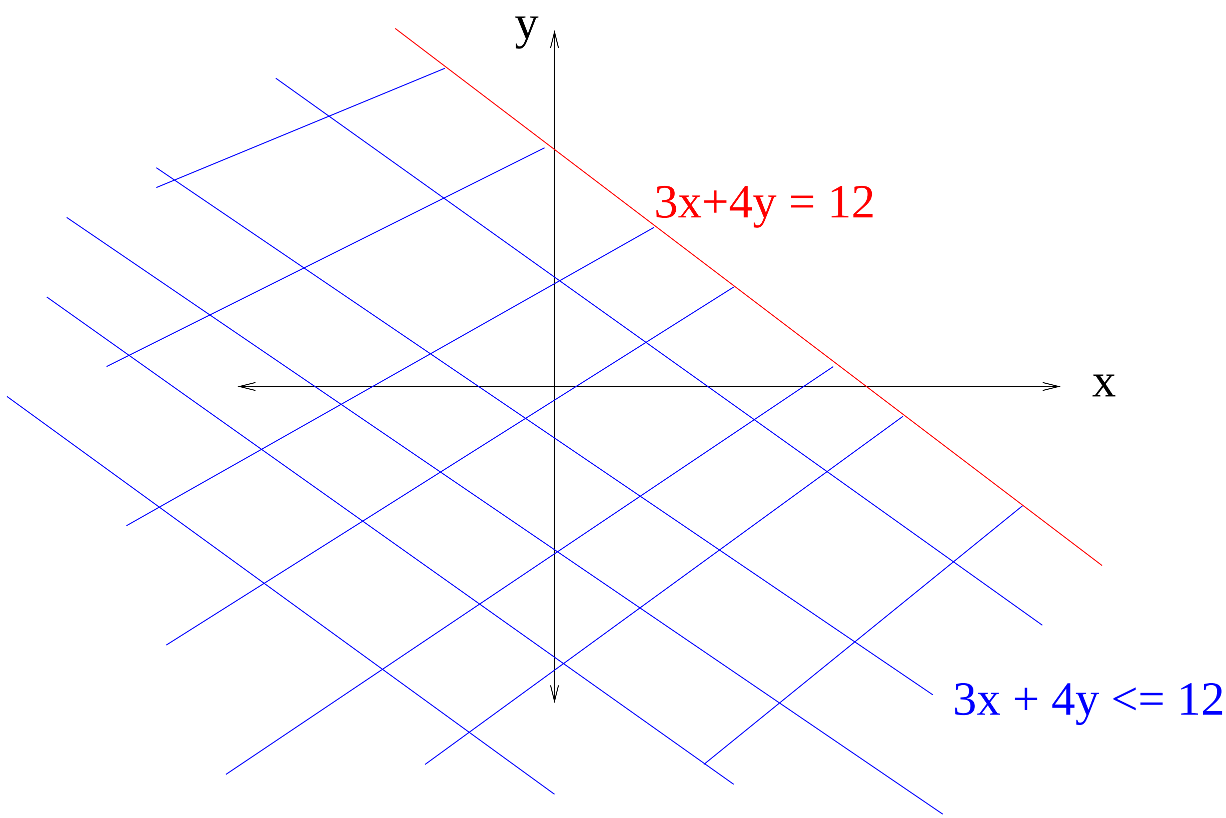
Execution order = lexicographic order on iteration space:

$(1,1) \preceq (1,2) \preceq ... \preceq (1,M) \preceq (2,1) \preceq (2,2) ... \preceq (N,M)$

Intuition about systems of linear inequalities:

Equality: line (2D), plane (3D), hyperplane (> 3D)

Inequality: half-plane (2D), half-space(>2D)

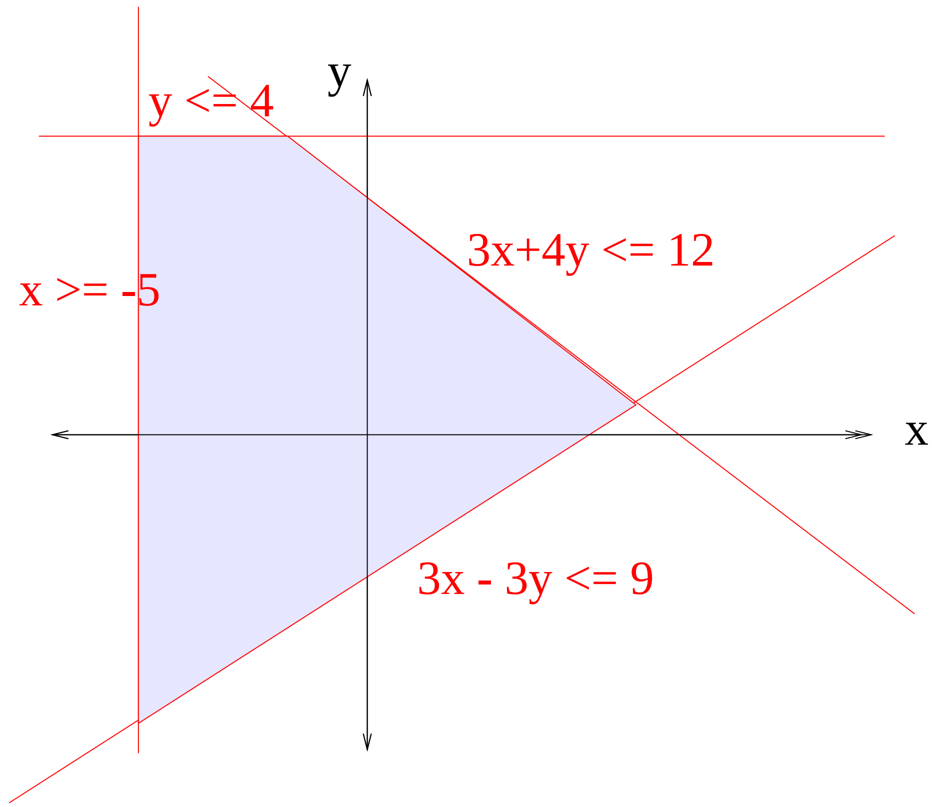


Region described by inequality is convex

(if two points are in region, all points in between them are in region)

Intuition about systems of linear inequalities:

Conjunction of inequalties = intersection of half-spaces
=> some convex region



Region described by inequalities is a convex polyhedron
(if two points are in region, all points in between them are in region)

Let us formulate correctness of loop permutation as ILP problem.

Intuition: If all iterations of a loop nest are independent, then permutation is certainly legal.

This is stronger than we need, but it is a good starting point.

What does independent mean?

Let us look at dependences.

Dependences in loops

```
FOR 10 I = 1, N
  X(f(I)) = ...
10          = ...X(g(I))..
```

- Conditions for flow dependence from iteration I_w to I_r:
  - $1 \leq I_w \leq I_r \leq N$ *(write before read)*
  - $f(I_w) = g(I_r)$ *(same array location)*
- Conditions for anti-dependence from iteration I_g to I_o:
  - $1 \leq I_g < I_o \leq N$ *(read before write)*
  - $f(I_o) = g(I_g)$ *(same array location)*
- Conditions for output dependence from iteration I_{w1} to I_{w2}:
  - $1 \leq I_{w1} < I_{w2} \leq N$ *(write in program order)*
  - $f(I_{w1}) = f(I_{w2})$ *(same array location)*

Dependences in nested loops

```
      FOR 10 I = 1, 100
       FOR 10 J = 1, 200
        X(f(I,J),g(I,J)) = ...
10              = ...X(h(I,J),k(I,J))..
```

Conditions for flow dependence from iteration (I_w, J_w) to (I_r, J_r):
Recall: $\preceq$ is the lexicographic order on iterations of nested loops.

$$
\begin{aligned}
1 &\leq I_w \leq 100 \\
1 &\leq J_w \leq 200 \\
1 &\leq I_r \leq 100 \\
1 &\leq J_r \leq 200 \\
(I_1, J_1) &\preceq (I_2, J_2) \\
f(I_1, J_1) &= h(I_2, J_2) \\
g(I_1, J_1) &= k(I_2, J_2)
\end{aligned}
$$

Anti and output dependences can be defined analogously.

Array subscripts are affine functions of loop variables

=>

dependence testing can be formulated as a set of ILP problems

ILP Formulation

```
FOR I = 1, 100
  X(2I) = .... X(2I+1)...
```

Is there a flow dependence between different iterations?

$$
\begin{aligned}
1 &\leq Iw < Ir \leq 100 \\
2Iw &= 2Ir + 1
\end{aligned}
$$

which can be written as

$$
\begin{aligned}
1 &\leq Iw \\
Iw &\leq Ir - 1 \\
Ir &\leq 100 \\
2Iw &\leq 2Ir + 1 \\
2Ir + 1 &\leq 2Iw
\end{aligned}
$$

The system

$$
\begin{aligned}
1 &\leq Iw \\
Iw &\leq Ir - 1 \\
Ir &\leq 100 \\
2Iw &\leq 2Ir + 1 \\
2Ir + 1 &\leq 2Iw
\end{aligned}
$$

can be expressed in the form $Ax \leq b$ as follows

$$
\begin{pmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \\ 2 & -2 \\ -2 & 2 \end{pmatrix} \begin{bmatrix} Iw \\ Ir \end{bmatrix} \leq \begin{bmatrix} -1 \\ -1 \\ 100 \\ 1 \\ -1 \end{bmatrix}
$$

ILP Formulation for Nested Loops

```
FOR I = 1, 100
 FOR J = 1, 100
   X(I,J) = ..X(I-1,J+1)...
```

Is there a flow dependence between different iterations?

$$
\begin{aligned}
1 &\leq Iw \leq 100 \\
1 &\leq Ir \leq 100 \\
1 &\leq Jw \leq 100 \\
1 &\leq Jr \leq 100 \\
(Iw, Jw) &\prec (Ir, Jr)(lexicographic\ order) \\
Ir - 1 &= Iw \\
Jr + 1 &= Jw
\end{aligned}
$$

Convert lexicographic order $\prec$ into integer equalities/inequalities.

$(Iw, Jw) \prec (Ir, Jr)$ is equivalent to

$Iw < Ir$ OR $((Iw = Ir)\ AND\ (Jw < Jr))$

We end up with two systems of inequalities:

$$
\begin{array}{l}
1 \leq Iw \leq 100 \\
1 \leq Ir \leq 100 \\
1 \leq Jw \leq 100 \\
1 \leq Jr \leq 100 \\
Iw < Ir \\
Ir - 1 = Iw \\
Jr + 1 = Jw
\end{array}
\quad OR \quad
\begin{array}{l}
1 \leq Iw \leq 100 \\
1 \leq Ir \leq 100 \\
1 \leq Jw \leq 100 \\
1 \leq Jr \leq 100 \\
Iw = Ir \\
Jw < Jr \\
Ir - 1 = Iw \\
Jr + 1 = Jw
\end{array}
$$

Dependence exists if either system has a solution.

What about affine loop bounds?

```
FOR I = 1, 100
  FOR J = 1, I
     X(I,J) = ..X(I-1,J+1)...
```

$$
\begin{aligned}
1 &\leq Iw \leq 100 \\
1 &\leq Ir \leq 100 \\
1 &\leq Jw \leq Iw \\
1 &\leq Jr \leq Ir \\
(Iw, Jw) &\prec (Ir, Jr)(lexicographicorder) \\
Ir - 1 &= Iw \\
Jr + 1 &= Jw
\end{aligned}
$$

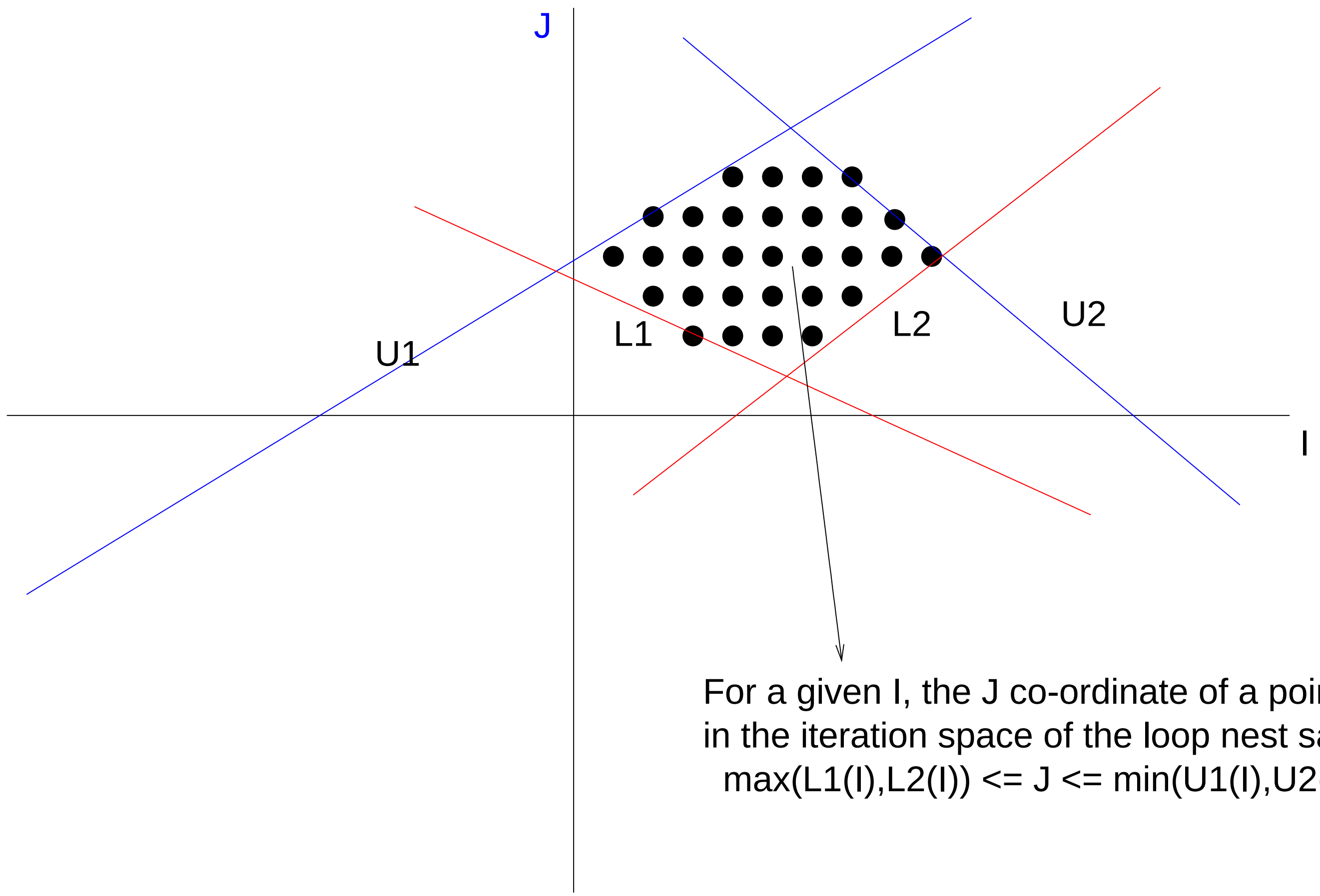
We can actually handle fairly complicated bounds involving min's and max's.

```
FOR I = 1, 100
  FOR J = max(F1(I),F2(I)) ,  min(G1(I),G2(I))
     X(I,J) = ..X(I-1,J+1)...
```

$$\begin{aligned} & \ldots. \\ F1(Ir) &\leq Jr \\ F2(Ir) &\leq Jr \\ Jr &\leq G1(Ir) \\ Jr &\leq G2(Ir) \\ & \ldots. \end{aligned}$$

Caveat: $F1, F2$ etc. must be affine functions.

Min’s and max’s in loop bounds mayseem weird, but actually they describe general polyhedral iteration spaces!



For a given I, the J co-ordinate of a point
in the iteration space of the loop nest satisfies
max(L1(I),L2(I)) <= J <= min(U1(I),U2(I))

More important case in practice: variables in upper/lower bounds

```
FOR I = 1, N
  FOR J = 1 , N-1
    ....
```

Solution: Treat N as though it was an unknown in system

$$
\begin{aligned}
1 &\leq Iw \leq N \\
1 &\leq Jw \leq N-1 \\
\ldots
\end{aligned}
$$

This is equivalent to seeing if there is a solution for any value of N.

Note: if we have more information about the range of N, we can easily add it as additional inequalities.

Summary

Problem of determining if a dependence exists between two iterations of a perfectly nested loop can be framed as ILP problem of the form

Is there an integer solution to system $Ax \leq b$?

How do we solve this decision problem?

Presentation sequence:

- one equation, several variables

  $2x + 3y = 5$

- several equations, several variables

  $2x + 3y + 5z = 5$
  $3x + 4y \quad\quad = 3$

Diophatine equations: use integer Gaussian elimination

- equations & inequalities

  $2x + 3y = 5$
  $x <= 5$
  $y <= -9$

Solve equalities first then use Fourier-Motzkin elimination

One equation, many variables:

Thm: The linear Diophatine equation a1 x1 + a2 x2 + ...+ an xn = c
has integer solutions iff gcd(a1,a2,...,an) divides c.

Examples:

(1) 2x = 3 No solutions

(2) 2x = 6 One solution: x = 3

(3) 2x + y = 3

GCD(2,1) = 1 which divides 3.

Solutions: x = t, y = (3 - 2t)

(4) 2x + 3y = 3

GCD(2,3) = 1 which divides 3.

Let z = x + floor(3/2) y = x + y

Rewrite equation as 2z + y = 3

Solutions: z = t
y = (3 - 2t)
=>
x = (3t - 3)
y = (3 - 2t)

Intuition: Think of underdetermined systems of eqns over reals.

Caution: Integer constraint => Diophantine system may have no solns

Thm: The linear Diophatine equation $a1\ x1 + a2\ x2 + \ldots + an\ xn = c$
has integer solutions iff gcd(a1,a2,...,an) divides c.

Proof: WLOG, assume that all coefficients a1,a2,...an are positive.
We prove only the IF case by induction, the proof in the other direction is trivial.
Induction is on min(smallest coefficient, number of variables).

Base case:

If (# of variables = 1) , then equation is a1 x1 = c which has integer solutions if a1 divides c.

If (smallest coefficient = 1), then gcd(a1,a2,...,an) = 1 which divides c.

Wlog, assume that a1 = 1, and observe that the equation has solutions of the form (c - a2 t2 - a3 t3 -....-an tn, t2, t3, ...tn).

Inductive case:

Suppose smallest coefficient is a1, and let $t = x1 + \text{floor}(a2/a1)\ x2 + \ldots + \text{floor}(an/a1)\ xn$

In terms of this variable, the equation can be rewritten as

$(a1)\ t + (a2 \bmod a1)\ x2 + \ldots + (an \bmod a1)\ xn = c$ (1)

where we assume that all terms with zero coefficient have been deleted.

Observe that (1) has integer solutions iff original equation does too.

Now gcd(a,b) = gcd(a mod b, b) => gcd(a1,a2,...,an) = gcd(a1, (a2 mod a1),..,(an mod a1))
=> gcd(a1, (a2 mod a1),..,(an mod a1)) divides c.

If a1 is the smallest co-efficient in (1), we are left with 1 variable base case.

Otherwise, the size of the smallest co-efficient has decreased, so we have made progress in the induction.

Summary:

Eqn: $a_1 x_1 + a_2 x_2 + \dots + a_n x_n = c$

- Does this have integer solutions?

= Does gcd(a1,a2,...,an) divide c ?

Systems of Diophatine Equations:

Key idea: use integer Gaussian elimination

Example:

$$2x + 3y + 4z = 5$$
$$x - y + 2z = 5$$

=>

$$\begin{bmatrix} 2 & 3 & 4 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

It is not easy to determine if this Diophatine system has solutions.

Easy special case: lower triangular matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 5 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

=>

$x = 5$
$y = 3$
z = arbitrary integer

Question: Can we convert general integer matrix into equivalent lower triangular system?

INTEGER GAUSSIAN ELIMINATION

Unimodular Column Operations:

(a) Interchange two columns

$$\begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \xrightarrow{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}} \begin{bmatrix} 3 & 2 \\ 7 & 6 \end{bmatrix}$$

Check

Let x,y satisfy first eqn.

Let x',y' satisfy second eqn.

x' = y , y' = x

(b) Negate a column

$$\begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \xrightarrow{\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}} \begin{bmatrix} 2 & -3 \\ 6 & -7 \end{bmatrix}$$

Check

x' = x, y' = - y

(c) Add an integer multiple of one column to another

$$\begin{bmatrix} 2 & 3 \\ 6 & 7 \end{bmatrix} \xrightarrow{\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}} \begin{bmatrix} 2 & 1 \\ 6 & 1 \end{bmatrix}$$

n = -1

Check

x = x' + n y'

y = y'

Example:

$$\begin{bmatrix} 2 & 3 & 4 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 3 & 4 \\ 1 & -1 & 2 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 3 & 0 \\ 1 & -1 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 1 & 0 \\ 1 & -2 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 5 & -2 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 \\ -2 & 5 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 5 & 0 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \Rightarrow \begin{array}{l} x' = 5 \\ y' = 3 \\ z' = t \end{array} \Rightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -1 & 3 & -2 \\ 1 & -2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 3 \\ t \end{bmatrix} = \begin{bmatrix} 4-2t \\ -1 \\ t \end{bmatrix}$$

Systems of Inequalities

Goals:

Given system of inequalities of the form $Ax \leq b$

- determine if system has an integer solution
- enumerate all integer solutions

Running example:

$$3x + 4y \geq 16 \tag{1}$$

$$4x + 7y \leq 56 \tag{2}$$

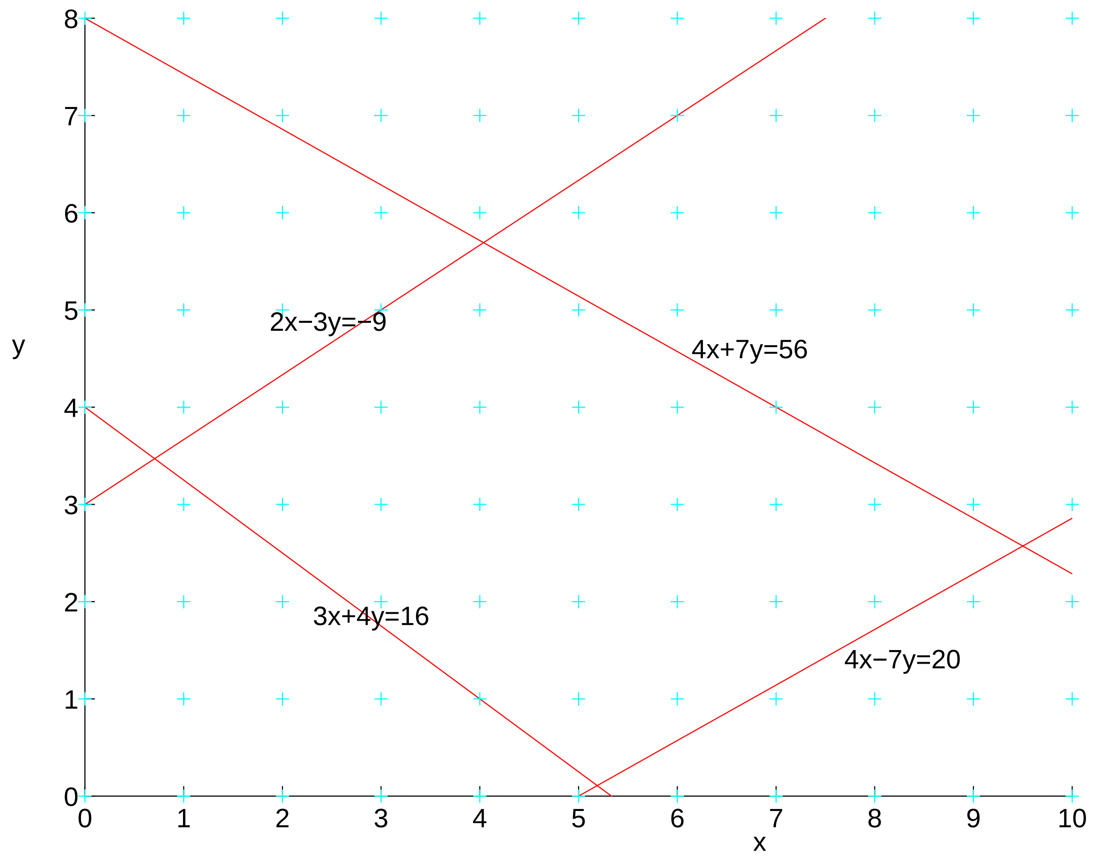
$$4x - 7y \leq 20 \tag{3}$$

$$2x - 3y \geq -9 \tag{4}$$

Upper bounds for x: (2) and (3)
Lower bounds for x: (1) and (4)

Upper bounds for y: (2) and (4)
Lower bounds for y: (1) and (3)

MATLAB graphs:



Code for enumerating integer points in polyhedron: (see graph)

Outer loop: Y, Inner loop: X

DO Y=$\lceil 4/37 \rceil$,$\lfloor 74/13 \rfloor$

  DO X=$\lceil max(16/3 - 4y/3, -9/2 + 3y/2) \rceil$,$\lfloor min(5 + 7y/4, 14 - 7y/4) \rfloor$

   .....

Outer loop: X, Inner loop: Y

```
DO X=1, 9
```

 DO Y=$\lceil max(4 - 3y/4, (4x - 20)/7) \rceil$,$\lfloor (min(8 - 4x/5, (2x + 9)/3) \rfloor$

   .....

How do we can determine loop bounds?

Fourier-Motzkin elimination: variable elimination technique for inequalities

$$3x + 4y \geq 16 \tag{5}$$

$$4x + 7y \leq 56 \tag{6}$$

$$4x - 7y \leq 20 \tag{7}$$

$$2x - 3y \geq -9 \tag{8}$$

Let us project out x.
First, express all inequalities as upper or lower bounds on x.

$$x \geq 16/3 - 4y/3 \tag{9}$$

$$x \leq 14 - 7y/4 \tag{10}$$

$$x \leq 5 + 7y/4 \tag{11}$$

$$x \geq -9/2 + 3y/2 \tag{12}$$

For any y, if there is an x that satisfies all inequalities, then every lower bound on x must be less than or equal to every upper bound on x.

Generate a new system of inequalities from each pair (upper,lower) bounds.

$$
\begin{aligned}
5 + 7y/4 &\geq 16/3 - 4y/3 (\text{Inequalities} 3, 1) \\
5 + 7y/4 &\geq -9/2 + 3y/2 (\text{Inequalities} 3, 4) \\
14 - 7y/4 &\geq 16/3 - 4y/3 (\text{Inequalities} 2, 1) \\
14 - 7y/4 &\geq -9/2 + 3y/2 (\text{Inequalities} 2, 4)
\end{aligned}
$$

Simplify:

$$
\begin{aligned}
y &\geq 4/37 \\
y &\geq -38 \\
y &\leq 104/5 \\
y &\leq 74/13
\end{aligned}
$$

=>

$$max(4/37, -38) \quad \leq \quad y \leq min(104/5, 74/13)$$

$$=>$$

$$4/37 \quad \leq y \quad \leq 74/13$$

This means there are rational solutions to original system of inequalities.

We can now express solutions in closed form as follows:

$$
\begin{aligned}
4/37 &\leq y \leq 4/37 \\
max(16/3 - 4y/3, -9/2 + 3y/2) &\leq x \leq min(5 + 7y/4, 14 - 7y/4)
\end{aligned}
$$

Fourier-Motzkin elimination: iterative algorithm

Iterative step:

- obtain reduced system by projecting out a variable
- if reduced system has a rational solution, so does the original

Termination: no variables left

Projection along variable x: Divide inequalities into three categories

$$a_1 * y + a_2 * z + ... \leq c_1 (no\ x)$$

$$b_1 * x \leq c_2 + b_2 * y + b_3 * z + ...(upper\ bound)$$

$$d_1 * x \geq c_3 + d_2 * y + d_3 * z + ...(lower\ bound)$$

New system of inequalities:

- All inequalities that do not involve x
- Each pair (lower,upper) bounds gives rise to one inequality:

$$b_1[c_3 + d_2 * y + d_3 * z + ...] \leq d_1[c_2 + b_2 * y + b_3 * z + ...]$$

Theorem: If $(y_1, z_1, ...)$ satisfies the reduced system, then $(x_1, y_1, z_1 ...)$ satisfies the original system, where x_1 is a rational number between

$min(1/b_1(c_2 + b_2 y_1 + b_3 z_1 + ...), ...)$ (over all upper bounds)

and

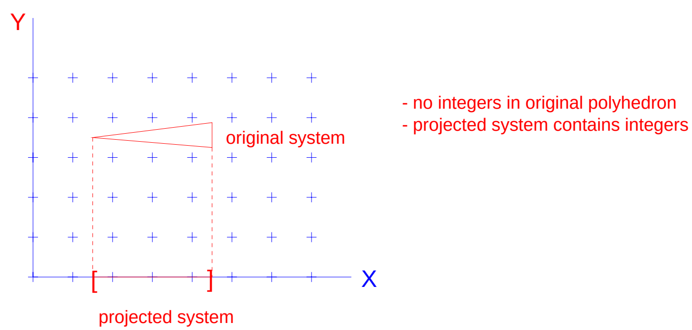
$max(1/d_1(c_3 + d_2 y_1 + d_3 z_1 + ...), ...)$ (over all lower bounds)

Proof: trivial

What can we conclude about integer solutions?

Corollary: If reduced system has no integer solutions, neither does the original system.

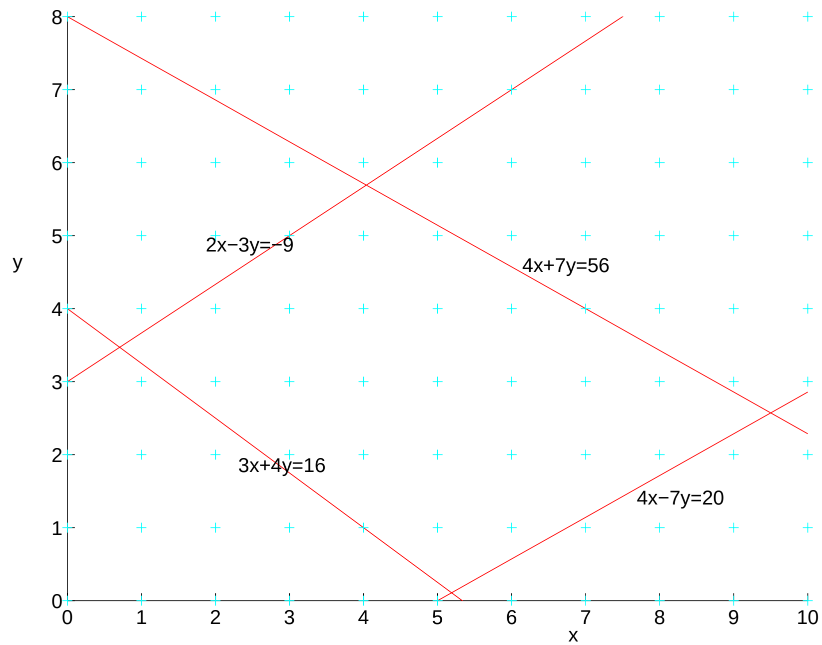
Not true: Reduced system has integer solutions => original system does too.



Key problem: Multiplying one inequality by b_1 and other by d_1 is not guaranteed to preserve "integrality" (cf. equalities)

Exact projection: If all upper bound coefficients b_i or all lower bound coefficients d_i happen to be 1, then integer solution to reduced system implies integer solution to original system.

More accurate algorithm for determining existence



Just because there are integers between $4/37$ and $74/13$, we cannot assume there are integers in feasible region.

However, if gap between lower and upper bounds is greater than or equal to 1 for some integer value of y, there must be an integer in feasible region.

Dark shadow: region of y for which gap between upper and lower bounds of x is guaranteed to be greater than or equal to 1.

Determining dark shadow region:

Ordinary FM elimination:

$x \leq u$, $x \geq l => u \geq l$

Dark shadow:

$x \leq u$, $x \geq l => u \geq l + 1$

For our example, dark shadow projection along x gives system

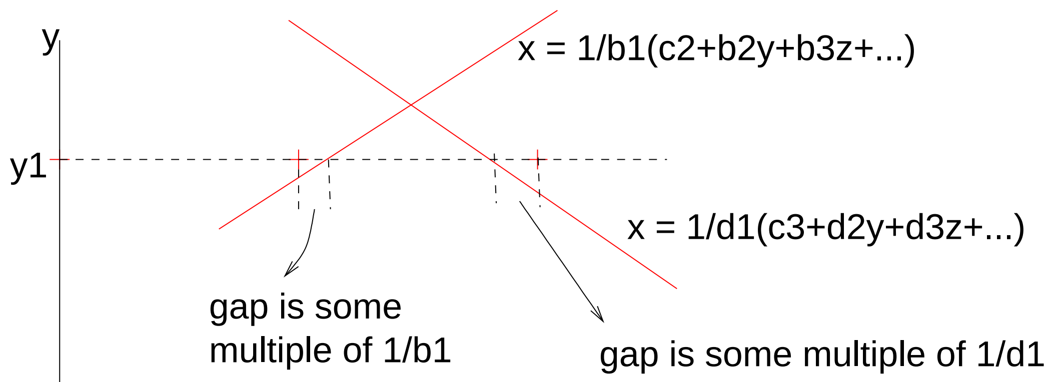
$$
\begin{aligned}
5 + 7y/4 &\geq 16/3 - 4y/3 + 1 (\text{Inequalities} 3, 1) \\
5 + 7y/4 &\geq -9/2 + 3y/2 + 1 (\text{Inequalities} 3, 4) \\
14 - 7y/4 &\geq 16/3 - 4y/3 + 1 (\text{Inequalities} 2, 1) \\
14 - 7y/4 &\geq -9/2 + 3y/2 + 1 (\text{Inequalities} 2, 4)
\end{aligned}
$$

=>

$66/13 \geq$ y $\geq 16/37$

There is an integer value of y in this range => integer in polyhedron.

More accurate estimate of dark shadow



For integer values of y1,z1,...., there is no integer value x1 between lower and upper bounds if

1/d1(c3+d2y1+d3z1+...) - 1/b1(c2+b2y1+b3z1+...) +1/b1+1/d1 <= 1

This means there is an integer between upper and lower bounds if

1/d1(c3+d2y1+d3z1+...) - 1/b1(c2+b2y1+b3z1+...) +1/b1+1/d1 > 1

To convert this to >=, notice that smallest change of lhs value is 1/b1d1.

So the inequality is

1/d1(c3+d2y1+d3z1+...) - 1/b1(c2+b2y1+b3z1+...) +1/b1+1/d1 >= 1 + 1/b1d1

=>

1/d1(c3+d2y1+d3z1+...) - 1/b1(c2+b2y1+b3z1+...) >= (1 - 1/b1)(1 - 1/d1)

Note: If $(b_1 = 1)$ or $(d_1 = 1)$, dark shadow constraint = real shadow constraint

Example:

$$3x \geq 16 - 4y$$
$$4x \leq 20 + 7y$$

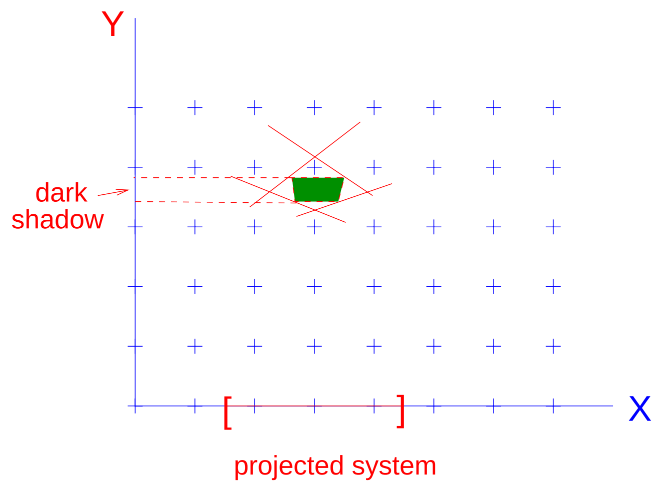
Real shadow: $(20 + 7y) * 3 \geq 4(16 - 4y)$

Dark shadow: $(20 + 7y) * 3 - 4(16 - 4y) \geq 12$

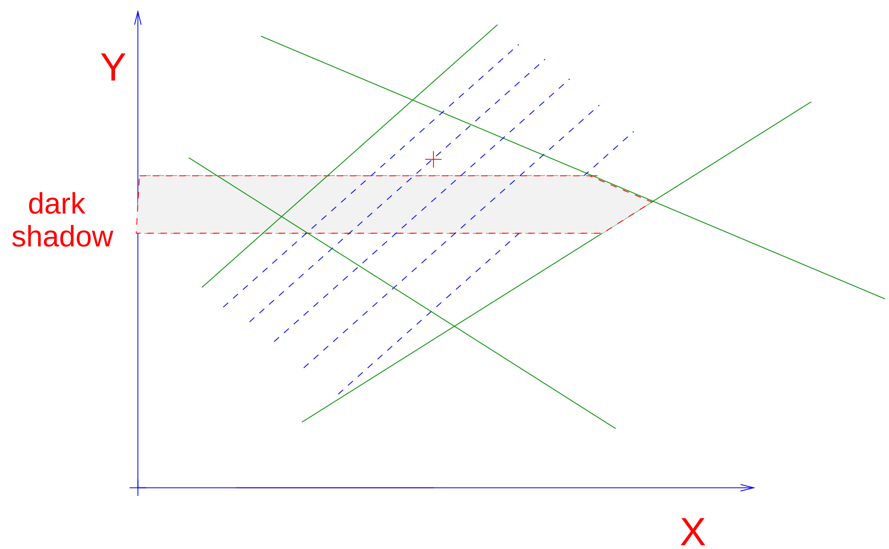
Dark shadow (improved): $(20 + 7y) * 3 - 4(16 - 4y) \geq 6$

What if dark shadow has no integers?

There may still be integer points nestled closely between an upper and lower bound.



One enumeration idea: splintering



Scan the corners with hyperplanes, looking for integer points.

Generate a succession of problems in which each lower bound is replaced with a sequence of hyperplanes. How many hyperplanes are needed?

Equation for lower bound: x = 1/b1(c2+b2y+b3z+....)

Hyperplanes:

x = 1/b1(c2+b2y+b3z+....)

x = 1/b1(c2+b2y+b3z+....)+ 1/b1

x = 1/b1(c2+b2y+b3z+....)+ 2/b1

x = 1/b1(c2+b2y+b3z+....)+ 3/b1

....

x = 1/b1(c2+b2y+b3z+....)+ 1 (in dark shadow region; if this is integer, so is)

Summary

- Two integer linear programming problems
  - Is there an integer point within a polyhedron $Ax \leq b$ where A is an integer matrix and b is an integer vector?
    - used for dependence analysis
  - Enumerate the integer points within a polyhedron $Ax \leq b$ where A is an integer matrix and b is an integer vector.
    - used for code generation after affine loop transformation