㊾ Meta-Interpreters and Reflective Operations in GHC

J.Tanaka(富士通，日本)

発表要旨

GHCにおけるメタプログラミングとリフレクションについての発表であった。OS等を記述する場合，メタレベルの記述は不可欠である。ad−hocなメタ述語ではオブジェクトレベルとメタレベルの区別が不明確であり，また成功／失敗等の計算状態を知ることがむずかしい。システマティックな方法が必要になってくる。

まずGHCのGHC自身によるメタインタプリタを考え，次にそれを以下のように拡張した。(1)オブジェクトプログラムの成功／失敗を表す引数を設ける。(2)スケジューリングキューを明示的に取り込む。(3)リダクションの回数を表す変数を導入する。(4)変数のバインディングを明示的に管理する。

リフレクションの実現は，上の(1)から(4)によって（メタレベルで）アクセスできるようになったものをオブジェクトレベルからもアクセスできるようにすることによって行った。実際にPSI−Ⅱ上に実現されたプロトタイプの速度は決して遅くないことが示されていた。

質疑応答

質問：FGHCでは一般に停止や失敗の検出はできないのでは。

回答：non−atomic variableの仮定のもとではたしかにそうだが，我々の場合はatomicであることを仮定している。

質問：誰が最初に"Prolog in Prolog"を書いたのか。

回答：わからない。私はDEC−10 Prologのユーザーガイドで読んだのだが。

質問：並列の場合のリフレクションは，逐次実行のときとどうちがうのか。

回答：GHCそのものは本来並列だが，我々の場合，スケジューリングキューを導入して逐次化してしまっている。まだ本質的に並列性がきいてくるものは考えていない。

質問：メタレベルとオブジェクトレベルで変数の共有をおこなったとすると，変数管理が非常に複雑になると思うのだが。

回答：今のところそのような状況に関してはすぐには答えられない。

質問：（オブジェクトプログラムの成功／失敗を引数に返すメタインタプリタで）組み込み述語では一般に成功／失敗は値として返すわけではないので，reduceを定義しなおすべきでは。

回答：それは論文に出ているので，proceedingsをみてほしい。