Extension 3

Sometimes, one loop alone does not provide enough opportunities for parallelization.
... but perhaps two successively in a row ...

Example

```c
for (x = 0; x < n; x++) {
    R = B[x];
    S = C[x];
    T1 = R + S;
    A[x] = T1;
}
```

```c
for (x = 0; x < n; x++) {
    R = B[x];
    S = C[x];
    T2 = R - S;
    C[x] = T2;
}
```
In order to fuse two loops into one, we require that:

- the iteration schemes coincide;
- the two loops access different data.

In case of individual variables, this can easily be verified.

This is more difficult in presence of arrays.

Taking the source program into account, accesses to distinct statically allocated arrays can be identified.

An analysis of accesses to the same array is significantly more difficult ...

The first loop may in iteration \( x \) not read data which the second loop writes to in iterations \( < x \).

The second loop may in iteration \( x \) not read data which the first loop writes to in iterations \( > x \).

If the index expressions of jointly accessed arrays are linear, the given constraints can be verified through integer linear programming ...

\[
\begin{align*}
   i & \geq 0 & \text{\texttt{xwrite}} = i \\
   i & \leq x - 1 & \text{\texttt{xread}} = x \\
   \text{\texttt{xread}} &= \text{\texttt{xwrite}}
\end{align*}
\]

\( \text{\texttt{xread}} \) read access to \( C \) by 1st loop
\( \text{\texttt{xwrite}} \) write access to \( C \) by 2nd loop

... obviously has no solution.

Assume that the blocks \( A, B, C \) are distinct.

Then we can combine the two loops into:

```c
for (x = 0; x < n; x++) {
    R = B[x];
    S = C[x];
    T_1 = R + S;
    A[x] = T_1;
    C[x] = T_2;
}
```

Assume that the blocks \( A, B, C \) are distinct.

Then we can combine the two loops into:

```c
for (x = 0; x < n; x++) {
    R = B[x];
    S = C[x];
    T_1 = R + S;
    A[x] = T_1;
    \textcolor{red}{C[x] = T_2;}
}
```
The first loop may in iteration \( x \) not read data which the second loop writes to in iterations \( < x \).

The second loop may in iteration \( x \) not read data which the first loop writes to in iterations \( > x \).

If the index expressions of jointly accessed arrays are linear, the given constraints can be verified through integer linear programming ...

\[
\begin{align*}
i & \geq 0 \\
i & \leq x - 1
\end{align*}
\]

\[
x_{\text{write}} = i \\
x_{\text{read}} = x
\]

\( \text{// } x_{\text{read}} \text{ read access to } C \text{ by 1st loop} \)

\( \text{// } x_{\text{write}} \text{ write access to } C \text{ by 2nd loop} \)

... obviously has no solution.

**Simple Case:**

The two inequations have no solution over \( \mathbb{Q} \).

Then they also have no solution over \( \mathbb{Z} \).

... in Our Example:

\[
\begin{align*}
x & = i \\
0 & \leq i \\
0 & \leq x - 1 - i = -1
\end{align*}
\]

The second inequation has no solution.

**General Form:**

\[
\begin{align*}
s & \geq t_1 \\
t_2 & \geq s \\
y_1 & = s_1 \\
y_2 & = s_2 \\
y_1 & = y_2
\end{align*}
\]

for linear expressions \( s, t_1, t_2, s_1, s_2 \) over \( i \) and the iteration variables.

This can be simplified to:

\[
0 \leq s - t_1 \\
0 \leq t_2 - s \\
0 = s_1 - s_2
\]

What should we do with it ???

**One Variable:**

The inequations where \( x \) occurs positive, provide lower bounds.

The inequations where \( x \) occurs negative, provide upper bounds.

If \( G, L \) are the greatest lower and the least upper bound, respectively, then all (integer) solution are in the interval \([G, L]\).

Example

\[
\begin{align*}
0 & \leq 13 - 7 \cdot x \\
0 & \leq -1 + 5 \cdot x
\end{align*}
\]

\( \iff \)

\[
0 \leq x \leq \frac{13}{7} \\
x \geq \frac{1}{5}
\]

The only integer solution of the system is \( x = 1 \).
Discussion

- Solutions only matter within the bounds to the iteration variables.
- Every integer solution there provides a conflict.
- Fusion of loops is possible if no conflicts occur.
- The given special case suffices to solve the case one variable over $\mathbb{Z}$.
- The number of variables in the inequations corresponds to the nesting-depth of for-loops $\implies$ in general, is quite small.

\[ \sum_{i=1}^{n} a_i \cdot x_i = b \quad \text{bzw.} \quad \sum_{i=1}^{n} a_i \cdot x_i \geq b, \quad a_i \in \mathbb{Z} \]

Moreover, a (linear) cost function can be optimized.

Warning: The decision problem is in general, already NP-hard!!!

Notwithstanding that, surprisingly efficient implementations exist.

Not just loop fusion, but also other re-organizations of loops yield ILP problems ...

Discussion

- Integer Linear Programming (ILP) can decide satisfiability of a finite set of equations/inequations over $\mathbb{Z}$ of the form:

\[ \sum_{i=1}^{n} a_i \cdot x_i = b \quad \text{bzw.} \quad \sum_{i=1}^{n} a_i \cdot x_i \geq b, \quad a_i \in \mathbb{Z} \]

Moreover, a (linear) cost function can be optimized.

Warning: The decision problem is in general, already NP-hard!!!

Notwithstanding that, surprisingly efficient implementations exist.

Not just loop fusion, but also other re-organizations of loops yield ILP problems ...
Background 5: Presburger Arithmetic

Many problems in computer science can be formulated without multiplication.

Let us first consider two simple special cases ...

1. Linear Equations

\[ 2x + 3y = 24 \]
\[ x - y + 5z = 3 \]

Question

- Is there a solution over \( \mathbb{Q} \)?
- Is there a solution over \( \mathbb{Z} \)?
- Is there a solution over \( \mathbb{N} \)?

Let us reconsider the equations:

\[ 2x + 3y = 24 \]
\[ x - y + 5z = 3 \]

Answers

- Is there a solution over \( \mathbb{Q} \)? Yes
- Is there a solution over \( \mathbb{Z} \)? No
- Is there a solution over \( \mathbb{N} \)? No

Solution Method for Integers

Observation 1

\[ a_1 x_1 + \ldots + a_k x_k = b \quad (\forall i : a_i \neq 0) \]

has a solution iff

\[ \gcd\{a_1, \ldots, a_k\} \mid b \]
Example

\[ 5y - 10z = 18 \]

has no solution over \( \mathbb{Z} \).

Example

\[ 5y - 10z = 18 \]

has no solution over \( \mathbb{Z} \).

Observation 2

Adding a multiple of one equation to another does not change the set of solutions.

Example

\[
\begin{align*}
2x + 3y &= 24 \\
x - y + 5z &= 3
\end{align*}
\]

Example

\[
\begin{align*}
2x + 3y &= 24 \\
x - y + 5z &= 3
\end{align*}
\]

\[
\begin{align*}
\rightarrow \\
5y - 10z &= 18 \\
x - y + 5z &= 3
\end{align*}
\]
Observation 3

Adding multiples of columns to another column is an invertible transformation which we keep track of in a separate matrix ...

\[ \begin{array}{c c c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \begin{array}{c} 5y - 10z = 18 \\ x - y + 5z = 3 \\ \end{array} \]

\[ \begin{array}{c c c} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{array} \begin{array}{c} 5y = 18 \\ x - y + 3z = 3 \end{array} \]

\[ \begin{array}{c c c} 1 & 0 & -3 \\ 0 & 1 & 2 \end{array} \begin{array}{c} 5y = 18 \\ x - \underline{y} = 3 \end{array} \]

\[ \begin{array}{c c c} 1 & 0 & 0 \end{array} \begin{array}{c} \text{triangular form} \end{array} \]

Observation 4

- A special solution of a triangular system can be directly read off.
- All solutions of a homogeneous triangular system can be directly read off.
- All solutions of the original system can be recovered from the solutions of the triangular system by means of the accumulated transformation matrix.
Example

\[
\begin{array}{ccc|c}
1 & 0 & -3 & x \\
0 & 1 & 2 & y \\
0 & 0 & 1 & \end{array}
\]

One special solution:

\[ [6, 3, 0]^T \]

All solutions of the homogeneous system are spanned by:

\[ [0, 0, 1]^T \]

---

Solving over \( \mathbb{N} \)

- ... is of major practical importance;
- ... has led to the development of many new techniques;
- ... easily allows to encode \textit{NP-hard} problems;
- ... remains difficult if just \textit{three} variables are allowed per equation.

---

Example

\[
\begin{array}{ccc|c}
1 & 0 & -3 & 5y \\
0 & 1 & 2 & x - y \\
0 & 0 & 1 & \end{array}
\]

One special solution:

\[ [6, 3, 0]^T \]

All solutions of the homogeneous system are spanned by:

\[ [0, 0, 1]^T \]

---

Solving over \( \mathbb{N} \)

- ... is of major practical importance;
- ... has led to the development of many new techniques;
- ... easily allows to encode \textit{NP-hard} problems;
- ... remains difficult if just \textit{three} variables are allowed per equation.
Idea: Represent the system by a graph:

2. One Polynomial Special Case

\[
\begin{align*}
  x &\geq y + 5 \\
  19 &\geq \sqrt{y} \\
  y &\geq 13 \\
  y &\geq x - 7 \\
\end{align*}
\]

- There are at most 2 variables per inequation;
- no scaling factors.

The inequations are satisfiable iff

- the weight of every cycle are at most 6;
- the weights of paths reaching x are bounded by the weights of edges from x into the sink.
3. A General Solution Method

**Idea:** Fourier-Motzkin Elimination

- Successively remove individual variables $x$!
- All in-equations with positive occurrences of $x$ yield lower bounds.
- All in-equations with negative occurrences of $x$ yield upper bounds.
- All lower bounds must be at most as big as all upper bounds.
The in-equations are **satisfiable** iff

- the weight of every cycle are at most 0;
- the weights of paths reaching $x$ are bounded by the weights of edges from $x$ into the sink.

Compute the reflexive and transitive closure of the edge weights!
3. A General Solution Method

Idea: Fourier-Motzkin Elimination

- Successively remove individual variables $x$!
- All in-equations with positive occurrences of $x$ yield lower bounds.
- All in-equations with negative occurrences of $x$ yield upper bounds.
- All lower bounds must be at most as big as all upper bounds.

Example

For $x_1$ we obtain:

\[
\begin{align*}
9 & \leq 4x_1 + x_2 \quad (1) \\
4 & \leq x_1 + 2x_2 \quad (2) \\
0 & \leq 2x_1 - x_2 \quad (3) \\
6 & \leq x_1 + 6x_2 \quad (4) \\
-11 & \leq -x_1 - 2x_2 \quad (5) \\
-17 & \leq -6x_1 + 2x_2 \quad (6) \\
-4 & \leq -x_2 \quad (7)
\end{align*}
\]

If such an $x_1$ exists, all lower bounds must be bounded by all upper bounds, i.e.,
\[
\begin{align*}
\frac{9}{4} - \frac{1}{2} x_2 & \leq 11 - 2 x_2 & (1, 5) & -35 \leq -7 x_2 & (1, 5) \\
\frac{\sqrt{a}}{b} - \frac{1}{c} x_2 & \leq \frac{d}{e} + \frac{1}{f} x_2 & (1, 6) & -\frac{g}{h} \leq \frac{i}{j} x_2 & (1, 6) \\
4 - 2 x_2 & \leq 11 - 2 x_2 & (2, 5) & -7 \leq 0 & (2, 5) \\
6 - 6 x_2 & \leq 11 - 2 x_2 & (3, 5) & -22 \leq -5 x_2 & (3, 5) \\
\frac{17}{6} - \frac{1}{3} x_2 & \leq \frac{17}{6} + \frac{1}{3} x_2 & (3, 6) & -\frac{17}{6} \leq -\frac{1}{6} x_2 & (3, 6) \\
6 - 6 x_2 & \leq 11 - 2 x_2 & (4, 5) & -5 \leq 4 x_2 & (4, 5) \\
6 - 6 x_2 & \leq 11 - 2 x_2 & (4, 6) & -\frac{10}{6} \leq \frac{10}{3} x_2 & (4, 6) \\
-4 \leq -x_2 & (7) & -4 \leq -x_2 & (7)
\end{align*}
\]

This is the one-variable case which we can solve exactly:

\[
\max \{ -1, \frac{1}{2} - \frac{5}{3}, \frac{1}{3} \} \leq x_2 \leq \min \{ 5, \frac{17}{6}, 17, \frac{1}{4} \}
\]

From which we conclude: \( x_2 \in \left[ \frac{1}{3}, 4 \right] \).

In General:

- The original system has a solution over \( \mathbb{Q} \) iff the system after elimination of one variable has a solution over \( \mathbb{Q} \).
- Every elimination step may square the number of in-equations \( \implies \) exponential run-time.
- It can be modified such that it also decides satisfiability over \( \mathbb{Z} \) \( \implies \) Omega Test.
Idea

- We successively remove variables. Thereby we omit division ...
- If $x$ only occurs with coefficient $\pm 1$, we apply Fourier-Motzkin elimination.
- Otherwise, we provide a bound for a positive multiple of $x$ ...

Consider, e.g., (1) and (6):

\[
\begin{align*}
6 \cdot x_1 & \leq 17 + 2x_2 \\
9 - x_2 & \leq 4 \cdot x_1
\end{align*}
\]

W.l.o.g., we only consider strict in-equations:

\[
\begin{align*}
6 \cdot x_1 & < 18 + 2x_2 \\
8 - x_2 & < 4 \cdot x_1
\end{align*}
\]

... where we always divide by gcds:

\[
\begin{align*}
3 \cdot x_1 & < 9 + x_2 \\
8 - x_2 & < 4 \cdot x_1
\end{align*}
\]

This implies:

\[
3 \cdot (8 - x_2) < 4 \cdot (9 + x_2)
\]