Chapter 1

Example: $C := AB^T + BA^T + C$ – Team: 20

1.1 Operation

Consider the operation

$$C := AB^T + BA^T + C$$

where C is a $m \times m$ lower triangular matrix and A and B is a $m \times m$ matrix. This is what we call a symmetric rank-2k update with the Lower triangular matrix on the Right. We will refer to this operation as SYR2K_LN where the LN stands for lower triangular no-transpose.

1.2 Precondition and postcondition

In the precondition

$$C = \widehat{C}$$

$\widehat{C}$ denotes the original contents of C. This allows us to express the state upon completion, the postcondition, as

$$C := AB^T + BA^T + \widehat{C}.$$

It is implicitly assumed that C is nonunit lower triangular.

1.3 Partitioned Matrix Expressions and loop invariants

There are two PMEs for this operation.

1.3.1 PME 1

To derive the first PME, partition

$$C \rightarrow \left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array} \right), \quad A \rightarrow \left(\begin{array}{c} A_T \\ \hline A_B \end{array} \right), \quad \text{and} \quad B \rightarrow \left(\begin{array}{c|c} B_T^T & B_B^T \end{array} \right)$$

Substituting these into the postcondition yields

$$\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c} A_T \\ \hline A_B \end{array}\right) \left(\begin{array}{c|c} B_T^T & B_B^T \end{array}\right) + \left(\begin{array}{c} B_T \\ \hline B_B \end{array}\right) \left(\begin{array}{c|c} A_T^T & A_B^T \end{array}\right) + \left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right)$$

or, equivalently,

$$\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c|c} A_T B_T^T + B_T A_T^T + C_{TL} & * \\ \hline A_B B_T^T + B_B A_T^T + C_{BL} & A_B B_B^T + B_B A_B^T + C_{BR} \end{array}\right)$$

which we refer to as the first PME for this operations.

From this, we can choose five loop invariants:

Invariant 1: $\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c|c} A_T B_T^T + B_T A_T^T + \widehat{C}_{TL} & * \\ \hline \widehat{C}_{BL} & \widehat{C}_{BR} \end{array}\right)$.

(The bottom part has been left alone and the top left parts have been completely computed)

Invariant 2: $\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c|c} \widehat{C}_{TL} & * \\ \hline \widehat{C}_{BL} & A_B B_B^T + B_B A_B^T + \widehat{C}_{BR} \end{array}\right)$.

(The left part has been left alone and the bottom right parts have been completely computed).

Invariant 3: $\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c|c} A_T B_T^T + B_T A_T^T + \widehat{C}_{TL} & * \\ \hline A_B B_T^T + \widehat{C}_{BL} & \widehat{C}_{BR} \end{array}\right)$.

(The bottom right part has been left alone, the bottom left part has been partially computed, and the top left part has been completely computed).

Invariant 4: $\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c|c} A_T B_T^T + B_T A_T^T + \widehat{C}_{TL} & * \\ \hline B_B A_T^T + \widehat{C}_{BL} & \widehat{C}_{BR} \end{array}\right)$.

(The bottom right part has been left alone, the bottom left part has been partially computed, and the bottom right part has been completely computed).

Invariant 5: $\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c|c} \widehat{C}_{TL} & * \\ \hline A_B B_T^T + \widehat{C}_{BL} & A_B B_B^T + B_B A_B^T + \widehat{C}_{BR} \end{array}\right)$.

(The top left part has been left alone, the bottom left part has been partially computed, and the bottom right part has been completely computed).

Invariant 6: $\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c|c} \widehat{C}_{TL} & * \\ \hline B_B A_T^T + \widehat{C}_{BL} & A_B B_B^T + B_B A_B^T + \widehat{C}_{BR} \end{array}\right)$.

(The top left part has been left alone, the bottom left part has been partially computed, and the bottom right part has been completely computed).

Invariant 7: $\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c|c} \widehat{C}_{TL} & * \\ \hline A_B B_T^T + B_B A_T^T + \widehat{C}_{BL} & A_B B_B^T + B_B A_B^T + \widehat{C}_{BR} \end{array}\right)$.
(The top left part has been left alone, the bottom left part has been completely computed, and the bottom right part has been completely computed).

Invariant 8: $\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c|c} A_T B_T^T + B_T A_T^T + \widehat{C}_{TL} & * \\ \hline A_B B_T^T + B_B A_T^T + \widehat{C}_{BL} & \widehat{C}_{BR} \end{array}\right)$.
(The bottom right part has been left alone, the bottom left part has been completely computed, and the top left part has been completely computed).

1.3.2 PME 2

To derive the second PME, partition

$$A \rightarrow \left(\begin{array}{c|c} A_L & A_R \end{array}\right), \quad \text{and} \quad B \rightarrow \left(\begin{array}{c|c} B_L & B_R \end{array}\right)$$

and do not partition C. Substituting these into the postcondition yields

$$C = \left(\begin{array}{c|c} A_L & A_R \end{array}\right) \left(\begin{array}{c} B_L^T \\ \hline B_R^T \end{array}\right) + \left(\begin{array}{c|c} B_L & B_R \end{array}\right) \left(\begin{array}{c} A_L^T \\ \hline A_R^T \end{array}\right) + C$$

or, equivalently,

$$C = A_L B_L^T + A_R B_R^T + B_L A_L^T + B_R A_R^T + \widehat{C}$$

which we refer to as the second PME.

From this, we can choose one more loop invariant:

Invariant 9: $C = A_L B_L^T + B_L A_L^T + \widehat{C}$. (The left part has been completely finished and the right part has been left untouched).

Invariant 10: $C = A_R B_R^T + B_R A_R^T + \widehat{C}$. (The right part has been completely finished and the left part has been left untouched).

1.3.3 Notes

How do I decide to partition the matrices in the postcondition?

- Pick a matrix (operand), any matrix.
- If that matrix has
  - a triangular structure (in storage), then you want to either partition is into four quadrants, or not at all. Symmetric matrices and triangular matrices have a triangular structure (in storage).
  - no particular structure, then you partition it vertically (left-right), horizontally (top-bottom), or not at all.

- Next, partition the other matrices similarly, but conformally (meaning the resulting multiplications with the parts are legal).

Take our problem here: $C := AB^T + BA^T + C$. Start by partitioning C in to quadrants:

$$\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = AB^T + BA^T + \left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right)$$

Now, the way partitioned matrix multiplication works, this doesn't make sense:

$$\underbrace{\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = AB^T + BA^T + \left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right)}_{\left(\begin{array}{c|c} C_{TL} + \text{something} & * + \text{something} \\ \hline C_{BL} + \text{something} & C_{BR} + \text{something} \end{array}\right)} .$$

So, we need to also partition A and B into a top part and a bottom part:

$$\left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \underbrace{\left(\begin{array}{c} A_T \\ \hline A_B \end{array}\right)\left(\begin{array}{c|c} B_T^T & B_B^T \end{array}\right) + \left(\begin{array}{c} B_T \\ \hline B_B \end{array}\right)\left(\begin{array}{c|c} A_T^T & A_B^T \end{array}\right) + \left(\begin{array}{c|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right)}_{\left(\begin{array}{c|c} A_T B_T^T + B_T A_T^T + C_{TL} & * \\ \hline A_B B_T^T + B_B A_T^T + C_{BL} & A_B B_B^T + B_B A_B^T + C_{BR} \end{array}\right)}$$

Alternatively, what if you don't partition C? You have to partition *something* so let's try partitioning A and B:

$$C = \left(\begin{array}{c|c} A_L & A_R \end{array}\right)\left(\begin{array}{c} B_L^T \\ \hline B_R^T \end{array}\right) + \left(\begin{array}{c|c} B_L & B_R \end{array}\right)\left(\begin{array}{c} A_L^T \\ \hline A_R^T \end{array}\right) + C$$

1.4 Deriving all unblocked algorithms

The below table summarizes all loop invariants, with links to all files related to this operation.
The worksheet and code skeletons were genered using the ☛ Spark webpage.

	Invariant	Derivations	Implementations
1	$\left(\begin{array}{c\|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c\|c} A_T B_T^T + B_T A_T^T + \widehat{C}_{TL} & * \\ \hline \widehat{C}_{BL} & \widehat{C}_{BR} \end{array}\right)$	PDF	syr2k_ln_unb_var1.mlx syr2k_ln_unb_var1.c
2	$\left(\begin{array}{c\|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c\|c} \widehat{C}_{TL} & * \\ \hline \widehat{C}_{BL} & A_B B_B^T + B_B A_B^T + \widehat{C}_{BR} \end{array}\right)$	PDF	syr2k_ln_unb_var2.mlx syr2k_ln_unb_var2.c
3	$\left(\begin{array}{c\|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c\|c} A_T B_T^T + B_T A_T^T + \widehat{C}_{TL} & * \\ \hline A_B B_T^T + \widehat{C}_{BL} & \widehat{C}_{BR} \end{array}\right)$	PDF	syr2k_ln_unb_var3.mlx syr2k_ln_unb_var3.c
4	$\left(\begin{array}{c\|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c\|c} A_T B_T^T + B_T A_T^T + \widehat{C}_{TL} & * \\ \hline B_B A_T^T + \widehat{C}_{BL} & \widehat{C}_{BR} \end{array}\right)$	PDF	syr2k_ln_unb_var4.mlx syr2k_ln_unb_var4.c
5	$\left(\begin{array}{c\|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c\|c} \widehat{C}_{TL} & * \\ \hline A_B B_T^T + \widehat{C}_{BL} & A_B B_B^T + B_B A_B^T + \widehat{C}_{BR} \end{array}\right)$	PDF	syr2k_ln_unb_var5.mlx syr2k_ln_unb_var5.c
9	$C = A_L B_L^T + B_L A_L^T + \widehat{C}$	PDF	syr2k_ln_unb_var9.mlx syr2k_ln_unb_var9.c

1.5 Deriving all blocked algorithms

The below table summarizes all loop invariants, with links to all files related to this operation. The worksheet and code skeletons were genered using the ☛ Spark webpage.

	Invariant	Derivations	Implementations
1	$\left(\begin{array}{c\|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c\|c} A_T B_T^T + B_T A_T^T + \widehat{C}_{TL} & * \\ \hline \widehat{C}_{BL} & \widehat{C}_{BR} \end{array}\right)$	PDF	syr2k_ln_blk_var1.mlx syr2k_ln_blk_var1.c
2	$\left(\begin{array}{c\|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c\|c} \widehat{C}_{TL} & * \\ \hline \widehat{C}_{BL} & A_B B_B^T + B_B A_B^T + \widehat{C}_{BR} \end{array}\right)$	PDF	syr2k_ln_blk_var2.mlx syr2k_ln_blk_var2.c
3	$\left(\begin{array}{c\|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c\|c} A_T B_T^T + B_T A_T^T + \widehat{C}_{TL} & * \\ \hline A_B B_T^T + \widehat{C}_{BL} & \widehat{C}_{BR} \end{array}\right)$	PDF	syr2k_ln_blk_var3.mlx syr2k_ln_blk_var3.c
4	$\left(\begin{array}{c\|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c\|c} A_T B_T^T + B_T A_T^T + \widehat{C}_{TL} & * \\ \hline B_B A_T^T + \widehat{C}_{BL} & \widehat{C}_{BR} \end{array}\right)$	PDF	syr2k_ln_blk_var4.mlx syr2k_ln_blk_var4.c
5	$\left(\begin{array}{c\|c} C_{TL} & * \\ \hline C_{BL} & C_{BR} \end{array}\right) = \left(\begin{array}{c\|c} \widehat{C}_{TL} & * \\ \hline A_B B_T^T + \widehat{C}_{BL} & A_B B_B^T + B_B A_B^T + \widehat{C}_{BR} \end{array}\right)$	PDF	syr2k_ln_blk_var5.mlx syr2k_ln_blk_var5.c
9	$C = A_L B_L^T + B_L A_L^T + \widehat{C}$	PDF	syr2k_ln_blk_var9.mlx syr2k_ln_blk_var9.c