"**"

"Iteration Domain of R:"

[TimeSteps, N] -> { DS_R[t, i1, j1] : 0 <= t < TimeSteps and 0 < i1 <= -2 + N and 0 < j1 <= -2 + N }

"Cardinality of R:"

[TimeSteps, N] -> { (4 * TimeSteps + -4 * TimeSteps * N + TimeSteps * N^2) : TimeSteps > 0 and N >= 3 }

"Iteration Domain of S:"

"Cardinality of S:"

[TimeSteps, N] -> { DS_S[t, i2, j2] : 0 <= t < TimeSteps and 0 < i2 <= -2 + N and 0 < j2 <= -2 + N }

[TimeSteps, N] -> { (4 * TimeSteps + -4 * TimeSteps * N + TimeSteps * N^2) : TimeSteps > 0 and N >= 3 }

"**"

"Access Function of R:"

[TimeSteps, N] -> { DS_R[t, i1, j1] -> B[i1, j1] }

[TimeSteps, N] -> { DS_R[t, i1, j1] -> A[i1, -1 + j1] }

[TimeSteps, N] -> { DS_R[t, i1, j1] -> A[i1, j1] }

[TimeSteps, N] -> { DS_R[t, i1, j1] -> A[i1, 1 + j1] }

[TimeSteps, N] -> { DS_R[t, i1, j1] -> A[1 + i1, j1] }

[TimeSteps, N] -> { DS_R[t, i1, j1] -> A[-1 + i1, j1] }

"Access Function of S:"

[TimeSteps, N] -> { DS_S[t, i2, j2] -> A[i2, j2] }

[TimeSteps, N] -> { DS_S[t, i2, j2] -> B[i2, -1 + j2] }

[TimeSteps, N] -> { DS_S[t, i2, j2] -> B[i2, j2] }

[TimeSteps, N] -> { DS_S[t, i2, j2] -> B[i2, 1 + j2] }

[TimeSteps, N] -> { DS_S[t, i2, j2] -> B[1 + i2, j2] }

[TimeSteps, N] -> { DS_S[t, i2, j2] -> B[-1 + i2, j2] }

"**"

"Union of Access functions on A"

[TimeSteps, N] -> { DS_S[t, i2, j2] -> A[i2, j2]; DS_R[t, i1, j1] -> A[i1, 1 + j1]; DS_R[t, i1, j1] -> A[1 + i1, j1]; DS_R[t, i1, j1] -> A[i1, j1]; DS_R[t, i1, j1] -> A[-1 + i1, j1]; DS_R[t, i1, j1] -> A[i1, -1 + j1] }

"**"

"Union of Access functions on B"

[TimeSteps, N] -> { DS_R[t, i1, j1] -> B[i1, j1]; DS_S[t, i2, j2] -> B[i2, 1 + j2]; DS_S[t, i2, j2] -> B[1 + i2, j2]; DS_S[t, i2, j2] -> B[i2, j2]; DS_S[t, i2, j2] -> B[-1 + i2, j2]; DS_S[t, i2, j2] -> B[i2, -1 + j2] }

"**"

"Data Space of A"

[TimeSteps, N] -> { A[x1, x2] : TimeSteps > 0 and 0 <= x1 < N and 0 <= x2 < N and ((x1 >= 2 and 0 < x2 <= -2 + N) or (0 < x1 <= -2 + N and x2 >= 2) or (0 < x1 <= -2 + N and 0 < x2 <= -2 + N) or (0 < x1 <= -2 + N and x2 <= -3 + N) or (x1 <= -3 + N and 0 < x2 <= -2 + N)) }

"**"

"Data Space of B"

[TimeSteps, N] -> { B[x1, x2] : TimeSteps > 0 and 0 <= x1 < N and 0 <= x2 < N and ((x1 >= 2 and 0 < x2 <= -2 + N) or (0 < x1 <= -2 + N and x2 >= 2) or (0 < x1 <= -2 + N and 0 < x2 <= -2 + N) or (0 < x1 <= -2 + N and x2 <= -3 + N) or (x1 <= -3 + N and 0 < x2 <= -2 + N)) }

"**"

"number of points in A"

[TimeSteps, N] -> { (-4 + N^2) : TimeSteps > 0 and N >= 4; 5 : N = 3 and TimeSteps > 0 }

"number of points in B"

[TimeSteps, N] -> { (-4 + N^2) : TimeSteps > 0 and N >= 4; 5 : N = 3 and TimeSteps > 0 }

"**"

"Combined Data space:"

[TimeSteps, N] -> { DS_R[t, i1, j1] : 0 <= t < TimeSteps and 0 < i1 <= -2 + N and 0 < j1 <= -2 + N; DS_S[t, i2, j2] : 0 <= t < TimeSteps and 0 < i2 <= -2 + N and 0 < j2 <= -2 + N }

"The sum for R and S of number of executed statement instances:"

[TimeSteps, N] -> { (8 * TimeSteps + -8 * TimeSteps * N + 2 * TimeSteps * N^2) : TimeSteps > 0 and N >= 3 }

"**"

"paramaterized access functions of A in R:"

[TimeSteps, N, c, d] -> { DS_R[t, c, 1 + d] -> A[c, d] }

[TimeSteps, N, c, d] -> { DS_R[t, c, d] -> A[c, d] }

[TimeSteps, N, c, d] -> { DS_R[t, c, -1 + d] -> A[c, d] }

[TimeSteps, N, c, d] -> { DS_R[t, -1 + c, d] -> A[c, d] }

[TimeSteps, N, c, d] -> { DS_R[t, 1 + c, d] -> A[c, d] }

"Union of paramaterized access functionsof A in R"

[TimeSteps, N, c, d] -> { DS_R[t, c, 1 + d] -> A[c, d]; DS_R[t, 1 + c, d] -> A[c, d]; DS_R[t, c, d] -> A[c, d]; DS_R[t, -1 + c, d] -> A[c, d]; DS_R[t, c, -1 + d] -> A[c, d] }

[TimeSteps, N, c, d] -> { DS_R[t, c, 1 + d] -> A[c, d] : 0 < c <= -2 + N and 0 <= d <= -3 + N and 0 <= t < TimeSteps; DS_R[t, 1 + c, d] -> A[c, d] : 0 <= c <= -3 + N and 0 < d <= -2 + N and 0 <= t < TimeSteps; DS_R[t, c, d] -> A[c, d] : 0 < c <= -2 + N and 0 < d <= -2 + N and 0 <= t < TimeSteps; DS_R[t, -1 + c, d] -> A[c, d] : 2 <= c < N and 0 < d <= -2 + N and 0 <= t < TimeSteps; DS_R[t, c, -1 + d] -> A[c, d] : 0 < c <= -2 + N and 2 <= d < N and 0 <= t < TimeSteps; DS_S[t, c, d] -> A[c, d] : 0 < c <= -2 + N and 0 < d <= -2 + N and 0 <= t < TimeSteps }

"number of times data element of A being accessed by the program"

[TimeSteps, N, c, d] -> { 6 * TimeSteps : TimeSteps > 0 and 2 <= c <= -3 + N and 2 <= d <= -3 + N; 5 * TimeSteps : c = -2 + N and TimeSteps > 0 and N >= 4 and 2 <= d <= -3 + N; 5 * TimeSteps : d = 1 and TimeSteps > 0 and N >= 4 and 2 <= c <= -3 + N; 4 * TimeSteps : c = -2 + N and d = 1 and TimeSteps > 0 and N >= 4; 5 * TimeSteps : c = 1 and TimeSteps > 0 and N >= 4 and 2 <= d <= -3 + N; 4 * TimeSteps : c = 1 and d = 1 and TimeSteps > 0 and N >= 4; 5 * TimeSteps : d = -2 + N and TimeSteps > 0 and N >= 4 and 2 <= c <= -3 + N; 4 * TimeSteps : c = -2 + N and d = -2 + N and TimeSteps > 0 and N >= 4; 4 * TimeSteps : c = 1 and d = -2 + N and TimeSteps > 0 and N >= 4; 2 * TimeSteps : N = 3 and c = 1 and d = 1 and TimeSteps > 0; TimeSteps : d = 0 and TimeSteps > 0 and 0 < c <= -2 + N; TimeSteps : c = -1 + N and TimeSteps > 0 and 0 < d <= -2 + N; TimeSteps : d = -1 + N and TimeSteps > 0 and ((2 <= c <= -2 + N) or (c = 1 and N >= 3)); TimeSteps : c = 0 and TimeSteps > 0 and 0 < d <= -2 + N }

"***"

"sliced iteration domain for A in R in j1 dimension"

[TimeSteps, N, c, d] -> { DS_R_Aj1[c, d, j1] : 0 <= c < TimeSteps and 0 < d <= -2 + N and 0 < j1 <= -2 + N }

"Elements in DS_R_Aj1x"

[TimeSteps, N, c, d] -> { (-2 + N) : 0 <= c < TimeSteps and 0 < d <= -2 + N }

"Access function for sliced iteration domain"

[TimeSteps, N] -> { DS_R_Aj1[t, i1, j1] -> A[i1, -1 + j1] }

[TimeSteps, N] -> { DS_R_Aj1[t, i1, j1] -> A[i1, j1] }

[TimeSteps, N] -> { DS_R_Aj1[t, i1, j1] -> A[i1, 1 + j1] }

[TimeSteps, N] -> { DS_R_Aj1[t, i1, j1] -> A[1 + i1, j1] }

[TimeSteps, N] -> { DS_R_Aj1[t, i1, j1] -> A[-1 + i1, j1] }

"union of access functions"

[TimeSteps, N] -> { DS_R_Aj1[t, i1, j1] -> A[i1, 1 + j1]; DS_R_Aj1[t, i1, j1] -> A[1 + i1, j1]; DS_R_Aj1[t, i1, j1] -> A[i1, j1]; DS_R_Aj1[t, i1, j1] -> A[-1 + i1, j1]; DS_R_Aj1[t, i1, j1] -> A[i1, -1 + j1] }

"Data space of A for one arbitrary execution of the j1 loop"

[TimeSteps, N, c, d] -> { A[1 + d, x2] : 0 <= c < TimeSteps and 0 < d <= -2 + N and 0 < x2 <= -2 + N; A[d, x2] : 0 <= c < TimeSteps and 0 < d <= -2 + N and 0 <= x2 < N and (x2 >= 2 or (0 < x2 <= -2 + N) or x2 <= -3 + N); A[-1 + d, x2] : 0 <= c < TimeSteps and 0 < d <= -2 + N and 0 < x2 <= -2 + N }

"**"

"code for data space of A for one arbitrary execution of the j1 loop"

```
if (TimeSteps >= c + 1 && c >= 0 && N >= d + 2 && d >= 1)
  for (int c0 = d - 1; c0 <= d + 1; c0 += 1)
    for (int c1 = max(-d + c0, d - c0); c1 < min(N - d + c0, N + d - c0); c1 += 1)
      A(c0, c1);
```

"**"

"number of elements is DS_R_A_j1"

[TimeSteps, N, c, d] -> { (-4 + 3 * N) : 0 <= c < TimeSteps and 0 < d <= -2 + N }

"The size of A_local in each dimension would be N,N"

"**"

"Data space of B for one arbitrary execution of the j1 loop"

[TimeSteps, N, c, d] -> { DS_R_Bj1[c, d, j1] : 0 <= c < TimeSteps and 0 < d <= -2 + N and 0 < j1 <= -2 + N }

"number of elements is DS_R_B_j1"

[TimeSteps, N, c, d] -> { (-2 + N) : 0 <= c < TimeSteps and 0 < d <= -2 + N }

"access function for B in R for one arbitrary execution of the j1 loop"

[TimeSteps, N] -> { DS_R_Bj1[t, i1, j1] -> B[i1, j1] }

"Dataspace of B in R for one execution of j1 loop"

[TimeSteps, N, c, d] -> { B[d, x2] : 0 <= c < TimeSteps and 0 < d <= -2 + N and 0 < x2 <= -2 + N }

"number of elements in DS_R_Bj1_F"

[TimeSteps, N, c, d] -> { (-2 + N) : 0 <= c < TimeSteps and 0 < d <= -2 + N }

"The size of B_local in each dimension would be N,N"

"**"

"iteration domain for A_local"

[TimeSteps, N] -> { Alocal[t, i1, i, j] : 0 <= t < TimeSteps and 0 < i1 <= -2 + N and -1 + i1 <= i <= 1 + i1 and 0 <= j < N }

"Codegen for copying A into Alocal"

```
for (int c0 = 0; c0 < TimeSteps; c0 += 1)
  for (int c1 = 1; c1 < N - 1; c1 += 1)
    for (int c2 = c1 - 1; c2 <= c1 + 1; c2 += 1)
      for (int c3 = 0; c3 < N; c3 += 1)
        Alocal(c0, c1, c2, c3);
```

"Here Alocal(c0,c1,c2,c3) implies A_local[c2][c3] = A[c2][c3]"

"**"

"iteration domain for B_local"

[TimeSteps, N] -> { Blocal[t, i1, i] : 0 <= t < TimeSteps and 0 < i1 <= -2 + N and 0 < i <= -2 + N }

"Codegen for copying B from Blocal"

```
for (int c0 = 0; c0 < TimeSteps; c0 += 1)
  for (int c1 = 1; c1 < N - 1; c1 += 1)
    for (int c2 = 1; c2 < N - 1; c2 += 1)
      Blocal(c0, c1, c2);
```

"Here Blocal(c0,c1,c2) implies B[c1][c2] = B_local[c1][c2]"

"**"

"B_local[i1][j1] = 0.2 * (A_local[i1][j1-1] + A_local[i1][j1] + A_local[i1][j1+1] + A_local[i1+1][j1] + A_local[i1-1][j1])"

"**"