報告番号 ※甲第2538号

主論文の要旨

題名

代数的手法に基づくコンパイラの仕様記述・検証・自動生成に関する研究

氏名 酒井正彦

主論文の要旨

報告番号	※甲第　　号	氏名	酒井正彦

FORTRAN, BASIC, COBOL, C, Pascal などの高級言語の出現によって，アセンブラや機械語によるプログラミングと比較すれば，プログラムを作成する効率が向上した．これらの高級言語で書かれたプログラムを高速に実行させるためにはコンパイラが必要であり，コンパイラは非常に重要なシステムプログラムであるといえる．

コンパイラはその複雑さ故，名人でなければ作成できないと考えられていた時代があった．しかし，文脈自由言語とその構文解析などに関する理解，ならびに，研究が進むにつれ，名人でなくてもコンパイラを作成できるようになってきた．それでもなお，他の複雑なソフトウエアと同様に，その作成や保守は大変であり，特に，コード生成などの意味処理に深く関わる部分については，まだまだ理論的な整理が不十分であり多くの問題が残されている．

読みやすい形式で仕様が記述でき，その仕様からコンパイラを自動的に作成できれば，コンパイラの作成や保守の手間を減少させることが可能になる．このように，ソース言語やターゲット言語の仕様，あるいは，それらの間の翻訳に関する仕様からコンパイラを自動的に合成するシステムはコンパイラ自動生成系と呼ばれる．

コンパイラ自動生成系が利用できれば仕様通りのコンパイラが得られる．しかしながら，正しいコンパイラを作成するためには正しい仕様が必要であるため，仕様の正しさの検証が重要になってくる．形式的な手法に基づく仕様の検証は一般には容易ではないが，すくなくとも，仕様に対して数学的な意味に基づいた形式性が必要となる．

コンパイラの入出力はプログラムであるため，プログラミング言語を形式的に記述する方法を与えることは，コンパイラの仕様記述，仕様の検証，ならびに，仕様からのコンパイラ自動生成の重要な基礎となる．

プログラミング言語の仕様記述法のうちで，シンタクスの記述とその処理に関する技術については，文脈自由言語とその構文解析などに関する多くの研究がなされ，実用的な字句解析プログラムや構文解析プログラムをそれらの形式的仕様から自動的に生成できるほどに実用的な段階に達している．

主 論 文 の 要 旨

報告番号	※甲第　　号	氏名	酒井 正彦

しかしながら，セマンティクスの形式的仕様記述法とその処理に関しては，最近，種々の研究を通して次第に理解が深められているとはいえ，シンタクスに比べればまだまだ不十分である．このため，プログラミング言語のセマンティクスの形式的仕様記述ならびにそれからのコンパイラの自動生成は現在の重要な研究課題である．

属性文法，表示的意味論，公理的意味論，操作的意味論などに基づく種々のプログラミング言語仕様記述法が研究されている中で，最近になって代数的意味論に基づく記述法が注目されている．この記述法では，プログラミング言語の意味を代数として，すなわち，抽象データ型として捉えてこれを等式の形式で仕様記述する．

代数的意味論に基づく仕様記述法は，等式論理を用いて意味を厳密に定義できるにもかかわらず，その枠組は単純であり項書換え系を利用した処理系の実現が可能である．このため，コンパイラの形式的な仕様記述法として有望である．さらに，代数的仕様記述の理論的な裏付けとなる抽象データ型の理論は1960年代後半から数多くの研究が行なわれてきており，それらの成果を利用することが可能である．

代数的仕様記述法に基づくプログラミング言語やコンパイラの仕様記述，ならびに，それらの記述からのコンパイラ生成は，ADJ グループや Mosses によって始められ，それ以後も研究が続けられている．

これらは，次の二つのアプローチに大きく分けられる．

アプローチ A: プログラミング言語の仕様を，構文領域，意味領域，ならびに，構文領域から意味領域への写像 (意味写像) として与える．これをソース言語の仕様として，言語処理系やコンパイラを生成する．この記述法は，意味領域の実現が可能であれば，プログラムを意味写像に基づいて意味領域における値を求めることによってプログラムの実行が可能になる．

アプローチ B: コンパイラの仕様を，ソース言語の構文領域，ターゲット言語の構文領域，ならびに，これらの構文領域間の写像として与える．コンパイラは，この構文領域間の写像を実現することによって自動生成できる．

主 論 文 の 要 旨

報告番号	※甲第　　号	氏名	酒井 正彦

これらの代数的手法に基づくコンパイラ自動生成に共通する問題点は，意味領域として用いられている抽象データ型の枠組に対して種々の拡張がなされているため形式性を低下させている点，もしくは，拡張をしていないためにコンパイラの記述には能力が十分でない点である．

本論文では，

1. コンパイラを記述するために十分な記述能力を持ちながら，コンパイラを純代数的に仕様記述するための枠組を与える，
2. 与えられた仕様からコンパイラを自動的に生成するシステムを構築する，
3. 与えられた仕様の検証を行なうための枠組を用意する，

を目的として研究を行なった．その内容と結果を以下に述べる．

代数的手法に基づくアプローチにおいて，実際に稼働するシステムを作成するためには，代数的に記述された抽象データ型の仕様を実現するシステムが不可欠である．そこで，第3章では，抽象データ型の直接実現システムのアルゴリズムを与え，実際にこのアルゴリズムを実装してCdimpleシステムを作成した．Cdipmleによって生成されるプログラムは，人が作成したそれと同じ働きをするLISPプログラムをLISPコンパイラによって実行するのと比較して同等の速度が得られた．このシステムは，コンパイラの自動生成に限らず，代数的手法に基づく仕様記述からのソフトウエア自動生成のためのツールとして，また，抽象データ型の機能を採り入れた言語を実現するためのツールとして有効である．

アプローチAについては，意味領域への補助関数の導入によって記述能力を向上させた純代数的なプログラミング言語の仕様記述法がいくつか提案されている．しかしながら，これらの方法ではプログラミング言語の入力の部分の意味を自然に与えることが困難であった．また，その仕様からプログラムの直接実行系の生成方法が示されていない．そこで，第4章では，アプローチAに基づく北らの記述法を採用し，この方法に従って記述されたプログラミング言語の仕様に基づいてプログラムを直接的に実行するための方法を明らかにし，実

主 論 文 の 要 旨

報告番号	※甲第　　号	氏名	酒井 正彦

際に言語処理系を生成するシステムLassを作成した．また，単純な言語ではあるが言語として基本的な機能を持つ言語PL/0, PL/0+について，Lassを用いてそれらの仕様から言語処理系を生成する実験を行なった．生成された言語処理系は効率的とはいえないが，実際にプログラムをその処理系で実行することによって仕様の意図からの誤りを迅速に発見し修正できた．これによって，Lassがプログラミング言語の設計のためのツールとしても役立つことが示された．

Lassでは，仕様がターゲット言語の情報を持たないため，コンパイラを自動的に生成することはできない．仕様にターゲット言語の形式的な情報を入れるように改良を進めると，それは，自然にアプローチBになる．そこで，第5章では，ソース言語とターゲット言語，ならびに，ソース言語からターゲット言語への変換の情報を代数的に記述したものをコンパイラの仕様とし，この仕様からコンパイラを生成する方法を検討した．その際に，これまでの方法では代数的な枠組の外で議論されていた部分についてもその枠組の中で記述できるようにした．さらに，この仕様からコンパイラを自動生成する方法を明らかにし，実際にコンパイラ自動生成システムを試作した．実際にPL/0コンパイラを仕様記述し，そのコンパイラの自動生成を行なった．PL/0コンパイラの仕様は自然に記述でき，自動生成されたコンパイラは，それ自身の効率は良くないが効率的なコードを生成した．

第5章の方法で記述されたコンパイラの仕様を検証するためには，与えられた抽象データ型の仕様が，等式の形式で与えられた性質を満たすこと，すなわち，帰納的定理であることを証明する必要が生ずる．第6章では，コンパイラの検証のための準備として，ある等式が抽象データ型の仕様の帰納的定理であることを証明するための被覆集合帰納法を提案した．この方法に従っていくつかの帰納的定理を証明することによってこの方法の有効性が示された．

第7章では，第5章の記述法で書かれたコンパイラの仕様の検証を行なうための枠組を与えた．また，第6章の帰納法を用いてPL/0サブセットのコンパイラの仕様を検証した．