

# 은은하게 퍼지는 쉬운전문용어와 소박한 공부

허기홍  
KAIST 전산학부

쉬운전문용어 포럼@KSC 2024

전과자

***POSTECH***

*POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY*

컴퓨터공학과

# 전과자

70화\_컴퓨터공학과



## TODAY

등교

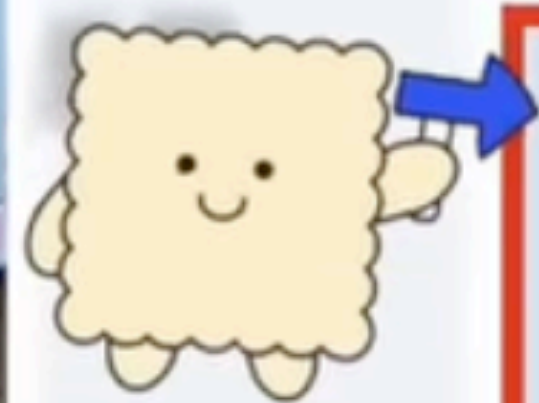
등록금 납부

수업1  
(객체지향프로그래밍)

점심시간

수업2  
(컴퓨터SW시스템개론)

하교



객체지향이란 무엇인가?

객체지향이란 무엇인가?

객체지향이 뭐야?

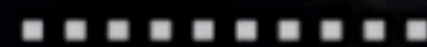


## 〈객체지향 프로그래밍〉

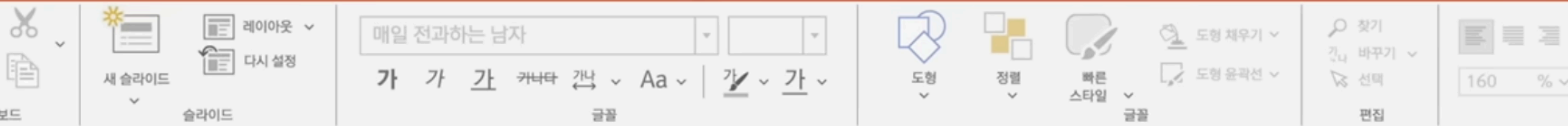
**TIP**

모든 데이터를 하나의 객체로 취급해서 프로그래밍하는 방법  
각 객체들끼리 스스로 상호작용하여 프로그래밍 진행





[조짐 감지]  
수업 잘 못 들어온 거 같은데?



# 객체지향프로그래밍

## 전공 (3학점)

프로그래밍의 역사와  
객체지향 프로그래밍 탐구





[수업 마무리]

여기까지 왔으면 이제 객체지향 프로그래밍이 뭔지



[이해 1도 못 함]  
그래서 객체지향 프로그램이 뭡니까?

# 대학교 1학년 시절



???

국정원?

???



???

↔ 주체?

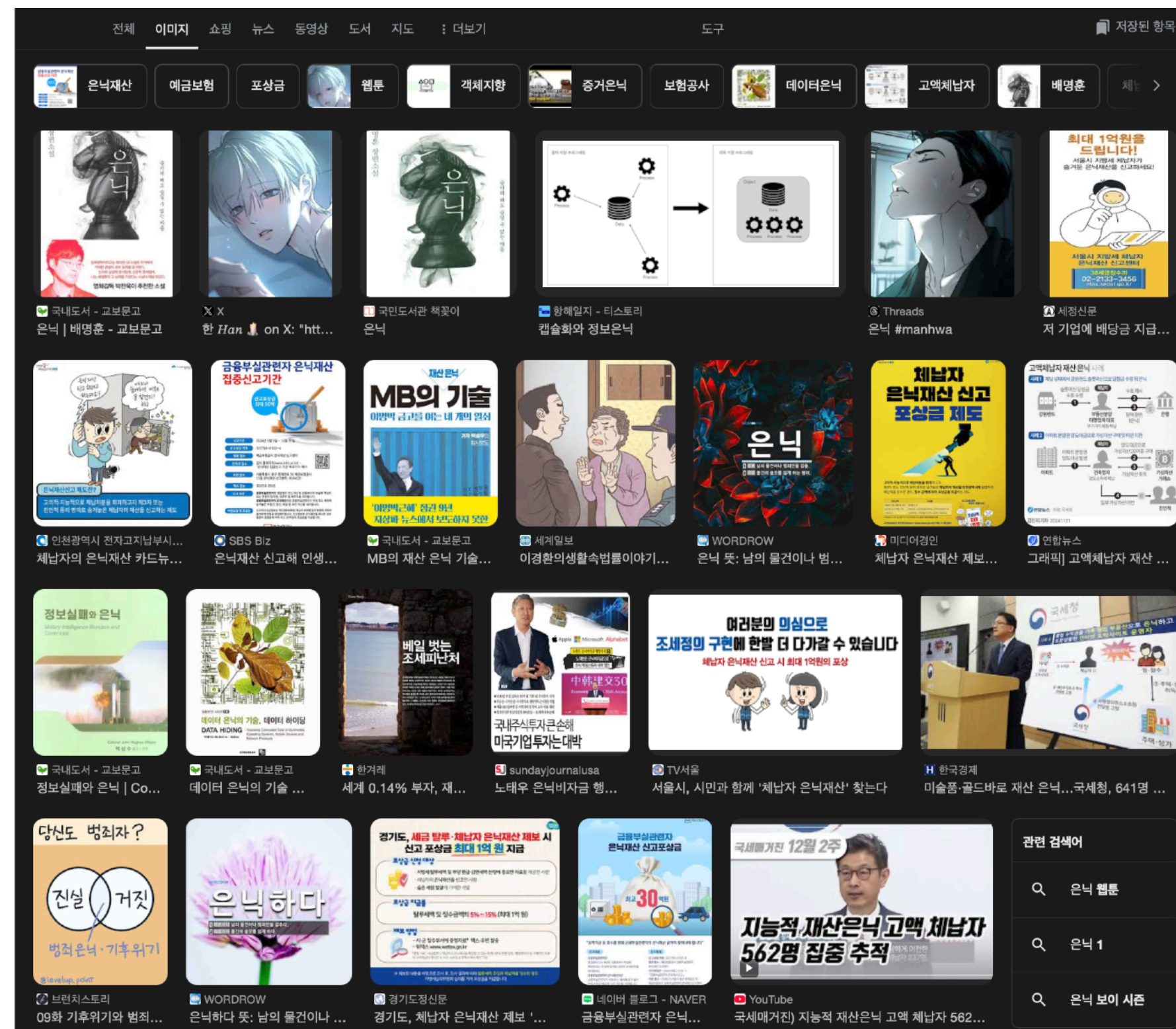
???

???

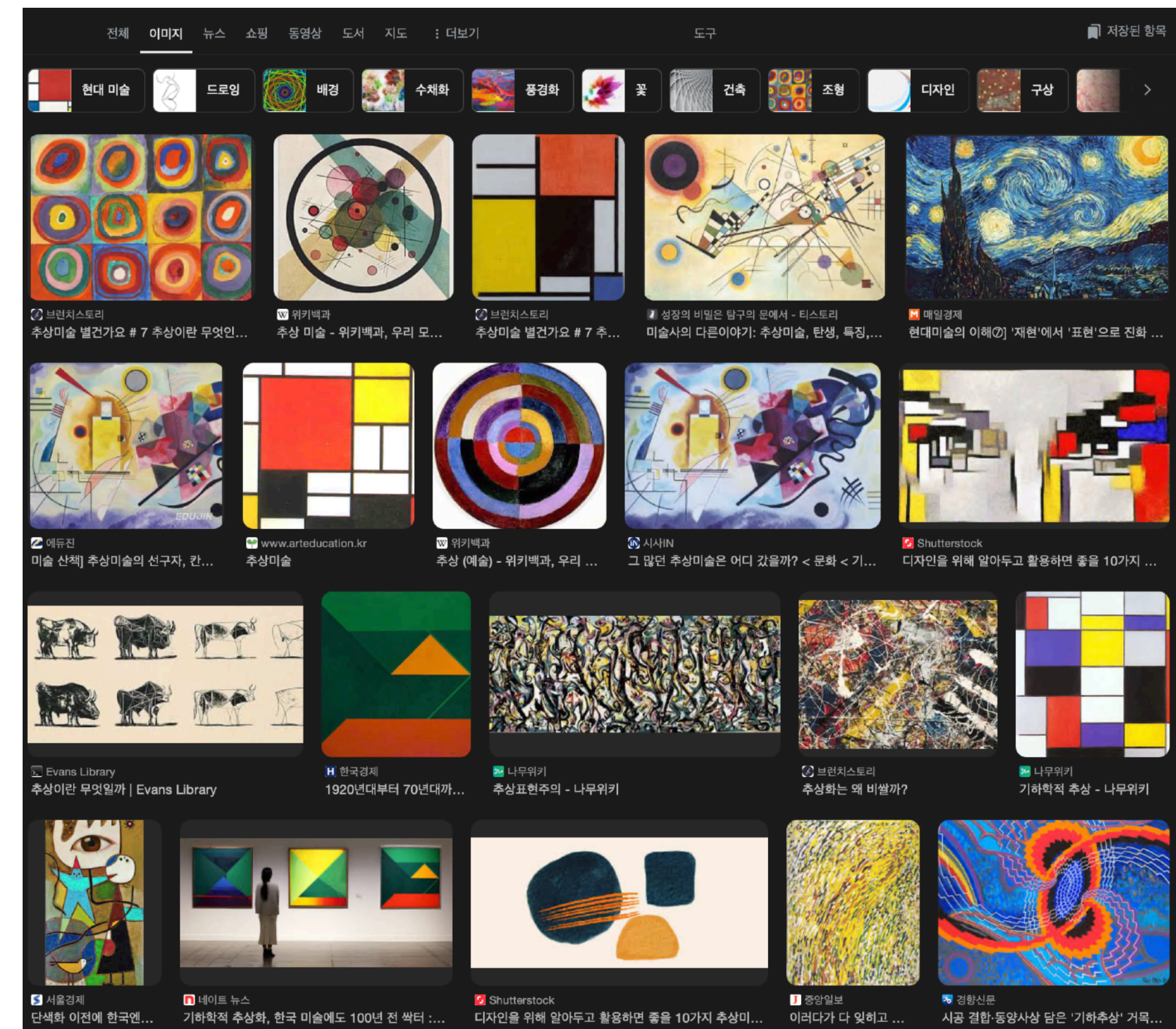
객체 **지향** 프로그램

# 그 후

## 정보 은닉 (information hiding)??



## 추상화 (abstraction)??



# 쉬운전문용어로 하는 소박한 공부

- 소박: 헛꾸밈 X, 과장 X, 담백 O, 직관 O
- 쉬운 설명과 쉬운 이해
  - “○○○라는 말 그대로”
  - 음악 예: 도돌이표, 당김음, 돌림노래, 셋잇단음표
  - 물리 예: 얽힘, 중첩, 결맞음
  - 우리 예: 거꾸로 분석 (역공학x), 식 (표현식x)
- 깊은 울림과 깊은 궁리
  - 예: 요약 (abstraction), 기억하며풀기 (dynamic programming)

## Abstraction

The purpose of abstracting is not to be vague, but to create a new semantic level in which one can be absolutely precise.

요약의 목적은 모호해가는 것이 아니라  
완벽하게 명료한 의미를 전할 수 있는  
새로운 수준을 만들어내는 것이다.

## 요약

“네. 취지는 좋은데요.  
쓸 기회가 없어요.  
영어로 논문쓰고,  
영어로 발표하고,  
영어로 강의하는데...”

# 은은하게 퍼지는 쉬운전문용어: 강의에서

- 중요성 강조: 일류 과학자를 위한 소통 능력, <https://easyword.kr/why>
- 직접 겪어보게끔: 강의자료, 과제, 질의응답 게시판 등

### 국문 글쓰기

- 제대로 된 우리말 번역과 전문 용어 원문 병기
  - 예: 프로그램 검증 (program verification)
- 바보들의 전문 용어 번역 알고리즘:
  1. 영문 용어를 단어별로 쪼갬다
  2. 각 용어를 사전에서 찾아 제일 처음 나오는 우리말 단어로 바꾼다
  3. 바뀐 단어들을 순서대로 나열한다
- 일류 과학자의 전문 용어 번역 알고리즘: 개념을 제대로 이해하고 핵심을 설명하는 쉬운 우리말로
- 참고: PL-Wiki, [easywork.kr](https://easyword.kr)의 번역어

1. Introduction CS424 / KAIST Kihong Heo 39 / 52

### 불완전하지만 유용한

#### 이동재

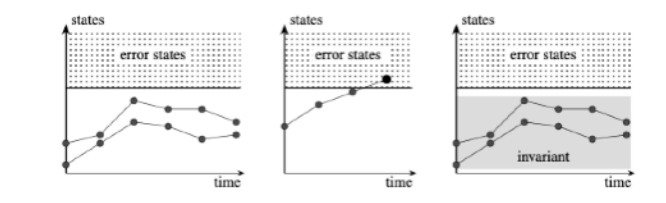
소프트웨어의 불완전함은 괴델의 불완전성 정리와 튜링의 멈춤 문제를 통해 이미 증명되었다. 그러나 소프트웨어는 현대 사회의 근간을 이루며, 우리의 삶에 필수적인 유용성을 제공하고 있다. 소프트웨어 공학자들은 이러한 불완전함을 최대한 억제하고 유용성을 극대화하기 위해 정적 분석, 동적 분석, 검증, 테스트 등 다양한 기술을 개발해왔다. 근본적으로 불완전한 시스템 속에서 유용한 소프트웨어를 창조하는 것은 소프트웨어 공학의 아름다움이자 커다란 도전이다. 본 글에서는 소프트웨어의 불완전함을 극복하기 위한 미래의 방향성을 예측하고, 그 과정에서 소프트웨어 공학 전문가로서 우리가 수행해야 할 역할을 논의하고자 한다.

소프트웨어의 근본적인 한계는 괴델의 불완전성 정리와 튜링의 멈춤 문제를 통해 명확히 드러난다. 괴델은 일관적이고 충분히 복잡한 형식 체계 내에서는 참이지만 증명할 수 없는 명제가 존재함을 증명하였다. 이는 수학적 시스템의 완전성이 불가능함을 의미하며, 컴퓨터 프로그램 역시 이러한 논리 체계 위에서 동작하기 때문에 모든 프로그램의 올바름을 완벽히 증명하는 것은 불가능하다. 튜링의 멈춤 문제는 어떤 일반적인 알고리즘으로도 모든 프로그램이 멈출지 아닐지를 결정하는 것이 불가능하다는 것을 보여준다. 이는 프로그램의 예측 불가능성을 내포하며, 소프트웨어의 완전한 검증을 어렵게 만드는 핵심 요인이다.

그럼에도 불구하고, 소프트웨어는 현대 사회에서 필수불가결한 역할을 수행하고 있다. 소프트웨어는 우리의 삶을 편리하게 해주는 도구이며, 스마트폰, 컴퓨터, 가전제품 등 거의 모든 전자기에 탑재되어 있다. 전기를 사용하는 거의 모든 물건에는 이를 제어하는 소프트웨어가 내장되어 있다고 해도 과언이 아니다. 소프트웨어의 불완전함은 그 유용성을 제한하지만, 그것이 소프트웨어가 불필요하다는 뜻은 아니다. 오히려, 이러한 불완전함을 극복하고 소프트웨어의 유용성을 극대화하기 위한 노력은 소프트웨어 공학의 발전을 촉진시켰다.

### Safety Property 항상성질

- A program **never** exhibit a behavior observable within **finite time**
  - “Bad things will never occur”
  - Bad things: integer overflow, buffer overrun, deadlock, etc
- If false, then there exists a **finite counterexample**
- To prove: all executions never reach error states



3. Concepts in Program Verification CS424 / KAIST Kihong Heo 8 / 28

### 안녕하세요!

여러분들의 동기부여를 위해 ChatGPT와 마찬가지로 2021년에 발표된 최신 SLIA 합성기 Duet을 Gradescope 순위표에 추가하였습니다. Duet은 92 / 100 점을 받았습니다.

Duet에 대한 정보는 아래를 참고해 주세요.

논문: <https://dl.acm.org/doi/pdf/10.1145/3434335>

저장소: <https://github.com/wslee/duet>

2022년도의 챔피언은 Duet보다 높은 점수를 받았습니다.

여러분도 창의력을 발휘하여 최신 기술을 이겨보세요!

감사합니다.

Hello everyone,

To inspire your motivation, I've added Duet, a state-of-the-art SLIA synthesizer published in 2021, to the Gradescope leaderboard alongside ChatGPT. Duet scored 92/100.

For more information about Duet, please check the links below:

Paper: <https://dl.acm.org/doi/pdf/10.1145/3434335>

Repository: <https://github.com/wslee/duet>

The champion of 2022 achieved a higher score than Duet.

Let your creativity shine and aim to surpass the latest technologies!

# 은은하게 퍼지는 쉬운전문용어: 연구에서 (1)

- 적극 사용: 연구 블로그, 제안서, 발표 자료, 논문, 책 등

Home  
KihongHeo edited this page on Dec 9, 2022 · 6 revisions

## PL 위키

KAIST 프로그래밍 시스템 연구실에서 진행하는 최신 연구와 관련된 지식저장소입니다. 프로그래밍 언어와 관련된 전문 지식을 소개합니다. 넓게는 소프트웨어 공학, 소프트웨어 보안 관련 내용도 다룹니다. 아래와 같은 목표를 염두에 두고 있습니다.

- 우리 연구실의 성과를 비전문가들 (예: 학부생, 타전공 대학원생, 고등학생) 도 이해하기 쉽도록 소개
- PL 분야의 입문자를 위한 고급 전문 자료 소개
- 정갈하게 국문으로 과학적인 글을 쓴 사례 모음

## 작성 원칙

- 올바른 국문으로 글을 작성하고 전문 용어는 괄호안에 병기한다.
  - 굳어진 우리말 용어가 있다면, 병기 없이 사용한다. (예: 컴파일러, 프로그램 분석)
  - 굳어진 용어가 없을 때는 참고자료의 내용을 바탕으로 하고 전문 용어를 병기한다.
  - 굳어진 용어가 없는 경우 새로운 용어를 적극 개발한다.
- 외부에서 따온 내용은 출처를 명시한다.
- 많은 사람들에게 익숙한 위키피디아 문서와 비슷하게 구성한다.

DAFL  
goodtaeun edited this page on Nov 12 · 11 revisions

## DAFL: 데이터 의존성 정보를 활용한 지향성 반투명 퍼징

### 1. 배경

지향성 퍼징 문서를 먼저 참고하면 본 문서의 내용을 이해하는 데 많은 도움이 될 것이다.

### 2. 개요

2017년, AFLGo를 시작으로 지금까지 다양한 지향성 반투명 퍼징 기법이 제안되어 왔다. AFLGo는 AFL을 기반으로 하였기에 변형 기반 반투명 퍼징의 특성을 가진다. 즉, 가지고 있는 입력 집합에서 하나를 선택한 후, 다양한 변종들을 생성하고 그 중 새로운 커버리지를 달성한 입력들만 다시 입력 집합에 추가하는 방식이다. 추가로 여기에 지향성의 개념을 부여하기 위해 목표와의 거리를 계산해 그 거리가 작은, 즉, 목표에 도달 가능성이 더 큰 입력에 집중한다. 이 때, 각 입력과 목표 사이의 거리는 실행흐름그래프(Control Flow Graph)상에서 입력이 실행한 프로그램 지점들과 목표 지점 사이의 거리를 계산하여 산출한다. AFLGo 이후로 대부분의 기법들도 이와 같은 방식을 채택해왔다. 그러나 이런 방식으로는 대상 프로그램이 크고 복잡할수록 목표 지점까지의 정확한 가이드를 제공하기 어렵다. 따라서 본 문서는 구체적으로 어떠한 문제가 있고, 왜 이러한 문제가 발생하는지, 그리고 DAFL이라는 새로운 기법은 이 문제를 어떻게 해결하는지 설명하려 한다.

## 안전하고 효율적인 코드 생성 모델

류연희, 박일범, 허기홍, 김기용  
KAIST Programming Systems Laboratory

배경: 언어 모델이 혼란 결함을 학습하고 재생산  
배경: 기존 기술의 한계

언어 모델이 생성한 코드의 40%는 보안 취약점을 내포 (Pierce et al., SOSP 2022)  
언어 모델에 학습시킬 안전한 코드 데이터 수집이 어려움  
"특정 결함만 있는 코드"와 "그 결함으로부터 안전한 코드"를 자동으로 수집하기 어려움  
거대 언어 모델 학습용 데이터셋을 수집하기 어려움 (Vilardot et al., arXiv 2022)  
거대 언어 모델 학습 비용이 비싸고 비효율적

기술	저작자/학명	데이터 규모	수용량
StarCoder	He & Viehler, CCC23	C & Python Commit 1.608개	사양 미정 수준
SafeCoder	He et al. ICML 24	수용 안전 Commit 468개	CodeQL, 악용
Code Poisoning Attack	Cotroneo et al., IOP24	C & Python Commit 1078개	사양 미정 수준

### 강화 학습을 이용한 언어 모델 학습 방법

기존 LLM: 코드 출력 중, 프렌스포어, 코드 입력 중  
CodeDT: 강화 학습 구조 적용, 코드 출력 중, 프렌스포어, 프렌스포어, 코드 입력 중  
CodeDT-LoRA: PEFT 학습 적용, 코드 출력 중, 프렌스포어, 프렌스포어, LoRA, 코드 입력 중

정적 분석을 이용한 결함 데이터 자동 수집  
정적 분석을 이용한 강화 학습의 개선과 적용  
PEFT: Parameter-Efficient Fine-Tuning

코드 생성 방법: 안전하거나 위험하거나  
실험: 평가 방법

목표 위험 지수 입력을 이용하여 모델이 생성하는 코드 품질  
학습 및 평가 데이터

1. 결함 생성을 조절하는 능력이 있는가?  
2. 다른 모델에 비해 안전한 코드를 생성하는가?

CodeDT에 적용된 학습 기법의 효과 확인

모델	모델 크기	학습 파라미터	학습 내용
Llama2-7B	6.73B	-	기본 모델
LoRA-tuned	6.743B	4.2M (0.06%)	학습 데이터에서 코드 텍스트만 LoRA 학습
CodeDT-tuned	6.743B	135.3M (2.01%)	코드와 결함을 강화 학습, LoRA 학습

목표 위험 지수에 따라 코드 안전성 조절 가능

테스트 성공률, 결함 유무, 결함 결함률, 테스트 성공률, 결함 유무, 결함 결함률





# 은은하게 퍼지는 쉬운전문용어: 연구에서 (2)

- 적극 생산, 개선
  - 자부심: “내 분야는 내가 국내 최고 전문가”
  - 재미: 널리 쓰이는 원동력
  - 예: “지향성 퍼징/마구실행”, “연구기록물상”

영문 용어	과거 번역어	현재 번역어
injective func	단사함수	일대일 함수
surjective func	전사함수	위로의 함수
bijective func	전단사함수	일대일대응
even func	우함수	짝함수
odd func	기함수	홀함수



# 정리

- 쉬운전문용어로 하는 공부 (교육, 연구)
  - 쉽게, 재미있게, 깊게, 넓게
- 잊지말자 올챙이적: 우리는 모두 초보자
  - 예: 비폐색? 부전지?
- 널리 퍼트리기
  - 과제, 교재, 유튜브, 소설, 제안서, ....., 논문
  - + 그게 걸맞은 포상 (예: 세종발표상)

