論理学的アプローチによるJAVA仮想機械の諸性質の分析および実装への応用

樋口 智之 (010093)

北陸先端科学技術大学院大学 情報科学研究科

2002年2月15日

キーワード: JAVA仮想機械, バイトコードベリファイア, 型システム, 型推論.

1 研究の背景および目的

JAVAは現在最も成功したプログラミング言語の一つである. その大きな特徴は異機種間のネットワークコンピューティングに対応していることであり, それを支えるのはJAVA仮想機械 (JVM) と呼ばれる仮想的なハードウェアである. JVMはプログラムを実行する前にその安全性を検証する. この中でJAVAバイトコード列を検証する過程はバイトコードベリファイアと呼ばれ, 実行中にバイトコードが未定義な動作をするのを防ぐことを目的とする. バイトコードベリファイアはJVMを構成する重要な部分の一つであるが, その仕様は自然言語で書かれているため曖昧で, また数学的な正しさが証明されていない. この問題を解決するため, 多くの研究者がJVMを形式的に記述しようと試みてきた. その大部分がJAVAバイトコードに対する型システムの導入である.

これらの型システムはプログラムの整合性を検証するためには有用であるが, まだ研究が浅いためその性質は完全に明かにされていない面がある. またよく知られたラムダ計算の型システムΛとは本質的に異なるため, そこで研究されてきた優れた性質を適用することが困難である.

本論文の目的はJAVAバイトコードを型理論的に分析し, バイトコードベリファイアの設計や実装への基礎を構築することである.

2 本研究の貢献

上記の目的を達成するために, 本研究が解決した具体的な問題は以下の通りである.

1. Javaバイトコードに対する型システムを構築しその健全性を証明する. 型システムを定義する上で基礎となるのはSequential sequent calculusと呼ばれる証明システムであり, これをJavaバイトコードに対して拡張する.

2. 定義した型システムを多相型システムに拡張し, それに対して型推論アルゴリズムを構築する. さらに, バイトコードベリファイアは型推論アルゴリズムとして記述できることを示す.

3. 型推論アルゴリズムを実装し, 本研究が示す型理論的なアプローチが JAVA のベリファイアを系統的に実装する基礎として有効であることを実証する. さらに, この論理学に基づく方式と Sun のベリファイアとの能力を比較する.

以下でこれらの概要について述べる.

2.1 Java バイトコードに対する型システム

Sequential sequent calculus はマシンコードに対応する証明システムである. この証明システムでは, マシンコードの個々の命令 I は次のような推論規則として表現される.

$$\frac{\Delta_2 \rhd I : \tau}{\Delta_1 \rhd I{\cdot}C : \tau}$$

ここで C はコード列,Δ はマシンメモリを表す. この型付け規則は I がマシンメモリを Δ_1 から Δ_2 へ変化させる命令であることを表す. リターン命令は証明システムにおける次のような形をした始式である. $\tau{\cdot}\Delta \rhd \texttt{return} : \tau$ プログラム (コード列) はこのような推論規則から合成される証明に対応する.

JVM ではバイトコードのそれぞれの命令はローカル変数とオペランドスタックと呼ばれる領域を操作する. これをモデル化するために, バイトコードの命令 I は次のような形の型付け規則として表される.

$$\frac{\Gamma_2, \Delta_2 \rhd B : \tau}{\Gamma_1, \Delta_1 \rhd I{\cdot}B : \tau}$$

ここで Γ, Δ はそれぞれローカル変数およびオペランドスタックを表す. JAVA バイトコードにはジャンプ命令が存在するため, プログラム (メソッド) は連続した一つのコード列ではなくラベル付けされたいくつかのコードブロック B の集合である. ラベルはジャンプ命令の引数として使用される. ジャンプ命令は既に存在する証明への参照として解釈できる. 例えば, l でラベル付けされたブロック B へジャンプする命令である $\texttt{goto}(l)$ は次のような型付け規則で表現される. $\Gamma, \Delta \rhd \texttt{goto}(l) : \tau$ (if $\Gamma, \Delta \rhd B : \tau$)

Java バイトコードの大きな特徴は, サブルーチンと呼ばれる一般のコードブロックとは異なる内部的な手続きを有することである. サブルーチンはメモリ状態を変化させるだけのコード列と考えることが出来る. サブルーチン SB が呼び出し元のメモリ状態 Γ_2, Δ_2 を Γ_1, Δ_1 へ変化させるとき,

$$SB : \langle \Gamma_1, \Delta_1 \rhd \tau // \Gamma_2, \Delta_2 \rhd \tau \rangle$$

と表す. つまり,SB は与えられた証明 $\Gamma_1, \Delta_1 \rhd \tau$ を $\Gamma_2, \Delta_2 \rhd \tau$ へ拡張する関数としてモデル化することが可能である.

2.2 型推論

JAVA バイトコードベリファイアの問題は型システムにおける型チェックに相当する. コードブロックは型付けされていないラベルを通して他のブロックを参照するため, プログラムの型チェックを行なうためには, 型推論アルゴリズムを構築する必要がある. さらにサブルーチンはメソッド内の任意の異なる場所, つまり異なるマシン状態で呼び出されるため, サブルーチンに対して多相型を与える必要がある. この多相性を表現するために, プログラムをサブルーチンの集合 M^s およびブロックの集合 M^b に分割し, プログラムをラムダ計算の多相的 let 式に似た項

$$\text{let } M^s \text{ in } M^b$$

と表現する. このプログラムに対して,ML の型推論アルゴリズム $\mathcal{W}$ と同様の考え方を適用することにより, プログラムの方推論アルゴリズムを構築することが可能になる.

2.3 実装と評価

型推論アルゴリズムを Standard ML により実装する. このシステムは最初にクラスファイルからバイトコード配列を読み込み, それをコードブロックの集合に分割する. 次に, 型推論アルゴリズムを適用することで, そのメソッドの型を推論する. このシステムを使用して,Sun のベリファイアと比較することで型システムの能力を検証した. その結果, それらは排反する能力を持つことが示された.

3 今後の課題

本研究が示した論理学的なアプローチを拡張し,JVM の分析や実装の基礎を理論的に確立するためには, 次の 3 点が重要と思われる.

- **種々の特徴を取り入れる**
  本論文では JVM のサブセットに対して型システムを構築した. 今回無視した例外やインターフェースなどの種々の特徴に対して型システムに対して型システム拡張する.
- **JVM を超える JVM**
  メソッドを引数や返り値として持つ高階のメソッドなど,JVM に対して様々な先進的な特徴を取り入れる.
- **より柔軟な型推論アルゴリズム**
  構築した型推論アルゴリズムはまだ不完全であり, いくつかの改善の余地がある. ブロックを多相型にするなどの改良が考えられる.