自己反映的逐次プログラミング言語の効率的なコンパイル手法について

佐伯 豊

北陸先端科学技術大学院大学 情報科学研究科

1997 年 2 月 14 日

キーワード： 自己反映計算, 再利用, モジュール化, コンパイラ, Scheme, メタレベルアーキテクチャ.

1 背景

自己反映的 (reflective) なシステムとは、そのシステム自身が、自己の構造や、計算過程に関する計算をおこなうことを可能とするような構造をもったシステムである。その利点は、システムが実行時の状況に応じて柔軟に自分自身の構造を変化させることが可能であることと、ユーザーによる意図的な拡張が可能であるということである。そのため特に言語処理系や、OS といった広義の記述系にたいして自己反映計算の能力を与えることによって、移動計算システムや分散システムなどの高度で複雑なシステムをより秩序立った方法で構築することが可能になることが示されている。特に自己反映的な言語については、様々なシステムが実装されており、その有効性が実証されている。しかしこうした自己反映的な言語を実現する場合、多くの処理系が meta-circular インタプリタによる拡張部分のメタレベル実行という方法をとるため、プログラムの解釈実行のためのオーバーヘッドによって実行効率の低下が生じるという問題点がある。さらに、プログラミングのしやすさの観点からすると、自己改変の影響がプログラム全体におよぶこと、およびメタレベルで扱う構造の抽象度が低いことから、記述の際、常にそれまでになされたすべての拡張事項に関して留意しておく必要があるために、メタレベルに関する記述の再利性が低く、プログラミングが困難であるという問題があげられる。このように、自己反映的な言語の処理系はいまだに実用性において多くの問題点を残している。

本研究では、Scheme ベースの自己反映的な言語処理系 Fleet の設計および実装をおこなった。その際、特に言語拡張に関する記述の結合関係に注意し、記述部分の再利用を可

能にした。また処理系のコンパイラとしての実装をおこない、プログラムの効率的な実行を試みた。

2 アプローチ

本研究におけるアプローチとして、拡張部分の記述のための、より抽象的な手段をもちいることで、拡張に関するより整理された枠組を与えることを基本方針とし、言語の設計をおこなった。Flect では、言語の意味を決定する部分を、処理系から切り離して提供している。その構造は、実行制御に関係した計算状態に関する言語の意味を決定するオブジェクト (control-object) と、それ以外の計算状態に関する言語の意味を決定するオブジェクト (control-object)によって与えられる。またそれぞれのオブジェクトの性質は、各計算状態の性質を表現したモジュールの組み込みによって決定される。

2.1 意味モジュール

Flectにおける言語の拡張とは、言語システムに対して環境や継続のような計算状態を新たに付け加えることである。概念的にはインタープリタにおける評価器を、計算状態ごとに分解し、それぞれを組合せ可能なモジュールとして定義することで、柔軟な言語の構造を実現している。

それぞれのモジュールでは、新たにシステムに組み込まれる計算状態、および単独の計算と、複合的な計算におけるシステムの計算状態の変更のための操作を定義している。各オブジェクトのふるまいは、こうしたモジュールの結合によって決まる。すなわち、各オブジェクトは、モジュールによって与えられた計算状態に特化したインタープリタであると考えられる (ただし式そのものは扱わないため、ソースコード自体はコンパイル可能である)。

ユーザーが、オブジェクトに対して、実行時におこなうことが可能な操作は、以下に示すものである、

- あらかじめ定義しておいたモジュールを組み込むこと。
- プログラム中からモジュールを定義し、組み込むこと。
- オブジェクトの状態にアクセスすること。

これらの操作によってユーザーは間接的に言語の意味そのものに変更を加えることが可能になる。なぜなら、オブジェクトが処理系自身のモデルであり、モジュールの集合がその表現であるため、モジュールに関する操作はそのままオブジェクトのふるまいを変化させ、その結果が即座にシステム自身に反映される、すなわち因果的結合が存在するためである。

2.2 研究の成果

こうしたアプローチによって次に示すような機能を実現することができた。

- 言語拡張に関する記述をプログラム自身とは独立したモジュールとして与えることで、その扱いが容易になった。
- 制御に影響をあたえる状態を、その他の状態と分離することで、各種のモジュールを組み合わせて用いることが容易になり、言語の高い拡張性を実現することができた。
- 言語の意味に対する拡張部分の構造をプログラムからできるだけ切り離すことで、十分な拡張性を保ったまま言語処理系をコンパイラとして実装することを可能にし、効率的な実行を実現した。
- 計算の意味を first-class なオブジェクトとして実現することが可能なため、プログラムの実行の流れのなかのある区間を指定した拡張をおこなうことができる。

3 結論

Flect ではモジュールが実現する言語の意味の構造を、独立した二つのオブジェクトとして明確に決めているため、モジュールの結合を比較的安全におこなうことができる。また、モジュールの組み込みを、領域を指定し、必要に応じておこなうため、少ない数のモジュール同士の整合性のみを考慮するだけでよく、プログラムの見通しが良くなる。その結果 Flect は、言語拡張をおこなう単位である各モジュールの高い再利用性を提供することができた。