報告番号	※甲 第2235号

主論文の要旨

題名

代数的手法に基づく
プログラミング言語の仕様記述法と
プログラム検証に関する研究

氏名 北 英彦

主論文の要旨

報告番号	※甲第　　号	氏名	北　英彦

本論文は，プログラミング言語の代数的仕様記述法および代数的仕様記述法で意味論が与えられた言語のプログラム検証について述べたものである．

第１章では，本研究の背景，動機について述べる．

近年，ソフトウェアの生産性，信頼性の向上が切実に要求されている．プログラミング言語の形式的仕様記述法および形式的意味論の研究は，ソフトウェア基礎論の一つであり，プログラム検証，プログラム合成，プログラム変換，コンパイラの正当性検証，コンパイラ自動生成などソフトウェアの生産性，信頼性を向上させるために現在進められている研究の基礎を提供する．

プログラミング言語の仕様記述法としては，操作的仕様記述法，公理的仕様記述法，表示的仕様記述法，代数的仕様記述法などがある．これらの中で代数的仕様記述法は，構文要素の領域（構文領域）と意味領域を多ソート代数（抽象データ型）として捉え，プログラムの意味をこの二つの代数間の準同型写像（意味写像）によって与える方法である．本方法の特徴は，意味領域を抽象データ型として代数的手法によって記述する点である．抽象データ型は，モジュール性，抽象化，情報隠ぺい，局所化というプログラミング方法論を実現する概念であり，その代数的仕様記述は，形式性，記述性，理解性を持ち，また，必要最小限の情報しか記述しないという特徴を持つ．また，意味論が等式論理に基づいており，他の仕様記述法に比べて理解がしやすい．

本論文では，このプログラミング言語の代数的仕様記述法について考察を行う．

第２章では，プログラミング言語の代数的仕様記述法を提案する．

言語の仕様は，一般に，構文の仕様と意味の仕様からなる．構文の仕様は字面としてのプログラムの集合（以下では，構文領域と呼ぶ）を記述するものである．意味の仕様では，意味を与える基礎となる領域（以下では，意味領域と呼ぶ）と字面のプログラムに意味を与える写像（以下では，意味写像と呼ぶ）の２つを記述する．

本論文で提案する代数的仕様記述法は，構文領域および意味領域を代数（抽象データ型）として，意味写像をこの２つの代数間の写像として記述する．具体的には，構文領域は文脈自由文法で，意味領域は抽象データ型の代数的仕様記述法で，意味写像は原始帰納的な意味方程式系で記述する．

主論文の要旨

報告番号	※甲第　　号	氏名	北　英彦

本方法の特徴は，意味領域の仕様を等式のみを用いて与えたことである．これにより，意味領域を厳密に抽象データ型として捉えることができ，意味写像を構文領域から意味領域への準同型写像で与えており，完全に代数的理論の枠組の中で言語の意味が記述されているので，目的言語ないしは機械とは独立に言語の仕様記述を可能にしている．そのために，記述性と読解性に優れているばかりでなく，十分な形式性を備えた仕様記述法になっており，これまでに抽象データ型の代数的仕様記述の理論の中で得られてきた成果を直接に利用することができるようになった．

さらに，本方法による仕様記述から意味が一意に定まること，すなわち，意味写像が一意に存在することを証明する．また，意味写像の性質，すなわち，その意味写像が，構文領域から，意味領域の演算記号で構成される項集合への写像 δ' とその項を意味領域上で評価する評価写像 ｅｖａｌ の合成で表されることを示す．

それと同時に，理解の助けのために，簡単ではあるがループ文を含むプログラミング言語 ＳＬ を例にとり，仕様記述法を具体的に説明する．

第３章では，第２章において提案したプログラミング言語の代数的仕様記述法の有効性を実証することを目的として，Ｐａｓｃａｌ 風言語 ＰＬ／０＋ の代数的仕様記述を与え，その説明を通して，Ｐａｓｃａｌ 風言語一般に対する代数的仕様記述の技法を明らかにする．なお，ＰＬ／０＋ は，Ｗｉｒｔｈ がその著書「アルゴリズム＋データ構造＝プログラム」の中でコンパイラの作成の例に用いられた言語 ＰＬ／０ を拡張した言語である．すなわち，ＰＬ／０＋ は，ＰＬ／０ の持つ諸機能に加えて，配列，入出力，パラメタ付き手続きなどの機能を持ち，整数配列のソーティングのプログラムを自然に記述できる程度の実用性のある言語である．

本章で与える ＰＬ／０＋ 仕様の次のような特徴を持っている．第１は，第２章で与えた代数的仕様記述法に従って，意味領域を抽象データ型として純粋に等式のみを用いて記述したことであり，プログラムの正当性検証などの問題を代数的意味論の枠組の中で取り扱うことができる．第２は，プログラムの意味を入力から出力への関数として考え，それを抽象データ型を用いて記述したことである．すなわち，抽象データ型の代数的仕様記述では，関数をそのままデータとして扱えないので工夫が必要であり，本論文では，関数の表

主論文の要旨

報告番号	※甲第　　号	氏名	北　英彦

現と関数の表現を関数と解釈して引数に適用する関数によって記述を行っている．第３は，意味領域を規定する等式の集合が項書き換え系として，線形性，無あいまい性を持つように記述したことである．意味領域中の等式は左辺から右辺への書き換え規則と見ることができ，さらに，意味領域の仕様は項書き換え系と見ることができる．また，項書き換え系が線形性，無あいまい性の性質を持てば，書き換え結果が一意に定まる．このことに基づいて，項書き換え系を用いて意味領域の直接実現（記号実行）を行うことができ，さらに，その直接実現を用いて仕様記述された言語のインタプリタを自動生成できる．実際，言語処理系生成システム Ｌａｓｓ により，本章の仕様から ＰＬ／０＋ の処理系を生成し，ＰＬ／０＋ のプログラムを実行してみて，この仕様のテストを行い，仕様の妥当性を確認している．

第４章および第５章では，代数的仕様記述法を確立する目的の一つであるプログラム検証について考察する．

第４章では，プログラム検証法としてよく知られているホーア流の公理的検証体系と代数的意味論との関係を明らかにする．そのために，代数的意味論に対する公理的検証体系の健全性について考察する．具体的には，ＷＨＩＬＥ プログラムを記述する簡単な言語 ＷＬ に対して，代数的意味論に関する公理的検証体系の健全性を示す．これにより，代数的意味論と公理的検証体系の間に矛盾のないこと，代数的意味論の与えられたプログラミング言語で書かれたプログラムの検証に公理的検証体系を用いてよいことがわかる．また，すでに広く認められているホーア流の公理的検証体系との無矛盾性は，代数的仕様の正しさを相対的に保証する．

同種の研究としては，表示的意味論と公理的検証体系の関係を明らかにしたものとして，Ｄｏｎａｈｕｅの研究がある．Ｄｏｎａｈｕｅは，Ｐａｓｃａｌ サブセットの Ｓｃｏｔｔ 流の表示的意味論を与え，公理的検証体系の表示的意味論に対する健全性を証明し，両者の間に矛盾がないことを示している．また，ホーア流の公理的検証体系の健全性と完全性の研究については，Ｃｏｏｋ の研究がよく知られている．Ｃｏｏｋ はプログラミング言語の意味論としては，関数的意味論を用いて，公理的検証体系の健全性と完全性の証明を行なっている．これらの研究と異なり，代数的意味論に対する公理的検証体系の健全性の証明を行なったことが本論文の特徴である．

主論文の要旨

報告番号	※甲第　　号	氏名	北　英彦

第５章では，代数的意味論の枠組の中でプログラムの正当性検証を行うための基本的な考え方について述べる．即ち，プログラムの意味を入力から出力への関数と考えたプログラミング言語の代数的仕様記述法，ならびに，プログラムの機能の代数的仕様記述法を与え，それに基づいてプログラムの正当性の定義を与える．言語の代数的意味論を基礎にしているため，検証の（半）自動化に項書き換え系を直接利用できる．

プログラムの検証とは，直観的に言えば，プログラム Ｐ とそれが満たすべき仕様 Ｓ が与えられたとき，Ｐ が確かに Ｓ を満たすことを証明することである．しかし，このままでは具体的に何をしたら良いのか分からない．形式的なプログラム検証のためには，『プログラムが仕様を満たす』ということの妥当でかつ厳密な定式化が必要である．この定式化は次の３項目の定義によってなされる．

（１）プログラムの実行によって達成される機能（プログラムの意味）

（２）プログラム仕様の表す機能（プログラムの仕様の意味）

（３）プログラム仕様に対するプログラムの正当性

（１）は，プログラミング言語の形式的意味論を与えることで定義される．（２）は，（１）と同様に，プログラム仕様の記述言語の形式的意味論を与えることで定められる．（３）は，プログラムの意味する機能がプログラム仕様の意味する機能を満たすことの定義である．これらの定義が定まれば，与えられたプログラムと仕様に対してプログラム検証をするには何をすれば良いのかが明確となる．ここで重要なことは，（３）の妥当な定義のためには，（１）と（２）が同じ枠組の中で定義されなければならないことである．

本章では，プログラム検証を念頭においた場合の（１），（２）の代数的な定義手法，および，（３）の一つの妥当な定式化を提案し，プログラムの代数的な検証法の基礎を明確にする．代数的枠組を用いることの結果として，プログラムの意味ばかりでなく，意図する機能も抽象データ型として仕様記述することとなり，抽象データ型の検証手法，特に，項書き換え系を利用した技法がそのままプログラム検証に応用できる．このことは，本章のプログラム検証の形式化の最も大きな特徴である．

最後に付録として，第３章で代数的仕様記述の説明に用いたプログラミング言語 ＰＬ／０＋ の全仕様を載せる．