Tillämpad Grafteori

inom

Kompilatorteknik

Torsdagen den 15 november

Stefan Björnander

Institutionen för Datavetenskap

```
   DO I = 1, 3
S1:   A(I) = 1              =>  A(1:3) = 1
   END DO                       S1(1:3)

   DO I = 1, 3
S2:   A(I) = I              =>  A(1:3) = (1:3)
   END DO

   DO I = 1, 2
S3:   A(I) = A(I + 1)       =>  A(1:2) = A(2:3)
   END DO

   DO I = 2, 3
S4:   A(I) = A(I - 1)       =/> A(2:3) = A(1:2)
   END DO

   DO I = 1, 2
     DO J = 1, 3
S5:     B(I, J) = 1         =>  B(1:2, 1:3) = 1
     END DO
   END DO

   DO I = 2, 2
     DO J = 1, 3
S6:     B(I,J) = B(I-1,J)   =>  B(2:2, 1:3) = B(1:1, 1:3)
     END DO
   END DO

   DO I = 2, 2
     DO J = 2, 3
S7:    B(I,J) = B(I-1,J-1) =>  B(2:2, 2:3) = B(1:1, 1:2)
     END DO
   END DO

   DO I = 1, 2
     DO J = 2, 3
S8:    B(I,J) = B(I,J-1)   =/> B(1:2, 2:3) = B(1:2, 1:2)
     END DO
   END DO
```

Vectorization – The Basic Approach

(1) Konstruera de starkt förbundna komponenterna (PI-blocken) av beroendegrafen D, och dess acykliska PI-graf G = AC(D).

(2) Sortera PI-blocken topologiskt

(3) Arrangera satserna så att ...

(a) Satser tillhörande olika PI-block sorteras topologiskt.

(b) Satser tillhörande olika PI-block sorteras i ursprunglig ordning.

(4) Distribuera loopnästet så att varje PI-block får ett eget loopnäste.

(5) Varje loopnäste med ett PI-block ({S}, ϕ} kan vektoriseras.

Rita beroendegrafen D med beroendetyp och beroendenivå för nedanstående loopnäste.

```
DO I = 2, 10
     DO J = 1, 20
S1:    A(I, J) = C(I + 1, J) + C(I, J + 1)
S2:    B(I, J) = A(I - 1, J)
S3:    C(I, J) = A(I + 1, J)
S4:    D(I, J) = A(I, J + 1) + D(I - 1, J) + B(I, J + 1)
     END DO
   END DO
```

D = (N, E) = ({S_1, S_2, S_3, S_4}, {(S_1, S_3, δ^a_1), (S_1, S_3, δ^a_2), (S_1, S_2, δ^t_1), (S_3, S_1, δ^a_1), (S_4, S_1, δ^a_2), (S_4, S_2, δ^a_2), (S_4, S_4, δ^t_1)})

Generera vektorkod utifrån ovanstående loopnäste med hjälp av *Vectorization - The Basic Approach*.

(1) Generara PI-grafen G = AC(D):

G = AC(D) = ({G_1, G_2, G_3}, {(G_1, G_2, δ^t_1), (G_3, G_1, δ^a_2), (G_3, G_2, δ^a_2)}), där

G_1 = ({S_1, S_3}, {(S_1, S_3, δ^a_1), {(S_1, S_3, δ^a_2), {(S_3, S_1, δ^a_1)})
G_2 = ({S_2}, ϕ)
G_3 = ({S_4}, {(S_4, S_4, δ^t_1)})

(2) Sortera G topologiskt:

G_3 -> G_1 -> G_2

(3) Sortera satserna så att ...

(a) Satser i olika PI-block sorteras topologiskt.
(b) Satser i samma PI-block sorteras i ursprunglig ordning.

```
   DO I = 2, 10
     DO J = 1, 20
S4:    D(I, J) = A(I, J + 1) + D(I - 1, J) + B(I, J + 1)
S1:    A(I, J) = C(I + 1, J) + C(I, J + 1)
S3:    C(I, J) = A(I + 1, J)
S2:    B(I, J) = A(I - 1, J)
     END DO
   END DO
```

(4) Distribuera loopen i topologisk ordning så att varje PI-block får ett egen loopnäste.

```
   DO I = 2, 10
     DO J = 1, 20
S4:    D(I, J) = A(I, J + 1) + D(I - 1, J) + B(I, J + 1)
     END DO
   END DO

   DO I = 2, 10
     DO J = 1, 20
S1:    A(I, J) = C(I + 1, J) + C(I, J + 1)
S3:    C(I, J) = A(I + 1, J)
     END DO
   END DO

   DO I = 2, 10
     DO J = 1, 20
S2:    B(I, J) = A(I - 1, J)
     END DO
   END DO
```

(5) Vektorisera alla loopnästen med PI-block $G_i = (\{S\}, \phi)$, dvs G_2.

```
   DO I = 2, 10
     DO J = 1, 20
S4:    D(I, J) = A(I, J + 1) + D(I - 1, J) + B(I, J + 1)
     END DO
   END DO

   DO I = 2, 10
     DO J = 1, 20
S1:    A(I, J) = C(I + 1, J) + C(I, J + 1)
S3:    C(I, J) = A(I + 1, J)
     END DO
   END DO

   S2(2:10, 1:20)

   eller

   B(2:10, 1:20) = A(1:9, 1:20)
```

Allen-Kennedy Vector Code Generation

Input: A Loop nest L with dependence graph D = DG(L) and statement set STAT.

Output: A program L', equivalent to L, where every statement, if possible, is vectorized.

```
procedure vectorize(R, c)
begin
    DD := constrain(D, c) | R
    G' = (N', E') := AC(DD)

    for every n' in N' in topologically order do
        if n' correspond to a cyclic SCC (PI-block) G_i = (N_i, E_i) of DD then
            generate("DO I_c = T_c, U_c")
            call vectorize(N_i, c + 1)
            generate("END DO")
        else
            generate("S(I_1, ..., I_c-1, T_c:U_c, ...,T_n:U_n)")
        end if
    end for
end procedure

begin program
    call vectorize(STAT, 1)
end program
```

Rita beroendegrafen D med beroendetyp och beroendenivå för nedanstående loopnäste.

```
   DO I = 2, 10
     DO J = 1, 20
S1:    A(I, J) = C(I + 1, J) + C(I, J + 1)
S2:    B(I, J) = A(I - 1, J)
S3:    C(I, J) = A(I + 1, J)
S4:    D(I, J) = A(I, J + 1) + D(I - 1, J) + B(I, J + 1)
     END DO
   END DO
```

D = (N, E) = ({S_1, S_2, S_3, S_4}, {(S_1, S_3, δ^a_1), (S_1, S_3, δ^a_2), (S_1, S_2, δ^t_1), (S_3, S_1, δ^a_1), (S_4, S_1, δ^a_2), (S_4, S_2, δ^a_2), (S_4, S_4, δ^t_1)})

Generera vektorkod utifrån ovanstående loopnäste med hjälp av *Allen-Kennedy Vector Code Generation*.

call vectorize({S_1, S_2, S_3, S_4}, 1)

DD := constrain(D, 1) | {S_1, S_2, S_3, S_4} = D

G' := (N', E') = AC(DD) = ({G_1, G_2, G_3},
{(G_1, G_2, δ^t_1), (G_3, G_1, δ^a_2), (G_3, G_2, δ^a_1)})

Topologisk ordning: G_3 -> G_1 -> G_2

G_3 = ({S_4}, {(S_4, S_4, δ^t_1)})
generate("DO I = 2, 10")
call vectorize({S_4}, 2)

DD := constrain(D, 2) | {S_4} = ({S_4}, ϕ)
G' := (N', E') = AC(DD) = ({$G_{3.1}$}, ϕ)

Topologisk ordning: $G_{3.1}$

$G_{3.1}$ = ({S_4}, ϕ)
generate("S_4(I, 1:20)")

generate("END DO")

G_1 = ({S_1, S_3}, {(S_1, S_3, δ^a_1), (S_1, S_3, δ^a_2), (S_3, S_1, δ^a_1)})
generate("DO I = 2, 10")
call vectorize({S_1, S_3}, 2)

DD := constrain(D, 2) | {S_1, S_3} = ({S_1, S_3}, {(S_1, S_3, δ^a_2)})
G' := (N', E') = AC(DD) = ({$G_{1.1}$, $G_{1.2}$}, {($G_{1.1}$, $G_{1.2}$, δ^a_2)})

Topologisk ordning: $G_{1.1}$ -> $G_{1.2}$

$G_{1.1}$ = ({S_1}, ϕ)
generate("S_1(I, 1:20)")

$G_{1.2}$ = ({S_3}, ϕ)
generate("S_3(I, 1:20)")

generate("END DO")

G_2 = ({S_2}, ϕ)
generate("S_2(2:10, 1:20)")

```
   DO I = 2, 10
     S4(I, 1:20)
   END DO

   DO I = 2, 10
     S1(I, 1:20)
     S3(I, 1:20)
   END DO

   S2(2:10, 1:20)
```

```
   DO I = 2, 10
S4:  D(I, 1:20) = A(I, 2:21) + D(I - 1,1:20) + B(I,2:21)
   END DO

   DO I = 2, 10
S1:  A(I, 1:20) = C(I + 1, 1:20) + C(I, 2:21)
S3:  C(I, 1:20) = A(I + 1, 1:20)
   END DO

   B(2:10, 1:20) = A(1:9, 1:20)
```

Stefan Björnander
stefanb@cs.umu.se
www.cs.umu.se/~stefanb