숨겨진 알짜 실행, 지향성 값-맥락 덮이로 관찰하세요!

김태은, 최재승, 허기홍, 차상길

2026.02.02

Programming Systems Laboratory

KAIST

배경: 지향성 퍼징의 목표

배경: 지향성 퍼징의 목표

목표 지점 도달

- 대부분의 지향성 퍼징 도구의 목적
- 일단 도달하면 좋은 일이 일어난다는 믿음

배경: 지향성 퍼징의 목표

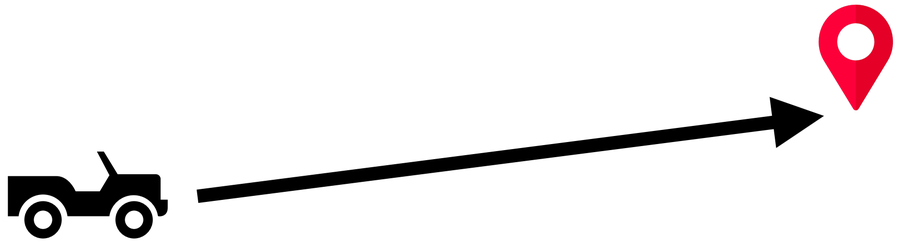
목표 지점 도달

- 대부분의 지향성 퍼징 도구의 목적
- 일단 도달하면 좋은 일이 일어난다는 믿음

배경: 지향성 퍼징의 목표

목표 지점 도달

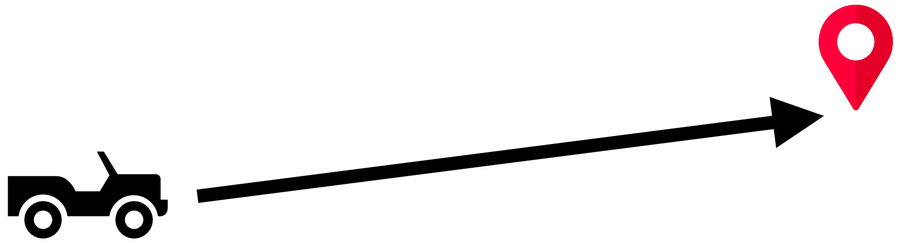
- 대부분의 지향성 퍼징 도구의 목적
- 일단 도달하면 좋은 일이 일어난다는 믿음



배경: 지향성 퍼징의 목표

목표 지점 도달

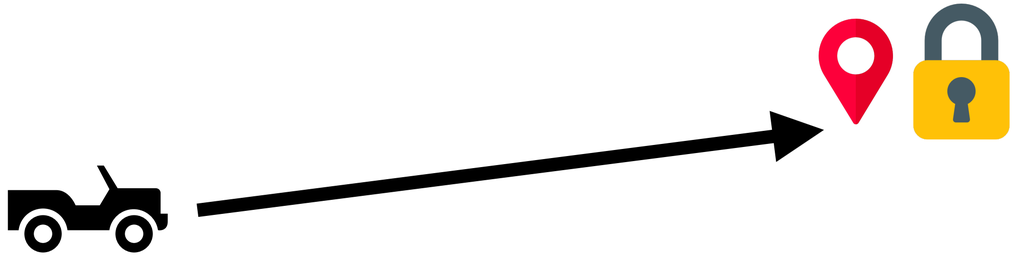
- 대부분의 지향성 퍼징 도구의 목적
- 일단 도달하면 좋은 일이 일어난다는 믿음



목표 지점의 오류 발견

- 도달은 필요조건일 뿐 충분 조건이 X

배경: 지향성 퍼징의 목표



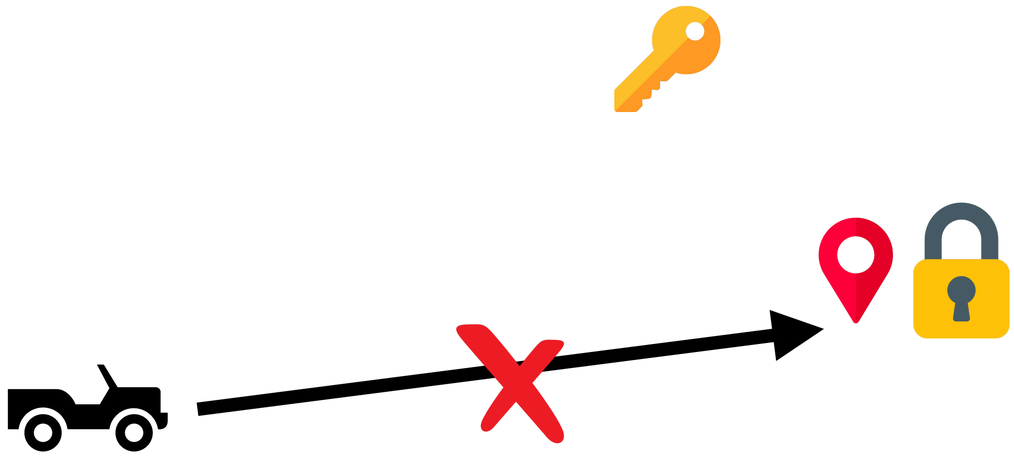
목표 지점 도달

- 대부분의 지향성 퍼징 도구의 목적
- 일단 도달하면 좋은 일이 일어난다는 믿음

목표 지점의 오류 발견

- 도달은 필요조건일 뿐 충분 조건이 X

배경: 지향성 퍼징의 목표



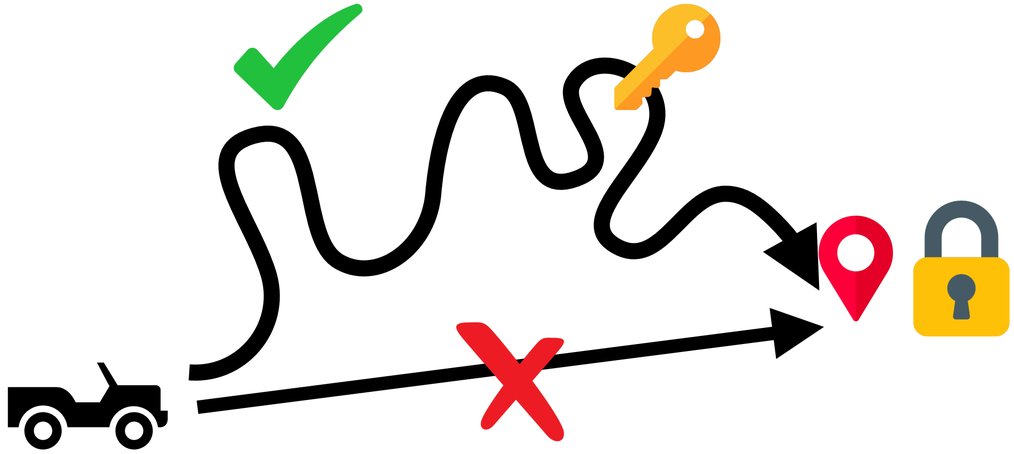
목표 지점 도달

- 대부분의 지향성 퍼징 도구의 목적
- 일단 도달하면 좋은 일이 일어난다는 믿음

목표 지점의 오류 발견

- 도달은 필요조건일 뿐 충분 조건이 X

배경: 지향성 퍼징의 목표



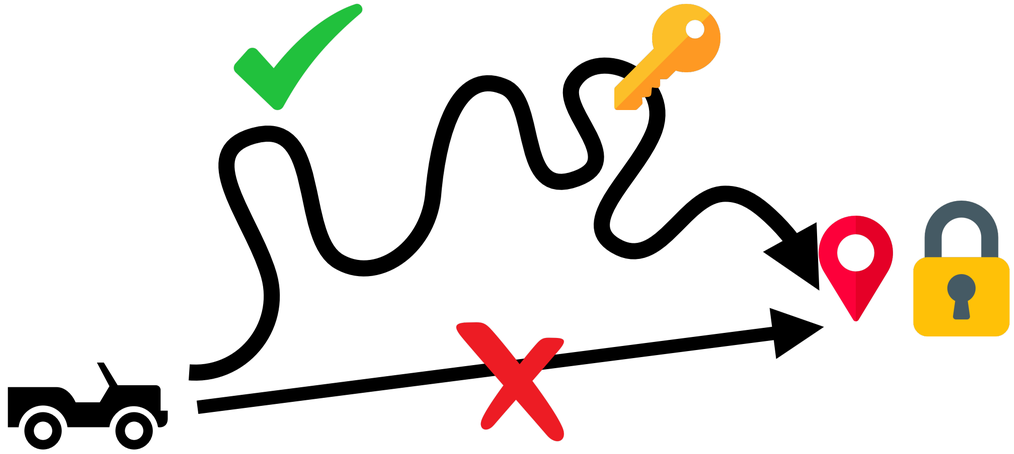
목표 지점 도달

- 대부분의 지향성 퍼징 도구의 목적
- 일단 도달하면 좋은 일이 일어난다는 믿음

목표 지점의 오류 발견

- 도달은 필요조건일 뿐 충분 조건이 X
- 특정한 경로 + 특정한 변수 값으로 목표 지점 도달 필요

배경: 지향성 퍼징의 목표



목표 지점 도달

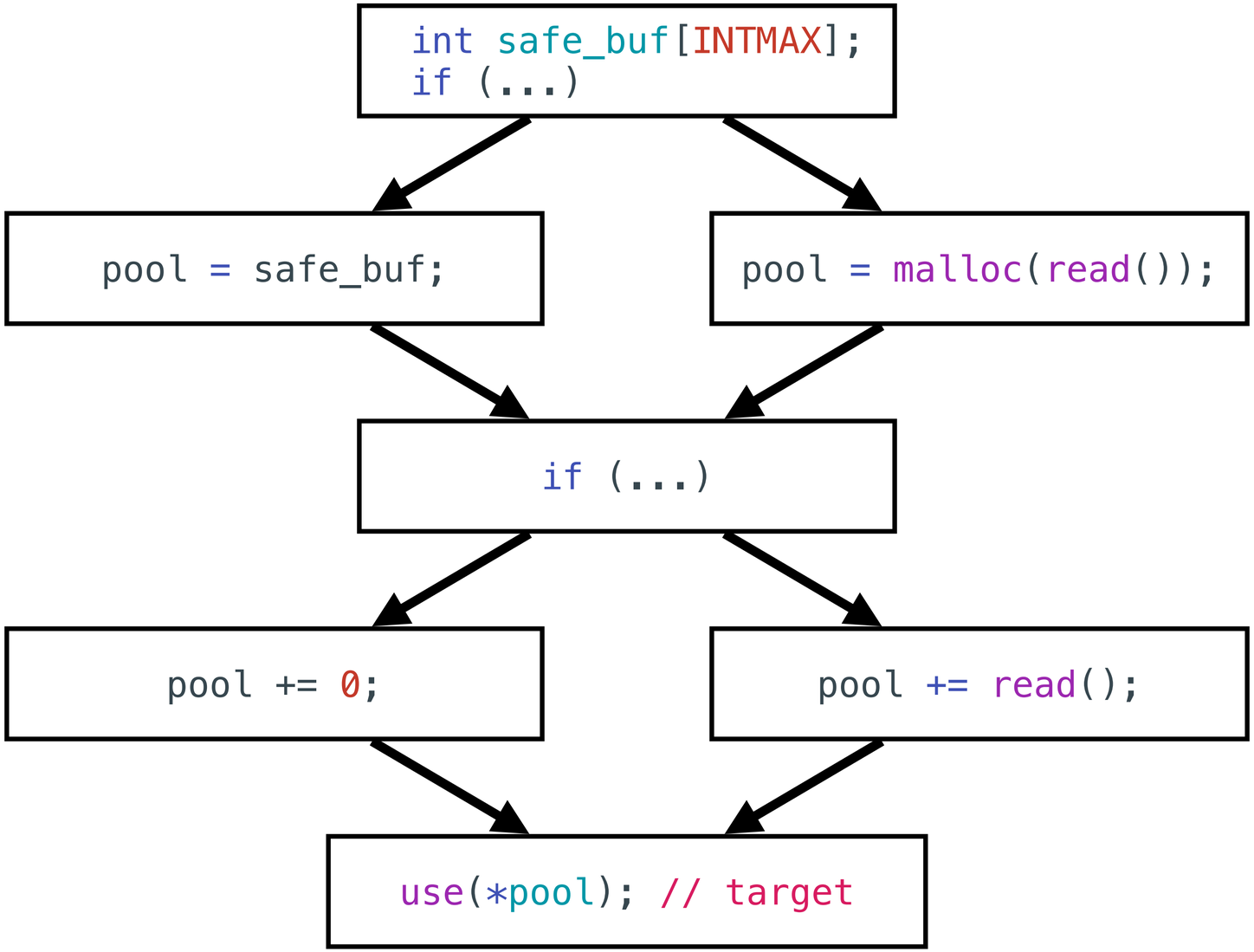
- 대부분의 지향성 퍼징 도구의 목적
- 일단 도달하면 좋은 일이 일어난다는 믿음

목표 지점의 오류 발견

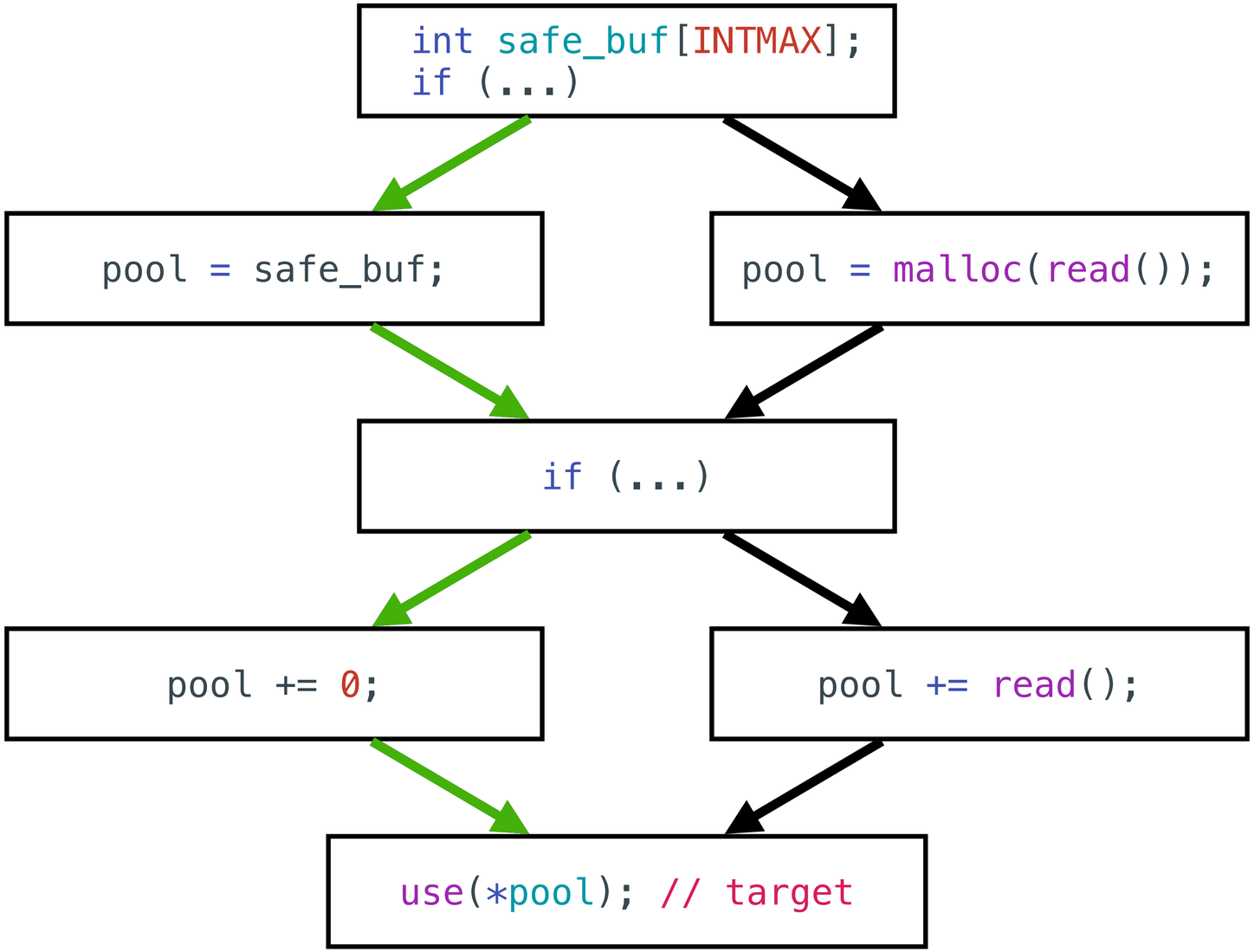
- 도달은 필요조건일 뿐 충분 조건이 X
- 특정한 경로 + 특정한 변수 값으로 목표 지점 도달 필요

기존의 덮이 측정 방식은 의미있는 실행을 충분히 탐색할 수 없음!

문제: 구문 덮이의 과도한 요약

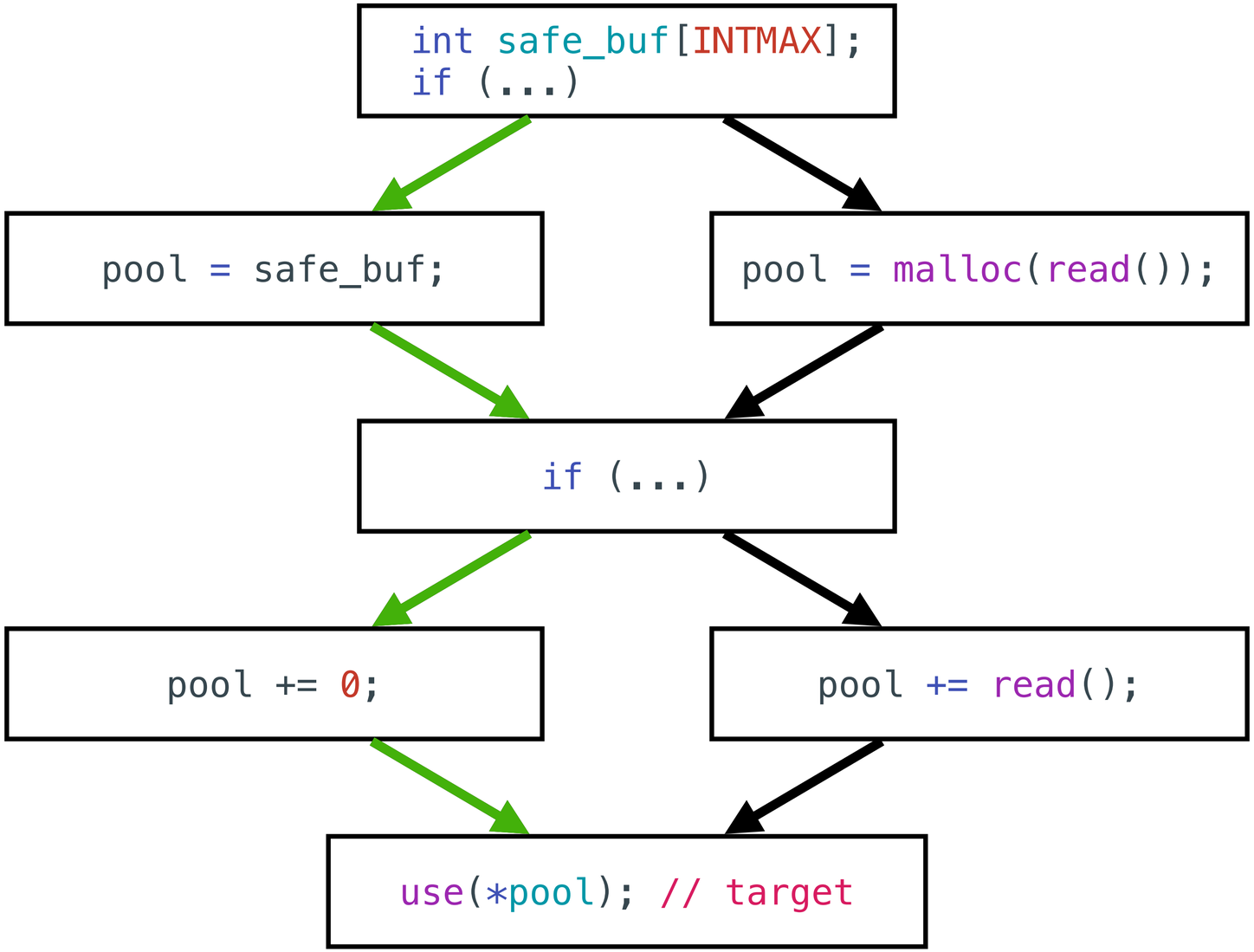


문제: 구문 덮이의 과도한 요약



문제: 구문 덮이의 과도한 요약

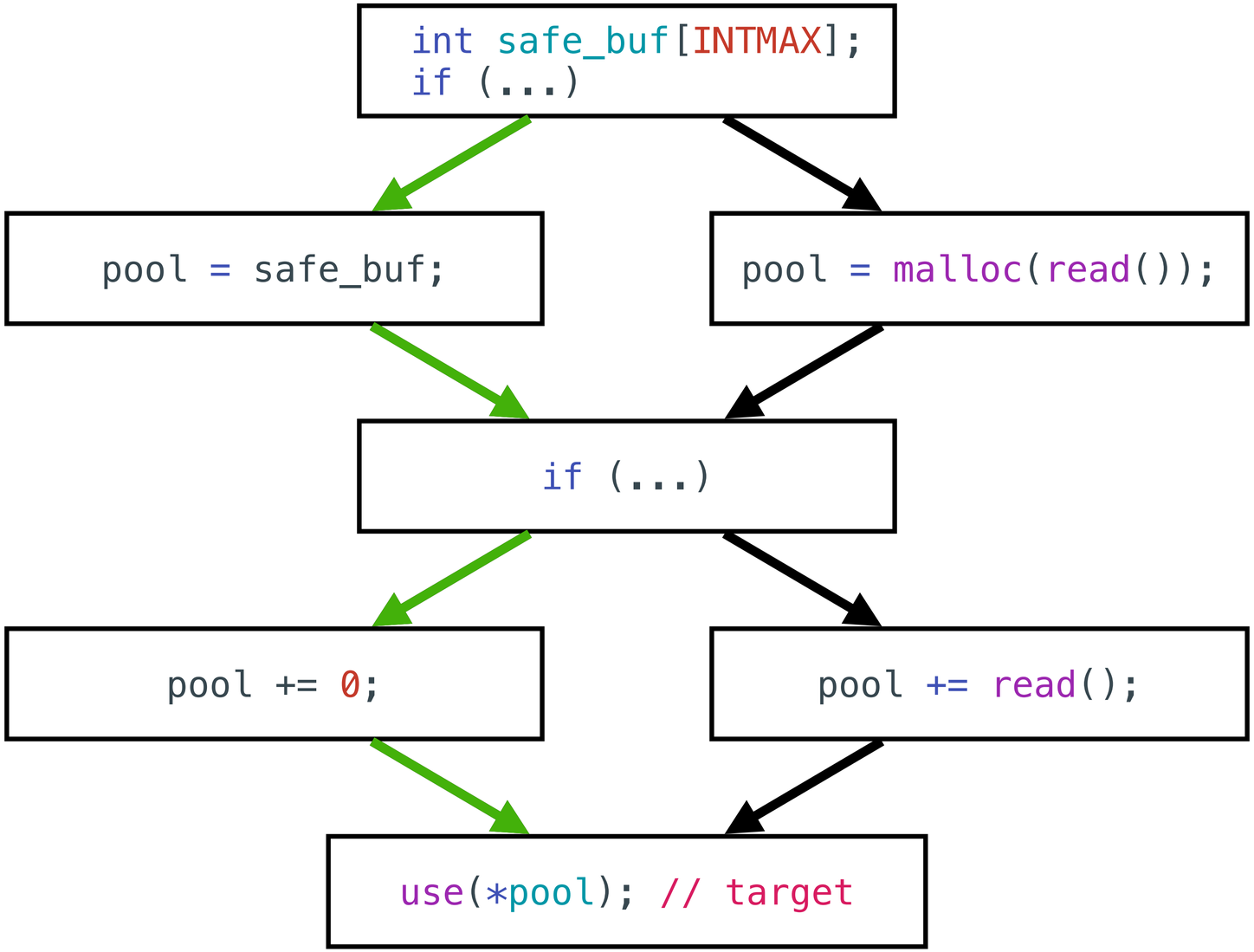
가능한 경로: 1개



문제: 구문 덮이의 과도한 요약

가능한 경로: 1개

가능한 실행: 1개

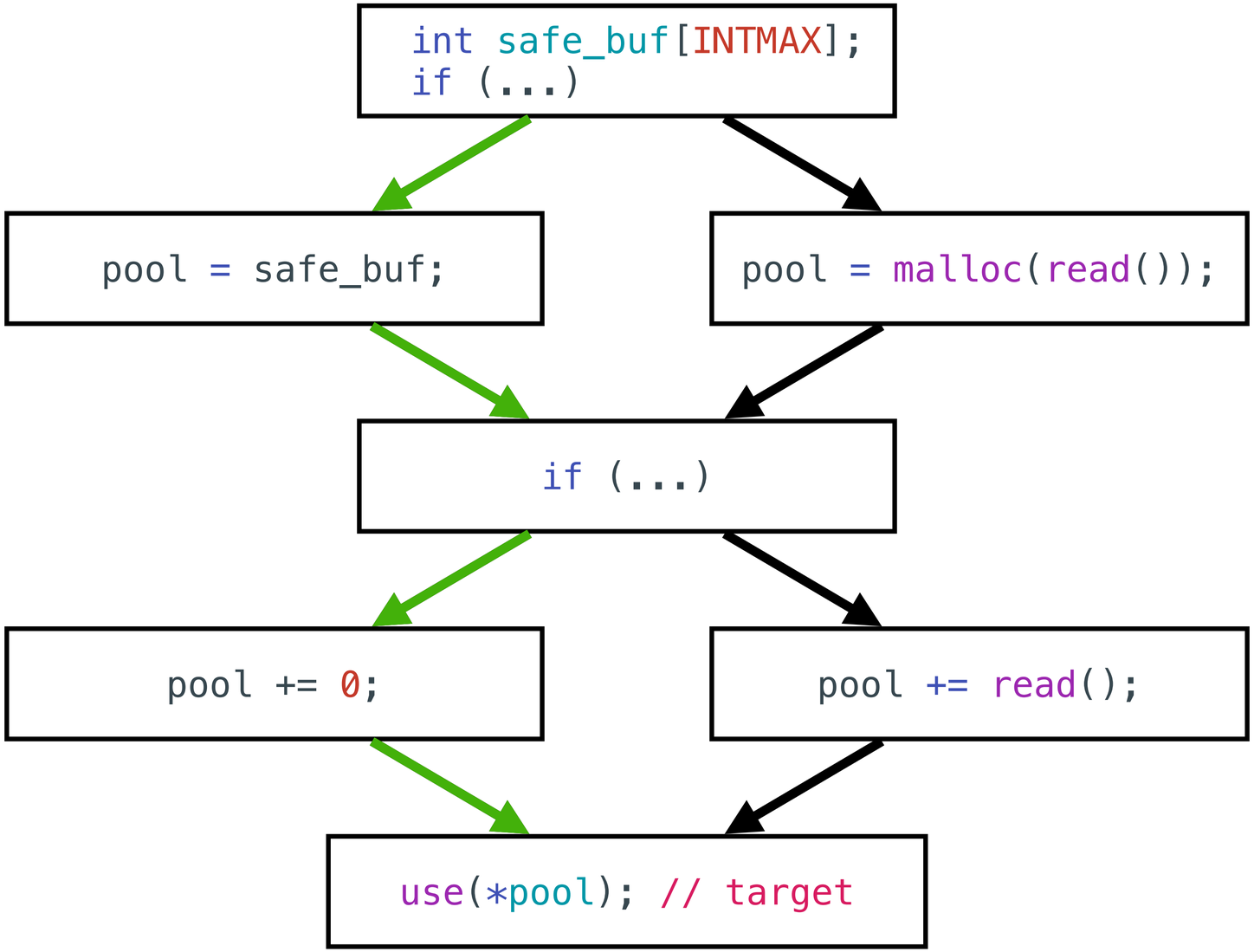


문제: 구문 덮이의 과도한 요약

가능한 경로: 1개

가능한 실행: 1개

오류 가능성: 없음

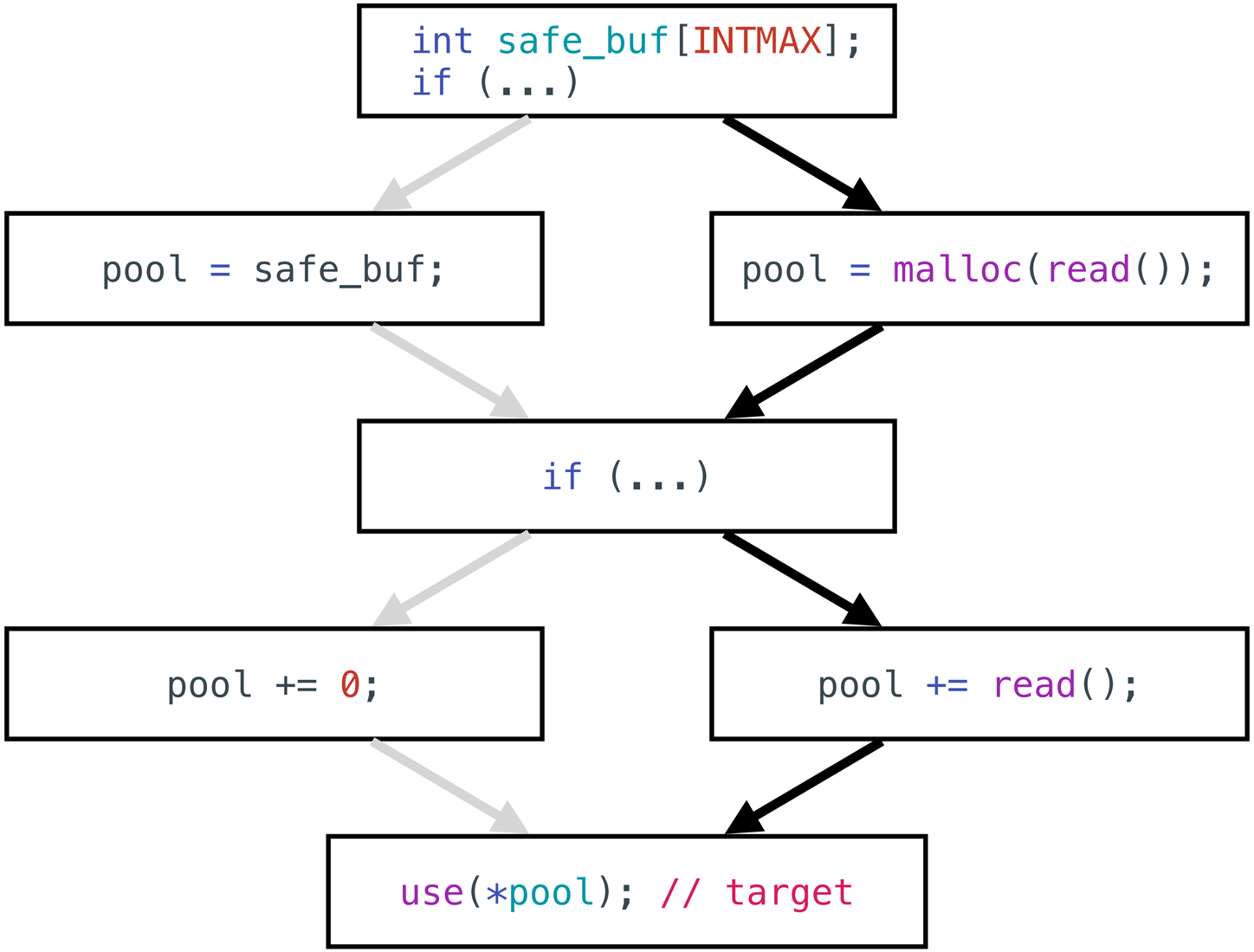


문제: 구문 덮이의 과도한 요약

가능한 경로: 1개

가능한 실행: 1개

오류 가능성: 없음

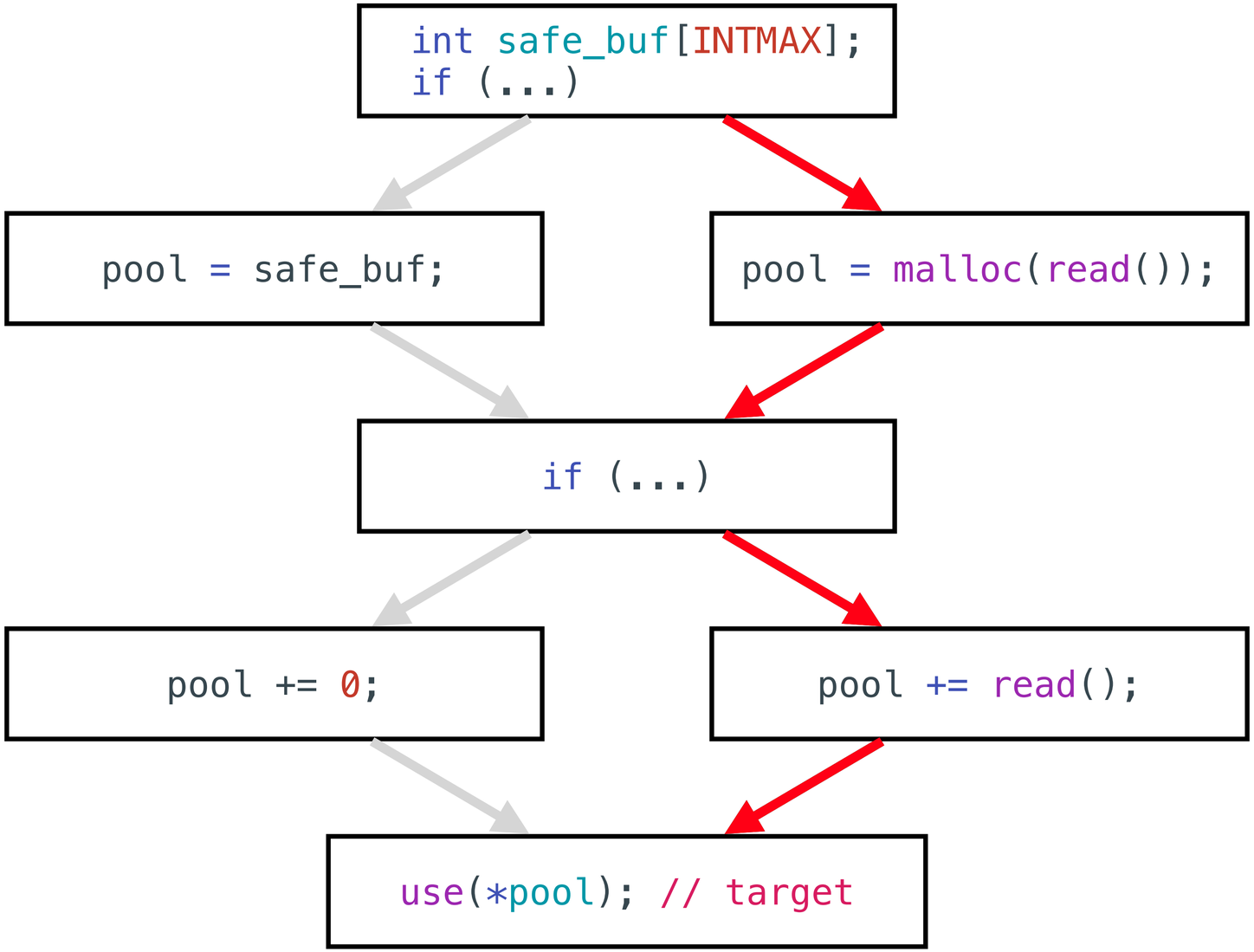


문제: 구문 덮이의 과도한 요약

가능한 경로: 1개

가능한 실행: 1개

오류 가능성: 없음

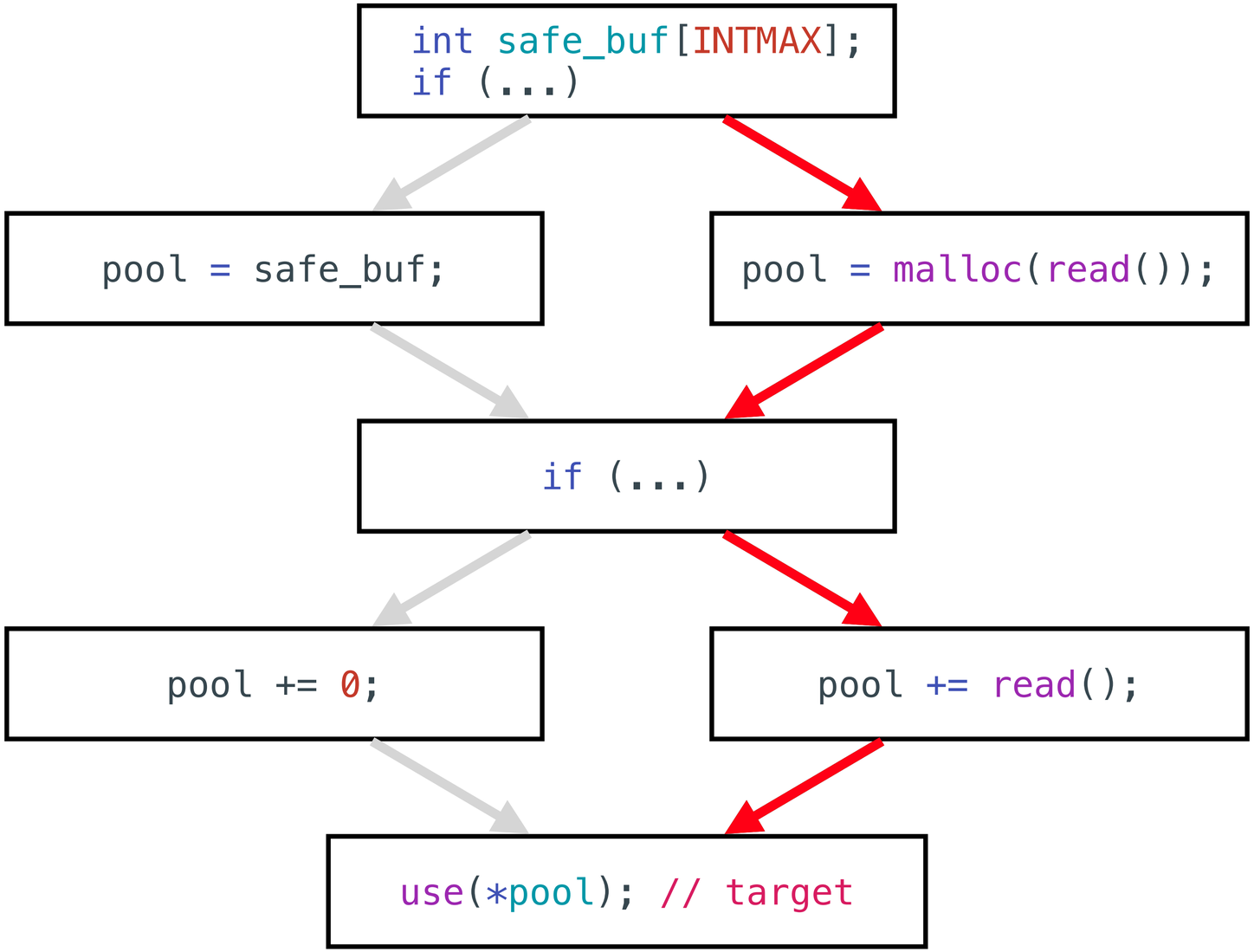


문제: 구문 덮이의 과도한 요약

가능한 경로: 1개

가능한 실행: 1개

오류 가능성: 없음



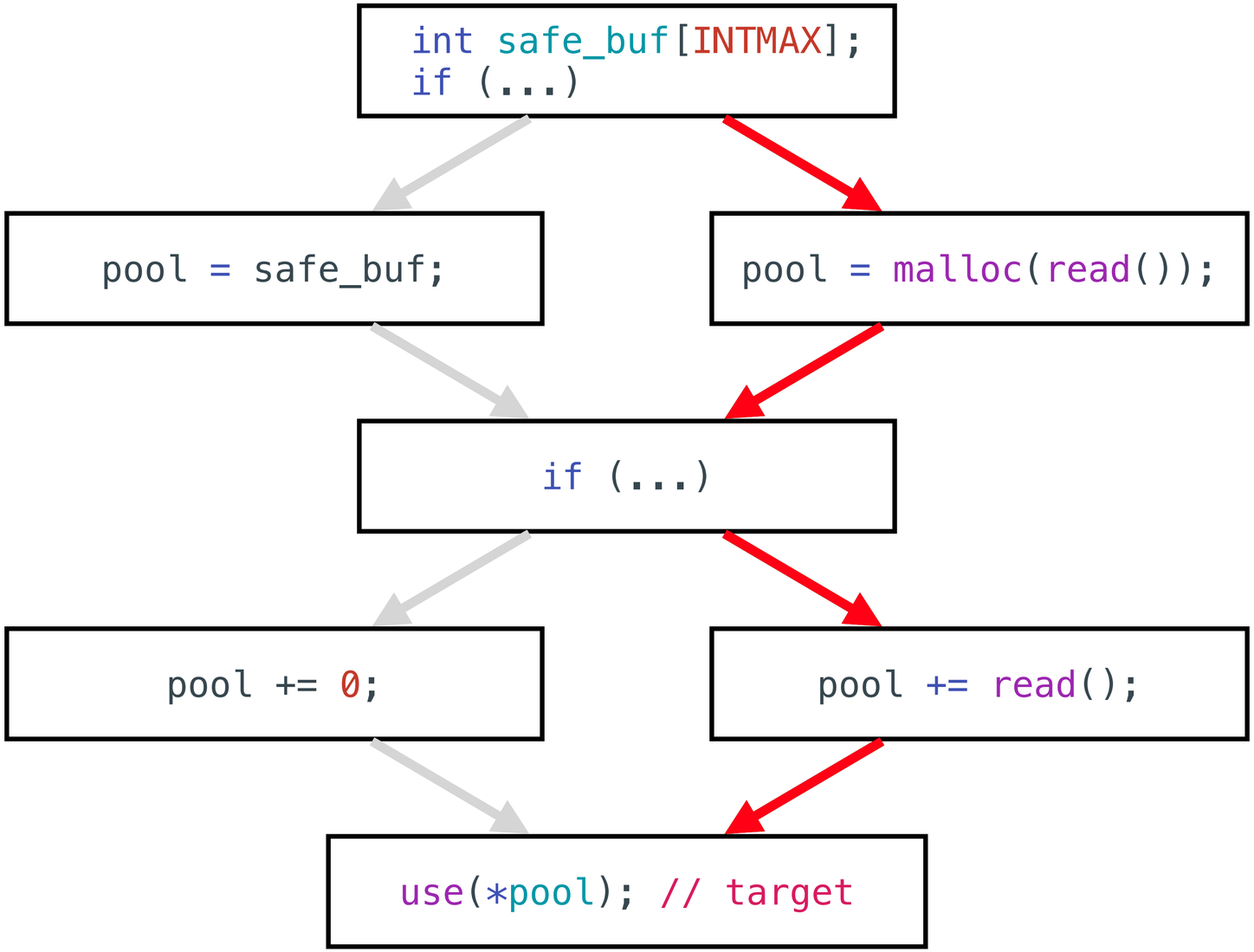
가능한 경로: 1개

문제: 구문 덮이의 과도한 요약

가능한 경로: 1개

가능한 실행: 1개

오류 가능성: 없음



가능한 경로: 1개

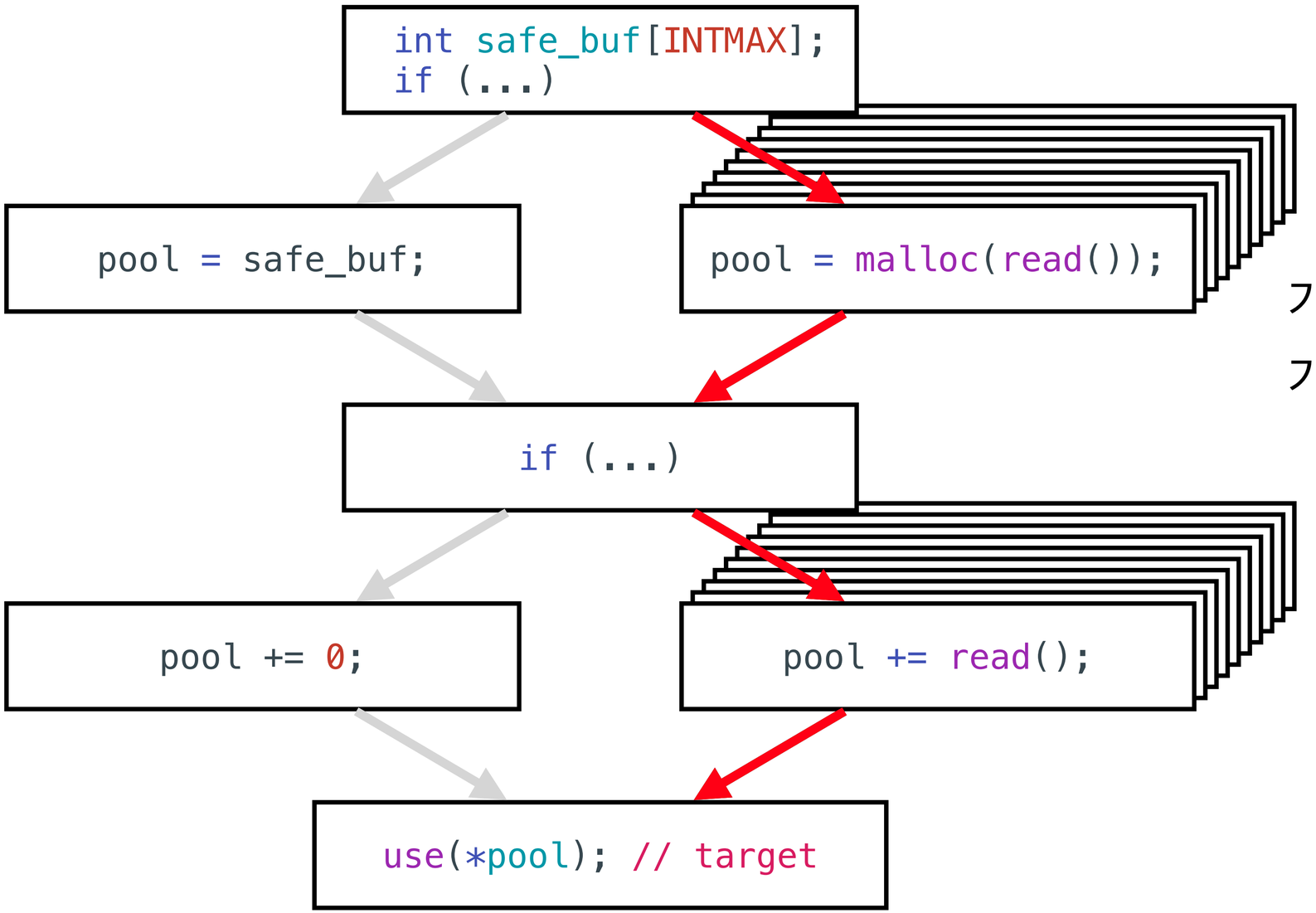
가능한 실행:

문제: 구문 덮이의 과도한 요약

가능한 경로: 1개

가능한 실행: 1개

오류 가능성: 없음



가능한 경로: 1개

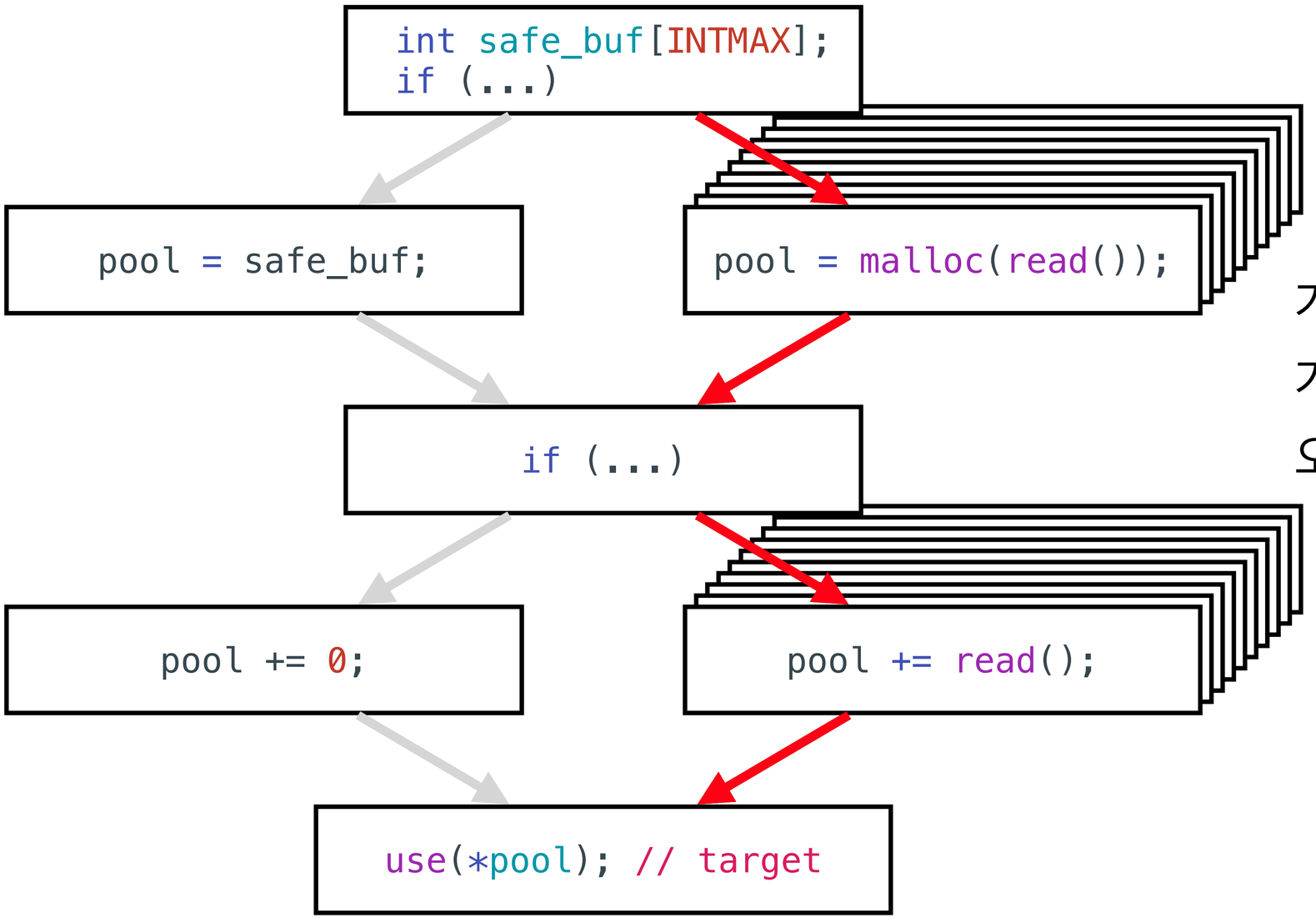
가능한 실행: ∞

문제: 구문 덮이의 과도한 요약

가능한 경로: 1개

가능한 실행: 1개

오류 가능성: 없음



가능한 경로: 1개

가능한 실행: ∞

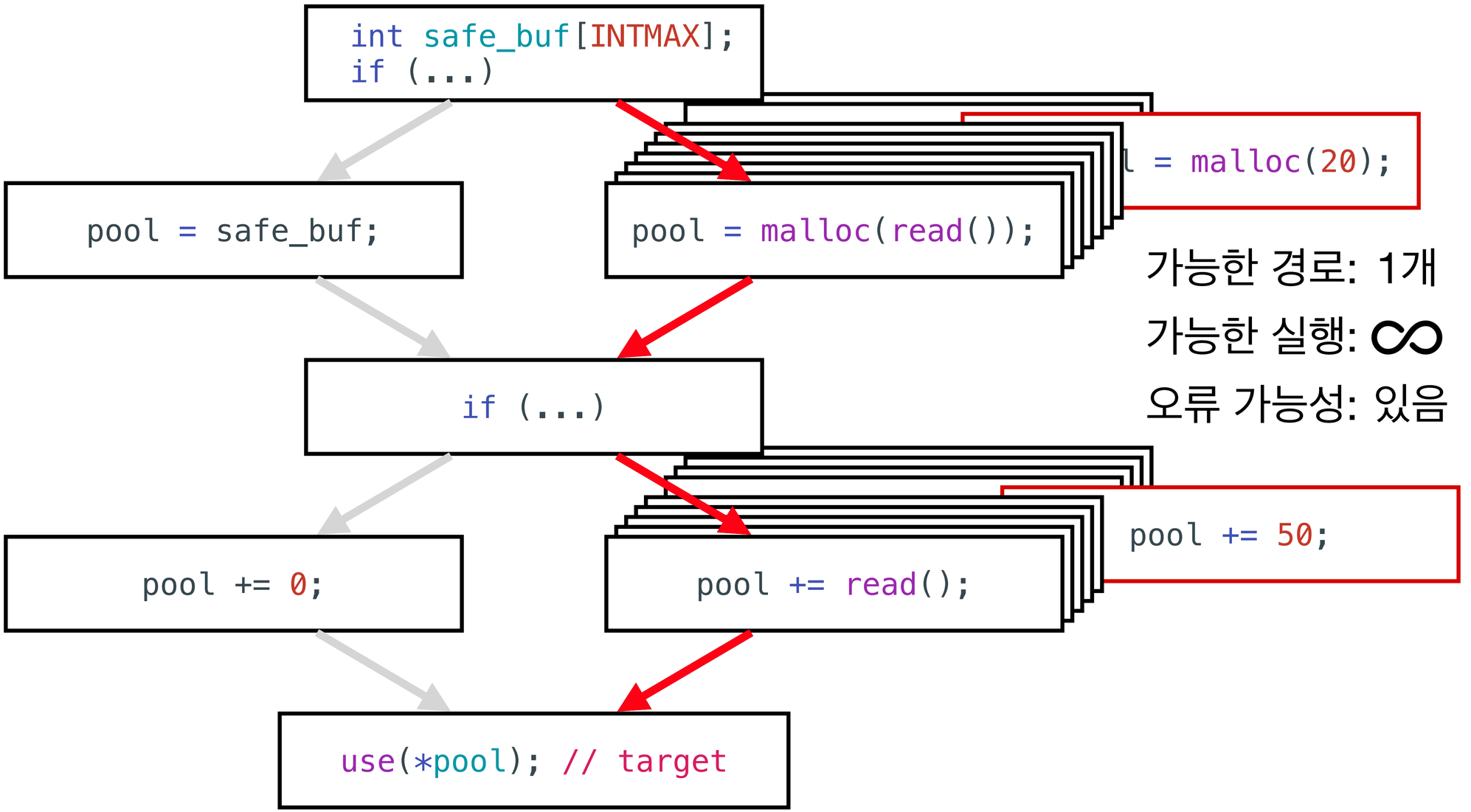
오류 가능성:

문제: 구문 덮이의 과도한 요약

가능한 경로: 1개

가능한 실행: 1개

오류 가능성: 없음



가능한 경로: 1개

가능한 실행: ∞

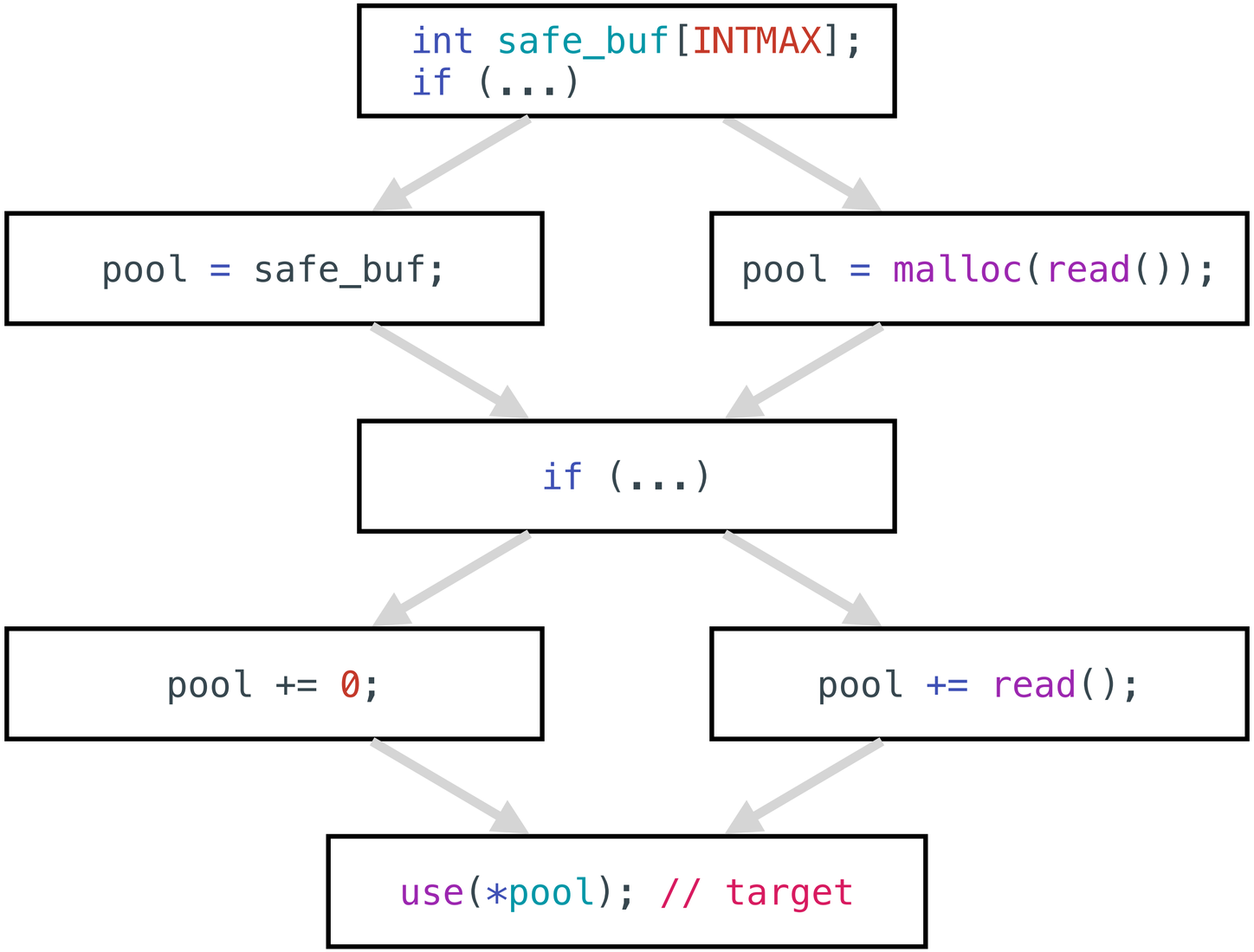
오류 가능성: 있음

문제: 구문 덮이의 과도한 요약

가능한 경로: 1개

가능한 실행: 1개

오류 가능성: 없음



가능한 경로: 1개

가능한 실행: ∞

오류 가능성: 있음

문제: 구문 덮이의 과도한 요약

기록되지 않는 알짜 실행

- 목표와 연관된 유용한 실행들이 전부 하나의 덮이로 요약됨

문제: 구문 덮이의 과도한 요약

기록되지 않는 알짜 실행

- 목표와 연관된 유용한 실행들이 전부 하나의 덮이로 요약됨

필요한 질문

- 어디를 덜 요약할 것인가? → 알짜 실행 지점 선정

문제: 구문 덮이의 과도한 요약

기록되지 않는 알짜 실행

- 목표와 연관된 유용한 실행들이 전부 하나의 덮이로 요약됨

필요한 질문

- 어디를 덜 요약할 것인가? → 알짜 실행 지점 선정
- 어떻게 덜 요약할 것인가? → 지향성 값-맥락 덮이

알짜 실행 지점

알짜 실행 지점

퍼징의 관심사: 메모리 오류

- 할당된 메모리의 크기를 넘어서는 포인터 사용

알짜 실행 지점

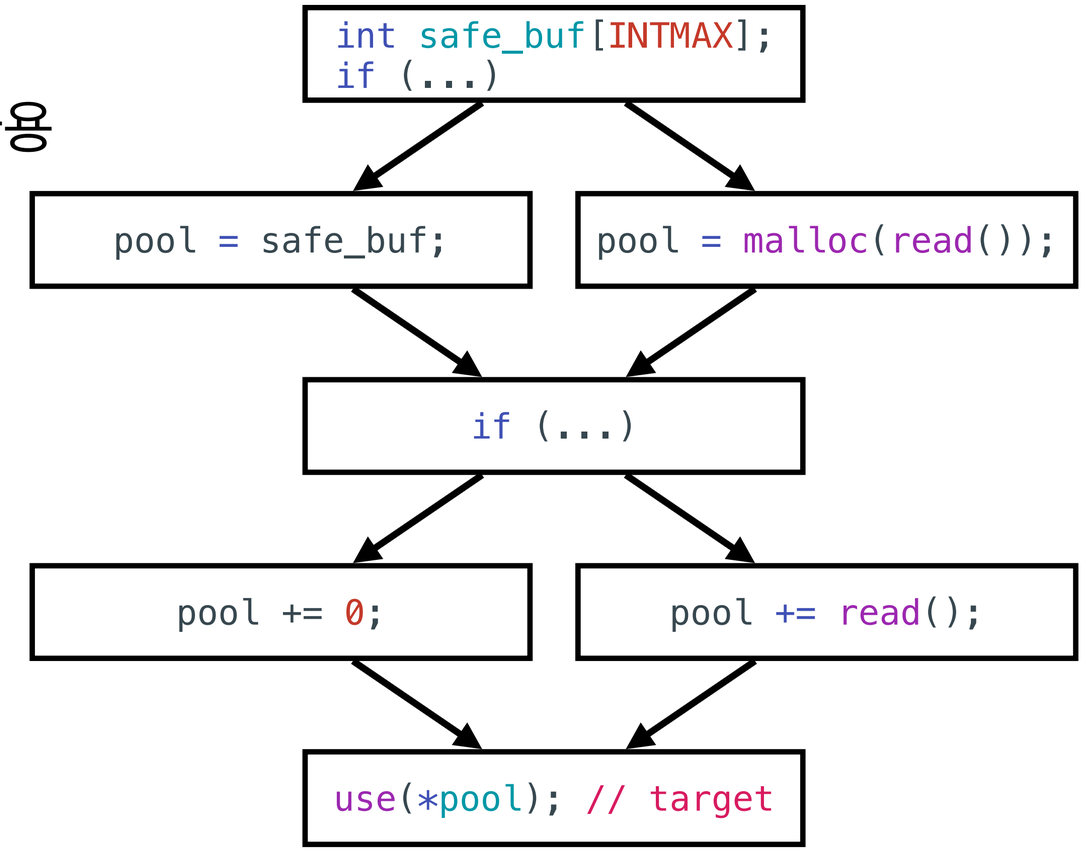
퍼징의 관심사: 메모리 오류

- 할당된 메모리의 크기를 넘어서는 포인터 사용
- 목표 지점에서 사용 되는 메모리와 포인터

알짜 실행 지점

퍼징의 관심사: 메모리 오류

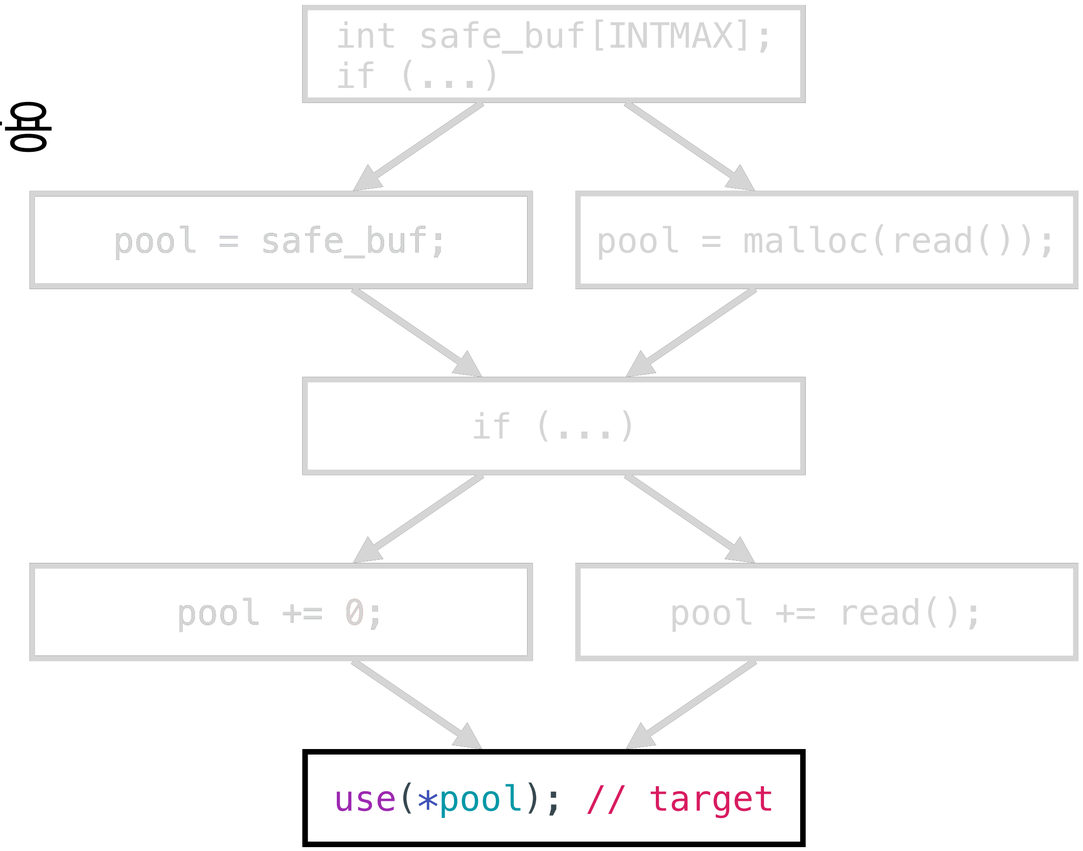
- 할당된 메모리의 크기를 넘어서는 포인터 사용
- 목표 지점에서 사용 되는 메모리와 포인터



알짜 실행 지점

퍼징의 관심사: 메모리 오류

- 할당된 메모리의 크기를 넘어서는 포인터 사용
- 목표 지점에서 사용 되는 메모리와 포인터

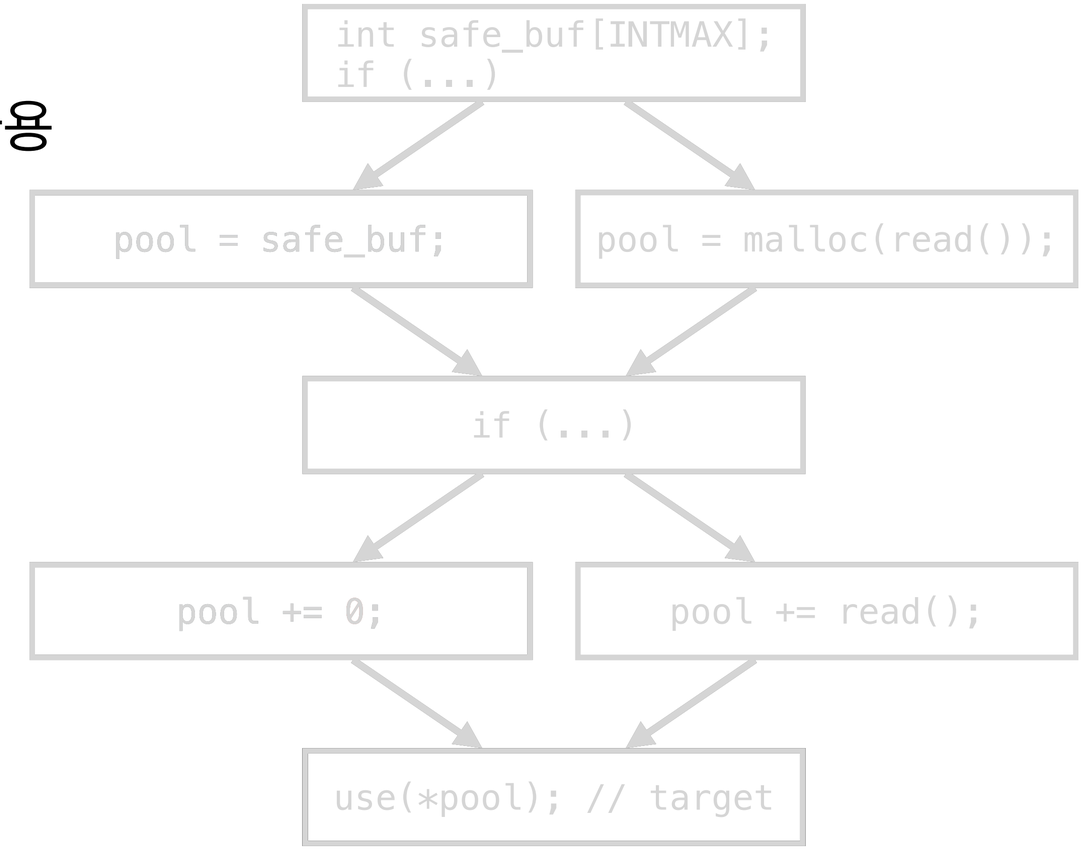


알짜 실행 지점

퍼징의 관심사: 메모리 오류

- 할당된 메모리의 크기를 넘어서는 포인터 사용
- 목표 지점에서 사용 되는 메모리와 포인터

알짜 실행 지점 선정



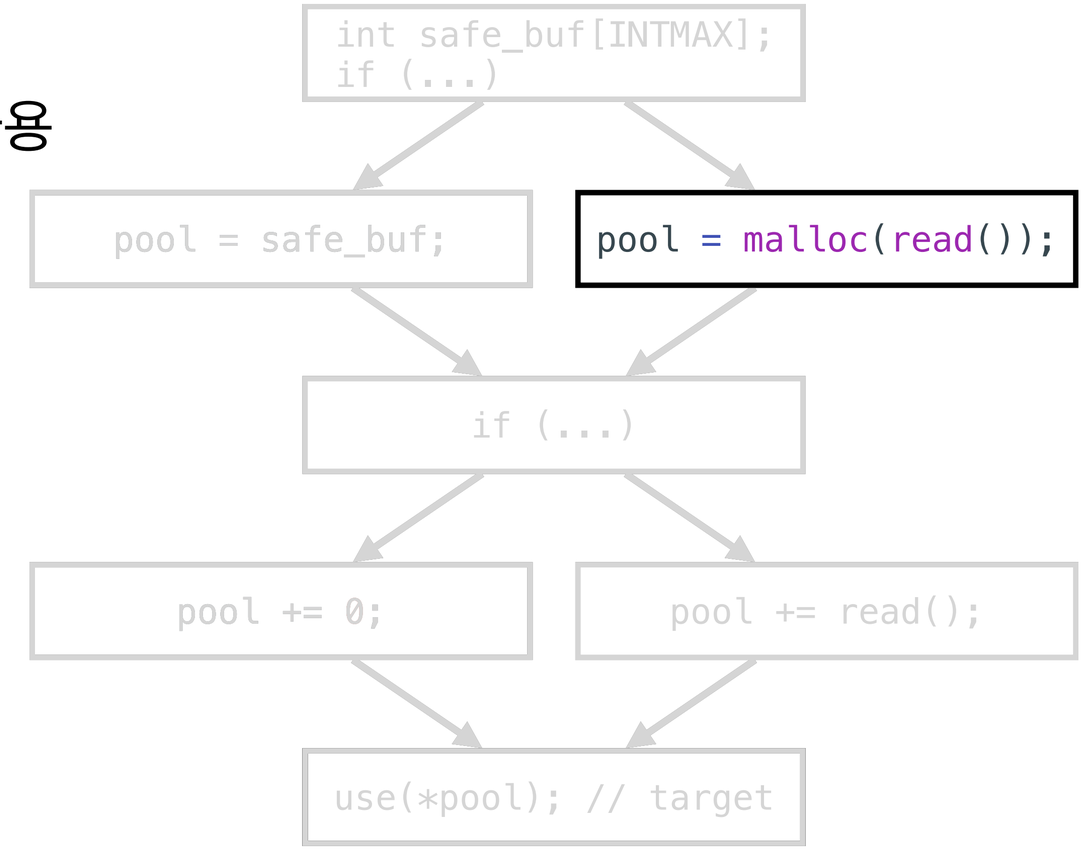
알짜 실행 지점

퍼징의 관심사: 메모리 오류

- 할당된 메모리의 크기를 넘어서는 포인터 사용
- 목표 지점에서 사용 되는 메모리와 포인터

알짜 실행 지점 선정

- 메모리의 할당 지점



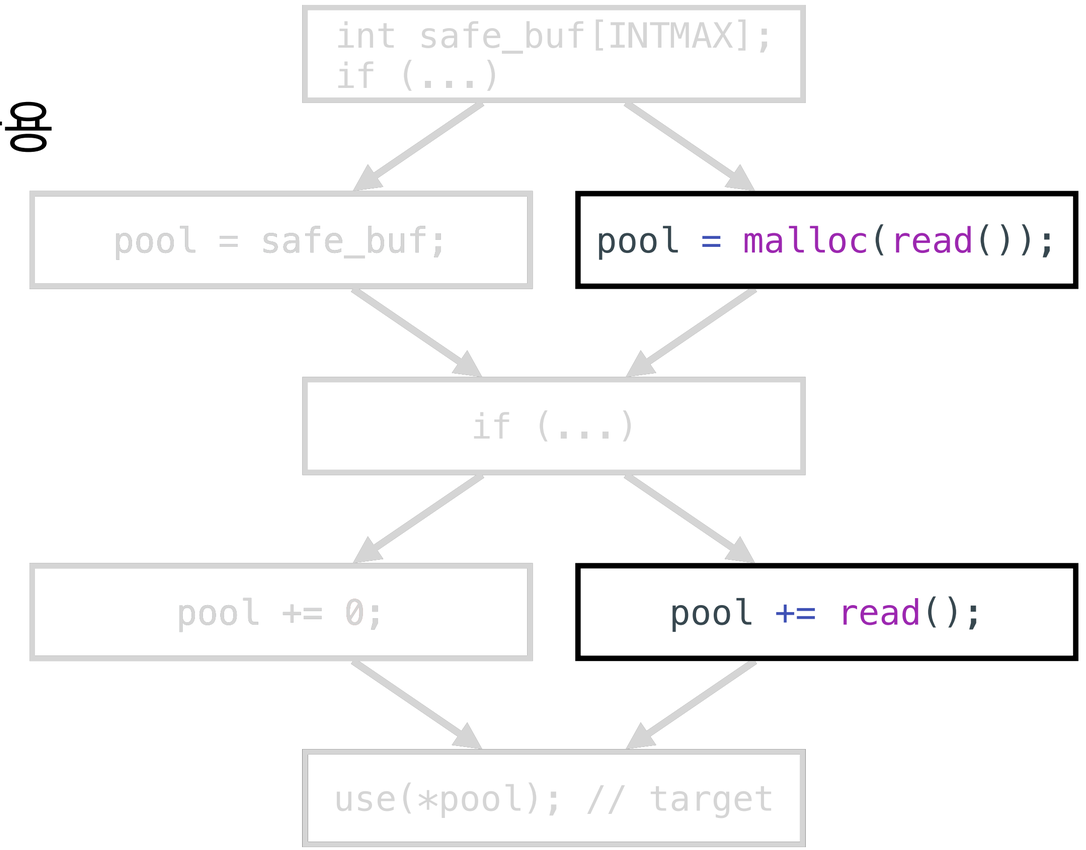
알짜 실행 지점

퍼징의 관심사: 메모리 오류

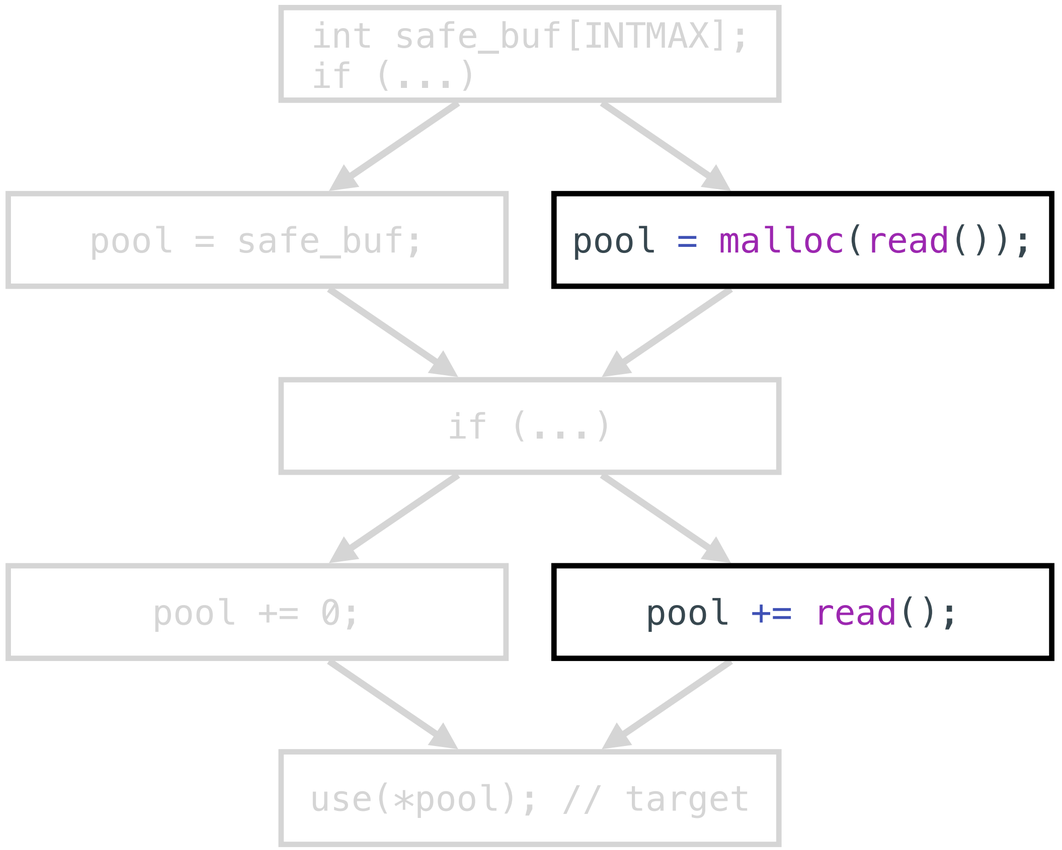
- 할당된 메모리의 크기를 넘어서는 포인터 사용
- 목표 지점에서 사용 되는 메모리와 포인터

알짜 실행 지점 선정

- 메모리의 할당 지점
- 유저 입력을 이용한 포인터의 정의 지점



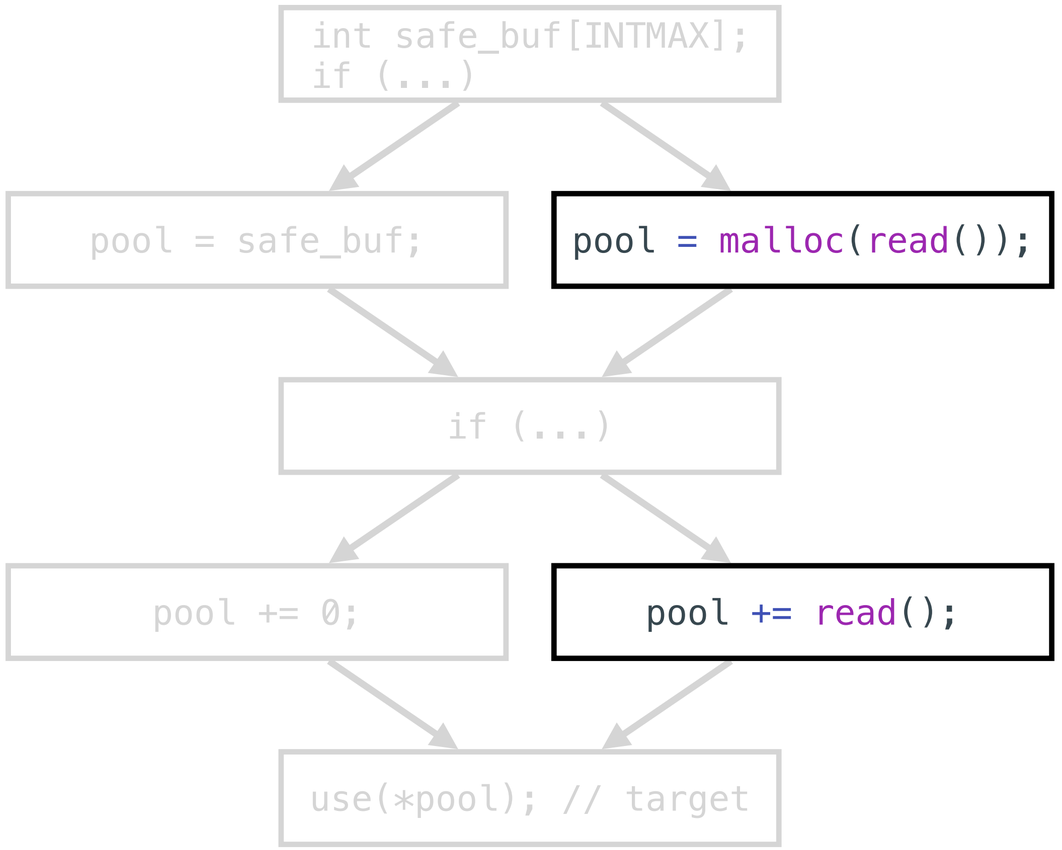
요약 풀어헤치기


int safe_buf[INTMAX];
if (...)
pool = safe_buf;
pool = malloc(read());
if (...)
pool += 0;
pool += read();
use(*pool); // target

요약 풀어헤치기

요약의 최대 피해자: 값

- 전통적 구문 덮이로 관찰 불가

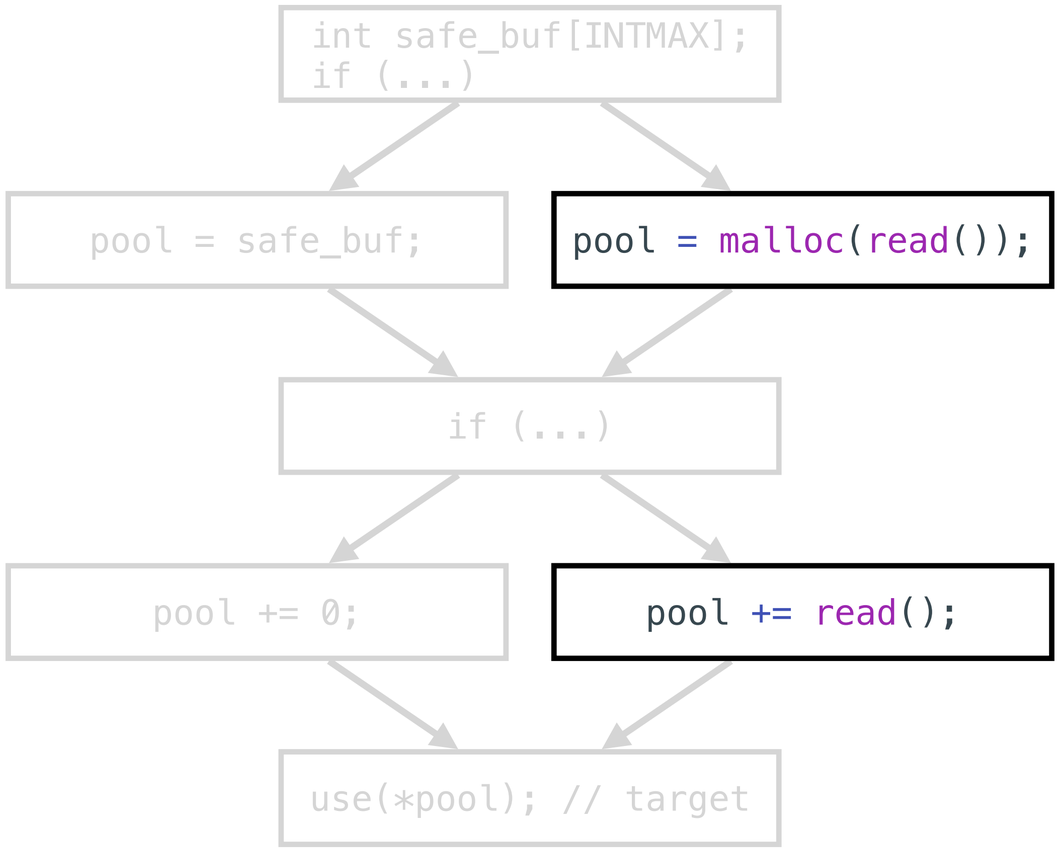


요약 풀어헤치기

요약의 최대 피해자: 값

- 전통적 구문 덮이로 관찰 불가

알짜 실행 지점의 값 관찰:



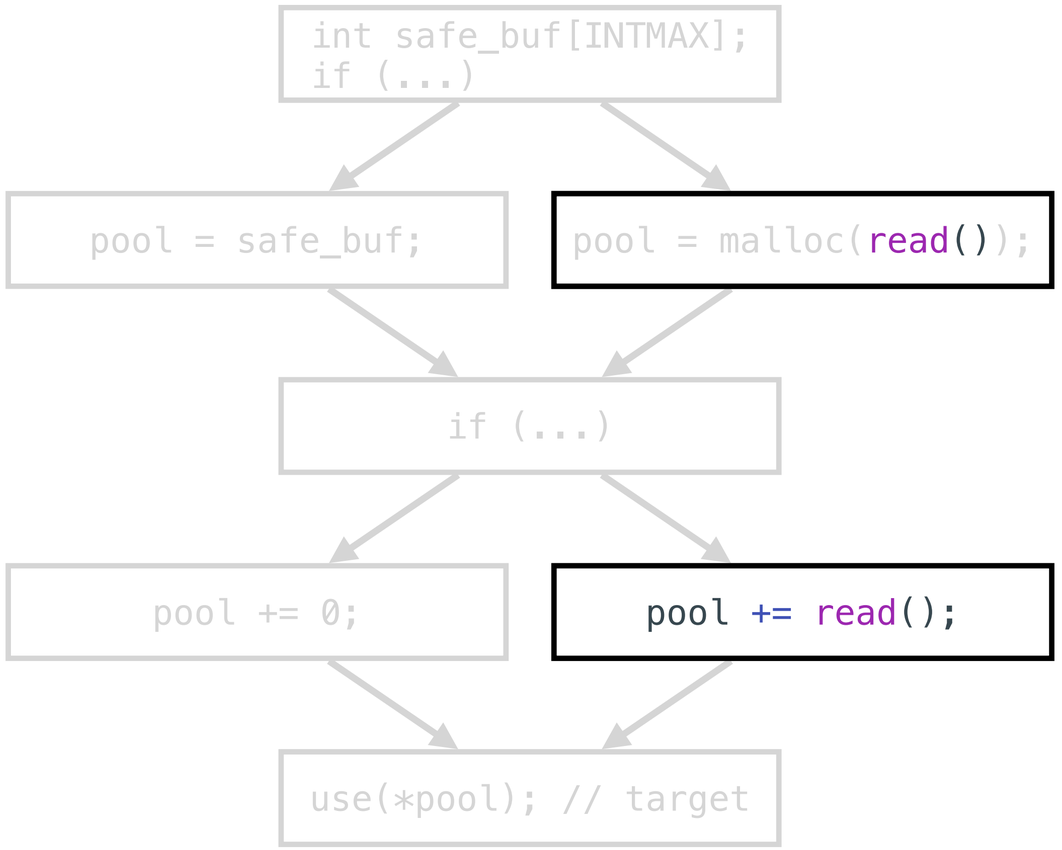
요약 풀어헤치기

요약의 최대 피해자: 값

- 전통적 구문 덮이로 관찰 불가

알짜 실행 지점의 값 관찰:

- 할당된 메모리의 크기



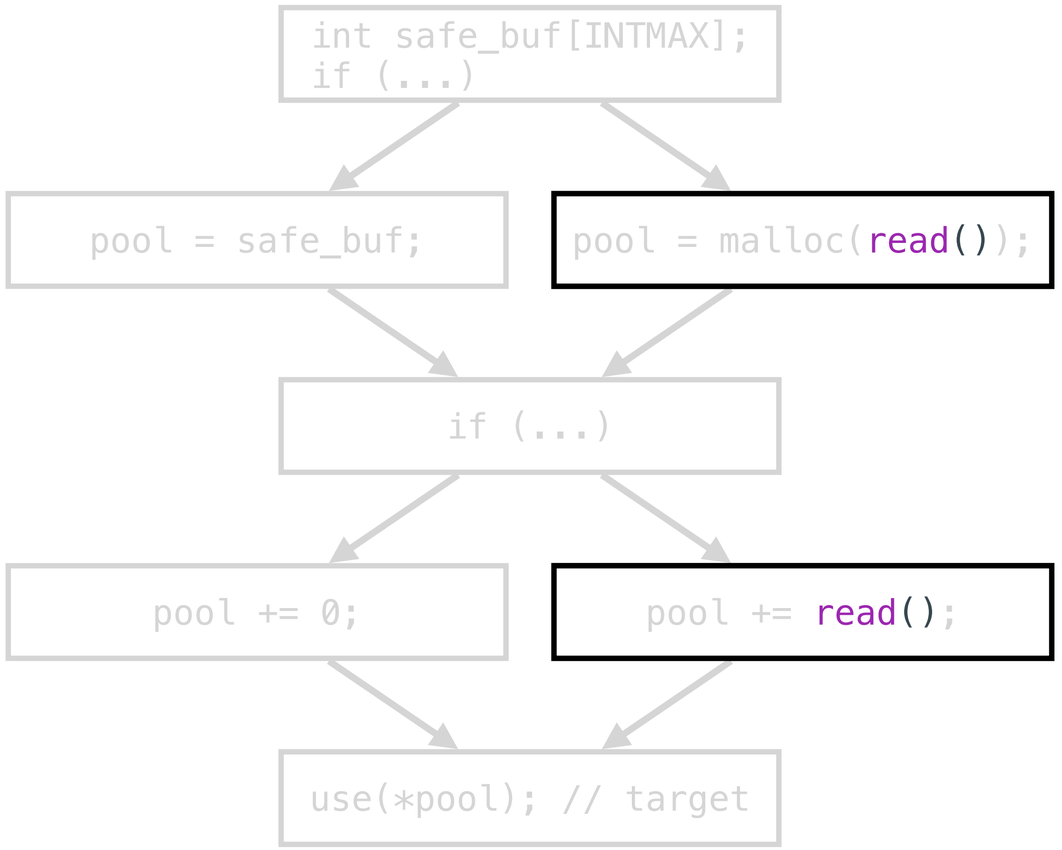
요약 풀어헤치기

요약의 최대 피해자: 값

- 전통적 구문 덮이로 관찰 불가

알짜 실행 지점의 값 관찰:

- 할당된 메모리의 크기
- 포인터를 정의한 유저 입력값



요약 풀어헤치기

요약의 최대 피해자: 값

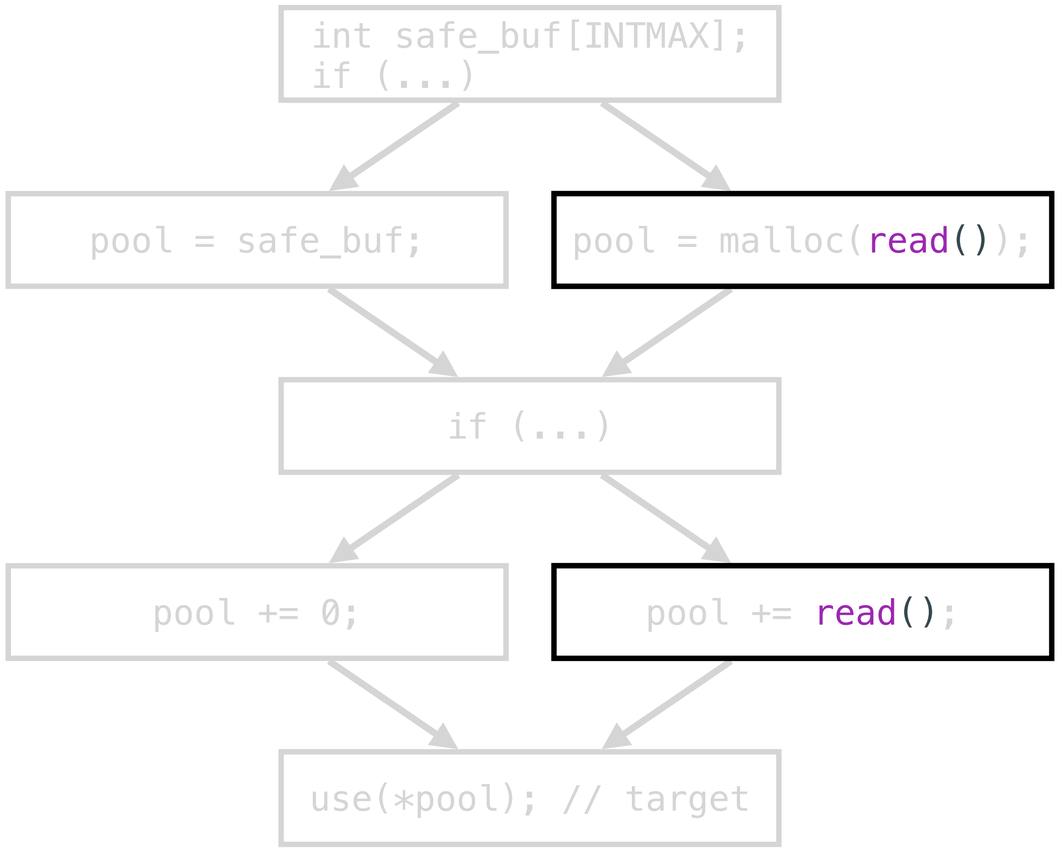
- 전통적 구문 덮이로 관찰 불가

알짜 실행 지점의 값 관찰:

- 할당된 메모리의 크기
- 포인터를 정의한 유저 입력값

지향성 값-맥락 덮이:

- 관찰한 값에 기반한 덮이 구분



요약 풀어헤치기

요약의 최대 피해자: 값

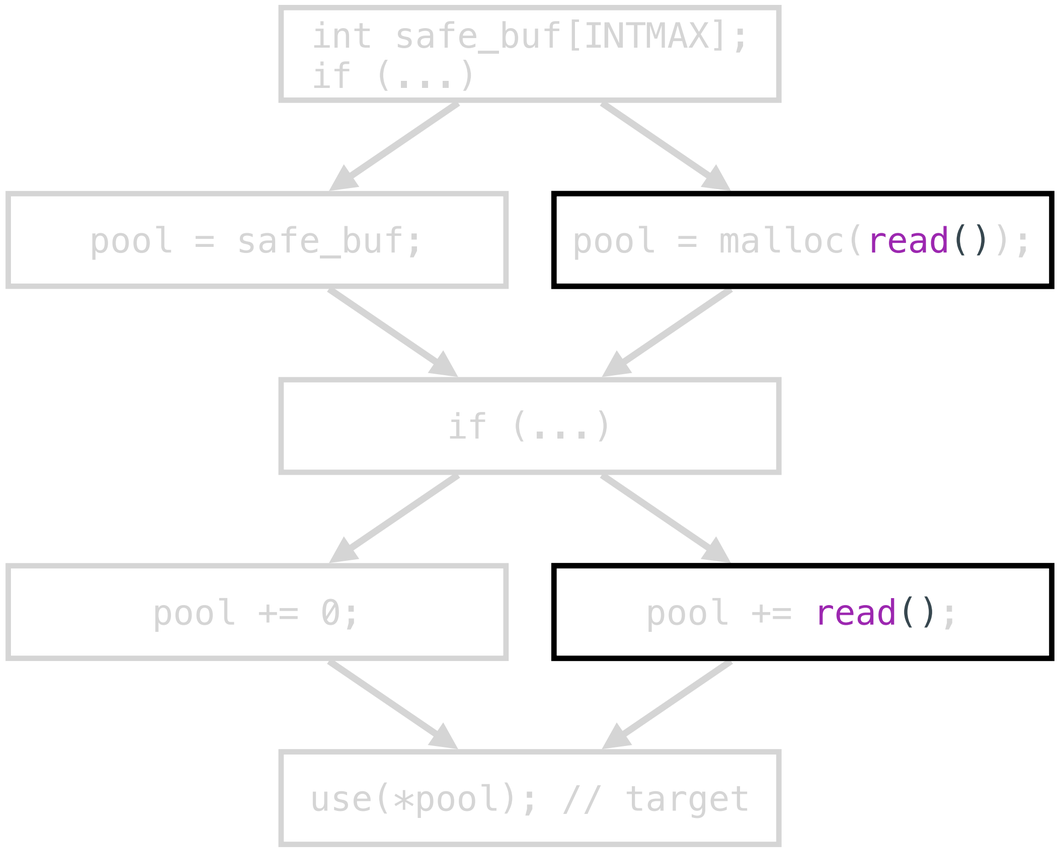
- 전통적 구문 덮이로 관찰 불가

알짜 실행 지점의 값 관찰:

- 할당된 메모리의 크기
- 포인터를 정의한 유저 입력값

지향성 값-맥락 덮이:

- 관찰한 값에 기반한 덮이 구분
- 경로가 같아도 값이 다른 실행 → 다른 덮이



요약 풀어헤치기

요약의 최대 피해자: 값

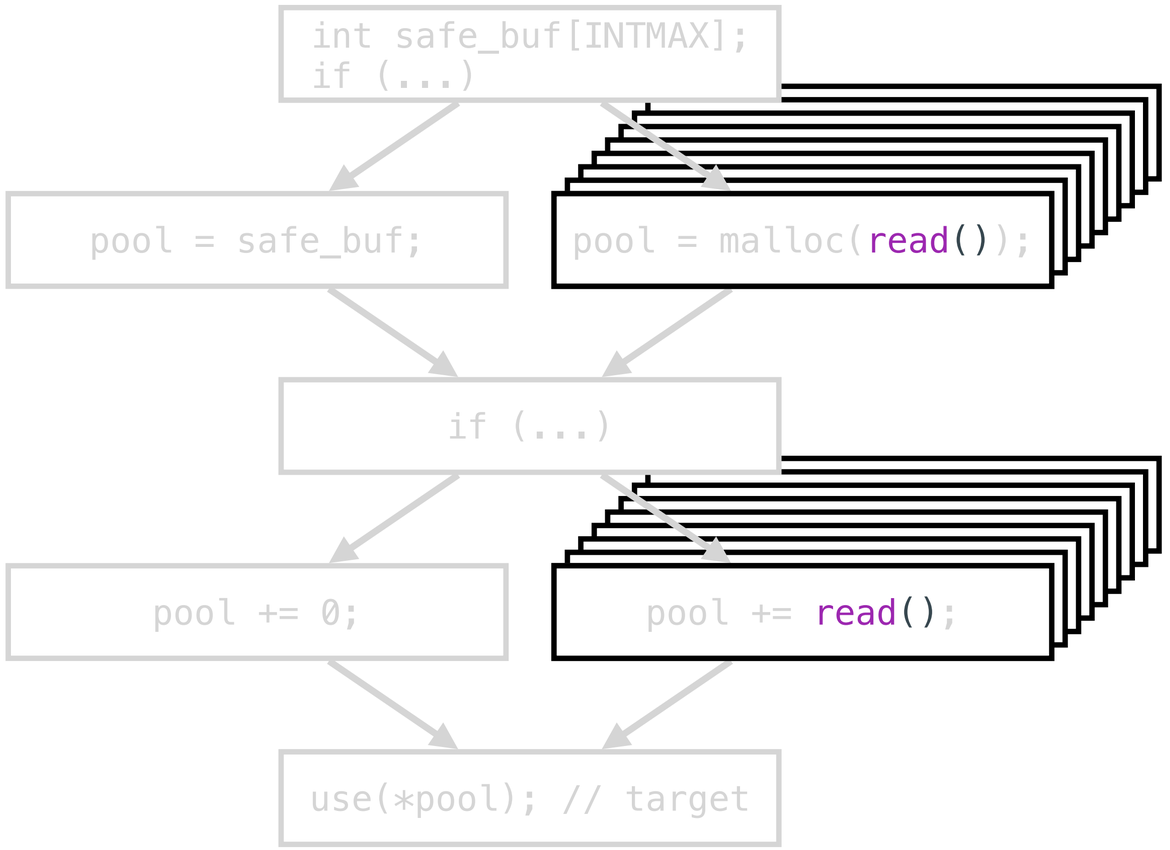
- 전통적 구문 덮이로 관찰 불가

알짜 실행 지점의 값 관찰:

- 할당된 메모리의 크기
- 포인터를 정의한 유저 입력값

지향성 값-맥락 덮이:

- 관찰한 값에 기반한 덮이 구분
- 경로가 같아도 값이 다른 실행 → 다른 덮이



지향성 값-맥락 덮이의 효과

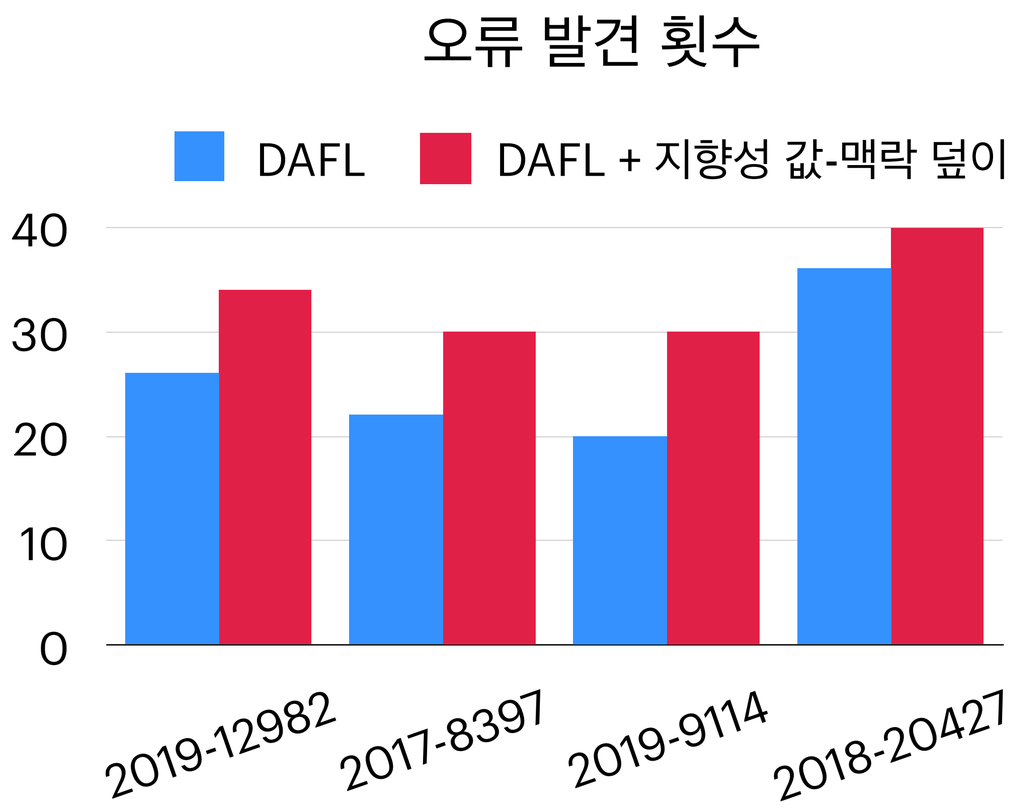
큰 성능 향상

- 더 많은 발견 횟수, 더 빠른 발견 시간

지향성 값-맥락 덮이의 효과

큰 성능 향상

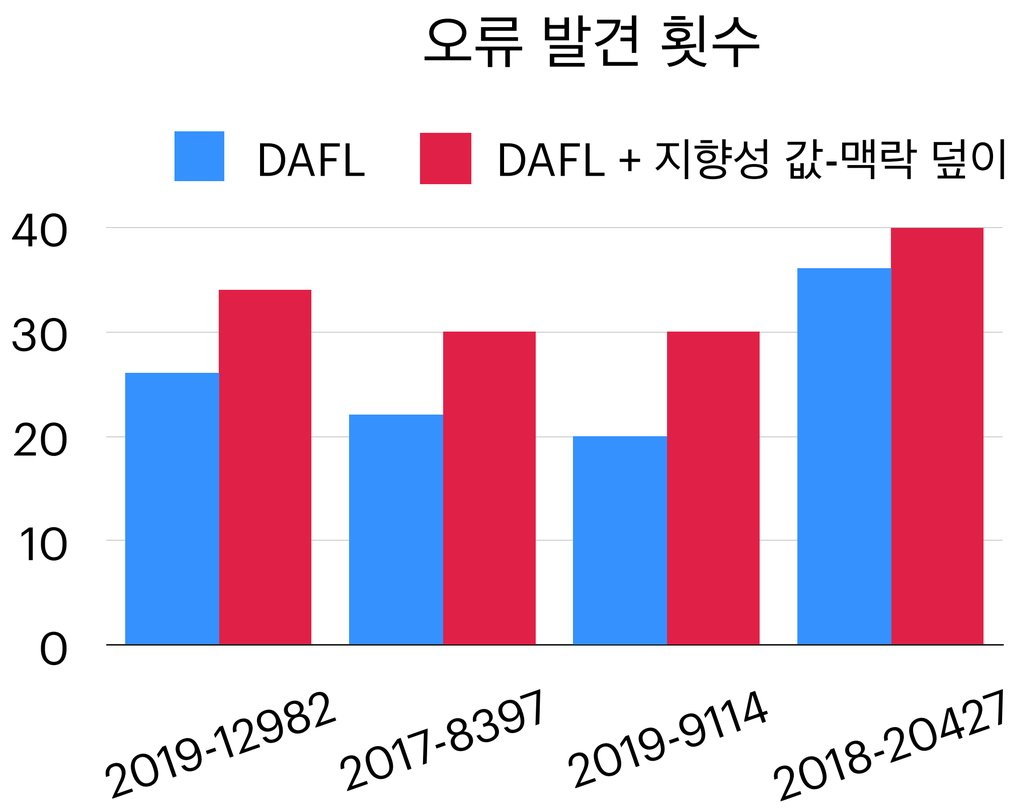
- 더 많은 발견 횟수, 더 빠른 발견 시간

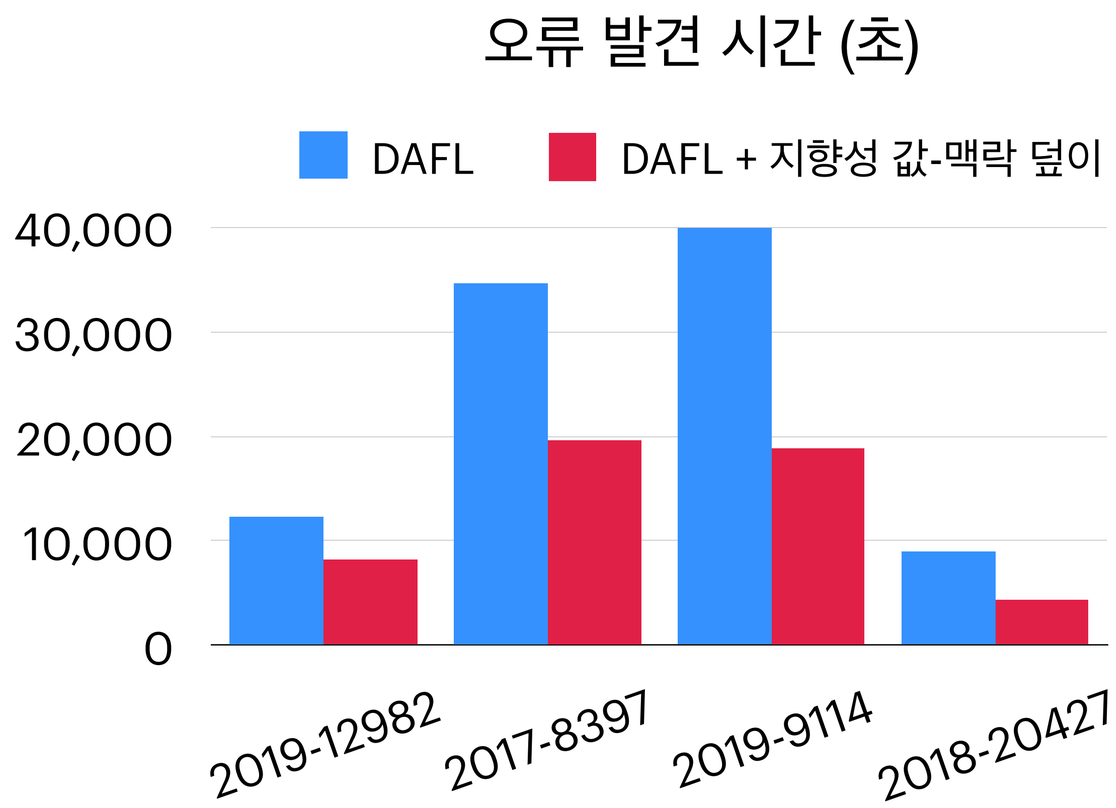


지향성 값-맥락 덮이의 효과

큰 성능 향상

- 더 많은 발견 횟수, 더 빠른 발견 시간





포스터 세션에서...

숨겨진 알짜 실행, 이것만 있으면 찾을 수 있다?

전문가들은 이미 쓰고 있는 지향성 값-맥락 덮이

덮이만 바꿨을 뿐인데, 오류가 콸콸

전문가들도 놀랐다… 그동안 오류 놓쳤던 원인 밝혀져

퍼징 성능 차이, 알고 보니 지향성 값-맥락 덮이였다

숨겨진 알짜 실행, 지향성 값-맥락 덮이로 관찰하세요!

김태은, 최재승, 허기홍, 차상길

SOGANG UNIVERSITY KAIST

1. 개요

배경

문제: 크고 복잡한 소프트웨어 전체를 매번 검사하는 것은 불가능하다

→ 일부 오류 의심 지점을 선정하여 자원을 집중할 필요 발생

지향성 퍼징 (Directed Fuzzing)

목표로 주어진 프로그램 지점에 도달하는 입력을 빠르게 생성하는 기법

오류 의심 지점을 집중적으로 검사할 수 있음

지향성 퍼징의 목표

궁극적인 목표: 목표 지점의 오류 발견

특정한 경로 + 특정한 변수 값으로 목표 지점 도달 필요

무지향성 퍼징에서 지향성 퍼징으로 가는 길

1. 목표 집중적인 시간 투자 — 최신 기술의 현주소

2. 목표를 보다 자세히 관찰

무지향성: 모든 프로그램 지점을 동일한 수준에서 관찰

지향성: 목표와 연관된 지점에서 보다 상세한 실행 정보 관찰

2. 구문 덮이의 한계

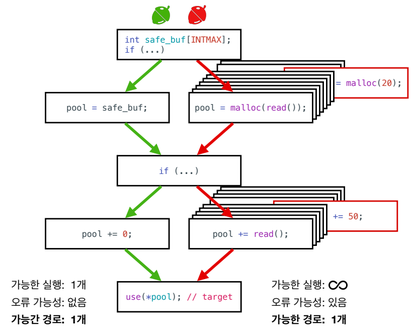
덮이 (Coverage)

주어진 입력(들)이 프로그램을 얼마나 잘 탐색했는지 알려주는 지표

새로운 덮이 달성을 기준으로 시드 입력 채택 여부를 결정

무지향성 퍼징은 구문 수준에서 실행된 분기의 집합으로 덮이를 평가함

예제: CVE-2018-7868 (swftophp)



문제: 구문 덮이의 과도한 요약

유저 입력에 따라 변화하는 무수한 실행이 단 하나의 실행으로 요약됨

오류 유발과 연관된 알짜 실행도 예외 없이 요약

→ 목표 지점과 연관된 지점들에서는 더 자세한 실행을 관찰해야 함

해결: 과도한 요약 풀어헤치기

어디를 덜 요약할 것이가? → 알짜 실행 지점

어떻게 덜 요약할 것인가? → 지향성 값-맥락 덮이

3. 알짜 실행 지점

관심사: 메모리 오류

할당된 메모리의 크기를 넘어서는 포인터 사용

요주의 변수: 목표 지점에서 사용되는 메모리와 포인터

알짜 실행 지점 선정

- 메모리의 할당 지점 `pool = malloc(read());`
- 유저 입력을 이용한 포인터 정의 지점 `pool += read();`

3. 지향성 값-맥락 덮이

값 관찰을 통한 요약 풀어헤치기

값: 요약의 최대 피해자 → 구문 덮이로 관찰 불가

알짜 실행 지점의 값 관찰

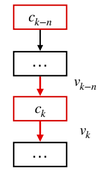
- 할당된 메모리의 크기 `pool = malloc(read());`
- 포인터를 정의한 유저 입력값 `pool += read();`

값을 관찰하는 새로운 덮이

무지향성 구문 덮이: $\{(c_1, c_2) \mid c \in \mathbb{C}\}$

지향성 값-맥락 덮이: $\{((c_1, c_2), v) \mid c \in \mathbb{C}, v \in \mathbb{V}\}$

- $\mathbb{C}$는 모든 프로그램 지점들
- $\mathbb{V}$는 값 요약 도메인 (원본 값의 $\log_2$)
- c_1 혹은 c_2가 알짜 실행 지점일 때, 가장 최근 관찰된 v 사용



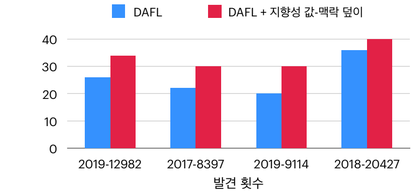
4. 평가

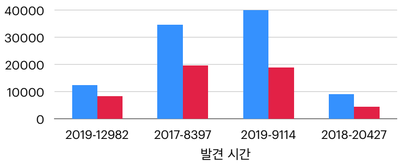
최신 지향성 퍼징 도구인 DAFL[1]에 지향성 값-맥락 커버리지를 도입한 실험

1. 대상: 4개의 알려진 오류 (CVE)

2. 목표: 오류 위치를 목표 지점으로 삼고 해당 오류를 재현하는 입력 생성

3. 평가: 12시간씩 40번의 반복실험 후 오류 발견 횟수와 오류 재현 시간 비교





1. DAFL : Directed Grey-Box Fuzzing Guided by Data Dependency, Kim et al., USENIX Security 2023