5Y-7

ＬＩＳＰからＣへのトランスレータ

関根　聡　　早川　佳宏　　安藤　敦史　　上田　謙一
松下電器産業　東京研究所

１．はじめに

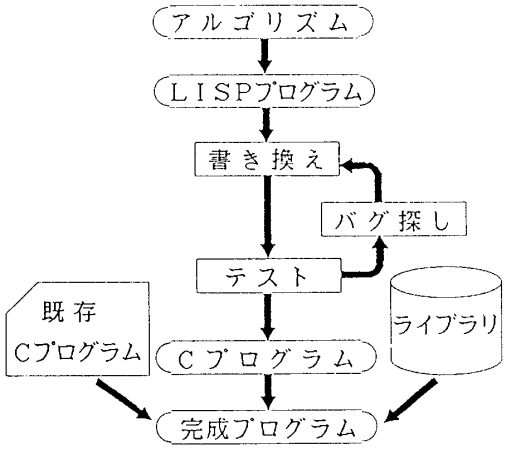
現在、ＡＩ応用システムをはじめ、数多くのプロトタイピングがＬＩＳＰ処理系で行なわれている。ＬＩＳＰは柔軟なデータ構造、データ宣言が不要、インタプリタで実行可能等の特徴によりプログラムの記述が容易である。また、最近のＬＩＳＰ処理系には強力なデバック環境が用意されているため、プログラムのプロトタイピングには適している。しかし、処理速度が遅い、オブジェクトサイズが大きい等の問題がありこれが応用プログラムの普及の妨げになっている。

そこで我々は、応用プログラムの普及を促進させるために、実行速度が速く、かつオブジェクトサイズが小さくなるようなＣプログラムを自動的に生成する、ＬＩＳＰからＣへのトランスレータの開発を行っている。以下、このトランスレータの概念及び実現へのアプローチについて述べる。

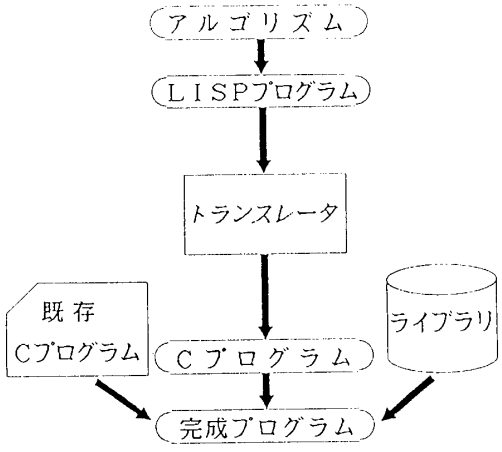
２．基本概念

従来の方式では（第１図）、プロトタイピングに適しているＬＩＳＰ処理系を用いてアルゴリズムを具現化し、実用化の際には処理速度の向上やオブジェクトサイズの縮小化のために、人間の手によってＣ言語やアセンブリ言語等に書き替えていた。しかし人手による書き替えにおいては、書き替えに要する時間が膨大であり、さらにバグが入る危険性が高い。

我々のトランスレータは（第２図）、ＣＯＭＭＯＮ　ＬＩＳＰで記述されたプログラムをＣ言語のプログラムに自動的に変換することを目的としている。



第１図．従来の方法



第２図．概念図

LISP to C translator
Satoshi Sekine, Yoshijiro Hayakawa, Atsushi Andou, Kenichi Ueda
Tokyo Research Laboratory Matsushita Electric Industrial Co.,Ltd.

ＬＩＳＰからＣへのトランスレータとしては、ＫＣＬ（京都コモンリスプ）コンパイラがよく知られている。しかし、ＫＣＬコンパイラはコンパイルしたオブジェクトをＬＩＳＰインタプリタにロードすることを前提としているので、生成されたＣプログラムの構造はＬＩＳＰインタプリタの処理方式に依存している。それに対して、我々のトランスレータはＬＩＳＰインタプリタにロードすることを前提としておらず、生成されたＣプログラムはＬＩＳＰライブラリや他のＣプログラムと結合して単独に実行する。すなわち、ＬＩＳＰ処理系には依存せず、独立して実行できる。そのために、次のような特徴が考えられる。

（１）　処理速度の向上
（２）　オブジェクトサイズの縮小
（３）　他のＣプログラムとの融合性

以上、我々のトランスレータの基本概念の概略を述べた。次に、上記の特徴を実現するためのアプローチについて記す。

３．実現のためのアプローチ

まず、処理速度の向上という観点からすると、関数の呼出し、復帰処理、制御、分岐などのコードの高速化、レジスタ宣言、インライン展開等が考えられる。そして、オブジェクトサイズの縮小、他のＣプログラムとの融合性の観点からするとデータ構造の変換等が考えられる。これらに対するアプローチについて記す。

（１）関数の呼出し、復帰処理
　　関数の引き数や戻り値をＬＩＳＰ処理系独自のスタックを使用せず、極力Ｃ言語のランタイムスタックを使用する。
（２）データ構造の変換（データサイズの縮小）
　　・ｆｉｘｎｕｍ型　－－－＞　即値データ
　　・不必要な属性データの削除
　　・ＬＩＳＰの構造体　－－－＞　Ｃの構造体
（３）制御、分岐などの最適化
　　・関数呼出しの内、分岐命令で置き換え可能なものを抽出すること。
　　（ｔａｉｌ－ｒｅｃｕｒｓｉｏｎの最適化）
　　・条件判定部の簡素化
（４）レジスタ宣言の活用
　　関数の引き数、多用している変数等に可能な限りレジスタ宣言を行う。
（５）インライン展開
　　可能な限り多くの関数を直接オブジェクト中に展開する。

４．まとめ

我々が開発を行っているトランスレータの基本概念と、実現のためのアプローチを述べた。現在、これらの実現へのアプローチを基に、トランスレータを作成中である。なお、本稿では記さなかったが、このトランスレータには万能関数や動的に変化する環境への対応が完全ではない等の問題が存在する。今後、これらの問題についての検討をしていく。