# PS-Block 구조 기반의 반복횟수 분석기 설계 및 구현

김윤관° 신원 김태완 장천현
건국대학교 컴퓨터 정보통신공학과
{apostlez°, wonjjang, twkim, chchang}@konkuk.ac.kr

# Design and Implementation of Loop Bound Analyzer based on PS-Block

Yun Kwan Kim°, Won Shin, Tae Wan Kim, Chun Hyon Chang
Dept of Computer Science and Engineering, Konkuk University

요 약

실시간 프로그램은 다양한 분야에서 사용되고 있으며, 그 개발자는 논리적, 시간적 정확성을 고려해야 한다. 시간적 정확성은 실시간 프로그램에서 가장 중요한 부분이며, 이를 위한 데드라인은 개발자에 의해 정의된다. 따라서 개발자는 데드라인의 정의를 위하여 기준점을 제시할 수 있는 정적 실행시간 분석이 필요하다. 정적 실행시간 분석에서 프로그램의 반복횟수 분석은 큰 비중을 차지한다. 현재 이러한 반복횟수 분석을 자동화하는 연구가 진행 중이다. 하지만 반복횟수 분석은 반복횟수에 영향을 주는 제어변수의 결정정책에 따라 결과가 달라지고, 자동화를 위한 반복횟수 계산 수식의 적용 범위가 제한되어 있다. 본 논문에서는 이러한 제어변수를 결정 및 탐색하고, 수집된 정보를 개선된 수식을 사용하여 반복횟수 분석을 수행할 수 있도록 PS-Block 구조를 기반으로 반복횟수 분석기를 설계 및 구현하였다. 반복횟수 분석기는 제어변수의 탐색 및 결정, 분석 과정을 자동화하고, 수식의 개선으로 자동화 범위를 확대하며, 개별 반복문 단위의 정밀한 반복횟수 분석을 통해 정확도를 높이고, 신뢰성을 향상시킬 수 있다.

## 1. 서 론

실시간 프로그램은 항공기, 선박, 철도 예매 시스템 등 다양한 분야에서 사용되고 있다. 이러한 실시간 프로그램은 우리의 생활 주변에서 다양한 모습으로 찾아 볼 수 있다. 실시간 프로그램은 프로그램의 본래 기능인 논리적 정확성 외에 정해진 시간 내에 작업을 완료하는 시간적 정확성을 가져야 한다. 시간적 정확성은 실시간 프로그램에서 가장 중요한 부분이며, 작업을 마쳐야 하는 시간인 데드라인은 개발자에 의해 정의된다. 따라서 개발자는 데드라인의 정의를 위하여 기준점을 제시할 수 있는 정적 실행 시간 분석이 필요하다. 정적 실행 시간 분석은 프로그램을 수행하기 전에 수행시간 등을 예측하여 그 정보를 개발자에게 제공할 수 있다.

정적 실행시간 분석에서 반복횟수 분석은 큰 비중을 차지한다. 기존 연구에서 반복횟수 분석은 분석 수행 중에 직접 입력을 받는 방식과 소스코드에 반복횟수 정보를 삽입하는 방식이 사용되었다[4]. 현재 이러한 사용자 입력을 자동화하는 연구가 진행 중이다. 반복횟수 분석 자동화 방법은 제어변수 결정 방법에 따라 분석 결과에 큰 차이가 나타날 수 있고, 분석의 자동화를 위한 반복횟수 분석 수식의 적용 범위가 제한되어 있다.

본 논문에서는 이러한 제어변수를 결정 및 탐색하고 수집된 정보를 개선된 수식을 사용해 반복횟수 분석을 수행할 수 있도록 PS-Block 구조를 기반으로 반복횟수 분석기를 설계 및 구현하였다. 반복횟수 분석기는 제어변수 탐색 및 결정, 분석 과정을 자동화 하고, 수식의 개선으로 자동화 범위를 확대하며, 각각의 반복문 단위의 정밀한 반복횟수 분석을 통해 정확도를 높이고, 신뢰성을 향상시킬 수 있다.

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음

## 2. 관련연구

### 2.1 정적 실행시간 분석

정적 실행시간 분석은 원시 소스코드를 분석하여 그 실행시간을 분석한다. 원시 소스 코드는 주기성을 가지고 독립되어 수행될 수 있는 프로그램 세그먼트로 분리되고 이를 분석하여 실행시간의 정보를 가지는 프로그램 세그먼트 시간 모델과 최악 실행 시간 그래프를 도출한다[2]. 다음 그림 1은 정적 실행시간 분석 과정을 나타낸다.

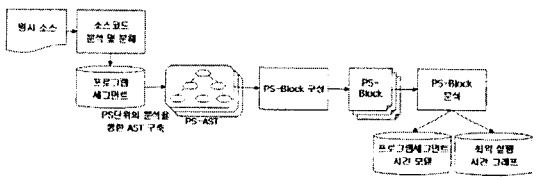


그림 1 정적 실행시간 분석 과정

원시 소스는 소스코드 분석 및 분해 과정을 통해 프로그램 세그먼트로 분리되고 어휘분석 및 구문분석을 통해 실행시간 분석을 위한 최소 단위인 PS-AST(Program Segment Abstract Syntax Tree)를 거쳐 PS-Block (Program Segment Block)으로 변형된다. PS-AST란 프로그램 세그먼트를 트리 구조로 구성하고 분석에 불필요한 노드들을 모두 제거하여 재구성한 트리를 말한다. 이러한 PS-AST의 실행시간을 분석하기 위하여 블록 구성 정책을 통하여 블록 단위로 묶게 되는데 이것을 PS-Block이라고 한다[1]. 이러한 PS-Block 단위로 실행시간 및 반복횟수를 분석하여 프로그램 세그먼트 시간 모델 및 최악 실행 시간 그래프를 생성한다.

### 2.2 반복횟수 분석 방법

반복횟수 분석은 정적 실행시간 분석에서 큰 비중을 차지한다. 반복문의 반복횟수에 따라 프로그램의 예상 실행시간은 큰 차이를 발생시킬 수 있기 때문에 이를 분석하기 위한 방법이 연구되었다.

이 방법은 반복문을 단순 반복문, 다중 출구를 가진 반복문, 중첩된 반복문으로 구분하고, 컴파일을 거친 기계어를 제어변수를 중심으로 분해하여 각각의 분기별로 분석을 수행한다. 제어변수란 반복문 내에서 반복문의 상태를 변화시켜 반복횟수에 영향을 미치는 변수를 말한다. 분해된 분기들은 분기들 사이의 관계를 나타낸 DAG(Direct Acycle Graph)를 통해 분석된다[3].

각 블록의 상태는 크게 Known과 Unknown으로 분류되며, 이는 프로그램 코드에서 분석가능 여부에 의해 분류된다. Unknown의 경우에는 정적 실행시간 분석이 불가능하기 때문에 컴파일 또는 분석 시에 개발자에게 입력을 받는 방식을 사용한다. Known의 경우 반복횟수는 제어변수의 초기 및 한계, 변화에 관한 값으로 수식을 통해 계산할 수 있다. 수식 1은 제어변수 증가분이 X=X+1과 같이 상수로 정해진 경우에만 반복횟수의 계산이 가능하고 X=X*2+1과 같이 증가분이 변하는 경우 반복횟수가 정확히 계산되지 않는 문제점이 있다.

$$N = \left\{ \frac{\text{limit} - (\text{initial} + \text{before}) + \text{adjust}}{\text{before} + \text{after}} \right\}$$

| | |
|---|---|
| limit | 반복문을 끝낼 수 있는 조건문의 비교값 |
| initial | 제어변수가 반복문에 들어오기 전의 초기값 |
| before | 블럭에 들어오기 전의 제어변수 값 |
| adjust | limit값과 비교기호에 + 포함여부 |
| after | 블럭을 나갔을 때 제어변수 값 |

수식 1 반복횟수 계산

### 3. 반복횟수 분석기의 설계 및 구현

반복횟수 분석은 PS-Block 구성, 제어변수 탐색, 반복횟수 분석 과정으로 이루어진다. 구현된 반복횟수 분석기는 다음과 같은 구조를 가진다. PS-Block 구성 모듈에서는 반복횟수 분석을 위한 준비과정으로 소스코드를 분석해 PS-Block구조로 재구성한다. 제어변수 탐색 모듈은 반복횟수 분석을 위한 제어변수 결정 및 정보수집 과정이고, 반복횟수 분석 모듈에서는 수집된 정보를 바탕으로 반복문의 반복횟수를 계산한다.

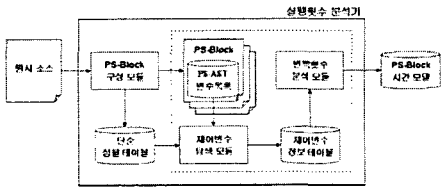


그림 2 반복횟수 분석기 전체 구조

### 3.1 반복횟수 분석기 설계

제어변수 탐색 과정은 반복문 블록에서 반복횟수에 영향을 주는 제어변수를 결정하고, 그 정보를 수집하여 반복횟수 분석 과정으로 보낸다. 제어변수란 반복문 또는 선택문에서 반복횟수에 영향을 주는 변수이다. 제어변수는 반복문에 진입할 때 초기값을 가지고, 반복문 수행여부를 결정하는 비교값을 가진다. 또한 반복문 내부에서 변화하는 값인 증가값을 가진다. 이러한 값들을 구하기 위해 PS-AST변수목록을 사용하여 설정 또는 비교, 연산 정보를 얻는다. PS-AST변수목록은 어떠한 값이 변하거나 사용되는 순서에 따라 PS-AST를 연결한 구조이다.

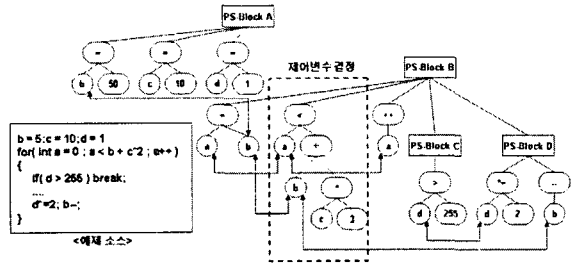


그림 3 PS-AST변수목록을 사용한 제어변수 탐색

반복횟수 분석 과정은 제어변수 탐색 과정을 통해 구성된 제어변수정보테이블에 수집된 정보를 이용하여 반복문의 반복횟수를 계산한다. 제어변수정보테이블은 2개 이상의 제어변수로 구성된 수식을 하나의 연산자를 중심으로 두 항의 연산 관계를 저장하고, 각각의 항도 제어변수정보테이블에 저장된다. 한 항은 항을 구성하는 제어변수나 연산자가 저장된 테이블상의 위치를 가리키고, 이를 참조하여 연산의 결과나 제어변수의 값을 얻을 수 있다. 반복횟수의 계산은 제어변수정보테이블에 저장된 반복문의 조건식을 중심으로 수식을 통해 계산한다.

(1) 반복횟수 계산을 위한 제어변수 결정

제어변수정보테이블은 연산자를 중심으로 두 항의 관계를 저장하여 제어변수로 이루어진 수식을 처리한다. 하지만 조건식의 양쪽 항이 변수로 이루어진 경우 반복횟수 계산 기준이 명확하지 않아 계산이 어렵다. 이 문제를 해결하기 위해 제어변수를 결정할 때 그 비교 대상을 명확히 정해야 한다. 이는 수식의 변화를 통해 해결할 수 있다. 두 변수로 이루어진 수식 (a<b) 는 (a-b<0)으로 변환하여 변수와 상수간의 연산으로 바꿀 수 있다. 따라서 제어변수는 (a-b)으로 결정되고, 비교값은 0이 된다. 이에 따라 초기값과 증가값도 결정된 제어변수와 같은 연산을 거쳐 결정된다.

(2) 반복횟수 분석 수식의 개선

그림 3 예제소스의 d와 같이 증가값이 배수인 경우 제어변수 d의 값은 {1,2,4,8, ...}로 등비 증가하게 되고 증가값은 {1,2,4,...}로 반복횟수에 따라 변한다. 이는 기존 수식이 가지는 제약 때문에 제어변수 d에 의한 반복횟수의 정확한 계산이 어렵다. 이를 해결하기 위해 제어변수의 증가분이 변하는 경우에 대한 수식으로 수식 2를 추가하였다. 수식 2를 적용하기 위해 제어변수 n의 증가값을 n = an + b의 형태로 수정한다. 이때에 a가 1이 아닌 경우 반복횟수를 계산하고 a가 1인 경우 증가값은 일정하게 되어 기존의 수식으로 반복횟수를 계산한다.

초기값 : X0
비교값 : Xn
증가값 : aX+b( a != 0, 1 )

$$\text{반복횟수} = \left\lfloor \frac{\log\left\lfloor \frac{(a-1)(Xn-X0)}{(aX0+b-X1)} + 1 \right\rfloor}{\log(a)} \right\rfloor + 1$$

수식 2 추가한 반복횟수 계산 수식

수식의 추가로 기존 수식으로 계산이 어려웠던 반복문들의 분석 자동화가 가능하다. 표 1은 개선 과정을 거쳐 생성된 제어변수 정보테이블의 구조이다.

표 1 제어변수정보테이블 구조

| ID | 위치 | 이름 | 연산 | 비교값 | 초기값 | 증가값 | | 구분 | 실행횟수 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | a | b | | |
| T1 | B | T2 | < | 0 | -25 | 1 | 2 | Known | 12 |
| T2 | Slave | T3 | - | T4 | -25 | 1 | 2 | Known | |
| T3 | Slave | a | = | | 0 | 1 | 1 | Known | |
| T4 | Slave | T5 | + | T6 | 25 | 1 | -1 | Known | |
| T5 | Slave | b | = | | 5 | 1 | -1 | Known | |
| T6 | Slave | T7 | * | 2 | 20 | | | Known | |
| T7 | Slave | c | = | | 10 | | | Known | |
| T8 | C | d | > | 255 | 1 | 2 | 0 | Known | 8 |

제어변수는 소스코드 상에서 알아낼 수 있는 Known과 사용자 또는 통신을 통한 입력이나 함수의 반환값과 소스코드에서 알아낼 수 없는 Unknown으로 구분되고, 증가값은 수식의 형태에 따라 1차항인 a와 상수항인 b로 구분하였다.

### 3.2 반복횟수 분석기 구현 및 실험

PS-AST변수리스트는 제어변수를 결정하고 값의 변화에 대한 정보의 수집을 위한 재탐색 과정을 줄여준다. 제어변수정보테이블은 제어변수에 영향을 주는 다른 제어변수와의 연산 관계를 저장하고 수집된 값들을 저장하여 제어변수들 간의 연산을 정적으로 분석할 수 있도록 한다. 이를 통해 반복횟수 분석기를 구현하고, 실험을 수행한 소스는 다음과 같다.

표 2 실험을 위한 벤치마크 소스

| 벤치마크 | 설명 | 특징 |
|---|---|---|
| bsort | 버블 정렬 | 기본적인 반복문 구조 |
| insertsort | 삽입 정렬 | 배열에 의존하는 중첩 반복문 사용 |
| qsort-exam | 퀵 정렬 | 비 재귀적으로 정렬을 수행 |
| compress | 데이터 압축 | 난수 데이터의 압축, 제어변수 등비 증가 |

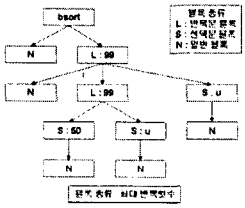


그림 4 버블 정렬의 블록 구조 및 상세 반복횟수

버블 정렬의 상세 반복횟수 정보는 그림 4와 같다. 버블 정렬은 2개의 반복문의 중첩과 선택문으로 이루어져 있다. 각 반복문의 최대 반복횟수는 99이지만, 내부 반복문은 최대 반복횟수가 50인 선택문을 포함하고 있어 최대 반복횟수는 50이 되고 전체 반복횟수는 99*50으로 4950번이 된다. 실험 결과는 다음 표 3과 같다.

표 3 반복횟수 분석 실험 결과

| 사용 예제 | 중첩 반복문 수 | 분석된 반복횟수 | 실제 반복횟수 | 오차율 |
|---|---|---|---|---|
| bsort | 2개 | 4950 | 4950 | 0% |
| insertsort | 2개 | 81 | 81 | 0% |
| qsort-exam | 6개 | 0 | 38 | 알 수 없음 |
| compress | 4개 | 107 | 107 | 0% |

반복횟수 분석 실험 결과를 살펴보면 퀵 정렬은 분석 결과가 알 수 없음으로 되었다. 이는 반복문의 제어변수 또는 제어변수에 영향을 주는 다른 제어변수의 값을 알 수 없는 경우에 발생한다. 버블 정렬과 삽입정렬은 제어변수의 초기, 비교, 증가값을 소스코드 상에서 알 수 있어 이를 통해 반복횟수를 계산한다. 하지만 퀵 정렬은 입력받은 데이터가 반복문의 제어변수로서 Unknown이기 때문에 반복횟수의 계산이 불가능하다.

## 4. 결론

실시간 프로그램의 시간적 정확성을 보장하기 위한 정적 실행시간 분석에서 반복횟수 분석은 큰 비중을 차지한다. 기존 연구에서 반복횟수 분석은 사용자 입력으로 처리하였고 현재 반복횟수 분석을 자동화하는 연구가 진행 중이다. 기존 연구에서 반복횟수 분석 자동화 방법은 제어변수 결정 방법에 따라 분석 결과가 달라질 수 있고 적용 범위가 제한된 반복횟수 분석 수식을 사용한다.

본 논문에서는 PS-Block 구조를 기반으로 제어변수를 결정하고 정보를 수집하기 위한 PS-AST변수리스트와 수집된 정보를 종합하고 각 반복문 단위의 반복횟수를 분석하기 위한 제어변수정보테이블을 통하여 개선된 수식을 사용하는 반복횟수 분석기를 설계 및 구현하였다. 반복횟수 분석기는 제어변수 탐색 및 결정, 분석 과정을 자동화하고, 반복문 블록 단위의 정밀한 반복횟수 분석을 통해 정확도를 높이고, 신뢰성을 향상시킬 수 있다.

향후 계획으로 제어변수에 영향을 주는 선택문에 의한 분기의 분석 방안에 대한 연구를 수행하고 반복횟수 분석기를 결합한 정적실행시간분석기를 구현할 예정이다.

## 5. 참고문헌

[1] 김윤관, 신원, 김태완, 장천현, "TMO기반 정적 분석 도구를 위한 PS-Block 구조의 설계," 정보처리학회 추계 학술발표논문집 제12권 제2호 pp.263-266, 2005

[2] 신원, 김태완, 장천현, "정적 실행시간 분석기의 기반 구조", 한국 소프트웨어공학 학술대회 논문집 제8권 1호 pp.115-123, 2006

[3] C. Healy, M. Sjödin, V. Rustagi, D. Whalley, "Bounding Loop Iterations for Timing Analysis" In Proc. 4th IEEE RTAS pp.12-21, 1998

[4] Christopher A. Healy and David B. Whally, "Automatic Detection and Exploitation of Branch Constraints for Timing Analysis", IEEE Transactions on Software Engineering", Vol 28 no 8 pp.763-781, 2002