2011-2 <소프트웨어공학개론> Team Project #2.

Control Flow Graph

CLASS A - T1

- 발표자 - 201011336백인선
- 팀원 - 201011309권선일,
  201011357이주희,
  201011342안혜수

Index

1. Control Flow Graph - (1) Test

- **검사(TEST)** : 소프트웨어 품질 보증 활동의 하나로, 소프트웨어에 대한 요구사항의 만족도 및 예상 결과와 실제 결과의 차이점을 여러 방법을 사용하여 검사하고 평가하는 일련의 과정

- 검사 기법 : 최소한의 시간과 노력으로 대부분의 오류를 찾아내기 위해 검사 사례(Test Case)를 설계하는 방법

* 검사 사례(Test Case) : 검사 자료나 실행될 조건을 의미하는 것으로, 소프트웨어 오류를 찾아낼 수 있는 가능성이 가장 높은 테스트를 얻어내는 것

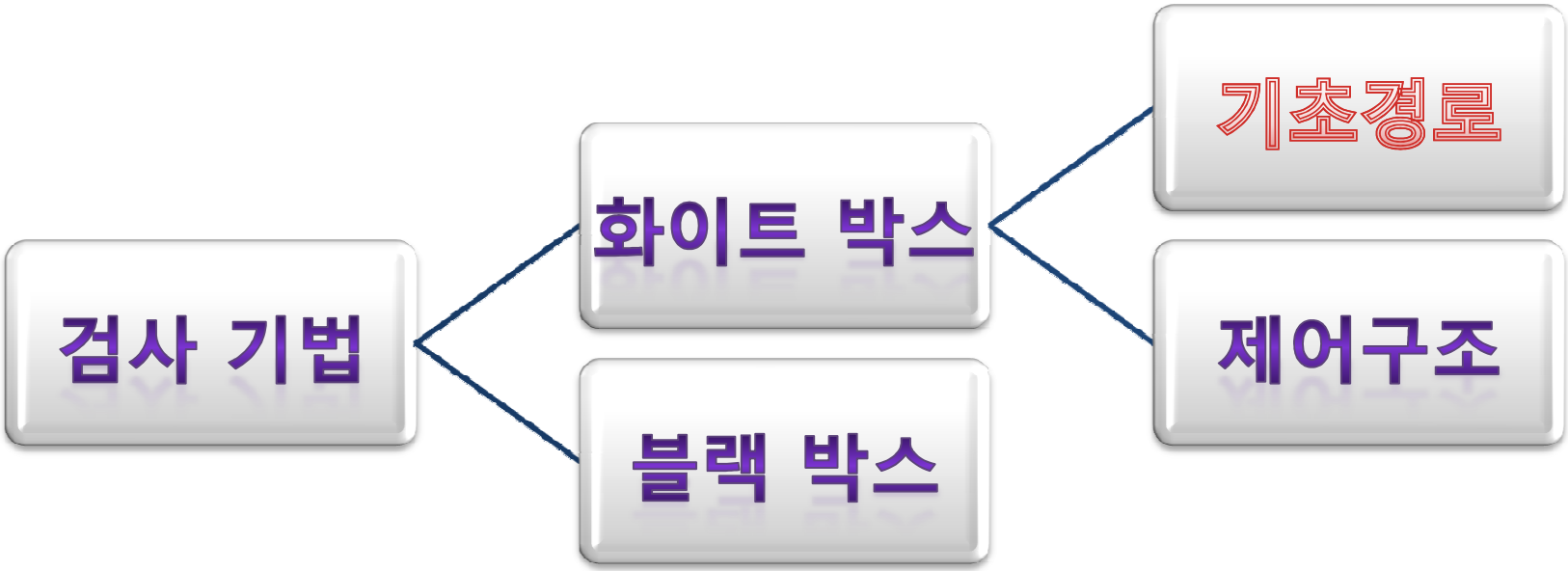
1. Control Flow Graph – (1) Test (Cont.)

① 블랙 박스 테스트(기능검사) : 소프트웨어가 수행할 특정 기능을 알기 위해서 각 기능이 완전히 작동되는 것을 입증하는 검사

② **화이트 박스 테스트** : 설계된 **절차에 초점**을 둔 구조적 테스트, 프로시저(절차) 설계의 제어 구조를 사용하여 검사 사례를 설계하며, 테스트 과정의 초기에 적용 ▶ 모듈 안의 작동을 직접 관찰 할 수 있고, 원시 코드의 모든 문장을 한 번 이상 수행함으로써 수행

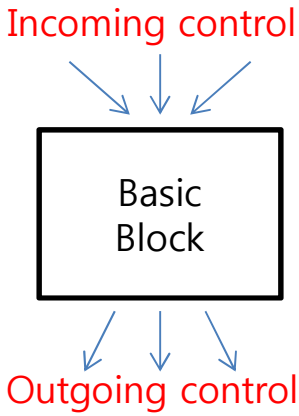
- 기초 경로 검사(Basic Path Testing) : **제어 흐름도**(Control Flow Graph), 순환 복잡도
- 제어 구조 검사 : 조건 검사(Condition Testing), 루프 검사(Loop Testing), 데이터 흐름 검사(Data Flow Testing)

1. Control Flow Graph – (1) Test (Cont.)



1. Control Flow Graph – (2) 개념

- 프로그램의 제어 구조를 그래프 형태로 나타낸 것
- 프로그램 그래프, 흐름 그래프
- 프로그램 제어흐름의 static analysis의 뼈대



- **Nodes** = 프로그램의 각 기본 블록(basic block) → 순서적으로 실행되는 문장들의 집합(블록 내에 존재하지 않는 경우, 문장 하나도 기본 블록의 단위로 봄)
- **Edges** = 제어 흐름을 표시

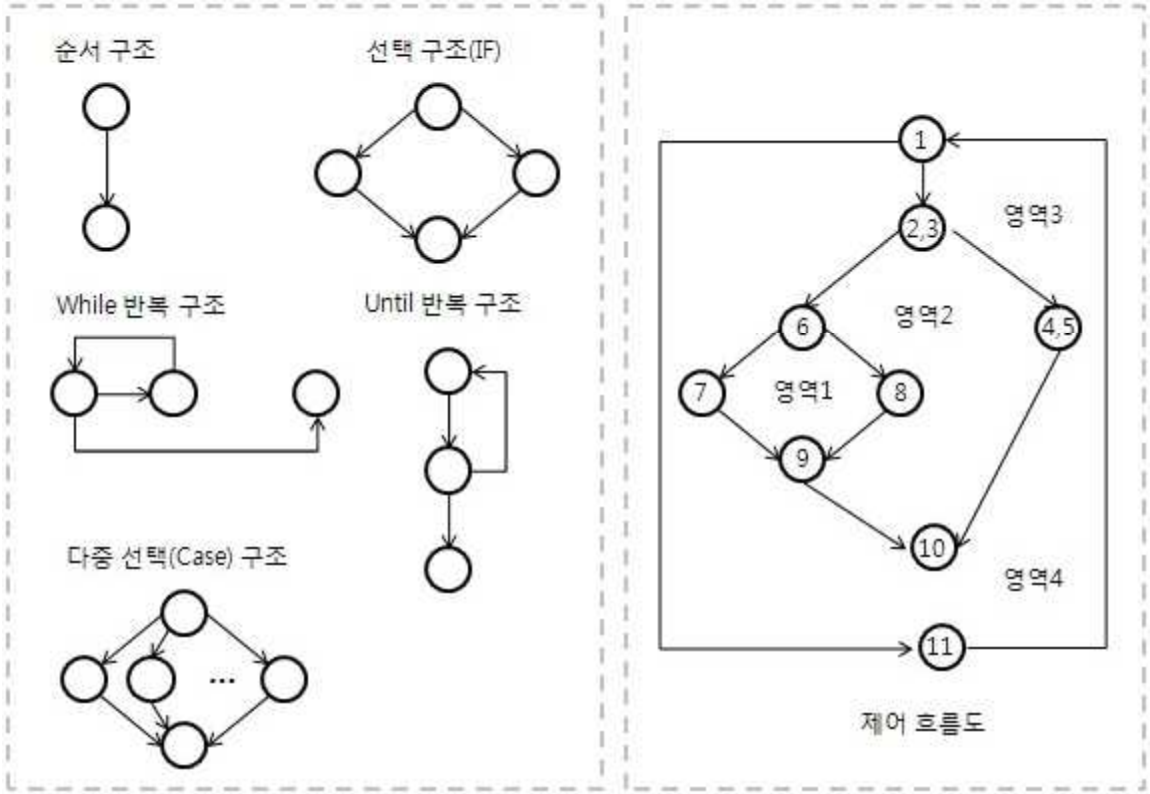
1. Control Flow Graph – (2) 개념 (Cont.)

- 표기 기호

① **노드** : 절차적 명령문

② **화살표** : 제어 흐름

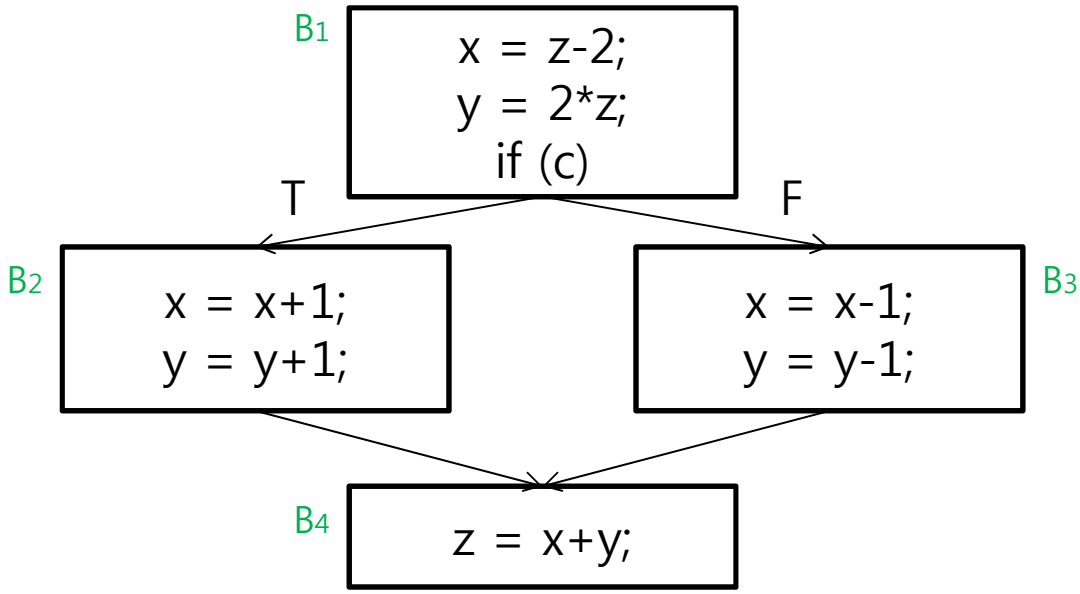
③ **영역** : 화살표와 노드(원)으로 둘러싸인 구역, 외부구역도 하나의 영역으로 포함



1. Control Flow Graph – (2) 개념 (Cont.)

- 예제)

```
x = z-2;
y= 2*z;
if(c){
        x = x+1;
        y = y+1;
}
else {
        x = x-1;
        y = y-1;
}
z = x+y;
```



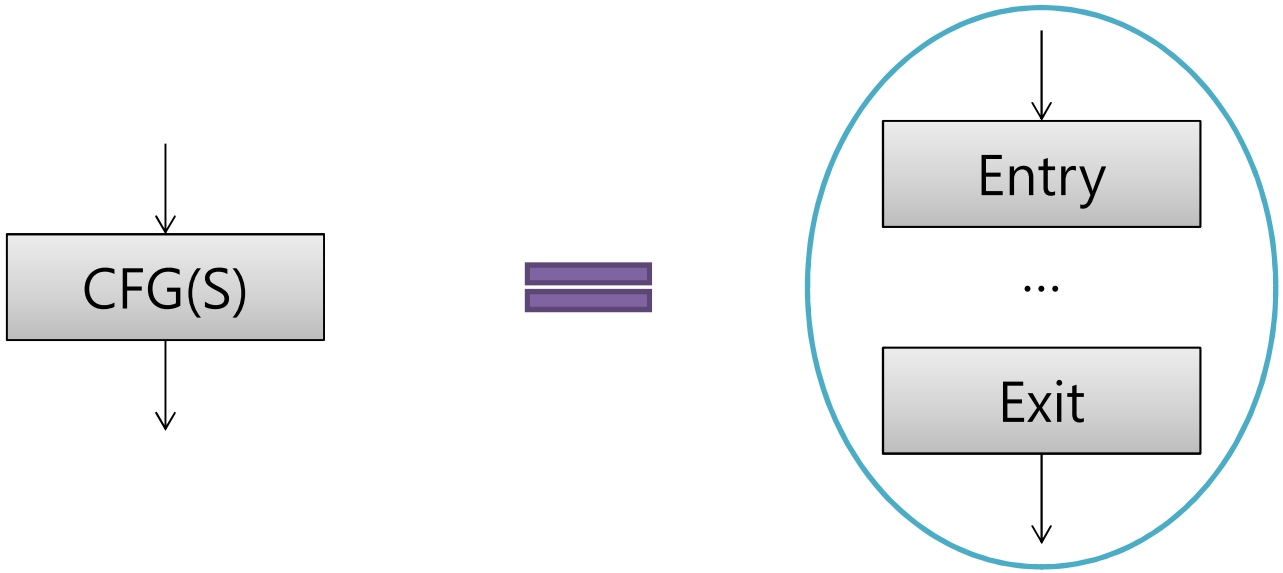
- C가 true라면 path는 B1, B2, B4
- C가 false라면 path는 B1, B3, B4

2. Building the CFG

- High-level IR : Input 언어에서 수행 가능한 언어 분석
  - if, while, for, switch(structured control flow)
  - 변수, 표현식, 문장, 함수
- Low-level IR : 단순한 instruction으로 구성, assembly어로 번역 용이, machine의 구조를 어느정도 표현
  - memory location, registers, unstructured jump

2. Building the CFG – (1) High-level IR

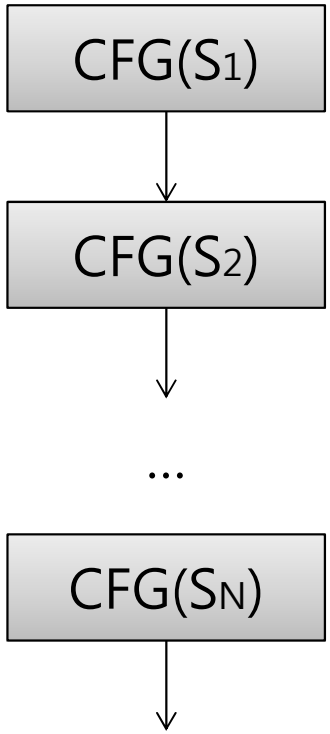
- CFG(S) : flow graph of high-level statement S
  - 하나의 Entry node (basic block)
  - 하나의 Exit node (basic block)



2. Building the CFG – (1) High-level IR (Cont.)

- CFG for Block Statement

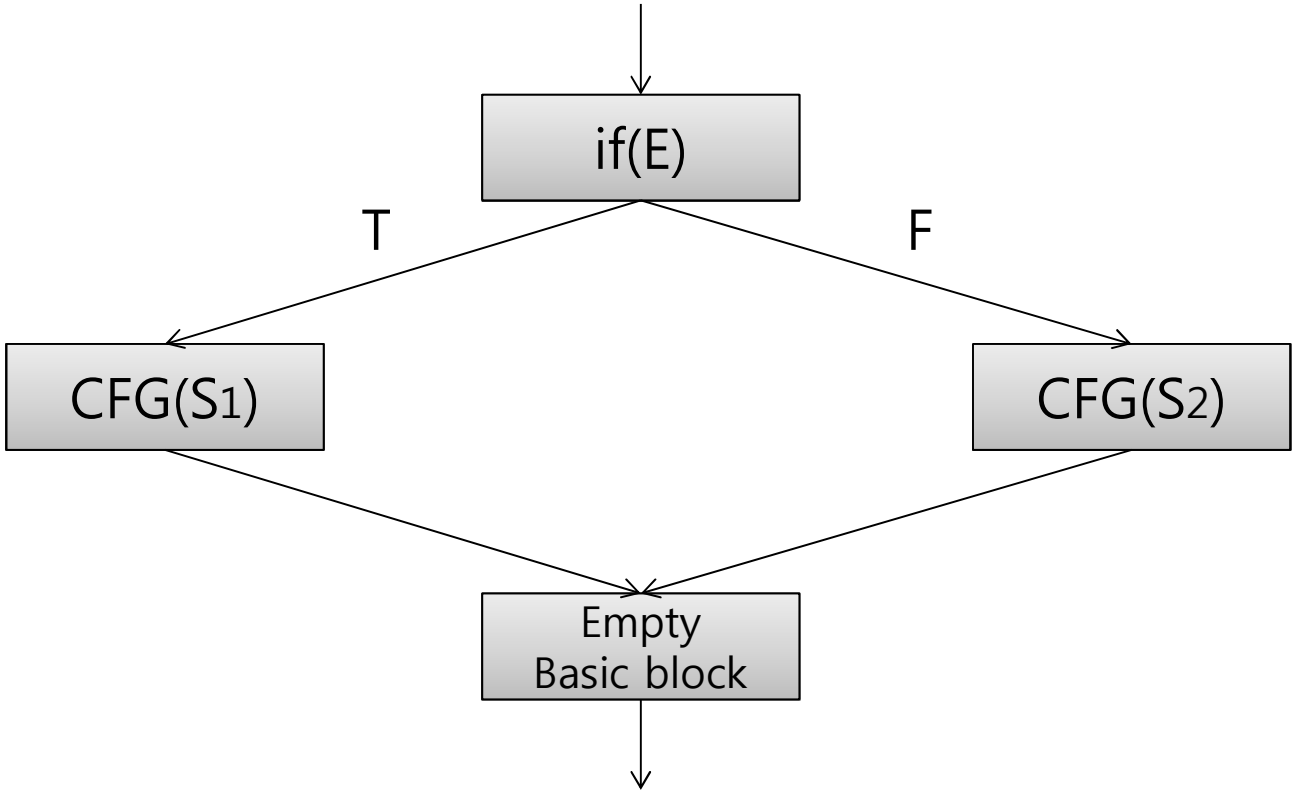
  - CFG(S_1; S_2; … ; S_N) =



2. Building the CFG – (1) High-level IR (Cont.)

- CFG for If-then-else Statement

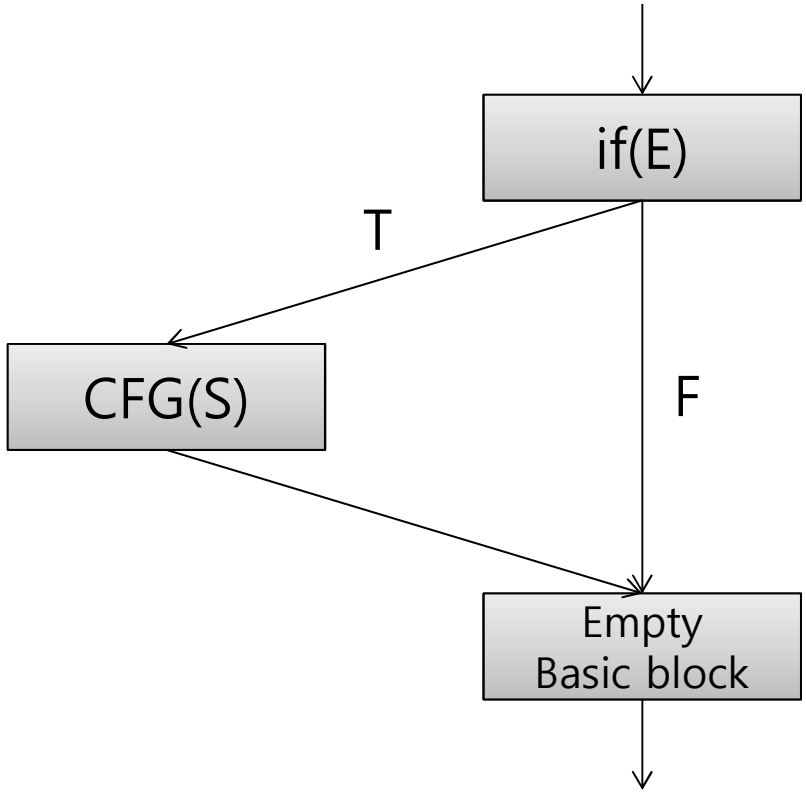
  - CFG(if(E) S_1 else S_2) =



2. Building the CFG – (1) High-level IR (Cont.)

- CFG for If-then Statement

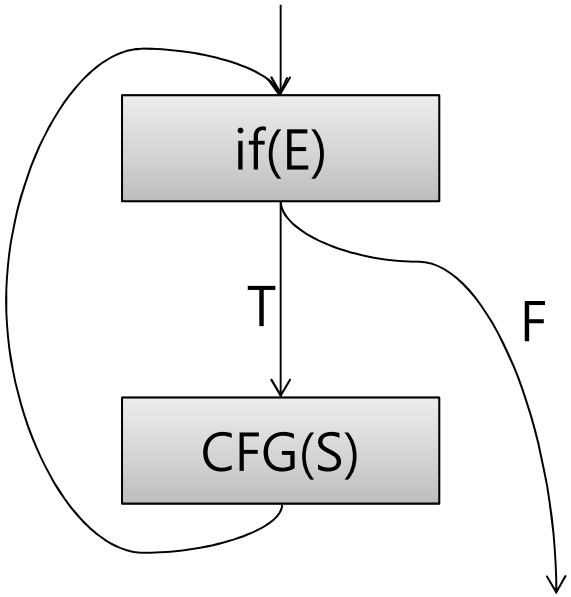
  - CFG(if(E) S) =



2. Building the CFG – (1) High-level IR (Cont.)

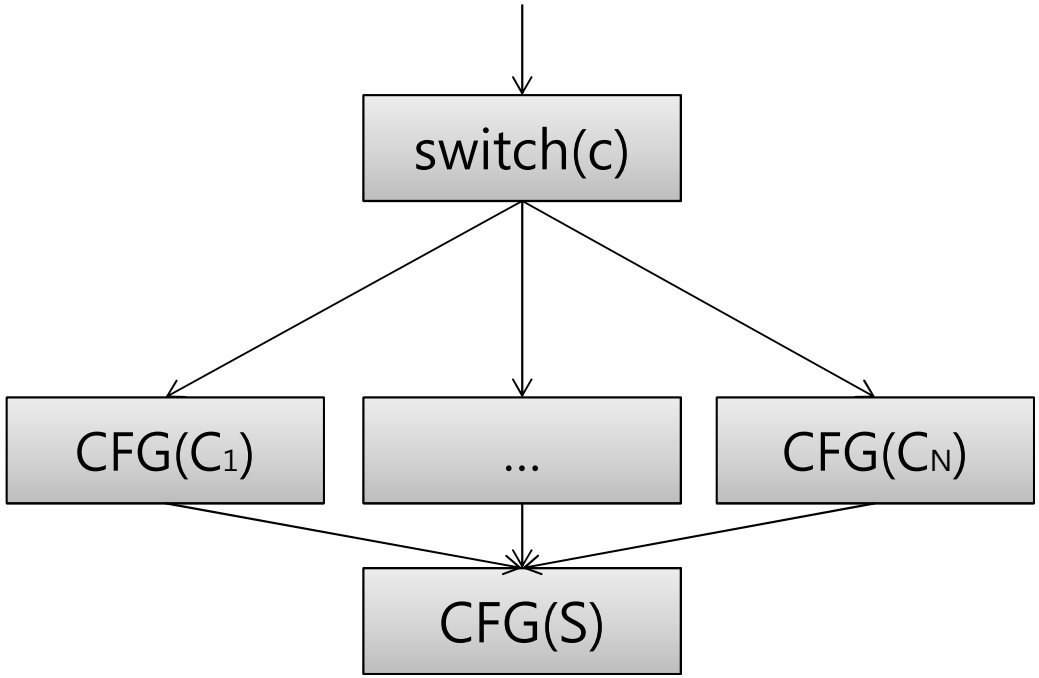
- CFG for While Statement

  - CFG(while(E) S) =



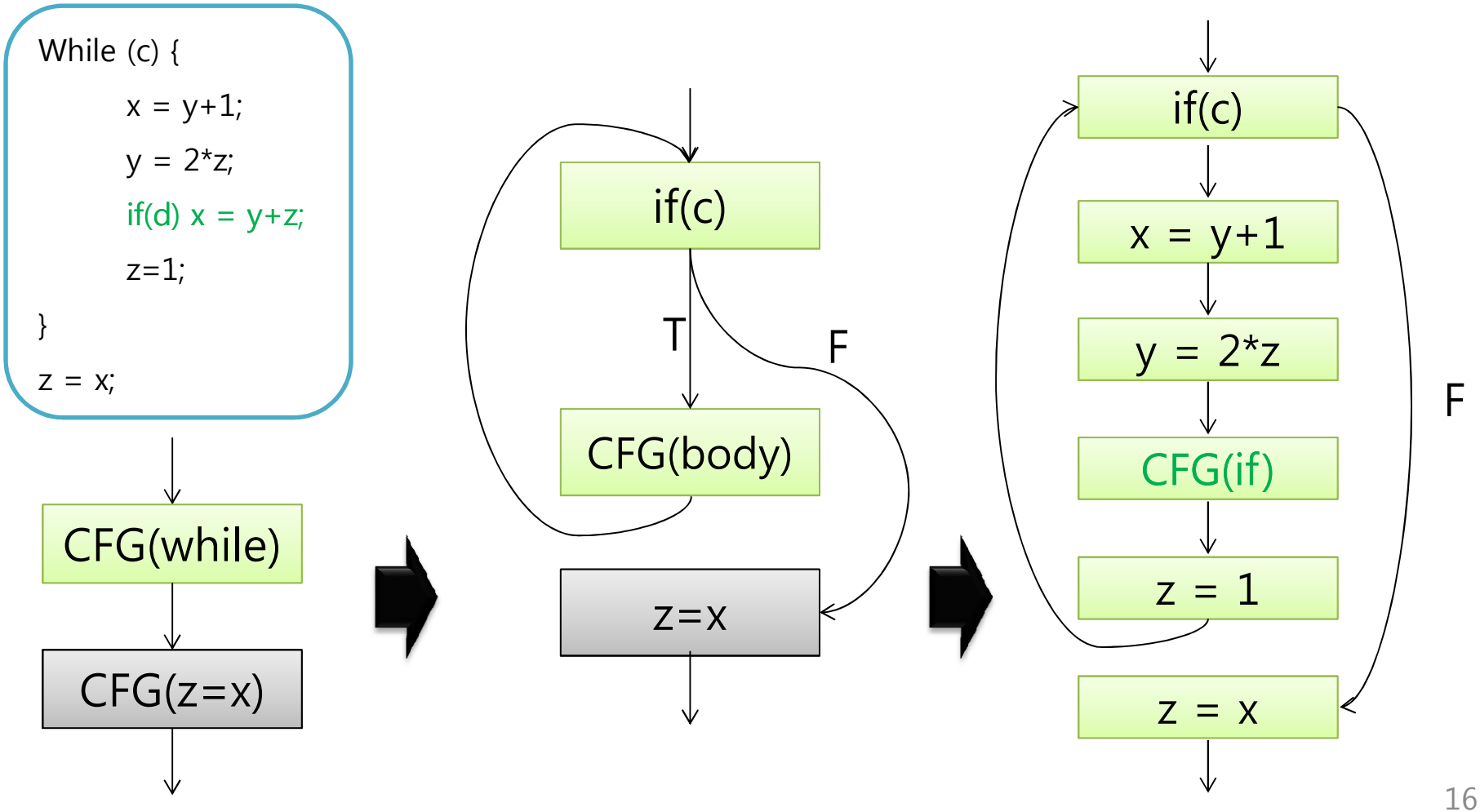
2. Building the CFG – (1) High-level IR (Cont.)

- CFG for Case-of Statement
  - CFG(switch(c);S) =



2. Building the CFG – (1) High-level IR (Cont.)

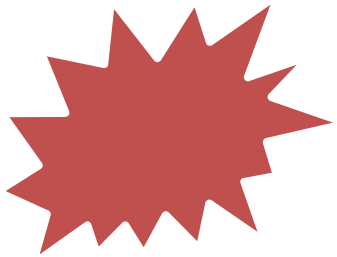
- Nested statements : recursively construct CFG



2. Building the CFG – (2) Efficient CFG

Efficient CFG

- Basic blocks :
  - as **few** as possible
  - as **large** as possible



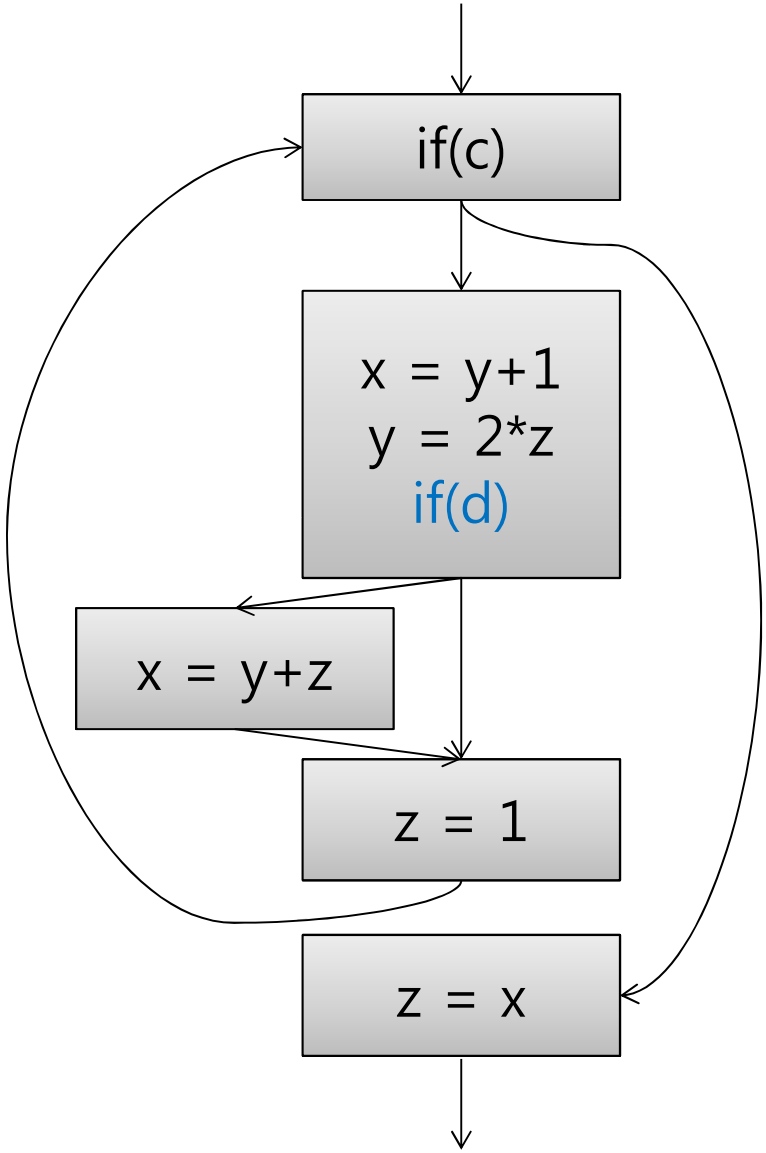
Inefficient CFG

- **Small basic blocks = many nodes**
  - time- and space-consuming ↑
- **Empty basic blocks**

2. Building the CFG – (2) Efficient CFG (Cont.)

- 예제)

```
While (c) {
        x = y+1;
        y = 2*z;
        if(d) x = y+z;
        z=1;
}
z = x;
```



2. Building the CFG – (3) Low-level IR

- Basic block = **L1; L2; L3; …. ;**
  - **Non-branching** instructions
  - **Non-label** instructions

Label L1
fjump c L2
x = y+1;
y= 2*z;
fjump d L3
x = y+z;
Label L3
z = 1;
jump L1
Label L2
z = x;

Label L1
fjump c L2

x = y+1;
y = 2*z;
fjump d L3

x = y+z;

Label L3
z = 1;
Jump L1

Label L2
z = x;

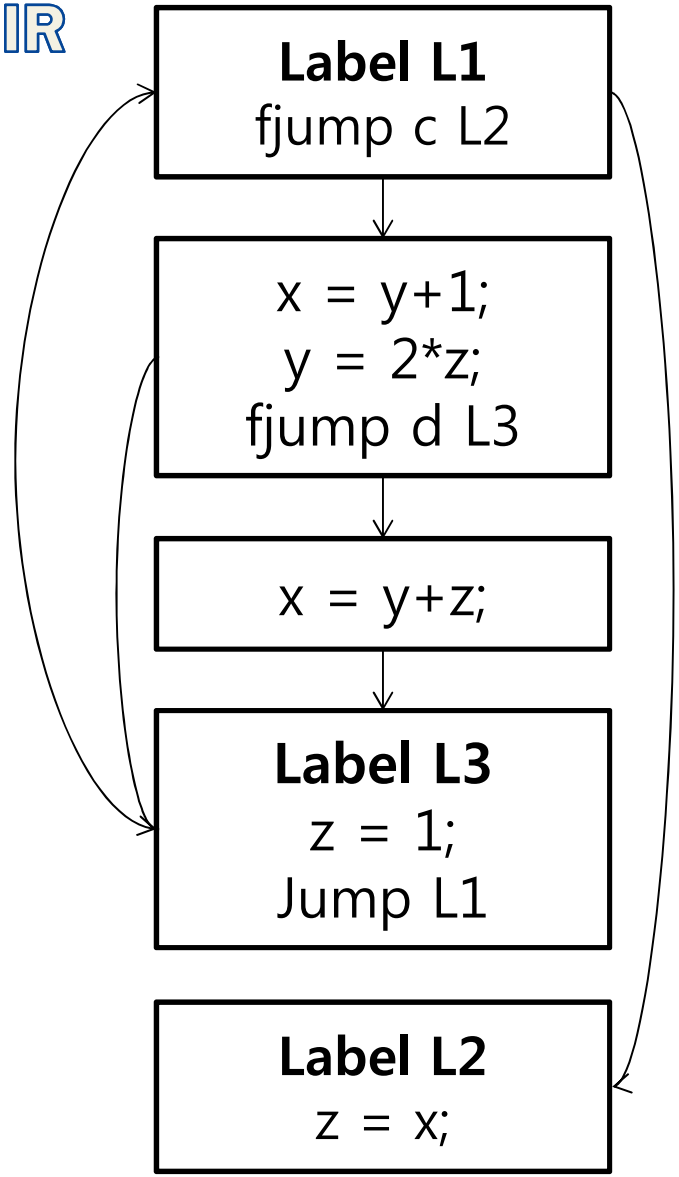
2. Building the CFG – (3) Low-level IR

- Conditional jump

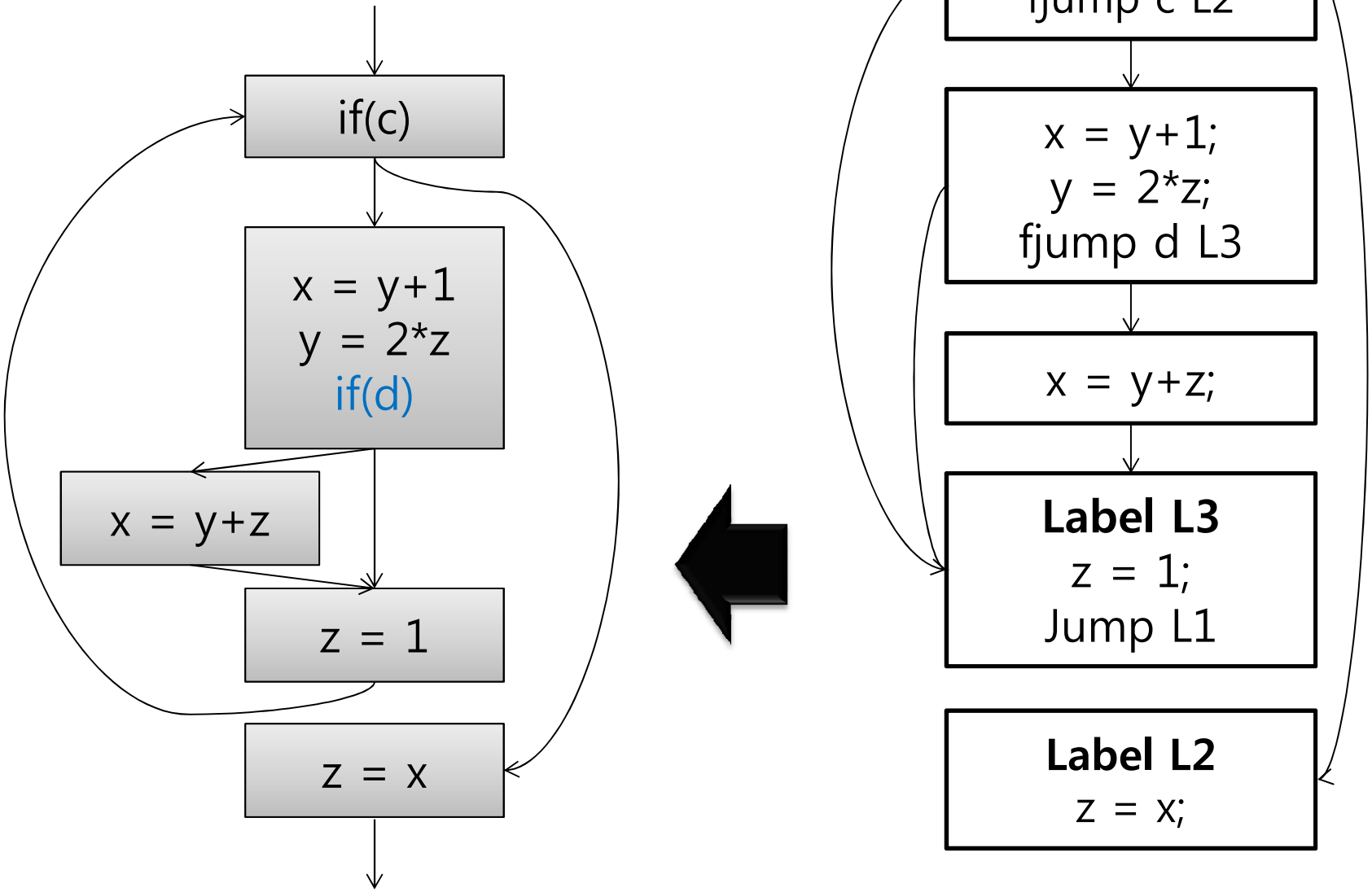
  - 2 successors

- Unconditional jump

  - 1 successor



2. Building the CFG – (3) Low-level IR



3. Statement of Purpose

Control Flow Graph Constructor for C programs

- C언어로 작성된 프로그램의 CFG를 작성
- 제어 Statement를 중점으로 하여 부합하는 알고리즘을 따른다
- C언어를 입력 받는 방법으로 파일이나 직접입력을 통한 방법 두가지 중 선택가능
- 구문 오류로 문제점 발생시 에러메시지 출력
- 무한루프 등의 논리적 오류는 탐지하지 못함

The End

감 사 합 니 다 !