소프트웨어의 행위를 이해하기 위한 가상 실행에 관한 연구

정양재[U] 이문근
전 북 대 학 교 전 산 학 과
{yjjung, mklee}@cs.chonbuk.ac.kr

A study on Virtual Execution To Understand the Behavior of Software

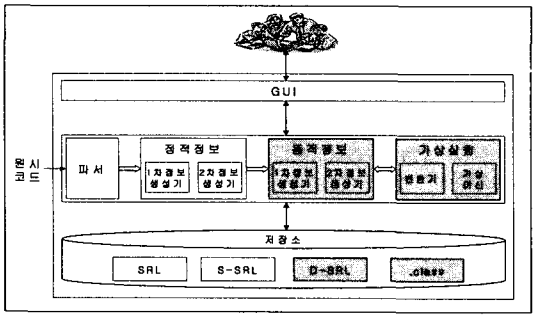
Yang-Jae Jeong[U] Mun-Keun Lee
Dept. of Computer Science, Chonbuk University

요 약

다양한 실행 경로가 존재하는 실시간 시스템을 이해하기 위해 시스템의 정적 정보와 함께 동적 정보가 사용자에게 적절히 제공되어야 한다. 본 논문은 정적 정보와 동적 정보를 표현하기 위해 SRL(*System Representation Language*)을 사용한다. 정적 정보는 SRL 노드를 분석해서 얻고 동적 정보는 SRL을 실행함으로 얻는다. SRL의 가상 실행은 시스템 독립적인 자바 가상 기계를 통해 이루어진다. 가상 실행은 순방향뿐만 아니라 역방향으로도 이루어진다. SRL 실행 라이브러리는 순·역방향 실행을 위해 SRL 각 구문의 의미 규칙에 맞게 정의하며 자바 가상 기계를 통해 실행되는 클래스 파일로 컴파일 된다. 메모리에 로딩된 SRL은 SRL 실행 라이브러리를 동적으로 호출하여 가상 실행을 이룬다. 동적 실행을 통해 추출된 동적 정보는 SRL에 포함된다.

1. 서론

다양한 실행 경로가 존재하는 실시간 시스템을 이해하기 위해서는 시스템의 정적 정보와 함께 동적 정보가 사용자에게 적절히 제공되어야 한다. [1]에서 소프트웨어의 정적인 정보와 동적인 정보를 표현하기 위해 매개 언어 SRL(*Software Representation Language*)을 사용했다. <그림 1>과 같이 원시언어는 파서를 통해 SRL로 변경된 후 부석을 통해 정·동적 정보를 얻는다. 정적 정보는 원시언어를 SRL로 변경한 후 SRL을 분석함으로 얻고 동적 정보는 자바 가상 기계[2]를 통한 가상실행을 통해 얻는다. 본 논문에서는 가상 실행을 통해 동적 정보를 얻는 과정을 기술한다.



<그림 1> SRL의 시스템 구조

본 연구는 한국과학재단 특정기초연구 (과제번호 1999-2-203-003-3) 지원으로 수행되었음.

SRL을 실행시키기 위해 각 노드에 합당한 의미 규칙(semantic rule)을 구현하여 자바의 클래스 바이트 코드로 만든다. 이를 SRL 실행 라이브러리라 한다. SRL 실행 라이브러리는 윈도우 시스템의 DLL 파일처럼 자바 가상 기계에 의해 실행 중 동적으로 로딩되어 동적 실행 가능하다.

SRL을 통한 실행에는 두 가지 방법이 있다. 첫 번째는 메모리 상태의 SRL을 사용해서 각 노드의 SRL 실행 라이브러리를 호출하여 실행한다. 시스템은 구문노드, 수식노드, 단말 수식노드 순으로 트리 형태를 이룬다. 따라서 시스템의 실행은 단말 수식노드부터 실행해서 시스템까지 실행하는 bottom-up 방식으로 이루어진다. 두 번째 방법은 첫 번째 방법에서 호출하는 라이브러리를 클래스 바이트 코드로 번역함으로 SRL을 메모리에 로드하지 않고 바로 실행 가능하도록 하는 방법이다.

2. 관련연구

일반적인 디버거는 순방향 실행만이 가능하다. [3]에서는 순방향 실행과 역방향 실행이 가능하도록 함으로 문제 영역에 빨리 도달할 수 있는 방법을 개발했다. [3]에서 사용한 방법은 구문 단위로 역실행문을 만들어서 역방향 실행을 수행했다. 역실행문을 만들 수 없는 경우에는 변수의 변화를 기록한 추적 파일을 사용해서 역방향 실행을 수행했다. 이런 방법은 많은 구문에 대한 역실행문이 존재하지 않을 수 있으며 복잡한 문장의 경우 세부적인 역실행이 불가능하다.

컴파일하기 위한 파스 트리의 과정은 단말 노드에서 루트 노드까지 실행하는 bottom-up 방식으로 이루어진다[4]. 각 노드를 실행하는 과정에서 각 노드의 의미 규칙을 평가함으로 파스 트리가 평가된다.

SRL의 메모리 상태는 컴파일 과정의 파스 트리와 유사한 트리 구조를 갖는다. 구문 노드가 루트 노드가 되고 수식 노드가 중간 노드가 되며 변수, 상수, 예약어는 단말 노드가 된다. SRL의 평가는 파스 트리의 평가 방법처럼 bottom-up 방식으로 이루어지며 각 노드의 의미 규칙은 SRL 실행 라이브러리를 호출하여 평가된다. 역방향 실행도 순방향 실행과 마찬가지 방법으로 평가된다. [3]과 다른 점은 수식 노드 단위로 역방향 실행을 수행함으로 보다 자세한 역방향 실행이 가능하다.

3. SRL의 가상 실행

3.1 SRL

SRL은 시스템을 구성하는 소프트웨어, 하드웨어, 통신망, 형상, 요구사항, 설계, 성능 등에 관한 정보를 저장소에 보관 및 관리하고 이를 재·역공학 과정에서 사용자의 특정 목적에 부합하게 표현하고 분석, 이해 및 평가하기 위한 메타언어이다. SRL에서 각 노드와의 관계는 구문과 구문의 관계를 표현하는 SS 튜플과 수식 사이의 관계를 표현하는 SE 튜플로 표현한다. 동적 정보는 SE튜플에 삽입된다.

SRL은 구문을 AST(Abstract Syntax Tree) 형태로 표현한다. 시스템과 구문 노드, 수식노드는 아래와 같이 구문은 구문 노드로 표현되고 상세 정보를 위해 수식 노드를 하위 노드로 갖는다. 소프트웨어의 중첩 구조에 따라 구문 노드는 형제 노드, 하위 노드로 연결된다. *If* 구문노드의 경우에는 참일 경우의 서브블록과 거짓일 경우의 서브블록을 표현하기 위해 두 개의 하위 노드를 갖는다.

수식 노드는 또 다른 수식 노드를 하위 노드로 가질 수 있으며 단말 수식노드는 변수, 상수, 예약어 등의 노드가 된다.

```
시 스 템  := 구문 노드 + … + 구문 노드
구문 노드 := 수식노드 + … + 수식 노드
수식 노드 := 수식노드 + … + 수식노드
           | 변수 | 상수 노드 | 예약어
```

3.2 SRL 실행 방법

시스템과 구문 노드, 수식노드의 실행은 아래와 같이 구문의 실행은 수식 노드의 실행으로 구성되고, 수식 노드의 실행은 하위 수식 노드의 실행으로 구성된다.

구문노드의 실행순서는 다음 형제 노드 순으로 순차적으로 이루어진다. 하위노드가 있을 경우 하위노드를 먼저 실행한다. 하위 노드의 모든 형제 노드가 실행 된 후 다음 형제 노드를 실행한다. 하나의 구문 노드는 하위 수식 노드를 실행한 후 각 구문의 정의된 동작을 수행한다. 수식 노드도 마찬가지로 하위 수식노드를 실행한 후 각 수식 노드에 정의된 동작을 수행한다. SRL의 평가는 변수, 상수, 예약어 등의 단말 수식노드부터 평가되어 구문노드를 평가하는 bottom-up 방식으로 이루어진다.

```
시스템 실행 := 구문노드 실행 + … + 구문노드 실행
구문노드 실행 := 수식노드 실행 + 구문 의미 규칙
수식 노드의 실행 := 수식노드 실행 + 수식 의미 규칙
           | Variable 노드 실행 | 상수 노드 실행
```

SRL을 통한 실행은 별도의 컴파일 과정 없이 동적으로 이루어져야 한다. SRL 실행 라이브러리를 동적으로 호출함으로 컴파일 과정 없이 동적 실행을 할 수 있다. 또한 실행 중 동적 정보를 SE 튜플에 보관해야 한다. SE 튜플에 보관되는 값은 주로 변수의 활용 정보가 들어간다. 변수의 정보는 변수의 값이 변하는 *define* 정보와 변수의 값을 사용하는 *use*정보가 있다. *define*은 할당문의 *left-hand-side*에서 사용되는 경우이고 *use* 정보는 *right-hand-side*에서 사용되는 경우이다. 변수의 값은 문장의 중첩 구조에 따른 해쉬 테이블에 저장한다.

SRL의 가상 실행 방법에는 두 가지 방법이 있다. 첫 번째 방법은 메모리 상태의 SRL을 사용하여 동적 실행하는 방법이다. 각 노드는 의미 규칙을 실행하기 위해 적절한 SRL 라이브러리를 동적으로 호출하여 실행된다. 두 번째 방법은 첫 번째 방법으로 호출하는 메쏘드 호출을 클래스 바이트 코드로 만드는 방법이다. 아래와 같이 원시언어는 동일한 의미를 갖는 SRL로 변경되고 SRL은 다시 동일한 의미를 갖는 SRL 실행 라이브러리 호출로 변경된다.

C -> SRL -> 클래스 바이트 코드

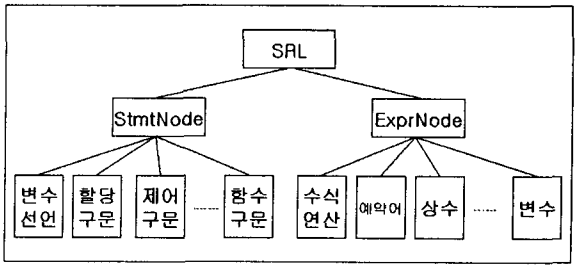
첫 번째 방법은 하위 노드의 정확한 실행 정보를 얻을 수 있지만 실행을 위해 모든 SRL을 메모리로 로딩해야 된다. 두 번째 방법은 SRL을 메모리로 로딩할 필요 없이 바로 자바 가상 기계를 통해 실행 가능하므로 메모리의 낭비를 줄이고 속도를 향상시킨다.

3.3 SRL 실행 라이브러리

SRL을 실행시키기 위해 각 노드에 합당한 의미 규칙을 구현하여 자바의 클래스 바이트 코드로 만든 SRL 실행 라이브러리를 만든다. SRL 실행 라이브러리는 윈도우 시스템의 DLL 파일처럼 자바 가상 기계에 의해 실행 중 동적으로 로딩되어 동적 실행 가능하다. SRL 실행 라이브러리는 순방향 실행을 위해 메쏘드와 역방향 실행을 위한 메쏘드를 정의한다.

3.3.1 클래스 계층도

<그림 2>는 SRL 실행 라이브러리의 클래스 계층을 표현한 그림이다.



<그림 2> SRL 실행 라이블러리 계층도

SRL은 모든 노드의 수퍼 클래스로서 순방향, 역방향 실행을 위해 다음과 같이 *forward*(), *backward*()를 추상 메쏘드로 정의한다

```
class SRL
{
   ......
   Object forward(Object aParameter);
   Object backward(Object aParameter);
   ......
}
```

StmtNode는 구문 노드의 의미 규칙을 정의한 클래스로서 *forward*(), *backward*()외에 구문노드의 형제 노드와 하위 노드를 얻기 위한 별도의 메쏘드가 추가된다.

```
class StmtNode
{
   ......
   Object getSibling();
   Object getSubStmtNode();
   ......
}
```

ExprNode는 수식 노드의 의미 규칙을 정의한 클래스로서 *forward*(), *backward*()외에 하위 수식노드를 얻기 위한 메쏘드가 추가된다.

```
class ExprNode
{
   ......
   Object getSubExprNode0();
   Object getSubExprNode1();
   Object getSubExprNode2();
   ......
}
```

SRL의 실행은 단말 수식 노드부터 bottom-up 방식으로 이루어지므로 단말 노드의 실행은 구체적인 값을 리턴 한다. 상수는 구체적인 상수 값을 리턴하고 예약어는 문자열로 예약어를 리턴 한다. 변수의 경우는 약간 복잡하다. l-value로 사용되면 해쉬 테이블에 저장된 변수 객체를 리턴하고 r-value로 사용되면 변수에 저장된 값을 리턴 한다.

```
class NumberNode
{
   ......
   String number;
   int    value;
   int follow( ) { return value;  }
   ......
}
```

```
class predifinedNode
{
   ......
   String predefinedName;
   String forward() { return predefinedName; }
   ......
}
```

```
class VariableNode
{
   Object  forward(boolean LOrR)
   {
     l-value 일 경우
       return hashtable에서 ID에 해당하는 변수 리턴
     r-value 일 경우
       SE 튜플에 값 저장 (use, value, counter)
       return hashtable에서 ID에 해당하는 변수의 data 리턴
   }
   String forward()  // variable Node에서 해쉬테이블에 추가
   {                 // 하기 위해 변수 이름 리턴
     변수 이름 리턴
   }
}
```

3.3.2 순방향 실행

순방향 실행을 위해 각 노드의 클래스는 *forward*() 메쏘드를 구현한다. 순방향 실행 도중 변수가 선언되면 해쉬테이블에 변수를 추가한다. 변수가 사용되었을 경우 SE 튜플에 값을 저장한다. SE 튜플에 값을 저장할 때 실행 순서를 표시하는 카운터 값을 같이 저장한다. 카운터는 자세한 실행 순서를 알기 위해 수식 노드 단위로 증가한다. 이 카운터 값은 역방향 실행을 수행할 때 사용된다.

순방향 실행에서 주의할 점은 같은 연산이더라도 변수 자료형에 따라 다른 연산을 수행한다는 점이다. 예를 들어 *plus* 연산을 수행할 때 정수의 연산과 실수의 연산은 다른 클래스 바이트 코드로 실행되므로 구현할 때도 구별을 해줘야 된다.

3.3.3 역방향 실행

역방향 실행을 위해 각 노드의 클래스는 *backward*() 메쏘드를 구현한다. 모든 노드에 대한 역방향 실행이 존재하는지 의문이 생길 수 있다. 이 논문은 동적 정보는 변수의 변화에 초점을 맞췄기 때문에 모든 구문에 대해 *backward*()를 구현할 필요는 없다.

역방향 실행에서는 *forward*()와 역순으로 수식노드 단위로 카운터 값을 감소시킨다. 변수 선언 노드에서는 해쉬 테이블에 저장된 변수를 삭제한다. 변수가 l-value로 사용될 경우 SE 튜플에 들어있는 값을 사용하여 예전의 값으로 변수 값을 변경한다. 정확한 동적 정보를 얻기 위해 카운터 값을 사용한다. r-value의 경우 변수 값은 변하지 않음으로 카운터 값만을 감소시킨다.

4. 결론 및 향후 연구과제

본 논문에서 SRL을 통해 가상 실행을 할 수 있음을 보였다. 모든 언어는 SRL로 변경되고 SRL은 자바 가상 기계를 통해 실행됨으로 모든 언어에 대해 가상 실행을 수행할 수 있다. 자바 가상 기계는 객체 지향을 위해 설계되었기 때문에 절차 지향 언어 뿐 아니라 객체 지향 언어도 가상 실행을 할 수 있다. 그러나 다중 상속은 자바 언어에서뿐 아니라 자바 가상 기계에서도 지원하지 않는다. 이런 특징 외에도 자바 가상 기계가 지원하지 않는 다른 언어들의 특징이 있다. 템플리트, 느슨한 타입 점검, Lisp의 람다(λ) 표현식 등이 그것이다. 이런 특징을 자바 가상 기계를 통해 대체할 수 있는 방법을 개발해야 한다.

[참 고 문 헌]

[1] 정양재, 박성옥, 이문근, "*가상실행에 기반을 둔 SW의 이해에 관한 연구.*" 정보과학회 학술발표논문집 제 27권 1호, pp. 579-581, 2000.

[2] Joshua Engel. "*Programming for the Java Virtual Machine.*" Addison Wesley. 1999.

[3] R.Mall, "*Revere Execution of Programs.*" ACM SIGPLAN Notices, April, pp. 61-69, 1999.

[4] Alfred V. Aho "*Compilers Principles, Techniques and Tools.*" Addison Sesley. 1986.