知能ソフトウェア特論
Intelligent Software

プログラムの正当性(2)
ホーア論理
Correctness of Programs (2)
Hoare Logic

1　構造化プログラミング(1/2)

どんな制御構造も，3つの基本構造

順次 (sequence)

選択 (selection)

反復 (iteration)

の組合せによる表現に変換できる.

構造化プログラミング
Structured programming

Any control structure can be transformed into an equivalent structure combining the following three basic structures:

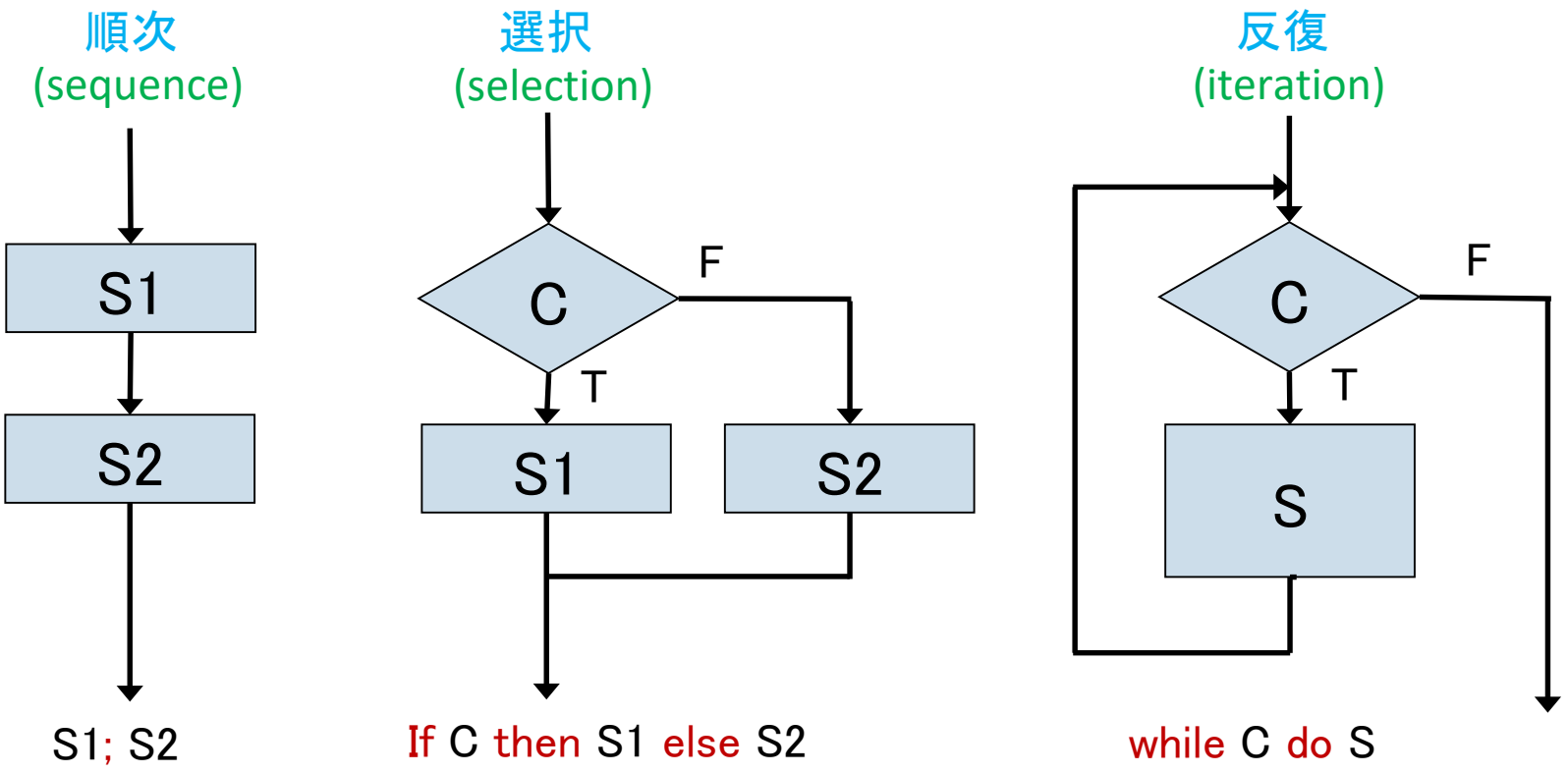
- sequence
- selection
- iteration

This observation led to the notion and techniques known as structured programming.

【今回の授業内容】
構造化プログラミングで書かれた
任意のプログラムについて，
（部分）正当性の証明方法を学ぶ

Today we will discuss how to prove the (partial) correctness of an arbitrary program written in structured programming.

1　構造化プログラミング(2/2)



- 出入口はそれぞれ1カ所　(Only one entrance and one exit)
- 流れを表す線が交差しない　(No crossings of flow lines)

2. 論理と推論　（Logic and inference）

公理　最初から知識ベースに組み込まれている自明な知識

Axioms are pieces of knowledge, regarded as trivially true, built-in in the knowledge base from the beginning.

公理
(axiom)

$$\frac{true}{1>0} \qquad \frac{true}{A=A} \qquad \frac{true}{P \rightarrow P}$$

推論規則　知識ベース内の知識（前提）から得られる新たな知識（結論）を知識ベースに追加するアルゴリズム

Inference rules are algorithms to add to the knowledge base a new piece of knowledge (consequent) obtained from the pieces of knowledge (antecedents) in the current knowledge base.

推論規則
(inference rule)

$$\frac{P \quad P \rightarrow Q}{Q}$$

前提 (antecedent)

結論 (consequent)

3. ホーア論理 (1/8)　(Hoare Logic)

【構文】部分正当性を表現する論理式を3つ組で表現:

【Syntax】Logical expressions that represent partial correctness are formed as a 3-tupple.

$$\{P\}\,S\,\{Q\}$$

precondition　sentence　postcondition

【意味】P が真の状態から文 S を実行すると，もし停止すれば，その時点で Q が真となる.

【Meaning】If the execution of the statement S starts from a state in which P is true, and if it terminates, then Q is true when the execution has terminated.

3. ホーア論理 (2/8) 空文 (Empty statement)

プログラミング言語の各構文の意味を3つ組を用いた公理や推論規則で表現

The meaning of each construct of the programming language is defined by axioms and inference rules using the 3-tupples.

(空文) Empty statement

$$\frac{P \rightarrow Q}{\{P\}\ skip\ \{Q\}}$$

P, *Q*は，プログラミング言語とは無関係な一般の数学の命題

P and Q are standard mathematical formulas unrelated to programming

Example: Prove that { a=1, b=2 } skip {a+b=3}.

Answer: We can verify that (a=1, b=2) → a+b=3.

Therefore, we conclude { a=1, b=2 } skip {a+b=3}.

結論から前提に向けて推論を進めると，最終的には3つ組はなくなり，部分的正当性の証明は，通常の数学的／論理学的な式の証明に帰着される.

Backward reasoning will eventually eliminate 3-tupples, and the proof of partial correctness will be reduced to the proof of ordinary mathematical and/or logical expressions.

3. ホーア論理 (3/8) 空文（続き）　(Empty statement: Cont'd)

(空文)　$\dfrac{P \rightarrow Q}{\{P\}\, skip\, \{Q\}}$　(Empty statement)

*P*が未知のとき：　*P* = *Q* とする

If *P* is unknown, let *P*=*Q*, because then the antecedent is true.

Example:　Find *P* such that {*P*} skip {a+b=3} is true.

Answer:　Let *P* be a+b=3.

Then clearly, {a+b=3} skip {a+b=3} is true, as

{a+b=3} → {a+b=3}

is true.

3. ホーア論理 (4/8) 代入 (Assignment)

(代入) Assignment $\dfrac{P \to Q[x := E]}{\{P\}\, x := E\, \{Q\}}$

Q に出現するすべての x を E に置き換えたアサーション

This expression denotes the assertion obtained by replacing all the occurrences of x in Q by E.

Example: Prove { a=9} a:=a+1 {a=10}.

Answer: It is clear that

$$a=9 \ \to a+1=10.$$

Therefore, { a=9} a:=a+1 {a=10}.

$a=10$ に出現するすべての a を $a+1$ に置き換えたアサーション

This expression is the assertion obtained by replacing all the occurrences of a in $a=10$ by $a+1$.

3. ホーア論理 (5/8) 代入 (Assignment Cont'd)

(代入) $$\frac{P \rightarrow Q[x := E]}{\{P\}\, x := E\, \{Q\}}$$ (Assignment)

*P*が未知のとき: $P = Q[x:=E]$

If P is unknown, let $P=Q[x:=E]$, because then the antecedent is true.

Example: Find *P* such that {*P*} a:= b+1 {a+b=3} is true.

Answer: Let *P* be (b+1)+b=3, i.e., b=1.

Then clearly, {b=1} a:= b+1 {a+b=3} is true, as

{b=1} → {(b+1)+b=3}

is true.

3. ホーア論理 (6/8) 連接 (Concatenation)

(連接) Concatenation

$$\frac{\{P\}\,S_1\,\{Q\} \quad \{Q\}\,S_2\,\{R\}}{\{P\}\,S_1;S_2\,\{R\}}$$

より単純な構文の
部分正当性に帰着させている

Divide a complex problem into two simpler problems

P が未知のとき:

再帰的に, *R*から*Q* を求め,
さらに*Q*から *P* を求める

If *P* is unknown, recursively obtain *Q* from R and then *P* from *Q*.

3. ホーア論理 (7/8) 選択 (Selection)

(選択) Selection

$$\frac{\{P \wedge C\}\, S_1\, \{Q\} \quad \{P \wedge \neg C\}\, S_2\, \{Q\}}{\{P\}\ \text{if } C \text{ then } S_1 \text{ else } S_2 \text{ end}\ \{Q\}}$$

Pが未知のとき:

$P = (C \rightarrow P_1) \wedge (\neg C \rightarrow P_2)$

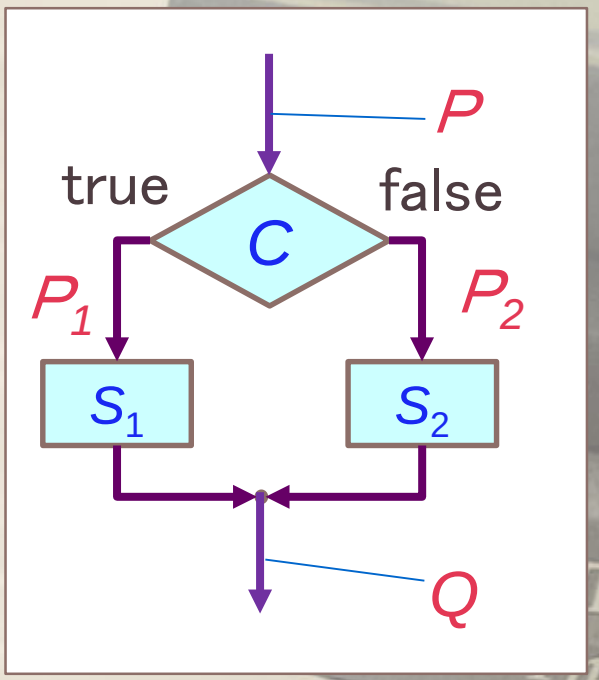
ただし,

$\{P_1\}\, S_1\, \{Q\},\ \{P_2\}\, S_2\, \{Q\}$

を満たすP_1とP_2を再帰的に求める.

If P is unknown,
let $P = (C \rightarrow P_1) \wedge (\neg C \rightarrow P_2)$,
where we recursively obtain P_1 and P_2
such that

$$\{P_1\}\, S_1\, \{Q\},\ \{P_2\}\, S_2\, \{Q\}\ .$$



3. ホーア論理(8/8) 反復　(Repetition)

(反復) Repetition

$$\frac{\{P \wedge C\}\, S\, \{P\} \quad P \wedge \neg C \rightarrow Q}{\{P\}\ \text{while } C \text{ do } S \text{ end}\ \{Q\}}$$

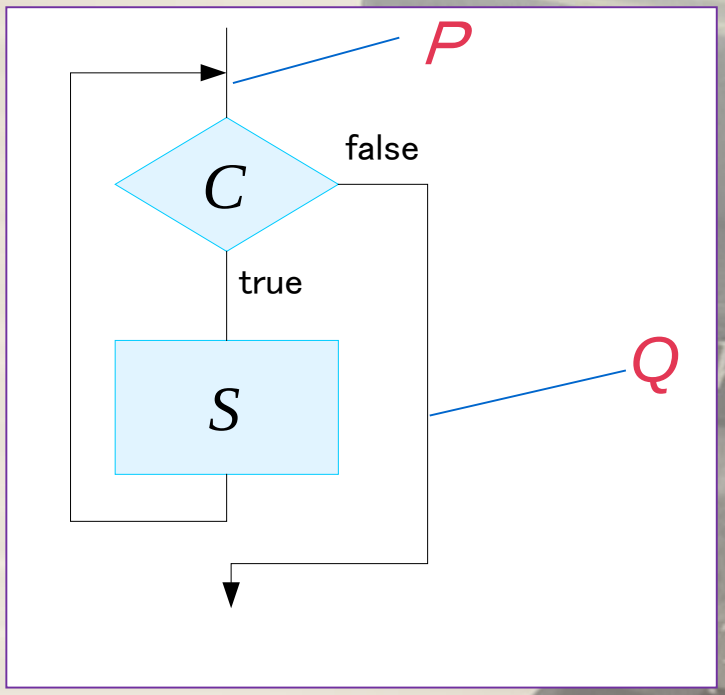
P : ループ不変条件

(P is a loop invariant)

P が未知のとき:

*P*は自動的に求められない.

プログラマがソースコードに必ず挿入しておく.



例題1　簡単な計算と代入
Example 1: Simple arithmetic and assignment

$$\dfrac{\dfrac{}{x=a,y=b \to (x+y)+y=a+2b}\ \{x=a,y=b\}\ x:=x+y\ \{x+y=a+2b\} \qquad \dfrac{}{x+y=a+2b \to x+y=a+2b}\ \{x+y=a+2b\}\ y:=x+y\ \{y=a+2b\}}{\{x=a,y=b\}\ x:=x+y;\ y:=x+y\ \{y=a+2b\}}$$

連接

（代入）　$\dfrac{P \to Q[x:=E]}{\{P\}\,x:=E\,\{Q\}}$

*P*が未知のとき：　*P* = *Q*[*x*:=*E*]

代入が連続するときの簡便な方法
An easy method for a sequence of assignments

$$\frac{\dfrac{P \to R[y := F][x := E]}{\{P\}\, x := E\, \{R[y := F]\}} \quad \dfrac{\dfrac{}{true}}{R[y := F] \to R[y := F]} \quad \{R[y := F]\}\, y := F\, \{R\}}{\{P\}\, x := E;\ y := F\, \{R\}}$$

(代入の連続) Sequence of assignments

$$\frac{P \to R[y := F][x := E]}{\{P\}\, x := E;\ y := F\, \{R\}}$$

例題2　m と n の積 (1/3)　初期設定
Example 2: Product of m, n　Initialization

$\{\text{int } m, n \geq 0\}$

$p := 0;$

$c := n;$

$\{p = m(n - c) \wedge c \geq 0\}$

while $c > 0$ do

$p := p + m;$

$c := c - 1$

end

$\{p = mn\}$

$$\frac{m, n \geq 0 \rightarrow 0 = m(n - n) \wedge n \geq 0}{\{\text{int } m, n \geq 0\}\ p := 0; c := n; \{p = m(n - c) \wedge c \geq 0\}}$$

例題2　mとnの積 (2/3) ループ継続
Loop repetition

$$p = m(n-c) \wedge c \geq 0 \wedge c > 0 \rightarrow p + m = m(n-(c-1)) \wedge c - 1 \geq 0$$

$$\{p = m(n-c) \wedge c \geq 0 \wedge c > 0\}\ p := p + m;\ c := c - 1\ \{p = m(n-c) \wedge c \geq 0\}$$

invariant　　invariant

$\{\text{int } m, n \geq 0\}$

$p := 0;$

$c := n;$

invariant　$\{p = m(n-c) \wedge c \geq 0\}$

while $c > 0$ do

　　$p := p + m;$

　　$c := c - 1$

end

$\{p = mn\}$

反復

例題2　mとnの積 (3/3) ループ終了
Loop termination

$$\frac{\textit{true}}{p = m(n-c) \wedge c \geq 0 \wedge \neg(c > 0) \rightarrow p = mn}$$

invariant

$c = 0$

$\{\text{int } m, n \geq 0\}$

$p := 0;$

$c := n;$

invariant $\{p = m(n-c) \wedge c \geq 0\}$

while $c > 0$ do

$p := p + m;$

$c := c - 1$

end

$\{p = mn\}$

反復

演習問題　２
EXERCISE 2

n の階乗を求める右の
プログラムの部分正当性を
ホーア論理に基づいて証明せよ.

This is a program computing the factorial n! of n. Prove its partial correctness based on Hoare Logic.

$$\{\text{int } n \geq 0\}$$

$$p := 1; \quad i := 0;$$

$$\{p = i! \wedge i \geq 0\}$$

$$\text{while } i \neq n \text{ do}$$

$$\quad i := i + 1;$$

$$\quad p := p \times i$$

$$\text{end}$$

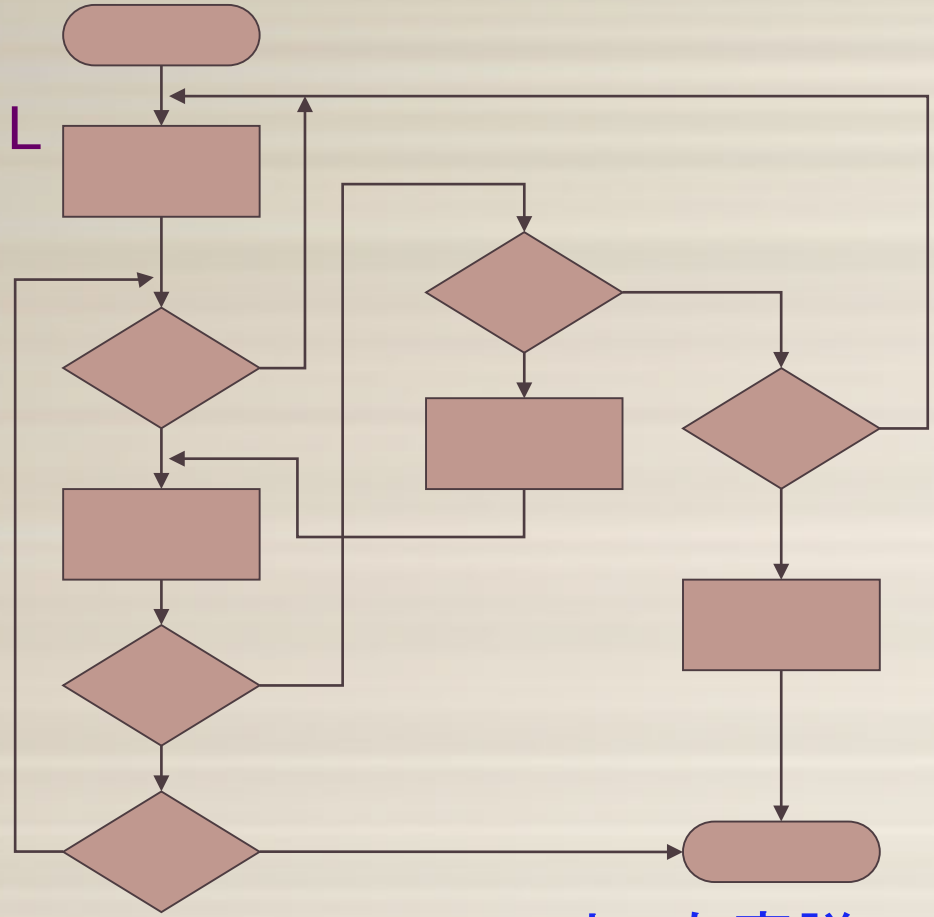
$$\{p = n!\}$$

以下のスライドは，参考までに，構造化プログラミングについてやや詳しく書いたものです

参考

構造化定理 (1/5) スパゲティ・プログラム

(Structure theorem)　(Spaghetti programs)



Go To 文　有害説　(Dijkstra)

(Go To Statement Considered Harmful)

構造化定理 (2/5) 構造化定理　(Structure theorem)

どんな流れ図も，3つの基本構造

- **連接** (concatenation)
- **選択** (selection)
- **反復** (repetition)

の組合せにより，等価な

構造化流れ図

(structured flowchart)

に変換できる.

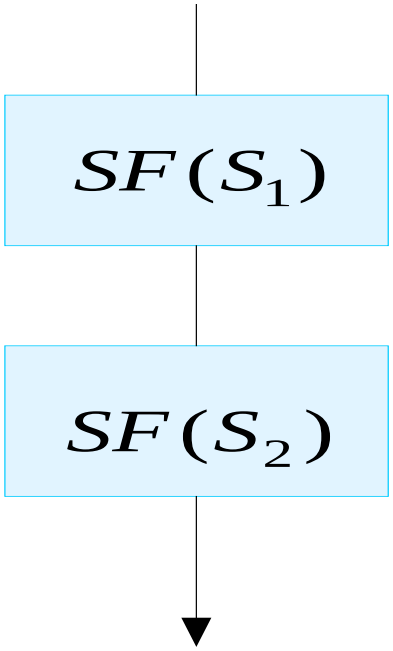
Any flowchart can be transformed into an equivalent structured flowchart by combining the following three structures.

- concatenation
- selection
- repetition

構造化プログラミング
Structured programming

構造化定理 (3/5) 構造化流れ図 (Structured Flowchart)

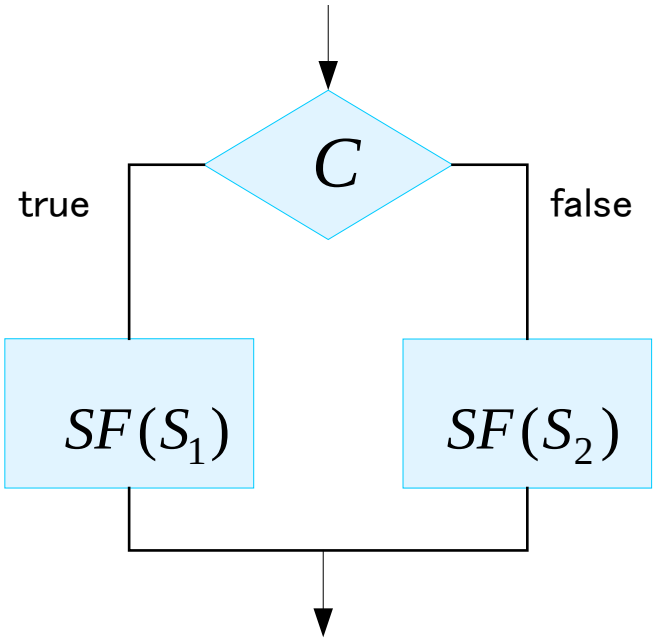
concatenation



連接

$S_1; S_2$

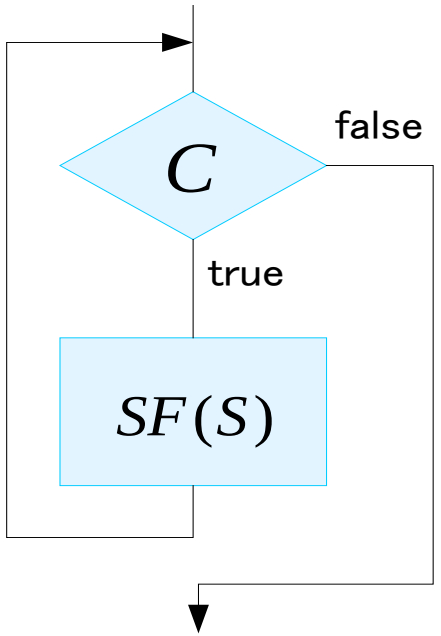
selection



選択

if C then S_1 else S_2 end

repetition



反復

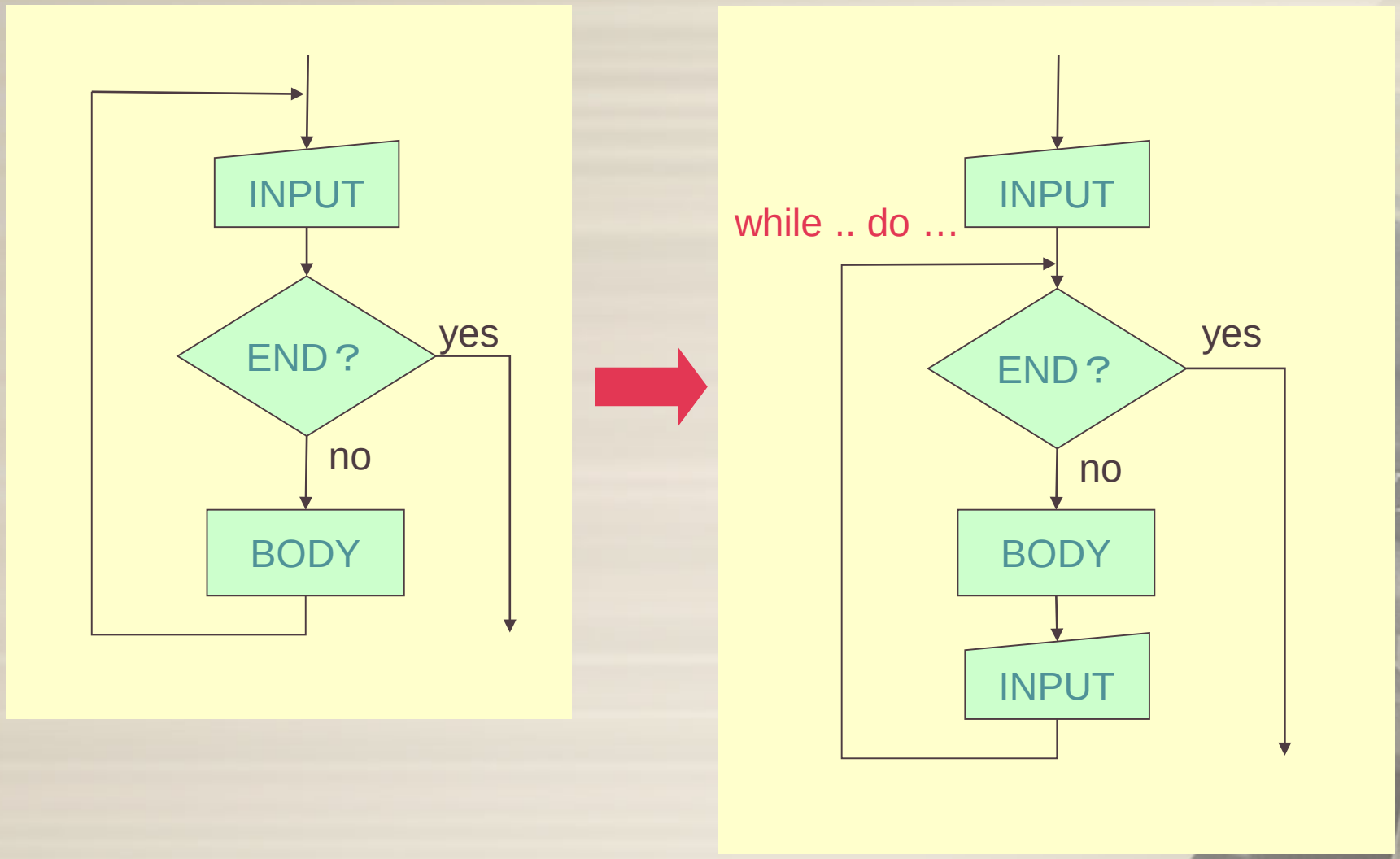
while C do S end

- 出入口はそれぞれ1カ所 (Only one entrance and one exit)
- 流れを表す線が交差しない (No crossings of flow lines)

現実には，`return` と `break` くらいは許す？
(In practice, we may allow at least `return` and `break`?)

構造化定理 (4/5) 前判定反復への変換

(Transforming a repetition into the while loop)



構造化定理 (5/5) より複雑な変換の例

(Example of more complex transformation)