Global Data-flow Analysis

- A number of optimization can be achieved by knowing various pieces of information that can be obtained only by examining the entire program.
- For example, value for identifier A may be used in some part of the program, but to know where it has been defined, we need to examine the entire program.
- Solution is a global data flow analysis
- Method is Ud-chaining(use-definition chaining)
- For ud-chaining, Reaching Definitions should be found
- Then data-flow equations have to be determined iteratively

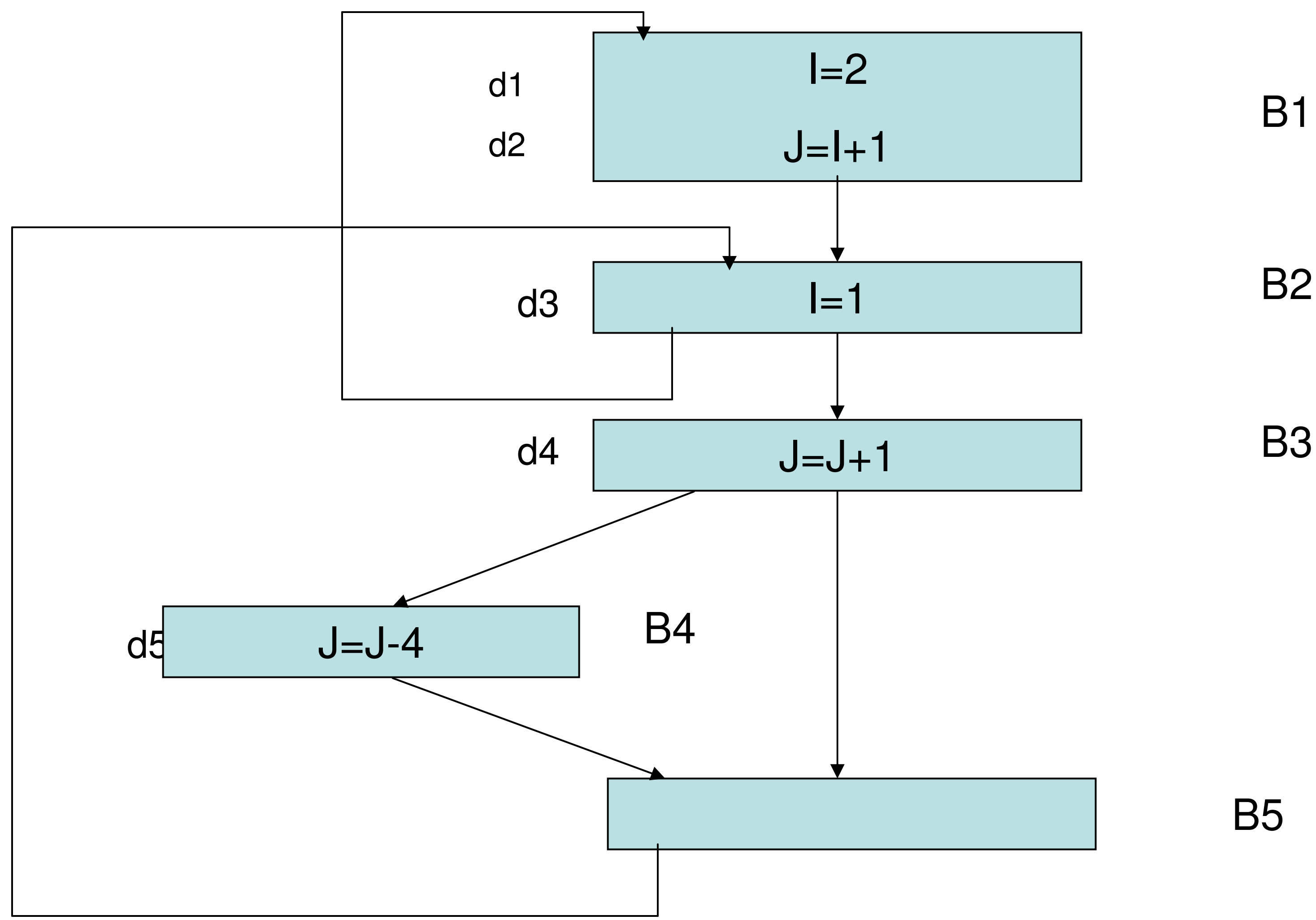
Reaching definition

1. Assign number to each definition
2. For simple variable A, make a list of all definitions anywhere in the program
3. Compute two sets for each basic block B
   - GEN[B]-set of generated definitions within basic block B that reach the end of the block
   - KILL[B]-set of definitions outside of B that have definition within B
   - Use bit vectors for representation
4. Compute set IN[B] for all blocks B-(all definitions reaching the point just before the first statement of Block B)
5. Compute set OUT[B] for all blocks B (the set of definitions reaching the point just after the last statement of B)
6. Repeat 4,5 iteratively until it gets converged

Data-flow equations

- OUT[B] = IN[B] – KILL[B] U GEN[B] {- means NOT AND}
- IN[B] = U OUT[P], P is a predecessor of B

Example



Step 1

Assign numbers to the definitions

- d1 -> I=2
- d2 -> J=I+1
- d3 -> I=1
- d4 -> J=J+1
- d5 -> J=J-4

Step 2

Make a list of all definitions

- I has the definitions
  - d1 in B1
  - d3 in B2
- J has the definitions
  - d2 in B1
  - d4 in B3
  - d5 in B4

Step 3

Compute GEN[B] and KILL[B]

Block B	GEN[B]	Bit Vector	KILL[B]	Bit Vector
B1	{d1,d2}	11000	{d3,d4,d5}	00111
B2	{d3}	00100	{d1}	10000
B3	{d4}	00010	{d2,d5}	01001
B4	{d5}	00001	{d2,d4}	01010
B5	ø	00000	ø	00000

Step 4,5

Compute data flow equations IN[B] and OUT[B]

Initial assumptions:

IN[B]= ø

OUT[B]=GEN[B]

With B = B1

NEWIN = OUT[B2] , B2 is the only predecessor

NEWIN=GEN[B2]=00100

NEWIN ≠ IN[B1]

IN[B1]=00100

OUT[B1]=IN[B1]-KILL[B1]UGEN[B1]

=00100-00111+11000=11000

With B = B2

NEWIN = OUT[B1]+OUT[B5] , B1, B5 are predecessors of B2

= 11000+00000=11000

IN[B2]=11000

OUT[B2]=11000-10000+00100=01100

Step 4,5 (cont.)

With B = B3

NEWIN = OUT[B2]= 01100

IN[B3]=01100

OUT[B3]=01100-01001+00010=00110

With B = B4

NEWIN = OUT[B3]= 00110

IN[B4]=00110

OUT[B4]=00110-01010+00001=00101

With B = B5

NEWIN = OUT[B3]+OUT[B4]= 00110+00101=00111

IN[B5]=00111

OUT[B5]=00111-00000+00000=00111

The use of J at d5 in block B4 is not preceded by a definition of J, so consider IN[B4]={d3,d4}, of these only d4 defines J, so the ud-chain for J in d5={d4}

Application: dead code elimination

Step 4,5

Compute data flow equations IN[B] and OUT[B]

	INITIAL		PASS-I	
Block	IN[B]	OUT[B]	IN[B]	OUT[B]
B1	00000	11000	00100	11000
B2	00000	00100	11000	01100
B3	00000	00010	01100	00110
B4	00000	00001	00110	00101
B5	00000	00000	00111	00111

	PASS-II		PASS=III	
Block	IN[B]	OUT[B]	IN[B]	OUT[B]
B1	01100	11000	01111	11000
B2	11111	01111	11111	01111
B3	01111	00110	01111	00110
B4	00110	00101	00110	00101
B5	00111	00111	00111	00111