KOREA UNIVERSITY

프로그래밍 언어 연구란?

KUGODS 연사 초청 특강

박지혁

고려대학교 정보대학 컴퓨터학과
프로그래밍 언어 연구실

2023.05.12

중학교

절차지향 언어
포인터

...

알고리즘 대회

중학교

고등학교

절차지향 언어
포인터
…

객체지향 언어
가비지 컬렉션
…

알고리즘 대회

전산학 수업

중학교

고등학교

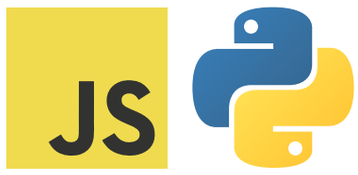
학부

절차지향 언어
포인터

...

객체지향 언어
가비지 컬렉션

...



스크립트 언어
동적 타입

...

알고리즘 대회

전산학 수업

개발 동아리

중학교

고등학교

학부

대학원

회사 연구원

교수

절차지향 언어
포인터

...

객체지향 언어
가비지 컬렉션

...

스크립트 언어
동적 타입

...

객체지향 + 함수형 언어
JVM 기반 언어

...

알고리즘 대회

전산학 수업

개발 동아리

연구 및 개발

프로그래밍 언어 연구를 시작하게 된 계기

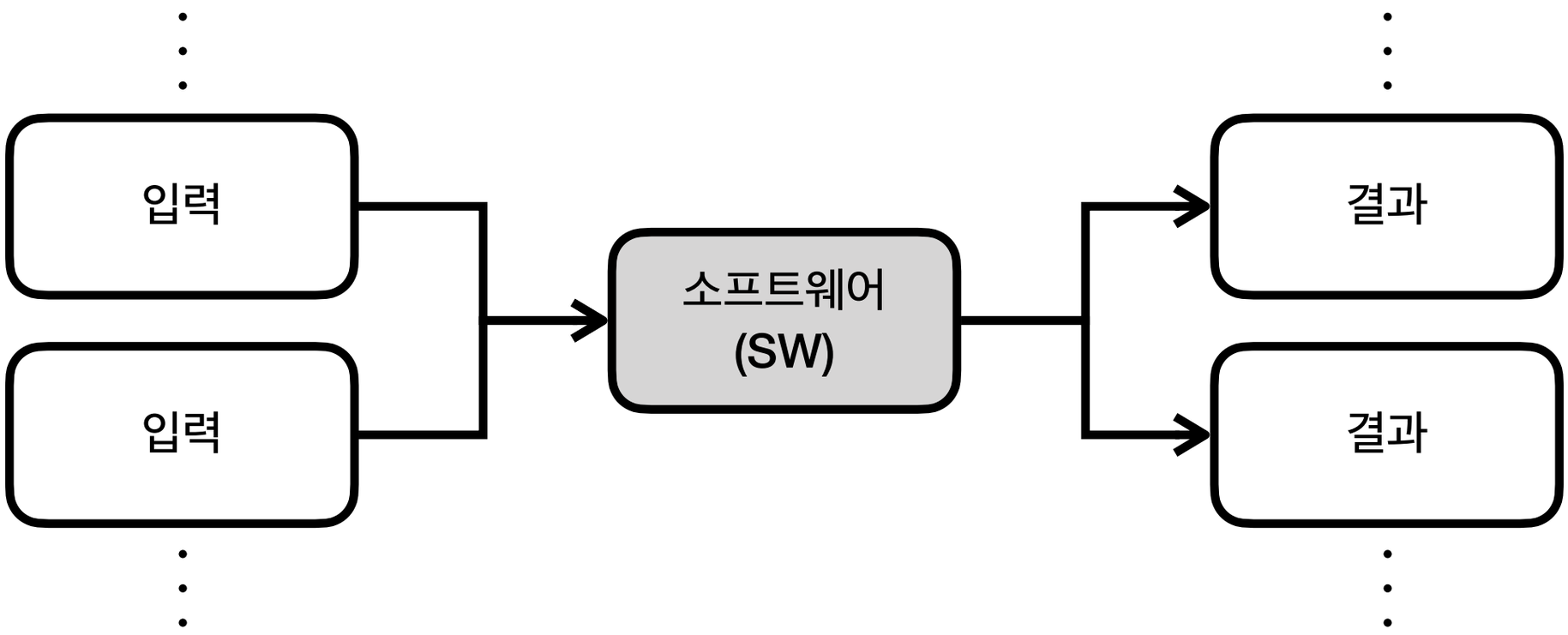
KAIST **류석영** 교수님

프로그래밍 언어 연구를 시작하게 된 계기

KAIST **류석영** 교수님

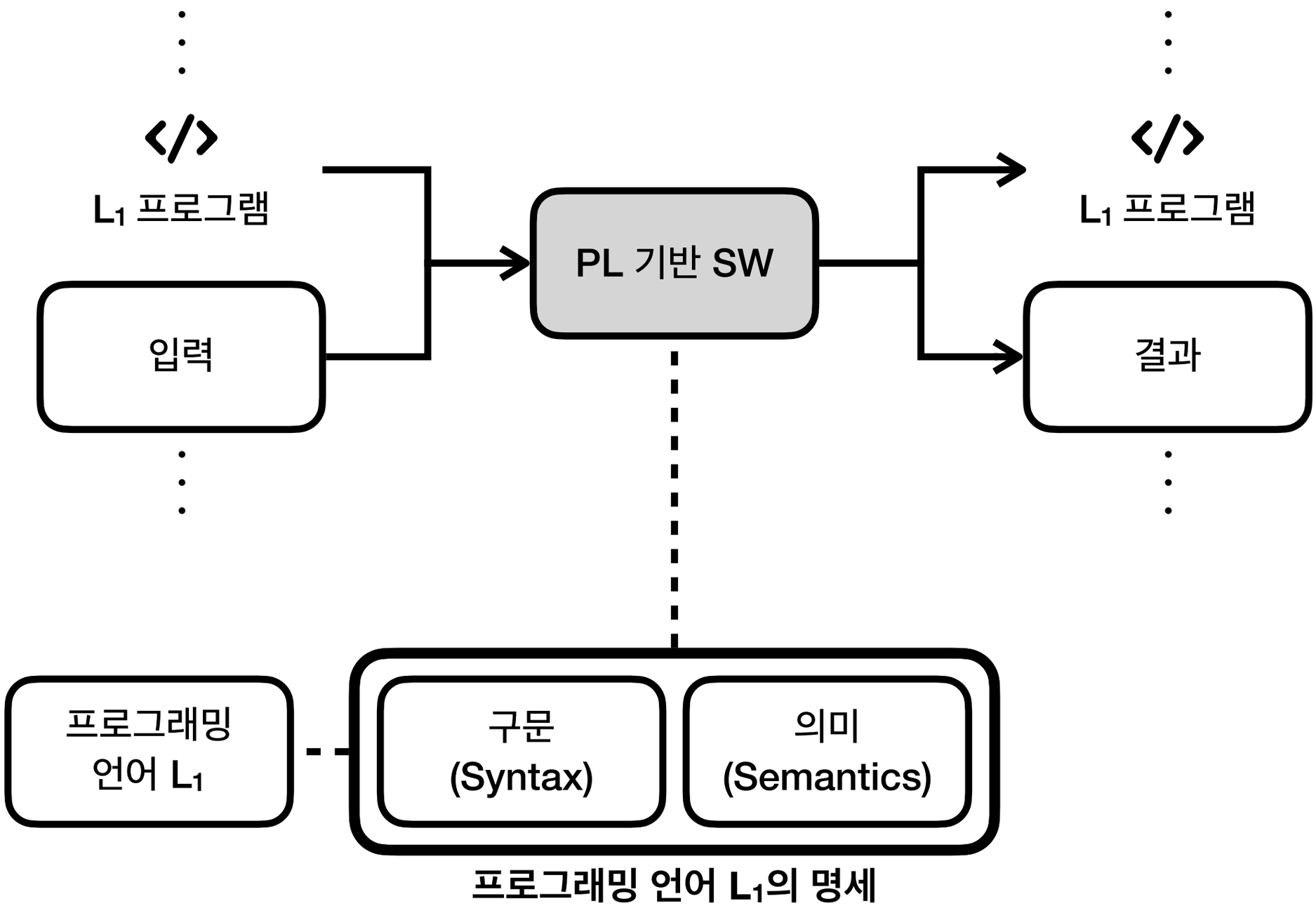
> **자바스크립트**로 개발한 소프트웨어가 **올바르게 동작**할지 확신할 수 있나요?

소프트웨어란?



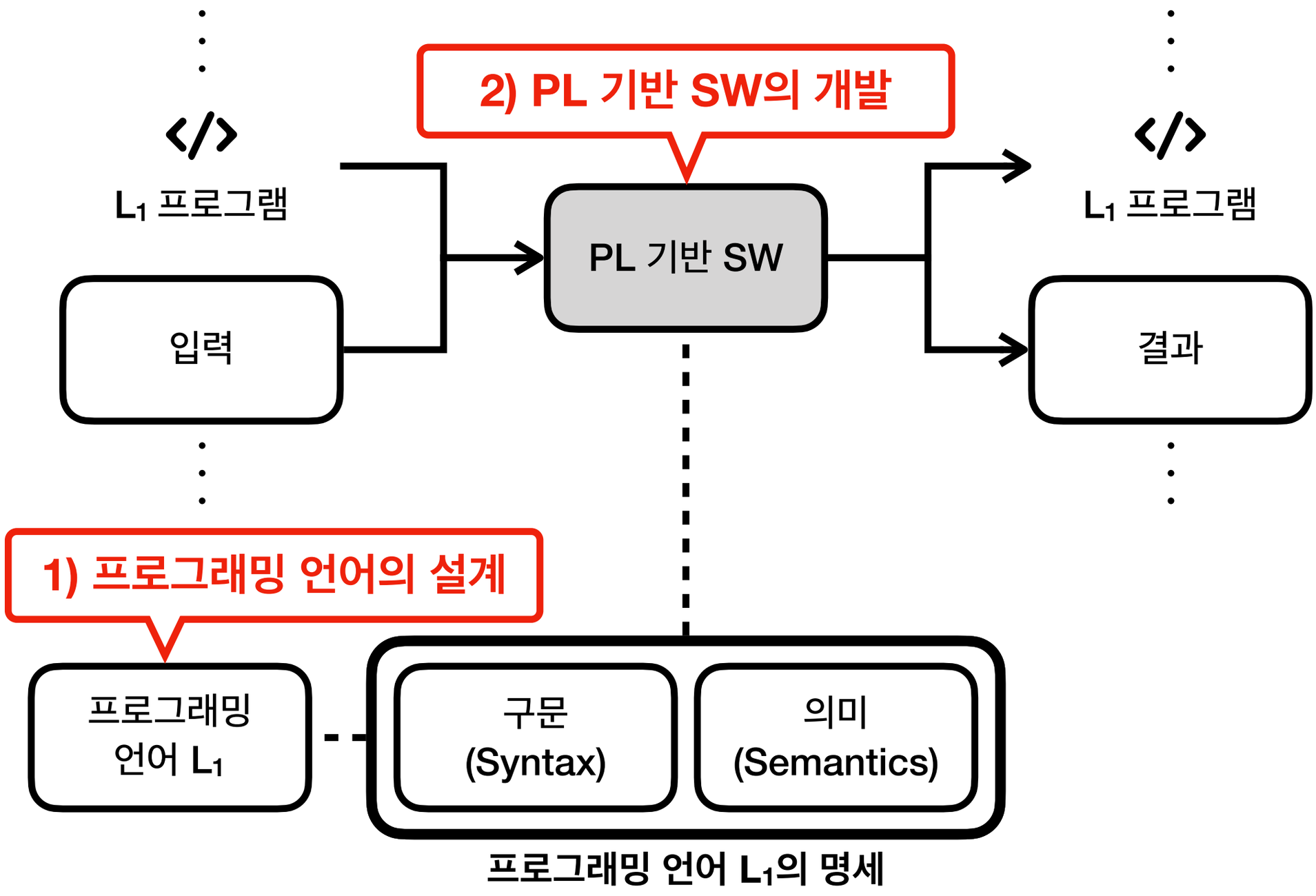

```
def add(x: Int, y: Int): Int = x + y
```

프로그래밍 언어 기반 SW란?

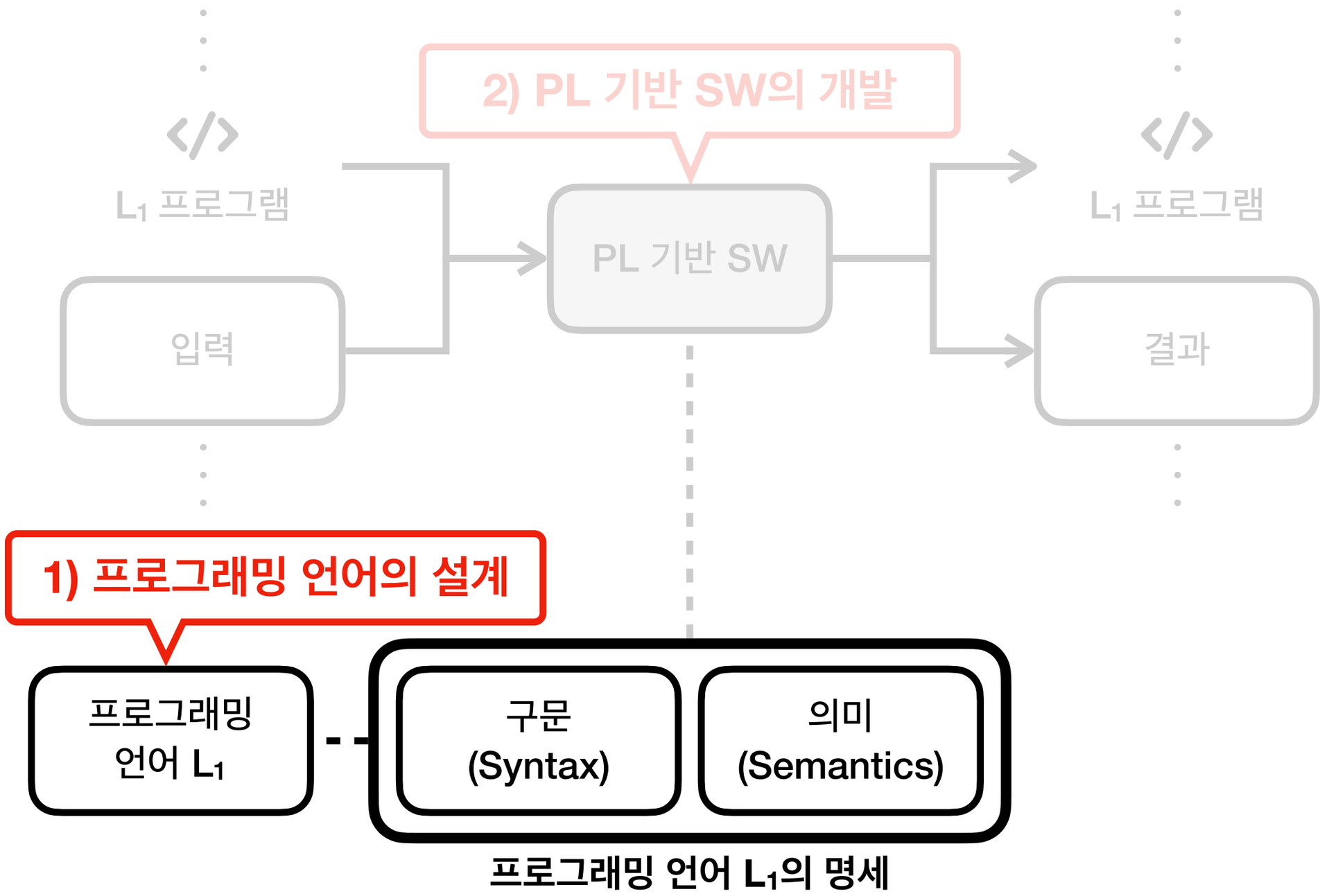


프로그래밍 언어 L_1의 명세

프로그래밍 언어 연구란?


2) PL 기반 SW의 개발
L1 프로그램
PL 기반 SW
L1 프로그램
입력
결과
1) 프로그래밍 언어의 설계
프로그래밍
언어 L1
구문
(Syntax)
의미
(Semantics)
프로그래밍 언어 L1의 명세

프로그래밍 언어 연구란?


2) PL 기반 SW의 개발
L1 프로그램
PL 기반 SW
입력
결과
L1 프로그램
1) 프로그래밍 언어의 설계
프로그래밍
언어 L1
구문
(Syntax)
의미
(Semantics)
프로그래밍 언어 L1의 명세

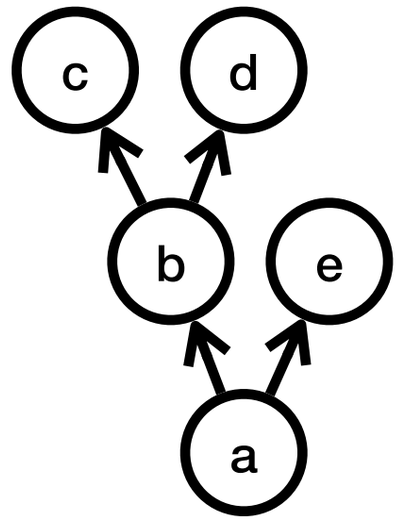
프로그래밍 언어의 설계

- **프로그래밍 패러다임** (Programming Paradigm)
  - 객체지향 (OOP) / 함수형 (FP) / 명령형 (Imperative) / ...
  - 선언형 (Declarative) / 미분 가능 (Differentiable) / …
  - e.g., **Datalog** - 선언형 논리 프로그래밍 언어

프로그래밍 언어의 설계

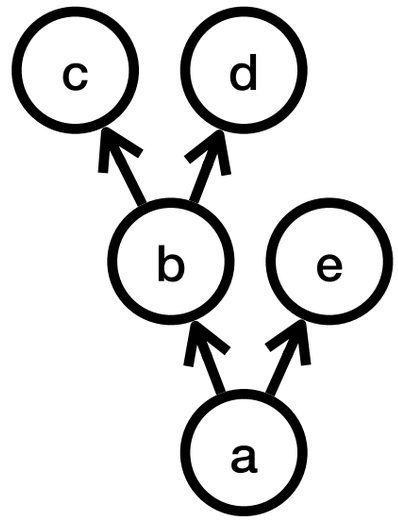
- **프로그래밍 패러다임** (Programming Paradigm)
  - 객체지향 (OOP) / 함수형 (FP) / 명령형 (Imperative) / ...
  - 선언형 (Declarative) / 미분 가능 (Differentiable) / …
  - e.g., **Datalog** - 선언형 논리 프로그래밍 언어

```
// initial facts
parent(a, b). parent(b, c).
parent(b, d). parent(a, e).
```



프로그래밍 언어의 설계

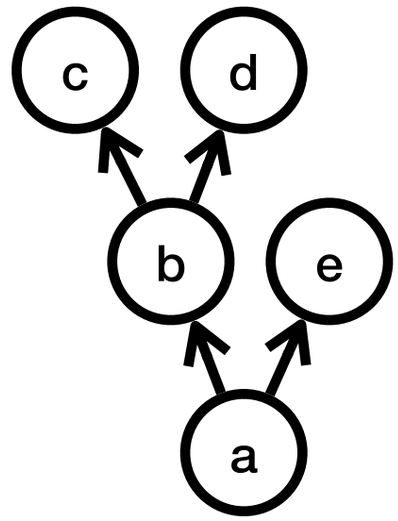
- **프로그래밍 패러다임** (Programming Paradigm)
  - 객체지향 (OOP) / 함수형 (FP) / 명령형 (Imperative) / ...
  - 선언형 (Declarative) / 미분 가능 (Differentiable) / …
  - e.g., **Datalog** - 선언형 논리 프로그래밍 언어




```
// initial facts
parent(a, b). parent(b, c).
parent(b, d). parent(a, e).

// rules
ancestor(X, Y) :- parent(X, Y).
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).
```

프로그래밍 언어의 설계

- **프로그래밍 패러다임** (Programming Paradigm)
  - 객체지향 (OOP) / 함수형 (FP) / 명령형 (Imperative) / ...
  - 선언형 (Declarative) / 미분 가능 (Differentiable) / …
  - e.g., **Datalog** - 선언형 논리 프로그래밍 언어




```
// initial facts
parent(a, b). parent(b, c).
parent(b, d). parent(a, e).

// rules
ancestor(X, Y) :- parent(X, Y).
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).

// derive facts
?- ancestor(a, d). // true
?- ancestor(e, b). // false
```

프로그래밍 언어의 설계

- **타입 시스템** (Type System)
  - Gradual Type (**TypeScript**) / Dependent Object Type (**Scala**) / …
  - Ownership (**Rust**) / Refinement Type (**LiquidHaskell**) / …
  - e.g., **Refinement Type** - 조건을 추가해서 더 정확하게 표현해보자

프로그래밍 언어의 설계

- **타입 시스템** (Type System)
  - Gradual Type (**TypeScript**) / Dependent Object Type (**Scala**) / …
  - Ownership (**Rust**) / Refinement Type (**LiquidHaskell**) / …
  - e.g., **Refinement Type** - 조건을 추가해서 더 정확하게 표현해보자

```
0 + true
```

프로그래밍 언어의 설계

- **타입 시스템** (Type System)
  - Gradual Type (**TypeScript**) / Dependent Object Type (**Scala**) / …
  - Ownership (**Rust**) / Refinement Type (**LiquidHaskell**) / …
  - e.g., **Refinement Type** - 조건을 추가해서 더 정확하게 표현해보자

```
0 + true
```

Compile Error - `+` not for (Int, Boolean)

프로그래밍 언어의 설계

- **타입 시스템** (Type System)
  - Gradual Type (**TypeScript**) / Dependent Object Type (**Scala**) / …
  - Ownership (**Rust**) / Refinement Type (**LiquidHaskell**) / …
  - e.g., **Refinement Type** - 조건을 추가해서 더 정확하게 표현해보자

```
0 + true
```

Compile Error - `+` not for (Int, Boolean)

```
def div(x: Int, y: Int): Int = x / y
div(4, 0)
```

프로그래밍 언어의 설계

- **타입 시스템** (Type System)
  - Gradual Type (**TypeScript**) / Dependent Object Type (**Scala**) / …
  - Ownership (**Rust**) / Refinement Type (**LiquidHaskell**) / …
  - e.g., **Refinement Type** - 조건을 추가해서 더 정확하게 표현해보자

```
0 + true
```

Compile Error - `+` not for (Int, Boolean)

```
def div(x: Int, y: Int): Int = x / y
div(4, 0)
```

Runtime Error - division by zero

프로그래밍 언어의 설계

- **타입 시스템** (Type System)
  - Gradual Type (**TypeScript**) / Dependent Object Type (**Scala**) / …
  - Ownership (**Rust**) / Refinement Type (**LiquidHaskell**) / …
  - e.g., **Refinement Type** - 조건을 추가해서 더 정확하게 표현해보자

```
0 + true
```

Compile Error - `+` not for (Int, Boolean)

```
def div(x: Int, y: Int): Int = x / y
div(4, 0)
```

Runtime Error - division by zero

```
def div(x: Int, y: {i: Int | i != 0}): Int = x / y
div(4, 0)
```

프로그래밍 언어의 설계

- **타입 시스템** (Type System)
  - Gradual Type (**TypeScript**) / Dependent Object Type (**Scala**) / …
  - Ownership (**Rust**) / Refinement Type (**LiquidHaskell**) / …
  - e.g., **Refinement Type** - 조건을 추가해서 더 정확하게 표현해보자

```
0 + true
```

Compile Error - `+` not for (Int, Boolean)

```
def div(x: Int, y: Int): Int = x / y
div(4, 0)
```

Runtime Error - division by zero

```
def div(x: Int, y: {i: Int | i != 0}): Int = x / y
div(4, 0)
```

Compile Error - `0` is not in `{i: Int | i != 0}`

프로그래밍 언어의 설계

- **도메인 특화 언어** (Domain Specific Language)

프로그래밍 언어의 설계

- **도메인 특화 언어** (Domain Specific Language)

프로그래밍 언어의 설계

- **도메인 특화 언어** (Domain Specific Language)

블록체인 - 스마트 컨트랙트

Solidity

Viper

하드웨어 기술 언어 (HDL)

Verilog

Bluespec VHDL

프로그래밍 언어의 설계

- **도메인 특화 언어** (Domain Specific Language)

블록체인 - 스마트 컨트랙트

Solidity

Viper

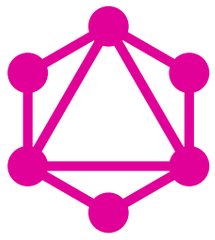
하드웨어 기술 언어 (HDL)

Verilog

Bluespec **VHDL**

쿼리 언어

SQL



GraphQL

CodeQL

프로그래밍 언어의 설계

- **도메인 특화 언어** (Domain Specific Language)

블록체인 - 스마트 컨트랙트

Solidity

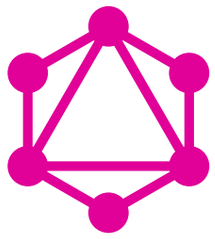
Viper

하드웨어 기술 언어 (HDL)

Verilog

Bluespec **VHDL**

쿼리 언어



GraphQL

SQL

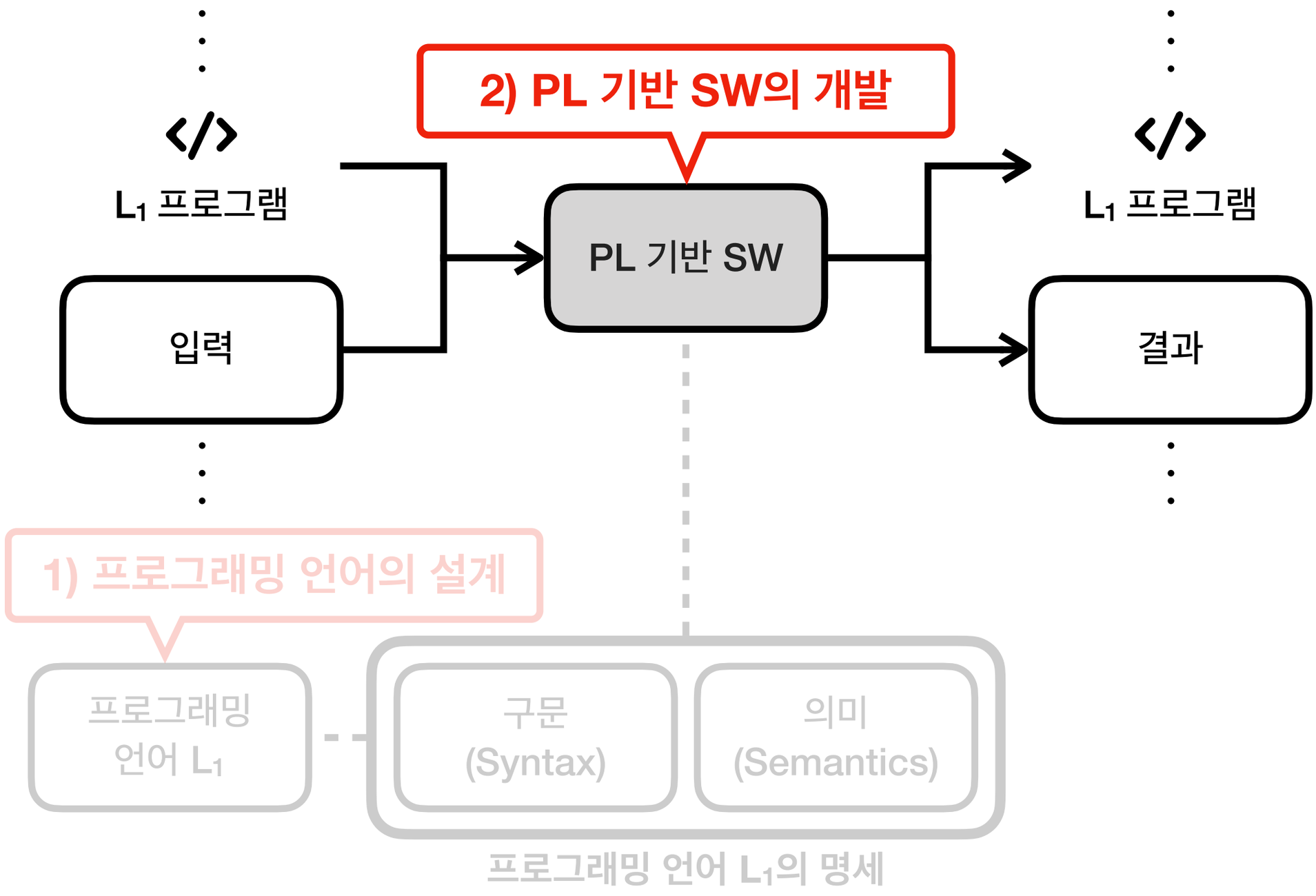
CodeQL

그 외

R
(통계 및 그래프)

Verse
(게임 시나리오)

프로그래밍 언어 연구


2) PL 기반 SW의 개발
L1 프로그램
PL 기반 SW
입력
L1 프로그램
결과
1) 프로그래밍 언어의 설계
프로그래밍
언어 L1
구문
(Syntax)
의미
(Semantics)
프로그래밍 언어 L1의 명세

PL 기반 SW의 개발

- **프로그램 분석** (Program Analysis)

동적 분석 - 퍼징 (Fuzzing)

```
function f(x) {
  if (x * 2 == 28322) {
    fail();
  } else if (x <= 0) {
    return 0;
  } else {
    return x;
  }
}
```

동적 분석 - 퍼징 (Fuzzing)

```
function f(x) {
  if (x * 2 == 28322) {
    fail();
  } else if (x <= 0) {
    return 0;
  } else {
    return x;
  }
}
```

```
f( 4) = 4
```

동적 분석 - 퍼징 (Fuzzing)

```
function f(x) {
  if (x * 2 == 28322) {
    fail();
  } else if (x <= 0) {
    return 0;
  } else {
    return x;
  }
}
```

```
f( 4) = 4

f(23) = 23
```

동적 분석 - 퍼징 (Fuzzing)

```
function f(x) {
  if (x * 2 == 28322) {
    fail();
  } else if (x <= 0) {
    return 0;
  } else {
    return x;
  }
}
```

```
f( 4) = 4
f(23) = 23
f(-3) = 0
```

동적 분석 - 퍼징 (Fuzzing)

```
function f(x) {
  if (x * 2 == 28322) {
    fail();
  } else if (x <= 0) {
    return 0;
  } else {
    return x;
  }
}
```

```
f( 4) = 4
f(23) = 23
f(-3) = 0
f( 5) = 5
```

동적 분석 - 퍼징 (Fuzzing)

```
function f(x) {
  if (x * 2 == 28322) {
    fail();
  } else if (x <= 0) {
    return 0;
  } else {
    return x;
  }
}
```

```
f( 4) = 4
f(23) = 23
f(-3) = 0
f( 5) = 5
...
```

정적 분석 - 기호 실행 (Symbolic Execution)

```
function f(x) {
  if (x * 2 == 28322) {
    fail();
  } else if (x <= 0) {
    return 0;
  } else {
    return x;
  }
}
```

정적 분석 - 기호 실행 (Symbolic Execution)

```
function f(x) {

  if (x * 2 == 28322) {
    // x * 2 == 28322
    fail();

  } else if (x <= 0) {
    return 0;

  } else {
    return x;

  }
}
```

정적 분석 - 기호 실행 (Symbolic Execution)

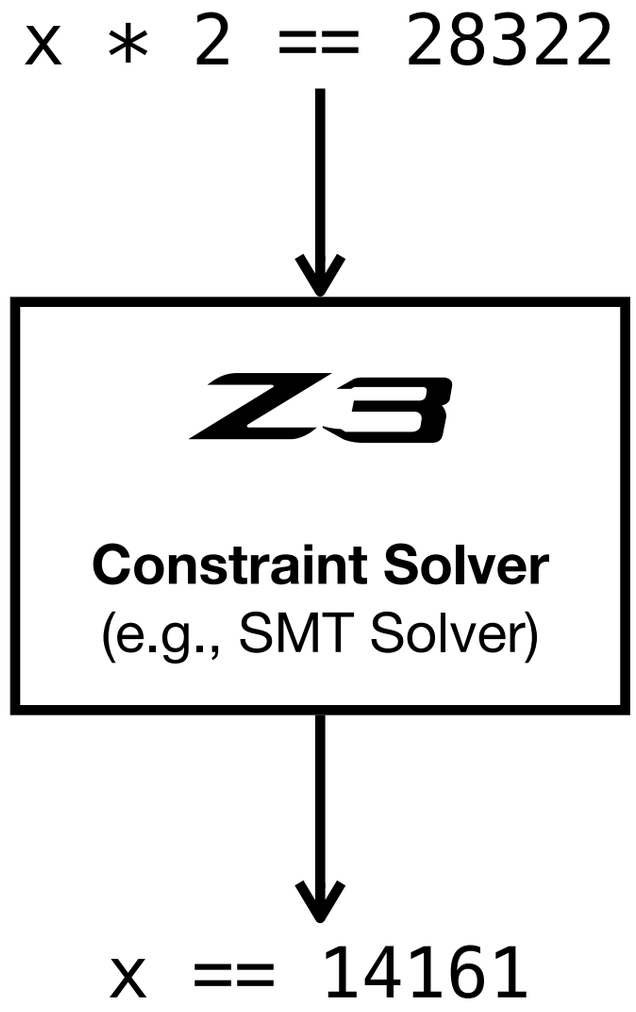
```
function f(x) {

  if (x * 2 == 28322) {
    // x * 2 == 28322
    fail();

  } else if (x <= 0) {
    // x * 2 != 28322 && x <= 0
    return 0;

  } else {

    return x;

  }
}
```

정적 분석 - 기호 실행 (Symbolic Execution)

```
function f(x) {

  if (x * 2 == 28322) {
    // x * 2 == 28322
    fail();

  } else if (x <= 0) {
    // x * 2 != 28322 && x <= 0
    return 0;

  } else {
    // x * 2 != 28322 && x > 0
    return x;

  }
}
```

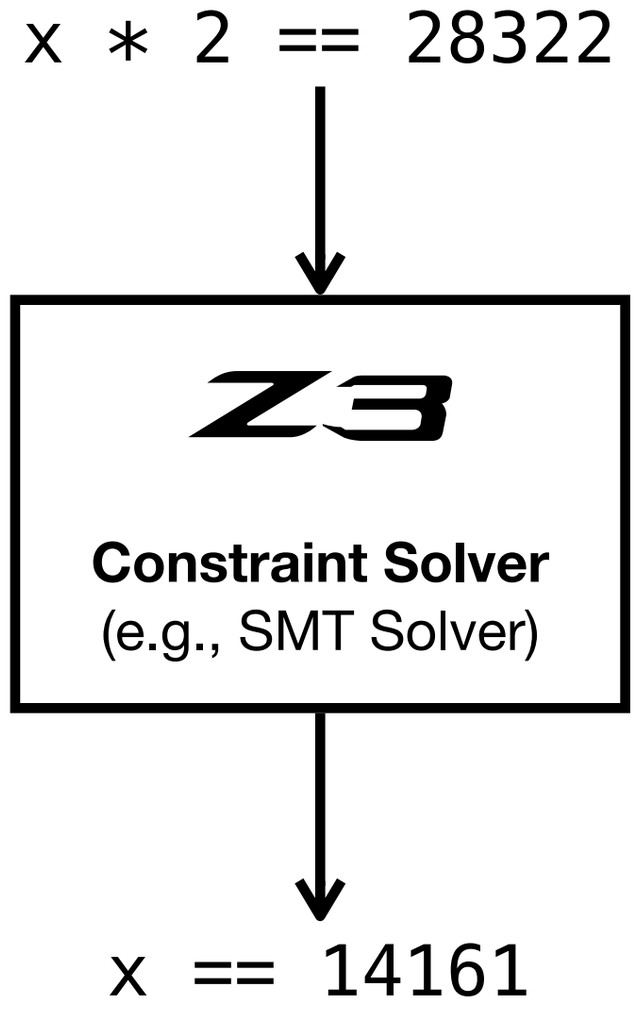
정적 분석 - 기호 실행 (Symbolic Execution)

```
function f(x) {

  if (x * 2 == 28322) {
    // x * 2 == 28322
    fail();

  } else if (x <= 0) {
    // x * 2 != 28322 && x <= 0
    return 0;

  } else {
    // x * 2 != 28322 && x > 0
    return x;

  }
}
```



정적 분석 - 기호 실행 (Symbolic Execution)

```
function f(x) {

  if (x * 2 == 28322) {
    // x * 2 == 28322
    fail();
    // Detect Error: 14161
  } else if (x <= 0) {
    // x * 2 != 28322 && x <= 0
    return 0;

  } else {
    // x * 2 != 28322 && x > 0
    return x;

  }
}
```

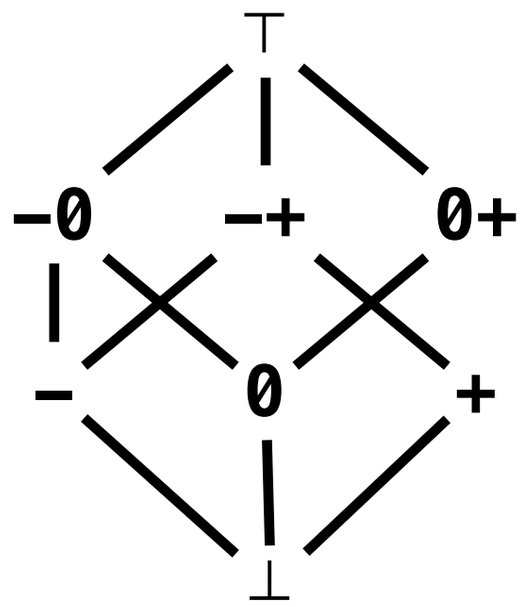


정적 분석 - 요약 해석 (Abstract Interp.)

```
function f(x) {
  if (x * 2 == 28322) {
    fail();
  } else if (x <= 0) {
    return 0;
  } else {
    return x;
  }
}
```

요약 도메인

(Abstract Domain)



⊤ : 모든 정수　⊥ : 공집합

− : 음수　0 : 0

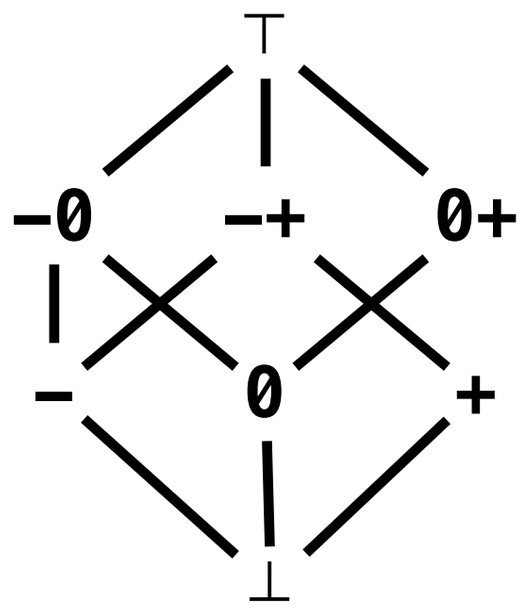
+ : 양수

정적 분석 - 요약 해석 (Abstract Interp.)

```
function f(x) {
  // x == ⊤
  if (x * 2 == 28322) {

    fail();

  } else if (x <= 0) {

    return 0;

  } else {

    return x;

  }
}
```

요약 도메인

(Abstract Domain)



⊤ : 모든 정수　⊥ : 공집합

- : 음수　0 : 0

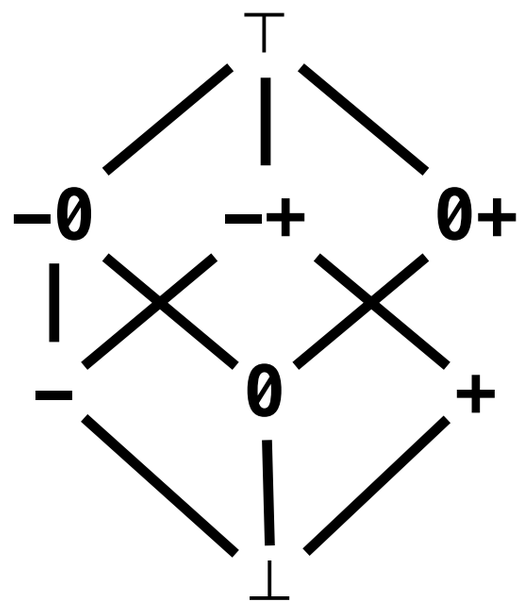
+ : 양수

정적 분석 - 요약 해석 (Abstract Interp.)

```
function f(x) {
  // x == ⊤
  if (x * 2 == 28322) {
    // x == ⊤
    fail();

  } else if (x <= 0) {

    return 0;

  } else {

    return x;

  }
}
```

요약 도메인

(Abstract Domain)



⊤ : 모든 정수　⊥ : 공집합

- : 음수　0 : 0

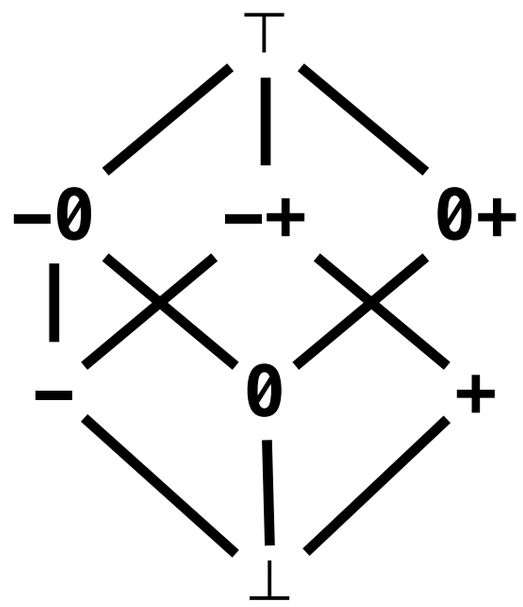
+ : 양수

정적 분석 - 요약 해석 (Abstract Interp.)

```
function f(x) {
  // x == ⊤
  if (x * 2 == 28322) {
    // x == ⊤
    fail();
    // May be Error
  } else if (x <= 0) {
    return 0;
  } else {
    return x;
  }
}
```

요약 도메인

(Abstract Domain)



⊤ : 모든 정수　⊥ : 공집합

- : 음수　0 : 0

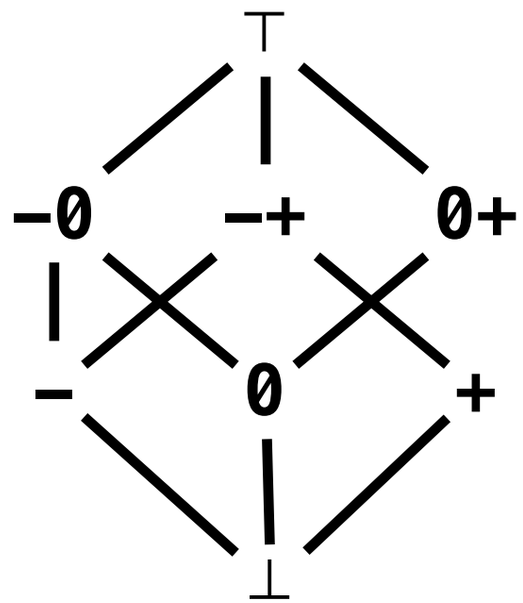
+ : 양수

정적 분석 - 요약 해석 (Abstract Interp.)

```
function f(x) {
  // x == ⊤
  if (x * 2 == 28322) {
    // x == ⊤
    fail();
    // May be Error
  } else if (x <= 0) {
    // x == -0
    return 0;

  } else {

    return x;

  }
}
```

요약 도메인

(Abstract Domain)



⊤ : 모든 정수　⊥ : 공집합

- : 음수　0 : 0

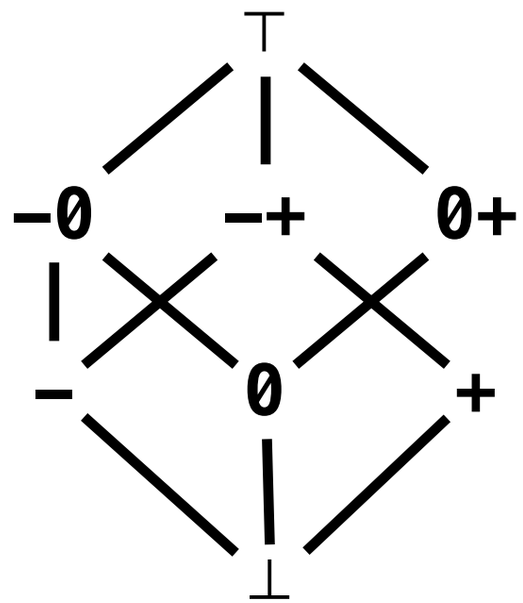
+ : 양수

정적 분석 - 요약 해석 (Abstract Interp.)

```
function f(x) {
  // x == ⊤
  if (x * 2 == 28322) {
    // x == ⊤
    fail();
    // May be Error
  } else if (x <= 0) {
    // x == -0
    return 0;
    // [RETURN] 0
  } else {

    return x;

  }
}
```

요약 도메인

(Abstract Domain)



⊤ : 모든 정수　⊥ : 공집합

- : 음수　0 : 0

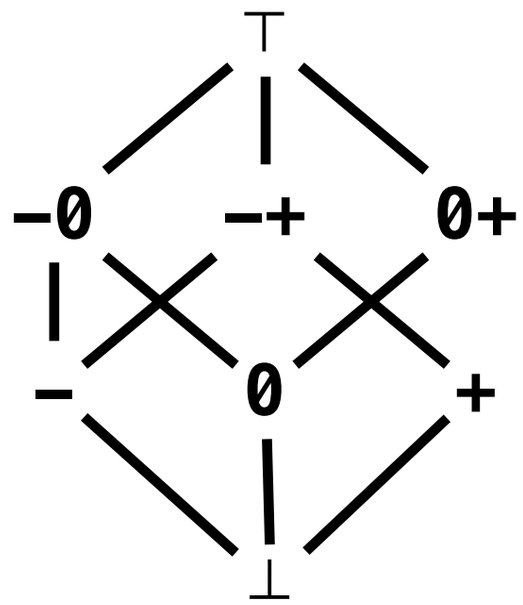
+ : 양수

정적 분석 - 요약 해석 (Abstract Interp.)

```
function f(x) {
  // x == ⊤
  if (x * 2 == 28322) {
    // x == ⊤
    fail();
    // May be Error
  } else if (x <= 0) {
    // x == -0
    return 0;
    // [RETURN] 0
  } else {
    // x == +
    return x;

  }
}
```

요약 도메인

(Abstract Domain)



⊤ : 모든 정수　⊥ : 공집합

- : 음수　0 : 0

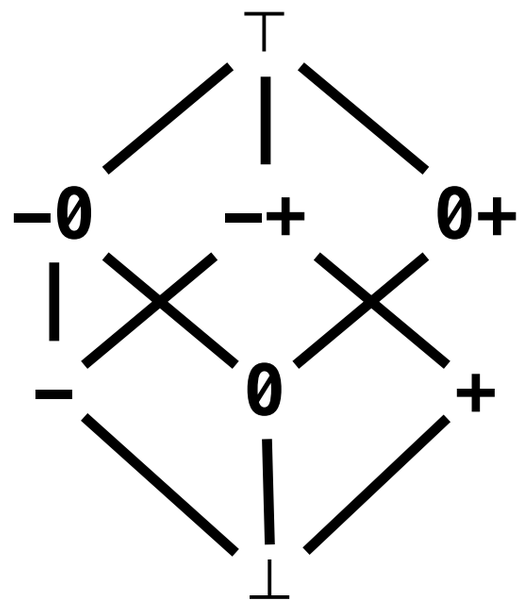
+ : 양수

정적 분석 - 요약 해석 (Abstract Interp.)

```
function f(x) {
  // x == ⊤
  if (x * 2 == 28322) {
    // x == ⊤
    fail();
    // May be Error
  } else if (x <= 0) {
    // x == -0
    return 0;
    // [RETURN] 0
  } else {
    // x == +
    return x;
    // [RETURN] +
  }
}
```

요약 도메인

(Abstract Domain)



⊤ : 모든 정수　⊥ : 공집합

- : 음수　0 : 0

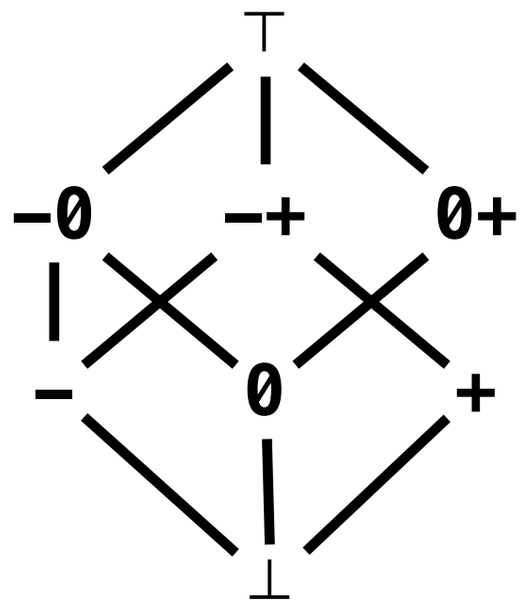
+ : 양수

정적 분석 - 요약 해석 (Abstract Interp.)

```
function f(x) {
  // x == ⊤
  if (x * 2 == 28322) {
    // x == ⊤
    fail();
    // May be Error
  } else if (x <= 0) {
    // x == -0
    return 0;
    // [RETURN] 0
  } else {
    // x == +
    return x;
    // [RETURN] +
  }
} // [RETURN] 0+
```

요약 도메인

(Abstract Domain)



⊤ : 모든 정수　⊥ : 공집합

- : 음수　0 : 0

+ : 양수

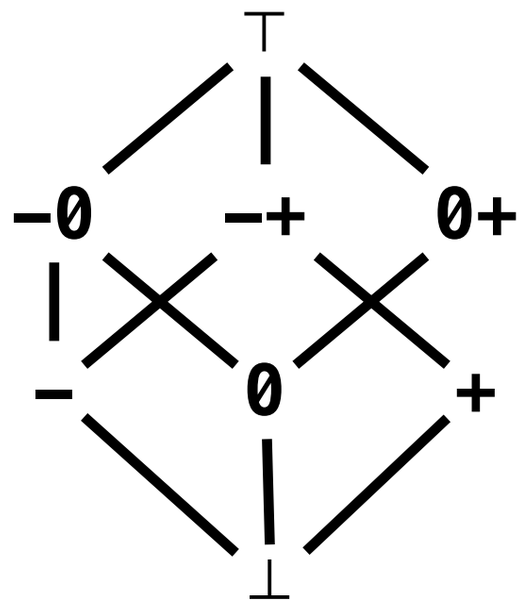
정적 분석 - 요약 해석 (Abstract Interp.)

```
function f(x) {
  // x == ⊤
  if (x * 2 == 28322) {
    // x == ⊤
    fail();
    // May be Error
  } else if (x <= 0) {
    // x == -0
    return 0;
    // [RETURN] 0
  } else {
    // x == +
    return x;
    // [RETURN] +
  }
} // [RETURN] 0+
```

항상 음이 아닌 정수 반환

요약 도메인

(Abstract Domain)



⊤ : 모든 정수　⊥ : 공집합

- : 음수　0 : 0

+ : 양수

PL 기반 SW의 개발

- **프로그램 분석** (Program Analysis)

- **컴파일러** (Compiler) / **트랜스파일러** (Transpiler) / **프로그램 자동 수정** (APR)

컴파일러 (Compiler)

- **고급 언어**에서 **저급 언어**로 변환

```c
int main(){
   int x = 10, y = 15;
   return 0;
}
```

C

컴파일러 (Compiler)

- **고급 언어**에서 **저급 언어**로 변환

```c
int main(){
  int x = 10, y = 15;
  return 0;
}
```

C

```asm
.globl main
    .type main, @function
main:
    pushq %rbp
    movq  %rsp, %rbp
    movl  $10, -8(%rbp)
    movl  $15, -4(%rbp)
    movl  $0, %eax
    leave
    ret
```

어셈블리

트랜스파일러 (Transpiler)

- **고급 언어**에서 (같은 혹은 다른) **고급 언어**로 변환

Scala

```
z = input()
def pow(x: Int, y: Int): Int =
  if (y == 0) 1
  else x * pow(x, y-1)
res = pow(z, 3)
```

트랜스파일러 (Transpiler)

- **고급 언어**에서 (같은 혹은 다른) **고급 언어**로 변환

Scala

```
z = input()
def pow(x: Int, y: Int): Int =
  if (y == 0) 1
  else x * pow(x, y-1)
res = pow(z, 3)
```

x = z AND y = 3

트랜스파일러 (Transpiler)

- **고급 언어**에서 (같은 혹은 다른) **고급 언어**로 변환

Scala

```
z = input()
def pow(x: Int, y: Int): Int =
  if (y == 0) 1
  else x * pow(x, y-1)
res = pow(z, 3)
```

x = z AND y = 3

최적화

Scala

```
z = input()
res = z * z * z
```

함수 호출 없이 빠르게 계산

트랜스파일러 (Transpiler)

- **고급 언어**에서 (같은 혹은 다른) **고급 언어**로 변환

```
x ??= y
```

JavaScript
(ES12)

트랜스파일러 (Transpiler)

- **고급 언어**에서 (같은 혹은 다른) **고급 언어**로 변환

??= 를 지원하지 않으면?

x ??= y

JavaScript
(ES12)

트랜스파일러 (Transpiler)

- **고급 언어**에서 (같은 혹은 다른) **고급 언어**로 변환

??= 를 지원하지 않으면?

```
x ??= y
```

호환성 변환

x ??= y와 같은 의미를 가짐

```
var _x; (_x = x) !== null &&
_x !== void 0 ? _x : x = y;
```

JavaScript
(ES12)

JavaScript
(ES5.1)

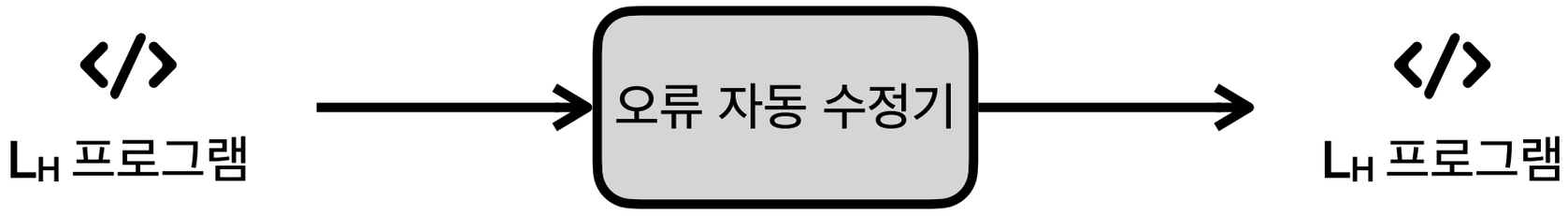
프로그램 자동 수정 (Automatic Program Repair, APR)

- **자동**으로 **프로그램**의 **오류를 수정**

C/C++

```
p = malloc(1);
q = p;
free(p);
free(q);
```

프로그램 자동 수정 (Automatic Program Repair, APR)



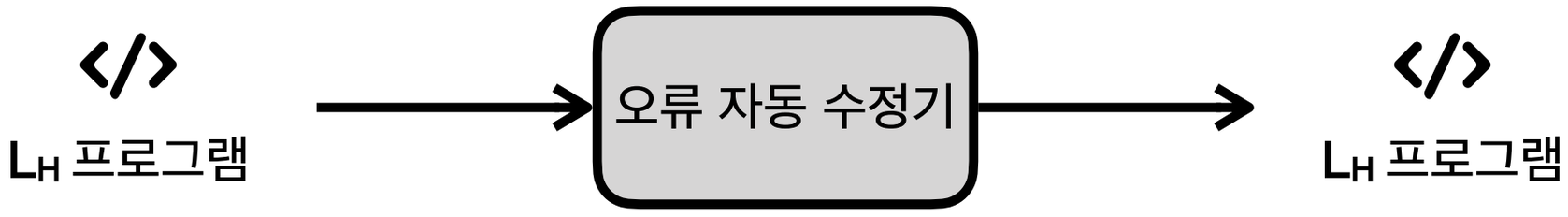
- **자동**으로 **프로그램**의 **오류를 수정**

C/C++

```
p = malloc(1);
q = p;
free(p);
free(q);
```

Double-Free

프로그램 자동 수정 (Automatic Program Repair, APR)



- **자동**으로 **프로그램**의 **오류를 수정**

C/C++

```
p = malloc(1);
q = p;
free(p);
free(q);
```

Double-Free

오류 수정

C/C++

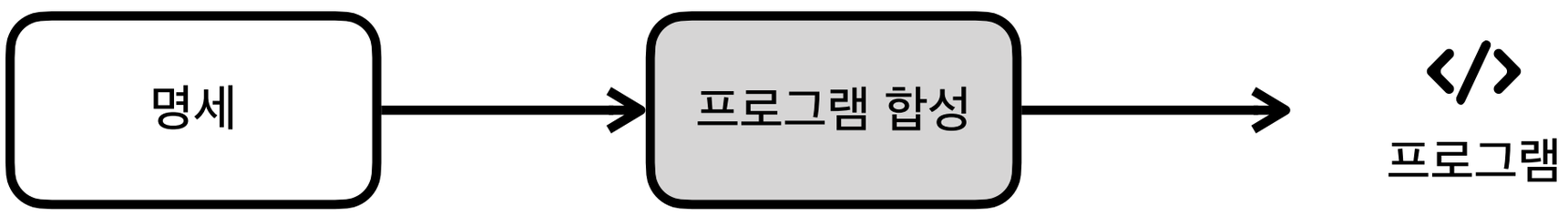
```
  p = malloc(1);
  q = p;
  free(p);
- free(q);
```

PL 기반 SW의 개발

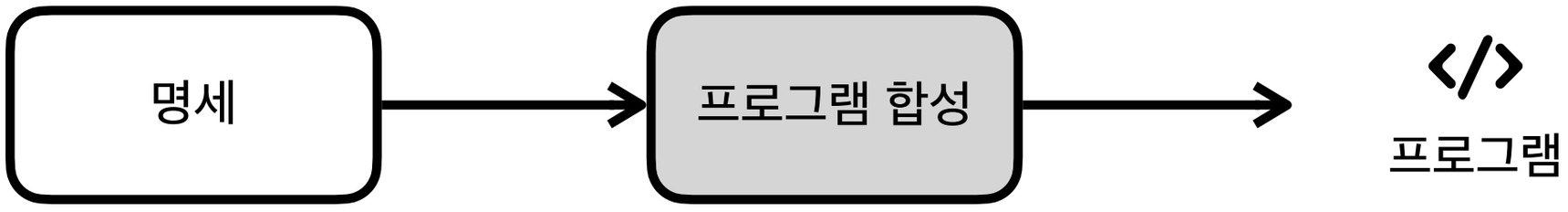
- **프로그램 분석** (Program Analysis)

- **컴파일러** (Compiler) / **트랜스파일러** (Transpiler) / **프로그램 자동 수정** (APR)

- **프로그램 합성** (Program Synthesis)



프로그램 합성 (Program Synthesis)

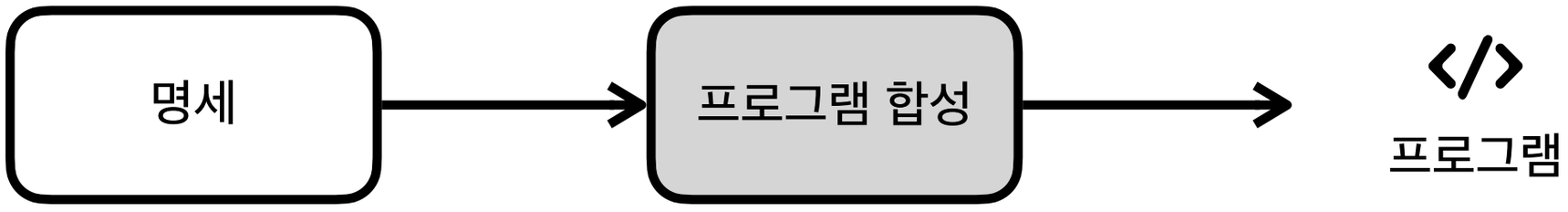


- **명세**를 만족하는 **프로그램**을 **자동으로 합성**

입력-출력 명세

```
[]                => []
[2,0]             => [0,2]
[2,4,5,3,1]       => [1,2,3,4,5]
[3,6,4,5,7,9]     => [3,4,5,6,7,9]
...
```

프로그램 합성 (Program Synthesis)



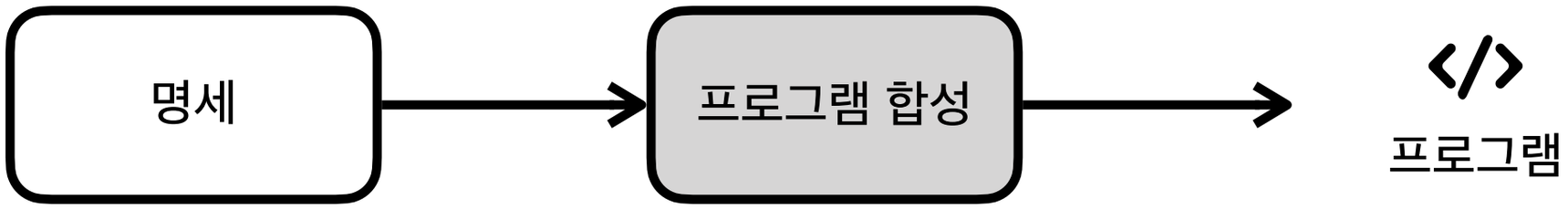
- **명세**를 만족하는 **프로그램**을 **자동으로 합성**

입력-출력 명세

```
[]                => []
[2,0]             => [0,2]
[2,4,5,3,1]       => [1,2,3,4,5]
[3,6,4,5,7,9]     => [3,4,5,6,7,9]
...
```

정렬 알고리즘

프로그램 합성 (Program Synthesis)



- **명세**를 만족하는 **프로그램**을 **자동으로 합성**

입력-출력 명세

```
[]                => []
[2,0]             => [0,2]
[2,4,5,3,1]       => [1,2,3,4,5]
[3,6,4,5,7,9]     => [3,4,5,6,7,9]
...
```

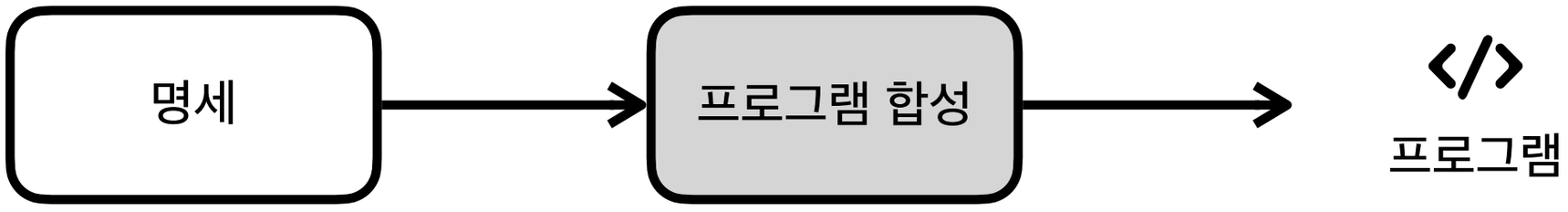
생성

Scala

```scala
def sort(
  list: List[Int]
): List[Int] = list.sorted
```

정렬 알고리즘

프로그램 합성 (Program Synthesis)

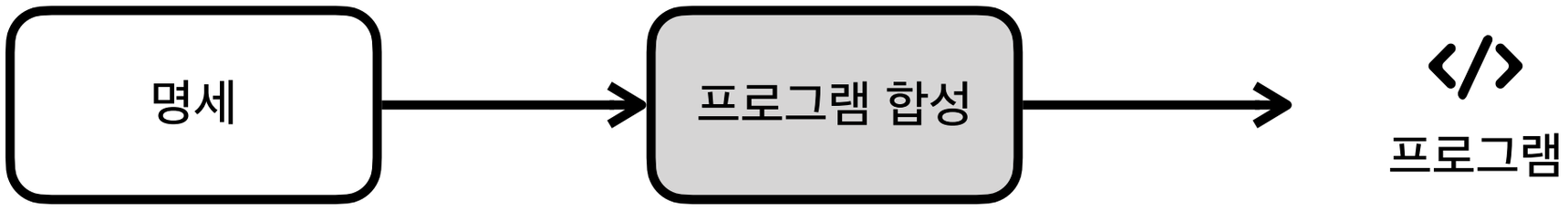


- **명세**를 만족하는 **프로그램**을 **자동으로 합성**

자연어 명세

```
A function `sort` should
take a list of integers and
returns its sorted list.
```

프로그램 합성 (Program Synthesis)



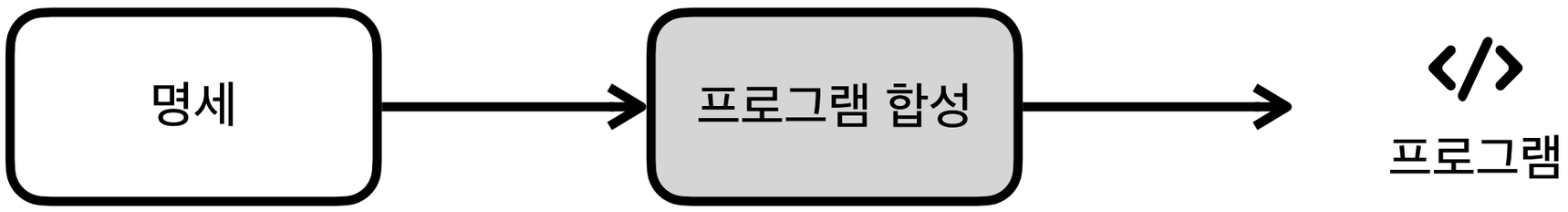
- **명세**를 만족하는 **프로그램**을 **자동으로 합성**

자연어 명세

```
A function `sort` should
take a list of integers and
returns its sorted list.
```

정렬 알고리즘

프로그램 합성 (Program Synthesis)



- **명세**를 만족하는 **프로그램**을 **자동으로 합성**

자연어 명세

```
A function `sort` should
take a list of integers and
returns its sorted list.
```

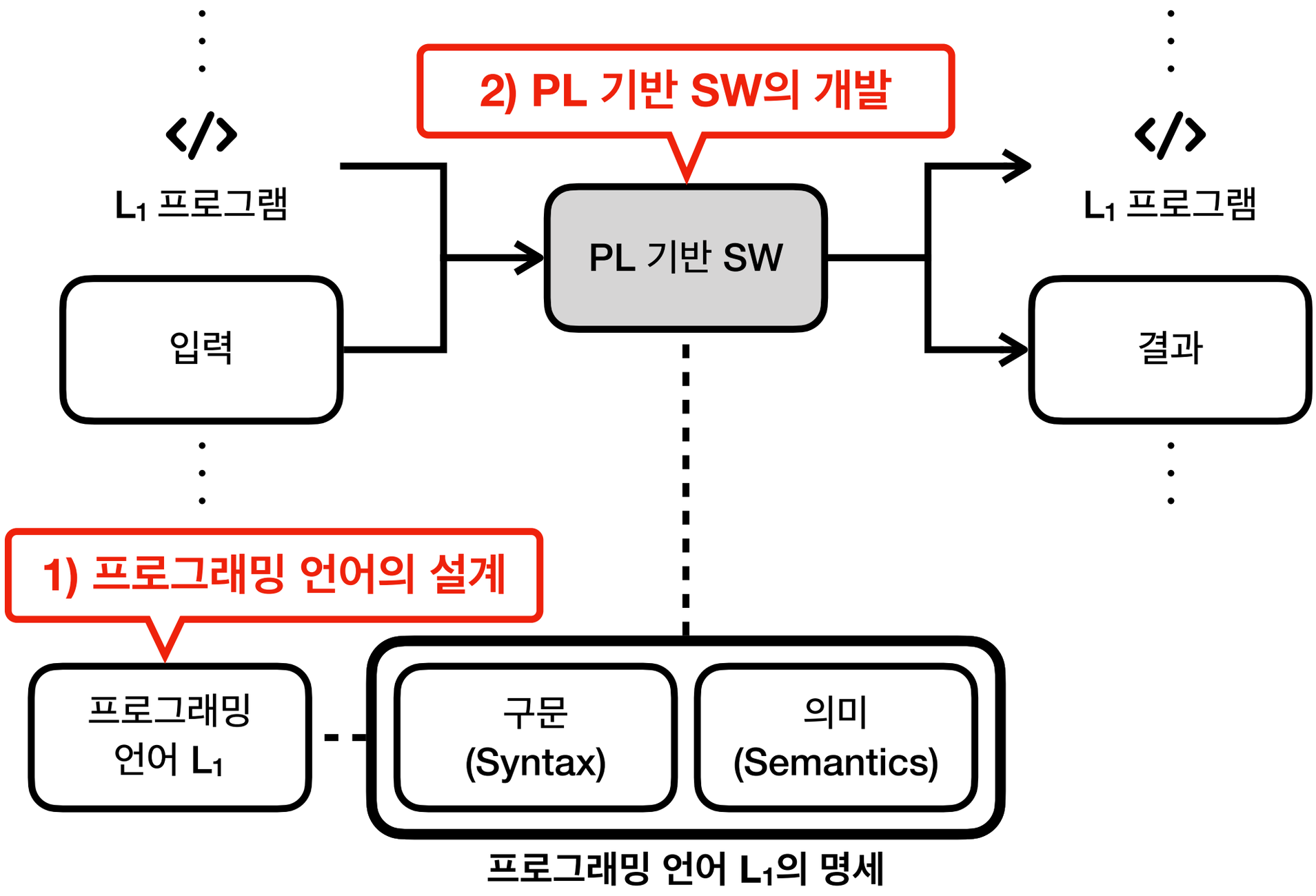
생성

Scala

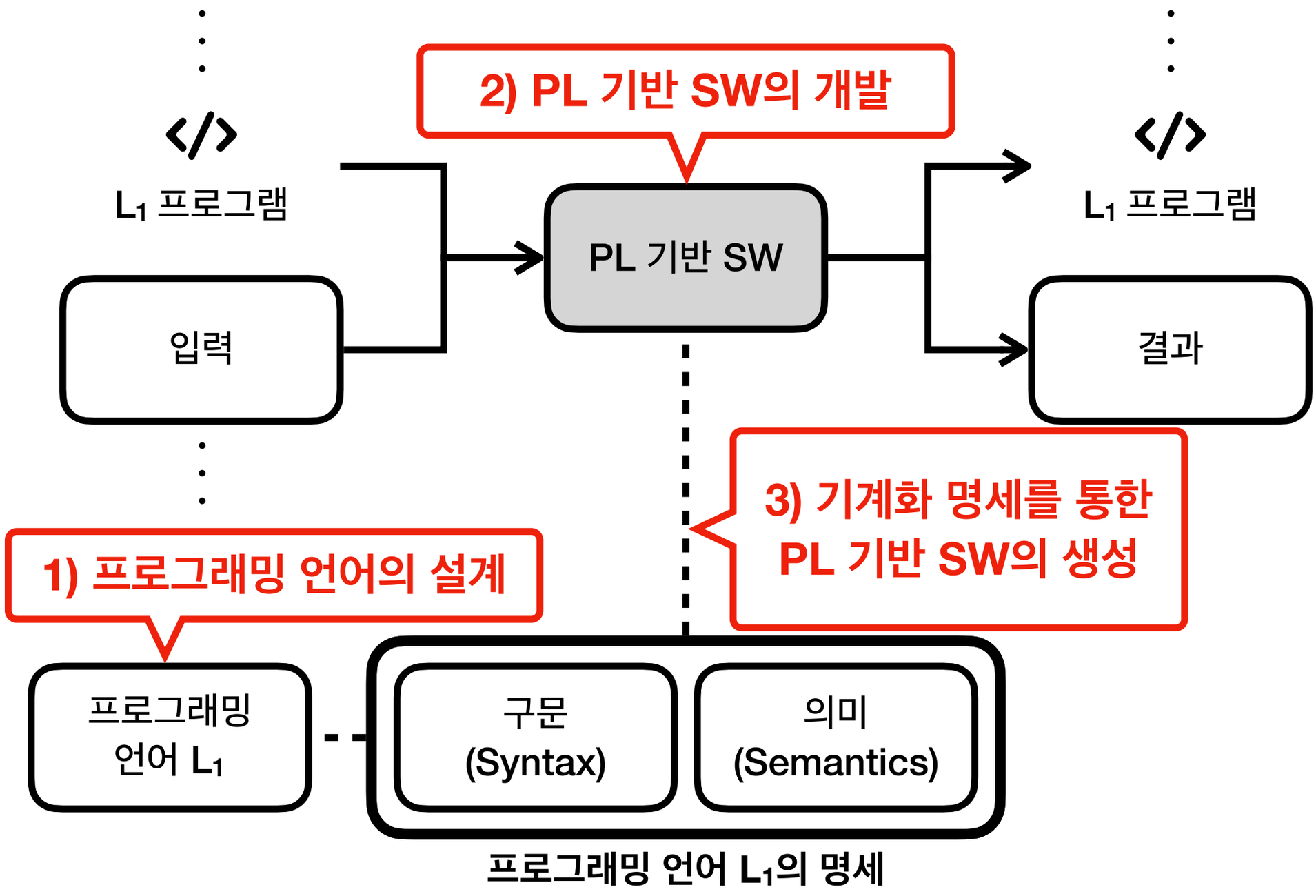
```scala
def sort(
  list: List[Int]
): List[Int] = list.sorted
```

정렬 알고리즘

프로그래밍 언어 연구


2) PL 기반 SW의 개발
L1 프로그램
PL 기반 SW
입력
L1 프로그램
결과
1) 프로그래밍 언어의 설계
프로그래밍
언어 L1
구문
(Syntax)
의미
(Semantics)
프로그래밍 언어 L1의 명세

프로그래밍 언어 연구


2) PL 기반 SW의 개발
L1 프로그램
PL 기반 SW
입력
L1 프로그램
결과
1) 프로그래밍 언어의 설계
3) 기계화 명세를 통한
PL 기반 SW의 생성
프로그래밍
언어 L1
구문
(Syntax)
의미
(Semantics)
프로그래밍 언어 L1의 명세