

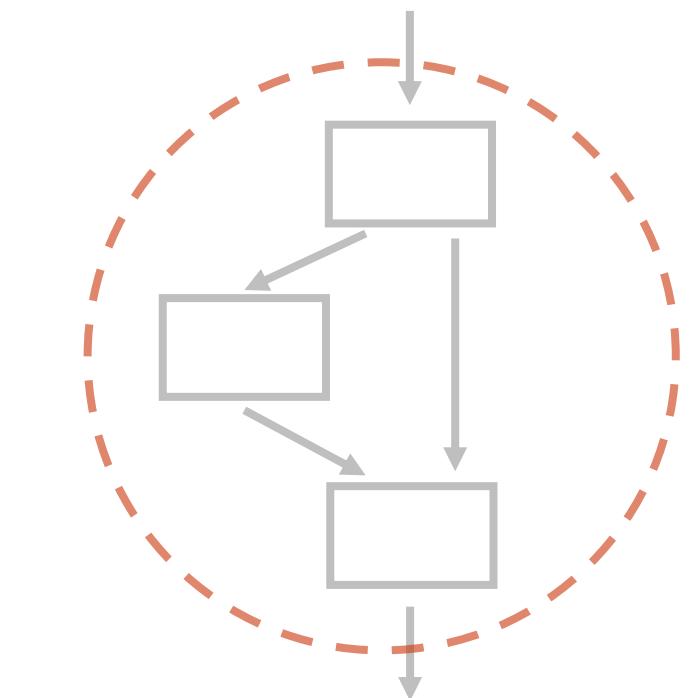
Dragon Book

Section 9.7
Region-Based Analysis

山本 航平

Region-Based Analysis

- 今まで見てきたデータフロー解析アルゴリズム：
各 basic block に対して transfer function を適用
→ 不動点に到達するまで繰り返す
- この節でのアプローチ：
Region (領域) に基づいた解析アルゴリズム
複数の basic block ・ 領域をまとめて考える

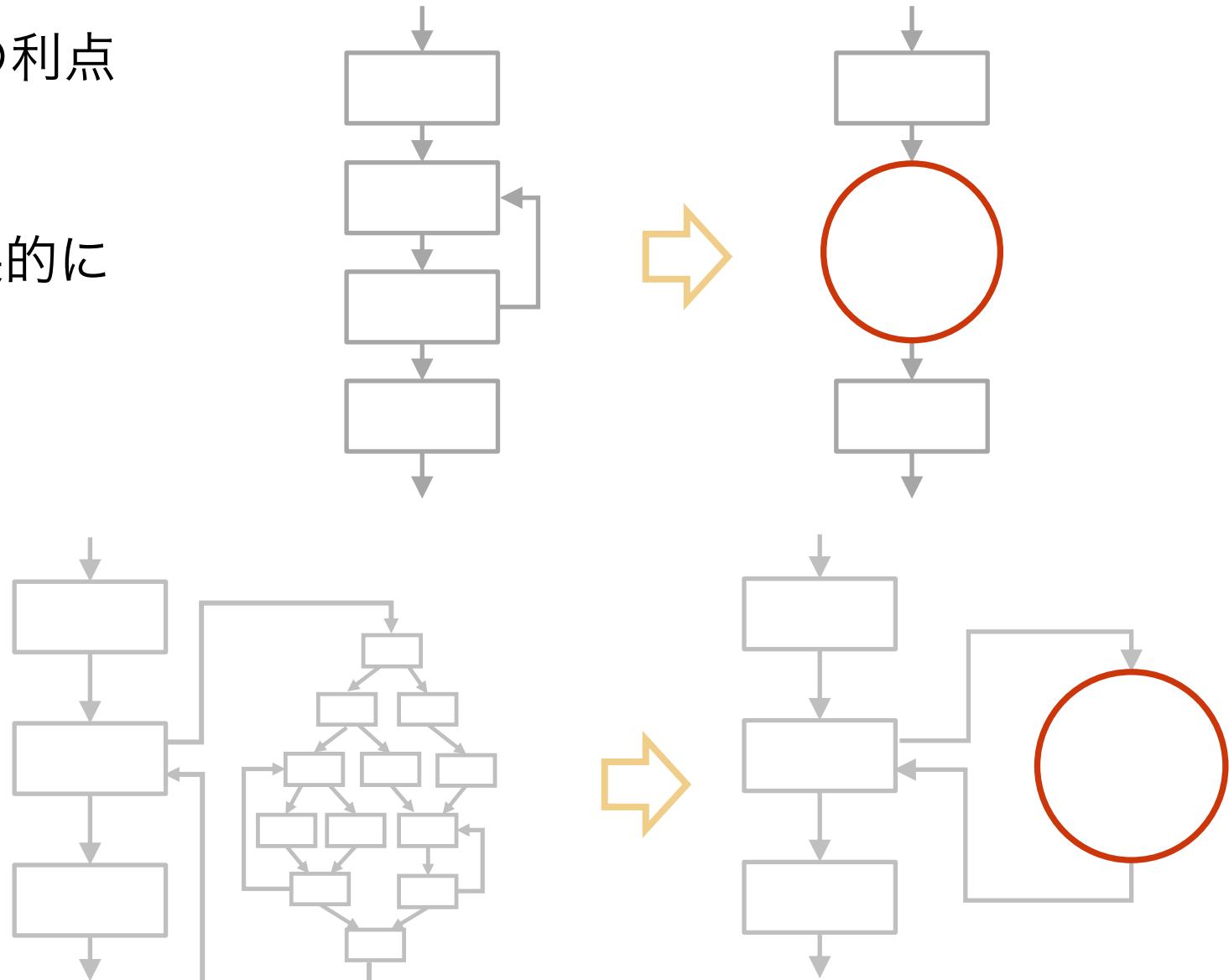


Region-Based Analysis

Region-based analysis の利点

- ・ループ
ループの影響をより効果的に
まとめられる

- ・手続き間の解析



この節でやること

1. 領域をどのように構築するか？
2. 領域間の transfer function をどう決めるか？
3. Region-based analysis アルゴリズムの概要

Contents

1. Regions
2. Overview of Region-Based Analysis
3. Transfer Functions
4. An Algorithm for Region-Based Analysis
5. Handling Nonreducible Flow Graphs

Contents

1. Regions
2. Overview of Region-Based Analysis
3. Transfer Functions
4. An Algorithm for Region-Based Analysis
5. Handling Nonreducible Flow Graphs

Regions

元々の flow graph の一部で, ただ一つの entry node を持つ

形式的定義:

次の条件を満たす頂点 N と辺 E の組

1. Header $h \in N$ が存在し, 領域上のすべての頂点を支配する

$$\forall n \in N, h \text{ dom } n$$

2. 頂点 m から h を通らずに $n \in N$ に到達できれば
 m は N に含まれる

3. E は $n_1, n_2 \in N$ 間の flow graph 上のすべての辺を含む

Regions

1. Header $h \in N$ が存在し, 領域上のすべての頂点を支配する

$$\forall n \in N, h \text{ dom } n$$

Q. $h \text{ dom } n$ の意味は?

Regions

1. Header $h \in N$ が存在し, 領域上のすべての頂点を支配する

$$\forall n \in N, h \text{ dom } n$$

Q. $h \text{ dom } n$ の意味は?

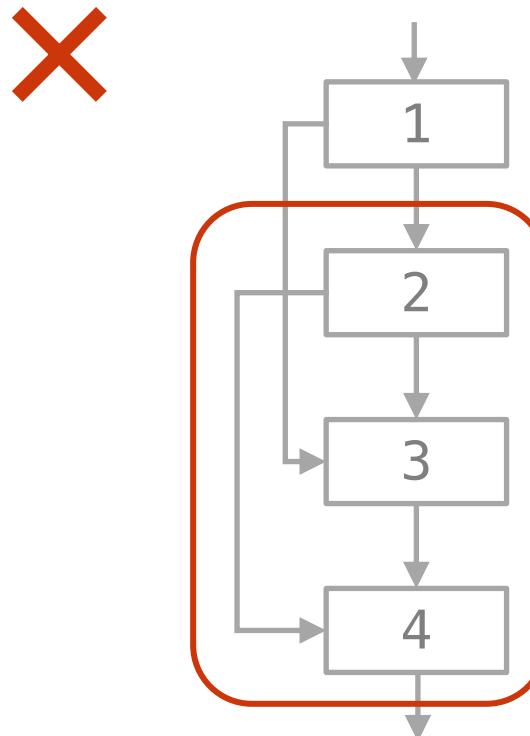
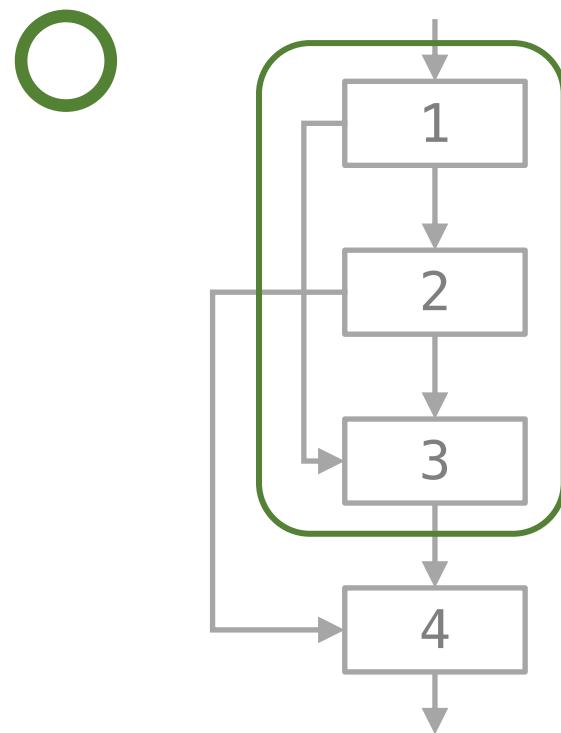
Flow graph の entry から n に到達するパスはすべて h を通る

Regions

1. Header $h \in N$ が存在し, 領域上のすべての頂点を支配する

$$\forall n \in N, h \text{ dom } n$$

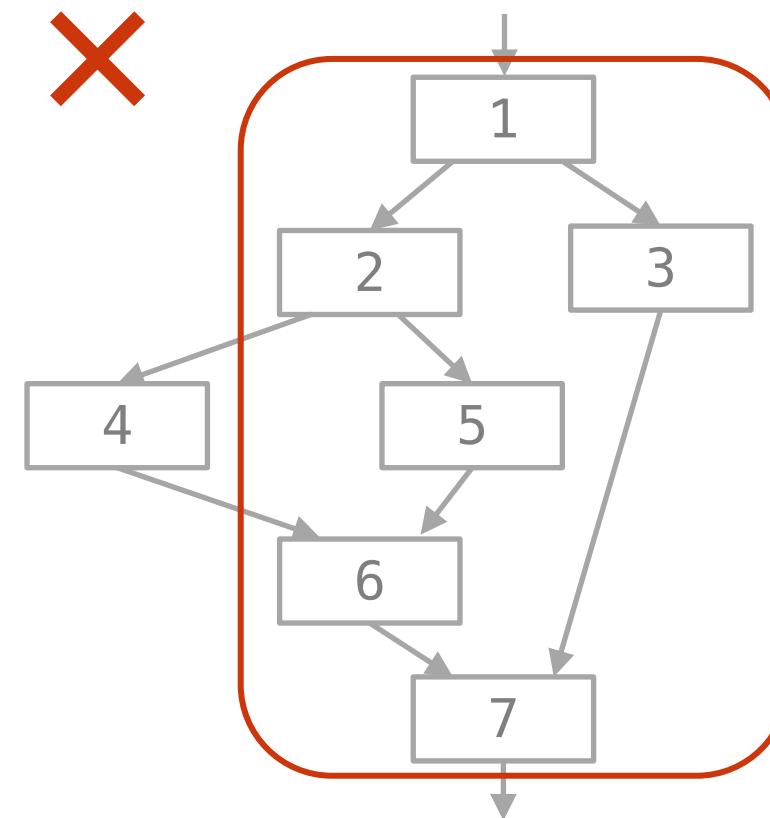
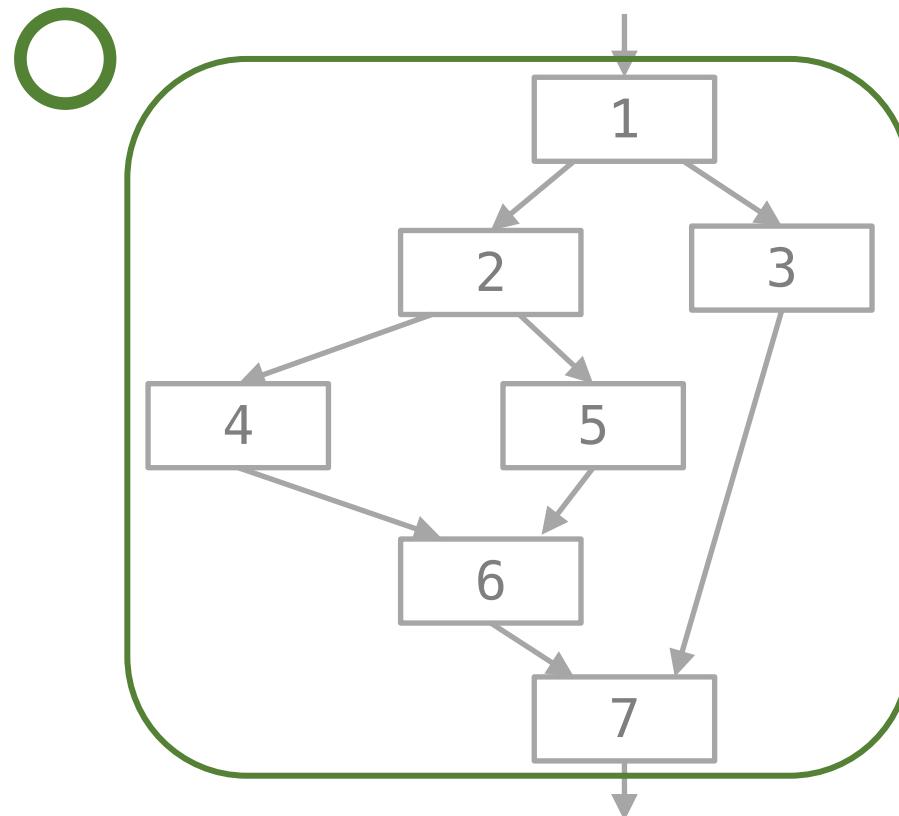
(= 領域の entry node はただ一つ存在する)



2 は 3 を支配しない
(3 には 2 を通らずに
到達できる)

Regions

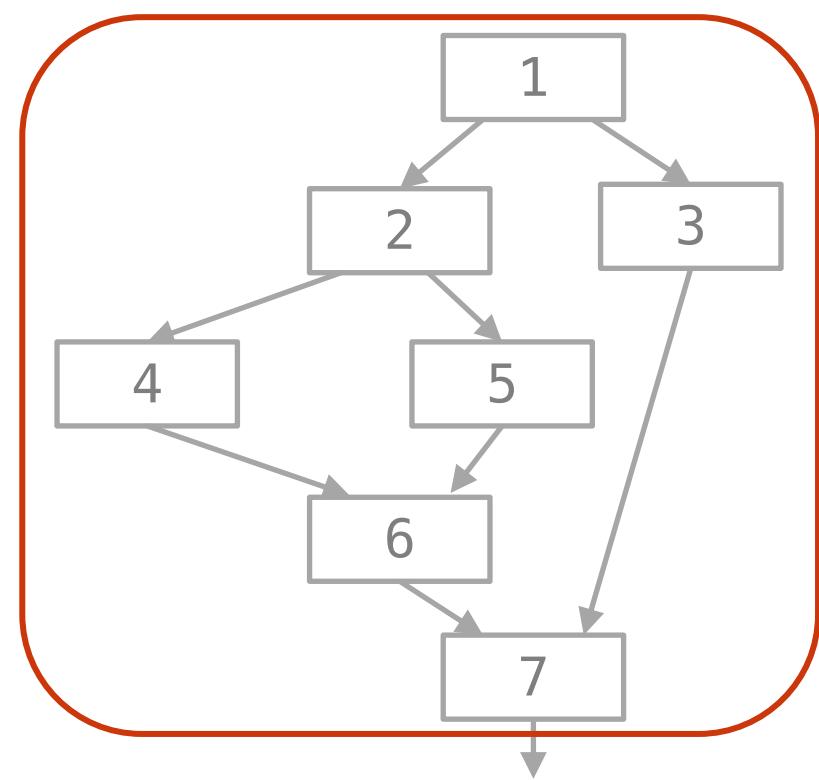
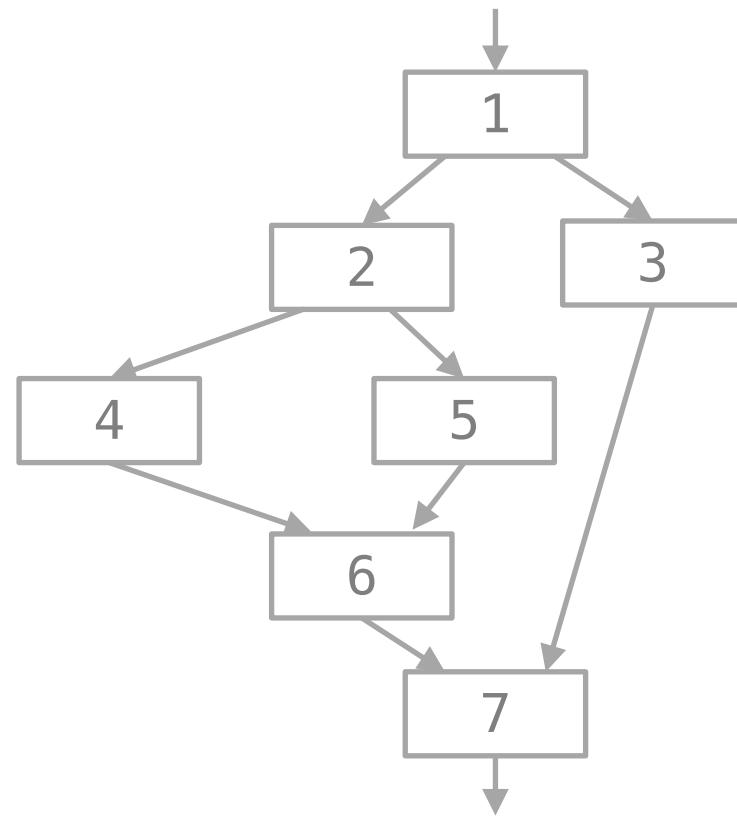
2. 頂点 m から h を通らずに $n \in N$ に到達できれば
 m は N に含まれる



header は 1
4 は 1 を通らずに
6, 7 に到達できる

Regions

3. E は $n_1, n_2 \in N$ 間の flow graph 上のすべての辺を含む
(ただし, h に入る辺は除く)



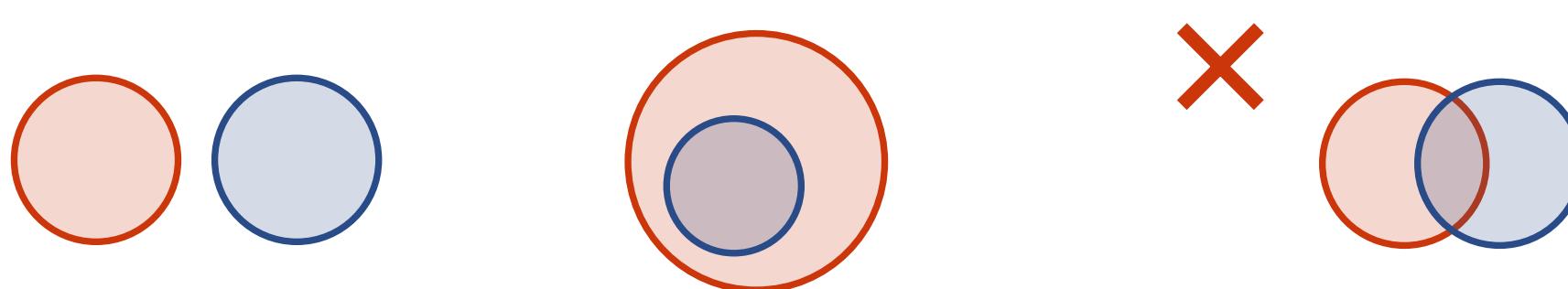
Region Hierarchies

これ以降の仮定 : flow graph が還元可能 (reducible)

- ・自然ループ (natural loop) は一つの領域

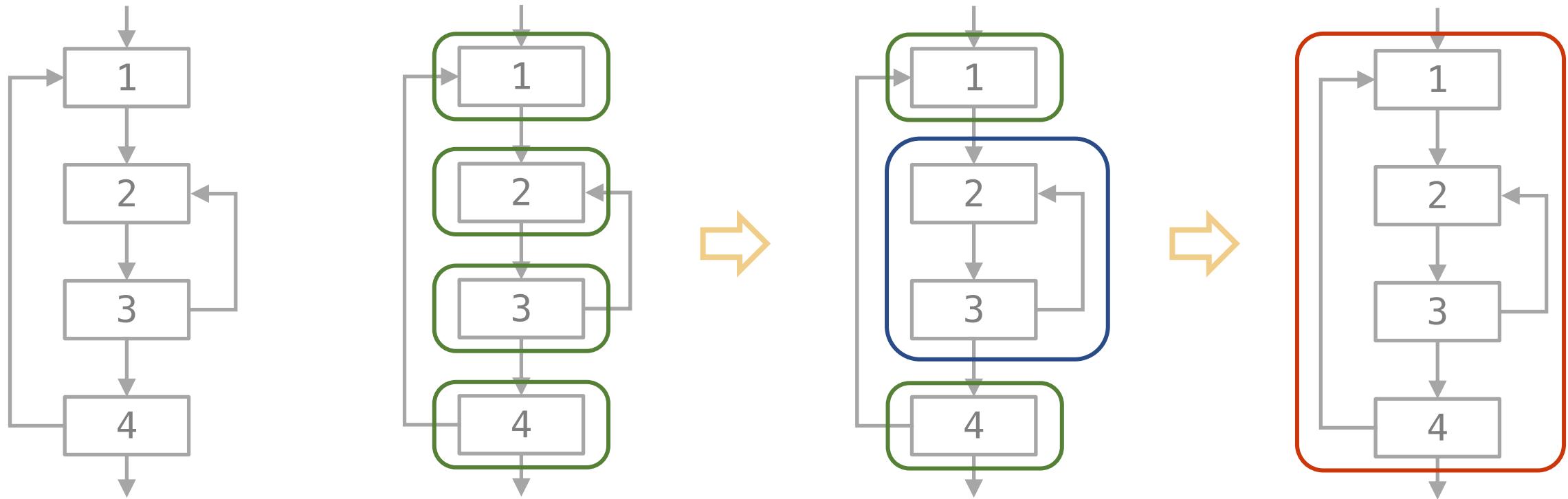
- ・ただ一つの header が存在
- ・Header への back edge が存在
- ・Back edge $n \rightarrow d$ が与えられたとき, d と d を通らずに n に到達できる頂点の集合

- ・グラフが還元可能なとき, 2つの自然ループは 別物 or 一方が他方に含まれる



Region Hierarchies

- ・自然ループを一つの領域として、階層構造を組み立てる
- ・初めは各 basic block を一つの領域として（葉要素, leaf region ），内側 → 外側のループに領域を広げていく



Region Hierarchies

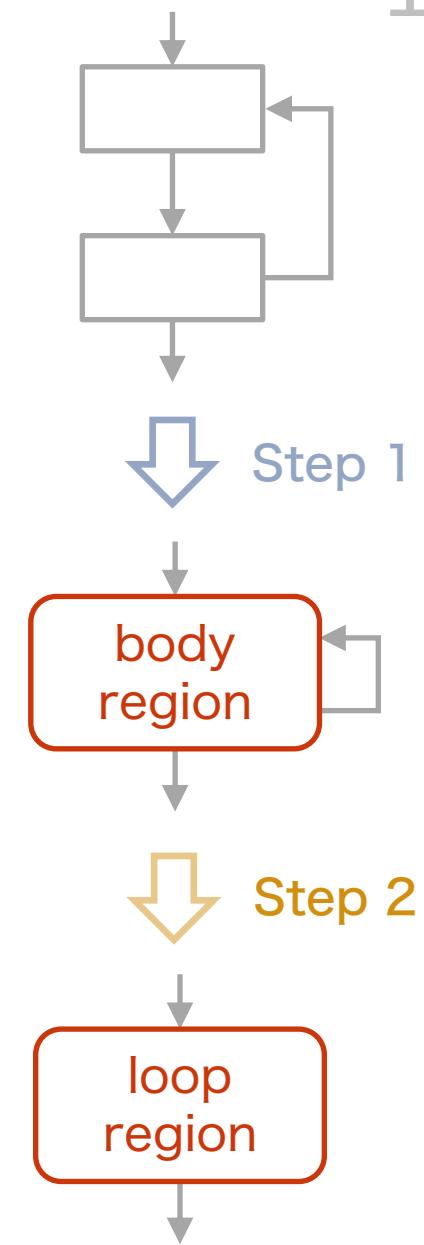
一つのループを一つの頂点で置き換える 2 ステップ

1. 本体領域 (body region) への置き換え

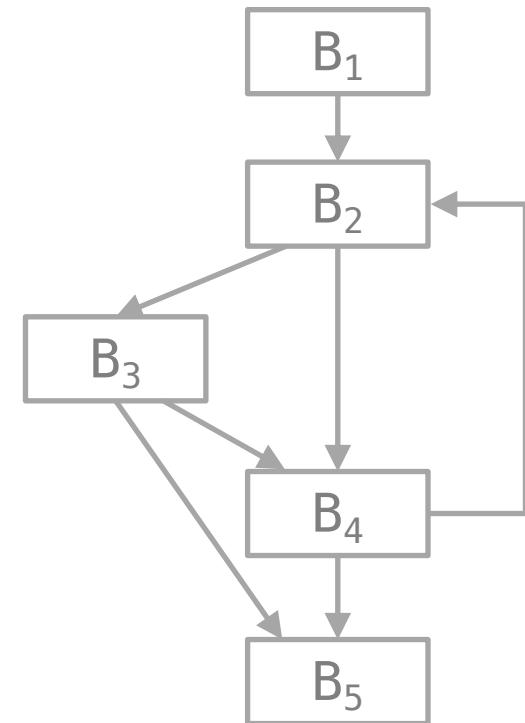
ループ L の本体 (body) (header への back edge を除いたもの) を一つの領域に置き換える

2. ループ領域 (loop region) への置き換え

本体領域からそれ自身への辺を取り除く

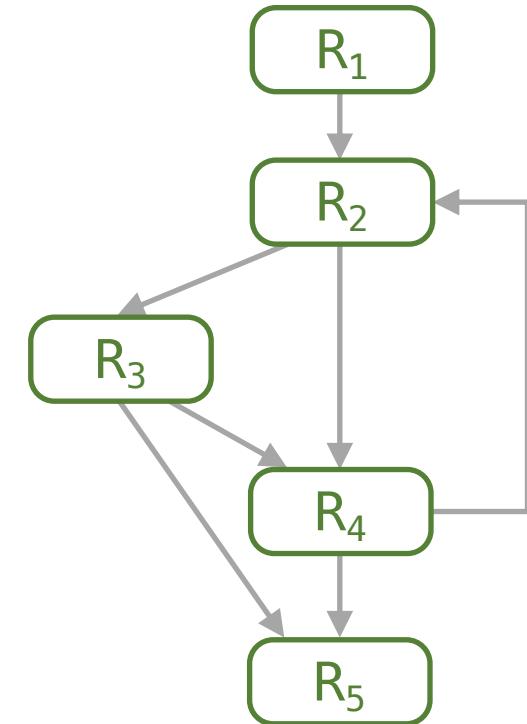


Example of Region Hierarchies



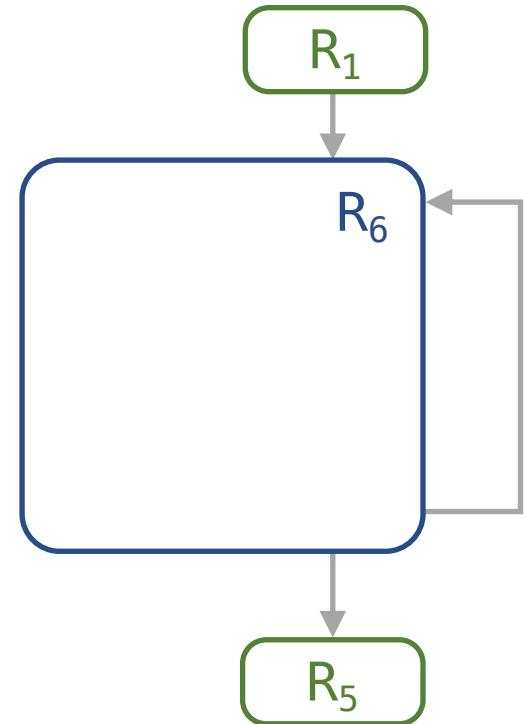
Example of Region Hierarchies

1. Basic block B_1, \dots, B_5 を表す葉領域 R_1, \dots, R_5



Example of Region Hierarchies

1. Basic block B_1, \dots, B_5 を表す葉領域 R_1, \dots, R_5
2. 領域 R_2, R_3, R_4 を含む本体領域 R_6



Example of Region Hierarchies

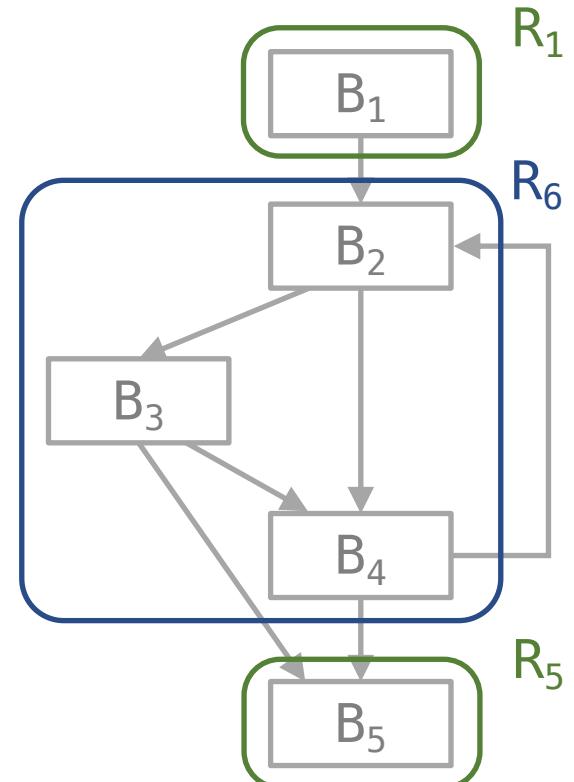
1. Basic block B_1, \dots, B_5 を表す葉領域 R_1, \dots, R_5

2. 領域 R_2, R_3, R_4 を含む本体領域 R_6

Transfer function の伝播に必要なので

$$B_3 \rightarrow B_5, B_4 \rightarrow B_5, B_4 \rightarrow B_2$$

の辺があったことを記録しておく



Example of Region Hierarchies

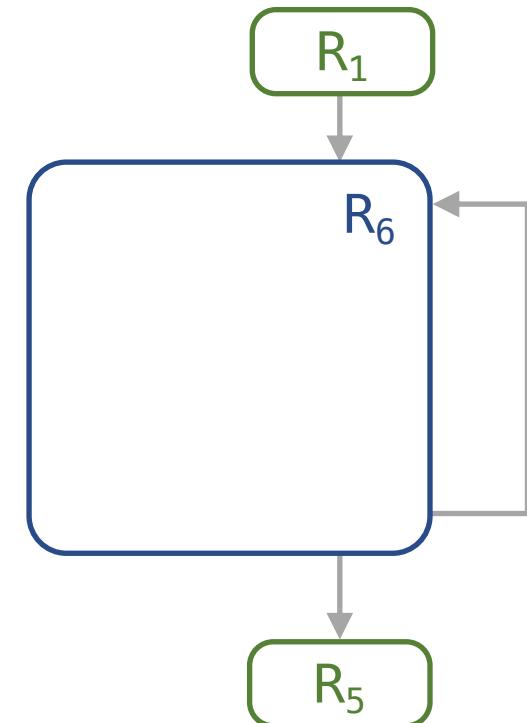
1. Basic block B_1, \dots, B_5 を表す葉領域 R_1, \dots, R_5

2. 領域 R_2, R_3, R_4 を含む本体領域 R_6

Transfer function の伝播に必要なので

$$B_3 \rightarrow B_5, B_4 \rightarrow B_5, B_4 \rightarrow B_2$$

の辺があったことを記録しておく



Example of Region Hierarchies

1. Basic block B_1, \dots, B_5 を表す葉領域 R_1, \dots, R_5

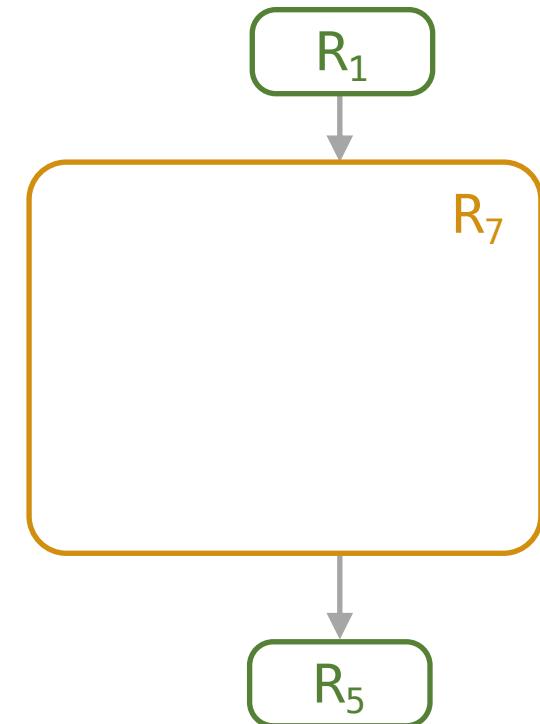
2. 領域 R_2, R_3, R_4 を含む本体領域 R_6

Transfer function の伝播に必要なので

$$B_3 \rightarrow B_5, B_4 \rightarrow B_5, B_4 \rightarrow B_2$$

の辺があったことを記録しておく

3. 一つの自然ループを表すループ領域 R_7



Example of Region Hierarchies

1. Basic block B_1, \dots, B_5 を表す葉領域 R_1, \dots, R_5

2. 領域 R_2, R_3, R_4 を含む本体領域 R_6

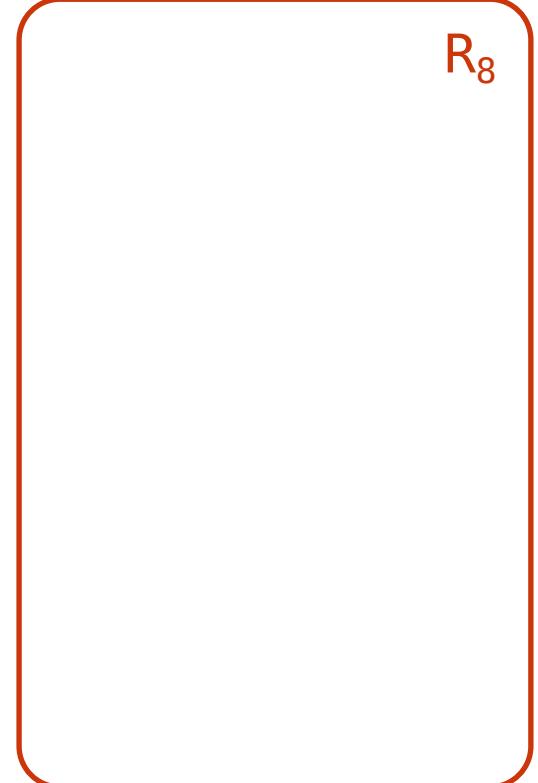
Transfer function の伝播に必要なので

$$B_3 \rightarrow B_5, B_4 \rightarrow B_5, B_4 \rightarrow B_2$$

の辺があったことを記録しておく

3. 一つの自然ループを表すループ領域 R_7

4. 最上位の本体領域 R_8



R_8

Example of Region Hierarchies

1. Basic block B_1, \dots, B_5 を表す葉領域 R_1, \dots, R_5

2. 領域 R_2, R_3, R_4 を含む本体領域 R_6

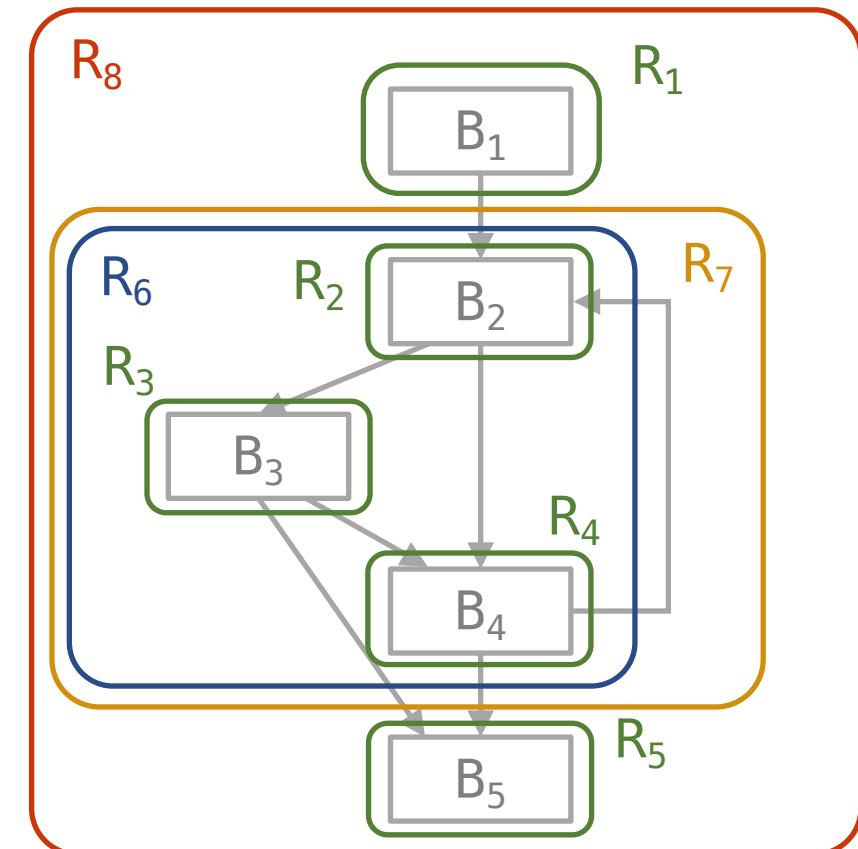
Transfer function の伝播に必要なので

$$B_3 \rightarrow B_5, B_4 \rightarrow B_5, B_4 \rightarrow B_2$$

の辺があったことを記録しておく

3. 一つの自然ループを表すループ領域 R_7

4. 最上位の本体領域 R_8

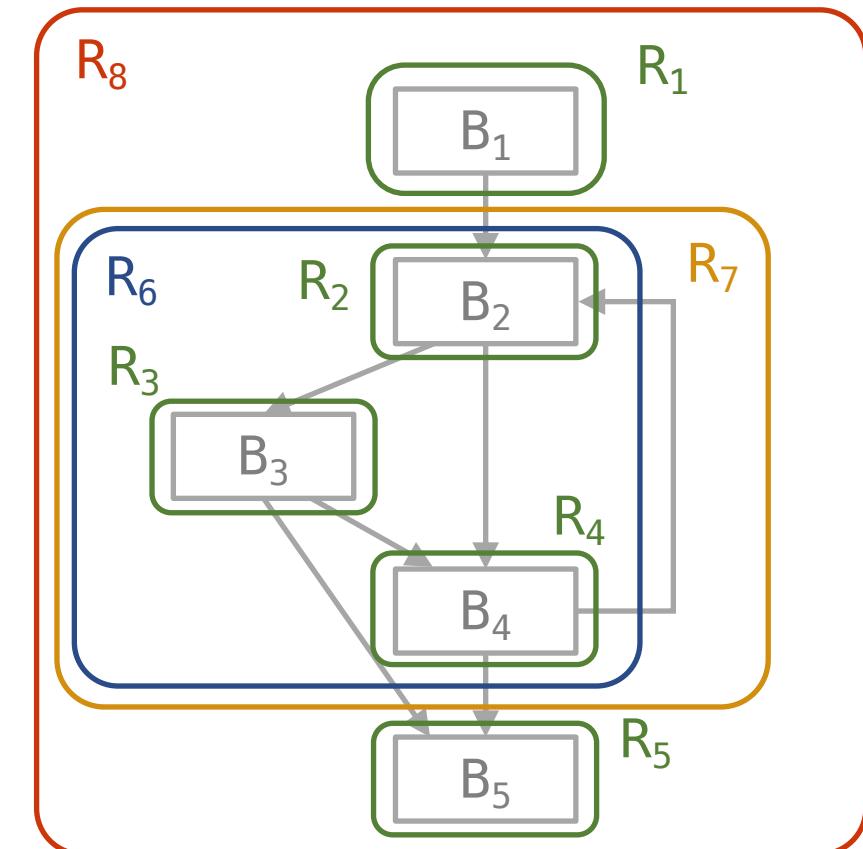


An Algorithm of Constructing Regions

Input : Reducible flow graph

Output : 領域の上昇列 $\{R_1, \dots, R_n\}$

1. ブロックに対応する葉領域のリストを構築
2. ループの内側が先に処理されるようにループを選択しながら, 本体領域, ループ領域の順でリストに加える
3. Flow graph 全体がループでなければ
flow graph 全体の本体領域をリストに加える



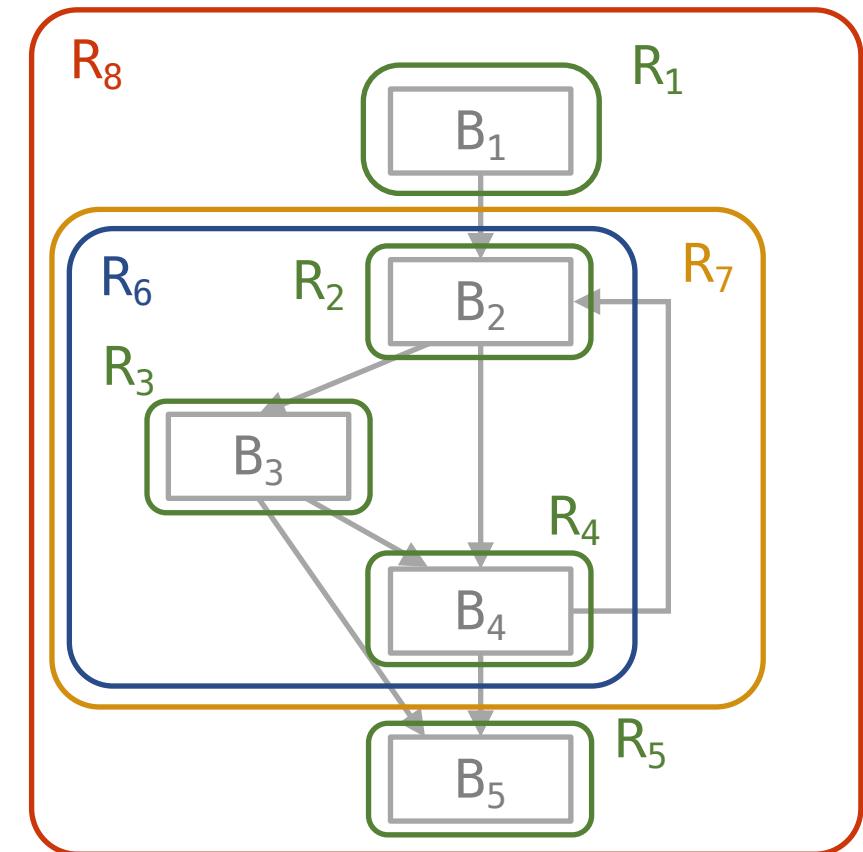
$R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8$

Contents

1. Regions
2. Overview of Region-Based Analysis
3. Transfer Functions
4. An Algorithm for Region-Based Analysis
5. Handling Nonreducible Flow Graphs

Overview of a Region-Based Analysis

1. Flow graph の領域階層を求める



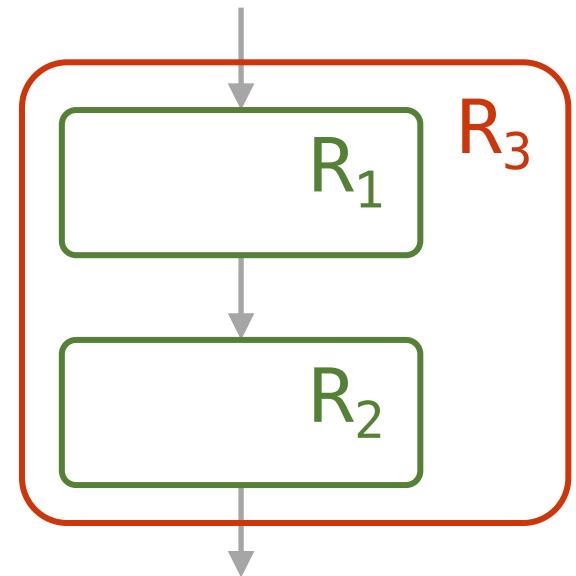
Overview of a Region-Based Analysis

1. Flow graph の領域階層を求める
2. 領域の entry からそれが含む領域の entry までの transfer function を求める

ex.) R_3 の entry から R_2 の entry までの transfer function

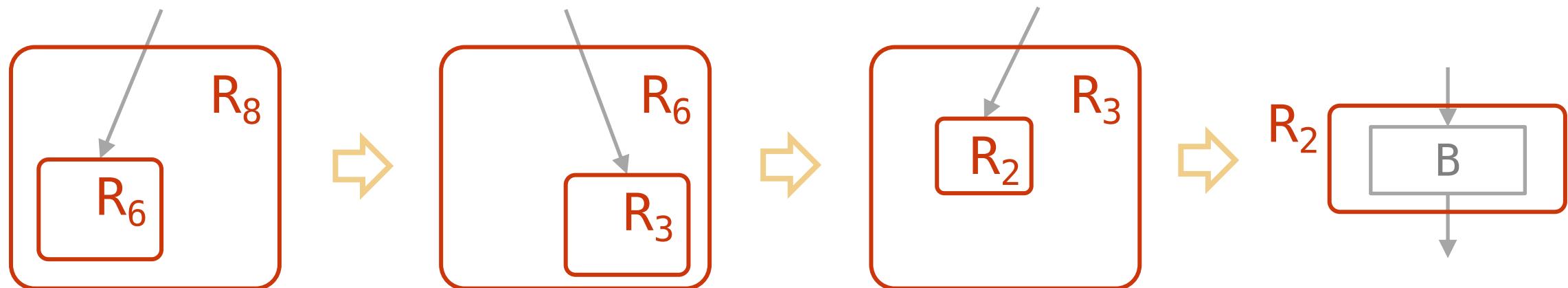
R_3 の entry から R_1 の最後までの transfer function を知る必要がある

→ 領域の小 → 大に向かって transfer function を求める



Overview of a Region-Based Analysis

1. Flow graph の領域階層を求める
2. 領域の entry からそれが含む領域の entry までの transfer function を求める
3. 各領域の entry における data-flow values を計算
→ 領域の大 → 小の順で計算すると、最終的に basic block の data-flow values が求められる



Contents

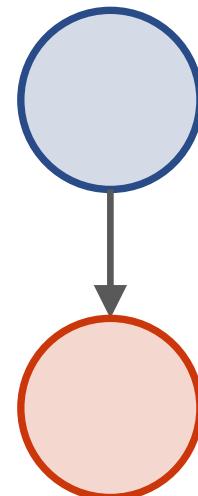
1. Regions
2. Overview of Region-Based Analysis
- 3. Transfer Functions**
4. An Algorithm for Region-Based Analysis
5. Handling Nonreducible Flow Graphs

Transfer Function

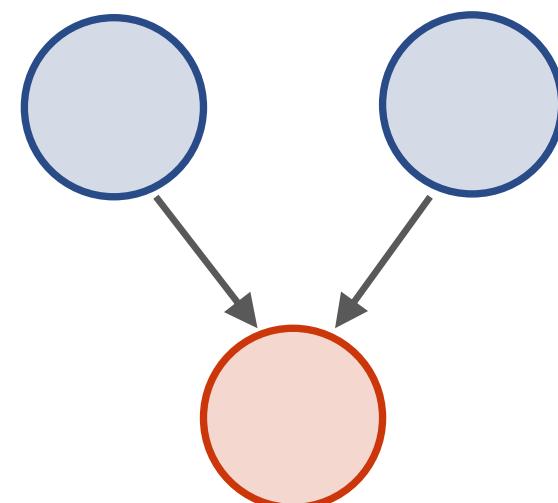
アルゴリズムでは transfer function をまとめた操作が必要

Transfer function に対して 3 つの操作を考える

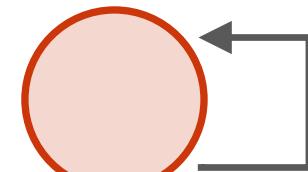
- Composition



- Meet



- Closure

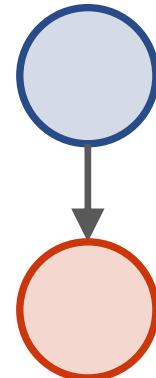


Composition on Transfer Function

頂点 n_1, n_2 の transfer function を f_1, f_2 とすると,

$n_1 \rightarrow n_2$ を実行したときの transfer function は

$f_2 \circ f_1$ で表される



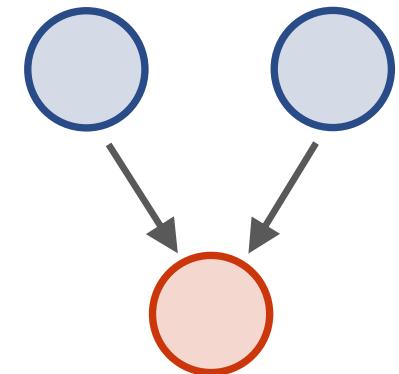
Example : reaching definitions

$$\begin{aligned}
 f_2 \circ f_1(x) &= \text{gen}_2 \cup \left((\text{gen}_1 \cup (x - \text{kill}_1)) - \text{kill}_2 \right) \\
 &= (\text{gen}_2 \cup (\text{gen}_1 - \text{kill}_2)) \cup (x - (\text{kill}_1 \cup \text{kill}_2))
 \end{aligned}$$

→ $\text{gen} \cup (x - \text{kill})$ の形式

Meet on Transfer Function

Transfer function 自体を meet-semilattice の値として考える



Meet operator \wedge_f :

$$(f_1 \wedge_f f_2)(x) = f_1(x) \wedge f_2(x)$$

(\wedge は data-flow value の meet operator)

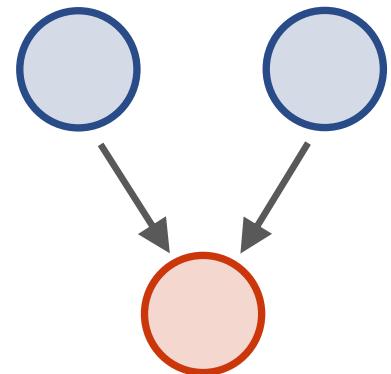
これ以降、明確な場合はどちらの meet operator も \wedge で表す

Meet on Transfer Function

Meet operator \wedge_f :

$$(f_1 \wedge_f f_2)(x) = f_1(x) \wedge f_2(x)$$

(\wedge は data-flow value の meet operator)



Example : reaching definitions

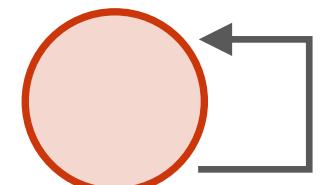
$$\begin{aligned}
 (f_1 \wedge f_2)(x) &= f_1(x) \wedge f_2(x) \\
 &= (gen_1 \cup (x - kill_1)) \cup (gen_2 \cup (x - kill_2)) \\
 &= (gen_1 \cup gen_2) \cup (x - (kill_1 \cap kill_2))
 \end{aligned}$$

→ $gen \cup (x - kill)$ の形式

Closure on Transfer Function

f : ループ 1 周分の transfer function

f^n : ループ n 周分の transfer function



ループが何回実行されるか分からない → 0 回以上実行されると仮定する

f^* : ループの transfer function (f の closure)

$$f^* = \bigwedge_{n \geq 0} f^n \quad (= \boxed{f^0} \wedge f^1 \wedge f^2 \wedge f^3 \wedge \cdots)$$

$$= \boxed{I} \wedge \left(\bigwedge_{n > 0} f^n \right) \quad I \text{ は恒等関数}$$

Closure on Transfer Function

Example : reaching definitions

$$f^1(x) = \text{gen} \cup (x - \text{kill})$$

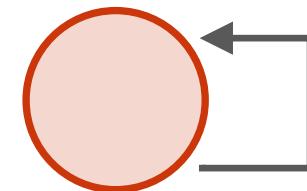
$$f^k(x) = \text{gen} \cup (x - \text{kill}) \text{ と仮定}$$

$$f^{k+1}(x) = f(f^k(x))$$

$$= \text{gen} \cup \left((\text{gen} \cup (x - \text{kill})) - \text{kill} \right)$$

$$= \text{gen} \cup (x - \text{kill})$$

$$\rightarrow \text{任意の } f^n(x) = \text{gen} \cup (x - \text{kill})$$



Closure on Transfer Function

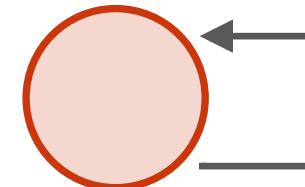
Example : reaching definitions

$$f^*(x) = I \wedge f^1(x) \wedge f^2(x) \wedge \dots$$

$$= x \cup (gen \cup (x - kill))$$

$$= gen \cup x$$

→ $kill = \emptyset$ として $gen \cup (x - kill)$ の形式

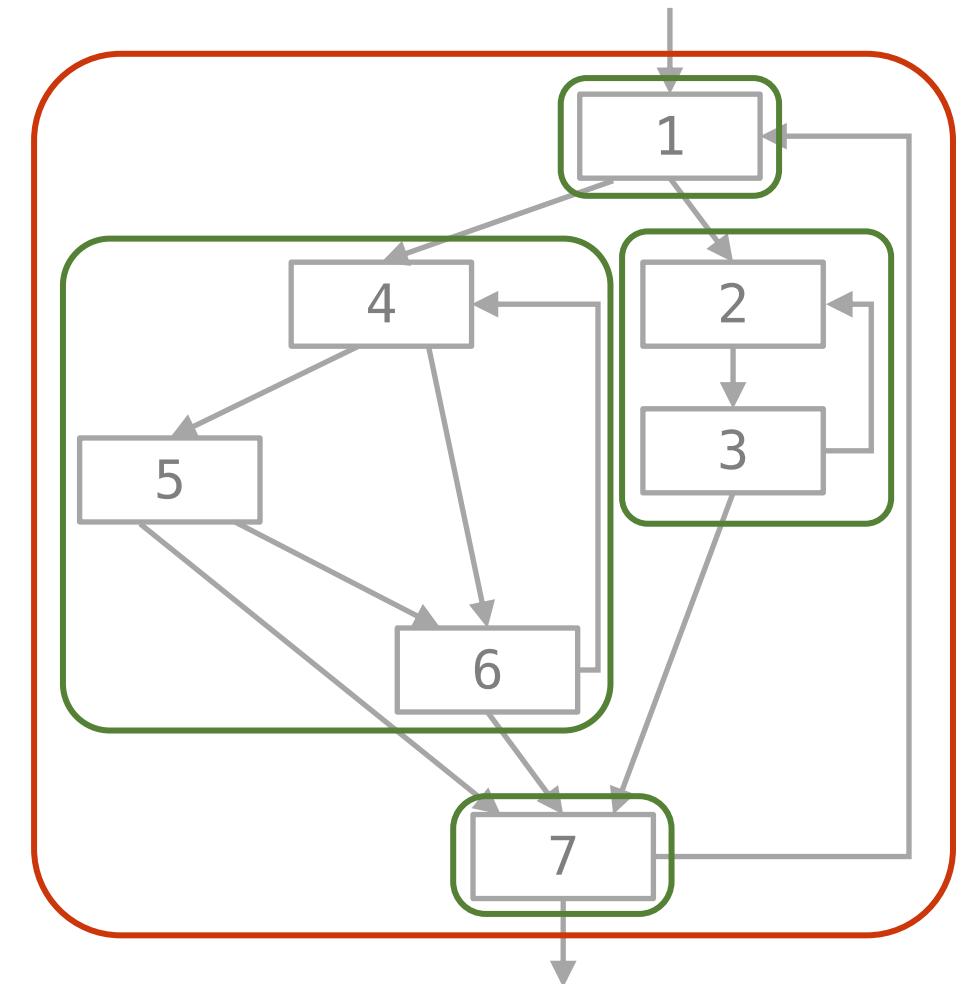


Contents

1. Regions
2. Overview of Region-Based Analysis
3. Transfer Functions
- 4. An Algorithm for Region-Based Analysis**
5. Handling Nonreducible Flow Graphs

An Algorithm for Region-Based Analysis

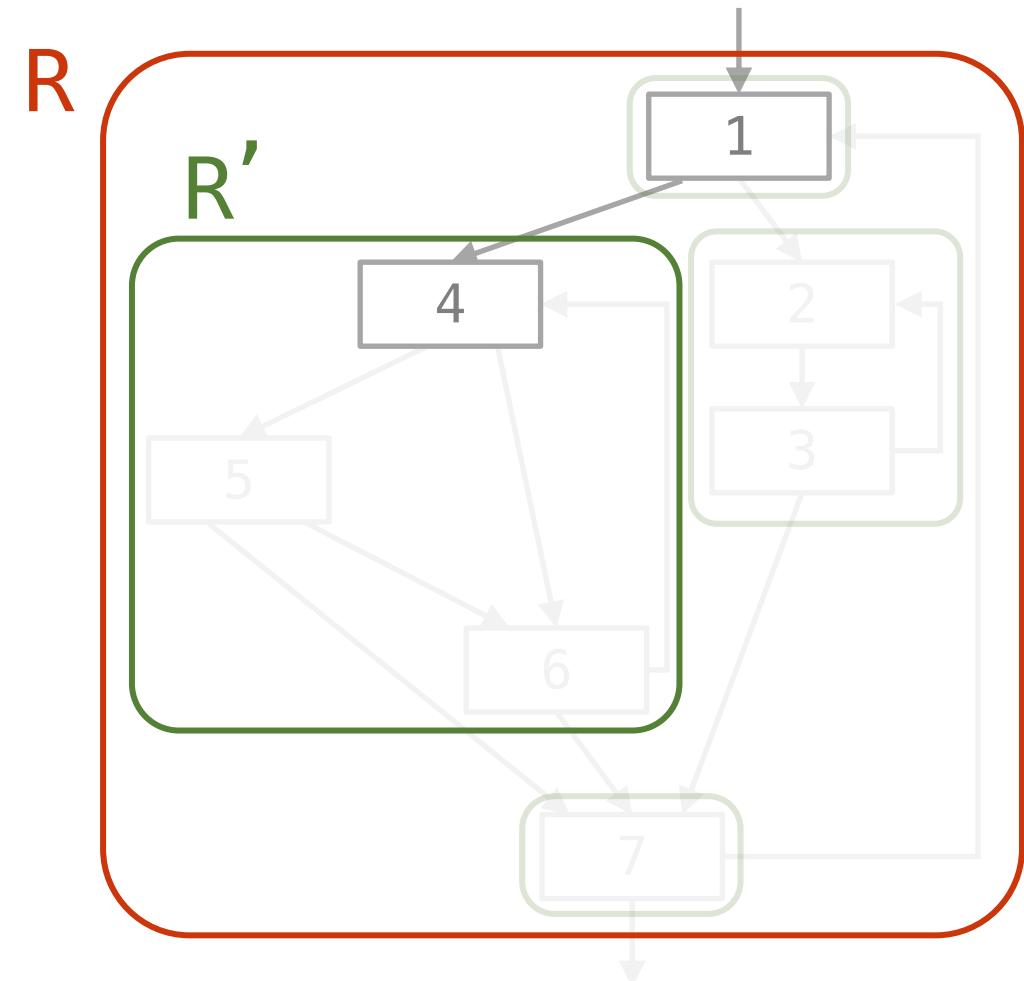
今まで見てきた枠組みを使って前向きのデータフロー解析を行う
アルゴリズムを考える



An Algorithm for Region-Based Analysis

今まで見てきた枠組みを使って前向きのデータフロー解析を行うアルゴリズムを考える

$f_{R,\text{IN}}[R']$: 領域 R の entry からそれが含む領域 R' の entry までの transfer function

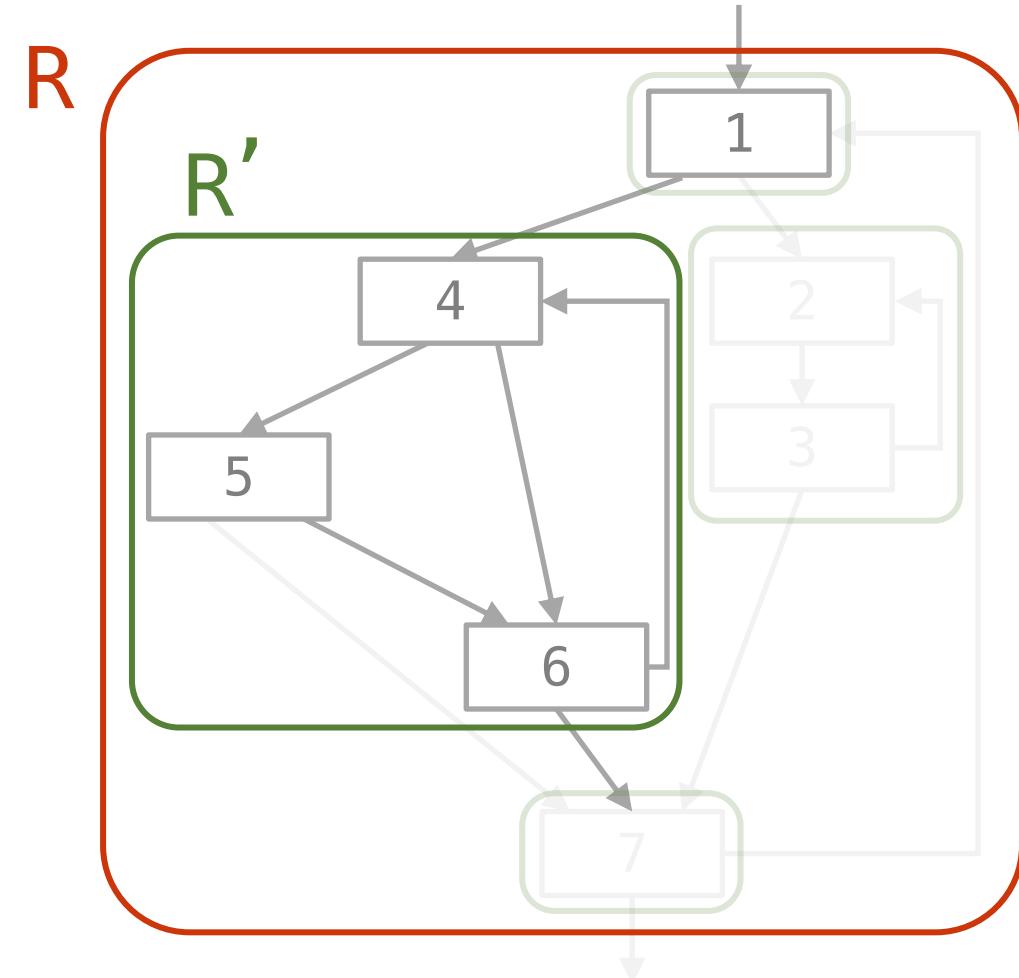


An Algorithm for Region-Based Analysis

今まで見てきた枠組みを使って前向きのデータフロー解析を行うアルゴリズムを考える

$f_{R,\text{IN}}[R']$: 領域 R の entry からそれが含む領域 R' の entry までの transfer function

$f_{R,\text{OUT}}[B]$: 領域 R の entry から exit block B までの transfer function



An Algorithm for Region-Based Analysis

アルゴリズム

入力 : Data-flow の枠組みと reducible flow graph

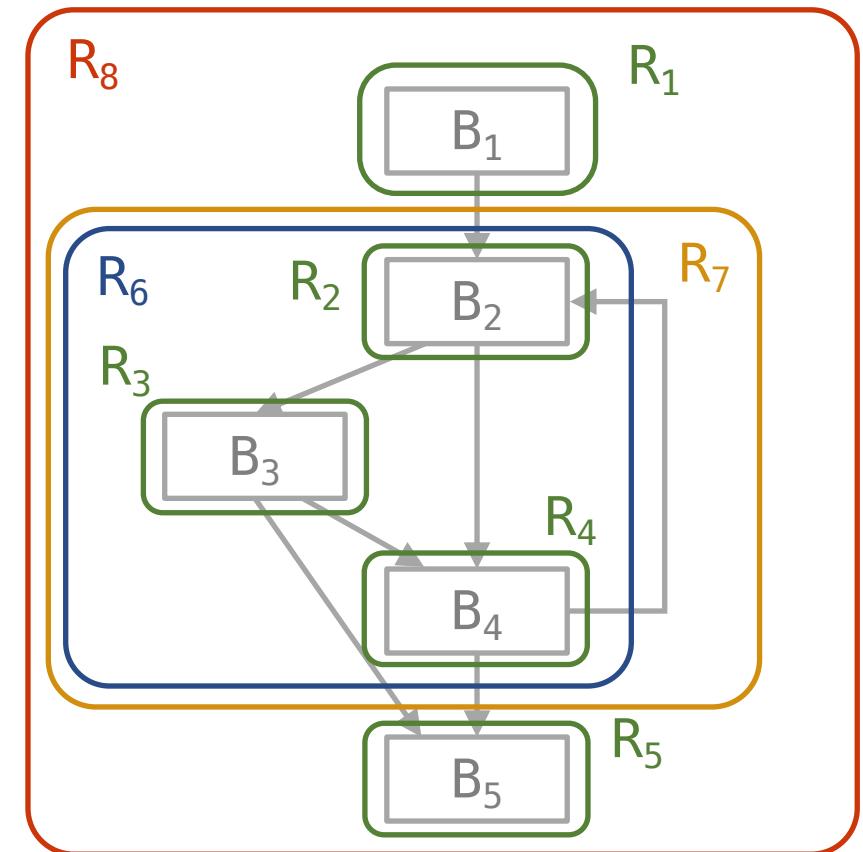
出力 : 各ブロック B の data-flow values $\text{IN}[B]$

1. Flow graph 上の領域を求める
2. 領域の小 → 大に向かって transfer function を求める
3. 領域の大 → 小に向かって 各領域の先頭における
data-flow values を求める

An Algorithm for Region-Based Analysis

1. Flow graph 上の領域を求める

領域の上昇列 R_1, R_2, \dots, R_n を求める



$R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8$

An Algorithm for Region-Based Analysis

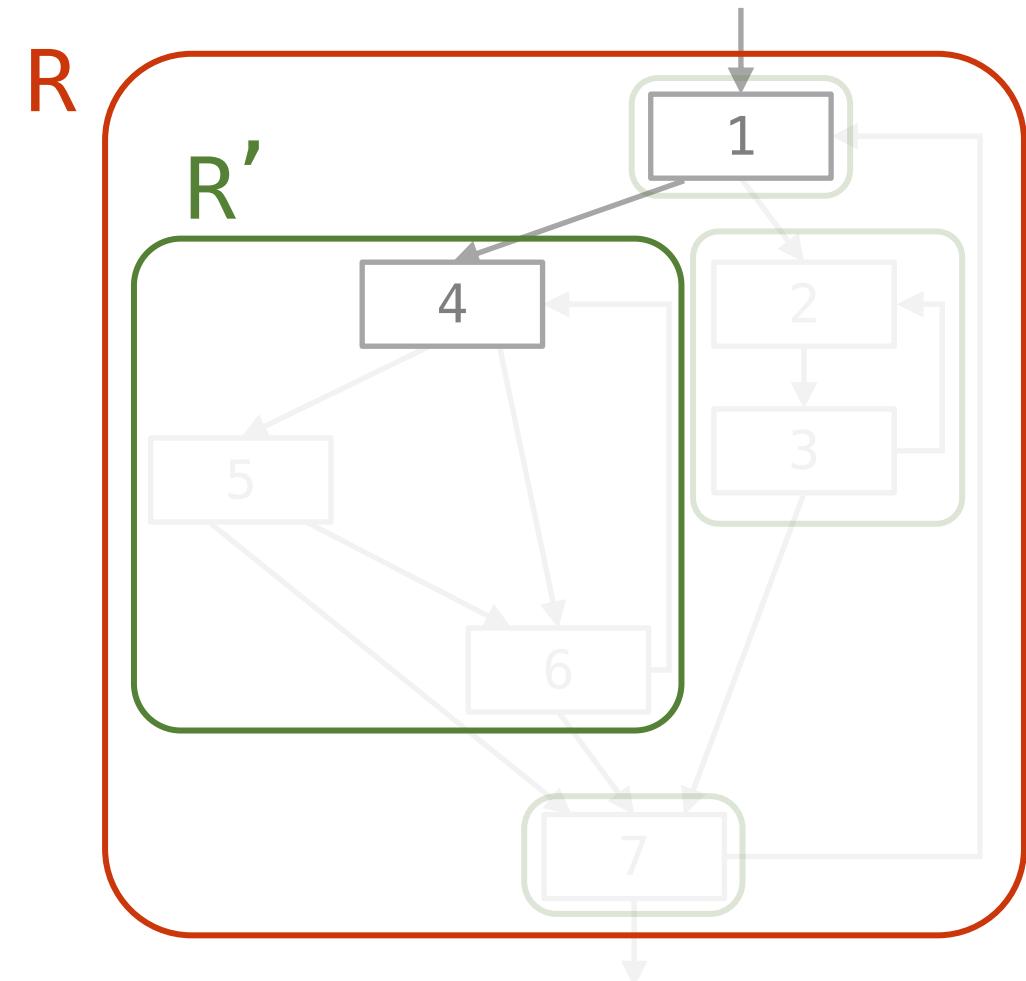
2. 領域の小 → 大に向かって transfer function を求める

求めたいもの :

$f_{R,\text{IN}}[R']$: 領域 R の entry からそれが含む領域 R' の entry までの transfer function

$f_{R,\text{OUT}}[B]$: 領域 R の entry から exit block B までの transfer function

$f_{R,\text{OUT}}[B]$ は $f_{R,\text{IN}}[R']$ を計算するために必要



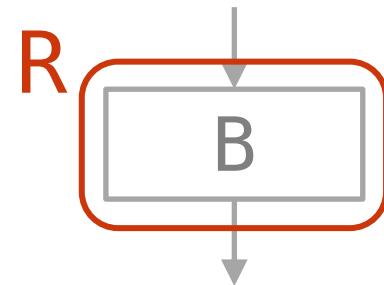
An Algorithm for Region-Based Analysis

2. 領域の小 → 大に向かって transfer function を求める

(a) R が B に対応する葉要素のとき

$$f_{R,\text{IN}[B]} = I$$

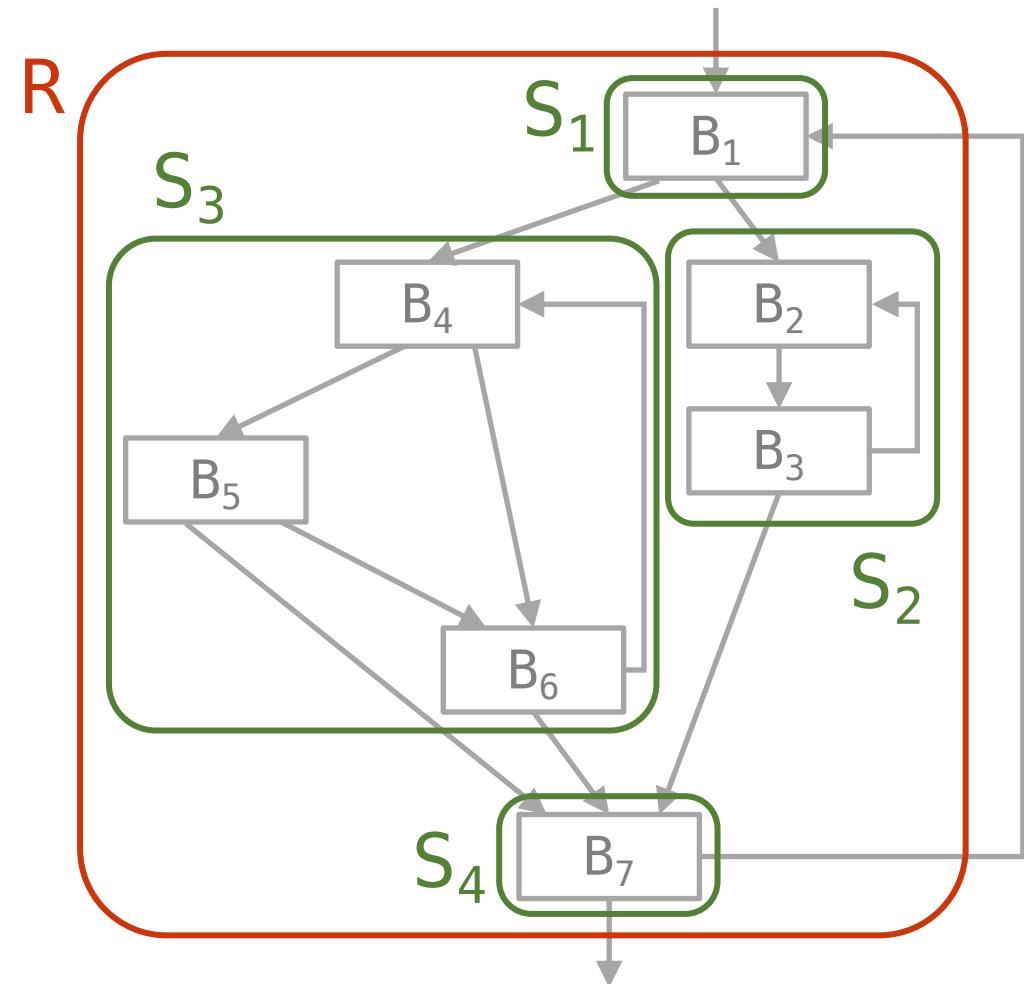
$$f_{R,\text{OUT}[B]} = f_B$$



An Algorithm for Region-Based Analysis

2. 領域の小 → 大に向かって transfer function を求める

(b) R が本体要素のとき



An Algorithm for Region-Based Analysis

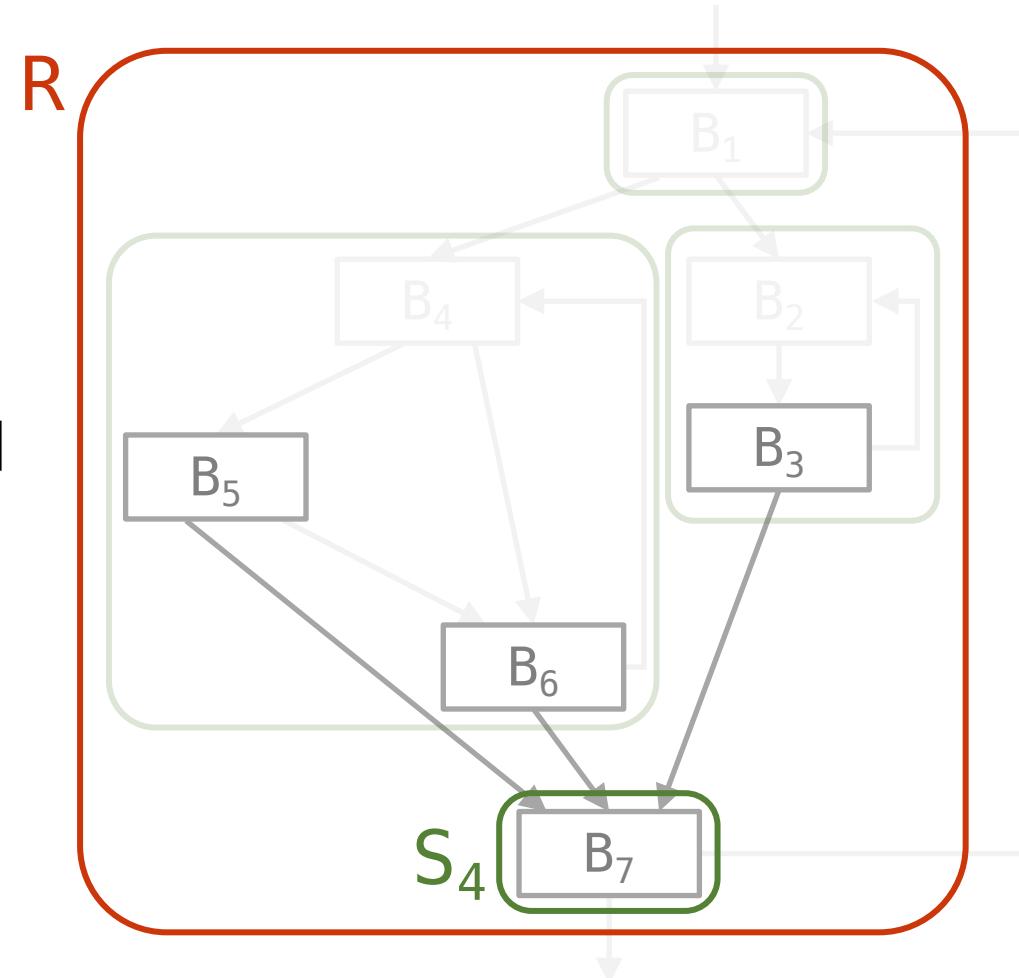
2. 領域の小 → 大に向かって transfer function を求める

(b) R が本体要素のとき

ex.)

- ・前の頂点の meet を取る

$$f_{R,\text{IN}[S_4]} = f_{R,\text{OUT}[B_3]} \wedge f_{R,\text{OUT}[B_5]} \wedge f_{R,\text{OUT}[B_6]}$$



An Algorithm for Region-Based Analysis

2. 領域の小 → 大に向かって transfer function を求める

(b) R が本体要素のとき

ex.)

- 前の頂点の meet を取る

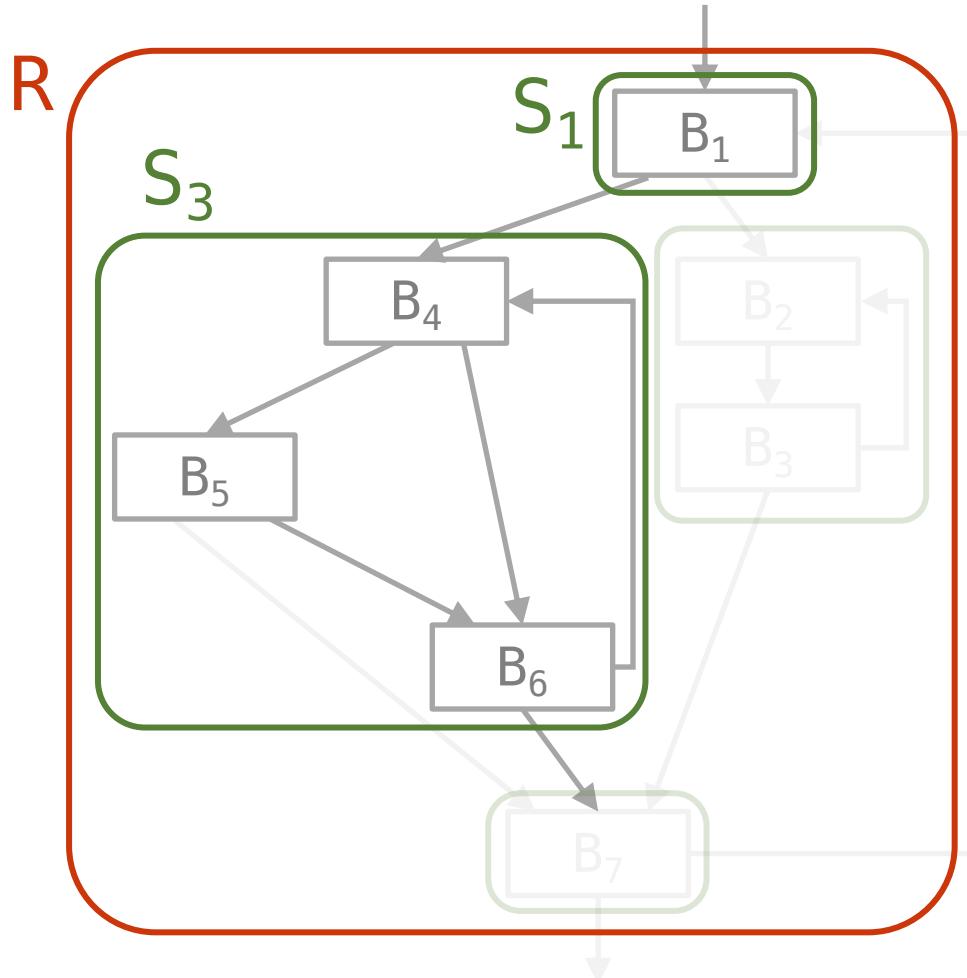
$$f_{R,\text{IN}}[S_4] = f_{R,\text{OUT}}[B_3] \wedge f_{R,\text{OUT}}[B_5] \wedge f_{R,\text{OUT}}[B_6]$$

- ある exit block までの transfer function

$$f_{R,\text{OUT}}[B_6] = \underline{f_{S_3,\text{OUT}}[B_6]} \circ \underline{f_{R,\text{IN}}[S_3]}$$

領域 S_3 から
exit block まで

R の entry から
領域 S_3 まで



An Algorithm for Region-Based Analysis

2. 領域の小 → 大に向かって transfer function を求める

(b) R が本体要素のとき

1 **for** (R の一つ下の階層の領域 S をトポロジカル順に) :

2 $f_{R,\text{IN}}[S] = \bigwedge_{(S \text{ の header の前の頂点かつ } R \text{ に含まれる } B)} f_{R,\text{OUT}}[B]$

3 **for** (each exit block B in S) :

4 $f_{R,\text{OUT}}[B] = f_{S,\text{OUT}}[B] \circ f_{R,\text{IN}}[S]$

トポロジカル順：どの頂点もその出力辺の指す頂点よりも早く探索

→ 本体要素 (back edge なし) なので可能

An Algorithm for Region-Based Analysis

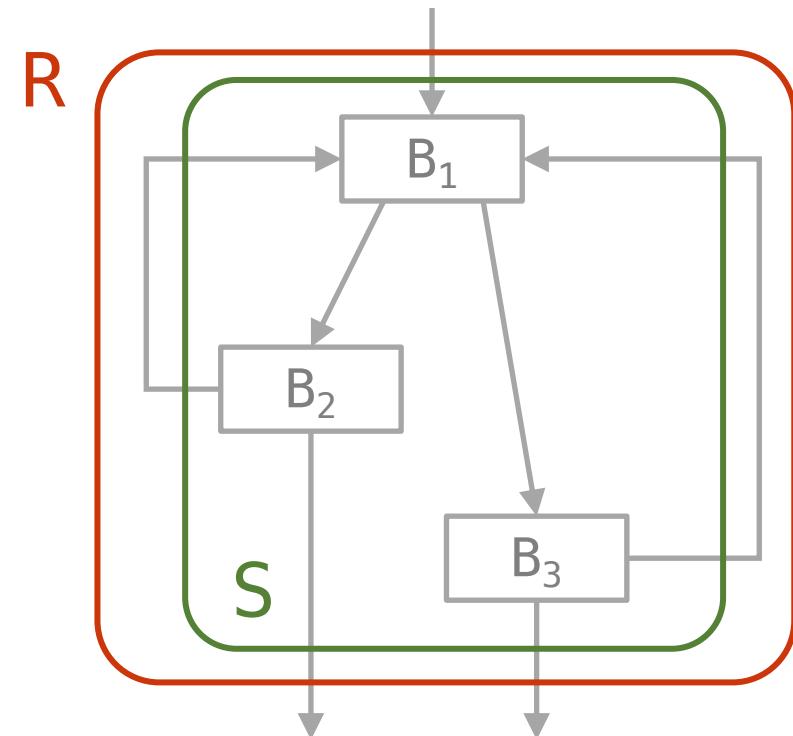
2. 領域の小 → 大に向かって transfer function を求める

(c) R がループ要素のとき

ex.)

- ループを無限回繰り返したときの効果

$$f_{R,\text{IN}[S]} = (f_{S,\text{OUT}[B_2]} \wedge f_{S,\text{OUT}[B_3]})^*$$



An Algorithm for Region-Based Analysis

2. 領域の小 → 大に向かって transfer function を求める

(c) R がループ要素のとき

1 let S は R の本体領域

2 $f_{R,IN[S]} = \left(\bigwedge_{(S \text{ の header の前の頂点かつ } R \text{ に含まれる } B)} f_{S,OUT[B]} \right)^*$

3 for (each exit block B in R) :

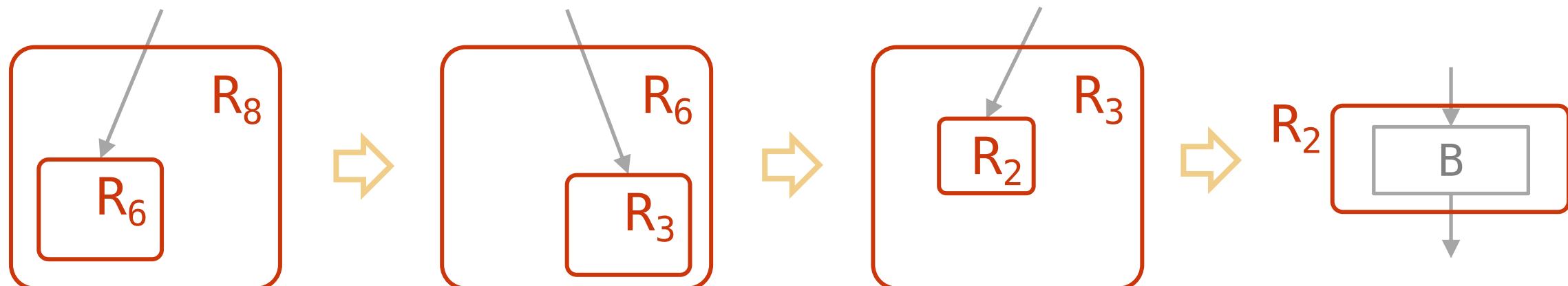
4 $f_{R,OUT[B]} = f_{S,OUT[B]} \circ f_{R,IN[S]}$

An Algorithm for Region-Based Analysis

3. 領域の大 → 小に向かって 各領域の先頭における
data-flow values を求める

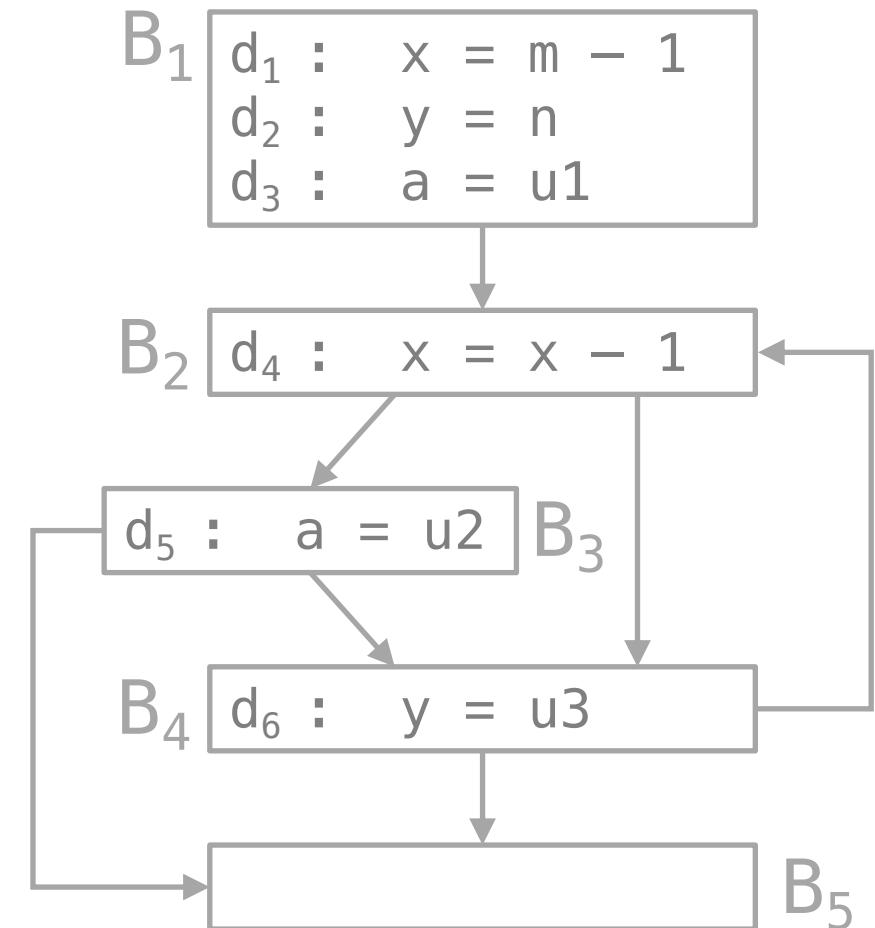
(a) 境界条件 : $\text{IN}[R_n] = \text{IN}[\text{ENTRY}]$

(b) $R \in \{R_1, \dots, R_{n-1}\}$ について領域の大きいものから順に
 $\text{IN}[R] = f_{R', \text{IN}[R]}(\text{IN}[R'])$ を求める (R' は R を直接囲む領域)



Example of the Algorithm

右の flow graph の reaching definition 解析を行う



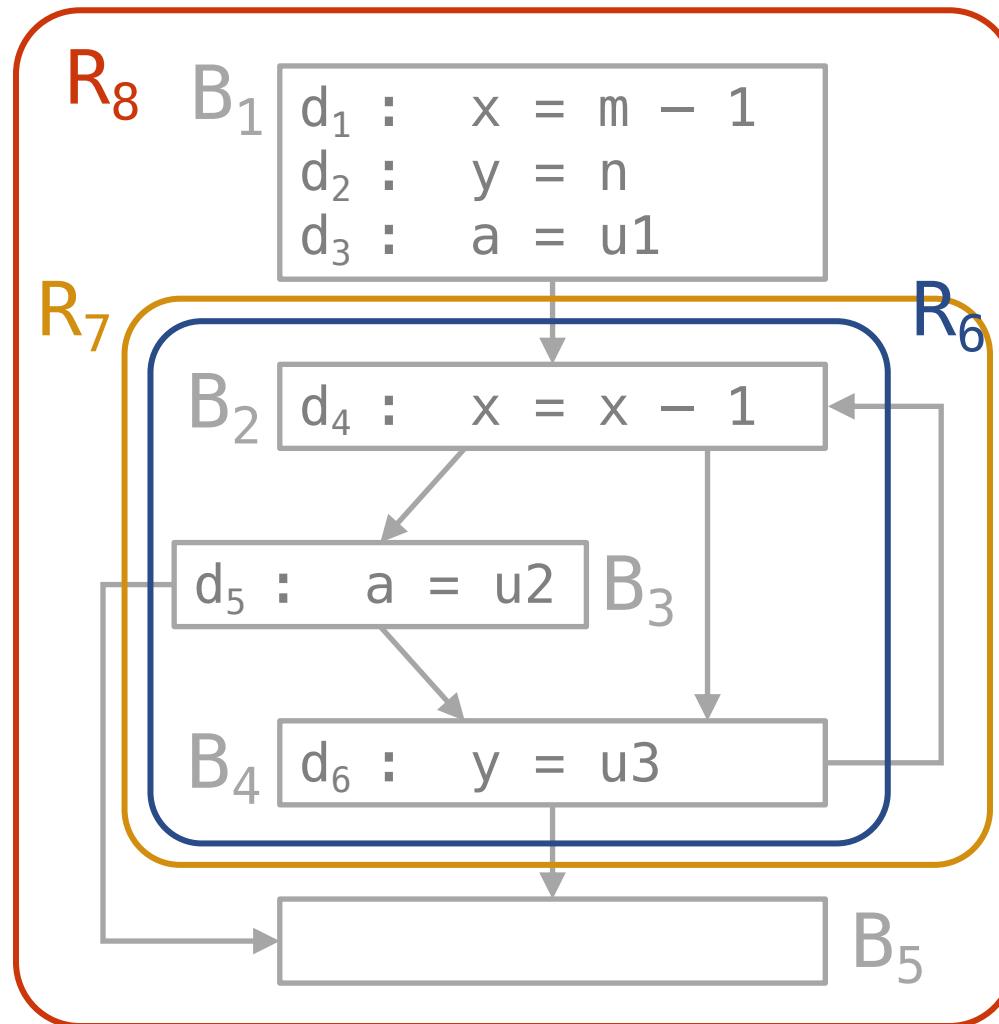
Example of the Algorithm

右の flow graph の reaching definition 解析を行う

1. 領域の上昇列を求める

$R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8$

R_i ($i = 1, 2, 3, 4, 5$) は B_i に対応する葉領域



Example of the Algorithm

2. 領域の小 → 大に向かって transfer function を求める

葉領域 :

$$f_{R_i, \text{IN}}[B_i] = I$$

$$f_{R_i, \text{OUT}}[B_i] = f_{B_i} \quad (i = 1, 2, 3, 4, 5)$$

領域 R_6 :

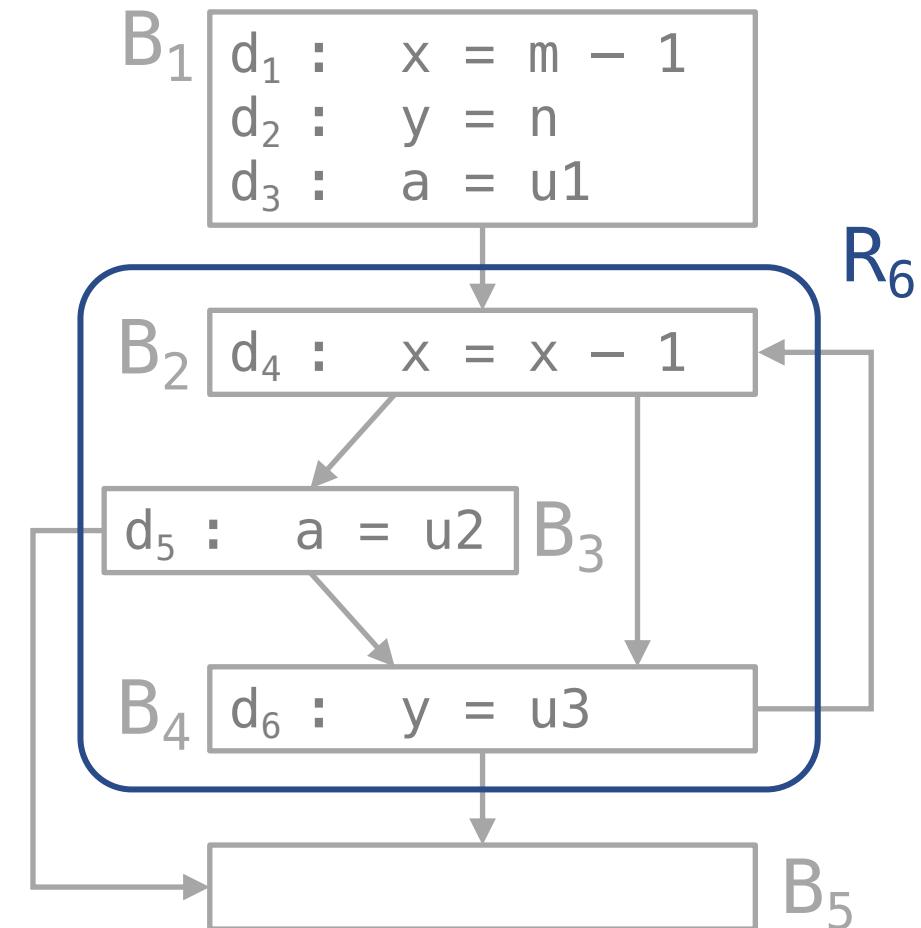
$$f_{R_6, \text{IN}}[R_2] = I$$

$$f_{R_6, \text{OUT}}[B_2] = f_{R_2, \text{OUT}}[B_2] \circ f_{R_6, \text{IN}}[R_2]$$

$$f_{R_6, \text{IN}}[R_3] = f_{R_6, \text{OUT}}[B_2]$$

$$f_{R_6, \text{OUT}}[B_3] = f_{R_3, \text{OUT}}[B_3] \circ f_{R_6, \text{IN}}[R_3]$$

Q $f_{R_6, \text{IN}}[R_4] =$



Example of the Algorithm

2. 領域の小 → 大に向かって transfer function を求める

葉領域 :

$$f_{R_i, \text{IN}}[B_i] = I$$

$$f_{R_i, \text{OUT}}[B_i] = f_{B_i} \quad (i = 1, 2, 3, 4, 5)$$

領域 R_6 :

$$f_{R_6, \text{IN}}[R_2] = I$$

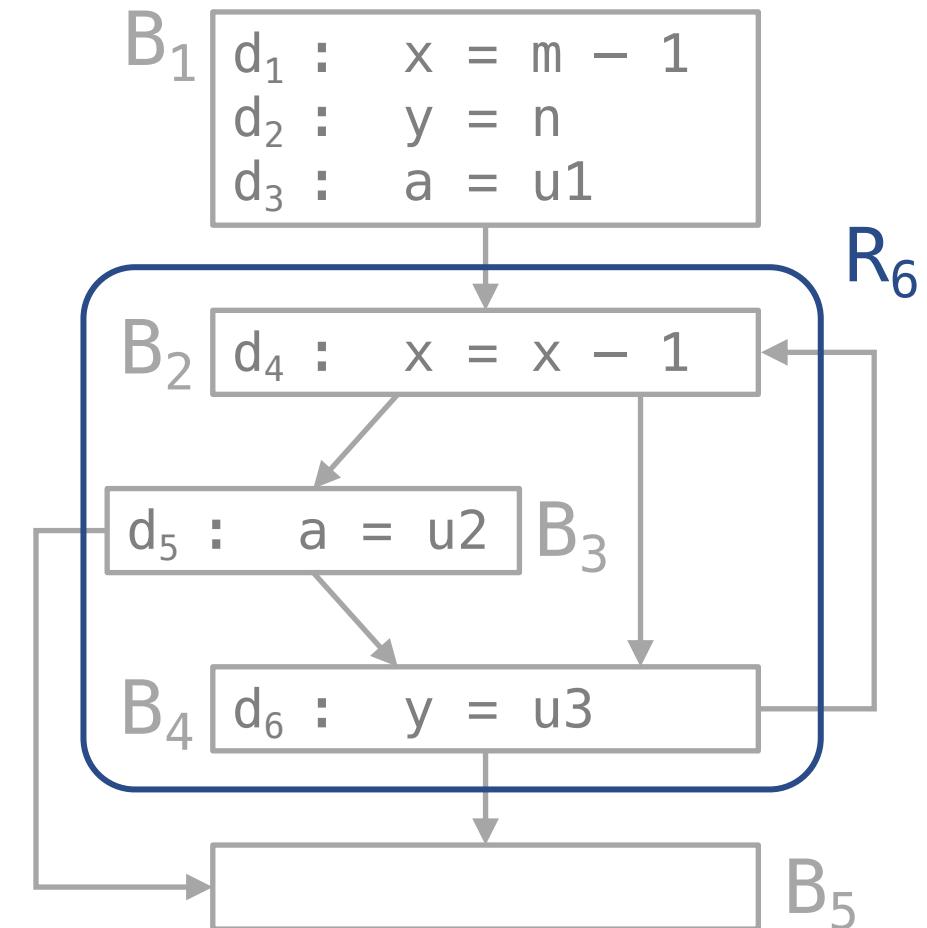
$$f_{R_6, \text{OUT}}[B_2] = f_{R_2, \text{OUT}}[B_2] \circ f_{R_6, \text{IN}}[R_2]$$

$$f_{R_6, \text{IN}}[R_3] = f_{R_6, \text{OUT}}[B_2]$$

$$f_{R_6, \text{OUT}}[B_3] = f_{R_3, \text{OUT}}[B_3] \circ f_{R_6, \text{IN}}[R_3]$$

$$f_{R_6, \text{IN}}[R_4] = f_{R_6, \text{OUT}}[B_2] \wedge f_{R_6, \text{OUT}}[B_3]$$

$$f_{R_6, \text{OUT}}[B_4] = f_{R_4, \text{OUT}}[B_4] \circ f_{R_6, \text{IN}}[R_4]$$



Example of the Algorithm

2. 領域の小 → 大に向かって transfer function を求める

葉領域 :

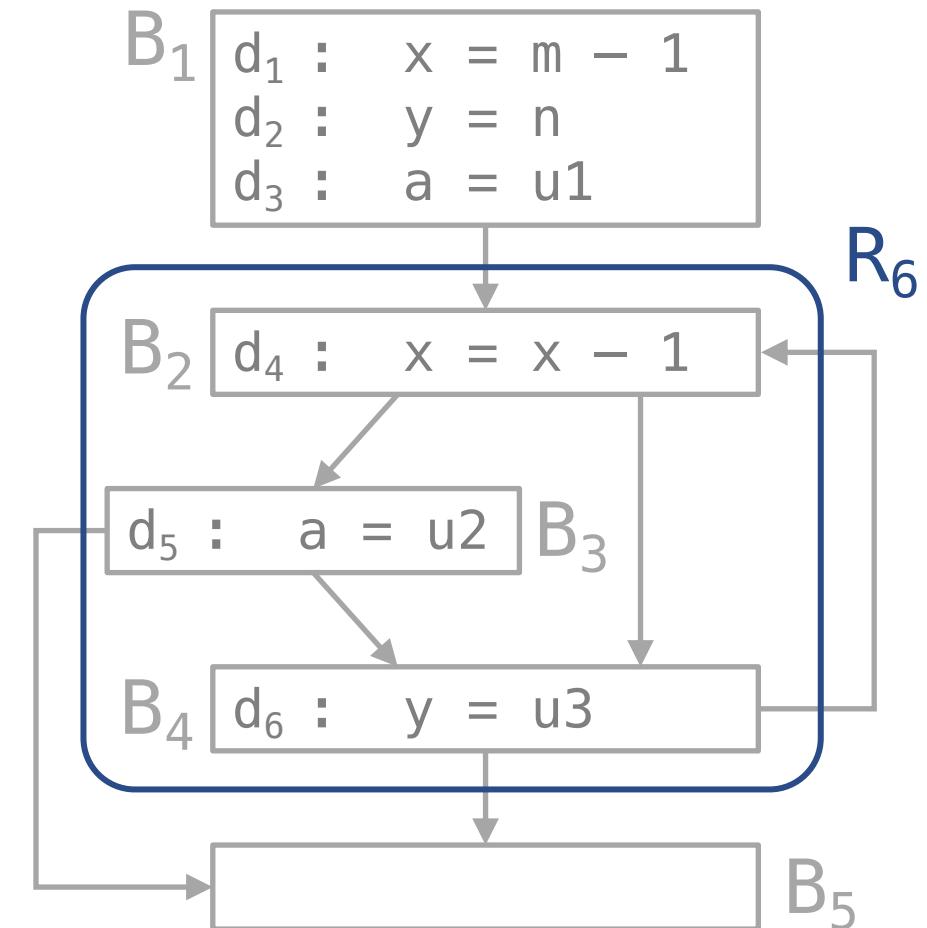
$$f_{R_i, \text{IN}[B_i]} = I$$

$$f_{R_i, \text{OUT}[B_i]} = f_{B_i} \quad (i = 1, 2, 3, 4, 5)$$

領域 R_6 :

$$f = \text{gen} \cup (x - \text{kill})$$

	Transfer Function	gen	kill
R_6	$f_{R_6, \text{IN}[R_2]} = I$	\emptyset	\emptyset
	$f_{R_6, \text{OUT}[B_2]} = f_{R_2, \text{OUT}[B_2]} \circ f_{R_6, \text{IN}[R_2]}$	$\{d_4\}$	$\{d_1\}$
	$f_{R_6, \text{IN}[R_3]} = f_{R_6, \text{OUT}[B_2]}$	$\{d_4\}$	$\{d_1\}$
	$f_{R_6, \text{OUT}[B_3]} = f_{R_3, \text{OUT}[B_3]} \circ f_{R_6, \text{IN}[R_3]}$	$\{d_4, d_5\}$	$\{d_1, d_3\}$
	$f_{R_6, \text{IN}[R_4]} = f_{R_6, \text{OUT}[B_2]} \wedge f_{R_6, \text{OUT}[B_3]}$	$\{d_4, d_5\}$	$\{d_1\}$
	$f_{R_6, \text{OUT}[B_4]} = f_{R_4, \text{OUT}[B_4]} \circ f_{R_6, \text{IN}[R_4]}$	$\{d_4, d_5, d_6\}$	$\{d_1, d_2\}$



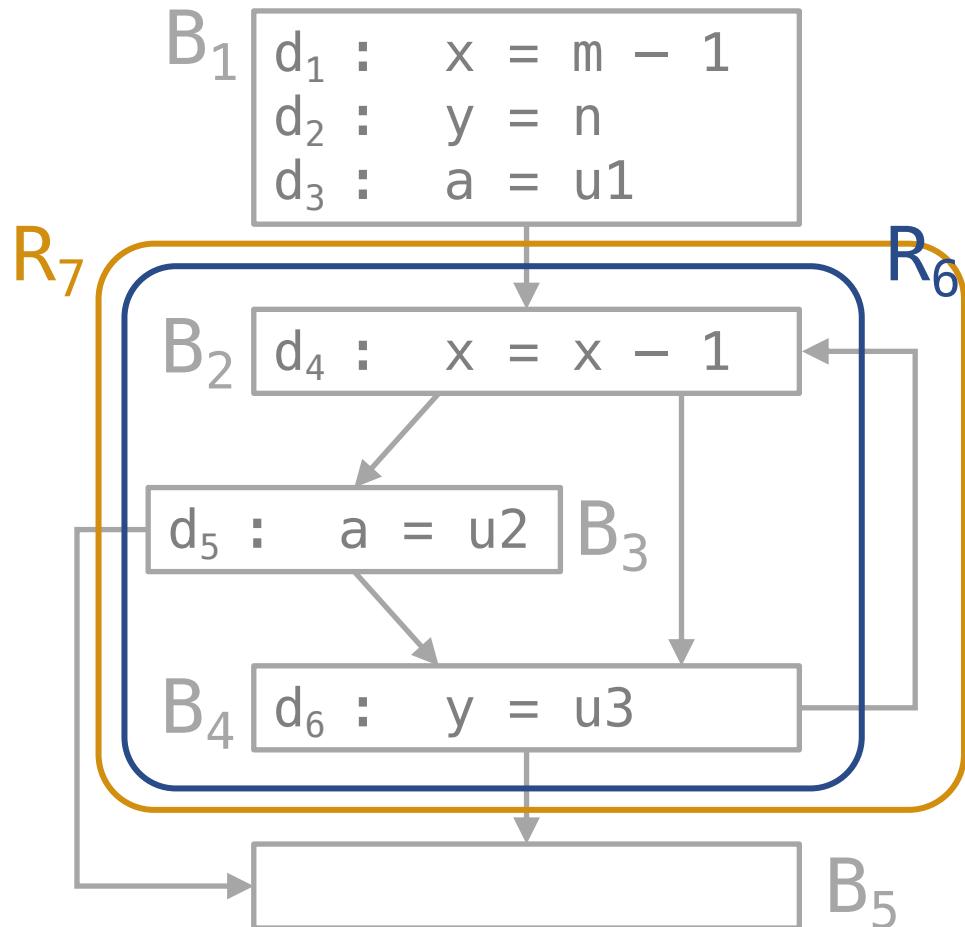
Example of the Algorithm

領域 R_6 :

$$\begin{aligned}
 f_{R_6, \text{IN}[R_2]} &= I \\
 f_{R_6, \text{OUT}[B_2]} &= f_{R_2, \text{OUT}[B_2]} \circ f_{R_6, \text{IN}[R_2]} \\
 f_{R_6, \text{IN}[R_3]} &= f_{R_6, \text{OUT}[B_2]} \\
 f_{R_6, \text{OUT}[B_3]} &= f_{R_3, \text{OUT}[B_3]} \circ f_{R_6, \text{IN}[R_3]} \\
 f_{R_6, \text{IN}[R_4]} &= f_{R_6, \text{OUT}[B_2]} \wedge f_{R_6, \text{OUT}[B_3]} \\
 f_{R_6, \text{OUT}[B_4]} &= f_{R_4, \text{OUT}[B_4]} \circ f_{R_6, \text{IN}[R_4]}
 \end{aligned}$$

領域 R_7 :

Q $f_{R_7, \text{IN}[R_6]} =$



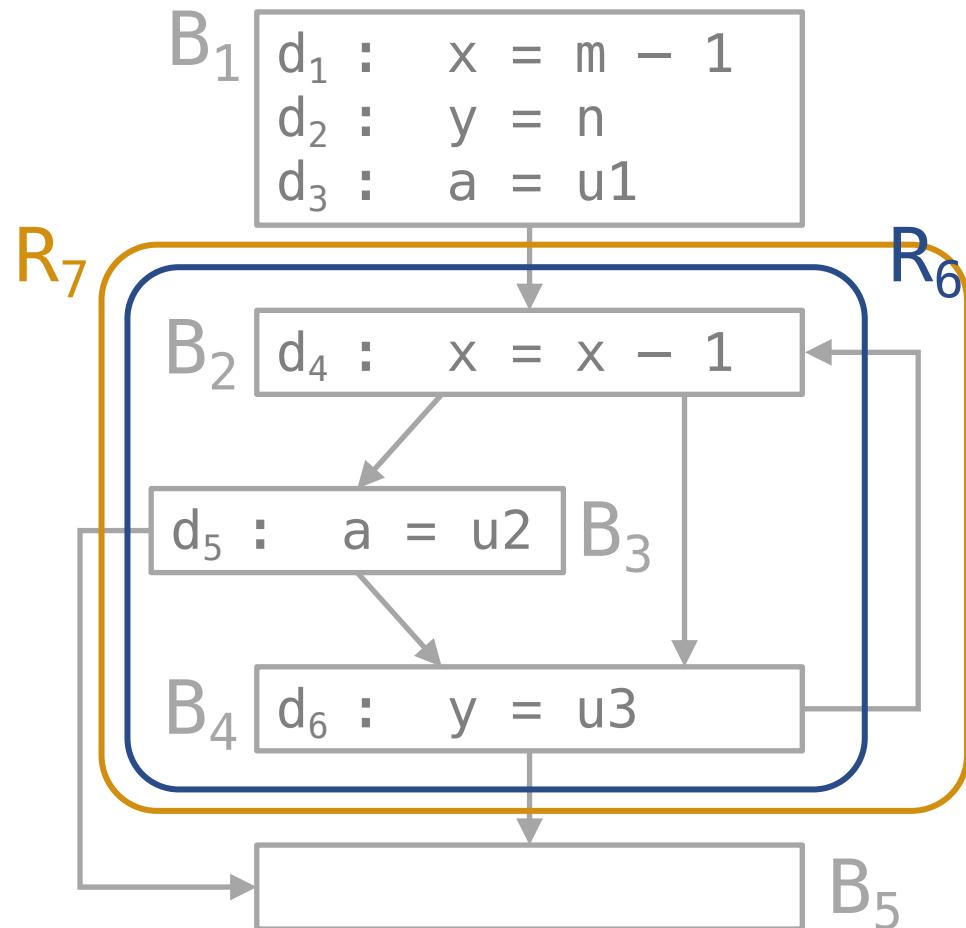
Example of the Algorithm

領域 R_6 :

$$\begin{aligned}
 f_{R_6, \text{IN}[R_2]} &= I \\
 f_{R_6, \text{OUT}[B_2]} &= f_{R_2, \text{OUT}[B_2]} \circ f_{R_6, \text{IN}[R_2]} \\
 f_{R_6, \text{IN}[R_3]} &= f_{R_6, \text{OUT}[B_2]} \\
 f_{R_6, \text{OUT}[B_3]} &= f_{R_3, \text{OUT}[B_3]} \circ f_{R_6, \text{IN}[R_3]} \\
 f_{R_6, \text{IN}[R_4]} &= f_{R_6, \text{OUT}[B_2]} \wedge f_{R_6, \text{OUT}[B_3]} \\
 f_{R_6, \text{OUT}[B_4]} &= f_{R_4, \text{OUT}[B_4]} \circ f_{R_6, \text{IN}[R_4]}
 \end{aligned}$$

領域 R_7 :

$$\begin{aligned}
 f_{R_7, \text{IN}[R_6]} &= (f_{R_6, \text{OUT}[B_4]})^* \\
 f_{R_7, \text{OUT}[B_3]} &= f_{R_6, \text{OUT}[B_3]} \circ f_{R_7, \text{IN}[R_6]} \\
 f_{R_7, \text{OUT}[B_4]} &= f_{R_6, \text{OUT}[B_4]} \circ f_{R_7, \text{IN}[R_6]}
 \end{aligned}$$

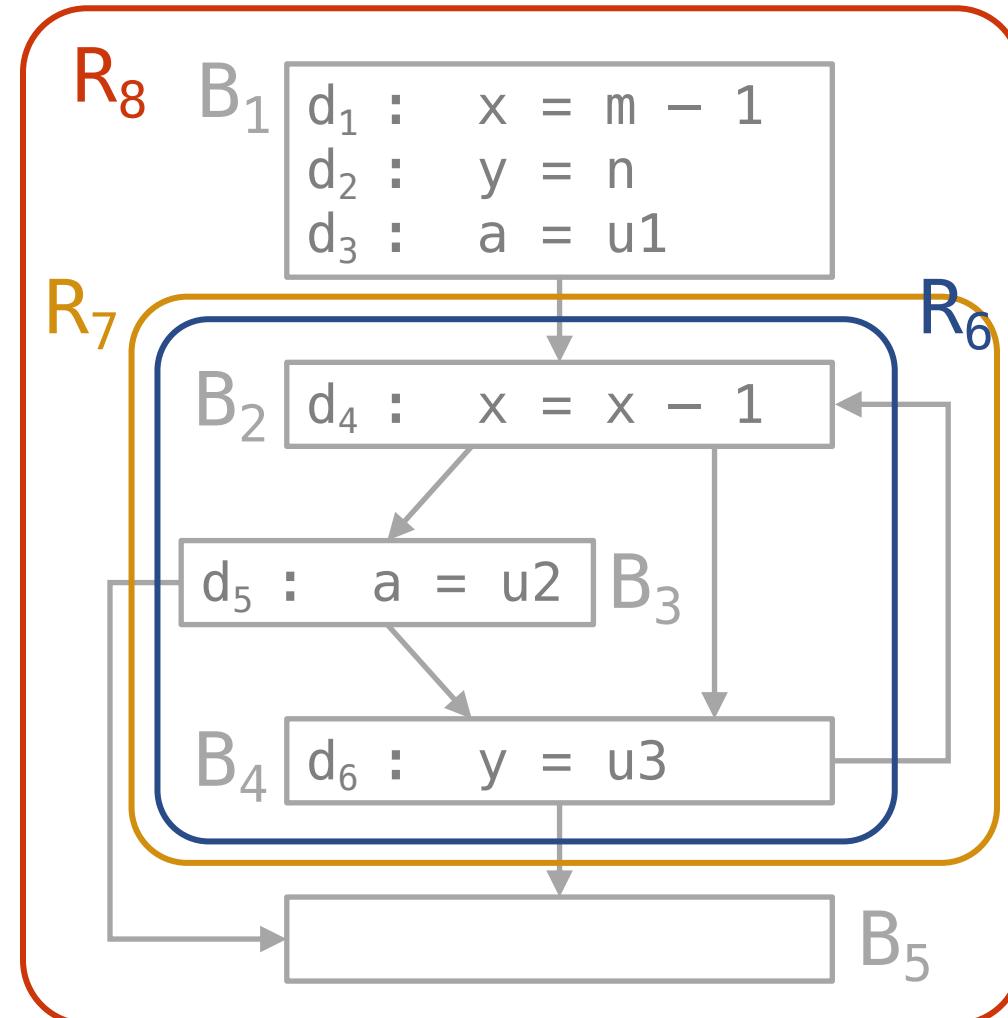


Example of the Algorithm

2. 領域の小 → 大に向かって transfer function を求める

$$f = \text{gen} \cup (x - \text{kill})$$

	Transfer Function	gen	kill
R_6	$f_{R_6, \text{IN}[R_2]} = I$	\emptyset	\emptyset
	$f_{R_6, \text{OUT}[B_2]} = f_{R_2, \text{OUT}[B_2]} \circ f_{R_6, \text{IN}[R_2]}$	$\{d_4\}$	$\{d_1\}$
	$f_{R_6, \text{IN}[R_3]} = f_{R_6, \text{OUT}[B_2]}$	$\{d_4\}$	$\{d_1\}$
	$f_{R_6, \text{OUT}[B_3]} = f_{R_3, \text{OUT}[B_3]} \circ f_{R_6, \text{IN}[R_3]}$	$\{d_4, d_5\}$	$\{d_1, d_3\}$
	$f_{R_6, \text{IN}[R_4]} = f_{R_6, \text{OUT}[B_2]} \wedge f_{R_6, \text{OUT}[B_3]}$	$\{d_4, d_5\}$	$\{d_1\}$
	$f_{R_6, \text{OUT}[B_4]} = f_{R_4, \text{OUT}[B_4]} \circ f_{R_6, \text{IN}[R_4]}$	$\{d_4, d_5, d_6\}$	$\{d_1, d_2\}$
R_7	$f_{R_7, \text{IN}[R_6]} = f_{R_6, \text{OUT}[B_4]}^*$	$\{d_4, d_5, d_6\}$	\emptyset
	$f_{R_7, \text{OUT}[B_3]} = f_{R_6, \text{OUT}[B_3]} \circ f_{R_7, \text{IN}[R_6]}$	$\{d_4, d_5, d_6\}$	$\{d_1, d_3\}$
	$f_{R_7, \text{OUT}[B_4]} = f_{R_6, \text{OUT}[B_4]} \circ f_{R_7, \text{IN}[R_6]}$	$\{d_4, d_5, d_6\}$	$\{d_1, d_2\}$
R_8	$f_{R_8, \text{IN}[R_1]} = I$	\emptyset	\emptyset
	$f_{R_8, \text{OUT}[B_1]} = f_{R_1, \text{OUT}[B_1]}$	$\{d_1, d_2, d_3\}$	$\{d_4, d_5, d_6\}$
	$f_{R_8, \text{IN}[R_7]} = f_{R_8, \text{OUT}[B_1]}$	$\{d_1, d_2, d_3\}$	$\{d_4, d_5, d_6\}$
	$f_{R_8, \text{OUT}[B_3]} = f_{R_7, \text{OUT}[B_3]} \circ f_{R_8, \text{IN}[R_7]}$	$\{d_2, d_4, d_5, d_6\}$	$\{d_1, d_3\}$
	$f_{R_8, \text{OUT}[B_4]} = f_{R_7, \text{OUT}[B_4]} \circ f_{R_8, \text{IN}[R_7]}$	$\{d_3, d_4, d_5, d_6\}$	$\{d_1, d_2\}$
	$f_{R_8, \text{IN}[R_5]} = f_{R_8, \text{OUT}[B_3]} \wedge f_{R_8, \text{OUT}[B_4]}$	$\{d_2, d_3, d_4, d_5, d_6\}$	$\{d_1\}$
	$f_{R_8, \text{OUT}[B_5]} = f_{R_5, \text{OUT}[B_5]} \circ f_{R_8, \text{IN}[R_5]}$	$\{d_2, d_3, d_4, d_5, d_6\}$	$\{d_1\}$

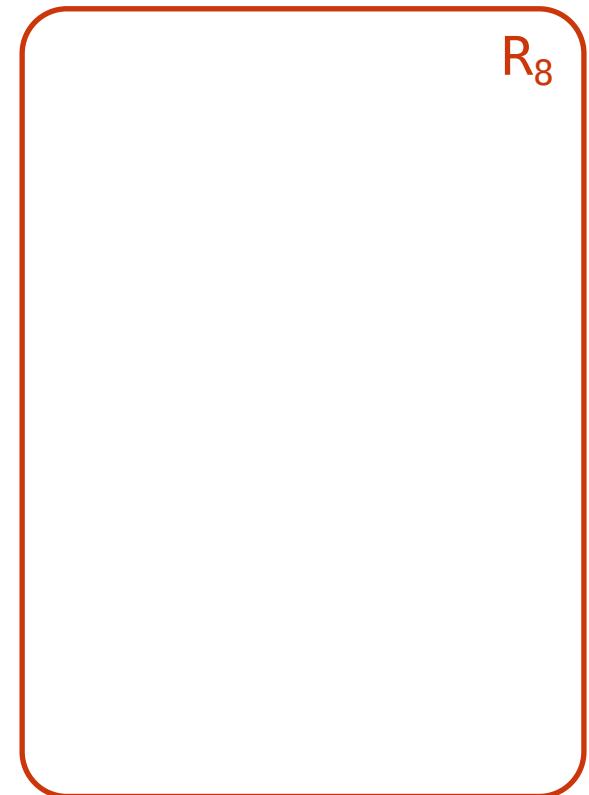


Example of the Algorithm

3. 領域の大 → 小に向かって 各領域の先頭における data-flow values を求める

(a) 境界条件 : $\text{IN}[R_n] = \text{IN}[\text{ENTRY}]$

$$\text{IN}[R_8] = \emptyset$$



Example of the Algorithm

3. 領域の大 → 小に向かって 各領域の先頭における data-flow values を求める

(b) $R \in \{R_1, \dots, R_{n-1}\}$ について領域の大きいもの

から順に $\text{IN}[R] = f_{R', \text{IN}[R]}(\text{IN}[R'])$ を
求める (R' は R を直接囲む領域)

$$\text{IN}[R_8] = \emptyset$$

$$\text{IN}[R_1] = f_{R_8, \text{IN}[R_1]}(\text{IN}[R_8]) = \emptyset$$

$$\text{IN}[R_7] = f_{R_8, \text{IN}[R_7]}(\text{IN}[R_8]) = \{d_1, d_2, d_3\}$$

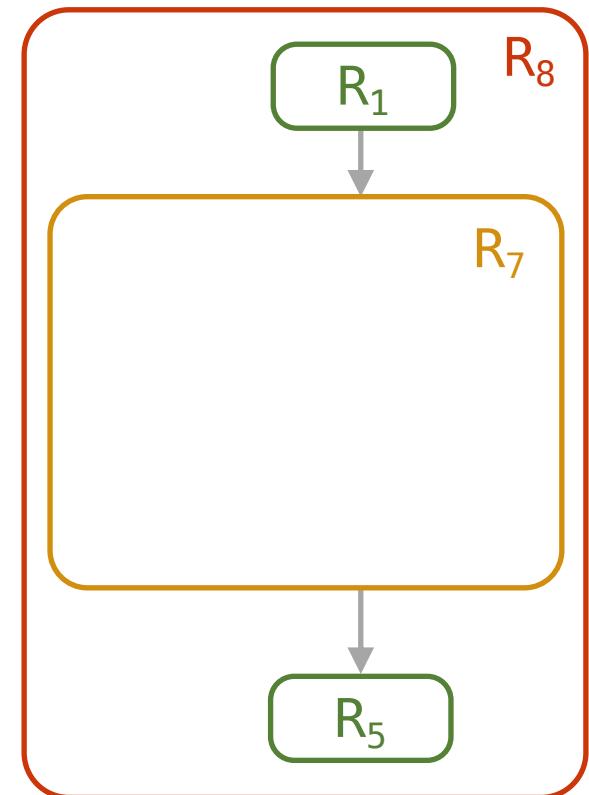
$$\text{IN}[R_5] = f_{R_8, \text{IN}[R_5]}(\text{IN}[R_8]) = \{d_2, d_3, d_4, d_5, d_6\}$$

$$f_{R_8, \text{IN}[R_1]}(x) = x$$

Step2 から

$$f_{R_8, \text{IN}[R_7]}(x) = \{d_1, d_2, d_3\} \cup (x - \{d_4, d_5, d_6\})$$

$$f_{R_8, \text{IN}[R_5]}(x) = \{d_2, d_3, d_4, d_5, d_6\} \cup (x - \{d_1\})$$



Example of the Algorithm

3. 領域の大 → 小に向かって 各領域の先頭における data-flow values を求める

(b) $R \in \{R_1, \dots, R_{n-1}\}$ について領域の大きいもの

から順に $\text{IN}[R] = f_{R', \text{IN}[R]}(\text{IN}[R'])$ を求める（ R' は R を直接囲む領域）

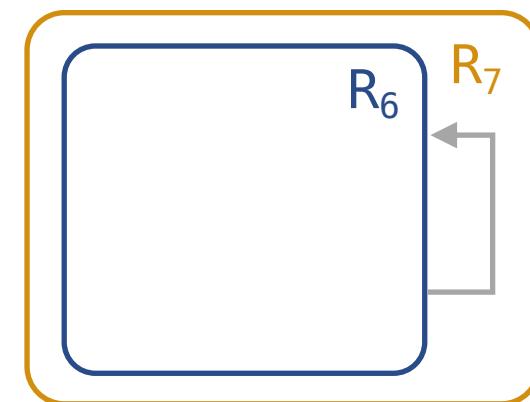
$$\text{IN}[R_8] = \emptyset$$

$$\text{IN}[R_1] = f_{R_8, \text{IN}[R_1]}(\text{IN}[R_8]) = \emptyset$$

$$\text{IN}[R_7] = f_{R_8, \text{IN}[R_7]}(\text{IN}[R_8]) = \{d_1, d_2, d_3\}$$

$$\text{IN}[R_5] = f_{R_8, \text{IN}[R_5]}(\text{IN}[R_8]) = \{d_2, d_3, d_4, d_5, d_6\}$$

$$\text{Q IN } [R_6] = ?$$



Example of the Algorithm

3. 領域の大 → 小に向かって 各領域の先頭における data-flow values を求める

(b) $R \in \{R_1, \dots, R_{n-1}\}$ について領域の大きいものから順に $\text{IN}[R] = f_{R', \text{IN}[R]}(\text{IN}[R'])$ を求める (R' は R を直接囲む領域)

$$\text{IN}[R_8] = \emptyset$$

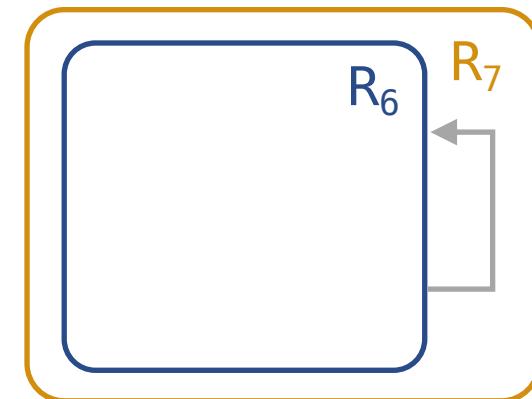
$$\text{IN}[R_1] = f_{R_8, \text{IN}[R_1]}(\text{IN}[R_8]) = \emptyset$$

$$\text{IN}[R_7] = f_{R_8, \text{IN}[R_7]}(\text{IN}[R_8]) = \{d_1, d_2, d_3\}$$

$$\text{IN}[R_5] = f_{R_8, \text{IN}[R_5]}(\text{IN}[R_8]) = \{d_2, d_3, d_4, d_5, d_6\}$$

$$\text{IN}[R_6] = f_{R_7, \text{IN}[R_6]}(\text{IN}[R_7]) = \{d_1, d_2, d_3, d_4, d_5, d_6\}$$

Step2 から $f_{R_7, \text{IN}[R_6]}(x) = \{d_4, d_5, d_6\} \cup x$



Example of the Algorithm

3. 領域の大 → 小に向かって 各領域の先頭における data-flow values を求める

$$\text{IN}[R_8] = \emptyset$$

- $\text{IN}[R_1] = f_{R_8, \text{IN}[R_1]}(\text{IN}[R_8]) = \emptyset$

$$\text{IN}[R_7] = f_{R_8, \text{IN}[R_7]}(\text{IN}[R_8]) = \{d_1, d_2, d_3\}$$

- $\text{IN}[R_5] = f_{R_8, \text{IN}[R_5]}(\text{IN}[R_8]) = \{d_2, d_3, d_4, d_5, d_6\}$

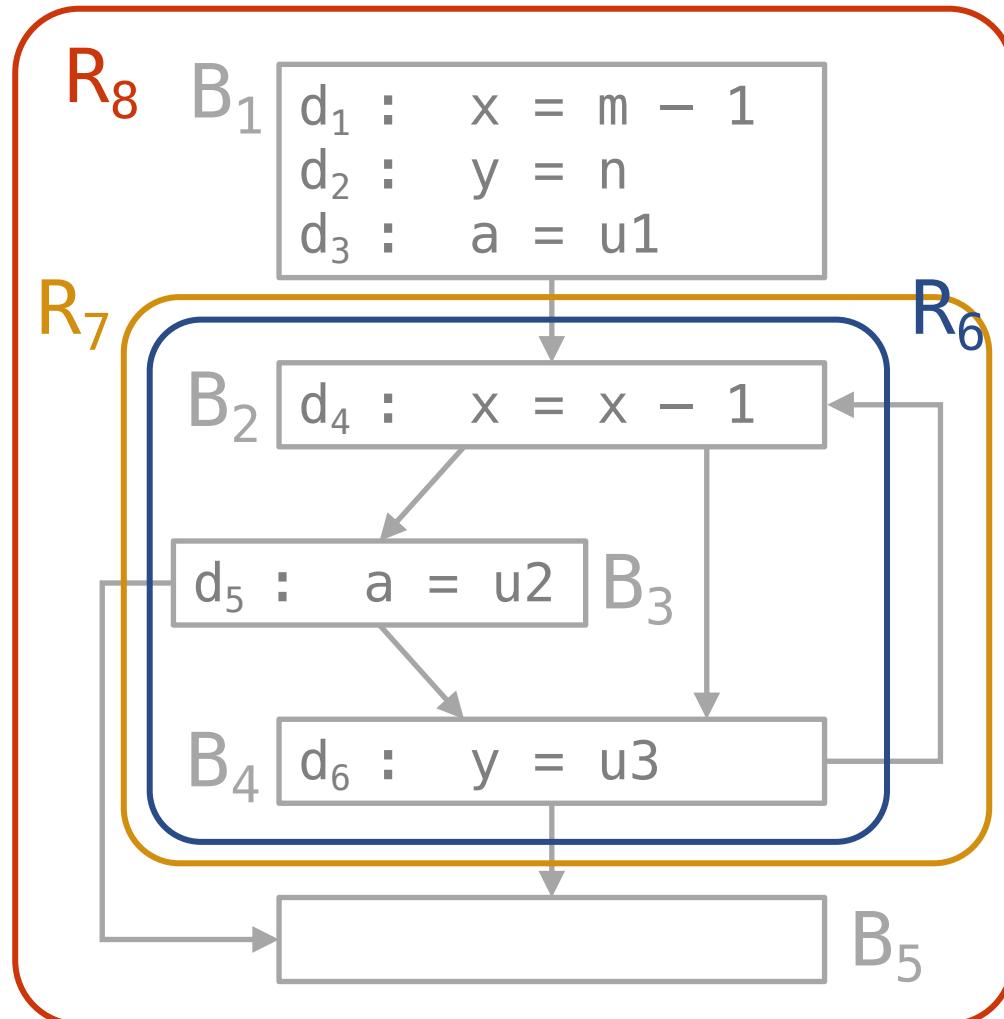
$$\text{IN}[R_6] = f_{R_7, \text{IN}[R_6]}(\text{IN}[R_7]) = \{d_1, d_2, d_3, d_4, d_5, d_6\}$$

- $\text{IN}[R_4] = f_{R_6, \text{IN}[R_4]}(\text{IN}[R_6]) = \{d_2, d_3, d_4, d_5, d_6\}$

- $\text{IN}[R_3] = f_{R_6, \text{IN}[R_3]}(\text{IN}[R_6]) = \{d_2, d_3, d_4, d_5, d_6\}$

- $\text{IN}[R_2] = f_{R_6, \text{IN}[R_2]}(\text{IN}[R_6]) = \{d_1, d_2, d_3, d_4, d_5, d_6\}$

→ 各領域の data-flow values がそれぞれ一回の transfer function の適用で求められた



Contents

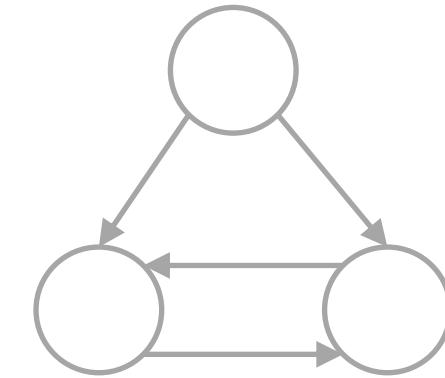
1. Regions
2. Overview of Region-Based Analysis
3. Transfer Functions
4. An Algorithm for Region-Based Analysis
5. Handling Nonreducible Flow Graphs

Handling Nonreducible Flow Graph

領域に基づく解析における nonreducible flow graph の扱い方

- ・ 解析対象が基本的に nonreducible なら
iterative アルゴリズムを使う
- ・ 解析対象がたまに nonreducible なら
node-splitting という技法が使える

直感的には nonreducible を reducible に
変換する



back edge \Rightarrow retreating edge
retreating edge $\not\Rightarrow$ back edge

Handling Nonreducible Flow Graph

Node-splitting の方法

R の前の頂点が k 個あるとき R を k 個複製し、それぞれ一つずつ前の頂点と結ぶ
→ 新しい back edge が作られる

イメージ

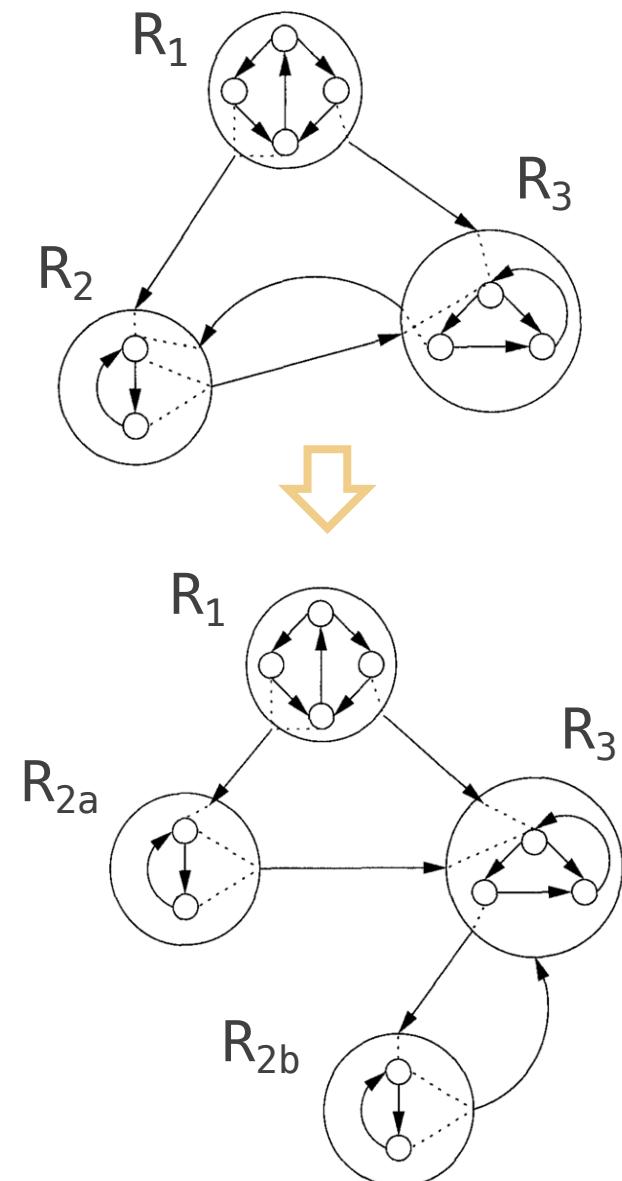
```

1  if (B)
2      goto label;
3  while (1) {
4      x;
5  label:
6      y;
7 }
```



```

1  if (B)
2      y;
3  while (1) {
4      x;
5
6      y;
7 }
```

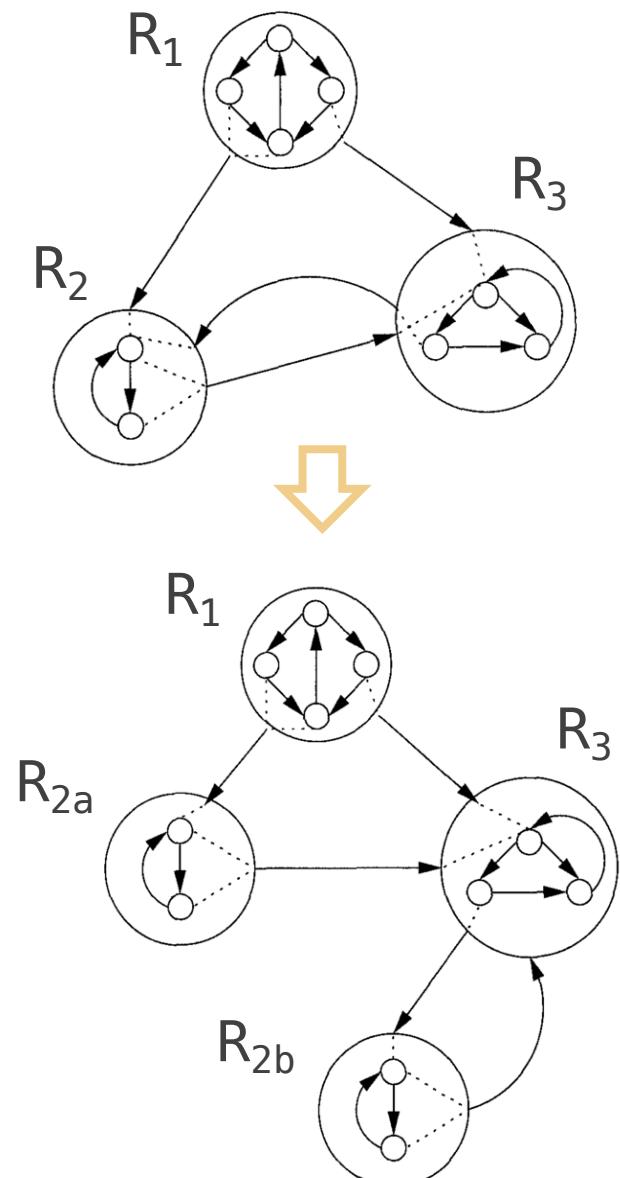


Handling Nonreducible Flow Graph

Node-splitting の方法

R の前の頂点が k 個あるとき R を k 個複製し、それぞれ一つずつ前の頂点と結ぶ
→ 新しい back edge が作られる

これを繰り返すことで最終的に reducible flow graph と同じように扱える
→ 最悪ブロックの数は指数的になる



Handling Nonreducible Flow Graph

Node-splitting をしたときの解析結果について

1. 複製したブロックの解は元々のブロックの解よりも情報が絞られる場合がある

ex.)

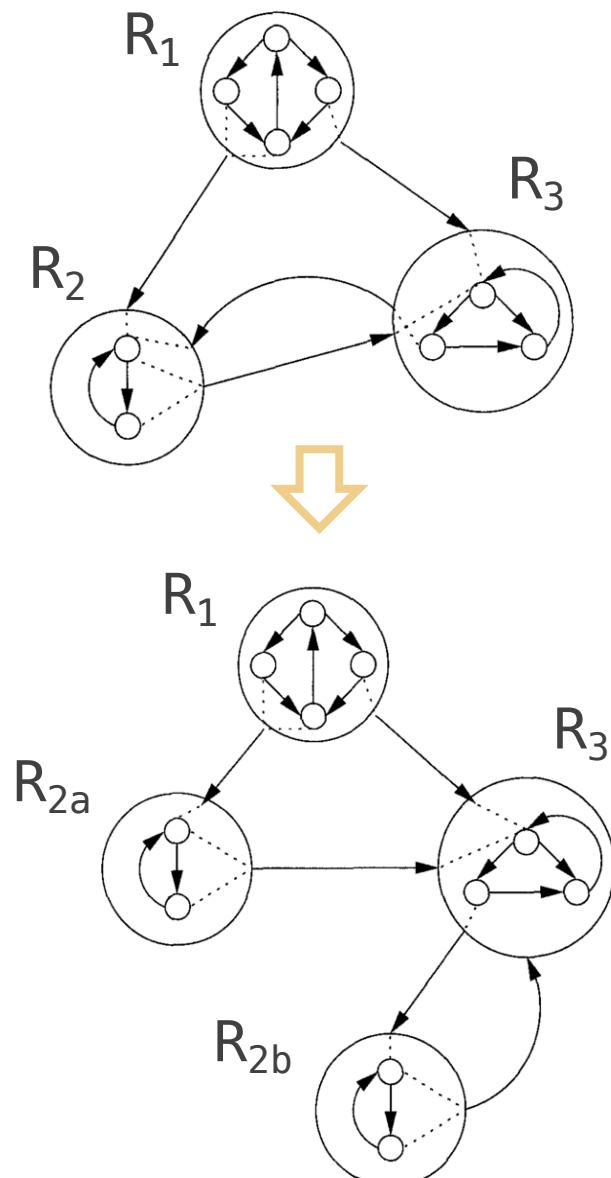
R_{2a} は R_2 に R_1 から直接到達した場合だけを表す

2. 分離したブロックを元に戻したいとき

B を複製して B_1, B_2, \dots, B_k が作られたとき

$$\text{IN}[B] = \text{IN}[B_1] \wedge \text{IN}[B_2] \wedge \dots \wedge \text{IN}[B_k]$$

を計算すればよい



まとめ

- ・領域に基づく解析手法を見てきた
- ・領域に基づく解析アルゴリズムは反復アルゴリズムに対して
 1. ループの影響をより効果的にまとめられる点
 2. ある手続きのかたまりを一つにまとめられる点

で有効である