두 코드의 의미차이를 보여주는 입력예제 안전하게 찾기 – PL/SQL을 대상으로

ROPAS 강동옥, 이영석, 이우석

프로젝트 목표

- PL/SQL 로 쓰여진 MES 코드가 변경되었을 때, 변경 내역을 잘 설명하기

설명하는 방법

- 입력과 출력 변수가 동일한 두 코드에 대해서 (예전과 새 revision) 서로 다른 출력을 야기시키는 DB테이블 집합의 입력 예제를 모두 찾아서 보여준다.

오라클 PL/SQL 언어?

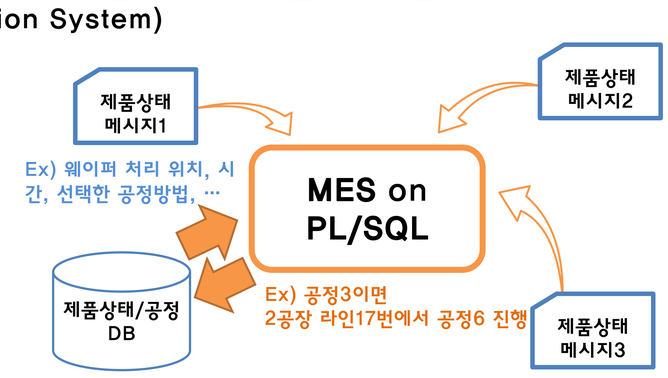
- 절차 언어 + SQL 쿼리문
- 변수, 함수, 루프, 조건문(IF), 예외처리등 절차언어의 특징을 대부분 가짐.

반도체 공정에서 MES?

제품공정 DB를 공정규칙에 따라 업데이트하는데 쓰임

MES(Manufacturing Execution System)

- 제품 상태 추적
- 반도체 제조 스케쥴링
- 공정 예외상황 처리
- 제품 저장고 관리
- 제품 수량 조절
- 제품 이동 관리



분석 결과로 무엇을 보여줄까?

```
OLD REVISION >

SELECT COUNT(*) FROM table INTO
var_count
WHERE
  table.year > 2000

SELECT MIN(id) FROM table INTO min_id

IF var_count > 0
BEGIN

IF min_id > 10 THEN
 UPDATE INTO table SET id = id * 2
 WHERE table.dept = "CSE"

IF MOD(var_count,2) = 0 THEN
 UPDATE INTO table SET id = id + 1
 WHERE table.dept = "CSE"

END
```

```
NEW REVISION >

SELECT COUNT(*) FROM table INTO
var_count
WHERE
  table.year > 2000
  AND table.dept = "CSE"

SELECT MIN(id) FROM table INTO min_id

BEGIN

IF MOD(var_count,2) = 0 THEN
 UPDATE INTO table SET id = id + 1
 WHERE table.dept = "CSE"

IF min_id > 10 THEN
 UPDATE INTO table SET id = id * 2
 WHERE table.dept = "CSE"

END
```

테이블 결과가 달라지는 **입력**을 **안전하게** 찾아 보여준다.

입력 테이블

id	year	dept
11	1999	CSE

id	year	dept
11	2001	CSE
12	2001	CSE

…

OLD REV. 결과 테이블

id	year	dept
11	1999	CSE

id	year	dept
23	2001	CSE
25	2001	CSE

…

NEW REV. 결과 테이블

id	year	dept
24	1999	CSE

id	year	dept
24	2001	CSE
26	2001	CSE

…

대상 프로그램의 크기

- 전체LOC : 약 10만라인
- 중요모듈LOC : 11,000
- 한 모듈에서 사용하는 변수의 수 : 400개

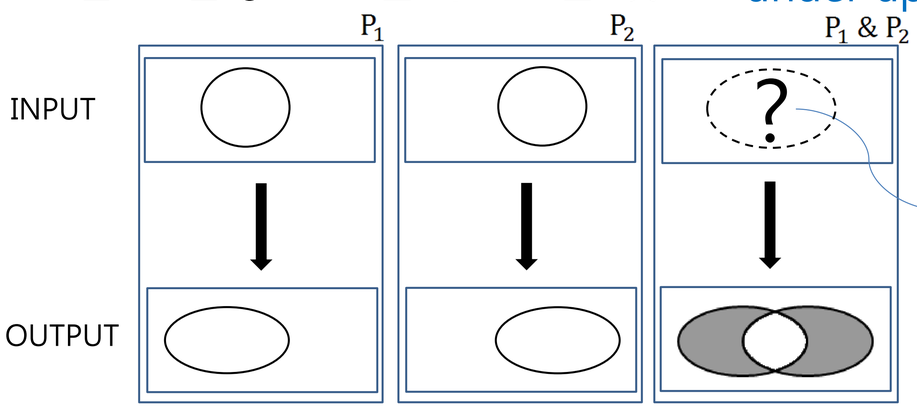
대상 프로그램의 특징

- 조건문과 DB업데이트가 대부분
- 재귀함수, 트랜잭션, 테이블 조인 사용되지 않음
- 반복문은 대부분 횟수가 정해져 있고, 주로 테이블의 값을 변수로 복사하는 일
- 산술연산, 문자열, 날짜 연산이 조건문에 자주 쓰임
- 문자열 관련 조건식은 주로 prefix, suffix, substring을 검사

분석 프레임워크

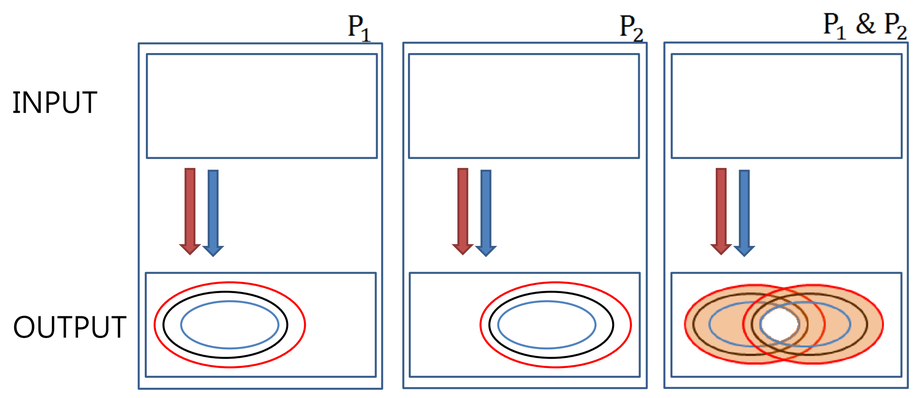
-실제실행
-over-approximation
-under-approximation

1. 실제 실행에서 분석하려는 것

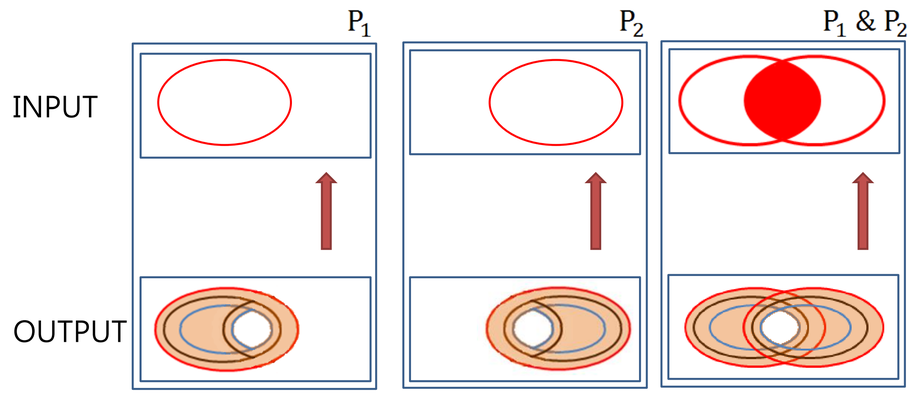


가능한 input중 **어떤 input**이 **다른 output**을 만들어내는가?

2. 전방분석

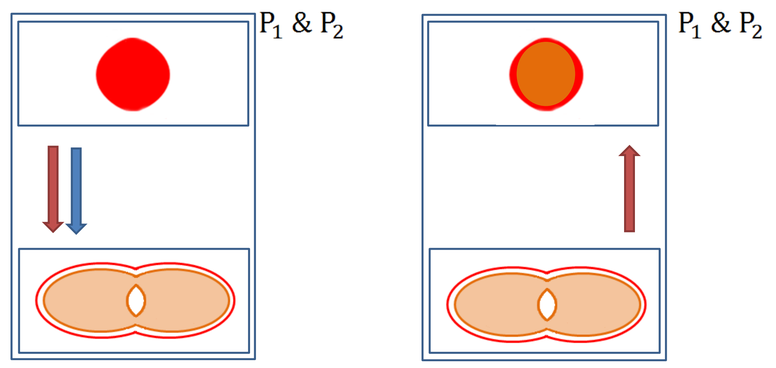


3. 후방분석



over-approximation과 under-approximation을 동시에 하여 **정교하고 안전한 분석 결과**를 얻을 수 있다.

4. 좀더 정교하게



분석 후에 얻어진 결과를 가지고, 전방분석과 후방분석을 반복하여 더욱더 정교해진 결과(안전한)를 얻을 수 있다. (forward–backward abstraction refinement)

PL/SQL 핵심 언어 정의 (Abstract Syntax)

variable	x	$\in$	ordinary variables	
	a	$\in$	attributes	
	t	$\in$	table variables	
	i	$=$	the input variable	
program	pgm	$\rightarrow$	$decl^* c$	
declaration	$decl$	$\rightarrow$	$\tau\ x$	
		$\mid$	$\rho\ table\ t$	(SQL table : record set)
type identifier	τ	$\rightarrow$	INT $\mid$ REAL $\mid$ STRING $\mid$ EXN	(primitive type)
		$\mid$	ρ	(record type)
		$\mid$	$\tau\ array$	
		$\mid$	$\rho\ table$	
	ρ	$\rightarrow$	$\{a_1 : \tau_1, \ldots, a_n : \tau_n\}$	
command	c	$\rightarrow$	$skip \mid c; c$	
		$\mid$	if r c c	
		$\mid$	for x in set c	
		$\mid$	$x := e$	
		$\mid$	$t := sql$	
		$\mid$	$x.a := e$	
		$\mid$	$x[e] := e$	
		$\mid$	raise x	
		$\mid$	try c handle $match$	
set	set	$\rightarrow$	$e..e$	(integer set)
		$\mid$	sql	(record set)
pattern match	$match$	$\rightarrow$	$x \Rightarrow c\ \langle \mid match \rangle$	
		$\mid$	default $\Rightarrow c$	
sql query	sql	$\rightarrow$	t	
		$\mid$	SELECT r t	
		$\mid$	PROJECT $[a_1, \ldots, a_n]$ t	
		$\mid$	RENAME $[a_1, \ldots, a_n]$ t	
		$\mid$	$t_1 \oplus t_2$ $\quad (\oplus \in \{\cup, -, *\})$	
		$\mid$	$[a_1 = e_1, \ldots, a_n = e_n]$	(record singleton set)
expression	e	$\rightarrow$	$const \mid e + e \mid x \mid t \mid x[e] \mid x.a \mid t.a \mid sql \mid \|e\|$	
		$\mid$	$f(e)$	(built-in function call)
constant	$const$	$\in$	$\{\text{NULL}\} \cup \mathbb{R} \cup \text{strings}$	
relation	r	$\rightarrow$	$e{=}e \mid e{<}e \mid !r \mid r \wedge r \mid$	

전방분석/후방분석

EX) $\{P\}x := x + 1\{Q\}$

- **전방분석**
  P 가 주어졌을때 Q 를 구하는것
- **후방분석**
  Q 가 주어졌을때 P 를 구하는것

앞으로 할일 & Challenge

- 요약 도메인 설계 – 테이블을 어떻게 요약할 것인가?
  - ➢ 차집합, 교집합을 구하기 용이해야 한다.
- 복잡한 PL/SQL 구문들을 핵심 언어로 변환
- 전방분석 분석기 설계
  안전한요약(over approximation),
  완전한요약(under approximation)
- 후방분석 분석기 설계

Programming Research Laboratory
SEOUL NATIONAL UNIVERSITY
ROSAEC center
Research On Software Analysis for Error-free Computing