미개한 프로그래밍 기술

이광근

서울대 컴퓨터공학부 교수

소프트웨어무결점 연구센터 소장
(교과부/한국연구재단 지정 우수연구센터)

02/04/2010 @ 1st CS4HS Workshop

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

컴퓨터과학은
머리로 궁리해서(abstraction) 기계로 돌리기(mechanization).

Computer Science is
about mechanization of abstraction.

컴퓨터과학은
머리로 궁리하는 것(intelligence)에 대한 연구(science)이고
그 응용은 인간 지능의 확장.

Computer Science is
the science of intelligence
and its application is extensions of human intelligence.

마치

생명과학이 생명현상에 대한 연구이고
그 응용이 인간 수명의 연장이듯이.

computer science		life science
science of intelligence	$\square$	science of life
extension of human intelligence		extension of human life

computer science		medical science
science of intelligence	$\square$	science of human disease
extension of human intelligence		conquering cancers

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

computer science		physics
science of intelligence	□	science of nature
extension of human intelligence		discovering energy source

computer science		mathematics
science of intelligence	$\square$	science of reasoning
enhancing human intelligence		enhancing human reasoning

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

그러나, 컴퓨터분야는 아직 미숙합니다

분야의 역사 = 겨우 50년

그래서

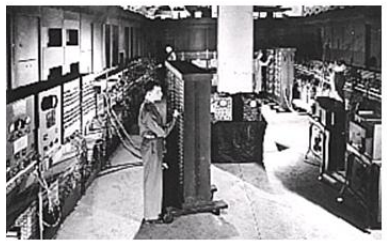
- 미숙하기 때문에 $\Longrightarrow$ 기회가 많다.
- 미숙하기 때문에 $\Longrightarrow$ 지금 “Newton”, “Galileo”, “Curie”, 와 같이 살고있다.

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

컴퓨터분야 발전 속도가 빠르다? (1/3)

- 컴퓨터속도: 10^7배/50년 발전

  10^3flops/cpu ENIAC(1945)
  $\longrightarrow 2 \times 10^{10}$flops/core IBM Sequoia(2010)



컴퓨터분야 발전 속도가 빠르다? (2/3)

다른 분야와 얼추 비슷: $10^7 \sim 10^9$배 발전/100년

- 에너지분야: 10^7배/100년 발전
  10^{-1}hp/hr (하인) $\longrightarrow$ 1.35×10^6 hp/hr (원전 1GW/hr)

- 교통분야: 10^9배/100년 발전
  10^2j (가마) $\longrightarrow$ 10^{11}j (Delta II)

컴퓨터분야 발전 속도가 빠르다? (3/3)

- 하드웨어: 다른 분야와 비슷한 발전
- 소프트웨어: 많이 미개함
  - 크기만 증가:
    휴대폰 100만줄
    아래아한글 200만줄
    스마트폰 1000만줄
    윈도우 3000만줄
- 소프트웨어: 다른 분야가 이루어 놓은 것을 아직 이루지 못했슴

무엇일까?

다른 분야의 현재 수준

- 제대로 작동할 지를 미리 검증할 수 없는 기계 설계는 없다.
- 제대로 작동할 지를 미리 검증할 수 없는 회로 설계는 없다.
- 제대로 작동할 지를 미리 검증할 수 없는 공장 설계는 없다.
- 제대로 서있을 지를 미리 검증할 수 없는 건축 설계는 없다.

뉴튼방정식, 미적분 방정식, 통계역학, 무슨무슨 방정식...

모든 엔지니어링의 근본 질문

우리가 만든 것이
우리가 의도한대로 움직인다는 것을
어떻게 미리
확인할 수 있는가?

(사실, 일상에서도 잦은 질문과 답: 입학시험, 입사시험, 궁합, 클럽관리)

이 근본 질문의 답이

컴퓨터 소프트웨어에 대해서는 어떤가?

아직 없다.

소프트웨어 분야의 근본 문제

작성한 SW의 오류를
자동으로 미리 모두 찾아주거나,
없으면 없다고 확인해주는
기술들은 있는가?

그래서, SW의 오류때문에 발생하는
개인/기업/국가/사회적 비용을
절감시켜주는 기술들은 있는가?

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

SW 오류(bug)(1/3)

- 소프트웨어에 있는 오류
- 소프트웨어가 생각대로 실행되지 않는 것
- 사람이 소프트웨어를 잘못 만들었기 때문
  - 천재지변이 아님

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

SW 오류(2/3)

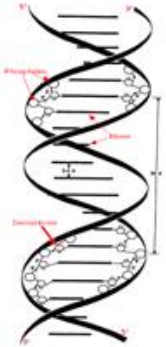
어머니가 전해준 슈퍼마켓 심부름 목록(프로그램)

1. 우유 1리터 2병
2. 우유가 없으면 오렌지쥬스 1리터 3병
3. 우유가 있으면 식빵 500그람 1봉
4. 쌀과 자두

만약에: 우유 1리터 1병만 있으면? 쌀과 자두? 쌀과자두?

SW 오류 (3/3)

DNA 오류



SW 오류

⇓

살다가 병이든다
사회적 비용
해결책 = 경제적기회

⇓

SW로 동작하는 제품이 고장난다
사회적 비용
해결책 = 경제적기회

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

무결점 SW의 시장가치

점점 막대해지는 SW 오류의 비용(개인/기업/사회/국가)

- 미국: $\geq$ 60조원/년
- 모든 제품 경쟁력 $\approx$ SW 품질
  - 가전, 기계, 통신, 교통, 국방, 의료, 에너지
  - 제품 리콜, 사회 혼란, 국가 손실

SW 제작중 오류수정에 쓰는 비용: ~20조원/년

- 2010 SW 시장: 310조원

경쟁: 더 결점이 적게, 더 적은 비용으로!

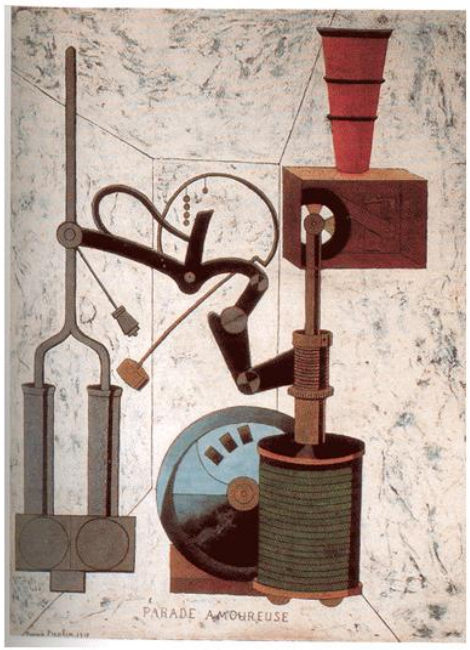
ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

SW 오류 검증기술의 변천사

- 1세대 기술(- '80s): 소프트웨어 소스 문법 오류 검증
- 2세대 기술(- '90s): 소프트웨어 소스 타입 오류 검증
- 3세대 기술(- future): 소프트웨어 소스 성질 오류 검증

SW 오류 잡는 기술: 미개한 현재

- 주먹구구식: 테스트, 소스읽기
- 성능:
  - AT&T 전화시스템 소프트웨어 생산성 = 10줄/1달(1996)
  - ETRI: 2달 걸려서 1글자 오류를 찾다(2000)



ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

SW 오류: 사람이 손으로 손쉽게 잡을 수 없다

- 엄청나게 커지고 복잡해 진 소프트웨어
- 새로운 컴퓨팅 환경: 지구 = 컴퓨터
  - 전지구적 네트웍을 타고
  - 불특정 다수의 코드가 내게로 온다(스마트폰 게임/apps)

자동 기술이 필요

SW 오류 잡는 기술의 꿈

SW 소스(C, assembly, binary 등)를 자동 분석

"소프트웨어 MRI" "소프트웨어 fMRI" "소프트웨어 PET"

우리 기술(예)

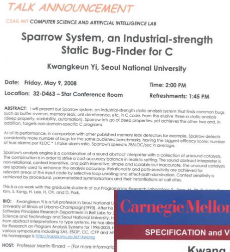
산업화된 기술:

Sparrow
The Early Bird

대상: C, memory leak/buffer overrun, 오류 검출률 6/KLOC, 속도 100Loc/sec

Memory leak detection (SPEC2000 and open sources) (as of 01/04/2008)

Programs	Size KLOC	Time (sec)	True Alarms	False Alarms
art	1.2	0.68	1	0
equake	1.5	1.03	0	0
mcf	1.9	2.77	0	0
bzip2	4.6	1.52	1	0
gzip	7.7	1.56	1	4
parser	10.9	15.93	0	0
ammp	13.2	9.68	20	0
vpr	16.9	7.85	0	9
crafty	19.4	84.32	0	0
twolf	19.7	68.80	5	0
mesa	50.2	43.15	9	0
vortex	52.6	34.79	0	1
gap	59.4	31.03	0	0
gcc	205.8	1330.33	44	1
gnuchess-5.07	17.8	9.44	4	0
tcl8.4.14	17.9	266.09	4	4
hanterm-3.1.6	25.6	13.66	0	0
sed-4.0.8	26.8	13.68	29	31
tar-1.13	28.3	13.88	5	3
grep-2.5.1a	31.5	22.19	2	3
openssh-3.5p1	36.7	10.75	18	4
bison-2.3	48.4	48.60	4	1
openssh-4.3p2	77.3	177.31	1	7
fftw-3.1.2	184.0	15.20	0	0
httpd-2.2.2	316.4	102.72	6	1
net-snmp-5.4	358.0	201.49	40	20
binutils-2.13.1	909.4	712.09	228	25


TALK ANNOUNCEMENT
CSAIL MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LAB

Sparrow System, an Industrial-strength Static Bug-Finder for C

Kwangkeun Yi, Seoul National University

Date: Friday, May 9, 2008 Time: 2:00 PM
Location: 32-D463 – Star Conference Room Refreshments: 1:45 PM

ABSTRACT: I will present our Sparrow system, an industrial-strength static analysis system that finds common bugs such as buffer overrun, memory leak, unll dereference, etc. in C code. From the elusive three in static analysis (deep property, scalability, automation), Sparrow lets go of deep properties, yet achieves the other two and, in addition, targets non-domain-specific C programs.

As of its performance, in comparison with other published memory leak detectors for example, Sparrow detects consistently more number of bugs for the same published benchmarks, having the biggest efficacy score: number of true alarms per KLOC * 1/false alarm ratio. Sparrow's speed is 785LOC/sec in average.

Sparrow's analysis engine is a combination of a sound abstract interpreter with a collection of unsound catalysts. The combination is in order to strike a cost-accuracy balance in realistic setting. The sound abstract interpreter is non-relational, context insensitive, and path insensitive: simple and scalable but inaccurate. The unsound catalysts are sparsely used to enhance the analysis accuracy. Relationality and path-sensitivity are achieved for relevant areas of the input code by selective loop unrolling and effect-path-domination. Context sensitivity is achieved by procedural, parameterized summarizations and their instantiations at call cites.

This is a co-work with the graduate students of our Programming Research Laboratory ... Kim, S. Kong, H. Lee, H. Oh, and D. Park.

BIO: Kwangkeun Yi is a full professor in Seoul National U... University of Illinois at Urbana-Champaign(1993). After his ... Software Principles Research Department in Bell Labs for ... Science and Technology and Seoul National University. H... from abstract interpretations to type systems. He has dire... for Research on Program Analysis Systems for 1998-2003. ... various symposiums including SAS, ESOP, CC, ICFP and AP... His homepage is http://ropas.snu.ac.kr/~kwang

HOST: Professor Martin Rinard – (For more informatio...


Carnegie Mellon SCHOOL OF COMPUTER SCIE...
SPECIFICATION and VERIFICATION CENTER

Kwang Keun Yi
Seoul National University
School of Computer Science & Computer Engineering

Title: Sparrow System, an Industrial-Stren...
Static Bug-Finder for C

Wean Hall - WEH7220
Friday, February 15, 2008
2:00 PM

Abstract:
I will present our Sparrow system, an industrial-strength static analysis system that finds co... bugs such as buffer overrun, memory leak, unll dereference, etc. in C code. From the elusive ... static analysis (deep property, scalability, automation), Sparrow lets go of deep properties, y... achieves the other two and, in addition, targets non-domain-specific C programs.

As of its performance, in comparison with other published memory leak detectors for examp... Sparrow detects consistently more number of bugs for the same published benchmarks, hav... biggest efficacy score: number of true alarms per KLOC * 1/false alarm ratio. Sparrow's spee... 785LOC/sec in average.

Sparrow's analysis engine is a combination of a sound abstract interpreter with a collection ... unsound catalysts. The combination is in order to strike a cost-accuracy balance in realistic ... The sound abstract interpreter is non-relational, context insensitive, and path insensitive: sin... scalable but inaccurate. The unsound catalysts are sparsely used to enhance the analysis ac... Relationality and path-sensitivity are achieved for relevant areas of the input code by selectiv... unrolling and effect-path-domination. Context sensitivity is archived by procedural, paramete... summarizations and their instantiations at call cites.

EC center
...ysis for Error-free Computing
...EF ERC

컴퓨터 연구의 한 역사

"프로그램에 오류가 없는지
확인해 주는 프로그램을 만들어보자"

혹은

"프로그램의 오류를
자동으로 찾아주는 프로그램을 만들어보자"

의 역사

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

SW 오류 잡는 기술: 1세대

문법 검증기: 70년대에 완성된 기술. lexical analyzer & parser

오류 = 소프트웨어의 생김새가 틀린 것

1. 우유 1리터 2병
2. 우유가 없으면 오렌지주스 1리터 3병
3. 있으면 빵식 우유가 500그람 1봉
4. 쌀과 자두

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

SW 오류 잡는 기술: 2세대

타입 검증(type checking): 90년대에 완성.
(30년간 프로그래밍언어 분야의 대표 성과)

오류 = 생긴것은 멀쩡한데, 잘못된 값이 계산에 흘러드는 경우

1. 우유를 담은 얼음을 가스 불위에 올려놓고 데운다
2. 식빵을 유리접시에 담고 방아로 빻는다
3. 2의 접시에 1의 우유를 튀긴다
4. 남자와 소나무를 결혼시킨다

SW 오류 잡는 기술: 3세대

프로그램분석검증: 아직 미완성
(static analysis, verification, model checking, ...)
오류 = 생긴것도 멀쩡하고 타입도 맞지만, 바라는 바가 아닌 것
오류 = 필요한 조건을 만족시킬 수 없는 경우

1. 우유를 담은 냄비를 가스 불위에 데운다
2. 식빵을 스텐접시에 담고 방아로 빻는다
3. 남자1명과 여자1명을 결혼 시킨다

- 가스불이 포항제철 용광로가 된다면?
- 방아가 현대중공업의 프레스가 된다면?
- 남녀 나이의 차가 100살이된다면?

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

Big Challenge: 오류잡는기술 제 3세대

- 소프트웨어는 오류가 없어야한다
- 소프트웨어는 끊임없이 크고 복잡해 진다
- 소프트웨어의 오류를 자동을 찾아주는 기술이 필요하다
- 이 기술의 현재수준은 이제 겨우 2살
- 이 기술 발전에 필요한 인재:
  - 수학/논리학/통계학/컴퓨터과학/컴퓨터공학의 긴밀하고 깊이있는 교류와 공부가 필요

컴퓨터과학은
머리로 궁리해서(abstraction) 기계로 돌리기(mechanization)?

Computer Science is
about mechanization of abstraction?

예) 오류자동 검증기술: 어떻게해야 소프트웨어에 있는 실수를 자동으로 찾아낼까?

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

컴퓨터과학은
머리로 궁리하는 것(intelligence)에 대한 연구(science)이고
그 응용은 인간 지능의 확장?

Computer Science is
the science of intelligence
and its application is extensions of human intelligence?

예) 오류자동 검증기술: 사람이 하는 일을 자동으로 해 낼 수 있을까?

감사합니다.

QnA