(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2022년12월30일
(11) 등록번호 10-2483489
(24) 등록일자 2022년12월28일

(51) 국제특허분류(Int. Cl.)
G06F 11/36 (2006.01) *G06N 20/00* (2019.01)
(52) CPC특허분류
G06F 11/3676 (2013.01)
G06F 11/3608 (2013.01)
(21) 출원번호 10-2022-0037001
(22) 출원일자 2022년03월25일
심사청구일자 2022년03월25일
(56) 선행기술조사문헌
KR1020200144051 A*
KR102304861 B1*
KR102353190 B1*
김윤삼, 김문주, "Concolic 테스팅과 Fuzzing을 결합한 유닛 테스팅 자동화 기술", 한국정보과학회, 한국 소프트웨어공학 학술대회(KCSE2022),(2022.1.20.)*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
연세대학교 산학협력단
서울특별시 서대문구 연세로 50 (신촌동, 연세대학교)
(72) 발명자
권태경
서울특별시 강남구 선릉로 221, 410동 1602호
조민기
서울특별시 서대문구 증가로 26, 302호
(뒷면에 계속)
(74) 대리인
권성현, 유광철, 백두진, 강일신, 김정연

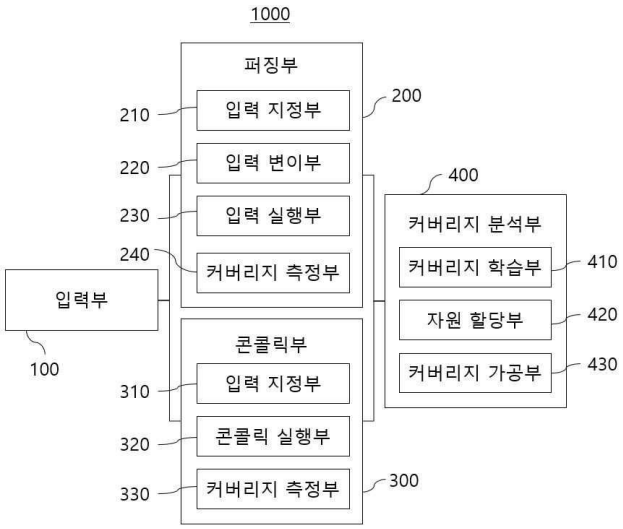
전체 청구항 수 : 총 1 항 심사관 : 장지혜

(54) 발명의 명칭 **동적 자원 분배가 가능한 하이브리드 퍼징 장치**

(57) 요 약

본 발명의 하이브리드 퍼징 장치는 소프트웨어에 대해 퍼징(fuzzing)을 실행하도록 구성된 퍼징부; 상기 소프트웨어에 대해 콘콜릭(concolic)을 실행하도록 구성된 콘콜릭부; 및 상기 퍼징부로부터 상기 소프트웨어의 퍼징 실행 결과에 대한 제1 커버리지 정보 및 상기 콘콜릭부로부터 상기 소프트웨어의 콘콜릭 실행 결과에 대한 제2 커버리지 정보를 수신하는 커버리지 분석부를 포함할 수 있다.

대 표 도 - 도1



(52) CPC특허분류
G06F 11/3624 (2013.01)
G06N 20/00 (2021.08)
(72) 발명자
안도현
대구광역시 수성구 달구벌대로 3280-1, 204동 601호

진호용
서울특별시 서대문구 연대동문길 113, 메이즈동 108호

이 발명을 지원한 국가연구개발사업

과제고유번호	1711152556
과제번호	2018-0-00513-005
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보보호핵심원천기술개발(R&D,정보화)
연구과제명	기계학습을 활용한 UNIX 기반 커널 취약점 탐지 자동화 연구
기 여 율	1/1
과제수행기관명	연세대학교 산학협력단
연구기간	2022.01.01 ~ 2022.12.31

명 세 서

청구범위

청구항 1

소프트웨어에 대해 퍼징(fuzzing)을 실행하도록 구성된 퍼징부;

상기 소프트웨어에 대해 콘콜릭(concolic)을 실행하도록 구성된 콘콜릭부;

상기 퍼징부로부터 상기 소프트웨어의 퍼징 실행 결과에 대한 제1 커버리지 정보 및 상기 콘콜릭부로부터 상기 소프트웨어의 콘콜릭 실행 결과에 대한 제2 커버리지 정보를 수신하는 커버리지 분석부;

상기 소프트웨어의 실행에 대한 입력값을 제공하는 입력부; 및

상기 퍼징부 또는 상기 콘콜릭부로부터 상기 소프트웨어의 실행에 대한 크래시 정보를 획득하는 크래시 관리부를 포함하고,

상기 커버리지 분석부는 상기 제1 커버리지 정보 및 상기 제2 커버리지 정보에 기초하여 머신 러닝을 수행하는 커버리지 학습부를 포함하고,

상기 커버리지 분석부는 상기 커버리지 학습부의 결과값에 기초하여 상기 소프트웨어에 대해 퍼징을 실행할지 또는 콘콜릭을 실행할지 여부를 결정하고,

상기 결과값은 상기 소프트웨어에 대한 퍼징 점수 및 콘콜릭 점수를 포함하고,

상기 제1 커버리지 정보 또는 상기 제2 커버리지 정보는 상기 소프트웨어에 포함된 복수의 파트 및 상기 복수의 파트 각각에 대응되는 복수의 파트값을 포함하고,

상기 커버리지 학습부는 상기 복수의 파트값에 기초하여 상기 복수의 파트 각각에 대한 퍼징 점수 및 콘콜릭 점수를 산출하고,

상기 커버리지 분석부는, 상기 소프트웨어의 제1 파트에 대한 퍼징 점수가 콘콜릭 점수보다 크거나 같을 경우 상기 제1 파트의 퍼징 실행과 관련된 제1 신호를 상기 퍼징부에 전송하고, 상기 제1 파트에 대한 퍼징 점수가 콘콜릭 점수보다 작은 경우 상기 제1 파트의 콘콜릭 실행과 관련된 제2 신호를 상기 콘콜릭부에 전송하는 자원 할당부를 포함하고,

상기 복수의 파트는 상기 소프트웨어의 소스 코드의 함수, 라인, 블록, 결정, 엣지, 분기 및 조건 중 적어도 하나에 의해 구별되고,

상기 커버리지 분석부는, 사용자가 인식가능한 문자 데이터를 출력하기 위해 상기 제1 커버리지 정보 또는 상기 제2 커버리지 정보를 처리하는 커버리지 가공부를 포함하고,

상기 크래시 관리부는 상기 퍼징부 또는 상기 콘콜릭부로부터 크래시 발생에 대한 신호를 수신한 경우, 크래시 발생에 대응되는 콜스택(call stack), 메모리 정보 및 상기 입력부로부터 제공받은 입력값을 저장하는

하이브리드 퍼징 장치.

청구항 2

삭제

청구항 3

삭제

청구항 4

삭제

청구항 5

삭제

청구항 6

삭제

청구항 7

삭제

청구항 8

삭제

청구항 9

삭제

청구항 10

삭제

청구항 11

삭제

발명의 설명

기 술 분 야

[0001] 본 발명은 하이브리드 퍼징 장치에 관한 것으로, 보다 상세하게는, 소프트웨어에 따라 동적 자원 분배가 가능한 하이브리드 퍼징 장치에 관한 것이다.

배 경 기 술

[0002] 소프트웨어에 존재하는 버그는 단순히 프로그램의 기능을 방해할 뿐만 아니라 악의적인 공격자를 통해 사용자의 장치를 공격할 수 있다. 그러나, 개발 단계에서 소프트웨어에 존재하는 모든 버그를 찾는 것은 현실적으로 어려우므로, 개발 단계에서 찾지 못한 소프트웨어의 버그를 찾기 위해 다양한 정적 및 동적 분석 기술이 사용되고 있다.

[0003] 버그를 찾기 위한 동적 분석 방법에는 퍼징(fuzzing)과 콘콜릭(concolic) 실행이 있다. 퍼징은 속도가 빨라 수많은 입력값들을 짧은 시간 내에 테스트할 수 있다는 장점이 있다. 그러나, 퍼징은 입력값을 무작위로 생성하거나 정해진 규칙대로 주어진 입력값을 변형시키기 때문에, 조건이 복잡하거나 한정적인 분기문에는 진입하기 어렵다는 단점이 있다. 반대로, 콘콜릭 실행은 복잡한 연산을 통해 입력값을 생성하여, 진입 조건이 복잡하거나 한정적인 분기를 진입할 수 있다는 장점이 있다. 그러나, 콘콜릭 실행은 퍼징에 비해 속도가 매우 느려 콘콜릭 실행을 단일 분석 방법으로 사용하는 것은 비효율적인 문제가 있다.

[0004] 본원 발명은 효율적인 소프트웨어의 동적 테스팅을 위해, 퍼징과 콘콜릭 실행을 같이 사용하는 하이브리드 퍼징에 관한 것이다. 본원 발명은 고정된 자원에 대하여 테스트 대상인 소프트웨어의 특성에 따라 퍼징 또는 콘콜릭에 대해 동적 자원 분배가 가능한 하이브리드 퍼징 기술에 대한 것이다.

발명의 내용

해결하려는 과제

[0005] 본 발명의 일 과제는 동적 자원 분배가 가능한 하이브리드 퍼징 장치에 관한 것이다.

과제의 해결 수단

[0006] 일 실시예에 따른 하이브리드 퍼징 장치는 소프트웨어에 대해 퍼징(fuzzing)을 실행하도록 구성된 퍼징부; 상기 소프트웨어에 대해 콘콜릭(concolic)을 실행하도록 구성된 콘콜릭부; 및 상기 퍼징부로부터 상기 소프트웨어의

퍼징 실행 결과에 대한 제1 커버리지 정보 및 상기 콘콜릭부로부터 상기 소프트웨어의 콘콜릭 실행 결과에 대한 제2 커버리지 정보를 수신하는 커버리지 분석부를 포함할 수 있다.

[0007] 여기서, 상기 커버리지 분석부는 상기 제1 커버리지 정보 및 상기 제2 커버리지 정보에 기초하여 머신 러닝을 수행하는 커버리지 학습부를 포함할 수 있다.

[0008] 여기서, 상기 커버리지 분석부는 상기 커버리지 학습부의 결과값에 기초하여 상기 소프트웨어에 대해 퍼징을 실행할지 또는 콘콜릭을 실행할지 여부를 결정할 수 있다.

[0009] 여기서, 상기 결과값은 상기 소프트웨어에 대한 퍼징 점수 및 콘콜릭 점수를 포함할 수 있다.

[0010] 여기서, 상기 제1 커버리지 정보 또는 상기 제2 커버리지 정보는 상기 소프트웨어에 포함된 복수의 파트 및 상기 복수의 파트 각각에 대응되는 복수의 파트값을 포함할 수 있다.

[0011] 여기서, 상기 커버리지 학습부는 상기 복수의 파트값에 기초하여 상기 복수의 파트 각각에 대한 퍼징 점수 및 콘콜릭 점수를 산출하고, 상기 커버리지 분석부는, 상기 소프트웨어의 제1 파트에 대한 퍼징 점수가 콘콜릭 점수보다 크거나 같을 경우 상기 제1 파트의 퍼징 실행과 관련된 제1 신호를 상기 퍼징부에 전송하고, 상기 제1 파트에 대한 퍼징 점수가 콘콜릭 점수보다 작은 경우 상기 제1 파트의 콘콜릭 실행과 관련된 제2 신호를 상기 콘콜릭부에 전송하는 자원 할당부를 포함할 수 있다.

[0012] 여기서, 상기 복수의 파트는 상기 소프트웨어의 소스 코드의 함수, 라인, 블록, 결정, 엣지, 분기 및 조건 중 적어도 하나에 의해 구별될 수 있다.

[0013] 여기서, 상기 커버리지 분석부는, 사용자가 인식가능한 문자 데이터를 출력하기 위해 상기 제1 커버리지 정보 또는 상기 제2 커버리지 정보를 처리하는 커버리지 가공부를 포함할 수 있다.

[0014] 여기서, 상기 소프트웨어의 실행에 대한 입력값을 제공하는 입력부; 및 상기 퍼징부 또는 상기 콘콜릭부로부터 상기 소프트웨어의 실행에 대한 크래시 정보를 획득하는 크래시 관리부를 더 포함하고, 상기 크래시 관리부는 상기 퍼징부 또는 상기 콘콜릭부로부터 크래시 발생에 대한 신호를 수신한 경우, 크래시 발생에 대응되는 콜스택(call stack), 메모리 정보 및 상기 입력부로부터 제공받은 입력값을 저장할 수 있다.

[0016] 일 실시예에 따른 하이브리드 퍼징 방법은 적어도 하나 이상의 프로세서에 의해 수행되는 하이브리드 퍼징 방법에 있어서, 소프트웨어에 대해 퍼징을 실행하는 단계; 상기 소프트웨어에 대해 콘콜릭을 실행하는 단계; 상기 소프트웨어의 퍼징 실행 결과에 대한 제1 커버리지 정보 및 상기 소프트웨어의 콘콜릭 실행 결과에 대한 제2 커버리지 정보를 수신하는 단계; 및 상기 제1 커버리지 정보 및 상기 제2 커버리지 정보에 기초하여 머신 러닝을 수행하는 단계를 포함할 수 있다.

[0017] 여기서, 상기 하이브리드 퍼징 방법을 실행시키도록 컴퓨터로 판독 가능한 기록 매체에 저장된 컴퓨터 프로그램이 제공될 수 있다.

발명의 효과

[0018] 본 발명의 일 실시예에 따르면 동적 자원 분배가 가능한 하이브리드 퍼징 장치가 제공될 수 있다.

도면의 간단한 설명

[0019] 도 1은 일 실시예에 따른 하이브리드 퍼징 장치의 블록도이다.

도 2는 일 실시예에 따른 하이브리드 퍼징 장치의 자원 할당 방법의 순서도이다.

도 3은 일 실시예에 따른 커버리지 정보를 설명하기 위한 도면이다.

도 4는 일 실시예에 따른 커버리지의 가공을 설명하기 위한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0020] 본 명세서에 기재된 실시예는 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에게 본 발명의 사상을 명확히 설명하기 위한 것이므로, 본 발명이 본 명세서에 기재된 실시예에 한정되는 것은 아니며, 본 발명의 범위는 본 발명의 사상을 벗어나지 아니하는 수정예 또는 변형예를 포함하는 것으로 해석되어야 한다.

[0021] 본 명세서에서 사용되는 용어는 본 발명에서의 기능을 고려하여 가능한 현재 널리 사용되고 있는 일반적인 용어

를 선택하였으나 이는 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자의 의도, 판례 또는 새로운 기술의 출현 등에 따라 달라질 수 있다. 다만, 이와 달리 특정한 용어를 임의의 의미로 정의하여 사용하는 경우에는 그 용어의 의미에 관하여 별도로 기재할 것이다. 따라서 본 명세서에서 사용되는 용어는 단순한 용어의 명칭이 아닌 그 용어가 가진 실질적인 의미와 본 명세서의 전반에 걸친 내용을 토대로 해석되어야 한다.

[0022] 본 명세서에 첨부된 도면은 본 발명을 용이하게 설명하기 위한 것으로 도면에 도시된 형상은 본 발명의 이해를 돕기 위하여 필요에 따라 과장되어 표시된 것일 수 있으므로 본 발명이 도면에 의해 한정되는 것은 아니다.

[0023] 본 명세서에서 본 발명에 관련된 공지의 구성 또는 기능에 대한 구체적인 설명이 본 발명의 요지를 흐릴 수 있다고 판단되는 경우에 이에 관한 자세한 설명은 필요에 따라 생략하기로 한다.

[0025] 버그 탐색은 소프트웨어의 실행뿐만 아니라 장치의 보완과도 관련이 있는 중요한 작업이다. 버그 탐색을 위한 동적 분석 방법 중 퍼징(fuzzing)은 입력값을 무작위로 생성하거나 주어진 입력값을 변형하여 소프트웨어의 비정상적인 동작을 탐지하는 동적 분석 방법이다. 이때, 소프트웨어의 비정상적인 동작이란, 개발자가 고려한대로 소프트웨어가 실행되지 않고 버그가 발생하는 것을 의미한다. 예를들어, 비정상적인 동작은 예상치 못한 값이 출력되는 등의 동작이나, 비정상 종료를 하는 등의 동작일 수 있다.

[0026] 퍼징은 속도가 빨라 수많은 입력값들을 짧은 시간 내에 테스트할 수 있다는 장점이 있다. 반면, 퍼징은 입력값을 무작위로 생성하거나 정해진 규칙대로 주어진 입력값을 변형시키기 때문에, 조건이 복잡하거나 한정적인 분기문에는 진입하기 어렵다는 단점이 있다.

[0027] 예를 들어, 테스트 대상인 소프트웨어에 입력을 통해 조작할 수 있는 4바이트 크기의 정수형 변수 x가 존재하고, 변수 x에 들어있는 값이 0xdeadbeef일 때, 진입할 수 있는 분기가 있다고 가정한다. 퍼징은 무작위로 입력값을 변형시키기 때문에, 해당 분기에 진입하기 위해서는 무작위로 뽑은 4바이트 입력값이 우연히 0xdeadbeef가 되어야 한다. 그러나, 무작위로 뽑은 4바이트 값이 정확히 0xdeadbeef일 확률은 2^(-32)으로 매우 낮기 때문에, 퍼징을 통해 해당 분기를 테스트하는 것은 매우 힘들다.

[0028] 반대로, 또 다른 동적 분석 방법인 콘콜릭 실행은 복잡한 연산을 통해 입력값을 생성하여, 진입 조건이 복잡하거나 한정적인 분기를 진입할 수 있는 장점이 있다. 그러나, 콘콜릭 실행은 퍼징에 비해 속도가 매우 느려, 콘콜릭 실행을 단일 분석 방법으로 사용하는 것은 비효율적이다.

[0029] 하이브리드 퍼징은 퍼징 또는 콘콜릭 실행의 단일 실행의 단점을 보완하기 위해, 퍼징과 콘콜릭 실행을 같이 사용하는 퍼징 전략이다. 퍼징을 통해 복잡하지 않은 분기를 빠르게 탐색하고, 퍼징으로 탐색하기 힘든 복잡한 분기는 콘콜릭 실행을 통해 탐색한다.

[0030] 하이브리드 퍼저를 사용할 때에는 사용자가 퍼징과 콘콜릭 실행 각각에 자원을 얼마나 할당할지를 미리 설정하고 퍼징을 수행하게 된다. 그러나, 테스트 대상 소프트웨어의 분기 또는 파트마다, 효율적인 탐색 방법이 다를 수 있다. 구체적으로, 한정적인 분기를 탐색할 때는 퍼징보다 콘콜릭 실행이 효율적이고, 많은 실행을 통해 테스트를 해야할 때는 콘콜릭 실행보다 퍼징이 효율적이다. 따라서, 퍼징과 콘콜릭 실행에 고정된 양의 자원을 할당 및 고정시켜 퍼징을 수행하면, 자원을 비효율적으로 사용하게 될 수 있다.

[0031] 종래에 하이브리드 퍼징을 빠르고 효율적으로 할 수 있는 QSYM이 도입되었다. 그러나, QSYM은 하이브리드 퍼징을 수행할 때, 퍼징과 콘콜릭 실행이 테스팅하는 소프트웨어에 따라, 보유하고 있는 테스트 입력에 따라, 진행 상황에 따라 각각의 효율이 달라진다는 점을 고려하지 않았다.

[0032] 구체적으로, QSYM에 대한 평가를 진행할 때에는 퍼징에 2개, 콘콜릭 실행에 1개의 코어를 할당하여, 총 3개의 코어를 사용하도록 설정했었다. 이때, 만약 퍼저가 복잡한 분기문을 만나 오랜 시간동안 진입하지 못하는 상황이 된다면, 퍼징보다 콘콜릭 실행에 자원을 할당해주는 것이 더 효율적이다. 그러나, QSYM은 여전히 처음에 설정된 퍼징에 2개, 콘콜릭 실행에 1개의 코어를 할당하여, 주어진 자원을 효율적으로 사용하지 못하게 된다.

[0033] QSYM은 퍼징 모듈과 경로 탐색 모듈을 이용해 하이브리드 퍼징을 진행하는 방법에 대해서만 다루고 있다. 즉, QSYM은 많은 컴퓨팅 자원을 각 장치에 어떻게 효율적으로 분배하여 사용할 것인지에 대한 내용은 다루고 있지 않다.

[0034] 다른 종래 기술로, 하이브리드 퍼징과 관련하여, 실행중인 복잡한 소프트웨어를 분석해 효율적으로 콘콜릭 실행을 해주는 콘콜릭 엔진인 Fuzzolic이 도입되었다. 또한, 기존에 비해 시간과 자원이 많이 들지만 정확한 정답을 계산해내는 SMT 솔버 대신, 근사값을 값싸게 계산해주는 Fuzzy-Sat을 사용하여 콘콜릭 실행을 더욱 빠르고 효율적으로 사용할 수 있는 방안도 존재했다. 그러나, 위 방안들도 콘콜릭 엔진과 퍼징에 자원을 어떻게 할당할 것

인지를 고려하지 않으므로, 주어진 자원을 효율적으로 활용하지 못한다는 한계가 존재한다.

[0035] 위 문제점은 퍼징의 시간이 길어질수록, 주어진 자원의 양이 많을수록 더 자주 발생하게 된다. 일반적으로 하이브리드 퍼저를 사용하여 소프트웨어 테스팅을 진행할 때, 수십개의 코어를 사용하게 된다. 따라서, 동적 자원 분배 방법 및 장치의 필요성이 실용적인 측면에서 더욱 필요한 실정이다.

[0036] 위 문제점을 극복하기 위해서는 퍼징을 수행하면서 퍼징과 콘콜릭 실행의 효율성을 실시간으로 파악하고 자원을 재할당하여야 한다. 그러나, 사용자가 이를 실시간으로 고려하여 자원을 할당하기에는 불가능하다. 따라서, 본원 발명은 실시간으로 퍼징과 콘콜릭 실행의 효율성을 파악하고, 동적으로 자원을 할당할 수 있는 기술에 대해 제안한다.

[0038] 도 1은 일 실시예에 따른 하이브리드 퍼징 장치의 블록도이다.

[0039] 도 1을 참조하면, 일 실시예에 따른 하이브리드 퍼징 장치(1000)는 입력부(100), 퍼징부(200), 콘콜릭부(300) 및 커버리지 분석부(400)를 포함할 수 있다. 하이브리드 퍼징 장치(1000)는 도 1의 도시에 한정되지 않고, 이보다 적거나 많은 구성 요소를 포함할 수 있다. 예를 들어, 하이브리드 퍼징 장치(1000)는 크래시 관리부(도시되지 않음)를 포함할 수 있다.

[0040] 도 1은 각각의 구성요소가 별도의 부서 또는 장치인 것으로 도시하였으나, 이에 한정되지 않고, 일부 구성요소가 하나의 장치로 구성될 수도 있다. 예를 들어, 퍼징부에 제1 입력부가 포함되거나, 콘콜릭부에 제2 입력부가 포함될 수도 있으나, 이에 한정되지 않는다.

[0041] 입력부(100)는 소프트웨어를 테스트하기 위한 입력값을 생성 및/또는 출력할 수 있다. 입력부(100)는 미리 저장된 입력 데이터에 기초하여 입력값을 출력하거나 랜덤 함수를 이용하여 무작위의 입력값을 생성 및/또는 출력할 수 있다. 또한, 입력부(100)는 미리 저장된 입력 데이터에 포함된 값들을 변형하여 새로운 입력값을 생성할 수도 있다.

[0042] 입력부(100)는 퍼징부(200) 또는 콘콜릭부(300)로부터 퍼징 또는 콘콜릭 실행에 사용된 입력값 중 유의미한 입력값을 입력풀에 저장할 수 있다. 이때, 유의미한 입력값이란, 기존 입력값들과 중복되지 않고 새로운 커버리지를 얻을 수 있는 입력값, 기존과 같은 커버리지를 같지만 간소화되어 실행 시간을 절약할 수 있는 입력값 등 소프트웨어 테스팅에 도움이 될 수 있는 입력값일 수 있다.

[0043] 입력부(100)의 입력풀에 저장된 입력값들은 퍼징부(200)에서 변이를 수행하기 위한 초기값으로 사용되거나, 콘콜릭부(300)의 테스트 입력값으로 사용될 수 있다.

[0044] 퍼징부(200)는 테스트 대상 소프트웨어에 대해 퍼징을 실행할 수 있다. 도 1의 일 실시예는 하이브리드 퍼징 장치(1000)에 퍼징부(200)가 포함되어 있는 것을 도시하였으나, 이에 한정되지 않고 하이브리드 퍼징 장치(1000)와 별도로 퍼징부(200)가 다른 장치로 존재할 수도 있다. 퍼징부(200)는 퍼징 장치로도 명명될 수 있다.

[0045] 퍼징부(200)는 입력 지정부(210), 입력 변이부(220), 입력 실행부(230) 및 커버리지 측정부(240)를 포함할 수 있다. 그러나, 이에 한정되지 않고 퍼징부(200)는 여러 요소에 의한 작업 수행이 아닌 하나의 프로세서에 의해 작업 수행을 할 수도 있다.

[0046] 퍼징부(200)는 입력부(100)로부터 입력값을 획득할 수 있다. 입력 지정부(210)는 입력부(100)로부터 획득한 입력값들 중 제1 입력값을 지정할 수 있다. 입력 변이부(220)는 입력 지정부(210)로부터 지정된 제1 입력값을 변이시켜, 새로운 입력인 제2 입력값을 생성할 수 있다. 구체적으로, 입력 변이부(220)는 제1 입력값에 대하여 비트 전환, 바이트 전환, 부분 치환, 부분 삭제, 바이트 추가 등을 이용하여 제2 입력값을 생성할 수 있다.

[0047] 퍼징부(200)의 입력 실행부(230)는 입력 변이부(220)로부터 제2 입력값을 수신하여 테스트 대상 소프트웨어에 대해 퍼징을 실행할 수 있다. 커버리지 측정부(240)는 퍼징을 실행하는 동안, 크래시 정보와 커버리지 정보를 생성할 수 있다.

[0048] 구체적으로, 크래시 정보는 테스트 대상 소프트웨어에서 발생하는 크래시 발생 여부 및 크래시 발생시의 콜스택, 메모리 정보 등을 포함할 수 있다. 이때, 크래시는 임의의 입력값으로 인해 테스트 대상 소트프웨어에 버그가 발생하여 비정상적으로 종료하게 되는 것을 의미하는 것일 수 있다. 크래시라 발생한다면, 이는 소프트웨어의 버그를 유발하는 입력값을 찾은 것을 의미하므로, 퍼징부(200)는 이에 대한 정보를 기록할 수 있다.

[0049] 또한, 커버리지 정보는 소프트웨어가 입력값에 따라 실행되는 흐름을 나타내는 정보를 의미하는 것일 수 있다. 구체적으로, 커버리지 정보는 테스트 대상 소프트웨어에 포함된 복수의 파트 및 상기 복수의 파트 각각에 대응

되는 복수의 파트값을 포함할 수 있다. 이때, 복수의 파트는 커버리지의 종류에 따라 달라질 수 있다. 예를 들어, 복수의 파트는 테스트 대상 소프트웨어의 소스 코드의 함수, 라인, 블록, 결정, 엣지, 분기 및 조건 중 적어도 하나에 의해 구별될 수 있다.

[0050] 구체적인 예를 들어, 소스 코드에 N개의 함수가 포함된 경우, 소프트웨어는 함수마다 파트를 나누어 N개의 파트를 포함할 수 있다. 또한, 소스 코드가 L줄의 라인으로 이루어진 경우, 소프트웨어는 라인마다 파트를 나누어 L개의 파트를 포함할 수 있다. 또한, 소스 코드에 B개의 블록이 포함된 경우, 소프트웨어는 블록마다 파트를 나누어 B개의 블록을 포함할 수 있다. 커버리지의 파트 및 파트값에 대한 구체적인 예시는 도 3을 참조하여 이하에서 설명한다.

[0051] 퍼징부(200)는 생성한 커버리지 정보를 커버리지 분석부(400)에 전송할 수 있다. 또한, 퍼징부(200)는 생성한 크래시 정보를 크래시 관리부(도시되지 않음)로 전송할 수 있다.

[0052] 콘콜릭부(300)는 테스트 대상 소프트웨어에 대해 콘콜릭을 실행할 수 있다. 도 1의 일 실시예는 하이브리드 퍼징 장치(1000)에 콘콜릭부(300)가 포함되어 있는 것을 도시하였으나, 이에 한정되지 않고 하이브리드 퍼징 장치(1000)와 별도로 콘콜릭부(300)가 다른 장치로 존재할 수도 있다. 콘콜릭부(300)는 콘콜릭 실행 장치로도 명명될 수 있다.

[0053] 콘콜릭부(300)는 입력 지정부(310), 콘콜릭 실행부(320) 및 커버리지 측정부(330)를 포함할 수 있다. 그러나, 이에 한정되지 않고 콘콜릭부(300)는 여러 요소에 의한 작업 수행이 아닌 하나의 프로세서에 의해 작업 수행을 할 수도 있다.

[0054] 콘콜릭부(300)는 입력부(100)로부터 입력값을 획득할 수 있다. 콘콜릭부(300)의 입력 지정부(310)는 입력부(100)로부터 획득한 입력값 및 테스트 대상 소트프웨어를 분석하여, 콘콜릭 실행에 적합한 테스트 입력값을 지정할 수 있다. 구체적으로, 입력 지정부(310)는 테스트 입력의 커버리지 정보와 테스트 대상 소프트웨어의 함수 호출 그래프를 통해, 테스트 입력값이 더 도달할 수 있는 하위 블록이나 분기의 수를 파악하고, 이를 기반으로 테스트 입력을 지정할 수 있다.

[0055] 콘콜릭부(300)의 콘콜릭 실행부(320)는 입력 지정부(310)로부터 테스트 입력값을 수신하여 테스트 대상 소프트웨어에 대해 콘콜릭을 실행할 수 있다. 커버리지 측정부(330)는 콘콜릭을 실행하는 동안, 크래시 정보와 커버리지 정보를 생성할 수 있다. 크래시 정보 및 커버리지 정보에 대한 내용은 퍼징부(200)의 크래시 정보 및 커버리지 정보의 설명과 중복될 수 있어, 자세한 내용은 생략한다.

[0056] 커버리지 분석부(400)는 퍼징부(200) 및 콘콜릭부(300)로부터 획득한 커버리지 정보에 기초하여, 테스트 대상 소프트웨어에 적합한 자원을 할당할 수 있다. 구체적으로, 커버리지 분석부(400)는 커버리지 정보를 분석하여, 소프트웨어의 특정 파트마다 동적인 자원 할당을 수행할 수 있다.

[0057] 커버리지 분석부(400)는 커버리지 학습부(410), 자원 할당부(420) 및 커버리지 가공부(430)를 포함할 수 있다.

[0058] 커버리지 분석부(400)는 퍼징부(200) 및 콘콜릭부(300)가 소프트웨어 테스팅을 진행할 때마다 생성되는 커버리지 정보를 실시간으로 수집할 수 있다. 구체적으로, 커버리지 분석부(400)는 테스트 대상 소프트웨어의 테스팅마다 퍼징부(200)로부터 제1 커버리지 정보 및 콘콜릭부(300)로부터 제2 커버리지 정보를 획득할 수 있다. 커버리지 정보의 획득은 커버리지 분석부(400)의 커버리지 학습부(410)가 수행할 수도 있다.

[0059] 커버리지 학습부(410)는 퍼징부(200) 및 콘콜릭부(300)로부터 획득한 커버리지 정보에 기초하여, 머신 러닝을 수행할 수 있다. 구체적으로, 커버리지 학습부(410)는 강화 학습 알고리즘을 이용하여 커버리지 정보를 학습할 수 있다. 그러나, 이에 한정되지 않고, 커버리지 학습부(410)는 강화 학습이 아닌 다른 머신 러닝 알고리즘을 사용할 수도 있다.

[0060] 커버리지 학습부(410)의 강화 학습 수행으로, 실행 시간 대비 새로 찾은 커버리지의 개수가 출력될 수 있다. 커버리지 학습부(410)는 소프트웨어 테스팅을 진행하는 동안 강화 학습 알고리즘에 의한 보상을 누적할 수 있다. 이때, 커버리지 학습부(410)는 누적된 보상을 일정 주기마다 초기화할 수 있다.

[0061] 커버리지 학습부(410)는 강화 학습을 통한 보상 누적으로 퍼징 점수 및 콘콜릭 점수를 산출할 수 있다. 즉, 커버리지 학습부(410)의 점수 산출을 기초로, 자원 할당부(420)는 점수가 높은 테스팅 수행에 대하여 자원을 할당할 수 있다. 이때, 퍼징 점수 또는 콘콜릭 점수는 실행 시간 대비 새로 찾은 커버리지의 수에 기초하여 산출될 수 있다.

[0062] 자원 할당부(420)는 커버리지 학습부(410)에서 학습한 모델을 바탕으로 테스팅이 끝나고 반납된 자원 또는 전체 자원을 퍼징과 콘콜릭 실행 각각에 얼마나 할당할지를 결정할 수 있다. 이때, 자원은 프로세서의 코어, 시간, 메모리, 저장 장치의 용량 등이 될 수 있다.

[0063] 구체적으로, 자원 할당부(420)는 퍼징과 콘콜릭 실행 각각에 최소 필요 자원과 최대 할당량에 제한을 둘 수 있다. 자원 할당부(420)는 강화 학습을 통해 누적된 보상과 새로 얻은 보상을 기반으로, 퍼징과 콘콜릭 실행 중 하나에 남아있는 자원으 모두 할당하거나, 각각의 자원 할당 비율을 결정할 수 있다. 이때, 남은 자원이란 할당되지 않았던 자원 또는 테스팅 수행이 종료되어 반환된 자원일 수 있다.

[0064] 예를 들어, 자원 할당부(420)는 소프트웨어의 제1 파트에 대한 퍼징 점수가 콘콜릭 점수보다 크거나 같을 경우, 상기 제1 파트에 대하여 퍼징 실행에 대한 자원을 할당할 수 있다. 또한 예를 들어, 자원 할당부(420)는 소프트웨어의 제2 파트에 대한 퍼징 점수가 콘콜릭 점수보다 작을 경우, 상기 제2 파트에 대하여 콘콜릭 실행에 대한 자원을 할당할 수 있다. 그러나, 퍼징 점수 또는 콘콜릭 점수의 상대적인 비교에 한정되지 않고, 사용자의 설정에 따라 점수의 절대값에 의해 자원의 퍼징 실행 또는 콘콜릭 실행이 정해질 수 있다.

[0065] 본원 발명의 하이브리드 퍼징 장치(1000)는 종래 퍼징 및 콘콜릭 실행에 초기 할당된 자원이 계속적으로 고정되어 비효율적인 자원 사용의 문제를 해결하기 위해, 실시간 커버리지 정보에 기초하여 동적인 자원 할당을 통해 자원을 효율적으로 사용할 수 있도록 한다.

[0066] 예를 들어, 5개의 코어가 존재할 경우, 종래는 테스팅 초기에 퍼징 실행에 3개, 콘콜릭 실행에 2개의 코어가 할당된 것을 가정한다. 종래에는 소프트웨어가 복잡한 분기문을 가졌는지 여부 등을 고려하지 않고, 3개의 코어는 퍼징 실행, 2개의 코어는 콘콜릭 실행에 할당되었다. 따라서, 복잡한 분기문을 가져 콘콜릭 실행이 유리한 소스 코드 파트에 대해서 3개의 코어는 계속적으로 퍼징을 실행했기 때문에, 제대로된 소프트웨어 테스트뿐만 아니라 효율적인 테스트가 어려웠다.

[0067] 그러나, 본원 발명의 하이브리드 퍼징 장치(1000)는 초기에 퍼징 실행에 3개, 콘콜릭 실행에 2개의 코어가 할당되었더라도, 한 번의 실행 이후 커버리지 분석을 통해 자원을 재분배할 수 있다. 따라서, 콘콜릭 실행이 유리한 소스 코드 파트에 대해서 초기에 설정된 2개의 코어뿐만 아니라, 남은 코어(사용되지 않았거나, 실행 이후 반환된 코어)도 콘콜릭 실행이 가능할 수 있다.

[0068] 본원 발명의 하이브리드 퍼징 장치(1000)는 크래시 관리부(도시되지 않음)를 포함할 수 있다. 크래시 관리부는 퍼징부(200) 또는 콘콜릭부(300)로부터 소프트웨어의 실행에 대한 크래시 정보를 포함할 수 있다. 이때, 크래시 정보는 크래시 발생 유무, 크래시 발생에 대응되는 콜스택(call stack), 메모리 정보 및 크래시를 발생시킨 입력값을 포함할 수 있다.

[0069] 입력부(100)로부터 획득하거나 변이된 입력이 크래시를 발생시킨다면, 이는 소프트웨어의 버그를 유발하는 입력값을 찾은 것을 의미하는 것일 수 있다. 따라서, 크래시 관리부는 이에 대한 정보를 기록하여 테스트 대상 소프트웨어에 대한 버그를 분석을 위한 크래시 정보를 저장할 수 있다.

[0071] 도 2는 일 실시예에 따른 하이브리드 퍼징 장치의 자원 할당 방법의 순서도이다.

[0072] 도 2를 참조하면, 일 실시예에 따른 자원 할당 방법은 초기 입력 지정 및 자원을 할당하는 단계(S110), 퍼징 및 콘콜릭 실행 단계(S120), 커버리지 정보를 획득하는 단계(S130), 커버리지에 대해 머신 러닝을 수행하는 단계(S140) 및 머신 러닝 결과값에 기초하여 자원을 할당하는 단계(S150)를 포함할 수 있다.

[0073] 초기 입력 지정 및 자원을 할당하는 단계(S110)는 커버리지 정보를 획득하기 위한 초기 테스트 실행을 위해, 초기 입력을 지정하고 자원을 할당하는 단계일 수 있다. 지정된 초기 입력은 미리 저장된 입력일 수도 있고, 무작위로 생성된 입력일 수도 있다.

[0074] 하이브리드 퍼징 장치(1000)의 프로세서는 초기에 퍼징과 콘콜릭 실행에 사용자의 설정에 따라 초기 자원을 할당할 수 있다. 예를 들어, 자원이 3개의 프로세서인 경우, 2개의 프로세서는 퍼징을, 1개의 프로세서는 콘콜릭 실행을 수행하도록 초기 자원을 할당할 수 있다. 이때, 퍼징과 콘콜릭 실행에는 같은 입력값이 사용될 수도 있고, 상이한 입력값이 사용될 수도 있다.

[0075] 퍼징 및 콘콜릭 실행 단계(S120)는 단계 S110에서 할당된 자원 및 입력값에 기초하여 퍼징부(200) 및 콘콜릭부(300)에 의해 테스트 대상 소프트웨어에 대한 퍼징 및 콘콜릭 실행이 수행되는 단계일 수 있다. 수행 과정에서 실시간으로 커버리지 정보가 생성되고, 생성된 커버리지 정보는 커버리지 분석부(400)로 전송될 수 있다.

[0076] 커버리지 정보를 획득하는 단계(S130)는 커버리지 분석부(400)가 퍼징부(200) 또는 콘콜릭부(300)로부터 실시간으로 커버리지 정보를 획득하는 단계일 수 있다.

[0077] 커버리지에 대해 머신 러닝을 수행하는 단계(S140)는 커버리지 학습부(410)가 퍼징부(200) 또는 콘콜릭부(300)로부터 획득한 커버리지 정보에 대하여 머신 러닝을 수행하는 단계일 수 있다. 구체적으로, 커버리지 학습부(410)는 커버리지 정보에 기초하여 강화 학습을 수행할 수 있다.

[0078] 머신 러닝 결과값에 기초하여 자원을 할당하는 단계(S150)는 자원 할당부(420)가 커버리지 학습부(410)의 강화 학습 결과에 기초하여 남은 자원을 퍼징부(200) 또는 콘콜릭부(300)로 할당하는 단계일 수 있다. 이때, 남은 자원은 단계 S110에서 초기 자원으로 할당되지 않은 자원이나, 테스팅 종료 후 반환된 자원을 의미하는 것일 수 있다.

[0079] 할당된 자원은 할당된 내용을 기초로 퍼징 및 콘콜릭 실행을 수행(S120)할 수 있다. 하이브리드 퍼징 장치(1000)는 퍼징 및 콘콜릭 실행에 대하여 계속적으로 커버리지 정보를 분석/학습하여 동적인 자원 할당을 수행한다.

[0080] 단계 S120 내지 S150의 과정은 사용자의 종료 요청 또는 저장 용량 등의 자원이 고갈되기 전까지 수행될 수 있다. 또한 해당 과정은 처음 퍼저를 실행할 때 사용자가 종료 시각 또는 타임 아웃을 설정해 놓았을 경우, 설정에 따라 종료될 수 있다.

[0082] 도 3은 일 실시예에 따른 커버리지 정보를 설명하기 위한 도면이다.

[0083] 도 3(a)는 일반적인 커버리지 정보의 예시를 나타낸 도면이고, 도 3(b)는 도 3(a)의 커버리지 정보가 간략화된 것을 나타낸 도면이다.

[0084] 도 3(a)을 참조하면, 커버리지 정보는 각 파트마다 몇 번 실행되었는지를 나타내는 파트값을 포함할 수 있다. 구체적인 예를 들어, 소스 코드에 B개의 블록이 포함된 경우, 소프트웨어는 B개의 파트를 포함한다. 이때, B개의 파트 중 제1 파트가 10번 실행되었고, 제2 파트가 10번 실행되었고, 제3 파트가 8번 실행된 경우, 커버리지 정보는 제1 파트에 대응되는 제1 파트값인 10, 제2 파트에 대응되는 제2 파트값인 8, 제3 파트에 대응되는 제3 파트값인 8을 포함할 수 있다.

[0085] 도 3(b)를 참조하면, 커버리지 정보는 파트값만을 포함하도록 간략화될 수 있다. 도 3(a)와 같은 커버리지 정보는 사용자가 보기에 간편하다는 장점이 있으나, 크기 및 속도에 대해 단점이 존재할 수 있다. 구체적으로, 도 3(a)는 파일에 기록해야 하는 정보가 많기 때문에 실행 속도가 느려지고, 프로그램이 커질수록 커버리지 정보를 저장하는 파일의 크기도 커지게 된다는 단점이 존재한다.

[0086] 도 3(b)은 커버리지 정보가 파트값만을 저장하는 예시로, 이를 통해 속도는 향상되고, 파일의 크기는 감소할 수 있다. 도 3(b)의 간략화된 커버리지 정보는 이를 해석해주는 별도의 도구를 통해 다시 도 3(a)와 같은 형태로 변형이 가능하다.

[0087] 커버리지 정보는 속도 향상 및 크기 감소를 위해 도 3(b)에서 더 나아가 사용자가 판단하기 힘든 문자(non-printable)를 포함하여 더 간략화될 수 있다. 이는 아래 도 4를 참조하여 설명한다.

[0089] 도 4는 일 실시예에 따른 커버리지의 가공을 설명하기 위한 도면이다.

[0090] 도 4(a)는 non-printable 문자를 포함하는 커버리지 정보의 예시를 나타내는 도면이고, 도 4(b)는 도 4(a)를 가공하여 생성된 printable 문자를 포함하는 커버리지 정보의 예시를 나타내는 도면이다.

[0091] 도 4(a)를 참조하면, 커버리지 정보는 효율적인 저장을 위해 바이트 단위 또는 비트 단위로 저장하기 때문에, 사람이 읽기 힘든 문자(non-printable 문자)를 포함할 수 있다. Non-printable 문자를 포함하는 커버리지 정보는 컴퓨팅 측면에서 매우 효율적일 수 있다. 그러나, 가끔 사용자의 요청으로 커버리지 정보 확인이 필요한 경우가 존재할 수 있다. 따라서, non-printable 문자를 사람이 확인할 수 있는 문자로 가공하는 작업이 필요하다.

[0092] 커버리지 분석부(400)는 non-printable 문자를 포함하는 커버리지 정보를 가공하기 위해 커버리지 가공부(430)를 포함할 수 있다. 커버리지 가공부(430)는 non-printable 문자를 포함하는 커버리지를 해석하는 도구를 포함하고, 해석 후 printable 문자를 생성하는 도구를 포함할 수 있다.

[0093] 도 4(b)를 참조하면, 커버리지 가공부(430)는 non-printable 문자를 포함하는 커버리지 정보를 가공하여 printable 문자를 포함하는 커버리지 정보를 생성할 수 있다. 따라서, 사용자는 커버리지 가공부(430)를 통해

커버리지 정보를 인식할 수 있다.

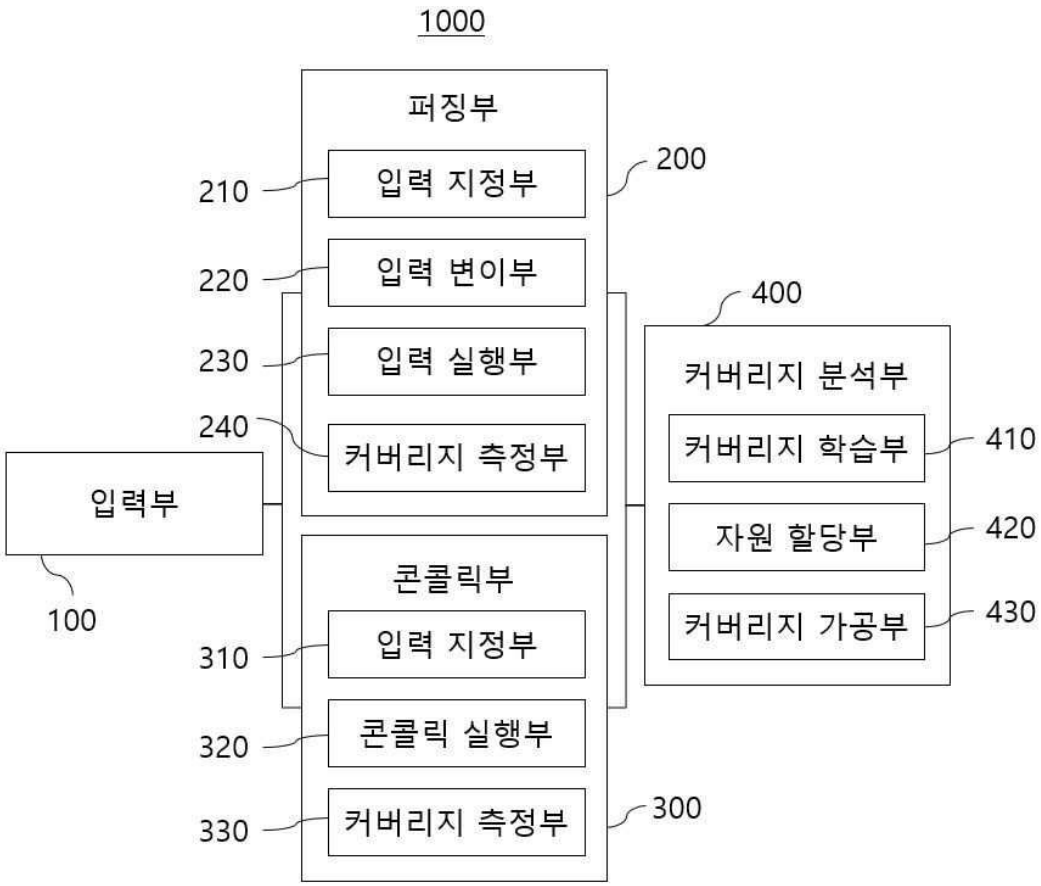
[0095] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다

[0096] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

[0097] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

도면

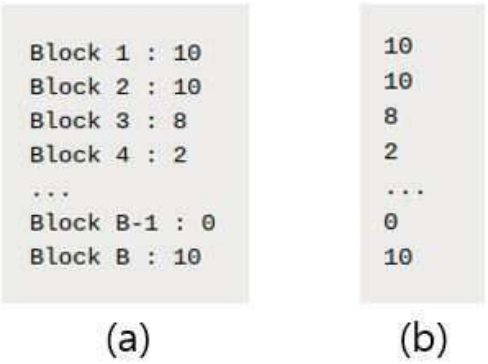
도면1



도면2


S110
초기 입력 지정 및 자원 할당
S120
퍼징 및 콘콜릭 실행
S130
커버리지 정보 획득
S140
커버리지에 대해 머신 러닝 수행
S150
머신 러닝 결과값에 기초하여 자원 할당

도면3


Block 1 : 10
Block 2 : 10
Block 3 : 8
Block 4 : 2
...
Block B-1 : 0
Block B : 10
10
10
8
2
...
0
10
(a)
(b)

도면4

(a)

```
File 'a.c'
Lines executed:87.50% of 8
Branches executed:100.00% of 6
Taken at least once:66.67% of 6
Calls executed:50.00% of 2
Creating 'a.c.gcov'
```

(b)