トランスレータ方式のエージェントの実装

3 F − 8

下川僚子　梅村恭司
豊橋技術科学大学　情報工学系

1　はじめに

分散処理の基礎技術として, エージェントがある. エージェントを実現する基礎技術は, 異なる機種の間でプロセスを移送することである [1][2][3][4]. われわれは, トランスレータ方式のエージェント言語を実現した. 具体的には, 原理的な動作を確認する簡単なエージェント言語を設計し, その言語からCへのトランスレータと, エージェントを移送するためのライブラリの整備を行った. その結果, 適切に設定されたUnixで動作しているシステムについて, マシンのアーキテクチャに依存せず, エージェントの動作コードがネイティブコードで動作できるようになった.

2　ターゲット言語と基本方針

我々は, ターゲット言語としてCを選びトランスレータを作成した.Cは高級アセンブラと呼ばれることがあり広く使われている言語であるためである.

エージェントの記述言語を方式を確認するため単純なものを設定し、それをP0と呼ぶことにした. 整数をデータの型として, 分岐, 繰り返し, 再帰できる関数呼出し, それにエージェントの移動先を指定する基礎機能を持つ. トランスレータ方式を実現できるようにするために, 制御と呼出しの情報が機種非依存で移動できる機能が必要である. このために, プログラムの実行コンテクストをテキスト化できるような機能を持つ. ここで工夫が必要だったものは, 関数呼び出しの処理である.

3　関数呼び出し

Cの実行途中の状態を機種独立なテキストに変換する際, 関数の呼び出しの状態をテキストから復元することは, 一般には困難である. なぜならば, 戻りアドレスはプログラムのコードの内部のエントリーであり, その値はC言語からは直接意味を与えられていないからである.

そこで, 関数の呼び出しはC言語の組み込み機能を利用せず, 自分で用意したグローバルなスタックを使い, 呼びだし戻り場所を関数名に対応させてスタックにプッシュすることで実現した. このため, 関数呼び出しについては性能のペナルティーが生じる. しかし, 関数呼び出しのないループなどについてはCと同じ速度で動作させることができる.

4　関数ポインタのテキストとの変換

関数ポインタは機種ごとに異なった値となる. 機種に依存せずに関数のポインタのスタックを移送するためには工夫が必要である. トランスレータは関数のスタックに積まれる可能性のある関数の全体を数えあげることができ, 関数ポインタの格納にはグローバルな配列を使用しているので, 関数ポインタを対応する関数名へと変換するモジュールをトランスレータが自動生成するようにした. 具体的には, 与えられた関数のポインタを既知の関数のポインタと比較して, 一致したら, 関数の名前の文字列を求める処理を行うモジュールである. その詳細については文献 [5] に譲る.

また, 関数名から関数ポインタへの変換も同様に実現できる. これもトランスレータが自動生成する. これは, テキストで与えられた関数名と既知の関数の名前を比較し, 一致すればその関数のポインタを求めるものである.

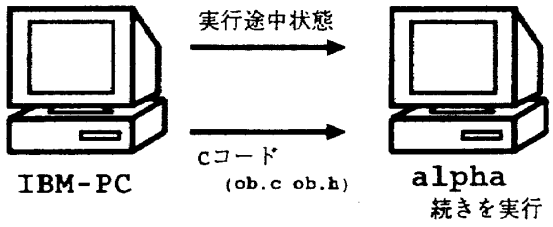
Implementation of Agent Language Using Translator

Ryoko Shimokawa and Kyoji Umemura
Department of Information and Computer Sciences, Toyohashi University of Technology

5　トランスレータ

エージェント言語P0をCのプログラムに翻訳するトランスレータは,C言語で記述したものである.これは,約1500行のプログラムである.現在最適化は行なっておらずコードの効率は悪いが,きちんと規約に基づいてコードを生成しており,方式の実証的な確認はできている.

6　エージェントの移送



実際に実行する際は,まずP0のプログラムを作成しトランスレータを用いる.するとob.cとob.hが作られる.これをgccでコンパイルしa.outを生成する.このa.outを実行することで,実際に実行が始まる.エージェントの移送プログラムが実行されると引数に指定された名前で,行き先のシステムが特定される.そのとき,実行の途中状態をテキストファイルに生成する.ob.c, ob.h, とこのテキストファイルを転送し,転送先の計算機でコンパイルを行なう.引数にテキストファイルを指定するかたちで,相手のプロセスを起動し,自分は終了する.

7　考察

中間コードを設定して,エージェントを実現するのが一般的であるが,トランスレータを用いてもエージェントが実現できることを示した.一つのトランスレータで,多くの機種に対応することできることが本方式の利点の1つである.

エージェント言語で問題となるセキュリティや,環境の変化への対応は,翻訳処理をする段階で,ある程度の対処ができる.対処を行うように改良すれば,セキュリティーや環境への対応は中間コードで実現するものと差はない.

エージェントに情報が有用かどうか判断させる状況では,エージェントの動作速度が問題となる.JITが必要であるのも,エージェントといえども動作速度が重要視される場合があるからである.このような状況では,トランスレータを利用した方式が有理になる状況が考えられる.

8　まとめ

トランスレータ方式でエージェント言語を実現する方法を論じた.そして,その方法にしたがって作成したトランスレータを実際に作成して,実現できることを確かめた.

参考文献

[1] Comm. ACM Special Issue on Intelligent Agents, Vol.37, No.7 (1994)

[2] Telescript Technology: The Foundation for the Electronic Marketplace, General Magic White Paper (1994)

[3] M.Genesereth and S.Ketchpel:"Software agents",Communications of the ACM, 37(7) (July 1994)

[4] 多田好克："移植性のあるCの継続ライブラリ", 情報処理学会プログラミング-言語・基礎・実践-研究会,18-14,PP.105-112,1994

[5] 下川僚子，山本英雄，梅村恭司:"トランスレータ方式のエージェントの実装"，情報処理学会研究報告98-OS-78,pp31-pp38,1998