PS-Block 구조 기반의 반복횟수 분석 구조 설계

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

Design of Structure for Loop Bound Analysis based on PS-Block

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

요 약

실시간 프로그램은 항공기, 선박, 철도 예매 시스템 등 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 반복횟수의 분석은 큰 비중을 차지한다. 기존 연구에서 반복횟수의 분석은 사용자 입력에 의존하였고 현재 반복횟수 분석을 자동화하는 연구가 진행 중이다. 하지만 반복횟수의 분석은 반복횟수에 영향을 주는 제어변수의 결정정책에 따라 결과가 달라진다. 따라서 본 논문에서는 PS-Block구조를 기반으로 반복횟수에 영향을 주는 제어변수들을 종합적으로 분석하여 보다 정밀하고 사용자의 입력을 자동화하는 반복횟수의 분석이 가능한 방법을 제시한다. 이로써 정적 실행시간 분석은 반복횟수의 정밀한 분석을 통하여 분석 결과의 정확도를 높이고 신뢰성을 향상시킬 수 있다.

1. 서론

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 실행시간 분석이 필요하다[6]. 정적 실행 시간 분석은 프로그램을 수행하기 전에 수행시간 등을 예측하여 그 정보를 개발자에게 제공할 수 있다.

정적 실행 시간 분석에서 프로그램의 반복횟수의 분석은 큰 비중을 차지한다. 기존 연구에서 반복횟수의 분석은 사용자 입력에 의존하였고 현재 반복횟수 분석을 자동화하는 연구가 진행 중이다[3][4]. 이러한 반복횟수의 분석은 반복횟수에 영향을 주는 제어변수의 결정 정책에 따라 분석 결과가 달라질 수 있다. 따라서 반복횟수에 영향을 줄 수 있는 모든 제어변수에 대해 개별적인 분석을 수행하고, 이들을 종합적으로 분석하여 정확한 반복횟수를 분석할 필요가 있다. 이에 본 논문에서는 정적 실행시간 분석을 위한 기반 구조인 PS-Block구조를 기반으로 반복횟수에 영향을 주는 제어변수들을 결정하고 그 정보를 분석하여 보다 정밀하고 사용자의 입력을 자동화하는 반복횟수의 분석이 가능한 방법을 제시한다. 이로서 정적 실행시간 분석은 반복횟수의 정밀한 분석을 통하여 분석결과의 정확도를 높이고 신뢰성을 향상시킬 수 있다.

본 논문은 2장에서 관련연구인 정적 실행시간 분석 과정에 대해 소개하고 반복횟수를 구하기 위한 수식을 설명하며, 3장에서 PS-Block 기반의 반복문

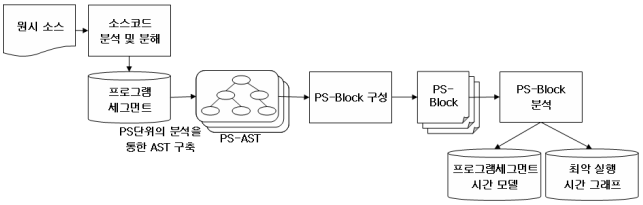
본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음

분석에 대한 내용을 설명한다. 마지막으로 4장에서는 결론 및 향후 연구 방향을 제시한다.

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 실행시간 그래프를 도출한다[2]. 다음 (그림 1)은 정적 실행시간 분석의 과정을 나타낸다.



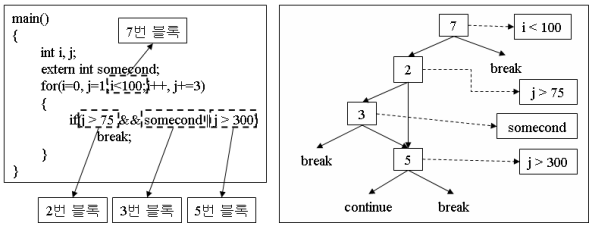
(그림 1) 정적 실행시간 분석 과정

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

반복횟수의 분석은 정적 실행시간 분석에서 큰 비중을 차지한다. 반복문의 반복횟수에 따라 프로그램의 예상 실행시간은 큰 차이를 발생시킬 수 있기 때문에 이를 분석하기 위한 방법이 연구되었다[3][4].

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 변화시키는 변수를 말한다. 분해된 분기들은 (그림 2)와 같이 분기들 사이의 관계를 나타낸 DAG(Direct Acycle Graph)를 통해 분석된다.



(그림 2) 소스코드 및 DAG 예제

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. 제어변수에 의한 반복횟수를 구하는 수식은 다음과 같다[3].

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

<표 1> 수식 용어 설명>

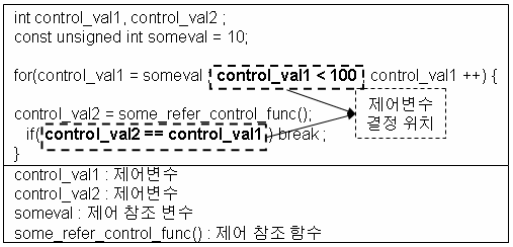
variable	limit의 비교대상 조건에 만족하지 않을 경우 반복문을 종료
limit	반복문을 끝낼 수 있는 조건문에 있는 상수값
initial	variable이 반복문에 들어오기 전에 초기값
before	블럭에 들어오기 전의 variable의 값
adjust	limit와의 비교기호에 '=' 포함여부
after	블럭을 나갈때의 variable의 값

3. 반복횟수 분석 구조 설계

정적 실행 시간 분석에서 반복횟수의 분석은 제어변수의 탐색과 제어변수를 통한 반복횟수의 계산으로 이루어진다. 제어변수의 탐색은 사용자 입력을 줄이기 위해 제어변수를 결정하고 그 정보를 수집하는 과정이고, 반복횟수의 계산은 수집된 정보를 취합하고 분석하여 제어변수에 의존하는 블록의 반복횟수를 계산하는 과정이다.

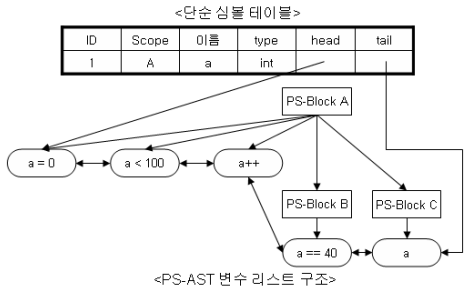
3.1 제어변수의 탐색

여러 제어변수에 의존하는 반복문의 반복횟수를 분석하기 위해 제어변수의 개념을 확장할 필요가 있다. 반복문과 마찬가지로 반복문을 빠져 나갈 수 있는 선택문에도 분기를 결정짓는 변수들이 존재한다. 이러한 변수들을 제어변수에 포함한다. 이로서 제어변수란 반복문 또는 선택문에서 반복횟수에 영향을 주는 변수를 가리킨다. 제어변수의 값은 다른 변수 등에 의해 그 값이 변화될 수 있고, 이러한 변화를 주는 다른 변수 또는 함수의 반환값을 제어참조변수라 지칭한다. (그림 3)은 제어변수의 결정 예를 나타낸다.



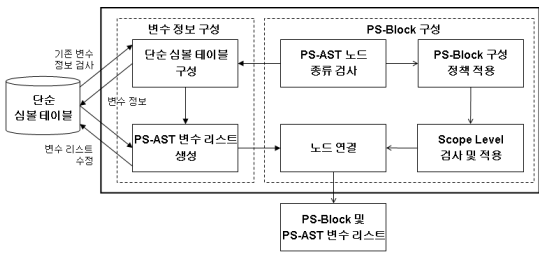
(그림 3) 제어변수 결정 예제

무한 반복문과 같은 경우 반복문의 제어변수는 고정된 상수가 사용될 수 있다. 따라서 제어변수는 제어참조변수를 포함하고, 반복횟수에 영향을 주는 변수, 상수, 함수의 반환값이 된다. 이러한 제어변수의 탐색 및 분석은 부하 문제로 인해 모든 PS-AST를 탐색하며 수행할 수 없다. 따라서 적은 탐색으로 빠르게 제어변수를 탐색하고, 그 정보를 수집할 수 있는 구조가 필요하다. 이를 위하여 PS-AST 변수 목록을 구성하였다. (그림 4)는 단순 심볼 테이블과 PS-AST 변수 목록의 구조 및 연관 관계이다.



(그림 4) PS-AST 변수 목록 구조

PS-AST 변수 목록은 어떠한 변수의 값이 변하거나 사용되는 순서에 따라 해당 PS-AST를 연결한 목록 구조로서 PS-AST로부터 PS-Block 구조를 구성할 때 생성된다. (그림 5)는 PS-Block 구성 과정에서 PS-AST 변수 목록이 생성되는 과정을 나타낸다.

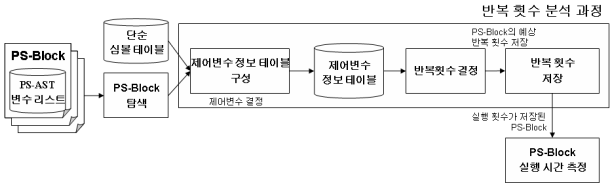


(그림 5) PS-Block 구성 과정

PS-AST 노드 탐색에서 변수를 만나면 단순 심볼 테이블을 검사하여 없는 경우 새로 추가한다. 단순 심볼 테이블은 각 변수의 형, 이름, 위치의 정보를 가지고 있고 변수 목록의 시작 위치와 마지막 위치를 가지고 있어 탐색이 용이한 구조이다. 중복 여부를 확인하고 존재하는 변수는 PS-AST 변수 목록의 마지막 위치에 추가한다.

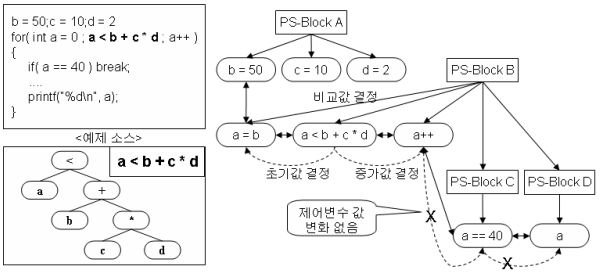
3.2 반복횟수 계산

반복횟수를 분석하기 위해서 해당 반복문과 출구를 가진 선택문의 제어변수를 결정하고 그 값의 변화 범위를 분석하여 반복횟수를 계산하는 일련의 과정이 필요하다. (그림 6)은 이러한 일련의 과정을 나타낸다.



(그림 6)반복횟수 분석 과정

제어변수를 통해 반복횟수를 계산하기 위해서 제어변수의 초기값과 비교값, 증가값을 알아야 한다 [2]. 반복문이나 조건문에서 제어변수는 제어변수의 탐색을 위한 구조인 PS-AST 변수 목록을 이용하여 값의 변화를 추적할 수 있다. 추적을 통해 얻은 정보는 제어변수 정보 테이블에 저장되고, 이를 참조하여 반복횟수를 결정한다. (그림 7)은 제어변수를 결정하고 그 값의 변화를 추적하여 정보를 얻는 과정을 나타낸다.



(그림 7) 제어변수의 추적

먼저, 제어변수가 결정된 위치에서 비교값을 얻고, 반복문에 진입하기 전 마지막 값의 설정부터 반복문 진입 직전까지 값의 추적으로 초기값을 얻는다. 마지막으로 반복문 내부에서 값의 변화를 추적해 증가값을 얻는다. 이러한 정보들을 종합하여 제어변수 정보 테이블을 생성하고, 이를 참조하여 PS-Block 단위로 반복횟수를 결정한다.

제어변수의 비교값이나 초기값, 또는 증가값은 제어참조변수에 의해 변하는 경우 참조를 'Y'로 설정하고, 테이블 상에 추가된 제어참조변수의 ID를 저장한다. 따라서 제어변수는 저장된 ID에 의한 제어참조변수의 참조를 통해 다른 변수의 영향을 적용할 수 있다. 제어변수 정보 테이블은 하나의 변수를 기준으로 분석을 수행할 수 있도록 한다. 하지만 제어

<표 2> 제어변수 정보 테이블

ID	영역	이름	참조	비교값	연산	참조	초기값	참조	증가값	구분	반복횟수	위치	종류
1	B	a	O	2	<	X	0	X	1	known	70	B	loop
2			O	3	+	O	4						
3	A	b			=	X	50					B	loop
4			O	5	*	O	6						
5	A	c			=	X	10					B	loop
6	A	d			=	X	2					B	loop
7	B	a	X	40	==	X	0	X	1	known	40	B	select

※ 참조 항목이 설정되면 다음 값은 테이블의 ID를 가리킨다.

변수는 2개 이상의 제어참조변수 사이의 연산으로 이루어 질 수 있다. 제어변수 테이블에서 수식은 연산자를 중심으로 2개의 항의 관계를 저장하고 이를 통해 수식에 존재하는 우선순위를 처리하며, 수식의 결과를 계산할 수 있다. 테이블의 2번 줄을 t(2)라고 표현할 때, t(2)=t(3)+t(4)로 표현할 수 있다.

(그림 7)의 예제에서 반복문 블록 B의 제어변수는 a이고, 제어참조변수 b, c, d가 존재한다. 제어변수 a는 1씩 증가하고, a의 범위는 0<a<b+c*d이다. b+c*d는 t(2)=t(3)+t(4)와 t(4)=t(5)*t(6)의 과정을 통해 70임을 알 수 있다. 따라서 a의 범위는 0<a<70이 된다. 선택문 블록 C의 제어변수 a도 블록 B의 반복횟수에 영향을 주고 40에서 실행된다. 따라서 반복문의 최대 반복횟수는 and연산에 의한 작은 값 선택으로 40이 된다. 이러한 제어변수의 상태는 known과 unknown으로 구분되며 프로그램 코드 상에서 분석 가능 여부를 나타낸다. unknown은 분석이 불가능한 경우를 나타낸다.

3.3 반복횟수 분석 구조의 특징

본 논문에서 제안하는 반복횟수 분석 방법은 PS-Block 구조를 기반으로 PS-AST 변수 목록과 제어변수 정보 테이블을 사용하여 반복횟수를 분석함으로서 실행시간의 분석을 더욱 정밀하게 하며, 다음과 같은 특징을 가진다.

첫째, 제어변수의 추적 및 분석을 위한 구조인 PS-AST 변수 목록을 사용한다. PS-AST 변수 목록의 사용은 제어변수의 결정 및 정보의 분석을 빠르게 수행할 수 있도록 한다.

둘째, 분석된 제어변수의 정보를 저장하고, 반복횟수를 계산하기 위한 구조인 제어변수 정보 테이블을 사용하였다. 이는 제어참조변수의 정보를 참조할 수 있도록 하여 값의 추적과 여러 제어변수간의 연산처리가 쉽도록 한다. 이는 테이블에 연산 관계를 표시하는 행을 추가함으로서 가능하다.

셋째, PS-Block 구조를 기반으로 분석을 수행함으로서 소스코드 상에서 자동화된 반복횟수 분석을 수행할 수 있다. 이를 통해 개발자는 더 빠르고 정확한 반복횟수 분석 결과를 얻을 수 있다.

4. 결론

실시간 프로그램은 시간적 정확성을 가져야 하기 때문에 개발자는 이를 고려하여 데드라인을 정의하고 그 정확성 검사를 수행해야 한다. 이를 위한 기준점을 제시하는 정적 실행 시간 분석에서 프로그램의 반복횟수의 분석은 큰 비중을 차지한다.

본 논문에서는 PS-Block구조 기반의 반복문 분석 구조를 제시한다. 제안된 반복문 분석 구조는 PS-AST 변수 목록과 제어변수 정보 테이블로 구성되며 반복문 블록의 분석에 사용된다. 블록 단위 제어변수의 탐색 및 분석으로 정확한 반복횟수의 분석을 수행함으로써 사용자의 입력을 줄이고 더욱 정밀한 실행시간의 예측이 가능하다. 정적 실행시간 분석은 반복횟수의 정확한 분석을 통하여 분석 결과에 대한 신뢰성을 향상시킬 수 있다.

향후 연구 계획은 제어변수가 프로그램의 경로에 영향을 주는 경우의 경로 및 반복횟수를 분석하고, 제시한 반복횟수 분석 구조를 적용한 반복횟수 분석 모듈의 구현을 통해 정적 분석 도구를 구현할 예정이다.

참고문헌

[1] 김윤관, 신원, 김태완, 장천현, "TMO기반 정적 분석 도구를 위한 PS-Block 구조의 설계," 정보처리학회 추계학술발표논문집 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] C. Healy, M. Sjödin, "Supporting Timing Analysis by Automatic Bounding of Loop Interations", Journal of Real-Time Systems pp.121-148, 2000

[5] Jakob Engblom and Andreas Ermedahl, "Worst-Case Execution-Time Analysis for Embedded Real-Time Systems", International Journal on STTT, Vol 4 Issue 4 pp437-455, 2003

[6] K.H(Kane) Kim, Lynn Choi, "Issues in Realization of an Execution Time Analyzer for Distributed Real-Time Objects" Proc. 3rd IEEE Symposium on ASSET 2000 pp.151-160, 2000