The Kernel Scheduling Problem: Given a computing device *D*, plus the specification of a computational kernel *K*, plus valid inputs for *K*, the kernel scheduling problem asks for the fastest implementation of *K* for these inputs on *D* (Simplified Definition, taken from *Canesche et al*. [1]]).

Below we see the specification of a kernel, and five ways to implement it.

```
void mm_jki_roll(f64* A, f64* B, f64* C,
                 int RA, int CA, int CB) {
  for (int j0 = 0; j0 < CB; j0++) {
    for (int k0 = 0; k0 < CA; k0++) {
      for (int i0 = 0; i0 < RA; i0 += 2) {
        int ij = i0 * CB + j0;
        int ik = i0 * CA + k0;
        int kj = k0 * CB + j0;
        C[ij] += A[ik] * B[kj];
        ij = (i0+1) * CB + j0;
        ik = (i0+1) * CA + k0;
        kj = k0 * CB + j0;
        C[ij] += A[ik] * B[kj];
} } } }
```

The specification of a computational kernel

Data space:

$A \subseteq (0 \dots R_A) \times (0 \dots C_A)$

$B \subseteq (0 \dots R_B) \times (0 \dots C_B)$

$C \subseteq (0 \dots R_C) \times (0 \dots C_C)$

Iteration space:

$i \in (0 \dots R_A)$

$j \in (0 \dots C_B)$

$k \in (0 \dots C_A)$

Computation:

$C_{ij} = C_{ij} + A_{ik} \times B_{kj}$

Constraints:

$C_A = R_B,\ R_A \ \%\ 2 = 0$

```
void mm_jki(f64* A, f64* B, f64* C,
            int RA, int CA, int CB) {
  for (int j0 = 0; j0 < CB; j0++) {
    for (int k0 = 0; k0 < CA; k0++) {
      for (int i0 = 0; i0 < RA; i0++) {
        int ij = i0 * CB + j0;
        int ik = i0 * CA + k0;
        int kj = k0 * CB + j0;
        C[ij] += A[ik] * B[kj];
} } } }
```

Question: "*Kernel scheduling*" is how this problem is known in almost every paper talking about optimizations of machine learning models. Why is this problem not called "*Kernel Optimization*" or "*Code Generation for Kernels*" or something like that?

```
void mm_jki_tile_16(f64* A, f64* B, f64* C,
                    int RA, int CA, int CB) {
  for (int j0 = 0; j0 < CB; j0 += 16) {
    for (int k0 = 0; k0 < CA; k0 += 32) {
      for (int i0 = 0; i0 < RA; i0 += 16) {
        int jmax = j0 + 16 > CB ? CB : j0 + 16;
        for (int j1 = j0; j1 < jmax; ++j1) {
          int kmax = k0 + 32 > CA ? CA : k0 + 32;
          for (int k1 = k0; k1 < kmax; ++k1) {
            int imax = i0 + 16 > RA ? RA : i0 + 16;
            for (int i1 = i0; i1 < imax; ++i1) {
              int ij = i1 * CB + j1;
              int ik = i1 * CA + k1;
              int kj = k1 * CB + j1;
              C[ij] += A[ik] * B[kj];
} } } } } }
```

```
void mm(f64* A, f64* B, f64* C,
        int RA, int CA, int CB) {
  for (int i0 = 0; i0 < RA; i0++) {
    for (int j0 = 0; j0 < CB; j0++) {
      for (int k0 = 0; k0 < CA; k0++) {
        int ij = i0 * CB + j0;
        int ik = i0 * CA + k0;
        int kj = k0 * CB + j0;
        C[ij] += A[ik] * B[kj];
} } } }
```

[1] *Michael Canesch, Anderson M. Rosário, Edson Borin, Fernando Magno Quintão Pereira:* ***The Droplet Search Algorithm for Kernel Scheduling***. ACM Transactions on Architecture and Code Optimization, 2024

UFMG
DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
Compilers Laboratory