

Static Analysis-based Repair of Memory Errors in C Programs

Hakjoo Oh
Korea University

2/25/2025@IFIP WG 2.4 Meeting 70 (Singapore)

PL/SE Research @Korea Univ.

- **Members:** ~15 graduate students
- **Research area:** intersection of programming languages (PL) and software engineering (SE)
 - program analysis and testing
 - program synthesis and repair
- **Publication:** PL, SE, and Security
 - **PL:** POPL('22), PLDI('12,'14,'20,'24), OOPSLA('15,'17a,'17b,'18a,'18b,'19,'20,'23,'24a,'24b,25)
 - **SE:** ICSE('17,'18,'19,'20,'21'22a,'22b,'23a,'23b,'23c), FSE('18,'19,'20,'21,'22,'23), ASE('18,'24a,'24b)
 - **Security:** IEEE S&P('17,'20), USENIX Security('21,'23)



<http://kupl.github.io>

SEOUL

A'REX
(Airport Railroad)
Operating Hours
05:15-23:50



Korea Univ.

Sightseeing

Local Hot Spot

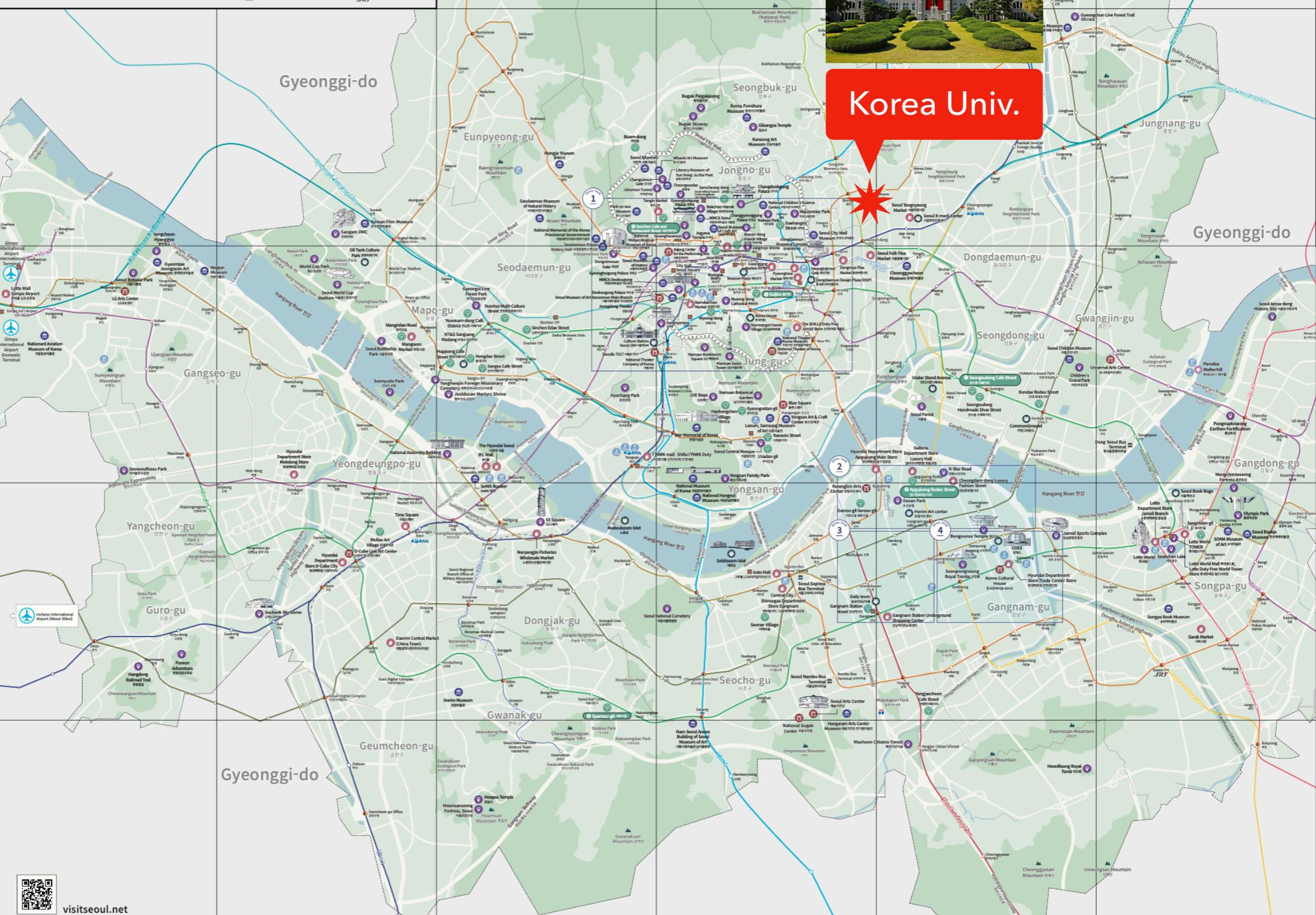
- 01 10th Steps
- 02 April 19th National Cemetery
- 03 G3 Square
- 04 Gyeongbokgung Palace
- 05 Gyeongju Line Forest Park
- 06 Gyeongju Line Forest Trail
- 07 Gyeongju Palace
- 08 Gyeongju Sky Dome
- 09 Gyeongju National Cemetery
- 10 National Assembly Building
- 11 Gilsangsa Temple
- 12 Naksan Park
- 13 Namangol Hanok Village
- 14 Namangol Hanok Village Street
- 15 Namangol Hanok Square
- 16 Namangol Hanok Tower
- 17 Namsan Botanical Garden
- 18 Namsan Tower
- 19 Namsan Tower
- 20 Namsan Tower
- 21 Namsan Tower
- 22 Namsan Tower
- 23 Namsan Tower
- 24 Namsan Tower
- 25 Namsan Tower
- 26 Namsan Tower
- 27 Namsan Tower
- 28 Namsan Tower
- 29 Namsan Tower
- 30 Namsan Tower
- 31 Namsan Tower
- 32 Namsan Tower
- 33 Namsan Tower
- 34 Namsan Tower
- 35 Namsan Tower
- 36 Namsan Tower
- 37 Namsan Tower
- 38 Namsan Tower
- 39 Namsan Tower
- 40 Namsan Tower
- 41 Namsan Tower
- 42 Namsan Tower
- 43 Namsan Tower
- 44 Namsan Tower
- 45 Namsan Tower
- 46 Namsan Tower
- 47 Namsan Tower
- 48 Namsan Tower
- 49 Namsan Tower
- 50 Namsan Tower
- 51 Namsan Tower
- 52 Namsan Tower
- 53 Namsan Tower
- 54 Namsan Tower
- 55 Namsan Tower
- 56 Namsan Tower
- 57 Namsan Tower
- 58 Namsan Tower
- 59 Namsan Tower
- 60 Namsan Tower
- 61 Namsan Tower
- 62 Namsan Tower
- 63 Namsan Tower
- 64 Namsan Tower
- 65 Namsan Tower
- 66 Namsan Tower
- 67 Namsan Tower
- 68 Namsan Tower
- 69 Namsan Tower
- 70 Namsan Tower
- 71 Namsan Tower
- 72 Namsan Tower
- 73 Namsan Tower
- 74 Namsan Tower
- 75 Namsan Tower
- 76 Namsan Tower
- 77 Namsan Tower
- 78 Namsan Tower
- 79 Namsan Tower
- 80 Namsan Tower
- 81 Namsan Tower
- 82 Namsan Tower
- 83 Namsan Tower
- 84 Namsan Tower
- 85 Namsan Tower
- 86 Namsan Tower
- 87 Namsan Tower
- 88 Namsan Tower
- 89 Namsan Tower
- 90 Namsan Tower
- 91 Namsan Tower
- 92 Namsan Tower
- 93 Namsan Tower
- 94 Namsan Tower
- 95 Namsan Tower
- 96 Namsan Tower
- 97 Namsan Tower
- 98 Namsan Tower
- 99 Namsan Tower
- 100 Namsan Tower

Museum/Gallery

- 01 Gyeongju-gil Seonje-gil
- 02 Gyeongju-gil Seonje-gil
- 03 Gyeongju-gil Seonje-gil
- 04 Gyeongju-gil Seonje-gil
- 05 Gyeongju-gil Seonje-gil
- 06 Gyeongju-gil Seonje-gil
- 07 Gyeongju-gil Seonje-gil
- 08 Gyeongju-gil Seonje-gil
- 09 Gyeongju-gil Seonje-gil
- 10 Gyeongju-gil Seonje-gil
- 11 Gyeongju-gil Seonje-gil
- 12 Gyeongju-gil Seonje-gil
- 13 Gyeongju-gil Seonje-gil
- 14 Gyeongju-gil Seonje-gil
- 15 Gyeongju-gil Seonje-gil
- 16 Gyeongju-gil Seonje-gil
- 17 Gyeongju-gil Seonje-gil
- 18 Gyeongju-gil Seonje-gil
- 19 Gyeongju-gil Seonje-gil
- 20 Gyeongju-gil Seonje-gil
- 21 Gyeongju-gil Seonje-gil
- 22 Gyeongju-gil Seonje-gil
- 23 Gyeongju-gil Seonje-gil
- 24 Gyeongju-gil Seonje-gil
- 25 Gyeongju-gil Seonje-gil
- 26 Gyeongju-gil Seonje-gil
- 27 Gyeongju-gil Seonje-gil
- 28 Gyeongju-gil Seonje-gil
- 29 Gyeongju-gil Seonje-gil
- 30 Gyeongju-gil Seonje-gil
- 31 Gyeongju-gil Seonje-gil
- 32 Gyeongju-gil Seonje-gil
- 33 Gyeongju-gil Seonje-gil
- 34 Gyeongju-gil Seonje-gil
- 35 Gyeongju-gil Seonje-gil
- 36 Gyeongju-gil Seonje-gil
- 37 Gyeongju-gil Seonje-gil
- 38 Gyeongju-gil Seonje-gil
- 39 Gyeongju-gil Seonje-gil
- 40 Gyeongju-gil Seonje-gil
- 41 Gyeongju-gil Seonje-gil
- 42 Gyeongju-gil Seonje-gil
- 43 Gyeongju-gil Seonje-gil
- 44 Gyeongju-gil Seonje-gil
- 45 Gyeongju-gil Seonje-gil
- 46 Gyeongju-gil Seonje-gil
- 47 Gyeongju-gil Seonje-gil
- 48 Gyeongju-gil Seonje-gil
- 49 Gyeongju-gil Seonje-gil
- 50 Gyeongju-gil Seonje-gil
- 51 Gyeongju-gil Seonje-gil
- 52 Gyeongju-gil Seonje-gil
- 53 Gyeongju-gil Seonje-gil
- 54 Gyeongju-gil Seonje-gil
- 55 Gyeongju-gil Seonje-gil
- 56 Gyeongju-gil Seonje-gil
- 57 Gyeongju-gil Seonje-gil
- 58 Gyeongju-gil Seonje-gil
- 59 Gyeongju-gil Seonje-gil
- 60 Gyeongju-gil Seonje-gil
- 61 Gyeongju-gil Seonje-gil
- 62 Gyeongju-gil Seonje-gil
- 63 Gyeongju-gil Seonje-gil
- 64 Gyeongju-gil Seonje-gil
- 65 Gyeongju-gil Seonje-gil
- 66 Gyeongju-gil Seonje-gil
- 67 Gyeongju-gil Seonje-gil
- 68 Gyeongju-gil Seonje-gil
- 69 Gyeongju-gil Seonje-gil
- 70 Gyeongju-gil Seonje-gil
- 71 Gyeongju-gil Seonje-gil
- 72 Gyeongju-gil Seonje-gil
- 73 Gyeongju-gil Seonje-gil
- 74 Gyeongju-gil Seonje-gil
- 75 Gyeongju-gil Seonje-gil
- 76 Gyeongju-gil Seonje-gil
- 77 Gyeongju-gil Seonje-gil
- 78 Gyeongju-gil Seonje-gil
- 79 Gyeongju-gil Seonje-gil
- 80 Gyeongju-gil Seonje-gil
- 81 Gyeongju-gil Seonje-gil
- 82 Gyeongju-gil Seonje-gil
- 83 Gyeongju-gil Seonje-gil
- 84 Gyeongju-gil Seonje-gil
- 85 Gyeongju-gil Seonje-gil
- 86 Gyeongju-gil Seonje-gil
- 87 Gyeongju-gil Seonje-gil
- 88 Gyeongju-gil Seonje-gil
- 89 Gyeongju-gil Seonje-gil
- 90 Gyeongju-gil Seonje-gil
- 91 Gyeongju-gil Seonje-gil
- 92 Gyeongju-gil Seonje-gil
- 93 Gyeongju-gil Seonje-gil
- 94 Gyeongju-gil Seonje-gil
- 95 Gyeongju-gil Seonje-gil
- 96 Gyeongju-gil Seonje-gil
- 97 Gyeongju-gil Seonje-gil
- 98 Gyeongju-gil Seonje-gil
- 99 Gyeongju-gil Seonje-gil
- 100 Gyeongju-gil Seonje-gil

Performance Hall

- 01 KTG&Sanggung Healing
- 02 N Seoul Tower
- 03 Dongdaemun Design Plaza (DDP)
- 04 Oil Tank Culture Park
- 05 Culture Station Seoul 284
- 06 Seoul Book Bogo
- 07 Seoul K-medi Center
- 08 Sebitseom Island
- 09 Under One Avenue
- 10 Daily Team
- 11 CommonGround
- 12 COEX
- 13 Korea House
- 14 Horim Art center
- 15 KTG&Sanggung Healing
- 16 N Seoul Tower
- 17 Dongdaemun Design Plaza (DDP)
- 18 Oil Tank Culture Park
- 19 Culture Station Seoul 284
- 20 Seoul Book Bogo
- 21 Seoul K-medi Center
- 22 Sebitseom Island
- 23 Under One Avenue
- 24 Daily Team
- 25 CommonGround
- 26 COEX
- 27 Korea House
- 28 Horim Art center
- 29 KTG&Sanggung Healing
- 30 N Seoul Tower
- 31 Dongdaemun Design Plaza (DDP)
- 32 Oil Tank Culture Park
- 33 Culture Station Seoul 284
- 34 Seoul Book Bogo
- 35 Seoul K-medi Center
- 36 Sebitseom Island
- 37 Under One Avenue
- 38 Daily Team
- 39 CommonGround
- 40 COEX
- 41 Korea House
- 42 Horim Art center
- 43 KTG&Sanggung Healing
- 44 N Seoul Tower
- 45 Dongdaemun Design Plaza (DDP)
- 46 Oil Tank Culture Park
- 47 Culture Station Seoul 284
- 48 Seoul Book Bogo
- 49 Seoul K-medi Center
- 50 Sebitseom Island
- 51 Under One Avenue
- 52 Daily Team
- 53 CommonGround
- 54 COEX
- 55 Korea House
- 56 Horim Art center
- 57 KTG&Sanggung Healing
- 58 N Seoul Tower
- 59 Dongdaemun Design Plaza (DDP)
- 60 Oil Tank Culture Park
- 61 Culture Station Seoul 284
- 62 Seoul Book Bogo
- 63 Seoul K-medi Center
- 64 Sebitseom Island
- 65 Under One Avenue
- 66 Daily Team
- 67 CommonGround
- 68 COEX
- 69 Korea House
- 70 Horim Art center
- 71 KTG&Sanggung Healing
- 72 N Seoul Tower
- 73 Dongdaemun Design Plaza (DDP)
- 74 Oil Tank Culture Park
- 75 Culture Station Seoul 284
- 76 Seoul Book Bogo
- 77 Seoul K-medi Center
- 78 Sebitseom Island
- 79 Under One Avenue
- 80 Daily Team
- 81 CommonGround
- 82 COEX
- 83 Korea House
- 84 Horim Art center
- 85 KTG&Sanggung Healing
- 86 N Seoul Tower
- 87 Dongdaemun Design Plaza (DDP)
- 88 Oil Tank Culture Park
- 89 Culture Station Seoul 284
- 90 Seoul Book Bogo
- 91 Seoul K-medi Center
- 92 Sebitseom Island
- 93 Under One Avenue
- 94 Daily Team
- 95 CommonGround
- 96 COEX
- 97 Korea House
- 98 Horim Art center
- 99 KTG&Sanggung Healing
- 100 N Seoul Tower



visitseoul.net

SEOUL



A'REX
(Airport Railroad)
Operating Hours

**Operating Hours
05:15-23:50**



Sightseeing

- 100 Steps
 - April 19th National Cemetery
 - Arirang Cultural Center
 - Gyeongju Palace
 - Gyeongju Line Forest Park
 - Gyeongju Line Forest Trail
 - Gyeongju Palace
 - Goacheok Sky Dome
 - Gwanghwamun
 - Gyeong National Cemetery
 - Gyeongju National Building
 - Gyeongju Temple
 - Haksan Park
 - Hanok Village
 - Nemangji Hanok Village
 - Namsan Baseline Observatory
 - Namsan Seoul Tower
 - Namsan Botanical Garden
 - Deoksugung Palace
 - Dongdaemun Gate
 - Dongdaemun Design Plaza
 - Earth World
 - Lotte World Tower
 - Marriwon Park
 - Myeong-dong Cathedral
 - Hongjeong Fortress
 - Bosingak Bell
 - Bongeunsa Temple
 - Dream Forest
 - Bugaksan Mountain
 - Bukchon Folk Village
 - Dulciken Hanok Village
 - Sungsan DMZ
 - Seonjeongneung
 - Seoul Hiking Tourism Center
 - Seoul Tower
 - Seoul Forest
 - Seoul National University
 - Aesop's Ansan-Hieng Village
 - Seoul World Cup Stadium
 - Seoul Bethlehem Park
 - Hoamseongpo Forests, Seoul
 - Seokchon Lake
 - Seonyudo Park
 - Seongjeongneung Royal Tombs
 - Sungryemun
 - Sunmangji Hanok Village
 - City Hall, Seoul Square
 - Yongsheon Hyanggyo
 - Yanghwadaegak
 - Children's Grand Park
 - Olympic Park
 - Yongsan Family Park
 - World Cup Park
 - European Village
 - Gyeongju National Museum
 - Seoul Central Mosque
 - Seoul Central Mosque
 - Jahamun Tunnel
 - Jamil Sports Complex
 - Jeodulsan Martyrs Shrine
 - Jigyesa Temple
 - Jingwasa Temple
 - Jungnangcheon Palace
 - Cheonggyecheon Palace
 - Cheonggyeum
 - Chongnyeong
 - Pungnaygoong Palace
 - Eungbong Mountain
 - Heonanseong Royal Tomb
 - Hoesa Temple
 - Hyechang Park
 - Heungjimun Gate

Local Hot Spot

- | | |
|----|--|
| D4 | Gangnam-gil Serene gil
Gangnam Street |
| E4 | Daehan-dong Street |
| F4 | Gyeonggi-do |
| D5 | Dashington Street |
| B4 | Mangridan Road |
| M4 | Myeong-dong |
| M4 | Mulfus Art Village |
| B4 | Baum-dong |
| C4 | Chungmung-dong |
| C4 | Sangju Cafe Street |
| C4 | Symnur |
| D4 | Searae Village |
| D4 | Seochon Cafe and Restaurant Street |
| E4 | Seungsundong Handmade Shoe Street |
| E4 | Seungsundong Cafe Street |
| G4 | Songgrang-dong |
| H4 | Songpa-dong Street |
| A4 | Apjunggwan Ryden Street |
| E4 | Vangjecheon Cafe Street |
| C4 | Yeomnam-dong Cafè District |
| C4 | Yeoheui Multi-Culture Street |
| B4 | Usadang-gil |
| B4 | Tutjeong |
| B4 | Wondoshim Old Town Street |
| B4 | Kaeson-dong Hanok Village |
| B4 | Insa-dong |
| C4 | Hapjeong Cafe Street |
| B4 | Haebangchon Village |

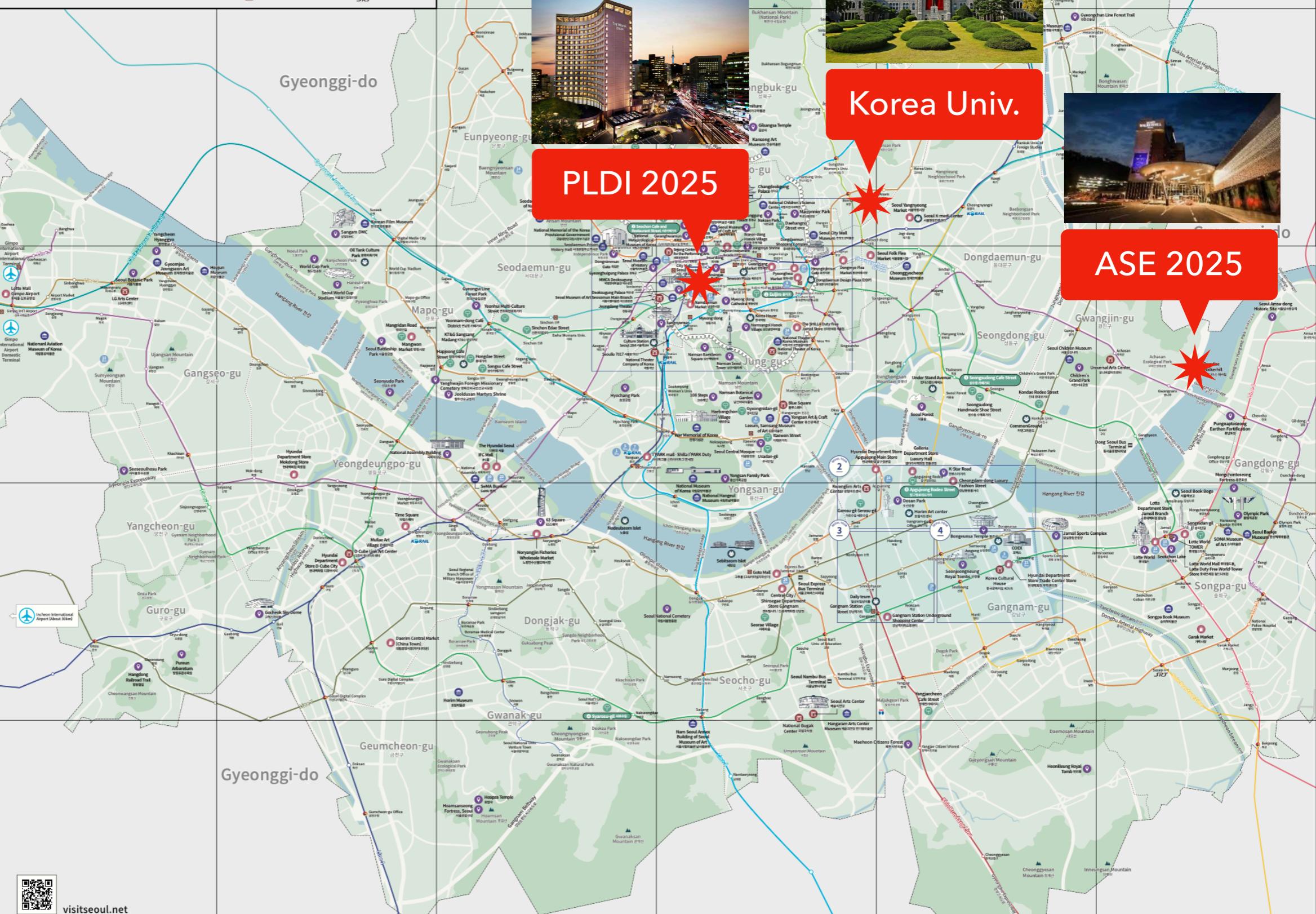
 Museum/Gallery

- | Museum & Gallery |
|---|
| C2 Hongik University
Seul Bunker |
| Kansong Art Museum |
| Gyeongju National Museum |
| National Theater of Korea |
| C3 National Meteorological Museum of Korea |
| National Memorial of the Korea Provisional Government |
| President's Cultural Center |
| National Museum of Korea |
| National Hangeul Museum |
| National Aviation Museum of Korea |
| NMCA Deoksugung |
| DMZ Seoul |
| DoDo Museum |
| Leem, Samsung Museum of Art |
| Non-Po Zoo Museum |
| Sesameum Museum of Natural History |
| Sesaddeum Prison History Hall |
| Seoul Museum of Craft Hall |
| Seoul Children Museum |
| Seoul Urban Life Museum |
| Seoul Museum of Art Sesession Main |
| Seoul Museum of Art |
| Han Seon Annes Building of Seonil Museum of Art |
| Seoul Museum of History |
| Seoul National University |
| SOHA Museum of Art |
| Songsang Book Museum |
| Hangaram Center Curator Museum |
| Yonggwan Art & Craft Center |
| Literary Museum of Yon-Ju, Dong Ju Poet |
| C4 Eungyeong History Hanok Museum |
| Wär Memorial of Korea |
| Chungscheongneung |
| Chungju National Museum |
| Korean Film Museum |
| Seoul Bakjip Museum |
| Seoul City Wall Museum |
| Aeheon Museum |

- C** 1 Grand Mercure Ambassador Hotel and Residences Seoul Yongsan
 - E** 2 Grand Walkerhill Seoul
 - E** 3 Grand InterContinental Seoul Parnas
 - E** 4 Grand Hyatt Seoul
 - C** 5 Novotel Suites Ambassador Seoul Yongsan
 - E** 6 Dongdaemun Hotel & Residences
 - C** 7 Novotel Ambassador Seoul Yongsan
 - E** 8 The Plaza Hotel Seoul
 - C** 9 Lotte Hotel Seoul
 - E** 10 Lotte Hotel World
 - C** 11 Mayfield Hotel
 - C** 12 Mondrian Seoul Itaewon
 - C** 13 Hotel Watermill Seoul
 - C** 14 Swiss Grand Hotel Seoul
 - C** 15 Signal Seoul
 - C** 16 Andaz Seoul Gangnam
 - C** 17 The Westin Chosun Seoul
 - E** 18 InterContinental Seoul COEX
 - C** 19 Jean Palace Seoul Gangnam
 - C** 20 Conrad Seoul
 - E** 21 Park Hyatt Seoul
 - C** 22 Fairmont Ambassador Seoul
 - C** 23 Four Seasons Hotel Seoul
 - C** 24 The Shilla Seoul

D4 JW Marriott Hotel Seoul

- | Performance Hall | |
|------------------|---------------------------------------|
| A3 | LG Arts Center |
| D4 | Kwanglim Arts Center |
| D4 | National Gugak Center |
| C3 | National Theater Company of Korea |
| D3 | National Theater of Korea |
| D3 | Jeongdong Theater |
| D4 | D-Cube Link Art Center |
| F3 | Blue Square |
| D4 | Sejong Center for the Performing Arts |
| D4 | Seoul Arts Center |
| F3 | Universal Arts Center |



My IFIP Talks

- Meeting 67, York Harbor (April 23-27, 2023)
- Meeting 69, Lugano (May 12-16, 2024)

Data-Driven Static Analysis

Hakjoo Oh



25 April 2023 @IFIP WG 2.4 Meeting 67, York Harbor

PL4XGL: A Programming Language Approach to Explainable Graph Learning

Hakjoo Oh
Korea University

(co-work with ~~Minseok Jeon~~ and Jihyeok Park)

IFIP WG 2.4 Meeting @Lugano, Switzerland

My IFIP Talks

- Meeting 67, York Harbor (April 23-27, 2023)
- Meeting 69, Lugano (May 12-16, 2024)

Data-Driven Static Analysis

Hakjoo Oh



25 April 2023 @IFIP WG 2.4 Meeting 67, York Harbor

PL4XGL: A Programming Language Approach to Explainable Graph Learning

Hakjoo Oh
Korea University

(co-work with ~~Minseok Jeon~~ and Jihyeok Park)

IFIP WG 2.4 Meeting @Lugano, Switzerland

Today: automated program repair (APR)

APR Research @Korea Univ.

- Automatic Diagnosis and Correction of Logical Errors for Functional Programming Assignments. OOPSLA 2018
- Context-Aware and Data-Driven Feedback Generation for Programming Assignments. ESEC/FSE 2021
- MemFix: Static Analysis-Based Repair of Memory Deallocation Errors for C. ESEC/FSE 2018
- SAVER: Scalable, Precise, and Safe Memory-Error Repair. ICSE 2020 (deployed in industry)
- NPEX: Repairing Java Null Pointer Exceptions without Tests. ICSE 2022
- PyTER: Effective Program Repair for Python Type Errors. ESEC/FSE 2022
- SmartFix: Fixing Vulnerable Smart Contracts by Accelerating Generate-and-Verify Repair using Statistical Models. ESEC/FSE 2023
- Reducing the Cost of LLM-based APR via Execution-Guided Static Analysis. In progress
- Accurate Detection of Overfitting Patches in Automated Program Repair through Semantic Anti-Patterns. In progress

APR Research @Korea Univ.

- Automatic Diagnosis and Correction of Logical Errors for Functional Programming Assignments. OOPSLA 2018
- Context-Aware and Data-Driven Feedback Generation for Programming Assignments. ESEC/FSE 2021
- MemFix: Static Analysis-Based Repair of Memory Deallocation Errors for C. ESEC/FSE 2018
- SAVER: Scalable, Precise, and Safe Memory-Error Repair. ICSE 2020 (deployed in industry)
- NPEX: Repairing Java Null Pointer Exceptions without Tests. ICSE 2022
- PyTER: Effective Program Repair for Python Type Errors. ESEC/FSE 2022
- SmartFix: Fixing Vulnerable Smart Contracts by Accelerating Generate-and-Verify Repair using Statistical Models. ESEC/FSE 2023
- Reducing the Cost of LLM-based APR via Execution-Guided Static Analysis. In progress
- Accurate Detection of Overfitting Patches in Automated Program Repair through Semantic Anti-Patterns. In progress

APR Research @Korea Univ.

- Automatic Diagnosis and Correction of Logical Errors for Functional Programming Assignments. OOPSLA 2018
- Context-Aware and Data-Driven Feedback Generation for Programming Assignments. ESEC/FSE 2021
- MemFix: Static Analysis-Based Repair of Memory Deallocation Errors for C. ESEC/FSE 2018
- SAVER: Scalable, Precise, and Safe Memory-Error Repair. ICSE 2020 (deployed in industry)
- NPEX: Repairing Java Null Pointer Exceptions without Tests. ICSE 2022
- PyTER: Effective Program Repair for Python Type Errors. ESEC/FSE 2022
- SmartFix: Fixing Vulnerable Smart Contracts by Accelerating Generate-and-Verify Repair using Statistical Models. ESEC/FSE 2023
- Reducing the Cost of LLM-based APR via Execution-Guided Static Analysis. In progress
- Accurate Detection of Overfitting Patches in Automated Program Repair through Semantic Anti-Patterns. In progress

APR Research @Korea Univ.

- Automatic Diagnosis and Correction of Logical Errors for Functional Programming Assignments. OOPSLA 2018
- Context-Aware and Data-Driven Feedback Generation for Programming Assignments. ESEC/FSE 2021
- MemFix: Static Analysis-Based Repair of Memory Deallocation Errors for C. ESEC/FSE 2018
- SAVER: Scalable, Precise, and Safe Memory-Error Repair. ICSE 2020 (deployed in industry)
- NPEX: Repairing Java Null Pointer Exceptions without Tests. ICSE 2022
- PyTER: Effective Program Repair for Python Type Errors. ESEC/FSE 2022
- SmartFix: Fixing Vulnerable Smart Contracts by Accelerating Generate-and-Verify Repair using Statistical Models. ESEC/FSE 2023
- Reducing the Cost of LLM-based APR via Execution-Guided Static Analysis. In progress
- Accurate Detection of Overfitting Patches in Automated Program Repair through Semantic Anti-Patterns. In progress

APR Research @Korea Univ.

- Automatic Diagnosis and Correction of Logical Errors for Functional Programming Assignments. OOPSLA 2018
- Context-Aware and Data-Driven Feedback Generation for Programming Assignments. ESEC/FSE 2021
- MemFix: Static Analysis-Based Repair of Memory Deallocation Errors for C. ESEC/FSE 2018
- SAVER: Scalable, Precise, and Safe Memory-Error Repair. ICSE 2020 (deployed in industry)
- NPEX: Repairing Java Null Pointer Exceptions without Tests. ICSE 2022
- PyTER: Effective Program Repair for Python Type Errors. ESEC/FSE 2022
- SmartFix: Fixing Vulnerable Smart Contracts by Accelerating Generate-and-Verify Repair using Statistical Models. ESEC/FSE 2023
- Reducing the Cost of LLM-based APR via Execution-Guided Static Analysis. In progress
- Accurate Detection of Overfitting Patches in Automated Program Repair through Semantic Anti-Patterns. In progress

APR Research @Korea Univ.

- Automatic Diagnosis and Correction of Logical Errors for Functional Programming Assignments. OOPSLA 2018
- Context-Aware and Data-Driven Feedback Generation for Programming Assignments. ESEC/FSE 2021
- MemFix: Static Analysis-Based Repair of Memory Deallocation Errors for C. ESEC/FSE 2018
- SAVER: Scalable, Precise, and Safe Memory-Error Repair. ICSE 2020 (deployed in industry)
- NPEX: Repairing Java Null Pointer Exceptions without Tests. ICSE 2022
- PyTER: Effective Program Repair for Python Type Errors. ESEC/FSE 2022
- SmartFix: Fixing Vulnerable Smart Contracts by Accelerating Generate-and-Verify Repair using Statistical Models. ESEC/FSE 2023
- Reducing the Cost of LLM-based APR via Execution-Guided Static Analysis. In progress
- Accurate Detection of Overfitting Patches in Automated Program Repair through Semantic Anti-Patterns. In progress

APR Research @Korea Univ.

- Automatic Diagnosis and Correction of Logical Errors for Functional Programming Assignments. OOPSLA 2018
- Context-Aware and Data-Driven Feedback Generation for Programming Assignments. ESEC/FSE 2021
- MemFix: Static Analysis-Based Repair of Memory Deallocation Errors for C. ESEC/FSE 2018
- SAVER: Scalable, Precise, and Safe Memory-Error Repair. ICSE 2020 (deployed in industry)
- NPEX: Repairing Java Null Pointer Exceptions without Tests. ICSE 2022
- PyTER: Effective Program Repair for Python Type Errors. ESEC/FSE 2022
- SmartFix: Fixing Vulnerable Smart Contracts by Accelerating Generate-and-Verify Repair using Statistical Models. ESEC/FSE 2023
- Reducing the Cost of LLM-based APR via Execution-Guided Static Analysis. In progress
- Accurate Detection of Overfitting Patches in Automated Program Repair through Semantic Anti-Patterns. In progress

Our approach: Static analysis-based program repair

APR Research @Korea Univ.

- Automatic Diagnosis and Correction of Logical Errors for Functional Programming Assignments. OOPSLA 2018
- Context-Aware and Data-Driven Feedback Generation for Programming Assignments. ESEC/FSE 2021
- MemFix: Static Analysis-Based Repair of Memory Deallocation Errors for C. ESEC/FSE 2018
- SAVER: Scalable, Precise, and Safe Memory-Error Repair. ICSE 2020 (deployed in industry)
- NPEX: Repairing Java Null Pointer Exceptions without Tests. ICSE 2022
- PyTER: Effective Program Repair for Python Type Errors. ESEC/FSE 2022
- SmartFix: Fixing Vulnerable Smart Contracts by Accelerating Generate-and-Verify Repair using Statistical Models. ESEC/FSE 2023
- Reducing the Cost of LLM-based APR via Execution-Guided Static Analysis. In progress
- Accurate Detection of Overfitting Patches in Automated Program Repair through Semantic Anti-Patterns. In progress

Our approach: Static analysis-based program repair

Memory Errors in C Programs

- Memory-leak (**ML**), use-after-free (**UAF**), and double-free (**DF**) are prevalent in real-world C programs

Repository	#commits	ML	DF	UAF	Total	*-overflow
linux	721,119	3,740	821	1,986	6,363	5,092
openssl	21,009	220	36	12	264	61
numpy	17,008	58	2	2	59	53
php	105,613	1,129	148	197	1,449	649
git	49,475	350	19	95	442	258

Memory Errors in C Programs

- Memory-leak (**ML**), use-after-free (**UAF**), and double-free (**DF**) are prevalent in real-world C programs

Repository	#commits	ML	DF	UAF	Total	*-overflow
linux	721,119	3,740	821	1,986	6,363	5,092
openssl	21,009	220	36	12	264	61
numpy	17,008	58	2	2	59	53
php	105,613	1,129	148	197	1,449	649
git	49,475	350	19	95	442	258

- Significant sources of security vulnerabilities

CVE-2017-9798 Optionsbleed - Apache memory leak
Alexandr Tumanov Updated 2 months ago

Linux kernel: CVE-2017-6074: DCCP double-free vulnerability (local root)

From: Andrey Konovalov <andreyknvl () google com>
Date: Wed, 22 Feb 2017 14:28:35 +0100

Hi,

This is an announcement about CVE-2017-6074 [1] which is a double-free vulnerability I found in the Linux kernel. It can be exploited to gain kernel code execution from an unprivileged processes.

Vulnerability Details : [CVE-2017-11274](#)

Adobe Digital Editions 4.5.4 and earlier has an exploitable use after free vulnerability.
Publish Date : 2017-08-11 Last Update Date : 2017-08-16

[Collapse All](#) [Expand All](#) [Select](#) [Select&Copy](#) [Search Twitter](#) [Search YouTube](#) [Search Google](#) [▼ Scroll To](#) [▼ Comments](#) [▼ External](#)

- CVSS Scores & Vulnerability Types

CVSS Score **10.0**

Goal

- Long-term goal: Fully automated detection and repair
- This talk: SAVER, a system to automatically fix memory errors



- To be practical, SAVER is designed to be
 - **Scalable:** Capable of handling large, industry programs
 - **Precise:** Effectively fixes diverse bugs with high fix rates
 - **Safe:** Generated patches are safe, not introducing new errors

(1) Double-Free in Linux Kernel

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

    goto err;
}

out = malloc(2);
if (out == NULL) {
    free(in);

    goto err;
}
... // use in, out
err:
    free(in);
    free(out);
    return;
```

(1) Double-Free in Linux Kernel

```
in = malloc(1);
out = malloc(1); ← Allocated
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {

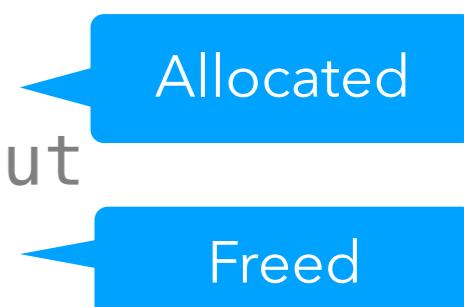
    goto err;
}

out = malloc(2);
if (out == NULL) {
    free(in);

    goto err;
}
... // use in, out
err:
    free(in);
    free(out);
    return;
```

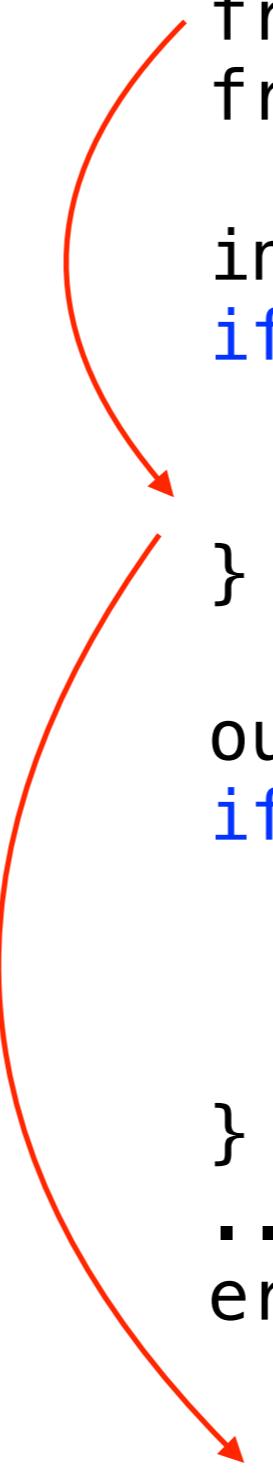
(1) Double-Free in Linux Kernel

```
in = malloc(1);  
out = malloc(1);  
... // use in, out  
free(out);  
free(in);  
  
in = malloc(2);  
if (in == NULL) {  
    goto err;  
}  
  
out = malloc(2);  
if (out == NULL) {  
    free(in);  
  
    goto err;  
}  
... // use in, out  
err:  
    free(in);  
    free(out);  
    return;
```



(1) Double-Free in Linux Kernel

```
in = malloc(1);  
out = malloc(1);  
... // use in, out  
free(out);  
free(in);  
  
in = malloc(2);  
if (in == NULL) {  
  
    goto err;  
}  
  
out = malloc(2);  
if (out == NULL) {  
    free(in);  
  
    goto err;  
}  
... // use in, out  
err:  
free(in);  
free(out);  
return;
```



Allocated

Freed

Double Free

(1) Double-Free in Linux Kernel

```
in = malloc(1);  
out = malloc(1);  
... // use in, out  
free(out);  
free(in);  
  
in = malloc(2);  
if (in == NULL) {  
  
    goto err;  
}  
  
out = malloc(2);  
if (out == NULL) {  
    free(in);  
  
    goto err;  
}  
... // use in, out  
err:  
free(in);  
free(out);  
return;
```

The diagram illustrates a sequence of memory operations. It starts with two allocations: `in = malloc(1);` and `out = malloc(1);`. Both pointers are used for some operations. Then, `out` is freed: `free(out);`. Next, another allocation is made: `in = malloc(2);`. If this fails, control goes to the `err` label. If successful, `out` is allocated again: `out = malloc(2);`. If this fails, `in` is freed: `free(in);` and control goes to `err`. Finally, both pointers are used again. At the `err` label, both `in` and `out` are freed: `free(in);` and `free(out);`, and the function returns.

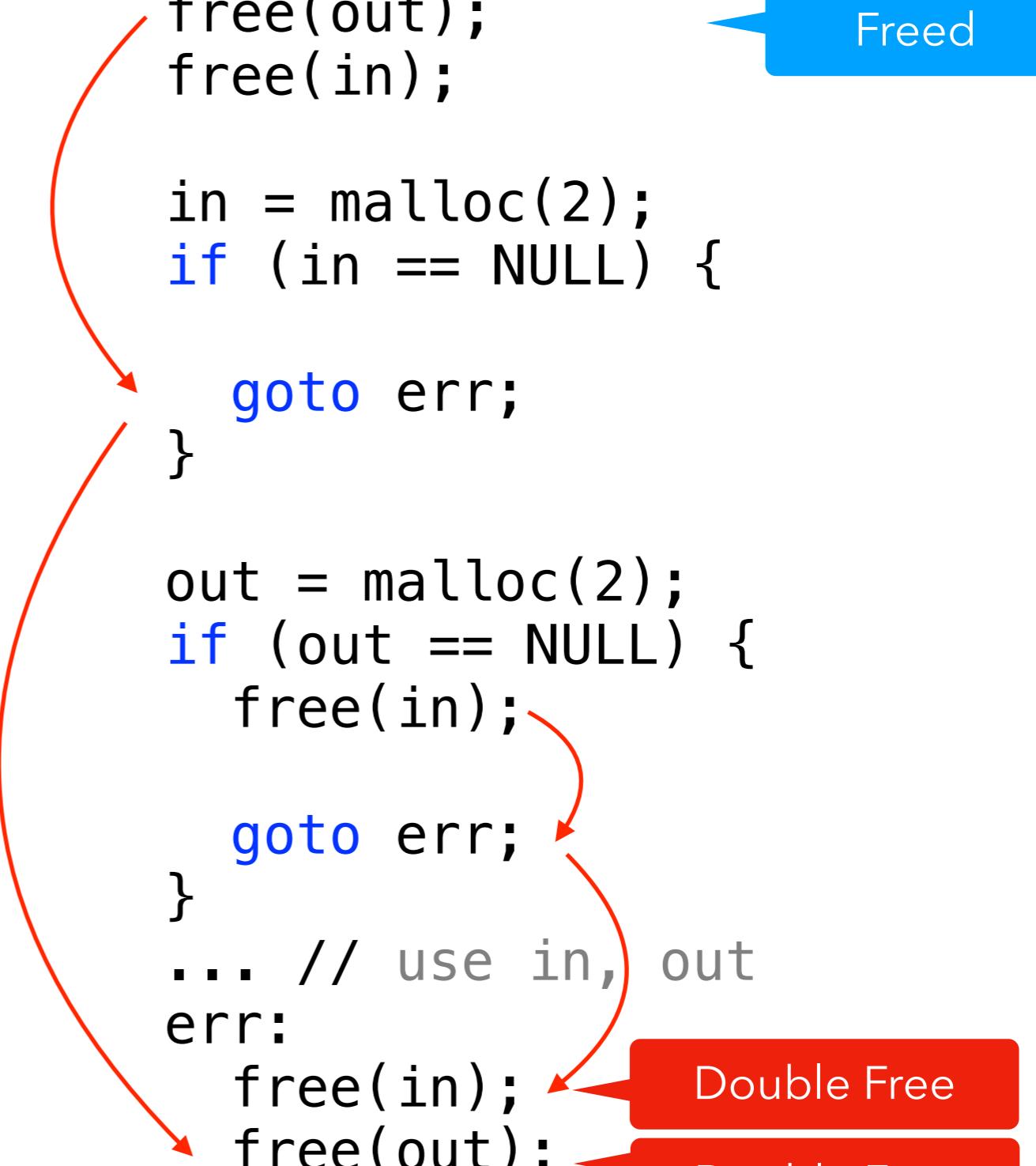
Allocated

Freed

Double Free

(1) Double-Free in Linux Kernel

```
in = malloc(1);  
out = malloc(1);  
... // use in, out  
free(out);  
free(in);  
  
in = malloc(2);  
if (in == NULL) {  
  
    goto err;  
}  
  
out = malloc(2);  
if (out == NULL) {  
    free(in);  
  
    goto err;  
}  
... // use in, out  
err:  
free(in);  
free(out);  
return;
```



Allocated

Freed

Double Free

Double Free

(1) Double-Free in Linux Kernel

USB: fix double frees in error code paths of ipaq driver

the error code paths can be enter with buffers to freed buffers.
Serial core would do a kfree() on memory already freed.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>

master → v4.15-rc1 ... v2.6.24-rc1

 Oliver Neukum committed with gregkh on 18 Sep 2007

1 par

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);
```

```
in = malloc(2);
if (in == NULL) {
    out = NULL;
    goto err;
}
```

```
out = malloc(2);
if (out == NULL) {
    free(in);
    in = NULL;
    goto err;
}
... // use in, out
err:
    free(in);
    free(out);
    return;
```

(1) Double-Free in Linux Kernel

USB: fix double frees in error code paths of ipaq driver

the error code paths can be enter with buffers to freed buffers.
Serial core would do a kfree() on memory already freed.

Signed-off-by: Oliver Neukum <oneukum@suse.de>

Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>

master → v4.15-rc1 ... v2.6.24-rc1

 Oliver Neukum committed with gregkh on 18 Sep 2007

1 par

Challenge 1: Difficult to ensure that
bugs are fixed correctly

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);
```

```
in = malloc(2);
if (in == NULL) {
    out = NULL;
    goto err;
}
```

```
out = malloc(2);
if (out == NULL) {
    free(in);
    in = NULL;
    goto err;
}
```

```
... // use in, out
err:
    free(in);
    free(out);
    return;
```

(1) Double-Free in Linux Kernel

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);
```

```
in = malloc(2);
if (in == NULL) {
    out = NULL;
    goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
    free(in);
    in = NULL;
    goto err;
}
... // use in, out
err:
    free(in);
    free(out);
    return;
```

USB: fix double kfree in ipaq in error case

in the error case the ipaq driver leaves a dangling pointer to already freed memory that will be freed again.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>

by master · v4.15-rc1 · ... · v2.6.27-rc1

 Oliver Neukum committed with gregkh on 30 Jun 2008

1 parent 35

Second attempt (9 months later)

(1) Double-Free in Linux Kernel

memory leak

Challenge 2: Patches may introduce
new errors

USB: fix double kfree in ipaq in error case

in the error case the ipaq driver leaves a dangling pointer to already freed memory that will be freed again.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>

master v4.15-rc1 ... v2.6.27-rc1

Oliver Neukum committed with gregkh on 30 Jun 2008

1 parent 35

Second attempt (9 months later)

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);

in = malloc(2);
if (in == NULL) {
    out = NULL;
    goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
    free(in);
    in = NULL;
    goto err;
}
... // use in, out
err:
    free(in);
    free(out);
return;
```

(1) Double-Free in Linux Kernel

fix for a memory leak in an error case introduced by fix for double free

The fix NULLed a pointer without freeing it.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Reported-by: Juha Motorsportcom <juha_motorsportcom@luukku.com>
Signed-off-by: Linus Torvalds <torvalds@linux-foundation.org>

master v4.15-rc1 ... v2.6.27-rc1



Oliver Neukum committed with torvalds on 27 Jul 2008

1 parent 9ee08c2

Third attempt
(10 months after bug detection)

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);
out = NULL;
in = malloc(2);
if (in == NULL) {
    out = NULL;
    goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
    free(in);
    in = NULL;
    goto err;
}
... // use in, out
err:
    free(in);
    free(out);
    return;
```

(1) Double-Free in Linux Kernel

Challenge 3: The resulting patches
are often not of high quality

fix for a memory leak in an error case introduced by fix for double free

The fix NULLed a pointer without freeing it.

Signed-off-by: Oliver Neukum <oneukum@suse.de>
Reported-by: Juha Motorsportcom <juha_motorsportcom@luukku.com>
Signed-off-by: Linus Torvalds <torvalds@linux-foundation.org>

master v4.15-rc1 ... v2.6.27-rc1



Oliver Neukum committed with torvalds on 27 Jul 2008

1 parent 9ee08c2

Third attempt
(10 months after bug detection)

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);
out = NULL;
in = malloc(2);
if (in == NULL) {
    out = NULL;
    goto err;
}
free(out);
out = malloc(2);
if (out == NULL) {
    free(in);
    in = NULL;
    goto err;
}
... // use in, out
err:
    free(in);
    free(out);
    return;
```

SAVER-Generated Patch

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);
```

```
in = malloc(2);
if (in == NULL) {
    goto err;
}
```

```
out = malloc(2);
if (out == NULL) {
    free(in);
```

```
    goto err;
}
... // use in, out
err:
```

```
    free(in); // double-free
    free(out); // double-free
    return;
```



SAVER

- ✓ Fast (few mins)
- ✓ Safety guarantee

```
in = malloc(1);
out = malloc(1);
... // use in, out
free(out);
free(in);
```

```
in = malloc(2);
if (in == NULL) {
```

```
    goto err;
}
```

```
free(out);
out = malloc(2);
if (out == NULL) {
    free(in);
```

```
    goto err;
}
```

```
... // use in, out
err:
```

```
    free(in);
    free(out);
    return;
```

(2) Memory Leak in Snort

```
1 int append_data (Node *node, int *ndata) {
2     if (!(Node *n = malloc(sizeof(Node))))
3         return -1; // failed to be appended
4     n->data = ndata;
5     n->next = node->next; node->next = n;
6     return 0; // successfully appended
7 }
8
9 Node *lx = ... // a linked list
10 Node *ly = ... // a linked list
11 for (Node *node = lx; node != NULL; node = node->next) {
12     int *dptr = malloc(sizeof(int));
13     if (!dptr) return;
14     *dptr = *(node->data);
15     (-) append_data(ly, dptr); // potential memory-leak
16
17 }
```



Memory Leak:

An object allocated at line 12
becomes unreachable after line 15

(2) Memory Leak in Snort

```
1 int append_data (Node *node, int *ndata) {
2     if (!(Node *n = malloc(sizeof(Node))))
3         return -1; // failed to be appended
4     n->data = ndata; // Potential memory leak
5     n->next = node->next; node->next = n;
6     return 0; // successfully appended
7 }
8
9 Node *lx = ... // a linked list
10 Node *ly = ... // a linked list
11 for (Node *node = lx; node != NULL, node = node->next) {
12     int *dptr = malloc(sizeof(int));
13     if (!dptr) return;
14     *dptr = *(node->data);
15     (-) append_data(ly, dptr); // potential memory-leak
16
17 }
```

Normal execution

Memory Leak:

An object allocated at line 12 becomes unreachable after line 15



(2) Memory Leak in Snort

```
1 int append_data (Node *node, int *ndata) {
2     if (!(Node *n = malloc(sizeof(Node))))
3         return -1; // failed to be appended
4     n->data = ndata;
5     n->next = node->next; node->next = n;
6     return 0; // successfully appended
7 }
8
9 Node *lx = ... // a linked list
10 Node *ly = ... // a linked list
11 for (Node *node = lx; node != NULL; node = node->next) {
12     int *dptr = malloc(sizeof(int));
13     if (!dptr) return;
14     *dptr = *(node->data);
15     (-) append_data(ly, dptr); // potential memory-leak
16
17 }
```

Buggy execution



Memory Leak:

An object allocated at line 12
becomes unreachable after line 15

(2) Memory Leak in Snort

```
1 int append_data (Node *node, int *ndata) {
2     if (!(Node *n = malloc(sizeof(Node))))
3         return -1; // failed to be appended
4     n->data = ndata;
5     n->next = node->next; node->next = n;
6     return 0; // successfully appended
7 }
8
9 Node *lx = ... // a linked list
10 Node *ly = ... // a linked list
11 for (Node *node = lx; node != NULL; node = node->next) {
12     int *dptr = malloc(sizeof(int));
13     if (!dptr) return;
14     *dptr = *(node->data);
15     (-) append_data(ly, dptr); // potential memory-leak
16     (+) if ((append_data(ly, dptr)) == -1) free(dptr);
17 }
```



Memory Leak:

An object allocated at line 12
becomes unreachable after line 15

cf) SAVER vs. Generative AI

- LLMs do not guarantee safety, e.g., GPT4-generated patch:

```
1 int append_data (Node *node, int *ndata) {  
2     if (!(Node *n = malloc(sizeof(Node)))  
3         return -1; // failed to be appended  
4     n->data = ndata;  
5     n->next = node->next; node->next = n;  
6     return 0; // successfully appended  
7 }  
8  
9 Node *lx = ... // a linked list  
10 Node *ly = ... // a linked list  
11 for (Node *node = lx; node != NULL; node = node->next) {  
12     int *dptr = malloc(sizeof(int));  
13     if (!dptr) return;  
14     *dptr = *(node->data);  
15     (-) append_data(ly, dptr); // potential memory-leak  
16     (+) append_data(ly, dptr); free(dptr);  
17 }
```

UAF or DF introduced

cf) SAVER vs. Generative AI

- LLMs do not guarantee safety, e.g., GPT4-generated patch:

```
1 int append_data (Node *node, int *ndata) {  
2     if (!(Node *n = malloc(sizeof(Node)))  
3         return -1; // failed to be appended  
4     n->data = ndata;  
5     n->next = node->next; node->next = n;  
6     return 0; // successfully appended  
7 }  
8  
9 Node *lx = ... // a linked list  
10 Node *ly = ... // a linked list  
11 for (Node *node = lx; node != NULL; node = node->next) {  
12     int *dptr = malloc(sizeof(int));  
13     if (!dptr) return;  
14     *dptr = *(node->data);  
15     (-) append_data(ly, dptr); // potential memory-leak  
16     (+) append_data(ly, dptr); free(dptr);  
17 }
```

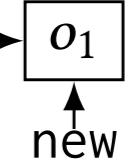
UAF or DF introduced

(3) Use-After-Free in Binutils

```
1 struct node *cleanup; // list of objects to be deallocated
2 struct node *first = NULL;
3 for (...) {
4     struct node *new = xmalloc(sizeof(*new));
5     make_cleanup(new); // add new to the cleanup list
6     new->name = ...;
7     ...
8     if (...) {
9         first = new;
10
11         continue;
12     }
13     /* potential use-after-free: `first->name` */
14     (-) if (first == NULL || new->name != first->name)
15
16         continue;
17     do_cleanups(); // deallocate all objects in cleanup
18 }
```

(3) Use-After-Free in Binutils

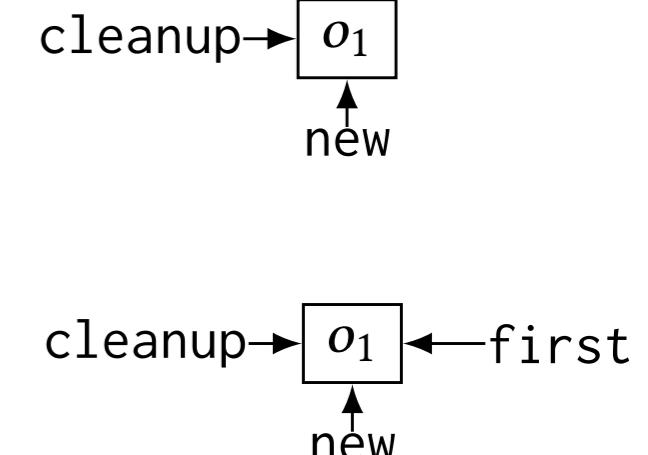
```
1 struct node *cleanup; // list of objects to be deallocated    cleanup→o1
2 struct node *first = NULL;
3 for (...) {
4     struct node *new = xmalloc(sizeof(*new));
5     make_cleanup(new); // add new to the cleanup list
6     new->name = ...;
7     ...
8     if (...) {
9         first = new;
10
11         continue;
12     }
13     /* potential use-after-free: `first->name` */
14     (-) if (first == NULL || new->name != first->name)
15
16         continue;
17     do_cleanups(); // deallocate all objects in cleanup
18 }
```



The diagram shows a pointer named 'cleanup' with an arrow pointing to a box labeled 'o₁'. An upward arrow from below points to the box, labeled 'new', indicating that the memory for object 'o₁' was dynamically allocated after the 'cleanup' pointer was set.

(3) Use-After-Free in Binutils

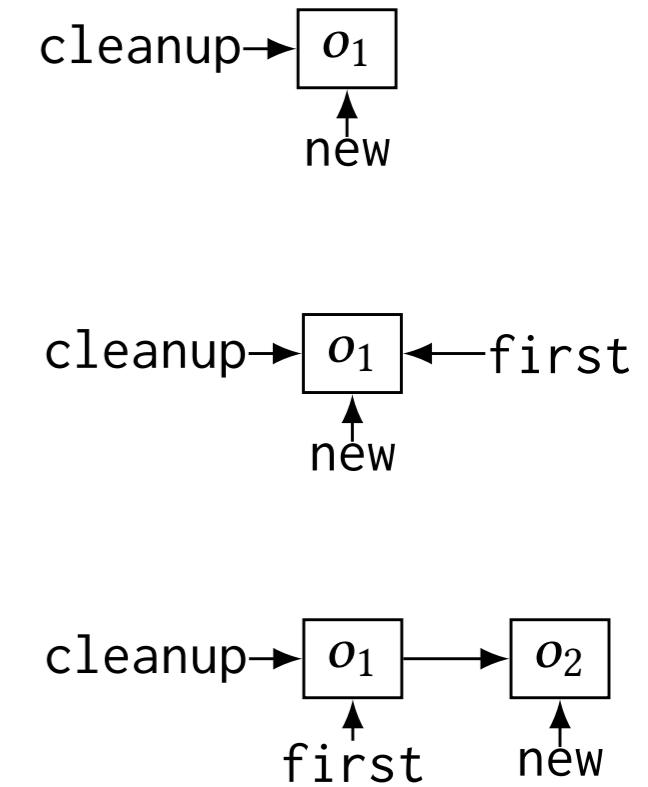
```
1 struct node *cleanup; // list of objects to be deallocated    cleanup→o1
2 struct node *first = NULL;
3 for (...) {
4     struct node *new = xmalloc(sizeof(*new));
5     make_cleanup(new); // add new to the cleanup list
6     new->name = ...;
7     ...
8     if (...) {
9         first = new;
10
11     continue;
12 }
13 /* potential use-after-free: `first->name` */
14 (-) if (first == NULL || new->name != first->name)
15
16     continue;
17 do_cleanups(); // deallocate all objects in cleanup
18 }
```



The diagram illustrates the state of pointers after line 9. It shows three nodes: o_1 , o_2 , and o_3 . Node o_1 is at the head of a list. A pointer $cleanup \rightarrow o_1$ and a pointer $new \rightarrow o_1$ both point to o_1 . Node o_2 is the previous node in the list, with o_3 being the next. A pointer $first \leftarrow o_1$ points to o_1 .

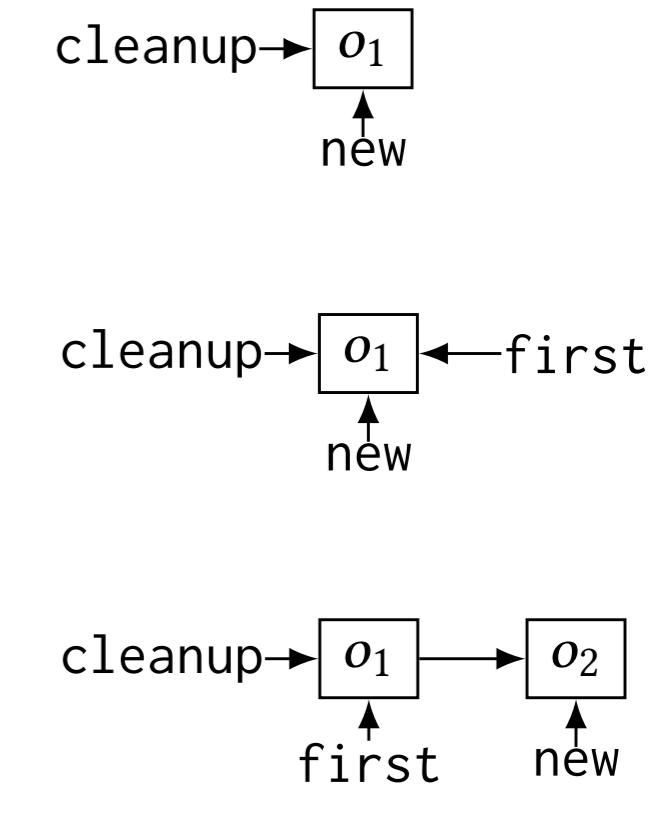
(3) Use-After-Free in Binutils

```
1 struct node *cleanup; // list of objects to be deallocated
2 struct node *first = NULL;
3 for (...) {
4     struct node *new = xmalloc(sizeof(*new));
5     make_cleanup(new); // add new to the cleanup list
6     new->name = ...;
7     ...
8     if (...) {
9         first = new;
10        continue;
11    }
12    /* potential use-after-free: `first->name` */
13    if (first == NULL || new->name != first->name)
14        continue;
15    do_cleanups(); // deallocate all objects in cleanup
16 }
17 }
```



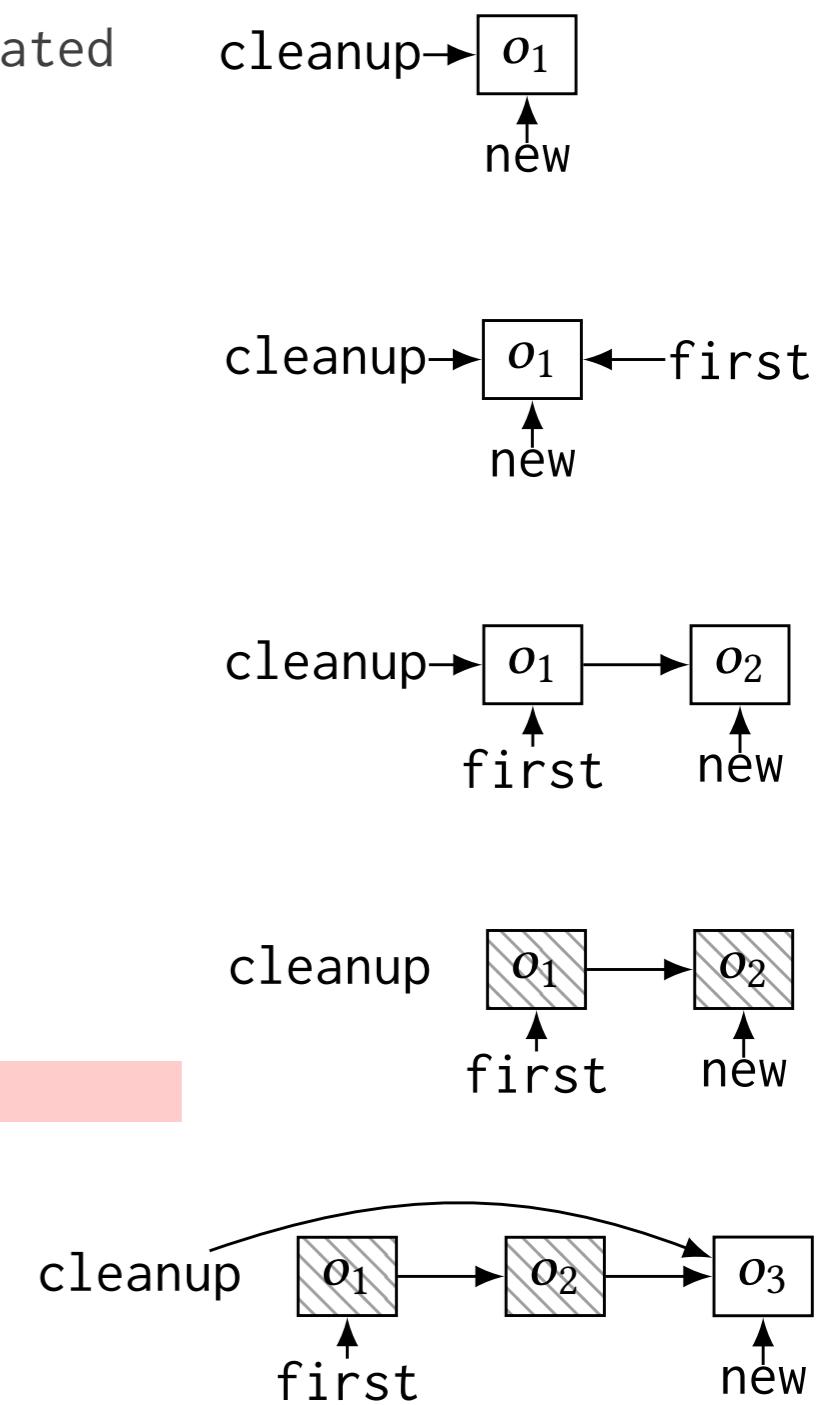
(3) Use-After-Free in Binutils

```
1 struct node *cleanup; // list of objects to be deallocated
2 struct node *first = NULL;
3 for (...) {
4     struct node *new = xmalloc(sizeof(*new));
5     make_cleanup(new); // add new to the cleanup list
6     new->name = ...;
7     ...
8     if (...) {
9         first = new;
10    continue;
11 }
12 /* potential use-after-free: `first->name` */
13 (-) if (first == NULL || new->name != first->name)
14
15
16     continue;
17 do_cleanups(); // deallocate all objects in cleanup
18 }
```



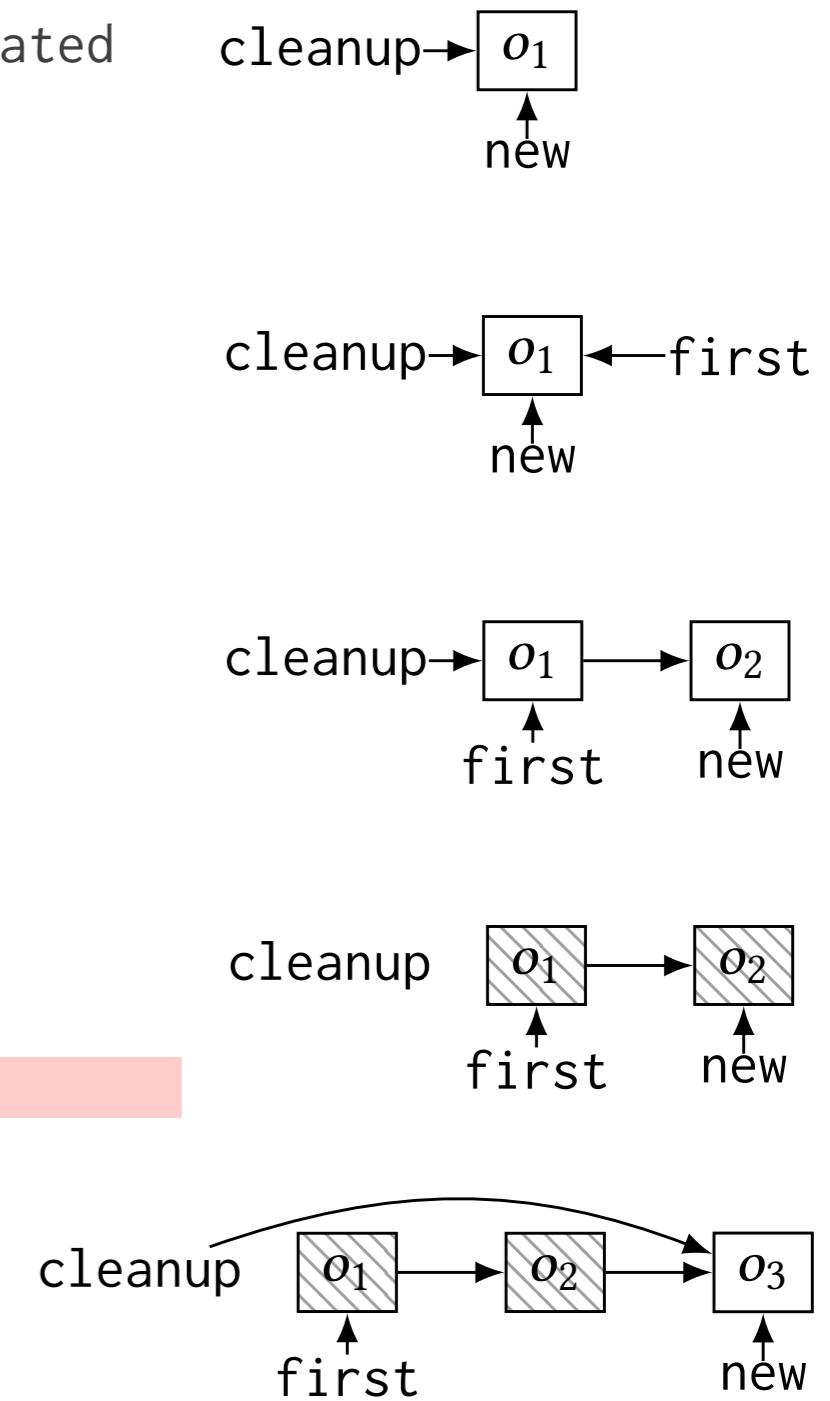
(3) Use-After-Free in Binutils

```
1 struct node *cleanup; // list of objects to be deallocated
2 struct node *first = NULL;
3 for (...) {
4     struct node *new = xmalloc(sizeof(*new));
5     make_cleanup(new); // add new to the cleanup list
6     new->name = ...;
7     ...
8     if (...) {
9         first = new;
10        continue;
11    }
12    /* potential use-after-free: `first->name` */
13    (-) if (first == NULL || new->name != first->name)
14        continue;
15    do_cleanups(); // deallocate all objects in cleanup
16 }
17
18 }
```



(3) Use-After-Free in Binutils

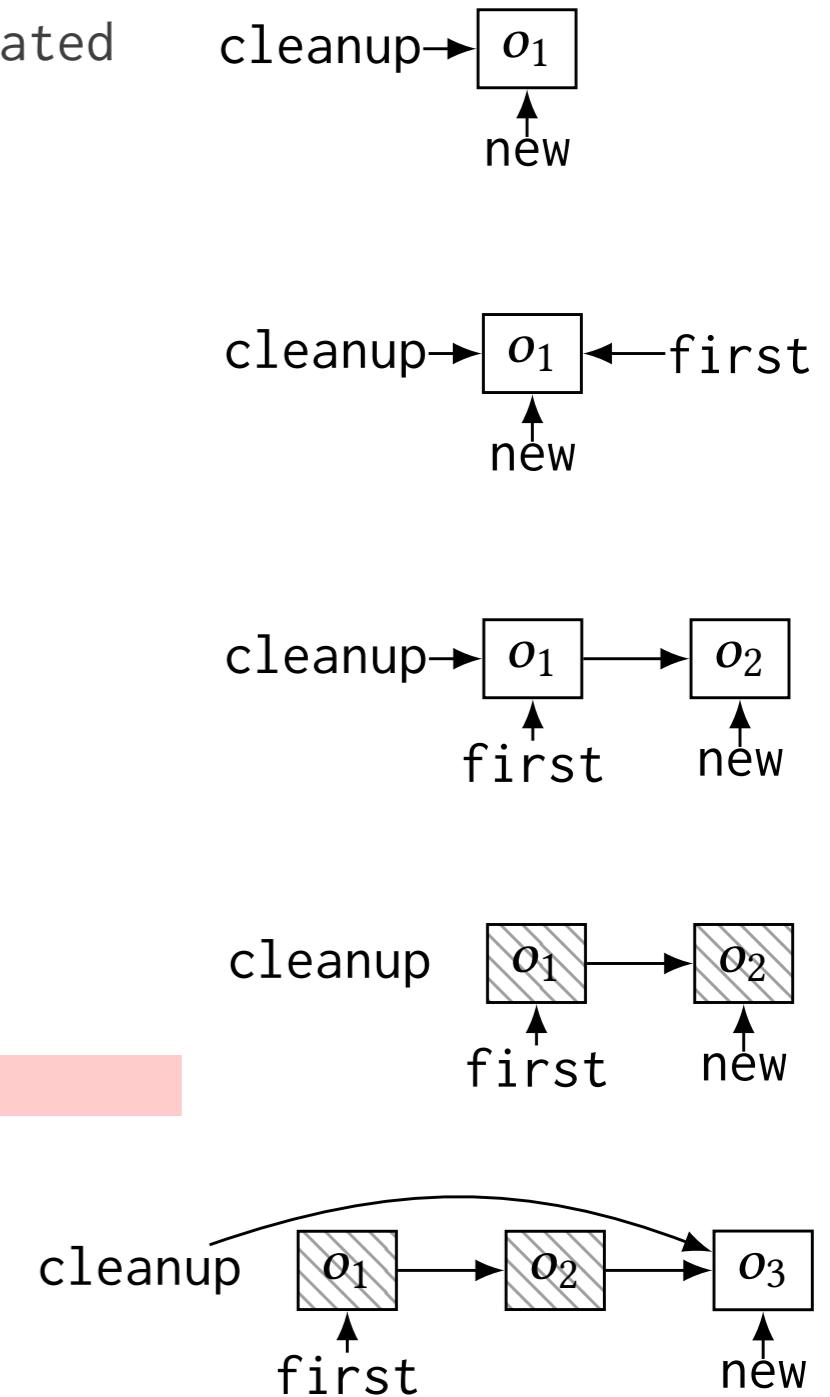
```
1 struct node *cleanup; // list of objects to be deallocated
2 struct node *first = NULL;
3 for (...) {
4     struct node *new = xmalloc(sizeof(*new));
5     make_cleanup(new); // add new to the cleanup list
6     new->name = ...;
7     ...
8     if (...) {
9         first = new;
10    continue;
11 }
12 /* potential use-after-free: `first->name` */
13 (-) if (first == NULL || new->name != first->name)
14     continue;
15 do_cleanups(); // deallocate all objects in cleanup
16 }
17 }
```



(3) Use-After-Free in Binutils

```
1 struct node *cleanup; // list of objects to be deallocated
2 struct node *first = NULL;
3 for (...) {
4     struct node *new = xmalloc(sizeof(*new));
5     make_cleanup(new); // add new to the cleanup list
6     new->name = ...;
7     ...
8     if (...) {
9         first = new;
10    continue;
11 }
12 /* potential use-after-free: `first->name` */
13 (-) if (first == NULL || new->name != first->name)
14     continue;
15 do_cleanups(); // deallocate all objects in cleanup
16 }
17
18 }
```

use-after-free



SAVER-Generated Patch

```
1 struct node *cleanup; // list of objects to be deallocated
2 struct node *first = NULL;
3 for (...) {
4     struct node *new = xmalloc(sizeof(*new));
5     make_cleanup(new); // add new to the cleanup list
6     new->name = ...;
7     ...
8     if (...) {
9         first = new;
10    (+)   tmp = first->name;
11        continue;
12    }
13    /* potential use-after-free: `first->name` */
14    (-)   if (first == NULL || new->name != first->name)
15    (+)   if (first == NULL || new->name != tmp)
16        continue;
17    do_cleanups(); // deallocate all objects in cleanup
18 }
```

How SAVER Works

```
1 p = malloc(1); //o1
2 if (C)
3     q = p;
4 else
5     q = malloc(1); //o2
6     *p = 1;
7 free(q);
```

Memory leak: o_1 is not freed when the false branch is taken

How SAVER Works

```
1 p = malloc(1); //o1
2 if (C)
3     q = p;
4 else
5     q = malloc(1); //o2
6     *p = 1;
7     free(q);
```

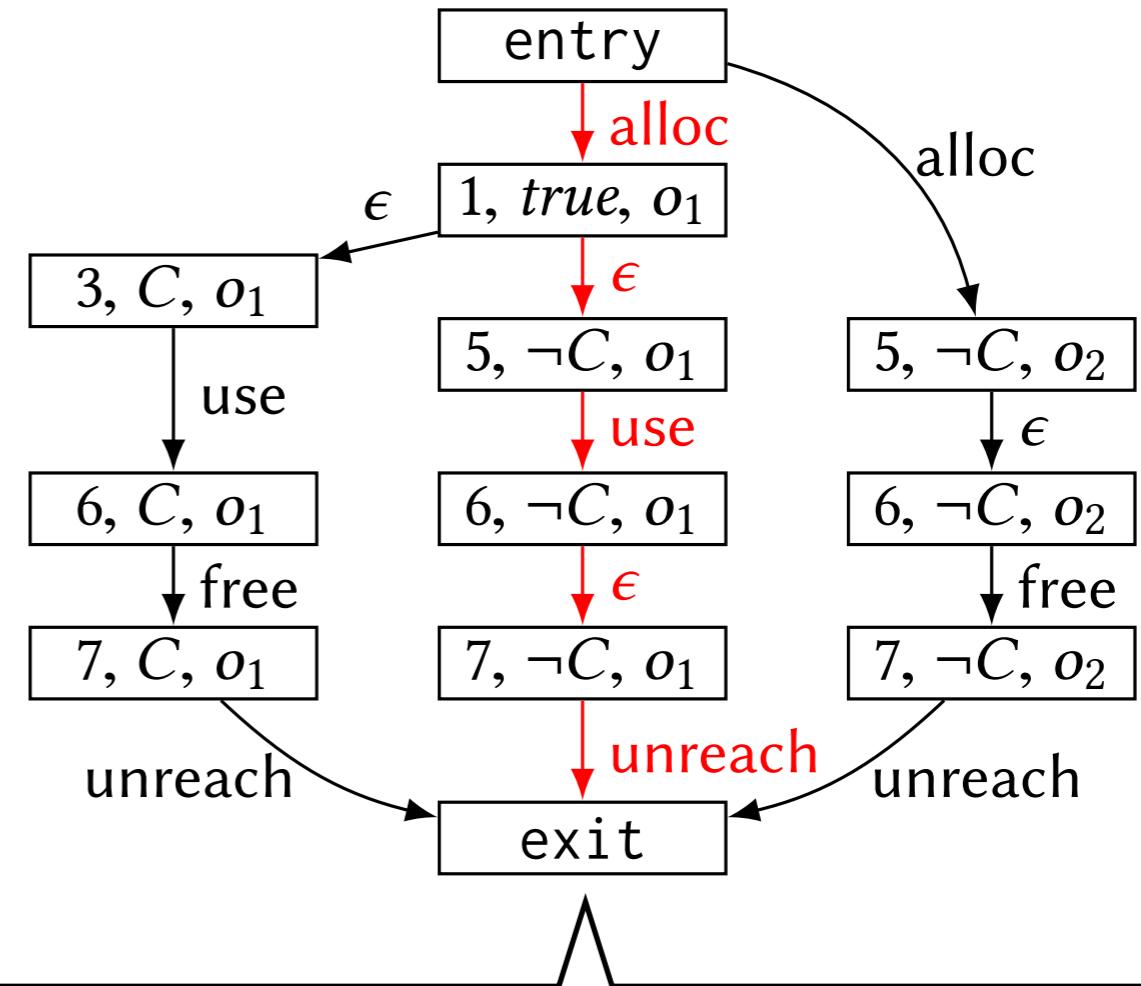
if (!C) free(p);

Memory leak: o_1 is not freed when the false branch is taken

How SAVER Works

1. Run a static analysis to generate *object flow graph*

```
1  p = malloc(1); //o1
2  if (C)
3    q = p;
4  else
5    q = malloc(1); //o2
6  *p = 1;
7  free(q);
```

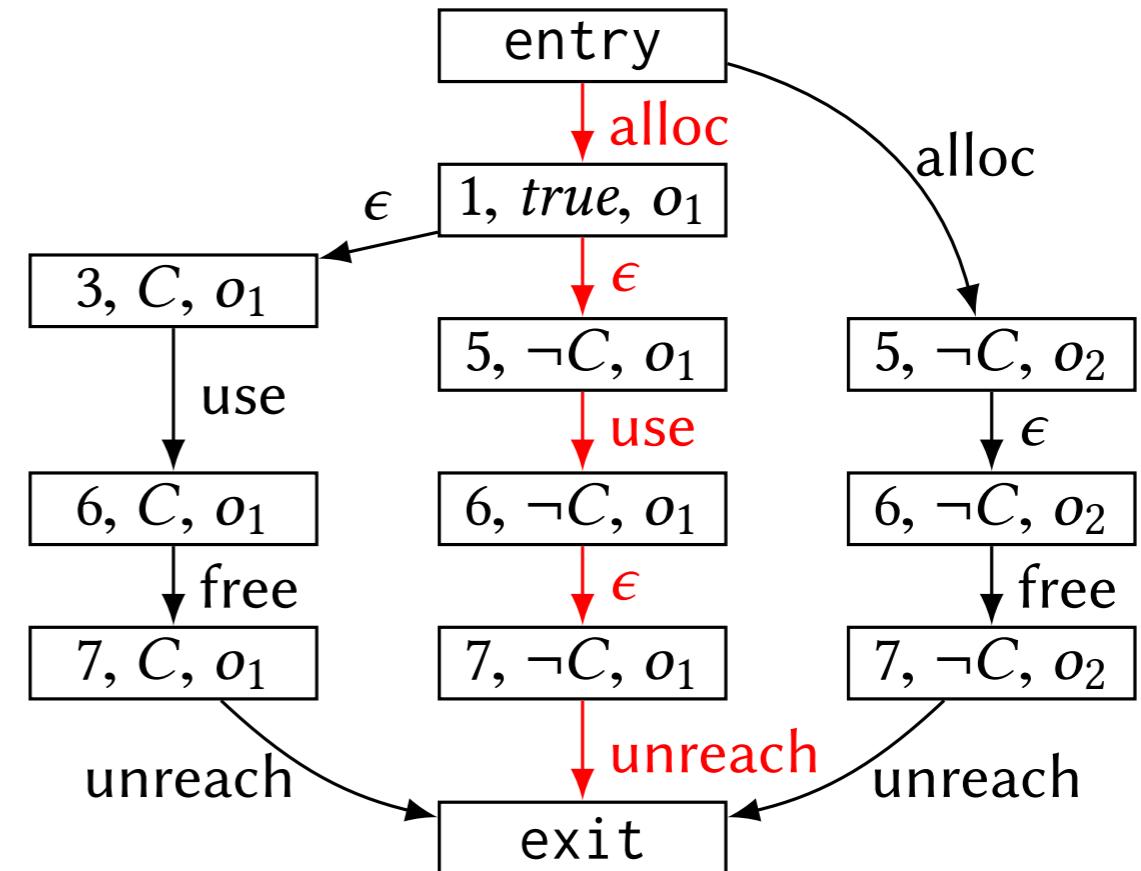


- Vertex: (program point, path condition, available heap object)
- Edge: control flow labeled w/ events that could occur for objects

How SAVER Works

2. Relabel object flow graph to eliminate buggy paths

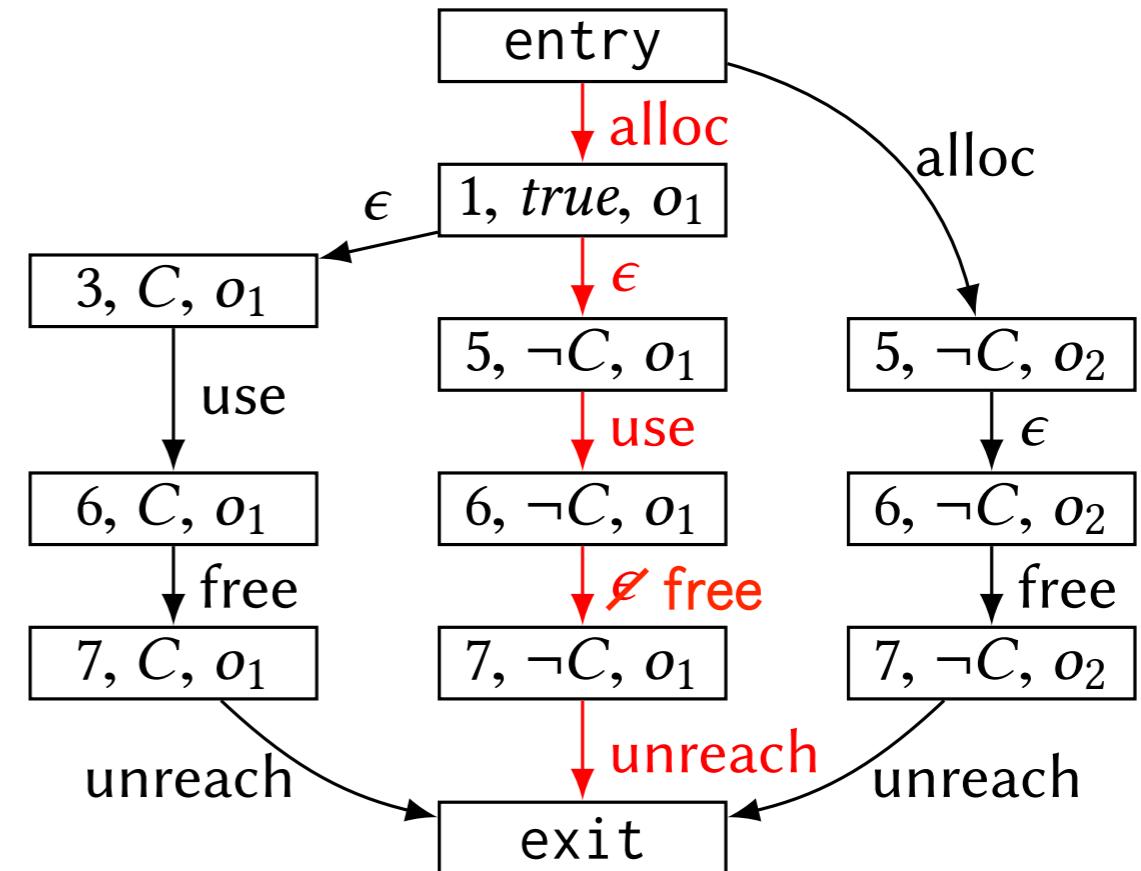
```
1  p = malloc(1); //o1
2  if (C)
3    q = p;
4  else
5    q = malloc(1); //o2
6  *p = 1;
7  free(q);
```



How SAVER Works

2. Relabel object flow graph to eliminate buggy paths

```
1  p = malloc(1); //o1
2  if (C)
3    q = p;
4  else
5    q = malloc(1); //o2
6  *p = 1;
7  free(q);
```



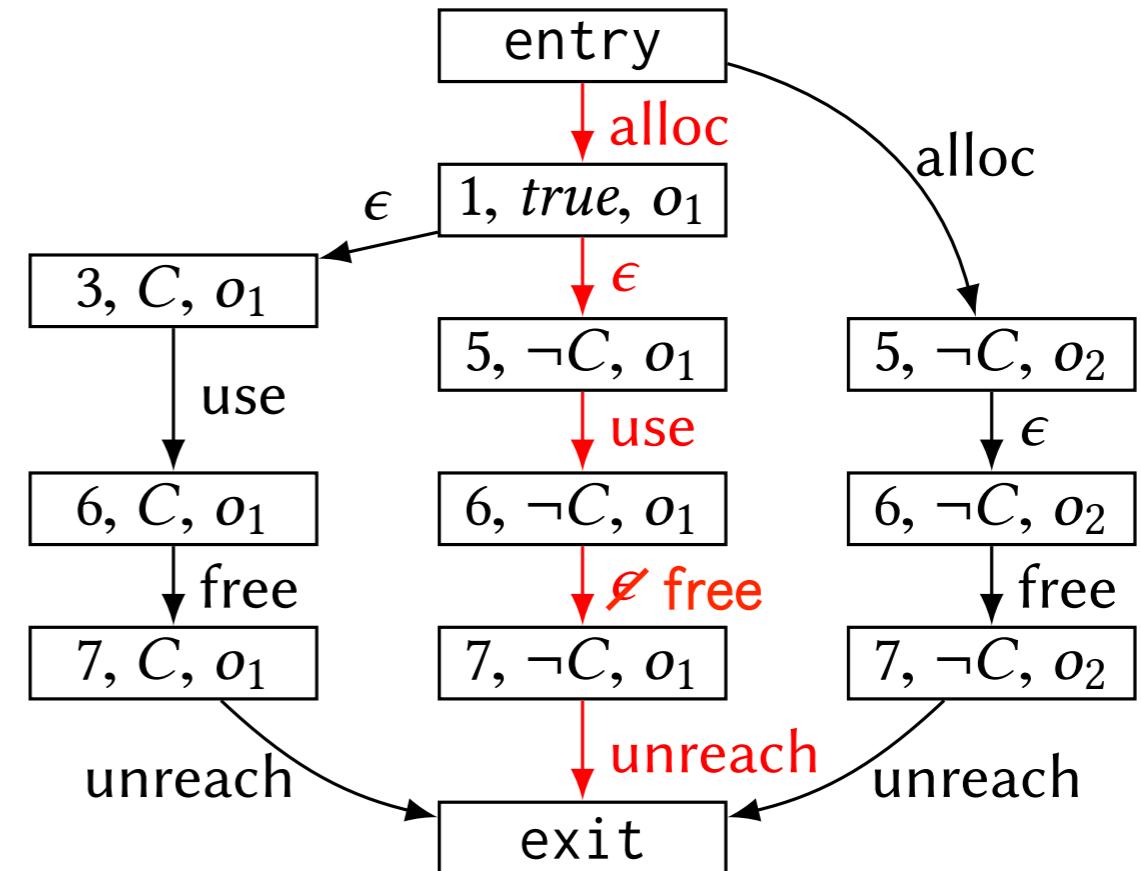
How SAVER Works

2. Relabel object flow graph to eliminate buggy paths

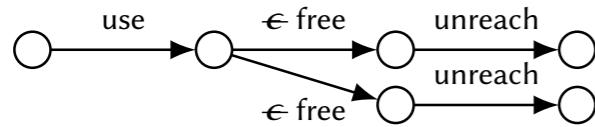
```

1   p = malloc(1); //o1
2   if (C)
3     q = p;
4   else
5     q = malloc(1); //o2
6   *p = 1;
7   free(q);

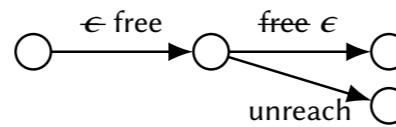
```



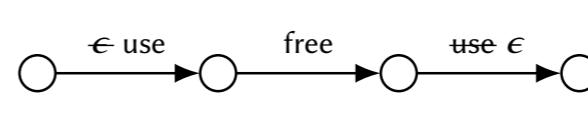
- SAVER supports four types of re-labeling strategies:



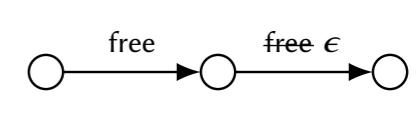
(a) Inserting free



(b) Relocating free



(c) Relocating use (dereference)



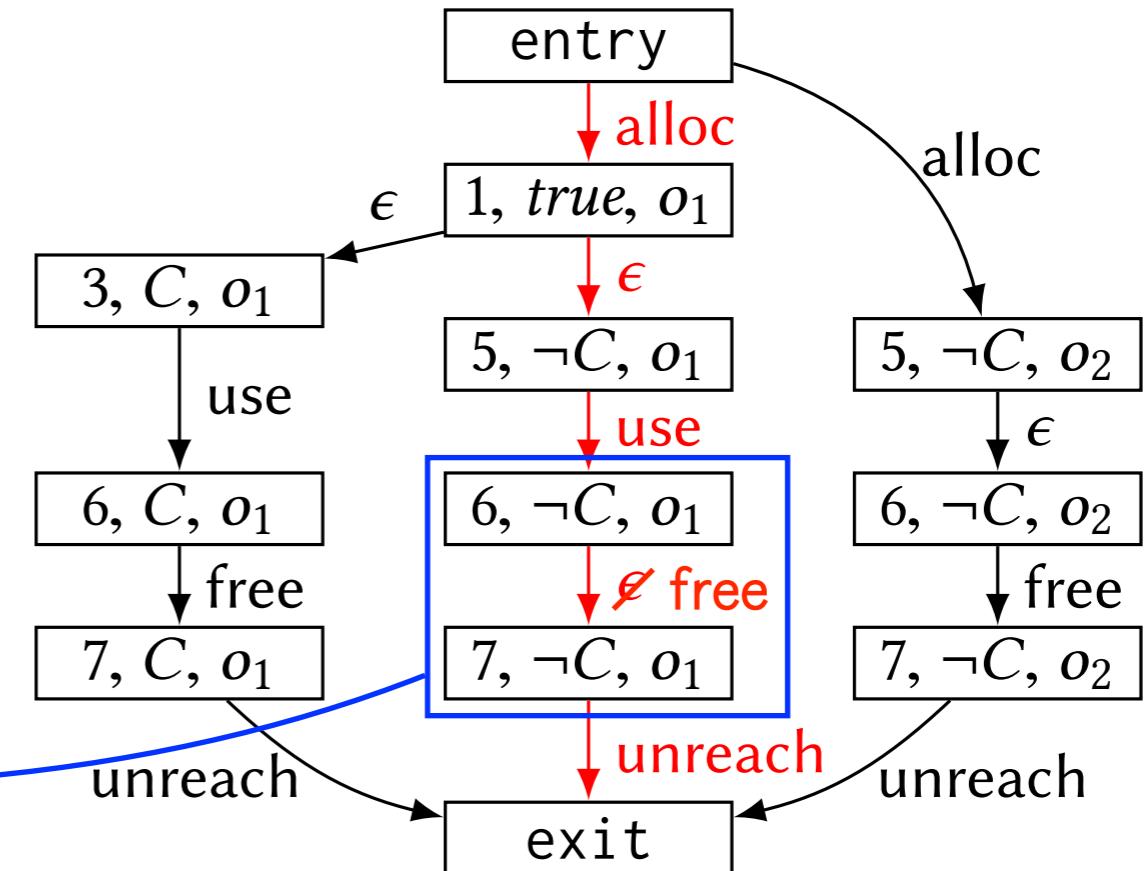
(d) Deleting free

How SAVER Works

3. Generate a patch from the re-labeled edge

```
1  p = malloc(1); //o1
2  if (C)
3    q = p;
4  else
5    q = malloc(1); //o2
6  *p = 1;
7  free(q);
```

`if (!C) free(p);`



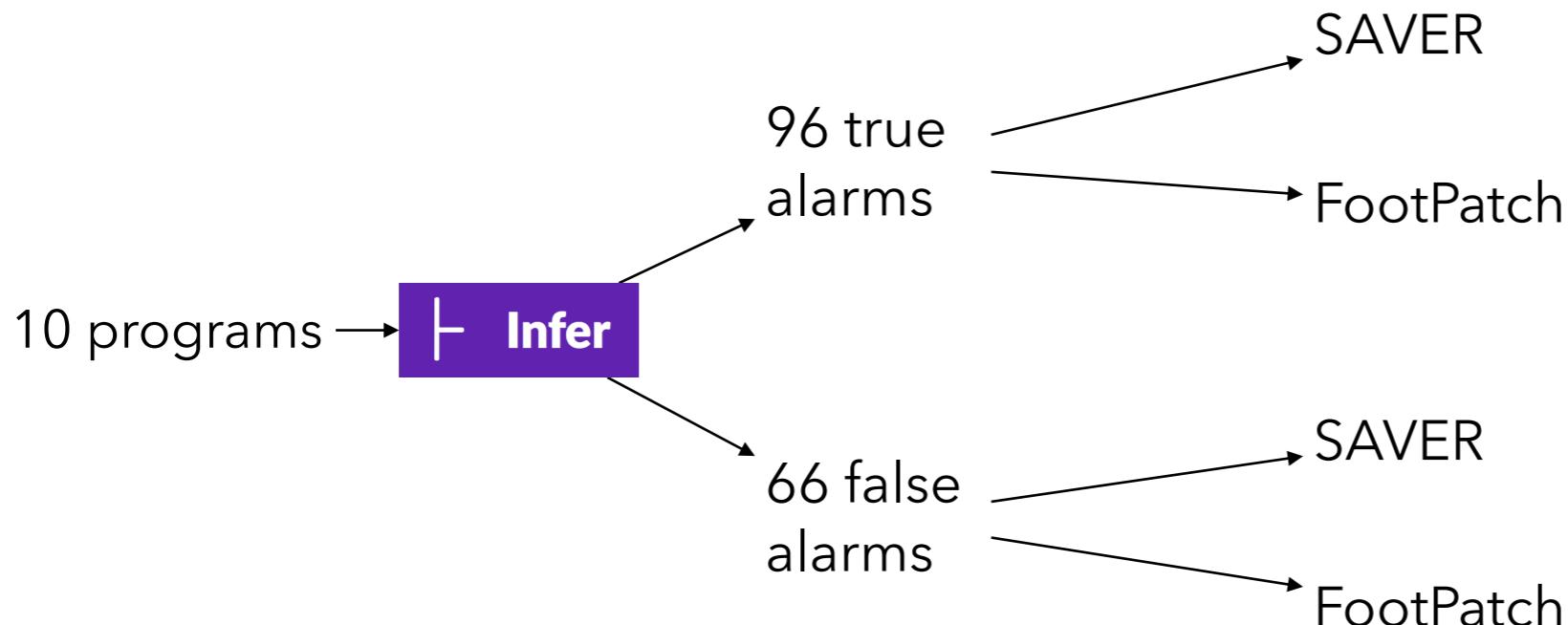
For Deployment

- Improved scalability (in the paper)
 - Selective path sensitivity
 - Program slicing
- Improved usability (not in the paper)
 - Build failures
 - Robust translation from IR-level patches to source-level

Effectiveness

Existing memory error repair tool [ICSE 2018]

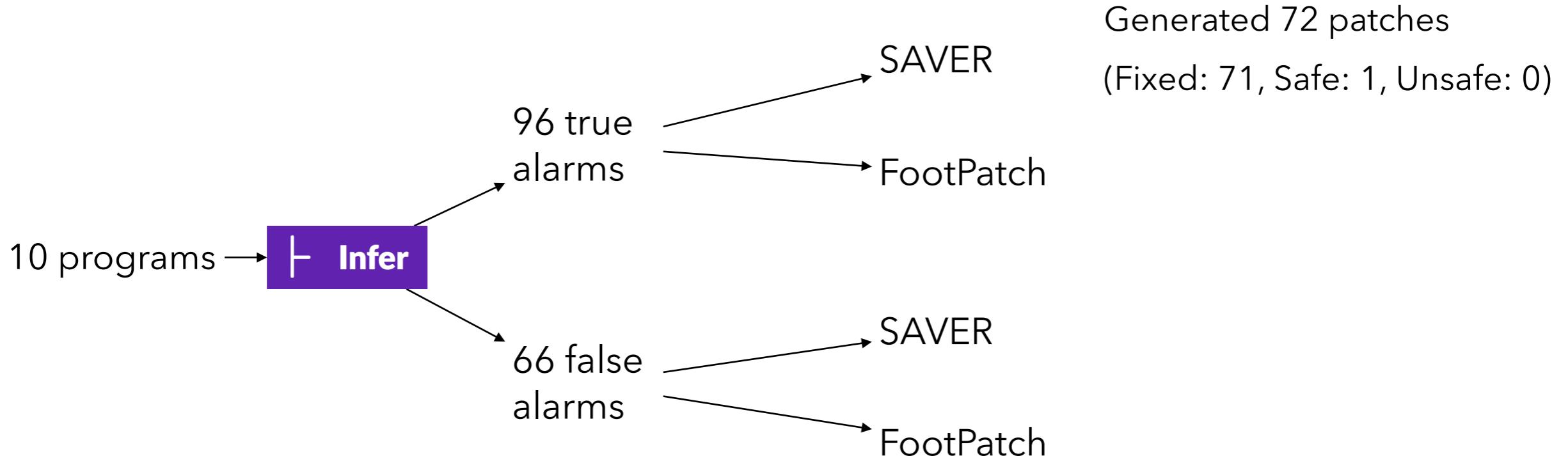
Program	kLoC	INFER			SAVER						FOOTPATCH [60]							
		#T	#F	Pre(s)	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F
rappel (ad8efd7)	2.2	1	0	2.2	0.0	1	1	0	0	0	0	8.9	1	1	0	0	0	0
flex (d3de49f)	22.3	3	4	26.3	2.5	0	0	0	0	0	0	51.0	0	0	0	0	1	1
WavPack (22977b2)	31.2	1	2	44.6	22.1	0	0	0	0	0	0	67.9	0	0	0	0	2	2
Swoole (a4256e4)	43.0	15	3	88.5	10.1	11	11	0	0	0	0	392.5	9	7	0	2	1	1
lxc (72cc48f)	49.9	3	5	230.6	5.8	3	3	0	0	0	0	179.6	0	0	0	0	1	1
p11-kit (ead7ara)	62.9	33	9	646.2	288.8	24	24	0	0	0	0	566.4	8	7	1	0	2	2
x264 (d4099dd)	73.2	10	0	144.3	9.9	10	10	0	0	0	0	426.9	2	2	0	0	0	0
recutils-1.8	92.0	10	11	144.1	44.4	8	8	0	0	0	0	662.2	3	2	1	0	0	0
inetutils-1.9.4	116.9	4	5	44.8	2.5	4	4	0	0	0	0	182.1	0	0	0	0	0	0
snort-2.9.13	320.8	16	27	2372.0	216.0	11	10	1	0	0	0	4636.4	3	0	0	3	19	18
Total	814.4	96	66	3743.6	602.1	72	71	1	0	0	0	7173.9	26	19	2	5	26	25



Effectiveness

Existing memory error repair tool [ICSE 2018]

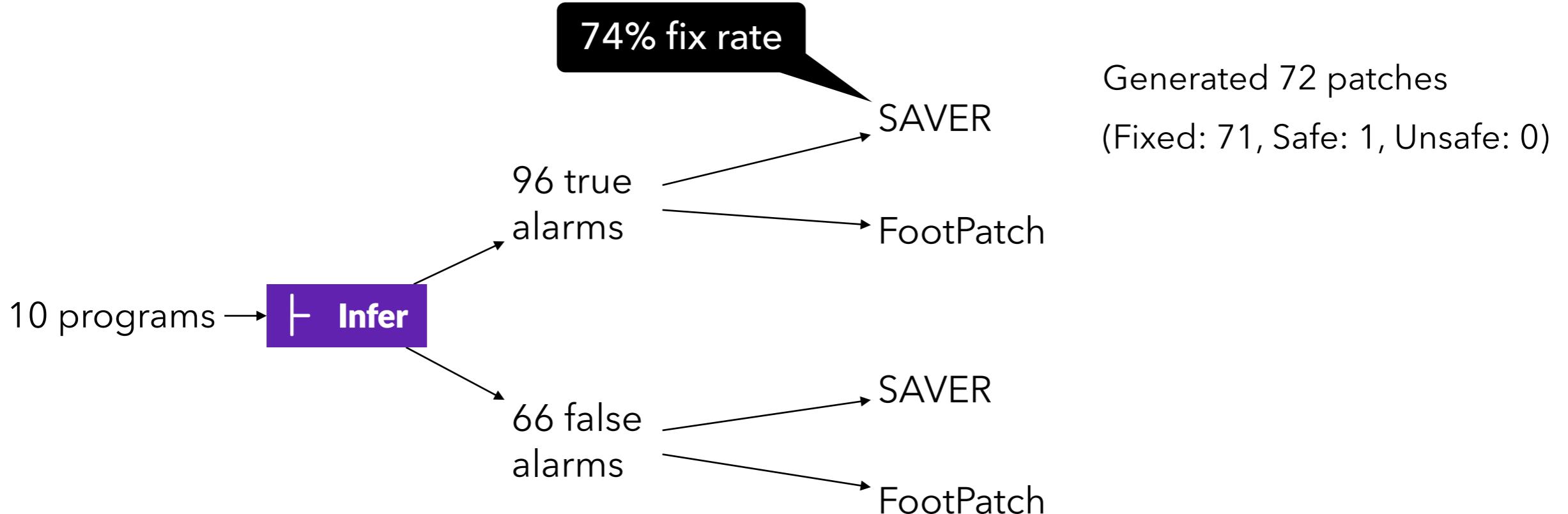
Program	kLoC	INFER			SAVER						FOOTPATCH [60]							
		#T	#F	Pre(s)	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F
rappel (ad8efd7)	2.2	1	0	2.2	0.0	1	1	0	0	0	0	8.9	1	1	0	0	0	0
flex (d3de49f)	22.3	3	4	26.3	2.5	0	0	0	0	0	0	51.0	0	0	0	0	1	1
WavPack (22977b2)	31.2	1	2	44.6	22.1	0	0	0	0	0	0	67.9	0	0	0	0	2	2
Swoole (a4256e4)	43.0	15	3	88.5	10.1	11	11	0	0	0	0	392.5	9	7	0	2	1	1
lxc (72cc48f)	49.9	3	5	230.6	5.8	3	3	0	0	0	0	179.6	0	0	0	0	1	1
p11-kit (ead7ara)	62.9	33	9	646.2	288.8	24	24	0	0	0	0	566.4	8	7	1	0	2	2
x264 (d4099dd)	73.2	10	0	144.3	9.9	10	10	0	0	0	0	426.9	2	2	0	0	0	0
recutils-1.8	92.0	10	11	144.1	44.4	8	8	0	0	0	0	662.2	3	2	1	0	0	0
inetutils-1.9.4	116.9	4	5	44.8	2.5	4	4	0	0	0	0	182.1	0	0	0	0	0	0
snort-2.9.13	320.8	16	27	2372.0	216.0	11	10	1	0	0	0	4636.4	3	0	0	3	19	18
Total	814.4	96	66	3743.6	602.1	72	71	1	0	0	0	7173.9	26	19	2	5	26	25



Effectiveness

Existing memory error repair tool [ICSE 2018]

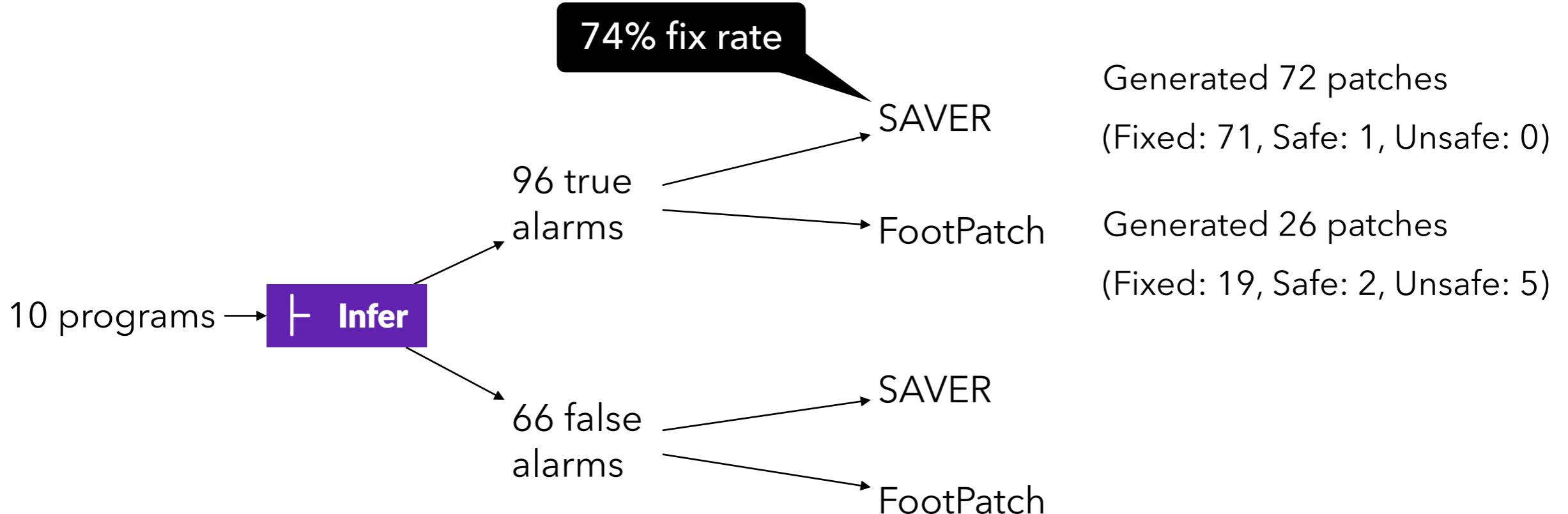
Program	kLoC	INFER				SAVER						FOOTPATCH [60]						
		#T	#F	Pre(s)	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F
rappel (ad8efd7)	2.2	1	0	2.2	0.0	1	1	0	0	0	0	8.9	1	1	0	0	0	0
flex (d3de49f)	22.3	3	4	26.3	2.5	0	0	0	0	0	0	51.0	0	0	0	0	1	1
WavPack (22977b2)	31.2	1	2	44.6	22.1	0	0	0	0	0	0	67.9	0	0	0	0	2	2
Swoole (a4256e4)	43.0	15	3	88.5	10.1	11	11	0	0	0	0	392.5	9	7	0	2	1	1
lxc (72cc48f)	49.9	3	5	230.6	5.8	3	3	0	0	0	0	179.6	0	0	0	0	1	1
p11-kit (ead7ara)	62.9	33	9	646.2	288.8	24	24	0	0	0	0	566.4	8	7	1	0	2	2
x264 (d4099dd)	73.2	10	0	144.3	9.9	10	10	0	0	0	0	426.9	2	2	0	0	0	0
recutils-1.8	92.0	10	11	144.1	44.4	8	8	0	0	0	0	662.2	3	2	1	0	0	0
inetutils-1.9.4	116.9	4	5	44.8	2.5	4	4	0	0	0	0	182.1	0	0	0	0	0	0
snort-2.9.13	320.8	16	27	2372.0	216.0	11	10	1	0	0	0	4636.4	3	0	0	3	19	18
Total	814.4	96	66	3743.6	602.1	72	71	1	0	0	0	7173.9	26	19	2	5	26	25



Effectiveness

Existing memory error repair tool [ICSE 2018]

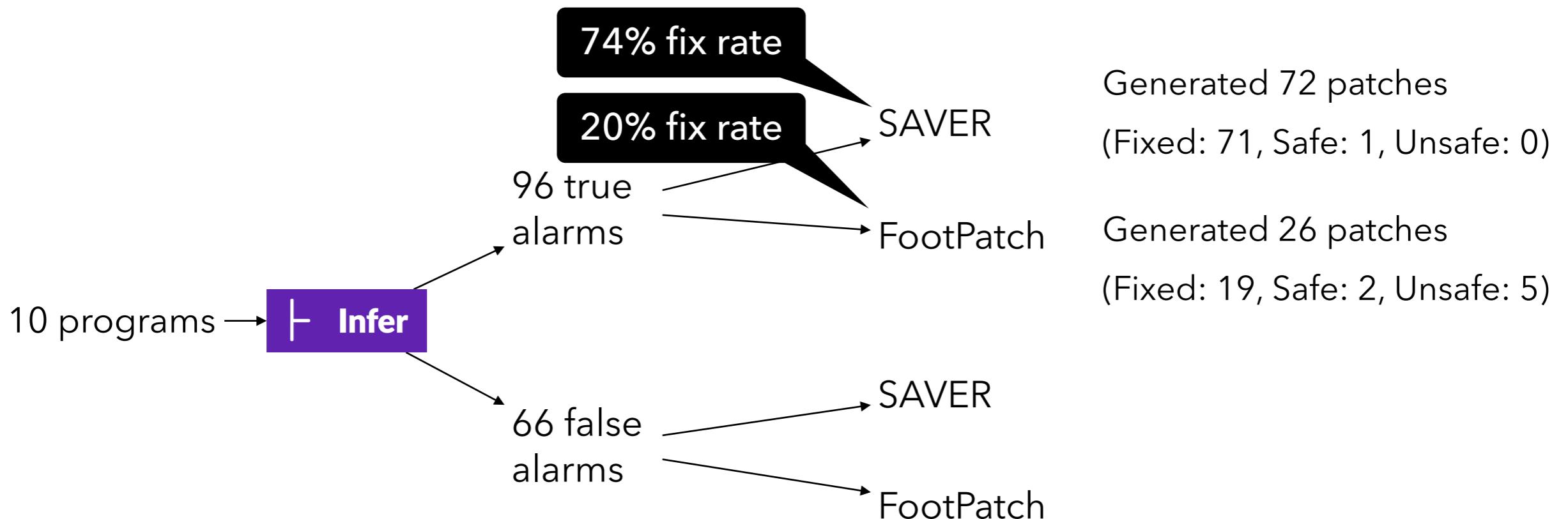
Program	kLoC	INFER				SAVER						FOOTPATCH [60]						
		#T	#F	Pre(s)	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F
rappel (ad8efd7)	2.2	1	0	2.2	0.0	1	1	0	0	0	0	8.9	1	1	0	0	0	0
flex (d3de49f)	22.3	3	4	26.3	2.5	0	0	0	0	0	0	51.0	0	0	0	0	1	1
WavPack (22977b2)	31.2	1	2	44.6	22.1	0	0	0	0	0	0	67.9	0	0	0	0	2	2
Swoole (a4256e4)	43.0	15	3	88.5	10.1	11	11	0	0	0	0	392.5	9	7	0	2	1	1
lxc (72cc48f)	49.9	3	5	230.6	5.8	3	3	0	0	0	0	179.6	0	0	0	0	1	1
p11-kit (ead7ara)	62.9	33	9	646.2	288.8	24	24	0	0	0	0	566.4	8	7	1	0	2	2
x264 (d4099dd)	73.2	10	0	144.3	9.9	10	10	0	0	0	0	426.9	2	2	0	0	0	0
recutils-1.8	92.0	10	11	144.1	44.4	8	8	0	0	0	0	662.2	3	2	1	0	0	0
inetutils-1.9.4	116.9	4	5	44.8	2.5	4	4	0	0	0	0	182.1	0	0	0	0	0	0
snort-2.9.13	320.8	16	27	2372.0	216.0	11	10	1	0	0	0	4636.4	3	0	0	3	19	18
Total	814.4	96	66	3743.6	602.1	72	71	1	0	0	0	7173.9	26	19	2	5	26	25



Effectiveness

Existing memory error repair tool [ICSE 2018]

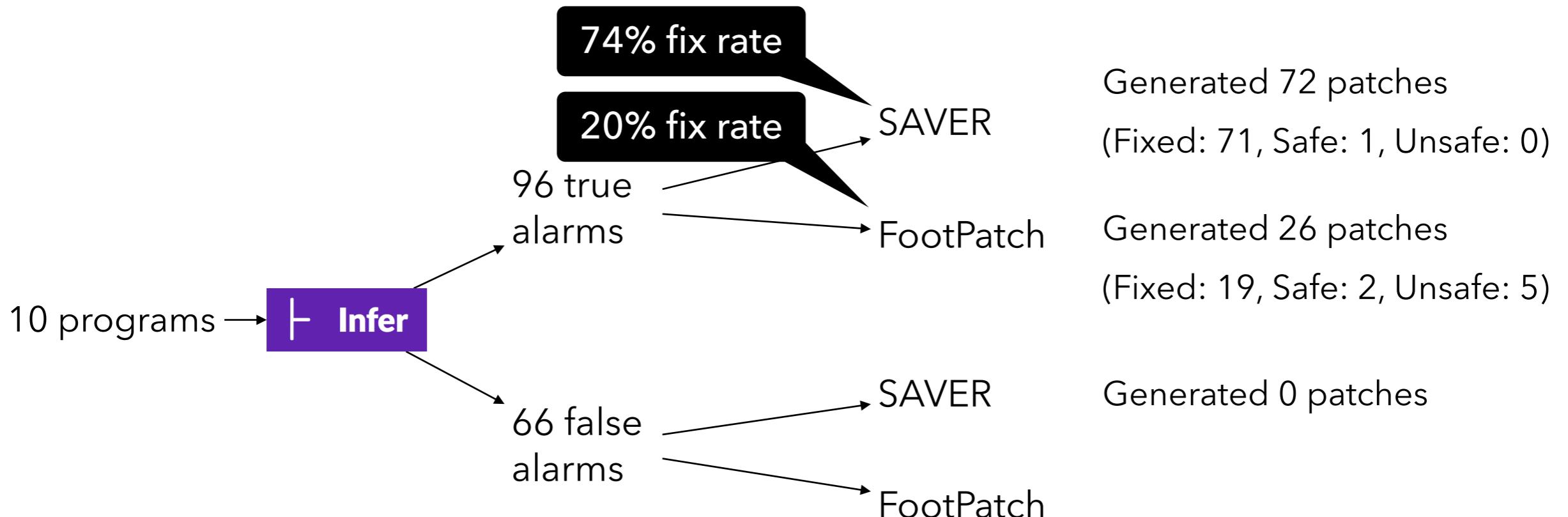
Program	kLoC	INFER				SAVER						FOOTPATCH [60]						
		#T	#F	Pre(s)	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F
rappel (ad8efd7)	2.2	1	0	2.2	0.0	1	1	0	0	0	0	8.9	1	1	0	0	0	0
flex (d3de49f)	22.3	3	4	26.3	2.5	0	0	0	0	0	0	51.0	0	0	0	0	1	1
WavPack (22977b2)	31.2	1	2	44.6	22.1	0	0	0	0	0	0	67.9	0	0	0	0	2	2
Swoole (a4256e4)	43.0	15	3	88.5	10.1	11	11	0	0	0	0	392.5	9	7	0	2	1	1
lxc (72cc48f)	49.9	3	5	230.6	5.8	3	3	0	0	0	0	179.6	0	0	0	0	1	1
p11-kit (ead7ara)	62.9	33	9	646.2	288.8	24	24	0	0	0	0	566.4	8	7	1	0	2	2
x264 (d4099dd)	73.2	10	0	144.3	9.9	10	10	0	0	0	0	426.9	2	2	0	0	0	0
recutils-1.8	92.0	10	11	144.1	44.4	8	8	0	0	0	0	662.2	3	2	1	0	0	0
inetutils-1.9.4	116.9	4	5	44.8	2.5	4	4	0	0	0	0	182.1	0	0	0	0	0	0
snort-2.9.13	320.8	16	27	2372.0	216.0	11	10	1	0	0	0	4636.4	3	0	0	3	19	18
Total	814.4	96	66	3743.6	602.1	72	71	1	0	0	0	7173.9	26	19	2	5	26	25



Effectiveness

Existing memory error repair tool [ICSE 2018]

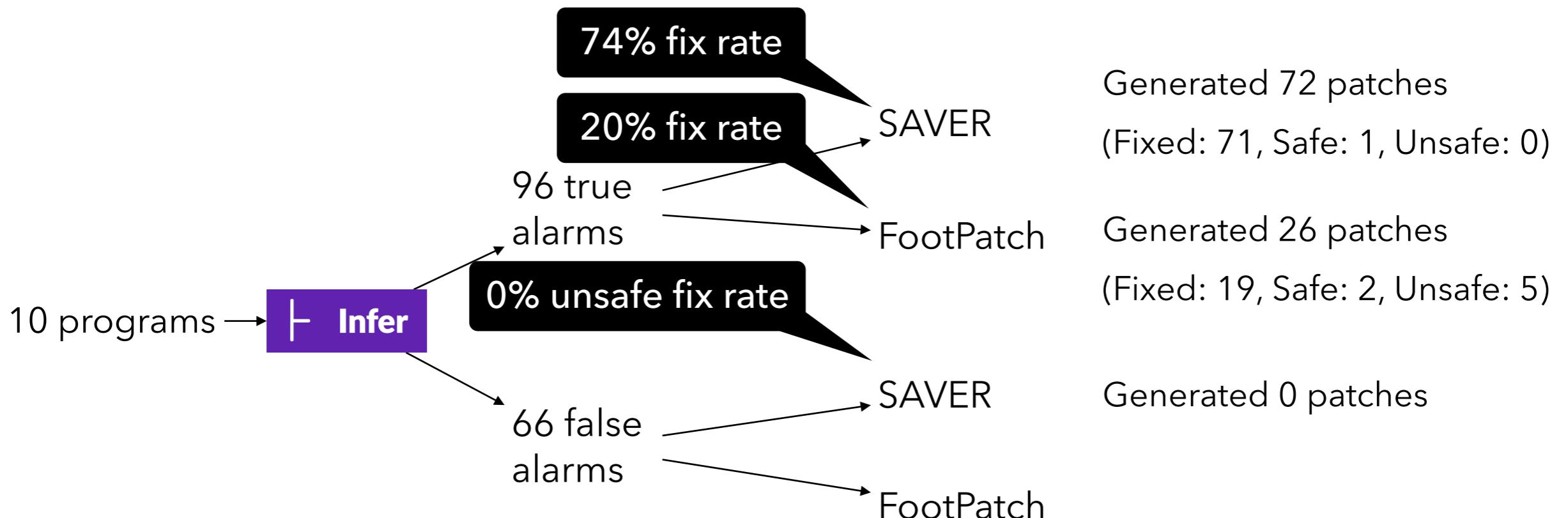
Program	kLoC	INFER				SAVER						FOOTPATCH [60]						
		#T	#F	Pre(s)	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F
rappel (ad8efd7)	2.2	1	0	2.2	0.0	1	1	0	0	0	0	8.9	1	1	0	0	0	0
flex (d3de49f)	22.3	3	4	26.3	2.5	0	0	0	0	0	0	51.0	0	0	0	0	1	1
WavPack (22977b2)	31.2	1	2	44.6	22.1	0	0	0	0	0	0	67.9	0	0	0	0	2	2
Swoole (a4256e4)	43.0	15	3	88.5	10.1	11	11	0	0	0	0	392.5	9	7	0	2	1	1
lxc (72cc48f)	49.9	3	5	230.6	5.8	3	3	0	0	0	0	179.6	0	0	0	0	1	1
p11-kit (ead7ara)	62.9	33	9	646.2	288.8	24	24	0	0	0	0	566.4	8	7	1	0	2	2
x264 (d4099dd)	73.2	10	0	144.3	9.9	10	10	0	0	0	0	426.9	2	2	0	0	0	0
recutils-1.8	92.0	10	11	144.1	44.4	8	8	0	0	0	0	662.2	3	2	1	0	0	0
inetutils-1.9.4	116.9	4	5	44.8	2.5	4	4	0	0	0	0	182.1	0	0	0	0	0	0
snort-2.9.13	320.8	16	27	2372.0	216.0	11	10	1	0	0	0	4636.4	3	0	0	3	19	18
Total	814.4	96	66	3743.6	602.1	72	71	1	0	0	0	7173.9	26	19	2	5	26	25



Effectiveness

Existing memory error repair tool [ICSE 2018]

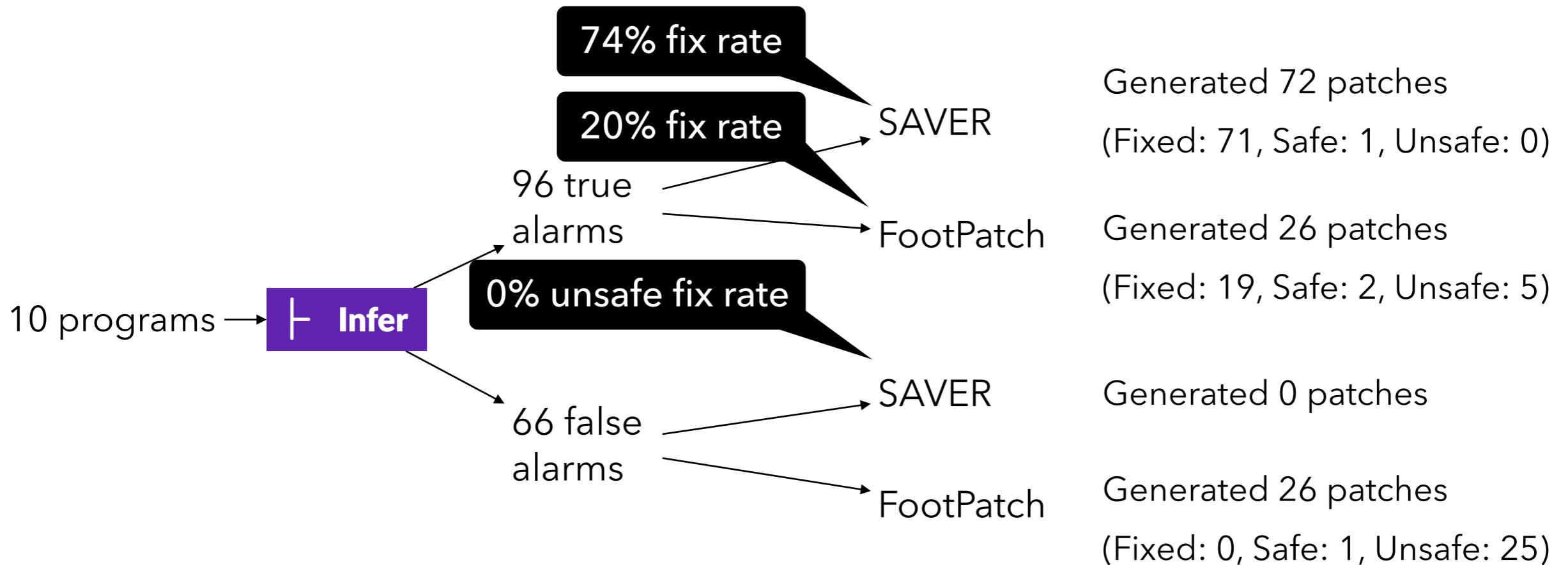
Program	kLoC	INFER				SAVER						FOOTPATCH [60]						
		#T	#F	Pre(s)	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F
rappel (ad8efd7)	2.2	1	0	2.2	0.0	1	1	0	0	0	0	8.9	1	1	0	0	0	0
flex (d3de49f)	22.3	3	4	26.3	2.5	0	0	0	0	0	0	51.0	0	0	0	0	1	1
WavPack (22977b2)	31.2	1	2	44.6	22.1	0	0	0	0	0	0	67.9	0	0	0	0	2	2
Swoole (a4256e4)	43.0	15	3	88.5	10.1	11	11	0	0	0	0	392.5	9	7	0	2	1	1
lxc (72cc48f)	49.9	3	5	230.6	5.8	3	3	0	0	0	0	179.6	0	0	0	0	1	1
p11-kit (ead7ara)	62.9	33	9	646.2	288.8	24	24	0	0	0	0	566.4	8	7	1	0	2	2
x264 (d4099dd)	73.2	10	0	144.3	9.9	10	10	0	0	0	0	426.9	2	2	0	0	0	0
recutils-1.8	92.0	10	11	144.1	44.4	8	8	0	0	0	0	662.2	3	2	1	0	0	0
inetutils-1.9.4	116.9	4	5	44.8	2.5	4	4	0	0	0	0	182.1	0	0	0	0	0	0
snort-2.9.13	320.8	16	27	2372.0	216.0	11	10	1	0	0	0	4636.4	3	0	0	3	19	18
Total	814.4	96	66	3743.6	602.1	72	71	1	0	0	0	7173.9	26	19	2	5	26	25



Effectiveness

Existing memory error repair tool [ICSE 2018]

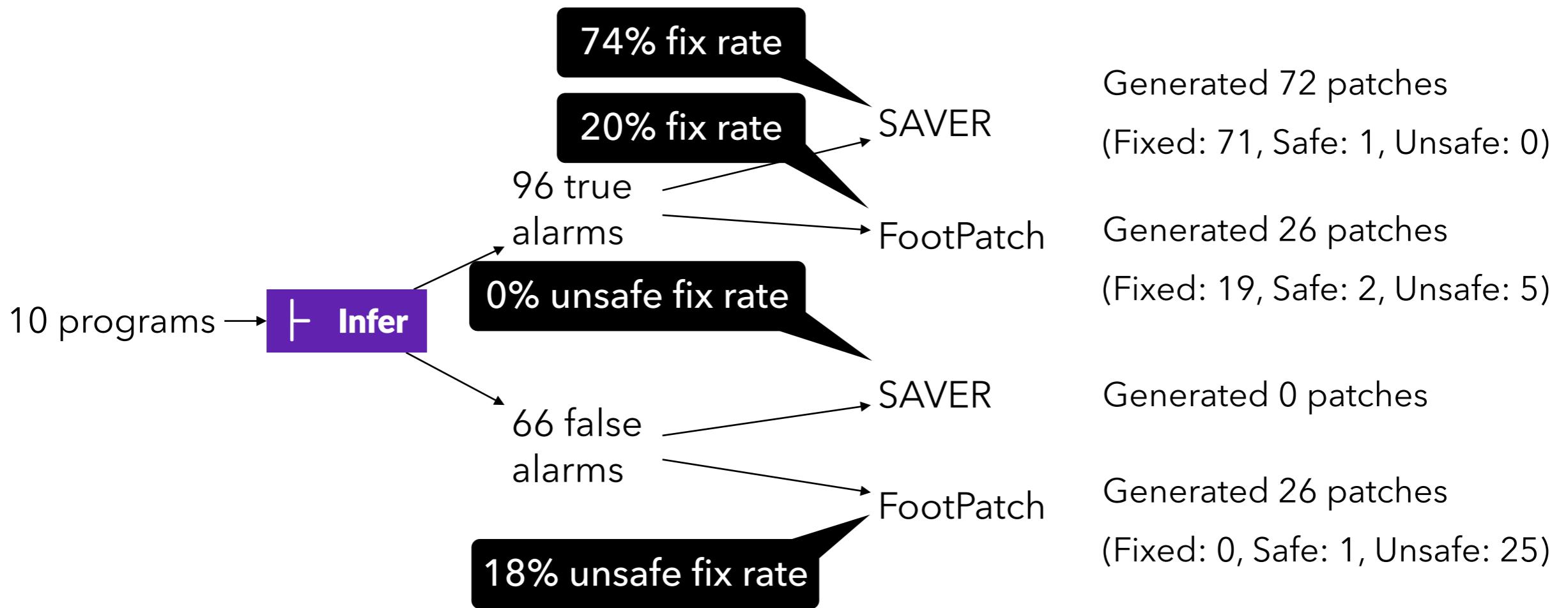
Program	kLoC	INFER				SAVER						FOOTPATCH [60]						
		#T	#F	Pre(s)	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F
rappel (ad8efd7)	2.2	1	0	2.2	0.0	1	1	0	0	0	0	8.9	1	1	0	0	0	0
flex (d3de49f)	22.3	3	4	26.3	2.5	0	0	0	0	0	0	51.0	0	0	0	0	1	1
WavPack (22977b2)	31.2	1	2	44.6	22.1	0	0	0	0	0	0	67.9	0	0	0	0	2	2
Swoole (a4256e4)	43.0	15	3	88.5	10.1	11	11	0	0	0	0	392.5	9	7	0	2	1	1
lxc (72cc48f)	49.9	3	5	230.6	5.8	3	3	0	0	0	0	179.6	0	0	0	0	1	1
p11-kit (ead7ara)	62.9	33	9	646.2	288.8	24	24	0	0	0	0	566.4	8	7	1	0	2	2
x264 (d4099dd)	73.2	10	0	144.3	9.9	10	10	0	0	0	0	426.9	2	2	0	0	0	0
recutils-1.8	92.0	10	11	144.1	44.4	8	8	0	0	0	0	662.2	3	2	1	0	0	0
inetutils-1.9.4	116.9	4	5	44.8	2.5	4	4	0	0	0	0	182.1	0	0	0	0	0	0
snort-2.9.13	320.8	16	27	2372.0	216.0	11	10	1	0	0	0	4636.4	3	0	0	3	19	18
Total	814.4	96	66	3743.6	602.1	72	71	1	0	0	0	7173.9	26	19	2	5	26	25



Effectiveness

Existing memory error repair tool [ICSE 2018]

Program	kLoC	INFER				SAVER						FOOTPATCH [60]						
		#T	#F	Pre(s)	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F	Fix(s)	G _T	✓ _T	△ _T	X _T	G _F	X _F
rappel (ad8efd7)	2.2	1	0	2.2	0.0	1	1	0	0	0	0	8.9	1	1	0	0	0	0
flex (d3de49f)	22.3	3	4	26.3	2.5	0	0	0	0	0	0	51.0	0	0	0	0	1	1
WavPack (22977b2)	31.2	1	2	44.6	22.1	0	0	0	0	0	0	67.9	0	0	0	0	2	2
Swoole (a4256e4)	43.0	15	3	88.5	10.1	11	11	0	0	0	0	392.5	9	7	0	2	1	1
lxc (72cc48f)	49.9	3	5	230.6	5.8	3	3	0	0	0	0	179.6	0	0	0	0	1	1
p11-kit (ead7ara)	62.9	33	9	646.2	288.8	24	24	0	0	0	0	566.4	8	7	1	0	2	2
x264 (d4099dd)	73.2	10	0	144.3	9.9	10	10	0	0	0	0	426.9	2	2	0	0	0	0
recutils-1.8	92.0	10	11	144.1	44.4	8	8	0	0	0	0	662.2	3	2	1	0	0	0
inetutils-1.9.4	116.9	4	5	44.8	2.5	4	4	0	0	0	0	182.1	0	0	0	0	0	0
snort-2.9.13	320.8	16	27	2372.0	216.0	11	10	1	0	0	0	4636.4	3	0	0	3	19	18
Total	814.4	96	66	3743.6	602.1	72	71	1	0	0	0	7173.9	26	19	2	5	26	25



Summary

- **Static analysis-based program repair** for C memory errors
 - Scalable, precise, and safe
 - Successfully deployed in industry
 - <https://github.com/kupl/kaprese>

Thank you!