

# Optimization of Arbitrary Loop Nests

```
CodeReg scop {
  perfect = BuiltIn.IsPerfectLoopNest();
  depth = BuiltIn.LoopNestDepth();
  if (RoseLocus.IsDepAvailable()) {
    if (perfect && depth > 1) {
      permorder = permutation(seq(0,depth));
      RoseLocus.Interchange(order=permorder);
    }
  }

  if (perfect) {
    indexT1 = integer(1..depth);
    T1fac = poweroftwo(2..32);
    RoseLocus.Tiling(loop=indexT1, factor=T1fac);
  }
  OR {
    if (depth > 1) {
      indexUAJ = integer(1..depth-1);
      UAJfac = poweroftwo(2..4);
      RoseLocus.UnrollAndJam(loop=indexUAJ,
                           factor=UAJfac);
    }
  } OR {
    None; # No tiling, interchange, or unroll and jam.
  }
  innerloops = BuiltIn.ListInnerLoops();
  *RoseLocus.Distribute(loop=innerloops);
}
innerloops = BuiltIn.ListInnerLoops();
RoseLocus.Unroll(loop=innerloops,
                 factor=poweroftwo(2..8));
}
```

**37 lines of code**

```

1  CodeReg scop {
2    perfect = BuiltIn.IsPerfectLoopNest();
3    depth = BuiltIn.LoopNestDepth();
4    if (RoseLocus.IsDepAvailable()) {
5      if (perfect && depth > 1) {
6        permorder = permutation(seq(0,depth));
7        RoseLocus.Interchange(order=permorder);
8      }
9    }
10
11    if (perfect) {
12      indexT1 = integer(1..depth);
13      T1fac = poweroftwo(2..32);
14      RoseLocus.Tiling(loop=indexT1, factor=T1fac);
15    }
16    OR {
17      if (depth > 1) {
18        indexUAJ = integer(1..depth-1);
19        UAJfac = poweroftwo(2..4);
20        RoseLocus.UnrollAndJam(loop=indexUAJ,
21                             factor=UAJfac);
22      }
23    } OR {
24      None; # No tiling, interchange, or unroll and jam.
25    }
26    innerloops = BuiltIn.ListInnerLoops();
27    *RoseLocus.Distribute(loop=innerloops);
28  }
29  innerloops = BuiltIn.ListInnerLoops();
30  RoseLocus.Unroll(loop=innerloops,
31                  factor=poweroftwo(2..8));
32  }
33
34  }
35
36  }
37
38  }
39
40  }
41
42  }
43
44  }
45
46  }
47
48  }
49
50  }
51
52  }
53
54  }
55
56  }
57
58  }
59
60  }
61
62  }
63
64  }
65
66  }
67
68  }
69
70  }
71
72  }
73
74  }
75
76  }
77
78  }
79
80  }
81
82  }
83
84  }
85
86  }
87
88  }
89
90  }
91
92  }
93
94  }
95
96  }
97
98  }
99
100 }
```

# Optimization of Arbitrary Loop Nests

```
CodeReg scop {
  perfect = BuiltIn.IsPerfectLoopNest();
  depth = BuiltIn.LoopNestDepth();
  if (RoseLocus.IsDepAvailable()) {
    if (perfect && depth > 1) {
      permorder = permutation(seq(0,depth));
      RoseLocus.Interchange(order=permorder);
    }
  }
  if (perfect) {
    indexT1 = integer(1..depth);
    T1fac = poweroftwo(2..32);
    RoseLocus.Tiling(loop=indexT1, factor=T1fac);
  }
  } OR {
    if (depth > 1) {
      indexUAJ = integer(1..depth-1);
      UAJfac = poweroftwo(2..4);
      RoseLocus.UnrollAndJam(loop=indexUAJ,
                           factor=UAJfac);
    }
  } OR {
    None; # No tiling, interchange, or unroll and jam.
  }
  innerloops = BuiltIn.ListInnerLoops();
  *RoseLocus.Distribute(loop=innerloops);
}
innerloops = BuiltIn.ListInnerLoops();
RoseLocus.Unroll(loop=innerloops,
                 factor=poweroftwo(2..8));
}
```

**37 lines of code**

**1200+ lines of code**