ISSN 2799-8231

Dream + Think + Idea + come true

COMMUNICATIONS OF SOFTWARE ENGINEERING SOCIETY

09/2024 VOL.4 NO.2

소프트웨어공학 소사이어티 소식

Contents

COMMUNICATIONS
OF SOFTWARE ENGINEERING SOCIETY

소프트웨어공학 소사이어티 소식

09/2024
VOL.4 NO.2
SOFTWARE ENGINEERING SOCIETY

• 발행인 이정원 • 편집인 이주용, 정우성, 지은경, 손정주 • 발행일 2024. 9.1

게스트 편집자 인사말

이 주 용
UNIST 교수

안녕하세요. UNIST 이주용입니다.

AI 바람이 해가 갈수록 거세게 불고 있습니다. AI와 관련된 연구 및 개발이 전 세계를 강타하고 있고, 소프트웨어 공학 분야도 예외가 아닙니다. 아니, LLM(Large Language Model)의 등장으로 소프트웨어 개발 방식이 뿌리채 바뀔 태세입니다. 과연 그럴까요? 여러분은 어떻게 생각하시나요?

"뿌리 깊은 나무는 바람에 흔들리지 않는다." 용비어천가에 나오는 유명한 문구이지요. AI 바람에 대한 얘기는 최근 귀가 박히도록 들으셨을 테니, 이번 호에서는 소프트웨어 공학의 뿌리에 대한 얘기를 해볼까 합니다. 물론, 이제 거목이 된 소프트웨어 공학의 모든 뿌리를 제 좁은 식견으로 편집하는 것은 불가능합니다. 그래서, 여기서는 편집자에게 가장 큰 영향을 끼친 뿌리 하나에 대해서 다루고자 합니다.

제 은사님이신 최진영 교수님이 올해 8월 고려대학교에서 은퇴를 하셨습니다. 최진영 교수님은 근 30년 동안 "정형기법 연구실"을 운영하셨지요. 현재 저는 정통 정형기법을 연구하고 있지는 않지만, 제가 견지하는 연구 관점과 연구 방법론은 많은 부분 정형기법에 그 뿌리를 두고 있습니다. 예를 들어, 제가 깊숙히 관여한 의미기반 자동 프로그램 수정 기법은 정형기법에서 널리 활용되는 논리학 지식에 기반하고 있습니다.

서두가 길었습니다. 이번 호는 "정형기법"을 근간에 놓고 아래의 분들을 초대해서 구성해 보았습니다.

- 정형기법을 포함한 "소프트웨어 공학 이야기"에 대해 최근 개정증보판을 출간하신 고려대학교 차성덕 교수님
- 삼성미래기술육성재단 지원으로 신경망 시스템의 정형검증을 주제로 연구를 하시는 연세대학교 김지응 교수님
- 실용적인 모델체킹 기반 테스팅 기술을 올해 FSE에 발표하신 최윤자 교수님과 지도 학생 김요엘 박사과정생
- 올해 ICST에서 Most Influential Paper Award 수상을 하신 카이스트 김문주 교수님
- 한국의 대표적인 정형기법 기반 기업인 (주)포멀웍스의 CEO 김태효 대표님
- 통계적 프로그램 분석이라는 새로운 영역을 개척 중인 Max Planck Institute의 이성민 박사님

이 외에도 올해에 저희 소사이어티 회장으로 추대되신 아주대학교 이정원 교수님 인사말과 올해 8월에 진행된 단기 강좌에 대한 후기를 수록했습니다. 단기 강좌 후기는 조직 위원장으로 수고를 아끼지 않으신 전북대학교 류덕산 교수님과, 단기 강좌에 참석한 한양대학교 조한결 박사과정생이 작성하였습니다. 소식지 구성을 위해 애써주신 모든 분들께 감사 드립니다.

요새 ChatGPT와 대화를 하시면서, 사람과의 대화가 그립지는 않으셨나요? 이번 호의 많은 부분을 대화 형식으로 구성하였습니다. 모쪼록 사람간의 품격 있는 대화를 만끽하시길 바랍니다.

2024년 8월 어느 밤, 무더위에 작별을 고하며

이 주 용 드림

이 정 원
아주대학교 교수

안녕하세요. 한국정보과학회 소프트웨어공학 소사이어티의 제 19대 회장을 맡고 있는 아주대 이정원입니다.

최근 자율주행자동차, UAM(Urban Air Mobility), 산업용로봇 등과 같은 스마트 모바일 디바이스에서부터 LLM(Large Language Model)을 활용한 ChatGPT, GitHub Copilot 등의 Cloud기반 AI도구 들에 이르기까지 인공지능 기술은 소프트웨어의 규모와 복잡도를 한층 더 배가시키고 있습니다. 이러한 스마트 디바이스와 AI모델은 소프트웨어공학적인 체계적인 개발방법 및 검증과정을 더욱 필요로하고 있으며 이에 소프트웨어공학 소사이어티의 역할과 책임이 매우 큰 시기라고 할 수 있습니다.

소프트웨어공학 소사이어티는 국내외 소프트웨어공학 분야의 연구, 개발, 그리고 산업에의 적용을 목적으로 학계와 산업계의 모든 분들에게 열려 있는 학술 단체입니다. 1987년 소프트웨어공학 연구회로 출범하여 올해 10월에는 만 37세를 맞이합니다. 이번 19대 회장단에서는 2년간 KCSE(Korea Conference of Software Engineering)와 같은 국내 학술대회를 더욱 활성화하고, AI와 SE를 접목한 소프트웨어공학 최신기술을 다루는 단기전문가 강좌를 통해 심화 기술을 확산하며, 2025년, 2026년에는 국제 저명 학술대회 ASE(Automated Soft Engineering)와 ICST(International Conference on Software Testing, Verification, and Validation)를성공적으로 치뤄내기 위해 최선을 다하겠습니다.

이러한 소사이어티의 활발한 활동소식을 '소프트웨어공학 소사이어티 소식지'를 통해 많은분들과 공유하고자 2021년 12월 창간하여 이번호가 7번째가 되었습니다. 소프트웨어공학 최신기술 소개, 소프트웨어공학자 및 산업전문가의 기고문, 관련기관 탐방, 신진 연구자 소개, 학술행사 보고 및 대학원생 참관기, 회원 동정 등의 내용을 알차게 구성하였으니, 회원 여러분들에게 유익하고 흥미로운 정보가 되시길 바랍니다.

저는 이 소식지를 통해 소사이어티의 발전은 물론 회원 여러분들의 결속력을 다지는 데에 큰 밑거름이 될 것이라고 믿습니다. 전임 편집장이셨던 경북대학교 최윤자 교수님께 감사드리며 이번호부터 편집부회장을 맡게 되신 UNIST의 이주용 교수님을 비롯한 편집위원님들께 큰 감사의 말씀을 전합니다.

항상 소프트웨어공학 소사이어티의 발전을 위해 도움 주시는 회원 여러분께 감사드리며 앞으로 우리 소식지에 더욱 큰 관심과 지원 부탁드립니다.

소프트웨어공학 소사이어티 회원 여러분의 발전과 번영이 항상 함께하기를 기원합니다.

감사합니다.

제19대 소프트웨어공학 소사이어티 회장 **이 정 원**

NEWS

고려대학교 차성덕 교수님의
"소프트웨어공학 이야기" 발췌

COMMUNICATIONS OF SOFTWARE ENGINEERING SOCIETY

편집자: 차성덕 교수님의 "소프트웨어공학 이야기" 책에 정형검증 기법에 대해 알기 쉽게 작성된 챕터가 있어 소식지 지면에 옮깁니다. 이 챕터를 통해 이번 호에서 다루는 내용들에 대해 감을 잡으실 수 있을 겁니다. 소식지에 싣는 것을 허락해 주신 차성덕 교수님과 홍릉과학출판사에 감사를 표합니다

정형검증 기법은 소프트웨어 품질에 어떻게 기여할 수 있을까?

■ **차성덕** 교수 (고려대학교)

정형검증 기법은 소프트웨어 품질보증에서 테스팅기법의 한계를 보완해 줄 수 있는 효과적인 대안으로 각광을 받고 있다. 테스팅의 근본적인 한계는 모든 경우의 입력에 대해서 소프트웨어가 제대로 작동한다는 것을 보이는 것이 현실적으로 불가능하다는 것이다. 대안으로 입력값을 유사한 그룹으로 묶어서 각 그룹에 속하는 값을 임의로 골라서 테스팅을 하는 기법이 equivalent partition 기법이다. 예를 들어서 정수형 입력값의 특성을 분석해 보니, 음수값을 하나의 그룹으로 묶을 수 있고, 0부터 10사이를 다른 그룹, 11부터 99까지를 또 다른 그룹, 100 이상되는 값들을 별도의 그룹으로 묶을 수 있다고 가정하면, equivalent value partition 기법[1]은 각 그룹에 속하는 값 중에서 하나를 임의로 골라서 테스트케이스의 입력값으로 사용하는 것이다. 또한 경험적으로 볼 때 각 그룹의 경계에 해당되는 값들을 제대로 처리하지 못해서 오류가 흔히 발생하므로, 경계값을 테스트케이스로 사용하는 방법이 있는데 이를 boundary value analysis라고 한다. 위의 예제라면 boundary value analysis는 특별히 -1, 0, 10, 11, 99, 100 등 각 그룹의 경계에 있는 값들을 사용하여 프로그램이 제대로 작동하는지 확인하는 것이다.

경험과 직관에 따라서 테스팅을 한다고 하여도 테스팅의 근본적인 한계는 극복할 수 없기에 소프트웨어공학 및 이론전산학 연구자들은 새로운 방법론을 연구하게 되었다. 그 중 대표적인 것이 정형기법인데 정형명세 기법과 정형검증 기법으로 나눌 수 있다. 정형검증의 구체적인 기법은 symbolic execution, program proof/verification, model checking, program synthesis 등을 포함한다.

높은 신뢰도를 요구하는 safety-critical 시스템의 경우에는 각 단계마다 체계적인 검증이 필수적이다. DO-178C에서 정의한 여러 등급(Design Assurance Level, 또는 DAL) 중에서 최고 등급인 catastrophic 등급으로 분류되는 시스템(또는 소프트웨어)의 경우 오류가 인명피해 그리고 일반적으로는 비행기의 추락 등의 사고를 유발할 수 있으므로 더욱 높은 신뢰도를 요구한다. 이런 제품의 경우 정형명세 기법 또는 정형검증 기법이 효과적인 방법으로 추천이 되며, 일부 표준에서는 정형기법의 사용을 필수로 요구하기도 한다.

EAL1: Functionally Tested
EAL2: Structurally Tested
EAL3: Methodically Tested and Checked
EAL4: Methodically Designed, Tested and Reviewed
EAL5: Semiformally Designed and Tested
EAL6: Semiformally Verified Design and Tested
EAL7: Formally Verified Design and Tested

침입탐지 시스템, 방화벽, Secure 운영체제 등 보안제품을 평가하는 기준인 Common Criteria[2] (CC)는 7개의 보안평가등급을 정의하고 있는데, 가장 낮은 EAL1 등급은 보안기능에 대한 기능검사만 요구될 뿐이다. 구현된 보안기능이 제대로 구현되었을 때 기대할 수 있는 입력값과 출력값을 포함하는 테스트케이스를 만들고, 이러한 테스트케이스를 사용하여 시스템의 보안기능을 시험하고 합격하면 EAL1 인증을 받는 것이다. 이런 보안 인증은 최소한의 보안 기능을 검사할 수 있을 뿐이므로 높은 신뢰성을 기대하기는 어렵다.

중간등급인 EAL4에서는 단순한 기능검사만으로는 부족하고, 개발 전 과정에 걸쳐서 보안 기능에 대한 분석이 적절하게 고려되었음을 보이도록 요구한다. "Methodically designed"라는 요구사항을 만족시키려면 소프트웨어 디자인 단계에서 보안에 필수적인 기능이 어느 모듈에서 구현되었는지, 그리고 전체적인 시스템의 디자인은 모듈화의 개념에 따라 architecture가 적절하게 정의되었는지 확인한다. 그리고 review가 요구사항에 포함되어 있으므로 평가자들은 security 관점에서 중요한 모듈의 소스코드까지 분석하여 보안기능에 대한 구현이 적절한지 확인한 후에 EAL 4 등급의 승인을 추천하게 된다.

CC에서 제일 높은 등급인 EAL 7은 더 강도높은 분석을 요구하는데 그것이 Formal Verification of Design이다. 보안 기능에 대한 정형명세는 물론이고, 시스템이 제공하는 security policy 또한 "formal" notation으로 기술되어야 하고, policy와 구현의 일관성 또한 수학적인 이론에 기반한 증명기법으로 확인하여야 한다.

1) https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html

2567 Certified Products by Category*		
Category	Products	Archived
Access Control Devices and Systems	55	81
Biometric Systems and Derives	3	0
Boundary Protection Devices and Systems	69	139
Data Protection	78	97
Databases	31	53
Detection Derices and Systems	13	58
ICs, Smart Cards and Smart Card-Related Devices and Systems	1229	60
Key Management Systems	24	28
Mobility	31	33
Multi-Function Devices	211	196
Network and Network-Related Devices and Systems	266	258
Operating Systems	110	76
Other Devices and Systems	287	347
Products for Digital Signatures	110	8
Trusted Computing	50	0
Totals:	2567	1434
Grand Total:		4001

Certified Products by Assurance Level and Certification Date																				
EAL	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	Total
EAL1	0	0	0	0	0	0	1	1	6	3	1	0	1	10	2	1	3	3	8	7	2	49
EAL1+	1	0	0	0	0	0	0	0	17	0	2	11	2	0	1	2	1	0	0	1	0	38
EAL2	0	0	0	0	0	0	1	0	8	1	6	2	3	1	6	13	18	15	26	13	10	123
EAL2+	0	0	0	1	1	1	2	2	8	8	8	4	4	7	7	19	59	76	66	42	21	336
EAL3	0	0	0	0	0	0	0	0	10	3	0	2	4	1	5	8	9	1	3	6	6	58
EAL3+	0	0	0	0	0	2	1	1	32	8	10	9	12	15	5	21	17	19	10	9	1	172
EAL4	0	1	0	1	0	0	0	0	15	2	3	4	5	1	6	2	0	5	2	9	3	59
EAL4+	0	1	1	2	2	3	2	2	141	54	63	52	57	82	61	49	56	55	52	49	31	815
EAL5	0	0	0	0	0	0	0	0	6	3	2	0	1	0	0	0	0	3	1	4	1	21
EAL5+	0	0	0	0	0	0	3	0	50	27	31	41	34	27	53	48	41	69	69	55	23	571
EAL6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EAL6+	0	0	0	0	0	0	0	0	0	0	2	3	0	4	5	6	9	8	12	22	12	83
EAL7	0	0	0	0	0	0	0	0	0	0	1	0	0	0	4	0	0	0	0	1	0	6
EAL7+	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	2
Basic	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Medium	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
US Standard	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
None	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	14	15	45	88	64	234
Totals	1	2	1	4	3	6	10	6	293	109	129	129	123	148	159	173	227	269	294	307	174	2567

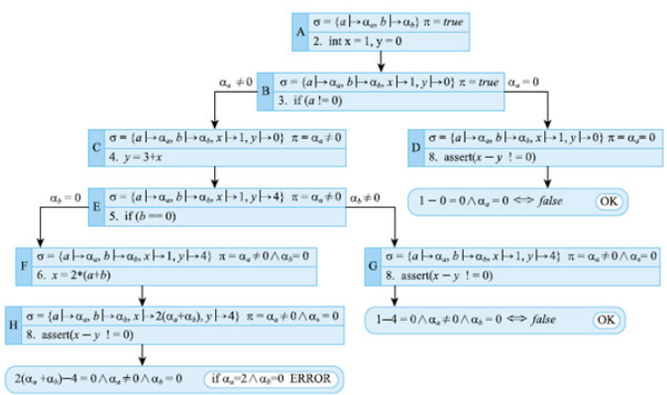
Common Criteria와 평가완료된 보안제품들

Common Criteria가 제정된 1999년부터 2019년까지 미국, 호주, 한국, 영국, 캐나다 등 각 국에서 평가를 마친 2,567개의 보안시스템에 대한 통계자료[3]를 분석하면 다음과 같다.

- IC나 스마트카드 등의 관련제품이 1,229개로 전체의 거의 절반에 해당하며, 그 외에 네트워크 보안에 관련된 장치들(266개), Secure 운영체제(110개), 전자서명 제품(110개) 등이 포함되었다.
- CC 평가를 완료한 많은 제품들의 등급이 EAL 4 또는 5등급에 몰려있고, EAL 1 또는 EAL 7 등급의 평가제품은 상대적으로 적다. EAL 1 등급의 인증을 받은 제품은 87개이고, 특히 EAL 7 등급의 인증을 받은 제품은 고작 8개에 불과하다. EAL 1 등급의 제품이 적은 것은 현실적으로 기대할 수 있는 보안 신뢰성이 낮아서 시장의 외면을 받은 것으로 보이고, 낮은 수준의 보안 기능이 요구되는 경우에도 EAL 2 또는 EAL 3 등급의 제품을 주로 선택한다는 추론을 가능하게 한다. 비슷한 이유로 중간 등급의 보안신뢰성이 요구되는 경우 EAL 4 등급의 제품이, 높은 보안 등급을 필요로 하는 경우에도 EAL 5 또는 EAL 6등급의 제품이 선택될 가능성이 높아 보인다.
- EAL 6 등급과 EAL 7 등급 차이는 semi-formal과 formal이 전부여서 사소한 것처럼 보일 수 있지만, 현실적으로 그 차이는 매우 크며 기술적인 장벽 또한 매우 높다. EAL 6 등급의 Semi-formal의 경우 UML 등의 기법을 사용하여 security architecture를 기술하고, 이에 대한 "설득"과 설명으로도 평가에 합격할 수 있는 가능성이 있는 반면에, EAL 7 등급에서는 security architecture, security policy 등에 대한 mathematical proof가 필수이다. EAL7 등급의 제품이 적은 것은 수요도 적어서 그렇겠지만 개발 및 인증의 어려움 또한 영향을 끼쳤을 것이다. EAL 7 등급 제품의 경우에는 정형명세와 정형검증이 필수적인데 디자인 문서가 자연어로 기술되어 있다면 정형검증은 불가능하기 때문이다.

```
1.  void foobar(int a, int b)  {
2.    int x = 1, y = 0;
3.    if (a != 0) {
4.      y = 3+x;
5.      if (b == 0)
6.        x = 2*(a+b);
7.    }
8.    assert(x-y != 0);
9.  }
```

정형검증 기법으로 제일 처음 제안된 symbolic execution은 어떤 프로그램을 1이나 -1 등의 특정한 값을 입력으로 사용해서 수행하는 것이 아니라, 입력을 a 등의 기호로 나타내고 프로그램의 수행결과를 symbolic expression을 통해서 나타내는 것이다. Baldoni의 논문[4]에서 예제로 사용된 같은 프로그램을 사용하여 기본적인 아이디어를 공부해보자.



프로그램의 분기조건은 입력값인 a와 b의 값이 0과 같은지 다른지이며, 이에 따라서 프로그램의 실행경로가 달라짐을 알 수 있고(line 3, line 5), a와 b의 값을 (0, 0), (0, 1), (1, 0), (1, 1) 등 네 가지의 입력을 사용하여 테스팅하면 모든 경로를 따라서 수행할 수 있다. 물론 (0, 0), (0, -1), (-1, 0), 그리고 (-1, -1)을 가지고도 같은 효과를 기대할 수 있다. 그러나 이 프로그램에서 언제 8번 줄에 표시된 assert((x-y) != 0) 조건이 만족되지 않는 a와 b의 입력값을 알고 싶다면, 임의의 a, b값을 가지고 assert문이 에러메시지를 출력하고 프로그램 수행이 종료될 때까지 반복적으로 수행해야 한다. 임의의 입력값을 생성하여 사용하는 테스팅의 경우 위의 조건을 만족하는 a와 b의 값이 언제 선정될지

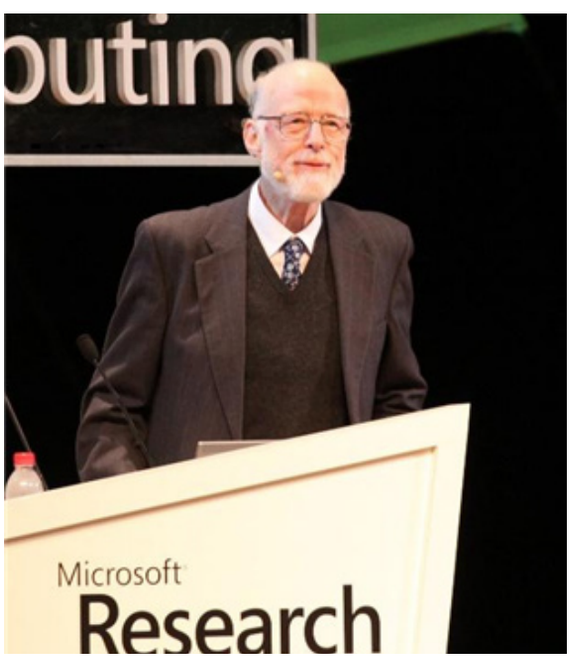
2) https://en.wikipedia.org/wiki/Common_Criteria
3) https://www.commoncriteriaportal.org/products/stats/
4) Baldoni et al., A survey of Symbolic Execution Techniques, ACM Computing Surveys, Vol 51, No 3, 2018

모르고, 설령 그런 경우가 발견되었다고 하여도 또 다른 어떤 a와 b의 값에 대하여 assert문이 에러메시지를 낼지 알 수 없다.

Symbolic execution은 특정 값이 아닌 symbolic value, 즉 a와 b의 값을 α와 β로 표현하고 프로그램을 수행하는 것이다. 프로그램의 수행 결과 또한 symbolic 값으로 표시되게 된다. π로 표시된 path condition의 초기값은 true이나 프로그램이 4번째 줄을 수행하기 위해서는 a의 값이 0이 아니어야 하고(B에서 C로의 전이), π는($\alpha_\alpha \neq 0$)로 변경되었다. 또한 y=3+x 구문을 수행하면 그 결과로 x와 y의 값이 1과 4로 변경된다(노드 E). 그리고 5번째 줄의 수행 여부에 따라서 path expression은 더 자세한 정보를 가지게 되고(노드 F와 G), 각각의 경우 symbolic execution을 계속하면 H노드에 표시된 대로 a의 값이 2, b의 값이 0인 경우에만 에러메시지가 생성될 수 있음을 알 수 있다.

이렇게만 생각하면 symbolic execution이라는 아이디어가 단순하고 모든 프로그램에 쉽게 적용될 수 있을 것 같지만, 현실은 그렇지 않은데, 한 두가지의 기술적인 어려움은 다음과 같다.

- table[x]와 같이 array 특히 index의 값이 변수로 표시되는 경우나 포인터가 사용된 프로그램의 symbolic execution은 난관에 직면하게 된다. Symbolic 값이니 pointer는 메모리 영역의 아무 주소나 임의로 나타낼 수 있기 때문이다. 또한 이런 값들이 함수 호출시에 parameter로 사용된다면 symbolic execution은 매우 어려운 문제가 된다.
- while(a>0) 등과 같은 반복문의 경우, 구체적인 값이 아닌 symbolic 값으로는 반복문이 몇 번이나 수행될지 결정할 수 없고, loop invariant를 알기 전에는 프로그램의 수행효과를 symbolic expression으로 표현할 수 없다. Loop invariant란 반복문을 몇 번을 수행하든 상관없이 항상 만족되는 조건을 나타내는데, 이를 구하는 것은 이론전산학 분야에서 오래전에 NP-complete 문제라는 것이 증명되었다. Loop invariant를 자동으로 생성할 수 없으니 간단한 프로그램에서도 symbolic execution을 수행하는 것이 어려운 것이다.



이러한 이유로 1976년에 제안된 symbolic execution의 아이디어가 실제 산업체의 프로그램에 적용할만한 수준으로 발전되는데는 거의 반세기에 가까운 시간이 필요했다. 최근에 들어서야 KLEE, PathFinder, SAGE, CUTE 등의 도구들이 각 연구기관에서 발표되어 사용될 수 있는 수준에 이르게 되었다. 예를 들어서 마이크로소프트 연구소에서 개발된 SAGE라는 도구는 MS사에서 개발 중 발견되는 전체 오류의 30% 가량을 자동적으로 찾아낸다고 하니 이러한 기법이 프로그램의 품질과 생산성의 향상에 얼마나 기여할 수 있는지 알 수 있다.

그 외에도 Program Verification이라는 아이디어를 사용해서 프로그램이 모든 경우의 입력에 대해서 제대로 작동함을 보여주려는 시도 또한 있었는데, 이 연구의 시초는 Turing Award를 수상한 C.A.R. Hoare경[5]이 1969년에 발표한 "Hoare logic"이라는 논문에 기초하고 있다. 흔히 Hoare triple이라고도 불리는 이 아이디어는 {P} C {Q}로 표시되는데, 프로그램 또는 특정 구문 C를 수행하기 전에 만족해야 하는 precondition을 P, 수행 후에 만족하는 postcondition을 Q로 표현할 때, 프로그램의 수행결과를 수학적으로 나타낼 수 있다는 것이다. 이런 제안은 프로그램이 모든 입력에 제대로 작동한다는 것을 테스팅을 통해서 보여줄 수도 없지만 그럴 필요도 없다는 "발상의 전환"에서 시작되었다.

이런 시도는 흔히 program correctness proof 또는 program verification이라고도 불리는데, 예를 들어서 두 개의 양의 정수 x와 y의 값을 가지고 나눗셈을 하여 몫 (q)과 나머지 (r)을 구하는 다음과 같은 코드를 생각해 보자.

```
1: r := x; q := 0;
2: while (r >= y) do  {
3:   r := r - y;
4:   q := q + 1;
5:  }
```

이 프로그램을 x와 y의 값을 가지고 끝없이 (?) 테스팅하는 것이 아니라, 문제의 정의로부터 precondition이 (x≥0, y>0)라는 것을 알 수 있고, 자동으로 생성할 수는 없지만 전문가의 노력으로 이 문제에 해당하는 Loop invariant가 (x = y*q + r, 0≤r<y) 것을 알고 있다고 가정해 보자. 그렇다면 program proof는 프로그램의 수행 과정 및 결과를 다음과 같이 표현할 수 있다 [6].

```
{ x ≥ 0 && y > 0 }  // p1: precondition
1: r := x; q := 0;
{ r ≥ 0 && y > 0 && x = y*q + r }  // p2
2: while (r >= y) do
{ r ≥ 0 && 0 < y ≦ r && x = y*q + r } // p3
3: r := r - y;
4: q := q + 1;
{ r ≥ 0 && 0 < y ≦ r && x = y*q + r } // p4
5:  }
{ 0 ≦ r < y && x = y*q + r }  // p5: postcondition
```

5) 사진은 Hoare경이 Oxford대학교에서 은퇴후 마이크로소프트 연구소의 Principal researcher로 재직중 2008년 북경 Microsoft 아시아연구소에 주최한 Computing in the 21st Century학회에서 기조연설을 하는 장면을 직접 찍은 사진이다. Hoare logic에 대한 자세한 설명은 https://en.wikipedia.org/wiki/Hoare_logic에서 구할 수 있다

p1으로 표시된 조건은 문제의 기본 가정precondition에서 도출된 것이다. 그리고 r: =x;라는 assignment 구문은 precondition의 x≥0을 r≥0로 변경하는 효과를 갖게 된다. 다시 말하면 1번 문장의 수행결과로 r이라는 변수는 x과 같은 값을 갖게 되므로 p1에서 x가 나타나는 곳에 r의 값을 대입한 것 뿐이다. q:=0;는 precondition에 영향을 미치지 않게 된다. 나머지 조건은 Loop invaraint에서 얻은 것이다. p3는 loop invariant라고 흔히 불리는데, 자동적으로 계산하는 방법은 알려져 있지 않지만, 반복문을 몇 번 수행하든 관계없이 항상 만족되어야 하는 조건식을 표시하는 것이다. 그리고 p3와 p4는 3번과 4번 문장을 수행하기 전의 precondition과 postcondition을 나타내는데, 1번 문장의 수행결과로 p1에서 p2를 도출하는 과정과 동일하다.

직접 해보기 전에는 이해가 쉽지 않을 수 있지만, p3에 r이 나타나는 조건에 r-y라는 값을 대입하고, q에는 q+1을 대입한 후, 식을 다시 전개해서 정리해 보면 p4가 생성됨을 알 수 있다. 마지막으로 5번 문장이 수행될 때, 즉 프로그램이 종료되는 시점에는 (r>=y) 반복문의 조건이 더 이상 참이 아니므로, 다시 쓰면 r<y로 수정할 수 있고, p2 또는 p4에 r<y 조건을 추가하면 p5를 얻게 된다. 물론 이 예제는 실제 프로그램의 분석에서 어려움을 초래하는 배열이나 포인터 등은 고려하지 않은 것이고, parameter passing 과정에서 생기는 변수의 aliasing 문제 또한 다루고 있지 않지만, 테스팅이 아닌 program verification이라는 것이 어떻게 이론적으로 가능한지를 쉽게 이해할 수 있도록 도와준다.

어안렌즈가 어떤 원리로 작동하는지 또는 줌렌즈에서 이미지 stabilizer 기능이 어떻게 작동하는지에 대한 광학적인 이론을 알아야만 좋은 사진을 찍을 수 있는 것이 아니다. 어떤 상황에서 작가의 촬영의도나 구도를 어떤 렌즈가 제일 효과적으로 나타낼 수 있는지, 노출과 셔터스피드는 어떻게 선택하면 원하는 사진을 얻을 수 있는지 아는 것으로 충분하다. 마찬가지로 프로그램 증명 또한 모든 원리를 다 이해하는 것 보다는 testing의 한계를 극복하기 위한 노력으로 연구자들이 어떻게 정형검증 기법을 발전시켰는지, 어떤 도구가 제일 효과적인지를 알면 현장에서 매우 유용하게 사용할 수 있다. 마찬가지로 소프트웨어 개발자나 테스트엔지니어의 입장에서 본다면 formal verification의 기본 개념을 이해하고 SAGE, PathFinder 등의 정형검증 도구의 작동법을 이해하고 실제 현장에서 잘 사용할 수 있으면 충분하다.

6)Cornell대학의 Gries교수가 1981년에 출판한 The Science of Programming이라는 이 분야의 "고전"과도 같은 책에서 인용한 예제이다.

INTERVIEW

연세대학교 소프트웨어 및 신경망 신뢰성 연구실
김지응 교수님과의 대화

• 김지응 박사
조교수
연세대학교 컴퓨터과학과
AI Convergence 연구실
jihun.park@cnu.ac.kr
https://sites.google.com/view/jihunpark/

신진연구자 소개 I

주요 약력

2024.09~현재 연세대학교 컴퓨터과학과 조교수
2022.09~2024.08 인하대학교 컴퓨터공학과 조교수
2020.05~2022.08 Google, Research Engineer & Software Engineer
2019.06~2020.04 Yale University, Postdoctoral Associate
2019.05 Yale University 전산학과 박사 (지도교수: Zhong Shao)
2011.08 KAIST 전산학과 석사 (지도교수: 류석영)
2009.08 성균관대학교 정보통신계열 컴퓨터공학과 학사

대표 논문

[1] Jieung Kim, Ronghui Gu, and Zhong Shao, SimplMM: A Simplified and Abstract Multicore Hardware Model for Large Scale System Software Formal Verification Journal of Systems Architecture, Volume 147, Article 103046, February 2024.

[2] Wolf Honore*, Jieung Kim*, Ji-Yong Shin, and Zhong Shao, Much ADO about Failures: A Fault-Aware Model for Compositional Verification of Strongly Consistent Distributed Systems Proceedings of 2021 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'21), October 2021. (*: both equally contribute to this work)

[3] Ronghui Gu, Zhong Shao, Jieung Kim, Xiongnan (Newman) Wu, Jeremie Koenig, Vilhelm Sjoberg, Hao Chen, David Costanzo, and Tahina Ramananandro, Certified Concurrent Abstraction Layers, Proceedings of 2018 ACM SIGPLAN Conference on Programming Language Design and Implementation, June 2018.

[4] Ronghui Gu, Zhong Shao, Hao Chen, Xiongnan (Newman) Wu, Jieung Kim, Vilhelm Sjoberg, and David Costanzo, CertiKOS: An Extensible Architecture for Building Certified Concurrent OS Kernels, 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), November 2016

김지응 교수님은 2024년부터 연세대학교에서 근무 중이십니다. 주요 연구분야는 소프트웨어 정형검증으로, 2019년에 예일 대학교에서 박사학위를 받으셨습니다.

박사학위 주제는 "Modular and Compositional Development of Certified Concurrent Software Systems"입니다. 대학에 교수직으로 돌아오시기 전에는 Google의 기계학습 모델 최적화 팀에서 근무하셨습니다. 이러한 다양한 경험을 바탕으로 소프트웨어 및 기계학습 모델에 정형기법을 이용한 신뢰성 향상을 목표로 다양한 연구를 하시고 계십니다. 일례로, 삼성미래기술육성재단으로터 지원받아 "신경망 분해/통합 검증을 위한 원천 기술 연구"를 수행하시고 계십니다. 김지응 교수님에 대한 보다 자세한 약력은 말미를 참조하시기 바랍니다. 아래 문답에서 "편"은 편집자를 "김"은 김지응 교수님을 지칭합니다.

편: 예일대학교에서 CertiKOS (Certified Kit Operating System) 프로젝트에 핵심 연구원으로 참여하였는데요, 프로젝트에 대해 대략적인 설명 부탁드립니다. 특히 프로젝트에서 핵심적으로 삼은 목표가 무엇이었나요?

김: CertiKOS는 수학적 검증을 통해 커널의 안전성과 보안을 보장하는 하이퍼바이저 및 운영체제입니다. CertiKOS 프로젝트의 궁극적인 목표는 비교적 적은 비용으로 완전히 검증된 운영체제 커널을 제공하여 신뢰할 수 있는 시스템 소프트웨어의 기반을 마련하는 것입니다. 이를 위해 CertiKOS 프로젝트는 시스템 소프트웨어 측면과 정형 검증 측면에서 여러 가지 목표를 가지고 있습니다. 시스템 소프트웨어 측면에서는 확장성(extensibility), 경쟁력 있는 성능(performance), 안전성(safety), 고신뢰성(reliability)을 완벽하게 확보한 운영체제를 개발하는 것이 목표입니다. 이를 위해 연구진은 저수준 언어인 C와 x86 어셈블리를 개발 주 언어로 삼고, 동시성 제어, 페이지 테이블, 가상 머신 관리, 스케줄러, IPC 등의 여러 서비스를 모듈화하여 하이퍼바이저 및 운영체제를 설계하였습니다. 또한, 안전성과 고신뢰성을 보장하기 위해 "정형 검증(formal verification)"을 활용하였습니다. 정형 검증 측면에서의 목표는 이러한 운영체제에 대한 정형 검증을 기존 연구들(예: seL4) 대비 획기적으로 낮은 비용과 단기간에 수행하되 더 높은 신뢰성을 가지는 방법론을 개발하는 것이었습니다. 이를 위해 연구진은 레이어 구조로 검증을 모듈화하여 수행 가능한 certified concurrent abstraction layer라는 정형 검증 방법론을 제안하였습니다. 해당 도구의 내부 컴파일러로는 정형 검증된 C 컴파일러인 CompCert를 확장한 컴파일러를 활용하고 있으며, 레이어 및 레이어의 내부 정의(함수 및 커널 자료 구조들)를 추가하거나 변경하더라도 정형 검증에서 변경이 필요한 부분만 소규모로 수정하여 전체 증명을 재구성할 수 있습니다. 이로

인해 연구진은 CertiKOS의 시스템 콜에 대한 정형 명세를 제공했을 뿐만 아니라, 해당 명세가 실제 하드웨어에서 구동되는 기계어와 동일한 성질을 가짐을 엄밀히 정형 검증할 수 있었습니다.

편: 정형 검증의 비용을 낮추는 것이 CertiKOS에서의 목표 중 하나라고 말씀하셨는데요, 여기서 정형 검증은 Coq와 유사하게 대화식의(interactive) 정리증명 기반 방법을 말씀하시는 것이고, 검증 비용을 낮춘다는 것은 사용자가 보다 손쉽게 증명할 수 있도록 한다는 말씀이신가요?

김: 네, 그렇습니다. 사용자가 보다 손쉽고 빠르게 증명할 수 있도록 한다는 말입니다. 대화식 정리 증명 기반 방법들은 자동화 정형 검증 기법에 비해 더 복잡한 문제를 정의하고 증명할 수 있지만, 정형 검증을 위해 작성해야 하는 증명의 양이 매우 많다는 것이 큰 단점입니다. 예를 들어, CertiKOS의 전체 코드, 정의, 증명의 양은 약 300,000 라인에 달하며, 이 중 증명을 위한 라인 수가 상당히 많습니다. 오히려 함수별로 분할/통합 정형 검증을 수행하지 않은 기존 방법들이 더 적은 소스 코드와 증명 양을 가질 수도 있습니다. 그럼에도 불구하고 CertiKOS의 증명 유지 보수성은 기존 운영체제 정형 검증 방법론과 비교해 매우 좋은 편이었습니다. 이는 매우 복잡한 소프트웨어에 기능을 수정 및 추가할 때 전체 소프트웨어 소스 코드를 항상 살펴봐야 하는 경우와, 모듈화가 잘 되어 있어 해당 기능과 관련된 부분만을 살펴봐야 하는 경우의 차이라고 할 수 있을 것 같습니다. 또한, CertiKOS는 자주 일어나는 증명 패턴들에 대한 증명 라이브러리를 제공합니다. 이러한 방법들을 통해 CertiKOS는 대규모 소프트웨어 정형 검증을 분할하여 여러 증명 엔지니어들이 동시에 작업할 수 있게 하였을 뿐만 아니라, 해당 증명 부분에 대한 최소한의 이해만으로도 증명을 수행할 수 있도록 만들었습니다. 즉, 기존 대화식 증명 기반 정형 검증 도구들과 비교하여 증명 도구 사용자의 편의성을 매우 높였다고 볼 수 있습니다.

편: CertiKOS 프로젝트를 수행하시면서 가장 성공적이었다고 평가할 만한 것이 무엇인가요?

김: 프로젝트 팀에서 가장 성공적인 측면은 동시성을 포함한 시스템 소프트웨어를 레이어 단위로 분할하고 각각을 개별적으로 정형 검증한 후 통합하는 certified concurrent abstraction layer 도구의 개발이었습니다. 이 도구는 CertiKOS뿐만 아니라 이후 분산 시스템과 하이퍼바이저 등 다양한 시스템 소프트웨어의 정형 검증에 활용되었습니다. 해당 도구는 크게 세 부분으로 구성됩니다. 첫째, 유명한 정형 검증 C 컴파일러인 CompCert를 변형하여 함수별로 컴파일 결과의 정확성을 검증하는 컴파일러를 개발하는 것입니다. 둘째, 함수 단위로 정형 검증 대상을 분리하고 각 함수의 증명 결과를 통합할 수 있는 layer calculus를 만드는 것입니다. 이 layer calculus는 함수 간 상호작용과 동시성 객체의 상호 간섭을 추상화하고 간소화하여 증명 가능한 도구를 제공합니다. 예를 들어, 멀티 코어 환경에서 각 코어에서 실행되는 프로그램들 간의 상호작용을 추상화하고 간소화하는 방법을 제공합니다. 셋째, 정형 검증 대상 코드와 명세 사이의 정확한 상호작용을 증명하기 위한 여러 라이브러리와 도구를 제공합니다. 또한, 제가 좋아하는 CertiKOS의 두 가지 부분은 다음과 같습니다. 첫째, CertiKOS의 회전잠금(spinlock) 모듈 정형 검증입니다. CertiKOS는 ticket lock과 MCS lock 두 가지 회전 잠금 모듈을 제공합니다. 이 두 알고리즘은 비슷한 기능을 수행하지만 동일한 최종 명세를 가진다는 것을 증명하는 것은 쉬운 일이 아니었습니다. 그러나 certified concurrent abstraction layer를 통해 두 회전잠금 모듈이 동일한 정형 명세를 가지고 있음을 증명하였고, 필요할 경우 증명 및 구현을 변경하지 않고 자유롭게 교체 사용할 수 있음을 보였습니다. 둘째, CompCert의 머신 모델을 확장하여 멀티 코어 환경의 머신 모델을 정의하는 작업입니다. CertiKOS는 멀티 코어 환경에서 작동하도록 설계되었지만, CompCert의 기존 머신 모델은 단일 코어 하드웨어를 가정하고 있었습니다. 우리는 CompCert의 머신 모델을 확장하여 여러 코어에서 실행되는 프로그램을 추상화하고 증명할 수 있도록 했습니다. 이는 쉽지 않은 작업이었지만, 간단하고 일반화 가능한 멀티 코어 머신 모델을 설계하고 이를 통해 필요한 경우에만 x86의 레지스터와 명령어에 대한 증명을 수행할 수 있었습니다.

편: CertiKOS의 정형검증을 간략히 얘기하면 두 단계 접근법으로 보입니다. (1) CertiKOS를 구성하는 각 함수의 컴파일 정확성을 확장된 CompCert로 보장하기와 (2) 함수간의 상호 작용의 무결점성을 layer calculus로 증명하기. 제가 이해한 것이 대강 맞나요? 그리고 layer calculus는 동시성 개념을 추가한 lambda calculus 정도로 이해하면 무리가 없을까요? 아마 제가 두리뭉실하게 이해하는 것일 테지만 되도록 많은 독자분들이 이해하시는 데에 도움이 되고자 하는 마음에서 비전문가로서 질문을 드려 봅니다.

김: 두 단계의 접근법이 적절합니다. 전체 소프트웨어를 함수 단위로 분할한 후, (1) 각 함수 단위의 컴파일 올바름을 확장된 CompCert로 보장하고, (2) 각 함수 단위로 검증된 결과를 layer calculus로 합치는 것입니다. Layer calculus는 lambda calculus보다 간단하게 이해할 수 있으며, horizontal composition(수평적 통합)과 vertical composition(수직적 통합) 두 가지가 핵심 구성 요소입니다. 수평적 통합은 동일한 데이터 구조를 사용하는 두 함수를 개별적으로 증명한 후 이를 합치는 것으로, 이는 tuple을 만드는 것과 비슷하다고 볼 수 있습니다. 예를 들어, 2 + 2 = 4와 3 + 5 = 8을 따로 증명한 후 통합하면 (2 + 2, 3 + 5) = (4, 8) 을 증명한 것과 동일합니다. 수직적 통합은 합성함수의 개념입니다. 예를 들어, 1 + 3 = 4를 증명하고 4 + 5 = 9를 증명한 후 이를 통합하면 (1 + 3) + 5 = 9를 증명한 것과 동일합니다. 물론, 실제 소프트웨어 정형 검증에서는 이 과정이 이렇게 간단하지 않습니다. 메모리 사용 및 복잡한 함수 사용 방법 등으로 인해 해결해야 할 문제가 많이 있습니다. 그럼에도 불구하고 C로 작성된 모든 프로그램과 함수에 대해 CertiKOS 기법을 활용한 분할/통합 정형 검증을 적용하는 것은 아직 불가능합니다. 사실, 매우 제한적인 프로그램에 대해서만 적용 가능하다고 보는 것이 더 적절하다고 생각합니다.

편: 거의 모든 프로젝트가 남은 숙제를 남기는데요, CertiKOS

프로젝트를 수행하시면서 이 문제는 앞으로 교수님 혹은 동료 연구자들이 꼭 해결해야겠다고 생각하시는 점 있으시면 말씀 부탁 드립니다.

김: CertiKOS 프로젝트는 그 규모에 걸맞게 해결해야 할 다양한 과제들이 남아 있습니다. 특히 제 지도 교수이자 CertiKOS 프로젝트 책임 교수인 Yale 대학교의 Zhong Shao 교수님 연구 그룹에서 진행 중인 몇 가지 중요한 문제들을 동료 연구자들과 함께 해결하고 싶습니다. 그 중 하나는 certified concurrent abstraction layer에서 검증하기 어려운 다양한 프로그래밍 기법들을 다루는 도구들을 활용하여 상용 프로그램의 일부를 검증하고 실제 서비스와 함께 배포하는 것입니다. Certified concurrent abstraction layer는 기존의 정형 검증 사례들과 비교하여 동시성을 제공하면서도 증명 비용을 크게 줄일 수 있는 장점이 있습니다. 그러나 이 도구를 활용하여 정형 검증을 하기 위해서는 소프트웨어를 해당 도구에서 사용 가능하게 재작성하는 과정이 필요합니다. 이는 때로는 시스템 소프트웨어의 특정 프로그래밍 컨벤션을 사용할 수 없는 경우도 포함됩니다. 또한, 수정된 코드가 시스템 소프트웨어의 다른 모듈들과 함께 연결되어야 하는 경우도 있어, 추가적인 수정이 필요할 수 있습니다. 따라서 저는 검증 대상 프로그램에서 사용되어야 하는 다양한 프로그래밍 기법들을 활용하고, 이를 증명 가능한 정형 검증 도구를 통해 대규모 범용 시스템 소프트웨어의 핵심 부분을 증명하고 해당 소프트웨어의 다른 모듈들과 자연스럽게 연결되도록 하는 일에 관심이 있습니다. 현재는 서울대학교 허충길 교수님 연구실과 협업하며 이러한 목표를 추구하고 있습니다. 또한, 증명 비용을 줄이기 위한 증명 자동화 연구, CertiKOS 프로젝트의 정형 명세 안전성을 확인하는 테스팅 기법 개발, CompCert에 포함되지 않은 다양한 하드웨어 기능들을 정형 검증과 함께 추가하는 방법, 시스템 소프트웨어 엔지니어들이 사용하기 편한 정형 명세 작성 방법 개발에도 관심이 있습니다.

편: 교수님과 동료 분들의 노력으로 증명 용이성과 사용 편리성이 모두 좋은 프로그래밍 언어 및 개발 방법론이 나오길 기대하겠습니다.

편: 다음 질문으로 넘어가겠습니다. Google에서의 경력이 사뭇 특이합니다. 정형기법을 하시다가 기계학습 쪽 팀에 합류하시게 된 계기가 있을까요?

김: 구글에서 가장 유명한 프레임워크 중 하나는 텐서플로우입니다. 텐서플로우의 일부를 직접 다룰 수 있다는 점이 제가 기계 학습 팀에 합류하게 된 중요한 계기가 되었습니다. 또한, 기계 학습에 대한 지식을 습득하고, 인공 신경망에 적용 가능한 다양한 구조적 및 정형적 방법론들을 탐구하고자 하는 열망도 있었습니다. 물론 구글에 입사를 하자마자 이런 생각을 가지고 있었던 것은 아닙니다. 제가 구글에서 처음 소속된 팀은 Google Research의 Cerebra (Personal AI) 팀이었습니다. 해당 팀에서 처음 맡았던 일은 안드로이드 생태계에 추가될 하이퍼바이저(pKVM)의 정형 검증 기술을 조사하고 습득하여 적용하는 것이었습니다. 하지만 프로젝트 진행이 더딘 점도 있었고, 기계 학습을 위한 하드웨어용 컴파일러 및 TinyML을 위한 하드웨어 디자인 등 여러 프로젝트를 접하다 보니 기계 학습 관련 일을 제대로 체험해 보는 것이 미래 연구 및 개발에 매우 중요하다는 생각이 들었습니다. 그래서 상황과 기회가 주어졌을 때 과감히 기계 학습 팀(Model Optimization Team)에 합류하게 되었습니다. 해당 팀에 속해 있던 기간은 길지 않았지만, 구글에 재직하는 동안 많은 내용을 습득할 수 있었고, 그때의 경험이 현재 연구 주제에 많은 영향을 미쳤습니다. 따라서 이 선택은 매우 좋은 결정이었다고 생각합니다.

편: 삼성미래기술육성재단으로터 “신경망 분해/통합 검증을 위한 원천 기술 연구”에 대한 지원을 받고 계신데요, 연구에 대해 설명을 부탁드립니다. 연구제목을 읽어보면 이전의 정형기법 내지는 프로그램 분석 기법을 기계학습에 적용하시는 것 같기도 한데요, 어떠한가요? 진행 중이신 연구이기는 하나 조금 더 구체적인 말씀을 해 주실 수 있을까요?

김:신경망 검증 연구들은 2010년대 중반부터 시작되어 이미 많이 진행되었지만, 관련 연구들을 조사하면서 검증 모듈화 및 검증의 목표가 되는 안전성 명세를 서술하는 법에서 향상 가능한 부분들이 있다고 느꼈습니다. 많은 연구자들이 신경망의 가장 큰 특성을 블랙박스(black-box)처럼 내부 구조와 결정을 이용하기 힘든 점이라고 이야기합니다. 신경망 정형 검증 연구들 중 널리 활용되는 선행 연구들 또한 이러한 특성 때문에 기본적인 입력값과 출력값 사이의 상관관계를 검증 대상으로 삼고 있습니다. 저희 연구 프로젝트는 신경망이 대체로 블랙박스 특성을 가지고 있더라도 구조적이고 정형적인 방법론을 통해 접근할 수 있는 영역이 있을 것이라는 질문에서 시작되었습니다. 다행히도, 실제 사례들을 찾아보니 이러한 접근 방법이 점점 늘어나고 있음을 확인할 수 있었습니다. 특히, 이미 몇몇 선행 연구에서는 구조적인 내용을 활용하여 신경망을 특정 목적으로 분해하는 연구들이 존재하며, 증명 재사용을 위한 연구들도 최근 저명한 학회들에서 발표된 바 있습니다. 저는 이러한 최근 구조적 접근 방법들을 저의 정형 검증 연구 경험과 결합하여 사용할 수 있는 방법을 연구 개발 중입니다. 조만간 이와 관련된 성과를 공유하고 더욱 자세하게 소개 드릴 수 있도록 노력 하겠습니다.

편: CertiKOS와 신경망 검증 등 현실 세계에서 실제로 널리 사용하고 있는 SW에 대해 정형기법을 적용해오고 계신데요, 일반적으로 정형기법의 문제점으로 지적되고 있는 확장성에 정면으로 도전하시고 있는 셈입니다. 지금까지 도전이 성공한 이유와 앞으로 해결되어야할 문제에 대해 말씀 부탁드립니다.

김: 연구를 진행하면서 몇 가지 중요한 교훈을 깨닫게 되었습니다. 요약하자면, 명확하지 않은 영역에 대해 두려움을 갖지 말고, 이를 해결해 가는 과정을 즐기라는 것입니다. 이를 조금 더 자세히 서술하면 세 가지로 나눌 수 있습니다. 첫 번째는, 너무 많은 문제를 한꺼번에 해결하려고 하지 말고 집중할 부분을 선택하는 것이 중요하다는 것입니다. 정형 검증 연구는 대부분 신뢰 컴퓨팅 기반(Trusted Computing Base, TCB)을 가지고 있습니다. 따라서 정형 검증 연구에서 가장 중요한 요소 중 하나는 어떤 부분을 검증하여 안전성을 보장할지,

그리고 어떤 부분을 TCB로 설정할지를 결정하는 것입니다. 예를 들어, 분산 시스템의 정형 검증에서는 TCP/IP 네트워크 통신 모듈 전체를 TCB로 설정할 수도 있습니다. 아니면 통신 모듈 전체가 아닌 실제 패킷을 네트워크로 전송하고 받는 핵심 부분들만 TCB로 설정할 수도 있습니다. 또한, 소스코드 전체를 TCB로 설정하거나 명세의 안전성만을 검증할 수도 있습니다. 이렇게 어떤 부분에 집중할지를 선택하는 것이 가장 중요한 문제라고 생각됩니다. 두 번째는 너무 먼 미래까지 생각하지 말고, 프로젝트 진행 중 맞닥뜨리게 되는 문제들에 좌절하지 말라는 것입니다. 대규모 정형 검증을 진행하다 보면 예상치 못한 문제들이 발생할 수 있습니다. 이러한 문제가 발생했을 때 이미 수행한 증명 및 코드들이 수만 줄 이상이 될 수도 있습니다. 이럴 때마다 다시 원점으로 돌아가서 모든 문제를 아름답게 해결할 수 있는 정형 검증 기법 및 이론을 찾거나 개발하는 것은 현실적이지 않습니다. 따라서 이러한 문제들이 발생하는 것이 대규모 정형 검증 연구의 당연한 일이라는 것을 인지하는 것이 중요합니다. 또한, 문제가 발생했을 때 해당 문제를 꼭 현재 프로젝트 기간 동안 해결해야 하는지, 아니면 다음 프로젝트로 미뤄야 하는지를 결정하는 것이 매우 중요하다는 것을 깨달았습니다. 그리고 이는 현재 맞닥뜨리고 있는 문제들을 복잡하게라도 풀 수 있는지 아니면 아예 불가능한지를 판단하고 선택하는 능력이 중요하다고 생각합니다. 세 번째는, 정형 검증의 대상을 100% 이해하고 시작하려고 하지 말라는 것입니다. 물론 정형 검증의 대상을 잘 알고 있고 검증에 필요한 여러 가지 이론들을 잘 알고 있다면 좋습니다. 하지만 정형 검증 연구를 시작하는 입장에서, 그리고 경험이 있는 입장에서도 이를 다 갖추는 것은 쉽지 않습니다. 모든 것을 갖추고 연구를 시작하려다 보면 몇 개월, 길게는 수년을 관련 연구 내용 습득에만 써야 할 수도 있습니다. 이럴 때, 결과물 없이 연구를 꾸준히 진행할 수 있는 경우가 얼마나 많을지 의문입니다. 경험상, 어느 정도 정형 검증 대상에 대해 이해하고 이를 위한 검증 기술을 알게 되면, 정형 검증 연구를 진행하면서 검증 대상과 이를 검증하기 위한 검증 이론 및 기술에 대해 점차 깊이 알게 된다는 것입니다. 물론 노력이 필요하지만요. 따라서 처음부터 모든 필요한 지식을 갖추고 연구를 시작하려고 하지 말라는 조언을 드리고 싶습니다. 다만, 검증 대상 및 검증 이론에 대한 지식 습득을 멈추지 말라는 조언도 함께 드리고 싶습니다. 적고 보니, 대규모 정형 검증을 수행해 나가는 과정에서 필요한 요소들은 결국 연구를 즐기기 위해 갖추어야 하는 요소들과 동일하다는 생각이 드네요. 너무 뻔한 이야기를 한 건 아닌가 싶기도 합니다.

편: 통찰력이 묻어나는 답변 주셔서 감사합니다. 일종의 애자일(Agile) 정형검증 방법론을 제시한 것 같습니다.

편: 끝으로 소식지 독자들에게 인사말 부탁 드립니다.

김: 소프트웨어공학 소사이어티의 구성원 여러분께 CertiKOS와 인공 신경망 검증에 관한 저의 경험과 생각을 공유할 수 있는 기회를 주셔서 감사합니다. 앞으로도 깊이 있는 연구와 다양한 학회 활동을 통해 더욱 재미있는 이야기를 전해드릴 수 있도록 노력하겠습니다. 또한, 대규모 정형 검증 연구의 핵심은 협업이라고 생각합니다. 소프트웨어공학 소사이어티의 구성원 중 비슷한 관심사를 가진 분들과 함께 즐겁게 연구할 수 있는 기회가 생기기를 기대하며, 이를 통해 다양한 대규모 정형 검증 연구를 수행할 수 있기를 희망합니다.

공부중인 연구실 학생들

Max Planck Institute 연구원 이성민 박사님과의 대화

• 이성민 박사
Max Planck Institute 연구원

신진연구자 소개 II

주요 약력

[Sep. 2022 - Present] Postdoctoral Researcher, Max Planck Institute for Security and Privacy

[Sep. 2016 - Aug. 2022] Doctor of Philosophy, School of Computing, KAIST

[Feb. 2012 - Aug. 2016] Bachelor of Science, School of Computing & Bachelor of Science, Department of Mathematical Sciences, KAIST

주요 연구분야

-Statistical Program Analysis, Software Testing, Causal Inference, Software Engineering

대표 논문

[1] Seongmin Lee, Shreyas Minocha, and Marcel Böhme. Accounting for missing events in statistical information leakage analysis, In Proceedings of the IEEE/ACM 47th International Conference on Software Engineering, ICSE '25

[2] Seongmin Lee, Dave W. Binkley, Robert Feldt, Nicolas Gold, and Shin Yoo. Causal program dependence analysis. Science of Computer Programming, 2024

[3] *Danushka Liyanage, *Seongmin Lee, Chakkrit Tantithamthavorn, and Marcel Böhme. Extrapolating coverage rate in greybox fuzzing. In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, ICSE '24 (*Co-first authors with equal contribution)

[4] Seongmin Lee and Marcel Böhme. Statistical reachability analysis. In Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023

[5] Seongmin Lee, David Binkley, Robert Feldt, Nicolas Gold, and Shin Yoo. Observation-based approximate dependency modeling and its use for program slicing. Journal of Systems and Software, 2021

[6] Seongmin Lee, David Binkley, Nicolas Gold, Syed Islam, Jens Krinke, and Shin Yoo. Evaluating lexical approximation of program dependence. Journal of Systems and Software, 2020

이성민 박사님은 2022년 카이스트에서 유신 교수님 지도로 박사학위를 취득 후, 독일 Max Planck Institute(MPI)의 연구원으로 근무 중이십니다.

이성민 박사님의 박사 학위 주제 "Statistical Program Dependence Approximation"에서 유추할 수 있다시피 이박사님은 통계적 프로그램 분석의 전문가이십니다. 졸업 후 MPI에서는 Marcel Böhme 박사님의 그룹에서 통계적 프로그램 분석을 활용한 퍼징 연구를 진행하고 계십니다. 이성민 박사님은 이번 정형기법 특집호에서 가장 눈에 띄는 분이 아닐까 싶습니다. 일반적으로는 통계적 프로그램 분석을 정형기법과 다른 영역으로 분류하기도 하지만, 수학적 기법에 근간한 방법론이라는 측면에서 소식지의 주제에서 크게 벗어나지 않는 것 같습니다. 오히려, 다양한 종류의 수학적 방법을 다룸으로써 소식지가 더욱 풍성해질 것으로 기대합니다. 이성민 박사님에 대한 보다 자세한 약력은 말미를 참조하시기 바랍니다. 아래 문답에서 "편"은 편집자를 "이"는 이성민 박사님을 지칭합니다.

편: 이번호가 정형기법 특집입니다. 일반적으로 논리나 오토마타를 기반으로 하는 정형기법과 통계적 프로그램 분석의 차이에 대해 설명 부탁드립니다.

이: 안녕하세요, 반갑습니다. :) 두 기법은 같은 문제를 푸는 서로 다른 방법이라고 생각합니다. 그리고 두 기법의 가장 큰 차이점은 연역적인 과정을 통해 문제를 해결하는지 아니면 경험적인 과정을 통해 문제를 해결하는지에 있어요.

예를들면, 워터파크에 워터슬라이드가 있다고 생각해보세요. 워터슬라이드에는 중간 중간에 갈라지는 길이 있어서 도착할 수 있는 곳이 여러 군데 있다고 해볼께요. 그리고 우리는 여러 출구중에 어디에 사람들이 도착할 수 있는지 궁금해 한다고 해봅시다. 정형기법은 워터슬라이드 구조, 즉, 갈라지는 부분들을 하나하나 확인해, 어디에 도착할 수 있는지 계산해내는 방법이라고 생각하시면 됩니다. 반면에 경험적/통계적 방법은 사람들이 실제로 워터슬라이드를 타는 것을 관찰해서, 실제로 어디에 도착하는지를 확인하는 방법이라고 생각하시면 됩니다.

이 두 방법은 같은 문제를 푸는데 있어서 서로 다른 방법을 사용하지만, 둘 다 장단점이 있어요. 정형기법은 계산이 잘 끝났다면 정확한 답을 줄 수 있지만, (워터슬라이드의 구조가 너무 어려워) 계산이 너무 복잡하거나 불가능할 때가 있어요. 반면에 통계적 방법은 실제로 사람들이 어디에 도착하는지를 보기 때문에 워터슬라이드의 구조가 얼마나 복잡하든지 상관없이 결과를 얻을 수 있지만, 샘플의 숫자가 부족할 경우 그 결과가 정확하지 않을 수도 있습니다.

경험적/통계적인 방법이 유망하다고 생각하는 부분이 바로 이 부분입니다. 제가 워터슬라이드라고 비유했던 것이 사실은 분석하고 싶은 프로그램/

소프트웨어인데요, 오래전과는 달리 현대의 프로그램/소프트웨어들는 너무나도 크고 복잡하여 계산적인 방법으로 정확하게 이들의 행동을 분석하기는 너무나도 어렵습니다. 하지만 (마치 사람들이 직접 워터슬라이드를 탄 결과처럼) 프로그램의 실행 결과를 관찰해 분석하는 경험적/통계적 방법은 프로그램이 얼마나 복잡한지에 관계없이 분석을 할 수 있기에 현대 소프트웨어를 분석하는데 유용합니다.

편: 정형기법 분야에도 확률적 모델체킹과 같이 확률 및 통계를 이용하는 방법들이 있는데요, 박사님이 하고 계시는 연구와는 지향점이나 방법론 등에서 어떠한 차이가 있을까요?

이: 확률적 모델체킹과 통계적 프로그램 분석, 두 방법의 가장 큰 차이점은 "확률을 연역적으로 '계산'하는가," 아니면 "관측(경험)을 통해 '추정'하는가" 입니다. 어쩌다보니 계속해서 워터슬라이드 예시를 사용할 수 있겠네요. 8-) 워터슬라이드에 두 출구 A와 B가 있다고 했을 때, 100명의 사람들이 워터슬라이드를 탔더니, A 출구에는 70명의 사람이, B 출구에는 30명의 사람이 나왔다면, 워터슬라이드를 탔을 때 각각 A와 B출구로 나올 확률을 0.7 그리고 0.3이라고 추측을 할 수 있겠죠. 이와 같이 경험적 데이터에서 확률을 추측하는 것이 통계적 방법입니다. 말씀하신 것 처럼 정형기법 분야, 즉, 계산적인 방법으로도 확률을 '계산'할 수 있습니다. 워터슬라이드를 예로 들면, 갈래가 나왔을 때 워터슬라이드의 각도를 분석한다던지 말이죠. 그렇게 하면, A출구로 나올 확률이 사실 정확히는 0.702xxx 이고 B출구로 나올 확률은 0.297xxx라는 것을 알아 낼 수도 있겠죠. :D

확률적인 분석은 단순히 예/아니오라는 결과를 내어 놓는 것 보다 훨씬 많은 정보를 줄 수 있기 때문에 유용합니다. 이렇게 확률적인 결과를 줄 수 있는 서로 다른 방법이 통계적 분석법과 확률적 정형기법이라고 할 수 있겠네요.

편: 통계적 프로그램 분석을 이용한 퍼징 연구를 최근에 활발히 진행해 오고 계신데요, 일반적인 퍼징과의 차이점에 대해서 설명 부탁드립니다.

이: 퍼징은 프로그램의 취약점을 찾아내는 방법 중 '경험적'인 방법의 끝판왕이라고 할 수 있어요. 퍼징은 테스트 하고자 하는 프로그램에 수 없이 많은 수의 인풋을 집어넣고 실행시켜 실제로 문제가 발생하는지를 확인하는 방법입니다. 퍼징은 이전에 언급한 '경험적'방법의 확장성의 이점으로 인해 복잡한 현대 소프트웨어 테스팅에서 매우 혁신적인 결과를 가져왔어요.

하지만, 단순히 경험적인 정보만을 통해서 프로그램을 테스트하는 퍼징은 근본적인 한계점이 존재하는데요, 그건 바로 프로그램의 모든 가능한 상황을 테스트할 수 없기 때문에 존재하는 취약점을 놓칠 수 있다는 것, 즉, 취약점의 부제를 보장(guarantee)할 수 없다는 점입니다. 단 하나의 놓친 취약점으로 인해 치명적인 보안 문제가 발생할 수 있기 때문에, 보장이 없다는 것은 프로그램의 안전성을 검증하는데 있어서 큰 단점입니다. (정형기법의 경우, 취약점의 부제를 형식적으로 보장 (formal guarantee) 할 수 있습니다.)

통계적 프로그램 분석을 사용한 퍼징의 가장 큰 장점은, '경험적' 데이터에 통계 기법을 사용하여 통계적 보장(statistical guarantee)을 제공 한다는 것입니다. 강력한 통계 이론은 경험적 데이터로부터 내가 얼마나 현재 결과에 대한 확신이 있는지, 즉, 내가 얼마만큼 놓치고 있는 부분이 있는지를 알려줄 수 있어요. 예를 들면, 통계적 보장은 "내가 이 프로그램의 가능한 행동의 99%를 테스트 하였다"와 같은 정보를 제공할 수 있습니다. 이는 퍼징의 장점인 확장성은 유지하면서 통계적 보장을 제공하여, 퍼징의 근본적인 한계점을 해결할 수 있게 해줍니다.

통계적 프로그램 분석은, 더 나아가, 퍼징의 경과를 예측할 수 있게 해줍니다. 기존에 퍼징은 아직 테스트 하지 못한 부분에 대해 아는 바가 없기 때문에, 얼마나 오래 시간과 자원을 쏟아야하는지 아무도 아는 바가 없었어요. 하지만 통계적 분석으로 테스트하지 못한 부분을 예측할 수 있게 되면, '현재 퍼징이 새로운 취약점을 발견하려면 얼마나 더 많은 프로그램 실행이 필요한가', '앞으로 퍼징을 12시간 더 수행하면 프로그램의 행동의 몇 퍼센트를 더 확인할 수 있는가'와 같은 질문에 대한 답을 추측할 수 있게 됩니다. 이는 퍼징을 효율적으로 사용하는데에 큰 도움이 되어요. :)

편: 통계적 프로그램 분석을 이용한 퍼징 기술과 정형기법 기술을 접목하는 연구에 대한 소개 및 의견 부탁 드립니다.

이: 좋은 질문이네요! 통계적 프로그램 분석과 정형기법은 장단점이 명확하게 나뉘어 있어요. 통계적 프로그램 분석은 프로그램의 구조를 모르는 상태에서도 프로그램의 행동을 분석할 수 있지만, 그 때문에 정형기법에 비해서 정확성이 떨어질 수 있어요. 반면에 정형기법은 프로그램의 구조를 알고 있을 때, 그 구조에 대한 정확한 분석을 제공할 수 있지만, 그 구조를 알아야만 하고 계산이 가능할 만큼 덜 복잡할 때만 사용할 수 있어요.

따라서 접목할 수 있는 대표적인 방법 중 하나는, 정형기법을 일반적으로 사용하여 프로그램을 분석하되, 정형기법이 불가능한 경우에 경험적인 데이터를 활용하여 부족한 확장성을 보완하는 방법입니다. 이미 여러 연구들이 경험적 데이터를 추가하는 것이 효과적이라는 것을 보여주고 있어요. 다만, 기존처럼 경험적 데이터를 사용하기만 하면 보장(guarantee)을 잃게 됩니다. 이에 반해 '통계적' 프로그램 분석만이 가지는 이점은 '통계적 보장'을 제공한다는 것이죠. 즉, 경험적 데이터를 사용함과 동시에 보장성도 유지할 수 있답니다.

위의 방법은 이미 여러 연구에서 제안되었기도 해 서, 제가 요새 관심을 가지고 있는 접목 방법은 다른 방법인데요, 바로 '돌연변이 실행 정보를 사용한 통계적 프로그램 분석 및 정형기법을 활용한 검증'입니다. 통계적 프로그램 분석이 잘 활용되려면, 프로그램의 많은 행동을 관찰할 수 있는 다양한 실행 정보가 있어야 해요. 하지만, 프로그램을 마구잡이로 실행시키게 되면, 다양한 행동을 관찰하기 쉽지가 않아요. 이 때, 프로그램 실행에 의도적으로 '어요. 이 때, 정형기법을 사용하여, 돌연변이 실행으로 얻은 정보 중, 실제로 발생할 수 있는 정보만을 걸러내어 통계적 분석에 활용하면, 다양한 실행 정보를 사용할 수 있으면서도, 옳은 결과만을 통계적 프로그램 분석으로 얻어 낼 수 있을거라 생각합니다. 재미있을 것 같지 않나요? :D

편: 네. 무척 흥미롭네요! 그런데 정형기법을 사용해서 실제로는 발생할 수 없는 정보를 걸러낼 수 있다는 주장이 다소 잘 이해가 안 됩니다. 조금 더 자세히 설명해주실 수 있을까요?

이: 돌연변이를 일으키는 방법'을 사용하면, 기존에 보지 못했던 새로운 행동을 관찰할 수 있고, 이를 통계적 분석에 활용할 수 있게 됩니다. 하지만, 이 때 발생하는 문제는, 돌연변이 실행이 실제 프로그램에서는 발생할 수 없는 거짓 정보일 수 있다는 점입니다. 그럴 경우, 통계적 프로그램 분석의 결과가 부정확할 수 있습니다.

편: 요즘 세상에 AI 얘기를 안 할 수가 없을 것 같습니다. 특히, 통계적 분석은 논리기반 기술에 비하여 딥러닝 기술과 사뭇 유사한 면들이 많습니다. 최근 거대언어모델을 이용한 긍정적인 퍼징 연구도 학계에 발표되고 있는데요, 통계적 프로그램 분석을 딥러닝이 대체할 가능성에 대해서는 어떻게 생각하시나요? 그리고, 통계적 프로그램 분석이 현재 혹은 미래의 딥러닝 기술과 비교하여 가지는 우위성이 무엇인가요?

이: 저는 통계적 방법이 딥러닝 기술의 대체하거나 혹은 우위에 있는 방법이라고 생각하지 않아요. 오히려 두 기술은 서로 보완적인 관계에 있다고 생각합니다. 딥러닝은 근본적으로 주어진 데이터로부터 패턴을 학습하여 실상황에서 예측을 하는 방법입니다. 실상황에서 문제 없이 예측을 잘 해내기 위해선 학습 과정에서 주어진 데이터가 최대한 많은 경우를 포함하도록 학습 데이터를 샘플해야 하죠. 이 과정에서 '내가 학습 데이터를에 얼마나 많은 경우를 포함하였는가', '놓친 부분은 얼마나 되는가', '충분한 학습 데이터를 구성하려면 얼마나 더 많은 데이터를 샘플해야하는가'와 같은 질문은 효율적으로, 효과적으로 학습데이터를 구성하는데 있어서 중요한 문제입니다. 바로 이러한 질문에 대한 답을 제공하는 것이 통계적 분석이라고 생각합니다. 마치 퍼징에서 통계적 분석이 도와줬듯이 말이죠.

반대로 통계적 분석도 마찬가지로 데이터에 기반한 방법입니다. 이전 질문에서도 얘기했듯이, 통계적 분석이 좋은 결과를 내어놓기 위해서는 다양한 데이터를 필요로 하죠. 딥러닝을 활용해 통계적 분석에 사용되는 데이터를 풍부하게 만드는 이미 여러 다른 분야에서 연구되고 있습니다. 마찬가지로, 통계적 '프로그램' 분석에서도 딥러닝이 프로그램에 대한 경험적 데이터를 만들어내는데에 많은 도움이 될 것이라 생각해요.

편: MPI와 연구소 소재지인 Bochum에 대해서 소개 부탁드립니다. 한국의 연구소 혹은 대학과 비교해 주시면 흥미로울 것 같습니다. 또한 Bochum은 작은 도시로 알고 있는데요, 대도시에 연구 기관이 집중되어 있는 한국과 비교해서 어떠한 장단점이 있나요?

이: Bochum은 말씀하신대로 독일에 있는 작은 도시에요. 주변에 작은 강과 푸릇푸릇한 숲과 언덕이 있는 평화로운 곳임과 동시에 중심가엔 여러 펍과 식당이 있고 크리스마스 마켓, 뮤직 페스티벌등 많은 축제가 벌어지기도 하는 재미있는 도시입니다 ;)

독일의 대표적인 연구기관인 Max Planck Institute는 분야에 따라 많은 브랜치들이 있는데요, 그 중 제가 연구하고 있는 Security and Privacy 브랜치가 바로 Bochum에 위치하고 있습니다. 한국처럼 분야를 가리지 않고 대도시에 연구 기관이 집중되어 있는 것과는 다르게, 독일은 종종 한 위치가 어떤 한 분야에 특화되어 집중되어 있는 경우가 왕왕 있습니다. 그 중 Bochum은 보안으로 유명한 곳이에요. 제가 있는 MPI-SP, 보안으로 유명한 Ruhr university, 그리고 MPI-SP, Ruhr university, 뮌헨 공대, 베를린 공대의 여러 연구 팀이 소속되어 있는 연구 hub 인 CaSa 등 여러 보안 연구 기관들이 모여있고, 또 관련한 여러 start-up도 활발하게 활동하고 있는 곳이 바로 여기 Bochum입니다. 따라서 자주 보안 관련 학회나 세미나가 열리기도 하고, 다양한 연구자들과의 교류가 많아 연구하기에도 매우 흥미롭고 좋은 환경이라고 생각해요.

편: 끝으로 소식지 독자들에게 인사말 부탁 드립니다.

이: 네, 안녕하세요, 독자님들. 긴 글을 읽어주셔서 감사합니다. 나름대로 최대한 제 연구에 대해서 쉽게 설명해보려고 노력했는데, 잘 전달이 되었는지 모르겠네요. 혹시나 흥미롭거나 궁금해지신 부분이 있다면 언제든 주저하지 말고 연락주세요. 새로운 질문은 언제나 환영입니다! 이 글을 읽는 분들은 소프트웨어 공학에 관심이 있으신 분들일테니 미래에 어디선가 뵙게 될 수 있으면 좋겠네요! 끝으로 소식지에 한 부분을 차지할 수 있게 invite 해주신 editor 이주용 교수님께 감사의 말씀드리며 이만 마치겠습니다. 감사합니다. :)

Startseite_Luftaufnahme

국내외 학술행사 소개 I

CONFERENCES

FSE 2024 참가 후기

경북대학교 김요엘 박사과정생과의 대화: FSE 2024에서의 국제무대 데뷔를 기념하며

■ 김 요 엘
(경북대학교 컴퓨터학부 박사과정)

올해(2024년) FSE(Foundations of Software Engineering)가 신선한 장소에서 열렸지요. 바로 브라질의 Porto de Galinhas에서 열렸습니다. (아래 지도 참조)

한국에서 이 먼 곳까지 몇 분이 다녀오셨는데요, 그 중에 경북대학교 김요엘 학생(지도교수: 최윤자 교수님)이 본인의 국제무대 데뷔를 하고 왔습니다. 바로 아래의 정규 논문을 발표하고 왔지요.

PBE-Based Selective Abstraction and Refinement for Efficient Property Falsification of Embedded Software, Yoel Kim and Yunja Choi

이번 소식지에 데뷔 소감 및 FSE 후기 작성을 김요엘 학생에게 부탁을 했는데요, 감사하게도 흔쾌히 승낙해 주었습니다. 그럼 문답 시작하겠습니다. 아래 문답에서 "편"은 편집자(이주용)를 "김"은 김요엘 박사과정생을 지칭합니다.

편: 축하합니다! 본인의 첫번째 국제 메이저 학회 논문입니다. 논문 채택에 대한 소감 한 마디 부탁 합니다.

김: 감사합니다! 석사과정 1년 차 때 시작하여 박사과정 1년 차에 논문을 제출하기까지, 2년에 걸친 연구가 결실을 맺어 정말 기쁩니다. 사실, 박사과정 진학을 고민하던 시기에는 연구가 잘 진전되지 않아 걱정이 많았습니다. 하지만 포기하지 않고 연구에 매진한 결과, 제 선택이 틀리지 않았음을 확인할 수 있었습니다. 물론, 논문이 채택되는 과정도 순탄치 않았습니다. 논문을 제출한 후, 지도 교수님의 지원으로 작년 샌프란시스코에서 열린 FSE 2023에 참여할 기회를 얻었는데, 학회 마지막 날 저녁식사 도중 리뷰 결과를 받았습니다. 당시 결과는 Accept 1개, Weak reject 2개로, 리뷰어의 코멘트를 읽으며 착잡한 마음에 술을 시켰던 기억이 생생합니다. 한국으로 돌아온 후, 지도 교수님과 함께 리뷰어의 질문 하나하나에 체계적으로 답변을 준비하며 반박 편지(rebuttal letter)를 작성했습니다. 다행히도, 저희의 노력이 통했는지 리뷰어 한 분이 의견을 Weak accept으로 변경해주셨고, 결국 논문이 최종적으로 채택되었습니다. 이 과정을 되돌아보니, 논문 한 편을 발표하는 것이 얼마나 어려운 일인지 다시금 깨닫게 되었습니다.

편: 학회 장소까지 먼 길이었지요? 어떠한 경로로 갔고, 얼마나 걸렸나요?

김: 네, 장장 36시간에 걸쳐 세 번의 비행 끝에 겨우 도착하였습니다. 첫 번째 비행은 한국에서 독일 프랑크푸르트로, 두 번째는 프랑크푸르트에서 브라질 상파울루로, 마지막으로 상파울루에서 학회 장소와 인접한 도시인 헤시피로 향했습니다. 이와 관련된 재미있는 일화가 있는데, 브라질 남부의 상파울루를 거쳐, 북부 도시인 헤시피에 도착한 뒤, 거기서 학회 장소에 가기 위해 또 남쪽으로 이동하는 여정을 보시고는, 카이스트 허기홍 교수님께서 마치 저희가 이진 탐색(binary search)을 하고있는 것 같다고 말씀하셨습니다. 심지어

프랑크푸르트에서 상파울루로 가는 비행경로에 헤시피가 정확히 겹치기도 했습니다. 그때는 정말 낙하산을 타고 내리고 싶은 마음이 들 정도였습니다.

편: 제가 학회에서 처음 발표할 때가 기억납니다. 저는 완전망했었는데요. ^^ 김요엘 학생은 발표 준비를 많이 한 것으로 들었는데요, 잘하고 왔지요? 발표 준비 과정과 발표 현장에 대해 전해주세요.

김: 다행히 발표 연습을 열심히 한 덕분에 나름 성공적으로 마칠 수 있었습니다. 발표 준비 과정은 본 발표 일주일 전으로 거슬러 올라가 SW재난연구센터 워크숍에서의 발표부터 이야기를 시작하겠습니다. 그때는 같은 내용을 한국어로 발표했으며, 발표 시간도 30분으로 더 여유가 있었지만, FSE 학회보다 더 많은 사람들 (약 100명) 앞에서 발표할 수 있었던 것이 큰 도움이 되었습니다. 또한, 워크숍 저녁 식사 자리에서 지도 교수님과 같은 테이블에 있던 교수님 4분께서 제 발표를 피드백해 주셨습니다. 피드백 당시 가장 심각하게 지적된 문제는 연구의 메인 아이디어가 너무 늦게 나오고 강조가 안 된다는 점이었고, 결국 발표 자료를 완전히 새로 고치게 되었습니다. 출국 하루 전에야 발표 자료를 완성했고, 비행기 안에서 스크립트를 작성했습니다. 학회 장소에 도착한 후에는 충북대학교 홍신 교수님께서 리허설을 도와주셨습니다. 마이크를 쥐는 법부터 영어 발음, 발표 속도, 제스처, 시선 처리까지, 지금껏 한 번도 신경 써보지 않은 부분들을 알려주셨습니다. 시차 적응으로 고생하시면서도 발표 연습을 도와주신 홍신 교수님께 이 자리를 빌려 다시 한번 감사의 말씀을 전하고 싶습니다.

발표 현장에서는 스크린이 작았던 점이 약간의 흠이었습니다. 또한, 앞에 단상이 없어 발표 슬라이드를 보려면 뒤의 스크린을 돌아봐야 했지만, 다행히 스크립트를 잘 외워 큰 문제는 없었습니다. 제 발표 때는 꽤 많은 30~40명 정도의 청중이 있었던 것 같은데, 이는 학회 첫날 마지막 세션이었고, 제 발표 전후로 Distinguished Paper Award에 선정된 발표들이 있었기 때문인 것 같습니다. 질의응답도 열심히 준비했는데, 받은 두 개의 질문은 생각보다 간단하여 무리 없이 답할 수 있었습니다.

편: 논문 내용에 대해서 간략히 소개 부탁드립니다. (편집자 주: 논문에 대한 보다 자세한 내용은 다음 코너인 “따끈한 논문" 코너에 준비 되어 있습니다.)

김: 저희 논문은 소프트웨어 검증기법의 효율성과 정확도를 동시에 향상시키기 위해 PBE (Programming-By-Example) 기법을 적용해 검증 대상 코드 전체가 아닌, 특정 함수들을 선택적으로 추상화한 후 검증하는 방식인 PBEAR을 제안하였는데요, 결과적으로 기존에 널리 사용되던 프레디킷 추상화 (Predicate Abstraction) 기법은 코드 전체를 추상화하여 빠르게 검증되지만, 부정확한 검증 결과를 유발하여 결국 제한 시간 내에 검증 속성 위반 (property violation)을 찾아내지 못했습니다. 반면, 추상화를 적용하지 않은 검증 기법에서는 원본 코드를 검증하기에 정확한 검증 결과를 도출할 수 있으나, 상당수의 실험 예제에서 Out-Of-Memory 오류가 발생했습니다. 그에 비해, PBEAR는 임베디드 소프트웨어 실험 예제에서 15개의 안전 검증 속성 중 1개를 제외하고 모든 속성 위반을 찾아냈습니다.

편: 본인 논문 외에 인상적인 발표가 있었으면 소개 부탁합니다.

김: 첫 번째는 카이스트 허기홍 교수님 연구실의 김태은 박사과정생이 발표한 “Evaluating Directed Fuzzers: Are We Heading in the Right Direction?”입니다. 이 발표를 소개하는 이유가 같은 비행기를 타고, 같은 방에서 잠을 자며 동고동락한 인연 때문만은 아닙니다. 해당 논문은 지향성 퍼저의 성능 비교 실험을 진행하면서 겪은 어려움과 그 과정을 담고 있습니다. 예를 들어, 같은 오류라도 서로 다른 오류 지점이 있을 수 있으며, 어떤 오류 지점을 설정하느냐에 따라 지향성 퍼저의 성능이 극단적으로 바뀔 수 있다는 점을 대규모 실험을 통해 입증했습니다. 그러나 기존 지향성 퍼저 논문들은 이러한 설정을 모호하게 기술하고 있는 경우가 많아, 이를 지적하며 실험의 표준을 마련하는 작업을 수행했습니다. 이 점에서, 해당 논문이 일반적인 연구와는 다른 차원의 기여를 하고 있다는 점이 인상적이었습니다.

두 번째는 독일 뮌헨 대학교 Dirk Beyer 교수님의 CPAChecker 팀에서 발표한 “Decomposing Software Verification Using Distributed Summary Synthesis”입니다. 사실, “summary synthesis”라는 용어가 제 논문에서도 자주 사용되기 때문에 발표를 하기 전부터 관심을 가지고 있었습니다. 저는 PBE 기법을 이용한 “function summary synthesis”을 제안하고 있는데, 해당 논문에서는 하나의 프로그램을 여러 블록(block)으로 나누어 각 블록에 대해 “block summary synthesis”을 수행하고, 이를 독립적인 검증 문제로 분할하여 멀티스레딩을 통해 검증하는 방안을 제안하고 있습니다. 이와는 별개로, 제 연구 또한 CPAChecker와 밀접한 관련이 있습니다. PBEAR의 성능 비교를 위해 CPAChecker 도구를 사용하기도 했고, 현재 진행 중인 연구도 CPAChecker을 기반으로 하고 있습니다. 학회에서도 마침 저자분께서 저를 먼저 알아보시고 말을 걸어주셔서 메일 주소를 주고받으며 CPAChecker에 대한 이야기를 나눌 수 있었습니다.

편: FSE 참여 소감 부탁합니다. 학회 장소가 일반적인 한국 사람들에게는 생소한 곳인데요, 학회 장소는 어땠는지 전해주세요.

김: 앞서 언급했듯이, 저는 작년 미국에서 열린 FSE에도 참여했습니다. 작년과 올해의 학회를 모두 돌아보며 느낀 점은, 개최국과 주최자에 따라 학회 분위기가 정말 다양하다는 것입니다. 올해 FSE는 작년의 전형적인 학회 장소 이미지와는 완전히 달랐습니다. 말 그대로 브라질 바닷가의 휴양지, 리조트에서 열렸습니다. 풀장에서 수영을 하며 즉석에서 코코넛 주스를 마실 수 있었고, 매일 저녁 야외 칵테일 바에서 작은 공연을 즐길 수 있었습니다. 특히 브라질에서 가장 인상 깊었던 점 중 하나는 제가 봤던 거의 모든 중소 규모의 식당에서도 공연을 한다는 것이었습니다. 피아노 연주부터 춤, 밴드 공연까지 다양했습니다. FSE도 예외는 아니었습니다. 첫날 리셉션에서는 쌈바와 뮤지컬 공연을 관람했고, 둘째 날 뱅퀫에서는 밴드 공연이 펼쳐져 사람들이 앞에 모여 춤을 추며 클럽 같은 분위기가 연출되었습니다.

편: 끝으로 소식지 독자들에게 인사말 부탁드립니다.

김: 저도 지금까지 한 명의 독자로서, 다른 분들의 다양한 이야기와 생각들을 재미있고 인상 깊게 읽어왔습니다. 이번 기회에 제 이야기와 생각을 소식지에 공유할 수 있어 기쁩니다. 읽어주셔서 감사합니다.

주요 약력

2023.03 ~ 현재 경북대학교 컴퓨터학부 박사과정 재학 (지도교수: 최윤자)

2023.02 경북대학교 컴퓨터학부 석사 (지도교수: 최윤자)

2021.02 경북대학교 컴퓨터학부 학사

주요 연구 분야

Model checking, program synthesis, model synthesis

따끈한 논문

프로그램합성 기법을 활용하여 엄밀하게 속성 위반 오류 찾기

• **최윤자 박사**
교수
경북대학교 컴퓨터학부
소프트웨어재난연구센터
yuchoi76@knu.ac.kr

모델검증 기법은 검증대상 모델이 주어진 요구사항을 만족하는지를 엄밀하게 검증하는 기법이다. 예를 들어, 검증대상이 교통신호 제어기이고, 요구사항이 "양 방향 신호가 동시에 녹색이 되는 일은 발생하지 않는다" 라면, 모델검증 기법은 양방향 교통신호 제어기가 취할 수 있는 모든 가능한 신호등 색의 변화 순서를 탐색하여 동시에 녹색이 될 수 있는 실행경로의 존재 여부를 보고한다. 이 기법은 현존하는 거의 유일한, 소프트웨어의 논리오류를 엄밀하게 검증할 수 있는 자동화된 검증기법이다[1]. 그러나, 나날이 복잡도가 증가하고 있는 프로그램 소스코드의 모든 가능한 실행경로를 엄밀하게 탐색하여 요구사항의 위반을 찾아낸다는 것은 매우 큰 시간과 비용이 소요되는 일이다.

지난 7월 브라질에서 열린 FSE2024 에서 발표한 논문 "PBE-based Selective Abstraction and Refinement for Efficient Property Falsification of Embedded Software"는 내장형소프트웨어 소스코드의 속성위반 여부를 검증함에 있어서, 프로그램 합성기법을 활용하여 모델검증의 성능향상을 도모한 최초의 논문이다. 이 글을 통해 해당 논문의 기본 아이디어 및 논문을 작성하기 까지의 소소한 이야기들을 공유하고자 한다.

1. 모델검증과 추상화

필자는 소프트웨어 모델검증 성능향상의 핵심은 추상화기법에 있다고 믿고 다양한 추상화 기법을 연구해왔다. 그동안 필자를 포함한 많은 연구자들이 추구해온 추상화 기법은 "건전한 추상화"로 추상화 이전에 소프트웨어가 지닌 행위들을 추상화 이후에도 그대로 보존해야 한다는 원칙에 입각한다. 예를 들어, 0 또는 1을 반환하는 함수를 추상화할 때, 임의의 값을 반환하는 nondet() 함수로 추상화하는 것은 건전한 추상화이나, 1 만 반환하는 함수로 추상화 하는 것은 건전하지 못한 추상화이다. 그 이유는 전자는 0 또는 1을 반환하는 기존의 행위를 그대로 포함하고 있으나 후자는 0을 반환하는 기존의 행위를 제거했기 때문이다. 그러나, 전자의 nondet() 함수처럼 너무 단순화된 추상화는 새롭게 추가된 무수한 잘못된 행위들로 인해 검증의 정확도가 매우 떨어지고, 잘못된 행위들에 대한 필터링 작업이 부수적으로 (매우 많이) 수행되어야 하는 또 다른 문제에 직면하게 된다. 건전하면서도 잘못된 행위를 최소한으로 포함하는 추상화기법을 개발하는 것은 모델검증 성능향상에 있어서 핵심 당면과제이다.

이러한 건전한 추상화에의 추구는 추상화의 건전성을 입증해야 하고 오경보에 대한 필터링이 필수적이라는 이론적/경제적 부담때문에 거시적인 추상화 보다는 프로그램 코드 명령어 단위의 미세수준 추상화가 일반적이다. 이는 규모가 크거나 높은 복잡도를 가진 소프트웨어를 대상으로 한 모델검증의 성능향상에는 크게 도움을 주지 못하는 결과를 보인다.

그러면 과연 거시적인 추상화는 어떻게 가능한가? 다수의 함수들이 호출관계로 엮여 있는 큰 규모의 소프트웨어 프로그램에서 속성검증에 필수적인 부분만 엄밀하게 검증하고 그 외는 적당히 활용만 하는 모델검증이 가능하지 않을까? 1,000라인 이상의 함수가 부작용(Side effect) 없이 블랙박스처럼 입력값에 대한 출력값을 반환하는 역할만 하는 것이라면, 그리고 해당 함수가 어떻게 결과값을 계산하는지에 관심이 있는 것이 아니고 결과값을 주요 제어논리에서 활용만 하는 것이라면, 과연 우리는 해당 함수의 모든 명령문들을 한 줄 한 줄 다 엄밀히 검증해야만 하는 것일까? 또한, 소스코드가 없는 외부함수를 호출해서 주요 작업을 수행해야 하는 경우는 어떻게 엄밀검증을 수행할 것인가? 필자는 꽤 오랜 시간동안 다양한 방법으로 이러한 문제들에 대한 해답을 찾고자 하였다.

2. PBE(Program-By-Example) 와 함수 추상화

그러던 중, KCSE2019 에서 KAIST 양홍석 교수의 확률적 프로그래밍에 대한 튜토리얼을 듣고 프로그램 합성 기법이 매우 흥미 있는 기법이라는 생각을 하게 된 것 같다. 그 후 POPL2020에 참석하면서 PBE 기법을 알게 되었고, "프로그램 합성기법을 이용한 Mock 자동생성" 이라는 연구주제로 합성기 EU-Solver 를 이용하여 여러가지 실험을 수행하면서 PBE를 거시적 추상화에 사용할 수 있을지 가능성을 시험해 보게 되었다[2].

PBE는 프로그램의 입력값 대비 출력값 정보를 바탕으로 프로그램을 합성해 내는 기법이다. PBE 합성의 결과로 만들어지는 프로그램은 최소한 주어진 입력값 대비 출력값의 관계는 만족시키는 함수이며, 상당히 일반화된 프로그램과 정확도가 높은 프로그램을 만들어 낼 수 있다. PBE 의 출현은 오랜 동안 풀리지 않았던 추상화의 문제, 특히 자동화된 거시적인 추상화의 문제가 풀릴 수도 있겠다는 반가운 전환점이었다. 중요도가 높지 않은, 그러나 복잡도는 매우 높은 함수들을 입출력 관계만 빼고 모두 추상화 할 수 있다면 모델검증의 비용을 현저히 낮출 수 있을 것이기 때문이다.

그러나, PBE 합성기법을 본격적으로 함수 추상화에 활용해 보겠다는 것은 매우 와일드한 생각이었는데, PBE로 생성해 낸 함수는 원본 함수의 건전한 추상화가 될 수 없기 때문에 기존의 "건전함"을 기본으로 한 추상화 방식들과는 완전히 다른 접근방식일 수밖에 없기 때문이다. "건전하지 않아도 오류를 잘 찾아내면 되는 것이 아닌가"라는 생각으로 기존의 "엄밀한 속성검증"에서 "엄밀한 속성오류 식별"로 목표를 수정하여 연구를 추진하게 되었다. 다시 말해, PBE 기반 엄밀검증은 소프트웨어가 속성을 만족하지 않음을 보일 수는 있으나, 추상화의 건전성이 보장되지 않았기 때문에 오류가 발견되지 않더라도 속성을 만족함을 증명했다고 할 수는 없다.

1) 논리오류 검증이 가능한 기법들 중 동적테스트는 엄밀성이 부족하고, 논제증명은 수작업이 요구된다 논리오류 검증이 가능한 기법들 중 동적테스트는 엄밀성이 부족하고, 논제증명은 수작업이 요구된다.

2) "내장형 소프트웨어 모듈의 추상화를 위한 실행 로그기반 점증적 Mock 자동생성", 정보과학회 논문지 (2022.5)

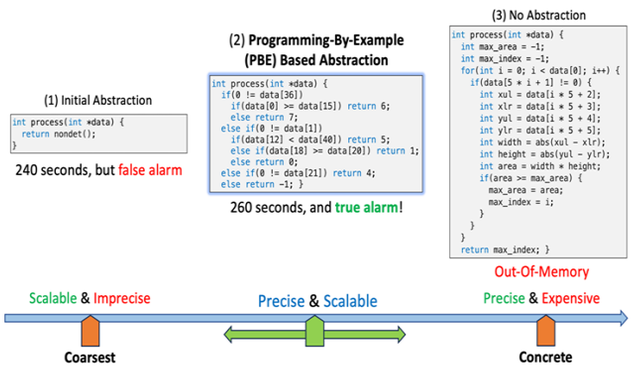


Figure 1 Expected Impact of PBE-based Abstraction

3. 논문의 주요 아이디어 및 성능향상에의 기대

<Figure 1>은 PBE를 활용한 추상화가 모델검증의 성능향상에 미칠 기대영향을 직관적으로 표현한 그림이다. 예시로 사용한 process 함수는 내장형 소프트웨어 코드의 일부에서 발췌한 것이며, "(1) Initial Abstraction"으로 표현된 코드는 "(3) No Abstraction"으로 표현된 process 함수의 소스 코드를 가장 높은 수준으로 건전하게 추상화한 예시이다. (1)번과 같이 가장 높은 수준의 추상화를 적용한 검증은 검증의 확장성을 높일 수 있고 (복잡하고 큰 규모의 소프트웨어에도 적용 가능) 빠르게 검증결과를 얻을 수 있지만, 검증 결과의 정확도가 낮을 수 밖에 없다. (3)번과 같이 추상화를 전혀 적용하지 않고 엄밀검증을 적용하면 검증결과의 정확도는 보장할 수 있으나, 확장성이 낮고 메모리 부족 등으로 검증에 실패할 가능성이 높다. (2)번에 표현된 예시는 (3)번 함수의 입출력 값을 이용하여 합성한 결과로 얻어진 코드이다. 해당 코드의 내용은 원본인 (3)번 코드와 의미적으로 일치하지는 않으나, 90% 이상의 입출력 값의 일치를 보이면서 반복문이 제거되고 코드의 사이즈가 줄어드는 등 복잡도가 현저히 낮은 코드임을 알 수 있다. 실제로 (2)번 코드를 이용하여 모델검증을 시행했을 때 260초 안에 속성 위반 경로를 찾아낼 수 있었다. 반면, (3)번 코드를 사용했을 경우에는 메모리 부족으로 검증을 마칠 수 가 없었고 (1)번 코드의 경우에는 240초 안에 검증 결과를 얻었으나 오경보였던 관계로 추상화 수준을 낮추고 재검증 하는 반복된 과정이 필요했다.

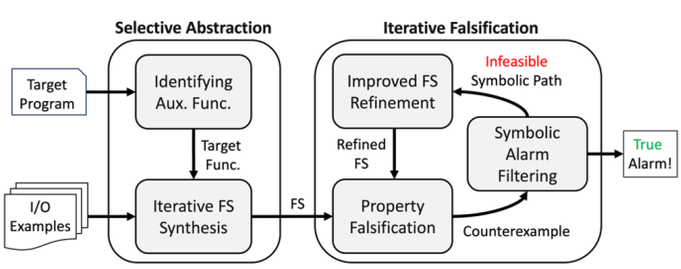


Figure 2 Overall Process of PBEAR

<Figure 2>는 이러한 아이디어에서 출발한, 새로운 속성오류를 엄밀하게 찾는 기법 PBEAR(PBE-based Selective Abstraction and Refinement) 의 개요이다. PBEAR는 먼저 검증대상 프로그램의 함수 단위로 PBE를 적용하며 추상화하고 그 결과를 평가하여 추상화에 적합한 함수들을 결정한다 (Selective Abstraction). 선정된 추상화된 함수들로 원본 함수들을 대체한 후 모델검증을 수행하여 속성오류를 빠르고 엄밀하게 탐색한다. PBE 합성함수의 정확도 문제로 인해 오탐이 발생하면 해당 오탐을 야기한 입출력 값들을 정정하여 재합성을 수행함으로써, 합성된 함수의 정확도를 점증적으로 높여가며 진탐을 찾을 때까지 모델검증을 반복한다 (Iterative Falsification). PBEAR 방식으로 내장형 소프트웨어 3종을 대상으로 15건의 요구사항 기반 속성검증을 수행한 결과, 추상화를 전혀 적용하지 않는 모델검증기 CBMC에 비해 44%의 메모리 소요량을 절감하면서 5건의 속성오류를 더 식별할 수 있었고, 적극적인 추상화와 반복된 정제과정을 거치는 CPAChecker에 비해서는 86%의 검증시간을 절감하면서 12건의 속성오류를 더 식별하였다. CBMC 가 5건의 속성오류를 발견하지 못한 이유는 메모리 부족이었고, CPAChecker가 12건의 속성오류를 발견하지 못한 이유는 높은 수준의 추상화를 초기부터 적용하여, 오탐을 제거하기 위한 "점증적인 추상화 수준 낮추기" 가 너무 많이 발생했기 때문에 72시간의 제한시간내에 검증을 마치지 못했기 때문이다.

4. 소회

지금까지 논문의 배경과 기본 아이디어, 적용결과를 간략하게 소개하였다. 짧은 글로 논문의 내용을 직관적으로 소개하기에는 어려움이 많아 부정확한 용어를 사용하기도 하였고, 기술적인 설명도 거의 생략하거나 두루뭉실하게 표현한 점 이해를 바라며, 자세한 내용은 논문을 참고하기 바란다. 논문의 기술적인 단점 및 향후 연구진행 방향 등 학술논문이라면 포함될 정형적인 이야기 대신에, 연구과정을 돌아본 소회로 글을 마치고자 한다.

먼저, 본 논문게재의 과정을 보면, 우수한 논문들이 실리는 학회에 참석하면서 얻을 수 있는 정보들이 연구의 방향설정에 큰 도움을 준다는 것을 알 수 있다. 2020년 처음 참석해본 POPL 은 필자에게는 매우 이해하기 어려운 내용의 학회였지만, 그럼에도 불구하고 최신동향을 어느 정도 볼 수 있었고 합성기법을 적용해 보겠다는 생각을 굳히게 된 곳이다. 그동안의 경험으로 비추어 볼 때, 새로운 아이디어는 내 전문 분야가 아닌 곳에서 찾을 가능성이 높았던 것 같다.

소프트웨어재난연구센터의 집단연구활동도 큰 도움이 되었다. 이우석 교수의 Duet 과 관심이 없었다면 이렇게 수월하게 PBE를 들여다볼 수 없었을 것이고, 2021년부터 지난 7월 초까지 6차례에 걸친 센터 정기 워크숍을 통해 얻은 약간의 긴장감과 피드백들이 연구 진행 상황 점검에 매우 큰 도움이 되었다. 리뷰 대응 기간에 소스코드 오픈에 대해 도움을 준 이주용 교수, 1저자 김요엘 연구원의 해외학회에서의 첫 발표 준비를 본인의 일처럼 살펴준 김윤호, 탁병철, 그리고 (특히, 브라질 현지에서까지 지도교수의 역할을 대신해 준) 홍신 교수에게 감사의 말을 전한다. 집단연구의 힘을 확인해 본 소중한 경험이었다.

마지막으로, (이쯤이면 포기하지 않을까 싶었던) 어려운 과정들을 잘 극복하고 한 단계 성장에 성공한 김요엘 연구원에게 격려의 말과 함께, 앞으로의 지속적인 성장에 대한 기대감을 전한다.

ICST 2024 Most Influential Paper 수상 기념

KAIST 김문주 교수님과의 대화 : ICST 2024 Most Influential Paper (MIP) 수상을 기념하며

• 김문주 박사
부교수
KAIST 전산학부
moonzoo@cs.kaist.ac.kr

올해(2024년) ICST(International Conference on Software Testing, Verification and Validation)의 MIP(Most Influential Paper)는 아래 논문에 수여되었습니다:

Ask the Mutants: Mutating Faulty Programs for Fault Localization

저자가 누구나고요? 김문주 교수님의 두 제자인 문석현 박사님 (삼성 SDS), 김윤호 교수님 (한양대학교)과 카이스트 김문주 교수님 본인, 그리고 같은 학교의 유신 교수님입니다.

많은 독자분들이 김문주 교수님을 이미 잘 아실 테니, 김문주 교수님에 대한 소개는, 소프트웨어 테스팅 분야 세계적 석학 정도로 갈음하겠습니다. 카이스트 부임 이전에는 Runtime Verification 분야에서, 재임 기간동안에는 Concolic 테스팅 분야에서 뛰어난 연구 업적을 남기셨죠. MIP 수상 논문은 결함위치 추정(fault localization) 분야에 관한 것으로, "뮤테이션 기반" 결함위치 방법을 최초로 선보였습니다.

서론이 길어졌습니다. 김문주 교수님에 대한 보다 자세한 약력은 이 글 말미를 참조하시기 바랍니다. 아래 문답에서 "편"은 편집자(이주용)를 "김"은 김문주 교수님을 지칭합니다.

편: 10년 전에 제가 교수님 연구실을 방문하였을 때에, 교수님께서 당시 제출 상태였던 이 논문에 대해 설명해 주셨던 기억이 납니다. 교수님의 눈빛과 목소리 톤을 통해 높은 수준의 지적 흥분감을 전달받았던 것으로 저는 기억하고 있습니다. 독자분들께 이 논문에 대한 간략한 소개를 부탁 드립니다. 그리고, 이 연구의 어떠한 점에 매료되셨는지 말씀 부탁 드립니다.

김: "Ask the Mutants:..." 논문은, SW 소스코드 안에 결함 위치를 정확하게 추정하는 mutation-based fault localization (MBFL) 기술을 개척한 논문으로, 기존 spectrum-based fault localization (SBFL) 기술 대비, 결함 위치 추정 정확도를 획기적으로 향상하였습니다.

저희 연구실이 SW 자동 테스팅 연구를 열심히 하다 보니, SW 자동 테스팅으로 발견된 결함의 정확한 위치 역시 자동으로 파악하는 것이 자연스러운 연구방향이었습니다. 하지만, 그 당시 널리 사용되던 SBFL의 결함 위치추정 정확도가 너무 낮아서, 다방면으로 결함 위치 정확도 향상 연구를 하다가, SW 테스팅에 쓰이는 mutation 기술이, SW 결함 위치 추정에도 큰 도움이 될 수 있겠다는 아이디어로 시작한 연구였습니다.

본 연구의 매력 포인트라면, 정적인 SW 소스코드와 동적인 SW 실행 사이의 연결 고리의 (형태 syntax와 의미(semantics) 중요성을 결함위치 추정 연구에 활용한 점으로, 재밌는 아이디어일 뿐 아니라, 실제로 결함위치 추정 정확도를 크게 향상했기에, 많은 연구자 들이 흥미롭게 받아들인 것 같습니다.

2014-SWTV-MT

편: 10여 년 전이면 GenProg를 필두로 하는 자동 프로그램 수정 연구가 SE 학계에서 떠오르고 있을 때입니다. GenProg를 교수님 논문 제목을 차용하여 표현하자면, Mutating Faulty Programs for Program Repair 정도가 될 수 있을 겁니다. 자동 프로그램 수정 기술은 지난 10년간 SE 내에서 가장 크게 성장한 분야이기도 하지만, 교수님을 포함해서 상당수의 연구자 분들이 신뢰성 등에 관련해서 우려를 표명해 오고 있기도 합니다. 큰 틀에서 뮤테이션을 이용하는 아이디어는 교수님 연구와 GenProg에서 동일한데요, 결함위치 추정과 프로그램 수정이라는 두 연관되지만 다른 분야에의 응용에 대한 교수님의 의견 부탁 드립니다.

김: 제 개인적인 생각으로는, 결함 위치추정은 (원인과 결과의 관계가 명확한) 과학의 영역인데 반해, 프로그램 수정은 보다 큰 문제를 다루고 있다고 봅니다. 모로 가도 서울만 가면 된다고, 프로그램 수정은, 결함 위치추정 보다, 문제의 범위도 넓고 솔루션의 자유도도 높아서 결함위치추정 연구 커뮤니티 보다 더 광범위한 연구 커뮤니티가 형성돼 있다고 생각합니다.

하지만, 자동 프로그램 수정 기술을 실제 현업에서 사용하기 위해서는, *왜* 자동생성된 patch가 결함을 해결했는지 이유를 설명할 수 있어야 하는데, 현재

프로그램수정기술은 아직 해당 문제를 만족스럽게 해결하지는 못하고 있는 것 같습니다.

편: MIP 수상 논문이 이후 연구에 끼친 영향에 대해서 설명 부탁드립니다.

김: 해당 논문은 결함 위치 추정 분야 뿐 아니라 mutation 분석 연구 분야에도 큰 영향력을 끼쳤고 (예. MBFL 연구를 기반으로, 2018년 Invasive Software Testing 논문으로 ICST 우수 논문상 수상), mutation을 통해 생성된 방대한 데이터 기반 SW 분석 연구를 개척하는데에 큰 공헌을 하였습니다 (ML-based fault localization, ML-based mutant selection 등). 보다 자세한 기술적인 영향력에 대해서는, 공동 저자이신 한양대학교 김윤호 교수님과 KAIST 유신 교수님이 잘 정리해 주신 MIP 발표 자료를 살펴보시면 좋을 것 같습니다.
https://swtv.kaist.ac.kr/publications/20240530%20ICST%20MIP%20v4.pdf

편: 임팩트 있는 연구의 중요성에 대한 이야기를 많이 듣습니다. 교수님은 ICST 2024 MIP 수상과 RV (Runtime Verification) 2019 Test of Time Award 수상으로 드러나듯 임팩트 있는 연구를 지속적으로 수행 중이십니다. 임팩트 있는 연구를 하기 위해서 지녀야할 자세나 마음가짐에 대해 말씀 부탁드립니다.

김: 제 경우를 돌아보면, 1순위로 좋은 연구 동료들이 갖춰지고, 2순위로 좋은 연구 행정 환경이 제공된 후에, 연구자 개인의 비전과 노력이 연구의 임팩트를 결정하는 것 같습니다. 좀 부정적으로 이야기하자면, 논문 편수를 강조하는 환경에서는, 영향력이 큰 연구를 하기 어려운 것 같습니다. 다행히 KAIST는 1, 2 순위 연구 환경이 잘 제공되는 곳 이어서, 좋은 연구성과를 낼 수 있었던 것 같습니다. 저도 3년 동안 국제학회 논문이 1편도 없던 적이 있었는데, 큰 스트레스 받지 않고 연구를 진행해서, 결국 더 좋은 연구성과들을 낸 경험도 있습니다.

편: 화제를 약간 바꾸어 보겠습니다. 2019년에 브이플러스랩(주)을 창업하셨습니다. 창업의 계기가 궁금합니다. 오래 전부터 창업을 계획하고 계셨나요?

김: 네, 창업 계획은 오래전부터 했습니다. 저는 KAIST 부임 전 삼성 SECUi.COM이라는 스타트업에서 병역특례를 수행했고, 실제 SW개발 현장에 도움되는 연구를 목표로 SW 자동 테스팅/디버깅 분야에서 다수의 산학연구과제를 성공적으로 수행했습니다. 그 결과 자연스럽게 창업으로 연결이 됐습니다.

편: 동일한 분야로 대학 연구실과 회사를 운영하시는 장점에 대해서 말씀 부탁 드립니다. 그리고 혹시, 단점도 있으면 함께 말씀 부탁 드립니다.

김: 장점으로는 세상을 보는 눈이 넓어졌습니다. 내가 하는 연구의 재미를 넘어서, 세상에 도움이 되는 연구를 해야겠구나, 세상에 선진 SW 기술의 유용성을 적극적으로 설득 해야겠구나 하는 사명감/의무감 같은 것들이 생겼습니다. 그리고, 연구실 학생들한테 너그러워졌습니다 (아무래도 회사 직원들보다 연구실 학생들이 더 우수합니다 ;-). 단점으로는, 회사 운영이 바빠서 아무래도 창업 전보다 연구에 소홀해진 점이 있습니다.

편: 끝으로 소식지 독자들에게 인사말 부탁드립니다.

김: SW 테스팅 및 디버깅 연구가 여러가지 공부할 것들이 많아서 쉽지는 않지만, 안전한 현대사회를 위해 꼭 필요한 연구 분야인 만큼, 소프트웨어공학 소사이어티 회원님들이 함께 관심 갖어 주시고 연구하시면 좋겠습니다.

KAIST 김문주 교수

주요 약력

현) KAIST 전산학부 부교수
현) 브이플러스랩(주) 대표이사 ('19 창업)
'06-현재 KAIST 전산학부 교수
'04-06 POSTECH 연구원
'02-04 삼성 SECUi.COM 차장
'01 Univ. of Pennsylvania 전산학 박사

주요 연구 분야

Verification and validation (V&V) of safety critical SW (자동차, 국방, 항공, 원자력 등)
Automated SW testing
Automated SW debugging

About The Institute

■ **김 태 효** 대표이사
(주)포멀웍스

(주)포멀웍스 CEO 김태효 대표님과의 대화

김태효 박사님은 2007년에 포멀웍스를 설립하신 이래 18년간 회사를 이끌어오시고 계십니다. 회사 이름에서 느껴지듯이 포멀웍스는 정형기법 및 관련 기술들에 기반한 SW 개발 및 검증 전문 기업입니다. 이번 정형기법 특집호에 매우 적격이지요. 또한 포멀웍스의 수장이신 김태효 박사님 스스로가 정형기법 전문가이시기도 합니다. 김태효 박사님은 2007년 카이스트에서 차성덕 교수님 지도로 박사학위를 취득하셨습니다. 김태효 박사님에 대한 보다 자세한 약력은 이 글 말미를 참조하시기 바랍니다. 아래 문답에서 "편"은 편집자를 "김"은 김태효 박사님을 지칭합니다.

편: 소프트웨어공학 소사이어티 회원분들께 포멀웍스에 대한 좀 더 구체적인 소개 부탁드립니다.

김: 포멀웍스는 안전필수분야의 소프트웨어 개발 및 검증을 전문으로 하는 회사입니다. 주로 원자력, 차량, 국방 분야에서 사용하는 소프트웨어를 개발 생명주기에 따라 단계별로 검증하는 서비스를 제공하고 있으며 나아가 인허가를 위한 컨설팅을 수행하고 있습니다. 이와 같은 검증 기술을 기반으로 차량용 소프트웨어 정적 분석 도구 HKSAT를 개발하여 판매하고 있으며, 고신뢰 소프트웨어도 직접 개발하여 납품하는 업무를 주로 하고 있습니다.

포멀웍스

편: 아시다시피 소식지의 독자 중에 학계에 계신 분이 많습니다. 자연스럽게 학계 관련 얘기 좀 드리겠습니다. 아마 다른 분야도 마찬가지겠지만 전산쪽 학계에서는 현실에 영향을 끼칠 수 있는 소위 "impact" 있는 연구를 해야한다는 목소리가 높은지 오래입니다. 하지만 한편으로는 학계의 보상 구조상 이미 해결책이 나와있는 문제를 이른바 "엔지니어링"하는 작업은 하기 어렵습니다. 결국, 현실 세계에 영향을 끼치기 위해서는 포멀웍스같이 전문 기업과의 협력이 절실히 필요해 보이는데요, 학계와 산업계를 모두 잘 이해하시는 박사님의 생각을 여쭙고 싶습니다. 학계에 계신 분들께 관련해서 조언을 부탁드립니다.

김: 저 또한 지금의 한국 R&D 상황에 대해 매우 안타깝게 생각합니다. 이 질문에 대해서는 이 자리를 빌어 학계에 계신 분들에게 국한되지는 않은 답변을 드리고 싶습니다.

우선 제가 생각하기에 국내에서 이루어지고 있는 많은 연구들이 너무 최신 동향에 초점이 맞추어져 있는 것 같습니다. 이 문제는 사실 역사적으로 쌓여 온 업보 같은 것인데 과거에 동일 주제에 대한 중복성 문제가 불거지면서 이제는 유사 주제에 대한 연구를 진행하면서 연구비를 마련하는 것이 쉽지 않은 환경으로 알고 있습니다. 이와 같은 배경으로 연구자들이 점점 연구의 내실을 다지기보다는 소위 'fancy' 하고 'trendy'한 주제들을 찾아 다니게 된 것 같습니다.

이와 같은 연구 경향은 최신 해외 기술을 빠르게 따라잡는데 유리할 수 있겠지만, 질문하신 소위 '엔지니어링'을 위한 노력은 등한시 할 수 밖에 없는 환경으로 이끌고 있는 것 같습니다. 학계의 연구 결과는 '엔지니어링' 없이 산업에 적용되기에는 한계가 있습니다. 아무리 좋은 기술도 고객이 이해하지 못하고, 바로 적용할 수 없다면 사실 쓸 수 없는 것이라 보아야 하기 때문입니다. 또한 연구 개발 자체도 큰 주제에 대한 유용성을 보이는데 급급한 나머지 충분한 엔지니어링이 이루어지지 않은 단편적인 결과물을 만드는데 그치는 경우가 많습니다.

산업계는 이와 같은 상황에서도 국내 연구 결과를 상용화하는데 적극적이어야 하나, 이 또한 쉽지 않은 경우가 많습니다. 앞서 말씀 드린 '엔지니어링' 간극을 좁히는 데에 국내 산업체의 기술력이 충분하다 보기 힘들기 때문입니다. 또한 기술이전을 받아도 이를 이어 연구 개발을 수행할 연구 인력도 턱없이 부족한

실정입니다.

제가 학계에 드리고 싶은 말씀은 비록 새로운 주제는 아니지만 '엔지니어링'이라고 여겨지는 단계에 수없이 많은 연구 주제가 있고, 이를 쌓아가는 과정 없이는 탄탄한 기술 토대가 만들어 질 수 없다는 것입니다. 즉 사상누각의 연구가 수행되고 또 다른 새로운 키워드의 연구로 옮겨가는 실적 위주의 악순환만 계속된다는 것입니다.

요즘 가장 뜨거운 deep learning은 새로운 이론일까요? 아니면 엔지니어링일까요? 제가 대학원을 다니던 시절의 인공지능은 연구의 긴긴 암흑기를 통과하고 있었습니다. 하지만 오래동안 연구하신 학자의 새로운 기술 하나가 분야를 다시 일으켜 세계의 중심에 가져다 놓았습니다. 이견이 많으실 수 있겠지만 제가 생각하기에 이 돌파구는 전혀 새로운 이론이기 보다는 인공신경망을 현실에 적용하기 위한 엔지니어링 쾌거에 더 가깝다 보아야 하지 않을까요?

이렇듯 학계의 이론적 성과와 산업계의 간극을 줄이는데 '엔지니어링'에 대한 연구 개발이 그 핵심인 것 같습니다. 학계도 산업계도 함께 노력해서 메꾸어야 하는 영역인 것이죠. 그리고 이와 같은 엔지니어링 성과 또한 연구 결과로 높이 인정 받는 풍토가 정착되었으면 좋겠습니다. 특히 국가 R&D에서도 주제에 대한 단순 중복성 검사도 좋지만 개발된 기술의 내실을 다질 수 있는 '엔지니어링'에도 많은 투자가 이루어지면 좋겠습니다.

편: 사업영역으로 화제를 바꾸겠습니다. 전통적으로 정형기법은 항공기, 자동차, 의료기기 등 안전이 매우 중요한 시스템에 국한되어 적용되었습니다. 하지만, 비교적 최근 들어 보안 이슈가 크게 부각되면서 네트워크에 연결된 모든 SW의 보안성 확보에 대한 기술수요가 높습니다. 포멀웍스도 보안쪽으로 사업영역을 확대해오고 있는 것으로 알고 있는데요, 어떠한 차별적 접근법을 취하고 있는지요? 전통적인 보안 기업과는 문제를 바라보시는 시각이 조금 다를 것도 같습니다.

김: 산업계에 오래 있다 보니 모든 분야의 발전이 일정한 패턴이 있는 것 같습니다. 처음에는 '기능'을 구현하는 것에 모든 노력이 들어갑니다. 당연히 어떤 소프트웨어든 본연에 역할을 수행하는 것이 제일 중요하겠지요. 다음으로는 예측할 수 있는 위험으로부터 보호하기 위한 '안전'의 역할이 중요해집니다. 안전필수분야들도 대부분 이와 같은 과정을 거쳐 어느 정도 성숙해 있는 단계입니다. 다음으로는 알 수 없는 위협으로부터 방어하기 위한 기술이 필요합니다. 바로 '사이버보안' 입니다. 이와 같은 과정에 따라 안전필수분야들도 모두 안전과 보안이 모두 중요한 시대로 가고 있습니다.

말씀하신 것처럼 저희는 안전을 위한 기술을 토대로 하고 있으나, 보안의 영역으로도 사업을 확장하고 있습니다. 기술적으로 보면 정적 분석이나 정형 검증 같은 기술은 보안의 영역에서도 그대로 사용될 수 있습니다. 안전에 '코딩가이드라인'이 있다면 보안에는 '시큐어코딩'이 있는 것처럼 말이죠. 두 가지 모두 코드 작성을 규칙을 다루고 있으나, 전자는 '안전'을 위한 패턴을, 후자는 '보안'을 위한 패턴을 정의하고 있다는 것이 다를 뿐입니다. 따라서 검증 기술 또한 검증하고자 하는 속성만 변경하여 서로 사용할 수 있습니다.

하지만 차이점도 확연하게 존재합니다. 예측할 수 있는 위험을 다루는 '안전'의 영역과 달리 '보안'의 영역에서는 상세히 알기 힘든 매우 저수준의 기술이 동원되는 경우도 있어 완전한 모델링을 통해 검증하기에는 한계가 있습니다. 또한 '보안' 취약점은 비정상적인 행위에 기반한 침투 기술을 활용한 경우도 많기 때문에 발생 가능성을 예측하기는 힘듭니다. 또 '보안' 분야는 구체적이고 세밀한 기술을 다루는 대신, 시스템 전체에 대해 체계적으로 분석하고 이를 검증하는 방법론의 발전이 '안전' 분야에 비해 느린 편입니다.

저희 회사의 경우에는 '안전' 분야에서 축적된 체계적인 분석 및 문서화 기술을 '보안' 영역 특히 산업제어시스템 분야에 접목하는 역할을 주로 하고 있습니다. '보안' 기술은 일반인들이 이해하기 매우 어렵고 복잡한 경우도 많기 때문에 비전공자가 보안 대책을 수립하는 데 한계가 있습니다. 반대로 보안 전문가는 안전필수분야의 분야 특성을 정확히 알지 못해 적절한 보안 대책을 적용하지 못하는 경우가 많습니다. 따라서 저희는 안전필수분야의 노하우를 가지고 보안 기술을 접목하는데 관심을 가지고 있으며, 이때 저희가 이미 보유한 '안전' 관련 검증 기술이 크게 도움이 되고 있습니다.

2022년 포멀웍스 워크숍 단체사진

편: AI의 신뢰성과 보안성에 대한 이슈가 뜨겁습니다. 해당 분야로 사업영역을 확대하고 계시거나 계획 중이신지요?

김: 요즘 들어 새로운 시스템에는 인공지능 기술이 탑재되지 않은 경우가 드문 것 같습니다. 산업계에도 이와 같은 시스템을 어떻게 검증할지 고민이 깊고, 시장의 니즈도 많은 상황입니다. 그러나 현재 인공지능 시스템에 대한 검증을 어떻게 할 것인가는 아주 초기 단계인 것 같습니다. 심지어 인공지능 시스템의 검증 대상을 어떻게 정의해야 할지도 이견이 많을 것 같습니다.

딥 러닝을 예로 들면, 인공지능 모듈의 구현은 잘 알려진 TensorFlow 같은 딥 러닝 프레임워크로 이루어져 있을 것입니다. 그러면 전통적인 소프트웨어 검증처럼 TensorFlow 구현 코드를 검증해야 할까요? 아마도 그렇지 않을 것입니다. 그렇다면 딥러닝 네트워크의 가중치를 검증해야 할까요? 혹은 학습 데이터를 검증해야 할까요? 아니면 결과만 확인하면 될까요?

세계적으로 많은 표준화 기관에서도 관련 연구들이 많이 이루어지고, 관련 가이드라인들도 여럿 작성되고 있는 것으로 알고 있습니다. 하지만 아직 인공지능이 적용된 시스템이 완전한 수준으로 체계적으로 검증되고 있다고 보기는 힘들 것 같습니다.

저희 회사에서도 몇몇 인공지능 시스템에 대한 검증 프로젝트를 수행하고 있습니다. 이 과제에서도 동일한 어려움이 있어, 인공지능 시스템의 검증의 대상은 무엇인가? 어떤 속성을 검증해야 하는가? 어떤 방법론을 사용할 수 있는가?와 같은 주제를 가지고 처음부터 토대를 쌓아 가고 있습니다. 인공지능 시스템은 그 나름의 독특한 특성을 지니고 있고 기존의 검증 기술을 그대로 적용하기 어렵기 때문에, 인공지능 시스템의 검증은 막 태동하고 있는 분야로 생각됩니다. 이러한 점에서, 기술 우위를 확보하고 시장을 선점할 수 있는 좋은 기회이기 때문에 국내에서도 관련 연구와 논의가 활발하게 이루어지면 좋겠습니다.

편: 다소 개인적인 질문을 드리겠습니다. 박사학위를 취득하신 같은 해에 포멀웍스를 설립하셨습니다. 학위 과정을 하시면서 아마 창업 준비를 이미 거의 다 해놓으신 것 같습니다. 창업 과정에 대한 말씀 좀 부탁드립니다. 아무나 창업을 할 수 있는 것은 아닐 테지만, 창업을 생각하고 있는 대학원생들과 교원들에게 좋은 참고가 될 것 같습니다.

김: 기대에 못 미치는 다소 심심한 답변을 드려야 할 것 같습니다. 저는 사실 대학원을 다니면서 창업에 대한 꿈이 크게 있지는 않았습니다. 물론 프로젝트를 수행하면서 PM과 같은 역할을 많이 하긴 하였지만 창업 자체는 크게 생각해 보지 못했습니다. 굳이 말하자면 산업체 취직보다는 해외 유명 기업의 연구원이 되고 싶다는 막연한 희망 정도가 있었던 것 같습니다. 창업하게 된 경위는 제가 박사과정을 졸업할 무렵 주변의 권유가 있어 처음 창업에 대한 고민을 시작했던 것 같습니다. 그리고 오히려 사업의 어려움과 무서움을 전혀 몰랐기 때문에 사실 어렵지 않게 선택할 수 있었습니다. '무식하면 용감하다'는 표현에 딱 적합한 예시인 것 같습니다.

이때 제가 결심할 때 가졌던 생각을 조금 공유하고 싶습니다. 저는 졸업할 무렵 산업체 취직이나 연구소에 가게 되면 앞으로 은퇴할 때까지 평생 있을 것인데, 굳이 몇 년 일찍 가는 것이 의미가 있을까 생각했고, 그렇다면 안정적인 직장에 다니기 전에 창업을 경험하고 도전하는 것이 나중에 미련이 남지 않을 것이라고 생각했습니다. 지금 돌이켜보면 용감하면서 무모한 생각이기도 했다고 생각합니다.

하지만 창업한 것에 대해 지금까지 아무런 후회가 없습니다. 제 생각입니다만 사람은 젊을 때 도전하고/경험하고/공부해서 쌓아 놓은 곳간을 나이 들어 조금씩 빼어 쓰면서 여생을 보내는 것 같습니다. 젊을 때 채워 놓은 곡식이 많으면 한참을 쓰면서 살 수 있을 것이고, 채워 놓은 것이 적으면 곳간은 금방 바닥나고 다른 것이 필요하게 될 것입니다. 결국 곳간이 비게 되면 지식이나 실력보다는 인간관계나 자신의 매력에 더 의존하게 되겠지요. 후자로도 물론 크게 성공할 수 있습니다. 하지만 저는 사회에 나오는 사람들이 더 오랜 기간 자신의 가치를 인정받으면서 자신이 하고 싶은 일을 하면서 살면 좋겠습니다.

저는 대학 강연 때마다 항상 학생들에게 젊을 때 제일 어렵고 남들이 하지 않는 일을 하라고 조언합니다. 그때는 어렵고 힘들지만 이렇게 채워진 지식과 노하우는 쉽게 대체할 수 없습니다. 하지만 쉬운 일로 빨리 성장한 사람에게는 자신을 바로 뒤에서 따라오는 수많은 후배들과 경쟁해야 합니다. 지금은 시들해 지고 있지만 대학생들의 최고 인기 목표가 공무원이라는 뉴스를 보고 참 슬펐습니다. 공무원도 훌륭하고 안정적인 직장이지만 가장 경직된 조직이기도 합니다. 잘 모르는 제가 느끼기에는 오직 한 가지 길이 포장된 대로로 나 있는 직업같이 느껴집니다.

창업은 제가 생각하기에 젊은 사람이 가장 많은 경험을 할 수 있고, 가장 빠르게 성장할 수 있는 비포장 도로 같은 길인 것 같습니다. 물론 그 끝에 오아시스가 있을지 사막이 있을지는 알 수 없지요. 하지만 확실한 것이 하나 있습니다. 한번 창업을 해본 사람이라면 어떤 직장 생활을 할지라도 성공할 수밖에 없습니다. 왜냐하면 경영자의 시각을 경험해 본 사람만이 가질 수 있는 넓은 시야를 갖게 되기 때문입니다. 회사를 경영하다가 작은 팀을 관리하면 얼마나 쉬울까요? 제품 기획, 개발, 납품, 판매, 마케팅, 회계를 모두 고민하다가 이 중 한 가지 일에 집중할 수 있다면 얼마나 일이 즐거울까요? 내가 직접적인 수익을 창출하지 않아도 월급을 받을 수 있다면 얼마나 기분 좋을까요?

저는 이런 넓은 시야는 창업을 해보지 않고는 책을 아무리 읽어도 얻을 수 없다고 생각합니다. 두렵더라도 창업에 많이 도전해 보셨으면 좋겠습니다. 그 어떤 경험으로 마무리될지 모르지만 그 경험과 지식과 노하우는 여러분의 곳간에 가득 차 있을 것이고, 앞으로의 인생을 어떠한 형태로든 풍요롭게 해 줄 것이라 확신합니다.

편: 포멀웍스의 인재상에 대해 말씀 부탁 드립니다. 어떠한 인재들이 포멀웍스와 잘 어울릴까요? 그리고 포멀웍스는 직원들에게 어떠한 기회를 제공한다고 할 수 있을까요?

김: 포멀웍스는 정직, 기술, 인간존중을 핵심 가치로 생각하고 있습니다. 이를 뒷받침하는 인재상을 한마디로 표현하자면 자신의 전문성에 대한 열정인 것 같습니다. 자신이 프로정신을 가지고 있어야 품질과 서비스로 정직한 사업을 할 수 있고, 끝없이 배우고자 하는 열정이 있어야 최고 수준의 기술을 가질 수 있습니다. 또 서로 일에 대한 열정이 느껴져야 상호 존중하고 신뢰하는 관계가 형성되는 것 같습니다.

경영자로서 부족하겠지만 위의 핵심 가치를 지키기 위해 다양한 형태로 노력하고 있으며, 이와 같은 노력이 직원들에게도 충분히 전달되고 있기를 바랍니다. 또 위의 핵심 가치를 통해 저희 회사의 비전인 "우리는 기술로 더 인간다운 삶을 구현하고자 존재합니다."라는 슬로건이 구현되기를 희망합니다.

편: 끝으로 소식지 독자들에게 인사말 부탁 드립니다.

김: 우선 정형기법 특집호에 인터뷰 기회를 주셔서 다시 한번 감사드립니다. 정작 정형 기법에 대한 이야기보다는 소프트웨어 검증에 대한 일반적인 이야기를 더 많이 한 것 같습니다. 오늘 말씀 드린 '안전'과 '보안'의 일들은 어쩌면 사회적으로 조연의 역할일 때가 많습니다. 하지만 조연 없이 영화가 만들어질 수 없듯이, 이 사회를 지탱해가는 최소한의 지지대로서의 사명감을 가지고 회사를 경영해 나가고 있습니다. 이 자리를 빌어 비록 화려한 조명을 받지 못하더라도 저보다 더 묵묵히 그리고 열정적으로 같은 분야에서 헌신하고 계신 모든 분들께 진심으로 감사와 격려의 말씀 드리고 싶습니다.

2018년 과학기술 정보통신부장관 표창장수여_김태효 대표이사

(주)포멀웍스 김태효 대표이사

주요 약력

한국과학기술원 전산학과 박사졸업

현 ㈜포멀웍스 대표이사

과학기술정보통신부장관 표창 (제25306호)

한국표준협회 AI+(에이아이플러스) 인증 심사위원

가천대학교 산업연계 교육자문위원회 자문위원

사단법인 한국정보보호학회 이사

국가보안기술연구소 미래 보안기술 포럼 전문위원

포멀웍스 주요 연혁

'2023 가족친화기업 인증

'2023 원전계측제어 심포지엄(NuPIC) 공로상

'2022 Microsoft Mixed Reality Partner

'2019 (사)한국정보보호학회 CPS보안연구회 공로상

'2018 포멀웍스-안랩 사이버보안 사업 MOU 체결

'2017 K-ESP 기술전문기업 선정

'2016 GS 인증: 타임바운더 v2.0 (TimeBounder v2.0)

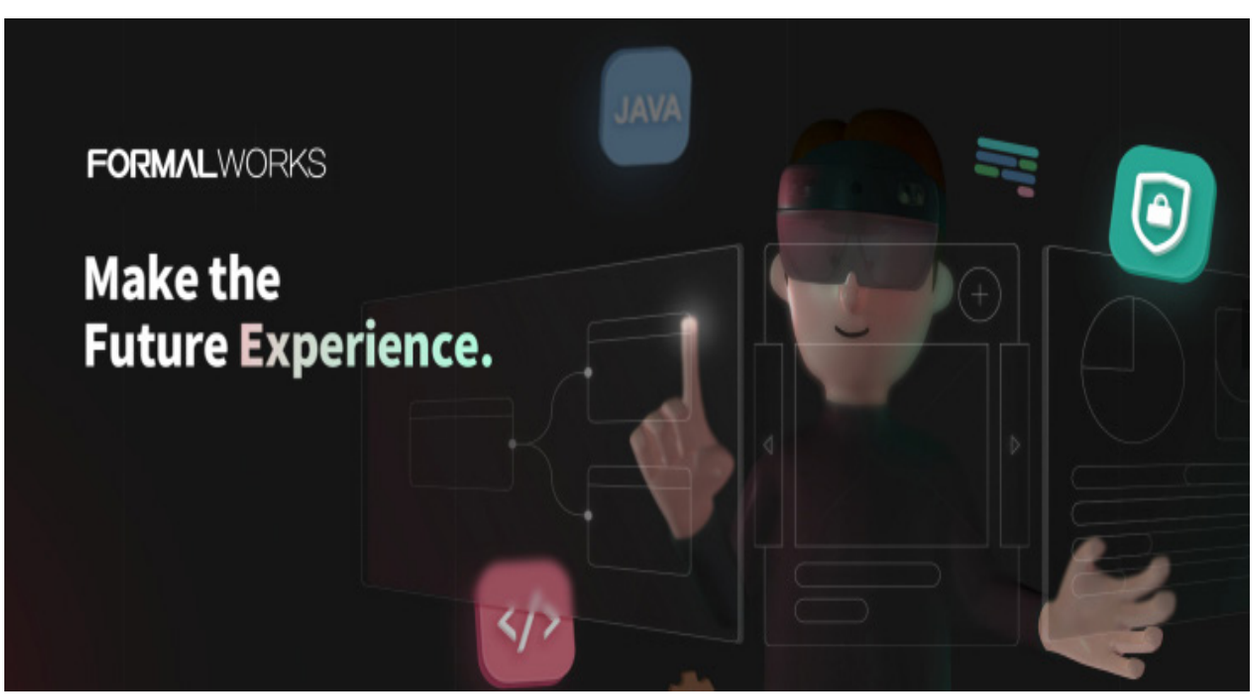
'2013 전문연구요원 병역업체 지정

'2012 이노비즈기업 인증, ISO-9001/ISO-14001 인증

'2011 기업부설연구소 설립

'2009 벤처 인증

'2007 (주)포멀웍스 설립



SW 공학 소사이어티 단기강좌 후기

SW 공학 소사이어티 단기강좌를 마무리하며

류덕산 교수
(전북대학교/단기강좌 조직위원장)

한국정보과학회 소프트웨어공학 소사이어티는 매년 여름마다 단기 전문가 강좌를 개최하고 있습니다. 2024년 8월 12일~13일 양일간 KAIST 도곡캠퍼스에서 열린 제12회 소프트웨어공학 단기 전문가 강좌에서는 최근 가장 주목받는 이슈중 하나인 'AI와 소프트웨어 테스팅'이라는 주제로 성균관대학교 차수영 교수님과 UNIST 김미정 교수님의 강의가 진행되었습니다.

차수영 교수님은 소프트웨어 테스팅 분야의 전문가로서, 테스팅 기법들의 성능을 기계학습 기법들로 향상하는 '데이터-기반 소프트웨어 테스팅' 연구를 수행하며, ICSE, FSE, ASE등 여러 최우수학회에 논문을 발표하고, 두번의 ACM SIGSOFT Distinguished Paper Award 를 수상한 최고의 전문가입니다.

김미정 교수님은 소프트웨어 테스팅 분야의 전문가로서, 소프트웨어 자동화 연구분야 중 테스트 자동 생성 및 퍼징 분야에서 연구를 수행하며, 소프트웨어공학 최우수학술대회에서 테스트 자동 생성 및 회귀 테스팅을 포함한 다양한 주제의 논문을 게재하고, 최근에는 인공지능 도메인에 특화된 퍼징 연구를 수행하고 있는 최고의 전문가입니다.

1일차 'Data-Driven Software Testing' 강의에서는, 화이트-박스 테스팅 기법중 '기호 실행'의 동작 원리를 학습하고, 코드 커버리지와 버그 식별에 효과적인 데이터기반 기호실행 기법을 다루었습니다.

2일차 'Software Testing for AI' 강의에서는, 인공지능 소프트웨어의 특성과 관련 오류의 분포 (Tensor:36%, Training:25%, Model:19%, API:13%, GPU Usage:7%)를 파악한 후, 텐서와 퍼징의 기본 개념을 배우고, 인공지능 소프트웨어에 퍼징을 적용하는 실습을 진행했습니다.

이틀 동안 총 14시간에 걸쳐 진행된 이번 단기강좌는 소프트웨어 테스팅의 기본개념을 체계적으로 확립하고, 인공지능을 소프트웨어 테스팅에 활용하는 기법과 소프트웨어 테스팅을 인공지능 소프트웨어에 적용하는 기법을 다루었습니다. 이 강좌는 인공지능과 소프트웨어 테스팅, 그리고 융합분야를 연구하는 연구자들에게 유익한 기회를 제공했습니다.

이번 단기강좌에는 총 34명이 등록하였으며, 그중 21명이 학생이고, 13명은 교수 및 산업체 관계자들이었습니다. 강의 평가가 4.8 / 5.0을 기록할 정도로 수강생들로부터 높은 평가를 받았습니다. 훌륭한 강의 장소를 마련해주신 고인영 교수님과 백종문 교수님께 감사드리며, 식사장소 섭외와 운영에 힘써준 전북대 인공지능&소프트웨어공학 연구실 학생들에게 깊은 감사를 표합니다.

SW 공학 소사이어티 단기강좌 후기

한양대학교 조한결 박사 과정생과의 대화

소프트웨어공학 단기전문가 강좌 후기

조한결 박사과정
한양대학교 컴퓨터학부
석박사통합과정)

올해, 2024년, 여름 단기전문가강좌가 "AI와 소프트웨어 테스팅"을 주제로 8월 12일~13일에 KAIST 도곡 캠퍼스에서 열렸습니다. 강좌는 성균관대학교 차수영 교수님과 UNIST의 김미정 교수님이 진행하셨습니다.

이번 단기강좌에 참여한 한양대학교 조한결 박사과정생과(지도교수: 이우석 교수님) 대화를 나누었습니다. 아래 문답에서 "편"은 편집자(이주용)를 "조"는 조한결 학생을 지칭합니다.

편: 먼저 본인 소개 부탁드립니다. 본인의 연구 분야도 함께 얘기해 주세요.

조: 안녕하세요, 저는 한양대학교에서 이우석 교수님의 지도 아래 박사과정 조한결입니다. 저는 프로그램합성(Program Synthesis) 분야에 매료되어 이 분야의 연구를 위해 진학하게 되었습니다.

프로그램 합성은 개발자가 일일이 코드를 작성하지 않고도 원하는 프로그램을 자동으로 생성할 수 있는 기술을 의미합니다. 이 기술은 소프트웨어 개발의 효율성을 크게 높일 수 있는 잠재력을 가지고 있고, 최근에는 AI 기반의 ChatGPT, Copilot 같은 프로그램이 나타나면서 큰 관심을 받고 있습니다.

현재 저는 탐색 기반 합성 방법과 AI 기반의 합성 방법의 장점을 결합하여 더 우수한 프로그램 합성을 실현할 수 있는 방법을 연구하고 있습니다. 탐색 기반 합성 방법은 그 안정성과 신뢰성 측면에서 강점을 가지고 있으며, AI 기반의 합성 방법은 유연성과 문제 해결 능력에서 두각을 나타냅니다. 저는 이 두 가지 접근법의 장점을 최대한 활용하여, 기존의 한계를 극복하고 더욱 정교하고 효과적인 프로그램 합성을 이루는 것을 목표로 하고 있습니다.

편: 해당 단기강좌를 수강하게 된 계기가 어떻게 되나요? 단기강좌가 본인 연구에 어떻게 도움이 될 거라고 기대했나요?

조: 소프트웨어 소사이어티로부터 보내주시는 메일을 통해 알게 되었습니다. 최근 연구실 관심 분야가 퍼징(Fuzzing)과 관련이 있었는데, 이번 단기 강좌를 통해 퍼징에 대한 전반적인 내용과 해당 분야의 최신 연구 정보를 배울 수 있는 좋은 기회로 보였습니다. 또한 다른 분야에서는 AI와의 융합연구가 어떻게 이루어지고 있는지에 대한 정보를 얻을 수 있을 것이라 생각했습니다.

편: 강좌 내용에 대해서 간략히 정리해 주세요. 어떠한 내용을 배웠죠?

조: 첫째날은 소프트웨어 테스팅의 전반적인 개념에 대해 배우고 다양한 소프트웨어 테스팅 기법들의 성능을 향상하는 Data-driven 테스팅에 대해 배울 수 있었습니다. 데이터로부터 일관되게 효과적인 전략을 자동으로 만들어 적용함으로써 기존 전략들을 향상시키는 다양한 연구들과 최신 퍼징 연구 트렌드에 대해 알 수 있었습니다.

둘째날은 AI 소프트웨어의 특징과 관련한 오류가 무엇이 있는지 그리고 AI 소프트웨어 테스팅을 위한 요구 조건에 대해 배웠습니다. 그리고 AI 소프트웨어에 퍼징을 적용해보는 실습을 통해 AI 소프트웨어 테스팅의 실제적인 어려움을 몸소 체험해볼 수 있었습니다.

편: 수강 소감 좀 얘기해 주세요. 아마도 기대했던 목적을 달성한 부분도 있을 거고, 의외의 수확이 있을 수도 있고, 아쉬운 부분도 있을 텐데요.

조: 이번 강좌를 수강하면서 여러 면에서 기대했던 목적을 달성할 수 있었습니다. 우선, 퍼징에 대한 이해가 크게 향상된 점이 가장 큰 성과였습니다. 이론적으로만 알고 있던 퍼징을 실습을 통해 직접 적용해 보면서 그 개념과 작동 방식을 보다 깊이 이해할 수 있었습니다. 그리고 소프트웨어 테스팅 분야에서는 AI에 대해 어떻게 접근하고 있는지 파악 알 수 있었습니다.

의외의 수확으로는, 프로그램 합성이 AI 도메인에 효과적인 퍼징을 위해 활용 될 수 있음을 발견한 점입니다. 합성을 통해 AI 소프트웨어의 실행 경로 조건(branch condition)을 추론해 퍼징의 효과를 향상시키는 방법이 기억에 남습니다.

반면에, AI 방법들이 기존에 해결하지 못하던 문제를 극복하거나 완화 해주는 결과들을 보여주는데 이런 강점을 기존 연구에 활용하는 융합적인 방향에 대해서는 다루지 않았던 점이 아쉬웠습니다.

편: 단기 강좌 분위기는 어떠했나요? 시설이나 강좌 환경은 만족스러웠나요? 개선할 사항이 있으면 얘기해 주세요.

조: 이번 단기 강좌는 쾌적한 환경에서 원활히 진행되어 분위기는 매우 좋았다고

생각합니다. 무엇보다 강의 하여 주시는 교수님들께서 복잡한 개념도 이해하기 쉽게 잘 설명해 주셔서 강의를 듣는 내내 큰 어려움 없이 따라갈 수 있었습니다. 또한, 실습 환경도 매우 만족스러웠습니다. 직관적이고 깔끔하게 준비된 실습 환경 덕분에 실습 과정이 원활하게 진행되었고, 실습을 통해 배운 이론을 직접 적용해 볼 수 있어서 학습 효과가 더욱 높았습니다. 개선할 사항을 굳이 꼽자면, 강의와 실습 간의 시간 배분이 조금 더 조정되면 좋겠다고 생각합니다. 준비해주신 실습 내용에 비해 실습 시간이 약간 부족하다고 느꼈습니다.

편: 끝으로 소식지 독자들에게 인사말 부탁 드립니다.

조: 연구를 하다 보면 체력이 얼마나 중요한지 절실히 느낍니다. 건강이 뒷받침되어야 꾸준히 좋은 연구를 이어갈 수 있다고 생각합니다. 모두들 건강 잘 챙기시길 바랍니다. 마지막으로, 강좌가 원활히 진행될 수 있도록 장소와 환경을 준비해주신 관계자 분들께도 감사드립니다.

조한결 박사과정

주요 약력

2020.02 한경국립대학교 컴퓨터학과 전공 (학사)

2020.03-현재 한양대학교 컴퓨터학부 석박사통합과정

주요 연구 분야

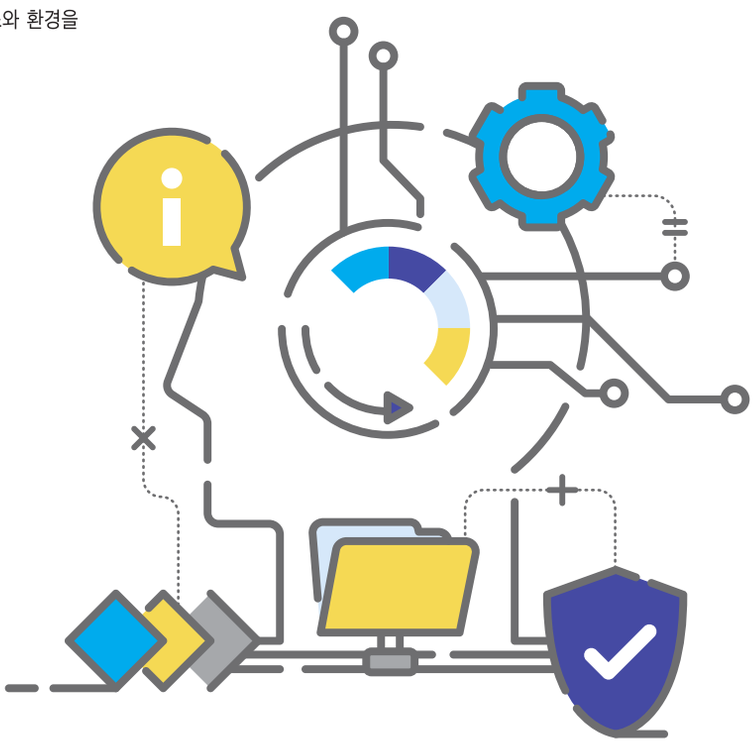
Program Synthesis

대표 논문

[1] 분할 정복 및 요약해석 기반 JavaScript 프로그램 합성, 조정민 , 조한결 , 이우석 - 정보과학회논문지, 2021

[2] MADUSA: mobile application demo generation based on usage scenarios, Jaehyung Lee, Hangyeol Cho, Woosuk Lee, Journal of Automated Software Engineering 2023

[3] Inductive Synthesis of Structurally Recursive Functional Programs from Non-recursive Expressions, Woosuk Lee, Hangyeol Cho, POPL 2023



국내외 학술행사 리스트

■ 2024 국내외 학술대회

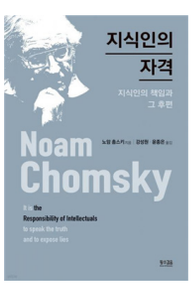
Full name	학술대회	대회일자	논문마감	URL	기준 (행사, 제출)
International Systems and Software Product Line Conference	SPLC 2024	2024.09.02~06		https://2024.splc.net/	행사
European Conference on Object Oriented Programming	ECOOP 2024	2024.09.16~20		https://2024.ecoop.org/	행사
International Conference on Automated Software Engineering	ASE 2024	2024.10.27.~11.01		https://conf.researchr.org/home/ase-2024	행사
International Conference on Software Maintenance and Evolution	ICSME 2024	2024.10.6~11		https://conf.researchr.org/home/icsme-2024	행사
International Conference on Model Driven Engineering Language and Systems	MODELS 2024	2024.09.22~27		https://conf.researchr.org/home/models-2024	행사
Conference on Object-Oriented Programming, System, Languages, and Applications	OOPSLA 2024	2024.10.20~25		https://2024.splashcon.org/track/splash-2024-oopsla	행사
International Symposium on Software Reliability Engineering	ISSRE 2024	2024.10.28~31		https://issre.github.io/2024/	행사
Asia-Pacific Software Engineering Conference	APSEC 2024	2024.12.03~06		https://conf.researchr.org/home/apsec-2024	행사
Symposium on Software Engineering for Adaptive and Self-Managing Systems	ICSE-SEAMS 2025	2025.04.26.~05.04	2024.10.08	https://conf.researchr.org/track/seams-2025/seams-2025-research-track	제출
Korea Software Congress	KSC 2025	TBD	TBD	TBD	행사, 제출
Korea Conference on Software Engineering	KCSE 2025	TBD	TBD	TBD	행사, 제출
International Conference on Software Testing, Verification and Validation	ICST 2025	2025.03.31~04.04	2024.10.25	https://conf.researchr.org/home/icst-2025	제출
Conference on Software Engineering Education and Training (CSEE&T)	ICSE-CSEE&T	2025.04.27.~05.03	2024.10.3	https://conf.researchr.org/home/icse-2025/cseet-2025	제출
International Conference on Software Engineering-Software Engineering in Practice	ICSE-SEIP	2025.04.27~05.03	2024.10.10	https://conf.researchr.org/track/icse-2025/icse-2025-software-engineering-in-practice	제출
International Symposium on Software Testing and Analysis	ISSTA 2025	2025.09.25-28	2024.10.31	https://conf.researchr.org/dates/issta-2025	제출

V VIEWPOINTS

소사이어티 광장

축하합니다!

- **박사학위 수여**
  - 강성민 (KAIST, 지도교수: 유신): 2024년 8월 박사학위 취득 (논문제목: 대형언어모델이 만들고 프로그램 실행으로 확인하는 소프트웨어 요소 생성 / Reliable Large Language Model Based Software Artifacts via Execution), 졸업 후 KAIST 박사후연구원 취업 예정
  - 안가빈 (KAIST, 지도교수: 유신): 2024년 8월 박사학위 취득 (논문제목: 대규모 소프트웨어 시스템의 효율적 버그 해결을 위한 결함 위치 식별 기술과 지속적 통합의 상호 보완적 결합 전략 / Synergizing Fault Localization and Continuous Integration to Streamline Bug Resolution in Large-Scale Software Systems), 졸업 후 ROKU Korea 취업
- **퇴임**
  - 최진영 교수, 고려대학교 정보보호대학원 정년퇴임, 2024.08
  - 정효택 책임연구원, 한국전자통신연구원 정년퇴임(36년 10개월 재직), 2024.09
- **이직**
  - 김동선 교수, 고려대학교 컴퓨터학과 부교수 임용, 2024.09
  - 김지응 교수, 연세대학교 컴퓨터과학과 조교수 임용, 2024.09
- **연구과제**
  - 배경민 교수, 2024 SW스타랩 선정 (과제명: 보안 운영체제의 실용적인 정형검증을 위한 자동화 기술 개발), 2024.08
- **논문지**
  - 정보과학회지 제42권 제8호가 “LLM과 소프트웨어공학” 주제로 발간되었다. 김진현 교수가 편집위원을 맡고, "소프트웨어 명세 제작을 위한 LLM" (김미정) / "ChatGPT를 이용한 요구사항 내의 중의성 탐지 및 해결 성능 평가" (이선구 • 강종구 • 백종문) / "LLM을 활용한 소프트웨어 문서 분류 및 생성 연구" (허주은 • 이선아) / "초거대 언어모델을 활용한 소프트웨어 결함 예측 연구 동향" (류덕산) / "신뢰할 수 있는 언어 기초 모델의 연구 동향" (박상돈) - 5편의 특집원고가 포함되었다. LLM과 소프트웨어공학의 다양한 측면을 깊이 있게 이해하고, 향후 연구와 개발에 있어 유용한 통찰을 얻을 수 있을 것으로 기대한다 .
- **출간**



- KAIST 강성원 교수, 노암 촘스키의 <지식인의 자격> 번역 출간

KAIST 전산학부 강성원 교수는 현대 언어학의 개척자이자 생존하는 가장 중요한 지식인으로 알려진 노암 촘스키의 저서 <지식인의 자격: 지식인의 책임과 그 후편>(황소걸음, 184쪽)을 윤종은 번역가와 공동 번역하여 2024년 3월 출간했다.

<지식인의 자격>은 촘스키가 베트남 전쟁이 한창이던 1967년에 쓴 〈지식인의 책임〉과 2011년에 쓴 〈지식인의 책임 후편: 국가를 견제하기 위한 특권의 사용〉을 묶어 발간한 책이다. 촘스키는 이 책에서 지식인의 책임에 대해 명쾌하지만 뼈아픈 답을 제시한다.

강성원 교수의 저서로는 <소프트웨어 아키텍처로의 초대 - 소프트웨어 아키텍처 설계의 근본 원리들>, <체계적인 소프트웨어 제품라인 개발>과 공저 <소프트웨어 제품라인 개발 입문>이 있다.

- 고려대학교 차성덕 교수, <소프트웨어공학 이야기> 개정증보판 출간, 2024.06

이 책은 소프트웨어 공학의 핵심을 깊이 있게 이해하고자 하는 개발자와 학생들을 위한 소중한 자료이다. 차성덕 교수는 정년퇴임을 앞두고 자신이 얻은 경험과 지식을 후학들과 현업의 개발자들에게 전하고자 이 개정증보판을 준비하였다. 2019년 출간한 초판 이후, 빠르게 변화하는 컴퓨터 과학의 최신 동향을 반영하기 위해 계속해서 내용을 보완해왔다. 이 책은 학부생뿐만 아니라, 전산학을 전공하지 않았지만 실무에서 소프트웨어 개발을 진행 중인 개발자들에게도 유익하도록 설계되었다. 저자는 IEEE Software와 같은 실무 중심의 출처뿐만 아니라, ICSE, ASE와 같은 권위 있는 학술대회의 연구 논문을 통해 얻은 인사이트를 포함하여, 소프트웨어 공학의 중요한 주제를 평이한 언어로 설명하고 있다. 저자는 독자들에게 단순한 지식의 암기를 넘어서, 실제 상황에서 어떻게 이론을 적용할 수 있을지에 대한 고민을 권장한다. 정년퇴임을 앞둔 저자의 마지막 노력이 담긴 이 책은, 컴퓨터학을 전공하는 학생들과 소프트웨어 개발 현장에서 일하는 모든 이들에게 큰 도움이 될 것이다.

: 기고문 및 소식 모집

Dream + Think + Idea + come true

소프트웨어공학 소사이어티 소식지는 여러 연구자분들의 생각과 소식을 나누는 광장입니다. 다음과 같은 구성으로 소식지를 구성하고자 하오니, 여러분들의 적극적인 참여를 바랍니다. 투고글의 형식은 자유형식이며, 분량은 A4 기준 2~4페이지 입니다.

- 기고문: 소프트웨어공학 및 소사이어티에 대한 생각 (자유주제)
- 신진연구자 소개: 만 40세 이하 또는 박사학위 취득 후 7년 이내의 연구자 소개
- 국내외 학술행사 소개: 주요 학술행사 소개, 학술행사 참여 후기 등
- 기관소개: 소프트웨어공학연구 관련기관 소개
- 소사이어티 광장: 소사이어티의 새로운 소식 나눔

▶ 소사이어티 알림

- 소프트웨어공학 소사이어티에서는 매년 소프트웨어공학 우수논문상을 추천하여 시상하고 있습니다. 최우수 학술대회에 논문 발표로 참가하는 학생에게 장려금(약 100만원 수준)을 지원할 예정입니다.
- 소프트웨어공학 소사이어티 소개 동영상 : https://www.youtube.com/watch?v=HWGsy-Pyle0
- 소프트웨어공학 소사이어티 페이스북 : https://www.facebook.com/groups/668196744037453

제출방법

- 이메일 제출 (소프트웨어공학 소사이어티 편집부: ksepup@gmail.com)

문의처

- 이주용 교수 (UNIST, 052-217-2123, jooyong@unist.ac.kr)
- 정우성 교수 (서울교육대학교, 02-3475-2562, wsjung@snue.ac.kr)
- 지은경 교수 (한국과학기술원, 042-350-7810, ekjee@se.kaist.ac.kr)
- 손정주 교수 (경북대학교, 053-950-5556, jeongju.sohn@knu.ac.kr)

SOFTWARE ENGINEERING SOCIETY

: 소사이어티 조직도

SOFTWARE ENGINEERING SOCIETY

회장 **이정원**
아주대학교

총무 **홍신**
충북대학교

감사 **민상윤**
㈜솔루션 링크

부회장(기획) **유준범**
건국대학교

부회장(학술) **유 신**
KAIST

부회장(교육) **류덕산**
전북대학교

부회장(편집) **이주용**
UNIST

부회장(대외협력) **백종문**
KAIST

부회장(조직 및 지원) **남재창**
한동대학교

분과위원장
(블록체인) **김순태**
전북대학교

- 기획이사 **김태호** / 서울여자대학교
  - 기획이사 **정세진** / 진주교육대학교
    - 기획이사 **윤경아** / KT AI테크랩
- 학술이사 **허기홍** / KAIST
  - 학술이사 **김미정** / UNIST
    - 학술이사 **배경민** / POSTECH
- 교육이사 **강종구** / 성신여대
  - 교육이사 **안성수** / 국립경상대학교
    - 교육이사 **이희진** / 동양미래대학교
- 편집이사 **정우성** / 서울교육대학교
  - 편집이사 **지은경** / KAIST
    - 편집이사 **손정주** / 경북대학교
- 대외협력이사 **이 근** / 삼성전자
  - 대외협력이사 **박인욱** / LG전자
    - 대외협력이사 **구태완** / 현대자동차
- 조직및지원이사 **김진대** / 서울과학기술대학교
  - 조직및지원이사 **김동선** / 고려대학교
    - 조직및지원이사 **이우석** / 한양대학교

발행정보

발행일 2024년 9월 1일
발행인 이정원
발행처 사단법인 한국정보과학회 소프트웨어공학소사이어티
연락처 경기도 수원시 영통구 월드컵로 206 아주대학교 원천관 305호 (우: 16499)
이정원 (전화: 031 219 1813, 홈페이지: https://eslab.ajou.ac.kr/members/professor)
홈페이지 : http://www.sigsoft.or.kr)

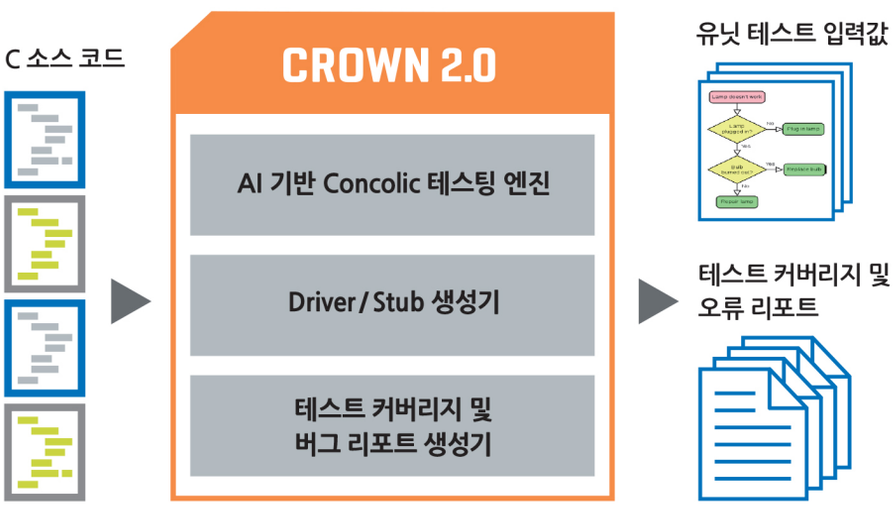
V+Lab

CROWN 2.0

C 프로그램 자동 유닛 테스팅 도구

| CROWN 2.0 소개 |

CROWN 2.0 은, C 프로그램 소스 코드를 자동으로 분석하여, 프로그램의 모든 가능한 시나리오들을 수행하는 다양한 테스트 입력값을 100% 자동 생성하는 SW 자동 테스팅 도구입니다.



CROWN 2.0 Workflow

1. 프로그램의 소스 코드를 자동으로 분석하여, 유닛 테스팅에 필요한 테스트 driver와 테스트 stub 코드를 자동으로 생성
2. 테스트 대상 프로그램의 함수와 해당 함수를 위해 자동 생성된 테스트 driver/ stub 코드를 대상으로, AI 기반 Concolic 테스팅 엔진을 사용하여, 모든 가능한 수행 경로를 실행하는 다양한 테스트 입력 값들을 생성
3. CROWN 2.0을 통해 달성한 테스트 커버리지 및 테스팅 과정에서 발견한 대상 프로그램의 오류를 사용자에게 보고

제품 특징

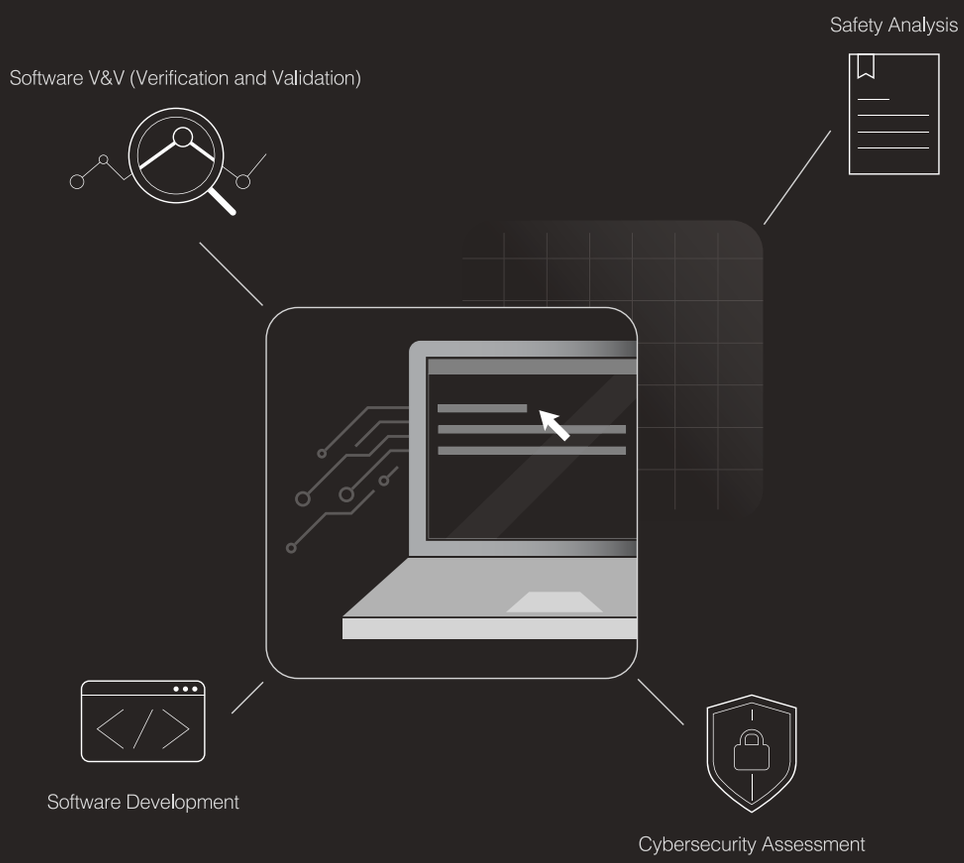
- 브이플러스랩이 10년간 자체 개발한 AI 기반 Concolic 자동 테스팅 엔진을 활용한 테스트 입력값 자동 생성
- 테스트 대상 함수를 위한 driver와 stub 코드 자동 생성
- 화이트 박스 소스 코드 테스트 커버리지 리포트
- 초보자도 사용하기 쉬운 웹 기반 GUI 인터페이스
- 생성된 테스트 입력값을 CSV 파일로 저장하여, 다른 테스팅 도구에서 활용 가능

입증된 자동 테스팅 성능

- 세계 최고의 SW 공학 국제학술대회 ICSE '19에, 현대모비스의 자동차 소프트웨어 적용 사례 연구 발표
- 분기 커버리지 90% 이상, MC/DC 커버리지 80% 이상 자동 달성 가능
- 현대자동차, LIG넥스원, ETRI, 국가보안기술연구소, 포항공대 등 납품 중

브이플러스랩(주) 제품 및 서비스 문의: contact@vpluslab.kr

(우 06648) 서울시 서초구 반포대로 110 산우빌딩 5층 (서초동 1543-11) TEL. 02 - 563 - 8763 FAX. 02 - 563 - 8764 E-MAIL. contact@formalworks.com



소프트웨어 개발 및 검증 전문 기업

FORMALWORKS

주요 서비스

소프트웨어 확인 및 검증 규제 지침/국제 표준에 기반한 소프트웨어 V&V 및 시험 서비스

소프트웨어 안전성 분석 안전성-필수 소프트웨어 위험 요소 분석 및 안전성 평가 서비스

SCADA 사이버보안 SCADA 장비/시스템에 대한 사이버 보안 위협 분석 및 대응 조치 수립 서비스

내장형 소프트웨어 및 분석 솔루션 개발 고품질 / 고신뢰 실시간 내장형 소프트웨어 및 분석 도구 개발 서비스

SOFTWARE
ENGINEERING
SOCIETY

사단법인 한국정보과학회 소프트웨어공학소사이어티

주소: 경기도 수원시 영통구 월드컵로 206 아주대학교 원천관 305호 (우: 16499)
이정원 (전화: 031 219 1813, 홈페이지: https://eslab.ajou.ac.kr/members/professor)

홈페이지 : http://www.sigsoft.or.kr