



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0081442
(43) 공개일자 2025년06월05일

(51) 국제특허분류(Int. Cl.)
G06F 21/14 (2013.01)

(52) CPC특허분류
G06F 21/14 (2013.01)
G06F 2221/2125 (2013.01)

(21) 출원번호 10-2023-0169791
(22) 출원일자 2023년11월29일
심사청구일자 2023년11월29일

(71) 출원인

충남대학교산학협력단

대전광역시 유성구 대학로 99 (궁동, 충남대학교)

(72) 발명자

조은선

대전광역시 유성구 어은로 57, 110동 1504호(어은동, 한빛아파트)

이석수

대전광역시 유성구 유성대로680번길 49, 초아빌 301호

(74) 대리인

이은철, 김재문

전체 청구항 수 : 총 6 항

(54) 발명의 명칭 E-graphs를 이용한 MBA식 간략화 시스템 및 방법

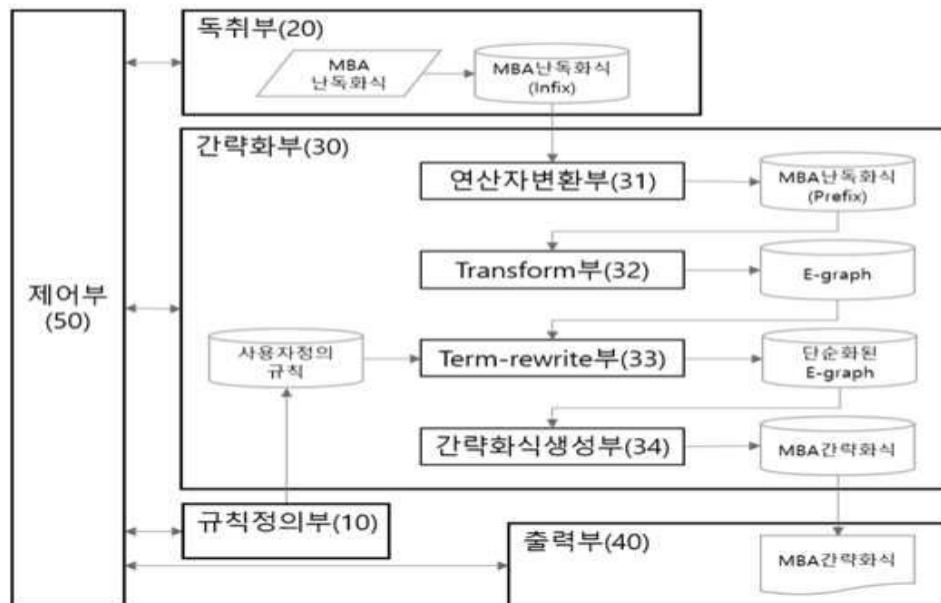
(57) 요약

본 발명은 악성코드 및 멀웨어 등 악의적인 행동을 하는 프로그램에 적용된 MBA 난독화 식을 코드 분석자들이 쉽게 분석할 수 있도록 하고 또한 무분별한 코드 난독화에 따른 부작용을 줄이고 코드 난독화의 선의적 이용을 도모하고자 MBA 기법으로 난독화된 식을 쉽고 간편하게 간략화할 수 있는 E-graph를 이용한 MBA식 간략화 시스템

(뒷면에 계속)

대표도 - 도1

MBA난독화식 간략화 시스템



및 방법에 관한 것으로,

본 발명의 E-graphs를 이용한 MBA식 간략화 시스템은, Term-rewrite에 적용될 연산 규칙들을 사용자가 정의하는 규칙정의부(10); 사용자로부터 MBA 난독화 식을 취득하는 독취부(20); E-graph를 이용한 Term-rewrite 기법으로 MBA식을 간략화하는 간략화부(30); 간략화된 MBA식을 사용자가 원하는 형태 및 수단으로 출력하는 출력부(40); 상기 규칙정의부(10), 독취부(20), 간략화부(30), 출력부(40)의 기능을 제어하는 제어부(50);를 포함하고,

상기 간략화부(30)는, 상기 독취부(20)에서 독취한 Infix 형태의 MBA 식을 Prefix 형태로 변환하는 연산자변환부(31); 상기 Prefix 형태로 변환된 MBA 식을 E-graph로 변환하는 Transform부(32); 사용자 정의 규칙에 따라 Term-rewrite 방식으로 E-graph를 간략화하는 Term-rewrite부(33); 상기 간략화된 E-graph로부터 MBA 간략화 식을 생성하는 간략화식생성부(34);를 포함함을 특징으로 한다.

이 발명을 지원한 국가연구개발사업

과제고유번호	1711193395
과제번호	2022-0-01200-002
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보통신방송혁신인재양성
연구과제명	융합보안대학원(충남대학교)
기 여 율	1/1
과제수행기관명	충남대학교 산학협력단
연구기간	2023.01.01 ~ 2023.12.31

명세서

청구범위

청구항 1

Term-rewrite에 적용될 연산 규칙들을 사용자가 정의하는 규칙정의부(10);

사용자로부터 MBA 난독화 식을 취득하는 독취부(20);

E-graph를 이용한 Term-rewrite 기법으로 MBA식을 간략화하는 간략화부(30);

간략화된 MBA식을 사용자가 원하는 형태 및 수단으로 출력하는 출력부(40);

상기 규칙정의부(10), 독취부(20), 간략화부(30), 출력부(40)의 기능을 제어하는 제어부(50);를 포함하는 E-graphs를 이용한 MBA식 간략화 시스템에 있어서,

상기 간략화부(30)는,

상기 독취부(20)에서 독취한 Infix 형태의 MBA 난독화 식을 Prefix 형태로 변환하는 연산자변환부(31);

상기 Prefix 형태로 변환된 MBA 난독화 식을 E-graph로 변환하는 Transform부(32);

사용자 정의 규칙에 따라 Term-rewrite 방식으로 E-graph를 간략화하는 Term-rewrite부(33);

상기 간략화된 E-graph로부터 MBA 간략화 식을 생성하는 간략화식생성부(34);를 포함함을 특징으로 하는 E-graphs를 이용한 MBA식 간략화 시스템.

청구항 2

제1항에 있어서,

상기 연산자변환부(31)는, 오픈소스 역공학 프레임워크인 Miasm을 활용하여 Infix 형태의 MBA 난독화 식을 Prefix 형태로 연산자를 변환함을 특징으로 하는 E-graphs를 이용한 MBA식 간략화 시스템.

청구항 3

제1항에 있어서,

상기 Transform부(32)는, Rust 언어 기반의 프레임워크인 Egg를 사용하여 Prefix 형태의 MBA식을 E-graph로 변환함을 특징으로 하는 E-graphs를 이용한 MBA식 간략화 시스템.

청구항 4

Term-rewrite에 적용될 연산 규칙들을 사용자가 정의하는 규칙정의단계(S10);

사용자로부터 MBA 난독화 식을 독취하는 독취단계(S20);

E-graph를 이용한 Term-rewrite 기법으로 MBA 난독화 식을 간략화하는 간략화단계(S30);

간략화된 MBA식을 사용자가 원하는 형태 및 수단으로 출력하는 출력단계(S40);를 포함하는 E-graphs를 이용한 MBA식 간략화 방법에 있어서,

상기 간략화단계(S30)는,

상기 독취단계(S20)에서 독취한 Infix 형태의 MBA 식을 Prefix 형태로 연산자를 변환하는 연산자변환단계(S31);

상기 Prefix 형태로 변환된 MBA식을 E-graph로 변환하는 Transform단계(S32);

사용자 정의 규칙에 따라 Term-rewrite 방식으로 E-graph를 간략화하는 Term-rewrite단계(S33);

상기 간략화된 E-graph로부터 MBA 간략화 식을 생성하는 간략화식생성단계(S34);를 포함함을 특징으로 하는 E-graphs를 이용한 MBA식 간략화 방법.

청구항 5

제4항에 있어서,

상기 연산자변환단계(S31)는, 오픈소스 기반의 역공학 프레임워크인 Miasm을 활용하여 Infix 형태의 MBA 난독화 식을 Prefix 형태로 연산자를 변환함을 특징으로 하는 E-graphs를 이용한 MBA식 간략화 방법.

청구항 6

제4항에 있어서,

상기 Transform단계(S32)는, Rust 언어 기반의 프레임워크인 Egg를 사용하여 Prefix 형태의 MBA식을 E-graph로 변환함을 특징으로 하는 E-graphs를 이용한 MBA식 간략화 방법.

발명의 설명

기술 분야

[0001] 본 발명은 MBA 난독화 식의 간략화 시스템 및 방법에 관한 것으로, 더욱 상세하게는 코드 분석가들이 MBA 난독화된 코드를 쉽게 해석할 수 있도록 E-graph를 이용하여 MBA 난독화 식을 간략화하는 시스템 및 방법에 관한 것이다.

배경 기술

[0002] 코드 난독화(Code Obfuscation)는 프로그램을 보호하기 위해 의미 없는 코드를 추가하거나 같은 코드를 복잡하게 작성하여 코드 해독을 어렵게 만드는 것으로써, 소프트웨어의 보안성 향상과 불법복제 및 해킹 방지 수단으로 널리 사용되고 있지만, 최근에는 악의적 사용자들에 의해 바이러스 및 멀웨어와 같은 악성코드에도 난독화 기술이 적용되어 역공학에 의한 코드 분석을 더욱 어렵게 한다.

[0003] 현재 널리 사용되고 있는 난독화 기법에는, 산술연산과 논리연산을 조합하여 복잡하고 불투명한 코드를 생성하는 MBA 난독화(Mixed Boolean Arithmetic Obfuscation), 항상 참 또는 거짓을 반환하는 가짜 조건문을 도입하여 코드를 복잡하게 만드는 Opaque Predicate 난독화, 프로그램이 실행될 때 코드를 변환하는 동적 난독화(Dynamic Obfuscation), 코드를 가상화하여 실행환경을 모방하는 방법으로 코드 해석을 어렵게 만드는 가상화(Virtualization), 문자열을 암호화하거나 변형하여 코드에서 문자열 리터럴을 이해하기 어렵게 만드는 문자열 난독화 방법 등이 있으나, 최근에는 코드의 복잡성을 높여 역공학을 어렵게 만들기 위해 MBA 난독화와 Opaque Predicate 난독화를 결합한 코드 난독화 방식을 채용하고 있어, 역공학에 의한 코드 분석을 더욱 어렵게 하고 있다.

[0004] 이러한 난독화된 코드를 분석하기 위해 SMT-Solver, Z3-Solver 등을 이용하여 의미적으로 같은 기능을 수행하는 부분을 찾아 역난독화 하는 방법을 시도해왔으나, MBA 난독화 기법의 채용으로 인해 역난독화 분석을 더욱 어렵게 하고 있다.

[0005] 또한, 기존의 MBA 역난독화는 주로 프로그램 합성과 머신러닝, SMT Solver를 활용해 왔으나, 해결하고자 하는 MBA 식에 따라 시간이 오래 걸리거나 의미가 동일하게 유지되지 않는 문제점이 있었다.

[0006] 이에 본 발명에서는 이러한 MBA 난독화 식을 rule에 기반한 Term-rewrite 기법으로 좀 더 빠르고 정확하게 역난독화 할 수 있도록 E-graph를 활용한 Term-rewrite 기법의 MBA식 역난독화 방법을 제시하고자 한다,

선행기술문헌

특허문헌

- [0007] (특허문헌 0001) 대한민국 등록특허 제10-2524627호 (2023. 04. 24.)
 (특허문헌 0002) 유럽 특허공개 제4033381호 (2022. 07. 27.)

비특허문헌

- [0008] (비특허문헌 0001) (논문 0001) 프로그램 합성기반 MBA 난독화 해제 도구, 한국정보과학회, 2022.9.
 (비특허문헌 0002) (논문 0002) 프로그램 합성을 활용한 MBA 표현 간소화 도구, 한국소프트웨어종합학술대회 논문, 2021.

발명의 내용**해결하려는 과제**

- [0009] 본 발명은 상기와 같은 종래기술의 제반 문제점들을 해결하기 위해 제안된 것으로, 본 발명의 목적은 악성코드 및 멀웨어 등 악의적인 행동을 하는 프로그램에 적용된 MBA 난독화 식을 코드 분석자들이 쉽게 분석할 수 있도록 MBA 난독화 식의 간략화 방법을 제공하기 위함이다.
- [0010] 또한, 본 발명의 다른 목적은 무분별한 코드 난독화에 따른 부작용을 줄이고 코드 난독화의 선의적 이용을 도모하고자 MBA 기법으로 난독화된 식을 쉽고 간편하게 간략화할 수 있는 E-graph를 이용한 MBA 난독화 식의 간략화 방법을 제공하기 위함이다.

과제의 해결 수단

- [0011] 상기와 같은 목적을 달성하기 위하여,
- [0012] 본 발명은 Term-rewrite에 적용될 연산 규칙들을 정의하는 규칙정의단계(S10); 사용자로부터 MBA 난독화 식을 독취하는 독취단계(S20); E-graph를 이용한 Term-rewrite 기법으로 MBA 난독화 식을 간략화하는 간략화단계(S30); 간략화된 MBA식을 사용자가 원하는 형태 및 수단으로 출력하는 출력단계(S40);를 통해 MBA 난독화 식을 간략화하는 E-graphs를 이용한 MBA식 간략화 방법을 특징으로 하며,
- [0013] 상기 간략화단계(S30)은 상기 독취단계(S20)에서 독취한 Infix 형태의 MBA 식을 Prefix 형태로 연산자를 변환하는 연산자변환단계(S31); 상기 Prefix 형태의 MBA식을 E-graph로 변환하는 E-graph화단계(S32); 사용자 정의 규칙에 따라 Term-rewrite 방식으로 E-graph를 간략화하는 Term-rewrite단계(S33); 상기 간략화된 E-graph로부터 MBA 간략화 식을 생성하는 간략화식생성단계(S34);를 수행함을 특징으로 한다.
- [0014] 또한, 본 발명은 Term-rewrite에 적용될 연산 규칙들을 정의하는 규칙정의부(10); 사용자로부터 MBA 난독화 식을 취득하는 독취부(20); E-graph를 이용한 Term-rewrite 기법으로 MBA식을 간략화하는 간략화부(30); 간략화된 MBA식을 사용자가 원하는 형태 및 수단으로 출력하는 출력부(40); 상기 규칙정의부(10), 독취부(20), 간략화부(30), 출력부(40)를 제어하는 제어부(50);를 구비한 E-graphs를 이용한 MBA식 간략화 시스템을 특징으로 하며,
- [0015] 상기 간략화부(30)은 독취부(20)에서 독취한 Infix 형태의 MBA 난독화 식을 Prefix 형태로 변환하는 연산자변환부(31); 상기 Prefix 형태로 변환된 MBA 난독화 식을 E-graph로 변환하는 Transform부(32); 사용자 정의 규칙에 따라 Term-rewrite 방식으로 E-graph를 간략화하는 Term-rewrite부(33); 상기 간략화된 E-graph로부터 MBA 간략화 식을 생성하는 간략화식생성부(34);를 구비함을 특징으로 한다.

발명의 효과

- [0016] 본 발명의 E-graph를 활용하여 MBA 난독화 식을 간략화할 경우의 예상되는 효과는 다음과 같다,
- [0017] 첫째, E-graph를 활용하면 역난독화 과정이 효율적으로 수행될 수 있어 MBA 난독화 식의 역난독화(간략화)에 걸리는 시간과 노력을 단축할 수 있다.
- [0018] 둘째, E-graph를 통해 같은 의미를 가진 식을 쉽게 식별하여 간략화함으로써 코드 크기를 줄일 수 있어 성능 최

적화 및 실행시간 단축이 가능하다.

[0019] 셋째, 역난독화를 통해 코드가 간략화됨으로써 유지보수 및 가독성이 향상될 수 있다.

[0020] 또한, 본 발명의 MBA 난독화 식을 간략화함에 있어 Term-rewrite 알고리즘을 통해 사용자 정의 규칙을 적용할 경우의 예상되는 효과는 다음과 같다,

[0021] 첫째, MBA 난독화 식을 Term-rewrite 알고리즘을 적용하여 간략화함으로써 사용자가 정의한 간략화 규칙을 쉽게 적용할 수 있다.

[0022] 둘째, MBA 난독화 식의 간략화에 사용자 정의 규칙을 적용함으로써 개별적인 요구사항에 따른 맞춤형 간략화가 가능하다.

[0023] 셋째, MBA 난독화 식을 간략화함에 있어 사용자 정의 규칙을 통해 간략화 대상, 범위, 수준 등을 세밀하게 제어할 수 있다.

도면의 간단한 설명

[0024] 도 1은 본 발명의 실시예에 따른 MBA 난독화 식을 간략화하는 시스템 구성도 이다.

도 2는 본 발명의 실시예에 따른 MBA 난독화 식을 간략화하는 프로세스 구성도이다.

도 3은 Infix 및 Prefix 형태로 표현된 MBA 식의 예시이다.

도 4는 Prefix 형태의 MBA 난독화 식을 E-graph로 표시한 E-graph 예시도 이다.

도 5는 사용자 정의 규칙에 따른 Rewrite rule을 적용하여 간략화한 E-graph 간략화 예시도 이다.

도 6은 사용자 정의 규칙들의 예시도 이다.

도 7은 Transform부(32), Term-rewrite부(33), 간략화식생성부(34)의 세부 기능구성을 나타낸 도면이다.

도 8은 Transform단계(S32), Term-rewrite단계(S33), 간략화식생성단계(S34)의 세부 프로세스를 나타낸 도면이다.

발명을 실시하기 위한 구체적인 내용

[0025] 이하, 첨부한 도면을 참조하여, 본 발명의 실시예를 상세히 설명한다.

[0026] 본 발명의 실시예를 설명함에 있어, 공지된 기능 또는 구성에 대한 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단될 경우, 그 설명을 생략할 수 있으며, 또한, 본 발명에서 달리 정의되지 않는 한 여기서 사용되는 용어들은 당해 기술분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 의미로 해석되어야 한다,

[0027]

[0028] 본 발명은 무분별한 코드 난독화에 따른 부작용을 줄이고 코드 난독화의 선의적 이용을 도모하고자 MBA 기법으로 난독화된 식을 쉽고 간편하게 간략화할 수 있는 E-graph를 이용한 MBA식 간략화 방안을 제안한다.

[0029]

[0030] 도 1은 본 발명의 일 실시예인 E-graphs를 이용한 MBA식 간략화 시스템의 구성을 나타낸 도면이다.

[0031] 도 1에 도시된 바와 같이, 본 발명의 일 실시예인 E-graphs를 이용한 MBA식 간략화 시스템은 Term-rewrite에 적용될 연산 규칙들을 정의하는 규칙정의부(10); 사용자로부터 MBA 난독화 식을 취득하는 독취부(20); E-graph를 이용한 Term-rewrite 기법으로 MBA식을 간략화하는 간략화부(30); 간략화된 MBA식을 사용자가 원하는 형태 및 수단으로 출력하는 출력부(40); 상기 규칙정의부(10), 독취부(20), 간략화부(30) 및 출력부(40)를 제어하는 제어부(50);를 포함한다.

[0032]

[0033] 규칙정의부(10)은 도 6과 같이, Term-rewrite에 적용될 연산 규칙들을 사용자가 미리 정의해 놓은 것으로써, 이곳에는 MBA 수식을 간소화하는데 필요한 사칙연산 및 Boolean 연산에서 널리 사용되는 교환법칙, 결합법칙, 분배법칙 등을 기록한다.

[0034] 독취부(20)은 사용자로부터 Infix 형태로 연산자가 배치된 MBA 난독화 식을 입력받아 ‘MBA난독화식(Infix)’

DB에 저장하는 기능을 수행한다, 이는 Infix 형태로 수식을 작성하는 것이 사용자가 이해하기 쉽고 편리하기 때문이다.

[0035] 간략화부(30)은 연산자변환부(31), Transform부(32), Term-rewrite부(33), 간략화식생성부(34)를 포함한다.

[0036] 출력부(40)은 간략화식생성부(34)에서 생성된 MBA 간략화 식을 사용자가 원하는 출력 형태 및 매체로 출력한다.

[0037] 제어부(50)는 규칙정의부(10), 독취부(20), 간략화부(30), 출력부(40)의 각 기능을 제어하는 역할을 수행한다.

[0038]

[0039] 연산자변환부(31)은 도 3과 같이, 독취부(20)에서 입력받은 Infix 형태로 연산자가 배치된 MBA 난독화 식을 컴퓨터에서의 처리가 쉽도록 Prefix 형태의 MBA 난독화 식으로 변환하는 기능을 수행하며, 그 결과는 ‘MBA난독화 식(Prefix)’ DB에 파일 형태로 저장된다. Infix 형식의 MBA 식을 Prefix 형식으로 변환하는 이유는 Prefix 형식의 MBA 식만을 간략화할 수 있기 때문이다.

[0040] 본 발명에서는 오픈소스 기반의 역공학 프레임워크인 Miasm(바이너리 프로그램을 분석, 수정, 생성)을 활용하여 MBA식에서의 연산자 변환을 수행하게 되며, 그 결과는 아래와 같이, Infix 수식에서는 괄호 안의 중간에 있던 연산자가 Prefix 수식에서는 괄호 앞으로 이동하게 된다.

Infix 형태로 표현된 MBA식	Prefix 형태로 표현된 MBA식
$(a((((d \& e) * (d e)) + ((d \& (e ^ 4294967295)) * ((d ^ 4294967295) \& e))) + ((2 ^ 4294967295) * (1 - 0))) + (1 * (1 - 0)))$	$(+ (+ (+ (* (\& d e) (d e)) (* (\& d (^ e 4294967295)) (\& (^ d 4294967295) e))) (* (^ 2 4294967295) (- 1 0))) (* 1 (- 1 0)))$

[0041]

[0042] 위와 같이, Infix 형식의 MBA 식을 Prefix로 변환해 주는 Miasm 소스코드는 다음과 같다.

```
def infix2prefix(miasmir:Expr) -> str:
    #exprprop (이진 연산자), exprid (변수), exprint(상수) 세 표현식만 사용

    try: #none miasmir type exception
        miasmir.is_op
    except:
        return miasmir

    if miasmir.is_op(): #ExprOp일 경우 - args에 대해 재귀 수행
        if len(miasmir.args) == 1: #unary_op 처리 -- egg 입력 전 - 부호 처리 가능
            if str(miasmir.op) == "-":
                return "( %s %s %s )" % ("*", str(infix2prefix(miasmir.args[0])), "(- 1)")
            return "( %s %s )" % (str(miasmir.op), str(infix2prefix(miasmir.args[0])))
        return "( %s %s %s )" % (str(miasmir.op), str(infix2prefix(miasmir.args[0])), str(infix2prefix(miasmir.args[1])))

    if miasmir.is_id(): #재귀 종료하고 변수 변환
        return miasmir.name

    if miasmir.is_int(): #재귀 종료하고 상수 변환
        return miasmir.arg
```

[0043]

[0044]

[0045] Transform부(32)는 Prefix 형태의 MBA 난독화 식을 E-graph 형태로 변환하는 기능으로써, 본 발명에서는 Rust 언어(메모리 안전성, 성능, 편의성에 중점을 둔 프로그래밍 언어) 기반의 프레임워크인 Egg를 사용하여 Prefix 형태의 MBA 난독화 식을 E-graph 형태로 변환하는 기능을 수행하여 ‘E-graph’ DB에 저장하게 되며, 아래 소스 코드와 같이, let 부분이 E-graph를 만들고 간단화하는 기능을 수행한다.

```
for expr : String in result
{
    let e : RecExpr<SymbolLang> = expr.parse().unwrap();
    let start : RecExpr<SymbolLang> = expr.parse().unwrap();
    let runner : Runner<SymbolLang, ()> = Runner::default().with_explanations_enabled().with_expr( expr & start).run(rules);

    let extractor : Extractor<...> = Extractor::new(&runner.egraph, cost_function: AstSize);
    let (best_cost : usize, best_expr : RecExpr<SymbolLang>) = extractor.find_best( eclass: runner.roots[0]);
    let origin_cost = AstSize.cost_rec( expr & e);

    let best_cost_f : f64 = best_cost as f64;
    let origin_cost_f : f64 = origin_cost as f64;
    let simp_rate : f64 = best_cost_f/origin_cost_f;
```

[0047] 즉, 도 4와 같이, Prefix 형태의 MBA 난독화 식을 E-graph로 간략히 표현할 수 있으며, E-graph는 동등한 의미를 가진 E-class 정보가 추가된 그래프형 자료구조이므로 같은 의미를 가진 여러 식(term)을 한 번에 표현할 수 있고, 또한 표현된 그래프에서 가장 간략화된 식을 빠르게 구할 수 있으며, Term-rewrite rule이 적용되어도 간단하게 표현될 수 있다.

[0048] 또한, 상기 Transform부(32)는, Rust 환경에서 Egg 프레임워크를 설치하고 프로젝트를 추가하는 Egg프레임워크 설치부(321), Prefix 형태의 MBA식을 분석하여 논리연산, 산술연산, 변수, 리터럴등을 식별하는 MBA식분석부(322), E-graph 라이브러리를 활용하여 Egg 프레임워크 내에서 사용할 E-graph의 노드와 링크를 정의하는 E-graph노드링크정의부(323), Egg 프레임워크의 API와 도구를 활용하여 MBA 식의 논리/산술연산을 E-graph의 링크로 매핑하는 논리산술연산매핑부(324), MBA 식에서 사용될 변수와 리터럴값을 E-graph의 노드로 매핑하는 변수리터럴값매핑부(325), 매핑된MBA식의 요소를 기반으로 Egg를 사용하여 E-graph를 생성하는 E-graph생성부(326), E-class를 설정하여 E-graph 내에서 동일한 값을 가지는 노드를 그룹화하는 E-class그룹화부(327), 중복된 노드를 제거하거나 구조를 간략화하여생성된 E-graph를 최적화하는 E-graph최적화부(328)로 그 기능을 좀 더 세분화할 수 있다.

[0049]

[0050] Term-rewrite부(33)는 Egg 프레임워크에 기초하여 사용자정의규칙에 의거 E-graph를 재구성하는 단계로써, E-graph 내에 저장된 수식에서 사용자정의규칙과 매칭되는 term(식)을 찾아 해당 term을 rewrite 하는 기능을 수행하며, 그 결과는 ‘간략화된 E-graph’ DB에 저장한다. 즉, Term-rewrite란 특정한 식(term)을 사용자 정의 규칙에 따라 간단한 형태로 바꾸는 것으로, E-graph 내의 각 Term과 매칭되는 Rewriting rule에 의거 기존의 term을 대체(substitute)하는 연산을 반복하여 간단한 식을 찾게 되며, 그 수행 결과는 아래의 예시와 같다.

```

=====
Origin cost: 33
Origin expr: (+ (+ (+ (& d a) (& d a)) (+ (& d (+ 4294967295)) (& d 4294967295) a))) (+ (+ 2 4294967295) (- 1 0))) (+ 1 (- 1 0)))
Simplified cost: 25
simplified expr: (+ 1 (+ (+ (& d a) (& d a)) (+ (& d (+ 4294967295)) (& d 4294967295) a))) (+ 2 4294967295)))
simplify rate: 0.758
=====
(+ (+ (+ (& d a) (& d a)) (+ (& d (+ 4294967295)) (& d 4294967295) a))) (+ (+ 2 4294967295) (- 1 0))) (+ 1 (- 1 0)))
(Rewrite> commutative-law-add (+ (+ 1 (- 1 0)) (+ (+ (+ (& d a) (& d a)) (+ (& d (+ 4294967295)) (& d 4294967295) a))) (+ (+ 2 4294967295) (- 1 0))))
(+ (+ (+ (+ (& d a) (& d a)) (+ (+ (+ (& d a) (& d a)) (+ (& d (+ 4294967295)) (& d 4294967295) a))) (+ (+ 2 4294967295) (- 1 0))))
(+ (Rewrite> mul-1 (- 1 0)) (+ (+ (+ (& d a) (& d a)) (+ (& d (+ 4294967295)) (& d 4294967295) a))) (+ (+ 2 4294967295) (- 1 0)))
(+ (Rewrite> minus-identity-law 1) (+ (+ (+ (& d a) (& d a)) (+ (& d (+ 4294967295)) (& d 4294967295) a))) (+ (+ 2 4294967295) (- 1 0)))
(+ 1 (+ (+ (+ (& d a) (& d a)) (+ (& d (+ 4294967295)) (& d 4294967295) a))) (+ (+ 2 4294967295) (- 1 0)))
(Rewrite> minus-identity-law 1)))
(+ 1 (+ (+ (+ (& d a) (& d a)) (+ (& d (+ 4294967295)) (& d 4294967295) a))) (+ (+ 2 4294967295) (- 1 0)))
(Rewrite> mul-1 (+ 2 4294967295)))
Best expr: (+ 1 (+ (+ (+ (& d a) (& d a)) (+ (& d (+ 4294967295)) (& d 4294967295) a))) (+ 2 4294967295)))
Elapsed time: 18.263527ms
=====

```

[0051] Process finished with exit code 0

[0052] 즉, 도 5에 제시되어 있는 바와 같이, 사용자 정의 규칙에 따른 Rewrite rule을 적용하여 Term-rewrite 방식을 통해 E-graph를 간략화할 수 있으며, Rewrite rule에 의해 동등한 의미를 가진 부분을 E-class로 쉽게 파악할 수 있도록 E-graph에서 E-class는 점선박스로 표시하고 노드는 실선의 동그라미로 표시함으로써, 사용자정의규칙 “ $(a \leq 1) \Rightarrow (a * 2)$ ”에 따라 “ $(a \leq 1)$ ” 텀이 E-class인 “ $(a * 2)$ ” 텀으로 Term-rewrite 되었음을 확인할 수 있다. 따라서, E-graph를 활용하면 Term-rewrite 과정에서 발생할 수 있는 규칙적용순서에 따른 부정확성 문제와 대체시 발생할 수 있는 정보의 누락이나 잘못된 의미의 추가를 예방할 수 있다.

[0053]

[0054] 또한, 상기 Term-rewrite부(33)는, E-class와그래프로표현된 E-graph를 분석하여 그래프 내의 노드와 링크를 이해하는 E-graph분석부(331), E-graph 내의 수식을 간략화하기위한 규칙 집합을 정의하는 간략화규칙정의부(332), 규칙을 적용할 수 있는 패턴을 찾아 E-graph 내의 각 노드에 대해 간략화 규칙을 적용하는 간략화규칙적용부(333), 규칙 적용으로 클래스 간의 합병?분할이발생됨에 따라 E-graph 내의 E-class가 변경되는 E-class갱신부(334), 더 이상 간략화할 수 없을 때까지 수식을 최적화하는 최적화반복부(335)로 그 기능을 좀 더 세분화할 수 있다.

[0055]

[0056] 간략화식생성부(34)는 사용자 정의 규칙에 따른 Term-rewrite로 간략화된 E-graph로부터 MBA 간략화 식을 생성하는 기능을 수행하여 ‘MBA간략화식’ DB에 파일 형태로 저장되며, 그 소스 코드와 출력 형태는 아래의 예시와 같다.

[0058]

[0059]

[0061]

[0063]

[0064]

[0065]

[0066]

[0067]

[0068]

- [0069] 연산자변환단계(S31)은 도 3과 같이, 독취단계(S20)에서 입력받은 Infix 형태의 MBA 난독화 식을 컴퓨터에서의 처리가 쉽도록 Prefix 형태의 MBA 난독화 식으로 변환하는 기능을 수행하는 단계로써, 본 발명에서는 오픈소스 기반의 역공학 프레임워크인 Miasm(바이너리 프로그램의 분석, 수정, 생성에 활용됨)을 활용하여 MBA식의 연산자 변환을 수행하였다.
- [0070]
- [0071] Transform단계(S32)는 도 4와 같이, Prefix 형태의 MBA 난독화 식을 E-graph 형태로 변환하는 단계로써, 본 발명에서는 Rust 언어(메모리 안전성, 성능, 편의성에 중점을 둔 프로그래밍 언어) 기반의 프레임워크인 Egg를 사용하여 Prefix 형태의 MBA 난독화 식을 E-graph 형태로 변환하였으며, 그 수행과정은 다음과 같다.
- [0072] (S321) Egg프레임워크설치 단계: Rust 환경에서 Egg 프레임워크를 설치하고 프로젝트에 추가한다,
- [0073] (S322) MBA식분석 단계: Prefix 형태의 MBA 난독화 식을 분석하여 논리연산, 산술연산, 변수, 리터럴 등의 구성요소를 식별한다.
- [0074] (S323) E-graph노드링크정의 단계: Egg는 E-graph를 효과적으로 다룰 수 있는 라이브러리를 제공하므로, E-graph 라이브러리를 활용하여 Egg 프레임워크 내에서 사용할 E-graph의 노드와 링크를 정의한다.
- [0075] (S324) 논리산술연산매핑 단계: Egg프레임워크의 API와 도구를 활용하여 MBA 난독화 식의 각 논리·산술연산을 E-graph의 링크로 매핑한다.
- [0076] (S325) 변수리터럴값매핑 단계: MBA 식에서 사용된 변수와 리터럴 값을 E-graph의 노드로 매핑한다.
- [0077] (S326) E-graph생성 단계: 매핑된 MBA 난독화 식의 요소를 기반으로 Egg를 사용하여 E-graph를 생성한다. 이 과정에서 MBA 난독화 식의 구조를 보존하고, E-graph에서 동등한 값에 대한 E-class를 설정한다.
- [0078] (S327) E-class그룹화 단계: E-class는 동일한 값 또는 표현식을 나타내는 데 사용되므로, E-class를 설정하여 E-graph 내에서 동일한 값을 가지는 노드를 그룹화한다.
- [0079] (S328) E-graph최적화 단계: 중복된 노드를 제거하거나 구조를 간략화하여, 생성된 E-graph를 최적화하고 간략화한다.
- [0080]
- [0081] Term-rewrite단계(S33)는 도 5와 같이, Egg 프레임워크에 기초하여 사용자정의규칙에 의거 E-graph를 재구성하는 단계로써, E-graph 내에 저장된 수식에서 사용자정의규칙과 매칭되는 term(식)을 찾아 해당 term을 rewrite하는 기능을 수행한다. 즉, Term-rewrite란 특정한 식(term)을 사용자 정의 규칙에 따라 간단한 형태로 바꾸는 것으로, E-graph 내의 각 Term과 매칭되는 Rewriting rule에 의거 기존의 term을 대체(substitute)하는 연산을 반복하여 간단한 식을 찾는 과정이며, 그 수행과정은 다음과 같다.
- [0082] (S331) E-graph분석 단계: E-class와 그래프로 표현된 E-graph를 분석하여 그래프 내의 노드와 링크를 이해한다.
- [0083] (S332) 간략화규칙정의 단계: E-graph 내의 수식을 간략화하기 위한 규칙 집합을 정의한다. 이러한 사용자 정의 규칙은 수식 내의 특정 패턴을 식별하고, 그 패턴을 새로운 형태로 변환하는 데 사용된다.
- [0084] (S333) 간략화규칙적용 단계: 수식 내에서 규칙을 적용할 수 있는 패턴을 찾아 E-graph 내의 각 노드에 대한 간략화 규칙을 적용한다.
- [0085] (S334) E-class갱신 단계: 규칙 적용으로 클래스 간의 합병과 분할이 발생됨에 따라 E-graph 내의 E-class를 갱신한다.
- [0086] (S335) 최적화반복 단계: 더 이상 간략화 또는 변환이 불가능할 때까지 수식 최적화를 반복한다.
- [0087]
- [0088] 간략화식생성단계(S34)는 사용자 정의 규칙에 따른 Term-rewrite로 간략화된 E-graph로부터 MBA 간략화 식을 생성하는 기능을 수행하며, 그 수행과정은 다음과 같다.
- [0089] (S341) E-graph해석 단계: 사용자 정의 규칙에 따른 Term-rewrite로 간략화된 E-graph를 분석하여 그래프 내의 노드와 링크 및 수식의 구조를 이해한다.

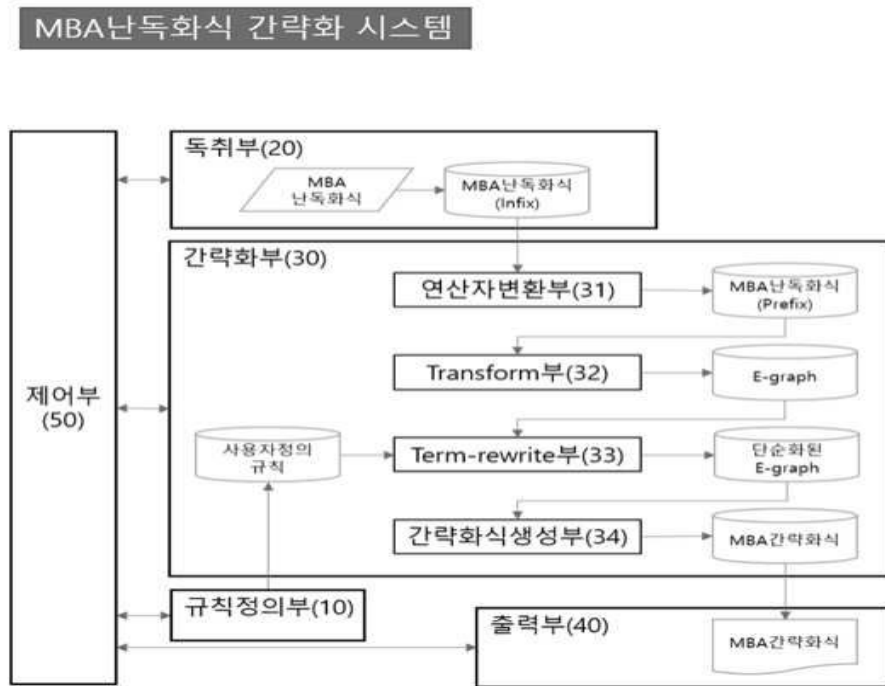
- [0090] (S342) 논리산술연산식별 단계: E-graph 내에서 논리연산과 산술연산을 나타내는 노드와 링크를 식별한다.
- [0091] (S343) 변수리터럴식별 단계: MBA 식에서 사용될 변수와 리터럴 값을 나타내는 E-graph 노드를 식별한다.
- [0092] (S344) 노드링크매핑 단계: E-graph의 각 노드와 링크를 MBA 식의 변수 및 연산자(논리연산, 산술연산) 등을 매핑한다.
- [0093] (S345) MBA식생성 단계: E-class와 매핑된 E-graph의 노드와 링크를 기반으로 MBA 식을 생성한다, 이때 E-graph에서 MBA 식으로의 변환규칙을 적용한다.
- [0094] (S346) E-class고려 단계: E-class를 고려하여 E-graph 내에서 동등한 값을 가지는 노드들을 MBA 식에 반영한다.
- [0095] (S347) MBA식최적화 단계: 중복된 부분을 제거 및 구조 간략화를 통해 생성된 MBA 식을 최적화하고 간략화한다,
- [0096]
- [0097] 본 발명의 실시예를 통해 제시된 구조와 기능들은 단지 본 발명의 기술적 사상에 따른 실시예를 설명하기 위한 목적으로 예시된 것으로써 본 발명의 범주 안에서 다양한 변형되어 실시될 수 있다.
- [0098] 따라서, 본 발명의 권리 범위는 본 명세서에 설명된 실시예들로 한정되는 것이 아니라, 본 발명의 기술적 사상의 범위에 포함되는 모든 균등물 내지 대체물을 포함하는 것으로 이해되어야 할 것이다.

부호의 설명

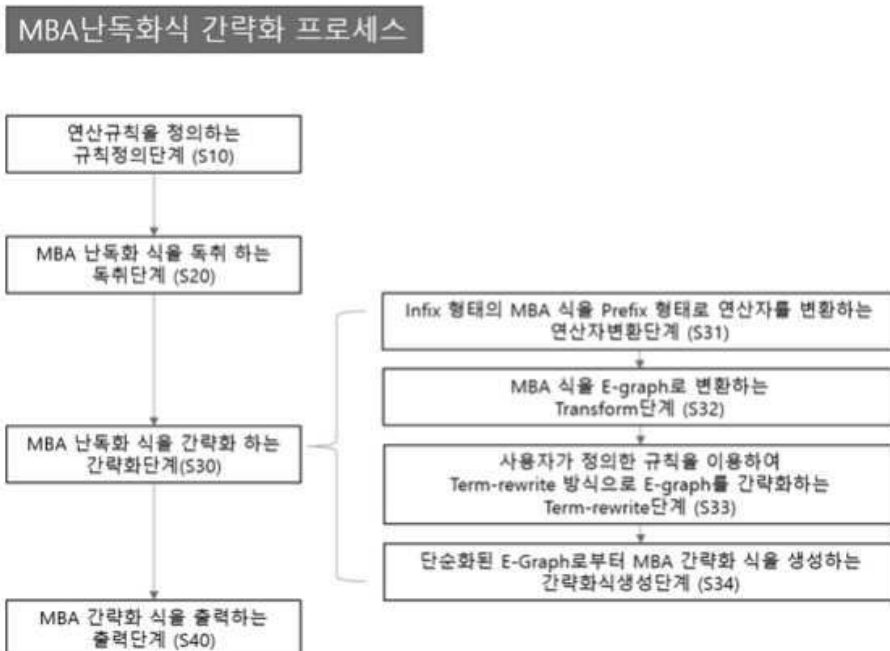
- [0099] 10 규칙정의부 20 독취부
30 간략화부 40 출력부
50 제어부
31 연산자변환부 32 Transform부
33 Term-rewrite부 34 간략화식생성부
S10 규칙정의단계 S20 독취단계
S30 간략화단계 S40 출력단계
S31 연산자변환단계
S32 Transform단계 S33 Term-rewrite단계
S34 간략화식생성단계

도면

도면1

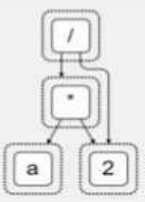


도면2



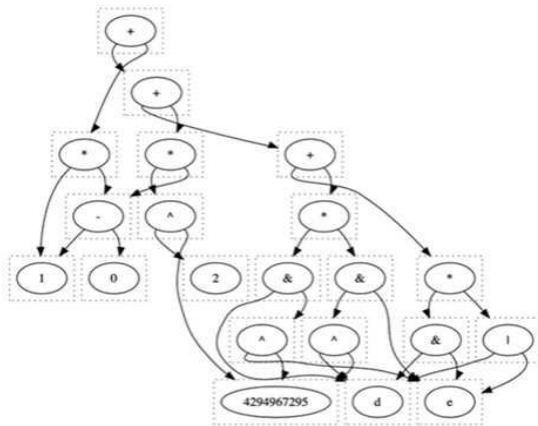
도면3

MBA식의 연산자 변환 및 E-graph화

Infix 형태로 표현된 MBA식	Prefix 형태로 표현된 MBA식	E-graph
$(a * 2) / 2$	$/ (* (a 2)) 2$	

도면4

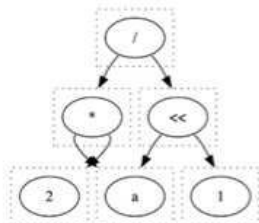
Prefix 형태의 MBA난독화식 E-graph화



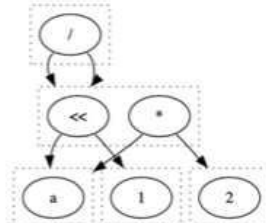
도면5

사용자정의 규칙에 따른 Rewrite rule 적용으로 단순화한 E-graph

$(a * 2) / (a \ll 1)$ 을 표현한 초기의 E-graph



$(a \ll 1) \Rightarrow (a * 2)$ Rewrite rule 적용으로 변형된 E-graph



도면6

사용자정의 규칙들

```
rw!("commutative-law-add"; "(+ ?x ?y)" => "(+ ?y ?x)"),
rw!("commutative-law-minus"; "(- ?x ?y)" => "(- ?y ?x)"),
rw!("commutative-law-mul"; "(* ?x ?y)" => "(* ?y ?x)"),
rw!("associative-law-add"; "(+ (+ ?x ?y) ?z)" => "(+ ?z (+ ?x ?y)))",
rw!("associative-law-mul"; "(* (* ?x ?y) ?z)" => "(* ?z (* ?x ?y)))",
rw!("distributive-law-left"; "(* (+ ?x ?y) ?z)" => "(+ (* ?z ?x) (* ?z ?y)))",
rw!("distributive-law-right"; "(* ?z (+ ?x ?y) )" => "(+ (* ?z ?x) (* ?z ?y)))",
rw!("add-identity-law"; "(+ ?x 0)" => "?x"),
rw!("minus-identity-law"; "(- ?x 0)" => "?x"),
rw!("mul-identity-law"; "(* ?x 1)" => "?x"),
rw!("add-inverse-law"; "(- ?x ?x)" => "0"),
rw!("mul-inverse-law"; "(/ ?x ?x)" => "1"),
```

도면7

Transform부 (32)

- Rust 환경에서 Egg 프레임워크를 설치하고 프로젝트를 추가하는 Egg프레임워크설치부 (321)
- Prefix 형태의 MBA식을 분석하여 논리연산, 산술연산, 변수, 리터럴 등을 식별하는 MBA식분석부 (322)
- E-graph 라이브러리를 활용하여 Egg 프레임워크 내에서 사용할 E-graph의 노드와 링크를 정의하는 E-graph노드링크정의부 (323)
- Egg 프레임워크의 API와 도구를 활용하여 MBA 식의 논리-산술연산을 E-graph의 링크로 매핑하는 논리산술연산매핑부 (324)
- MBA 식에서 사용될 변수와 리터럴 값을 E-graph의 노드로 매핑하는 변수리터럴값매핑부 (325)
- 매핑된 MBA식의 요소를 기반으로 Egg를 사용하여 E-graph를 생성하는 E-graph생성부 (326)
- E-class를 설정하여 E-graph 내에서 동일한 값을 가지는 노드를 그룹화하는 E-class그룹화부 (327)
- 중복된 노드를 제거하거나 구조를 간략화하여 생성된 E-graph를 최적화하는 E-graph최적화부 (328)

Term-rewrite부 (33)

- E-class와 그래프로 표현된 E-graph를 분석하여 그래프 내의 노드와 링크를 이해하는 E-graph분석부 (331)
- E-graph 내의 수식을 간략화 하기 위한 규칙 집합을 정의하는 간략화규칙정의부 (332)
- 규칙을 적용할 수 있는 패턴을 찾아 E-graph 내의 각 노드에 대해 간략화 규칙을 적용하는 간략화규칙적용부 (333)
- 규칙 적용으로 클래스 간의 합병-분할이 발생함에 따라 E-graph 내의 E-class가 변경되는 E-class경신부 (334)
- 더 이상 간략화할 수 없을 때까지 수식을 최적화하는 최적화반복부 (335)

간략화식생성부 (34)

- E-graph를 분석하여 그래프 내의 노드와 링크를 이해하는 E-graph해석부 (341)
- E-graph 내에서 논리연산과 산술연산을 나타내는 노드와 링크를 식별하는 논리산술연산식별부 (342)
- MBA 식에서 사용될 변수와 리터럴 값을 나타내는 E-graph 노드를 식별하는 변수리터럴식별부 (343)
- E-graph의 각 노드와 링크를 MBA 식의 변수 및 연산자 등에 매핑하는 노드링크매핑부 (344)
- E-class와 매핑된 E-graph의 노드와 링크를 기반으로 MBA 식을 생성하는 MBA식생성부 (345)
- E-class를 고려하여 E-graph 내에서 동등한 값을 가지는 노드들을 MBA 식에 반영하는 E-class고려부 (346)
- 중복된 부분의 제거 및 구조 단순화를 통해 생성된 MBA 식을 최적화하는 MBA식최적화부 (347)

도면8

