プログラム言語BQLとその処理系

池田靖雄[†] 細野千春[††] 辻尚史[†††]

ν 行為はプログラムを数学的に表現したものである．本論文では，ν 行為を計算機で解釈・実行するための言語BQLとその解釈系のアルゴリズムを紹介し，その正当性について述べる．ν 行為は論理式中の自由変数を少なくとも1つ νx なる表現である質変数で置き換えたもので，自由変数が現在の割当，質変数が新しい割当を表すものである．たとえば通常のプログラム言語の算術代入文`x := f(x)`は $\nu x = f(x)$ という ν 行為に対応している．解釈系が行う基本的な解釈は，ν 行為 $A[x, \nu x]$ と自由変数の現在の割当 σ に対し $\sigma \models \exists y A[x, y]$ であるとき新しい割当として $\sigma \models A[x, a]$ なる名前 a を求めるものである．BQLが対象とする ν 行為（解釈可能な ν 行為）は述語論理を基とした型理論の上で記述されたものであり，関係・集合に対応するアブストラクトを値としてとる高階の質変数などもその中に出現する．そしてこれらの ν 行為は，量記号は限量記号に限る，関数記号 $\times, /$ にはその2つの引数ともには質変数が出現しない，などの条件を満たすものである．これらは非常に強い制限ではあるが，通常の手続き型言語で書かれたプログラムの大部分は以上の範囲で表現できる．また，論理型プログラム言語という視点で考えると，他の幾つかの言語と異なり，特殊な形を要求しない，否定記号を完全に扱えるなどの性質がある．

The Language BQL and Its Implementation

YASUO IKEDA,[†] CHIHARU HOSONO[††] and TAKASHI TSUJI[†††]

An algorithm of the interpreter of the language BQL and its strong correctness are presented. The language BQL is introduced for the automatic interpretation of ν-definable acts ,or simply acts, which are mathematical representations of programs. An act is obtained from a formula A by substituting at least one occurrence of free variables $x, y, \ldots$ in A by qualitatives, which are of the form $\nu x, \nu y, \ldots$. The primitive interpretation of an act $A[x, \nu x]$ under a state σ is the following. If $\sigma \models \exists y A[x, y]$ holds then we choose a name a such that $\sigma \models A[x, a]$ as the new value of x. Otherwise the new state is not defined. The system interprets acts which satisfy several conditions, for example, for every quantifier the range of its bound variable must be explicitly represented. The interpreter of this logical language interpret the logical nagetion as it's usual interpretation, different from any other logical languages for example Prolog.

1. はじめに

コンピュータが社会に浸透するにつれ，その動作を表すプログラムに設計者の意図が正しく表されているかということは非常に重要な問題となる．解析的意味論（analytic semantics）[3]では，述語論理を基とした高階の理論上でプログラムを表現し，その性質の解析[4)~6)]，正当性の検証[8)]などを形式的かつ厳密に行うことが可能である．このとき数学的，形式的にプログラムを表現したものを ν-定義可能行為（ν-definable act 以下 ν 行為）と呼ぶ．

ν 行為の解釈には基本解釈と永久解釈とがある．自由変数 x に対し，νx という形は質変数（qualitative）と呼ばれ，自由変数が変化前の値，質変数が変化後の値をそれぞれ表す．これによって値の変化を表すことができる．たとえば ν 行為 $\nu x \geq x + 1$ は変数 x の値が1以上増える変化を表す．新しい記号 ν を導入するのは，理論の記述のために論理式とプログラム（行為）とを本来的に異なる2つの概念ないし数学的対象として区別することが望ましいためであ

† 埼玉短期大学情報処理学科
Institute of information processing, Saitama Junior College
†† 筑波大学電子・情報工学系
Institute of Information Science and Electronics, University of Tsukuba
††† 千葉大学理学部数学・情報数理学科
Department of Mathematics and Informatics, Faculty of Science, Chiba University

る☆．たとえば，変数の新しい値を表すのに $x', y', z', \ldots$ を用いる記法[10] を使うとすれば

$$\{\exists x' \exists y' \cdots A[x', y', \ldots, x, y, \ldots]\}$$
$$A[\nu x, \nu y, \ldots, x, y, \ldots]$$
$$\{A[x', y', \ldots, x, y \ldots]\}$$

が成り立つ．$A[x', y', \ldots, x, y, \ldots]$ を含め一般に $A[u, v, \ldots, x, y, \ldots]$ の形の論理式を $A[\nu x, \nu y, \ldots, x, y, \ldots]$ の質（quality）と呼ぶ．このように ν 行為に出現する各変数の値を変化させるのが基本解釈である．このとき $u, v, \ldots$ にどんな値を与えても質を満たすことができないならば行動不能という．行動不能のときには新しい値は未定義となる．

永久解釈は基本解釈を行動不能となるまで繰り返すものである．たとえば ν 行為 $x < 3 \wedge \nu x = x + 1$ が与えられたとき x の初期値を 1 として基本解釈を繰り返すと，x の値が 3 になったのち行動不能となる．永久解釈ではこの $x = 3$ を解釈結果とする．つまり永久解釈は基本解釈を繰り返し，行動不能となった時点での各変数に対する割当を解釈結果とする．

本研究の目的は検証などのために ν 行為を計算機で解釈・実行することである．もちろんすべての ν 行為を実行することは不可能であるので，どの範囲の ν 行為の解釈・実行が自動的に行えるのかということも研究の課題となる．

ν 行為によるプログラムの表現は仕様の一種としてもとらえることができる．よってその解釈系はプログラムの仕様が与えられたときに，そのプログラムの計算結果を求めるものとなる．こういった仕様をそのまま動かすという視点の研究には，自動プログラミングや論理型プログラム等がある．しかし，従来の研究では単に仕様を与えてプログラムを得るのではなく，手続き（アルゴリズム）をたとえば証明図の形で与えているか，手続きを陽に与えない場合には，たとえば Prolog では（存在）量記号を表現できないなどというように大きな制限があった．本研究はこれらに対し自然な記法で，手続きを与えずに仕様のみを与えたときにどのクラスの表現までが扱えるかを探る試みであるともいえる．

本論文では，そのクラスの一例として，現代解析体系 FA[12] の上で表現された ν 行為の部分集合を計算機で解釈実行するために設計された言語 BQL[13] を提案し，実際にこれが解釈可能（実行可能）であることを示すため，および今後の研究の方向を探るために作成した解釈系[8),9)] について述べる．直観的には BQL は通常の有理数論を基とし，有理数とその特定の形の集合を扱う理論上の ν 行為を表現できる．BQL の解釈系は必ず停止することを示す．

以下 2 章でプログラム言語 BQL の文法の定義を与え，3 章で解釈アルゴリズム，およびその正当性の証明を示し，4 章で使用例を示し，5 章で結論を述べる．

2. ν 行為とプログラム言語 BQL

ν 行為および基本解釈の形式的な定義を述べる．

定義 2.1（ν 行為・基本解釈） 述語論理を基にする理論 T の自由変数 x に対し表現 νx を質変数と呼ぶ．また，T の論理式（以下単に式という）$A[\underline{x}]$（下線は変数の並びを表す）の自由変数の出現を少なくとも 1 つ，質変数で置き換えたもの $A[\underline{\nu x}, \underline{x}]$ を ν 行為と呼ぶ．

M を T のモデルとし，自由変数への（現在の）値の割当（assignment）を σ とし，$\sigma(\underline{x})$ の表す値の列の M 上の名前（name）の列を $\underline{a}$ で表す．σ と ν 行為 $A[\underline{\nu x}, \underline{x}]$ に対して，

(1) $M \models \exists \underline{y} A[\underline{y}, \underline{a}]$ の場合
$A[\underline{\nu x}, \underline{x}]$ は行動可能といい，$\underline{x}$ の（次の）値として $A[\underline{b}, \underline{a}]$ なる名前の列 $\underline{b}$ を非決定的に選ぶ．

(2) $M \not\models \exists \underline{y} A[\underline{y}, \underline{a}]$ の場合
$A[\underline{\nu x}, \underline{x}]$ は行動不能といい，(次の）値は未定義となる．

現代解析体系 FA は古典論理 LK 上のペアノ（Peano）の自然数論の保存的拡張である高階の理論で，実数および複素数の通常の解析学が展開されている．この理論では基礎型 0 は有理数の集合を表し，この型での演算記号 $+, -, \times, /$ および述語記号 $=, <, \leq$, 整数であることを表す Z を持つ．プログラム言語 BQL は FA 上の，論理式，ν 行為のあるクラスを表現する．BQL の構文，および解釈系の説明に必要な概念の定義を述べる．

定義 2.2（型）
0 および $[0, \ldots, 0]$ は型である．
型 0 の要素は有理数を表し，型 $\underbrace{[0, \ldots, 0]}_{n}$ $(n \geq 1)$ の要素は n 個の有理数の集合の直積の部分集合を表す．通常の有理数（の名前）はすべて型 0 の定数記号とし，この集合を Q とする．各型ごとに十分な数の束縛変数と自由変数の集合を置く．自由変数をプログラム変数ともいう．

定義 2.3（質変数） x が型 τ のプログラム変数で

☆ 概念が意味論も含めていったん確立してしまえば，場面によって略記法や金物表現として ν の代わりに単に' を使ったり，最近の制約論理プログラムのように変数名を大文字化する記法を使うことができることはいうまでもない．

あるとき，νx は型 τ の質変数である．

定義 2.4（定値可能項）

(1) 有理定数，型 0 の自由変数は定値可能項である．

(2) s, t がそれぞれ定値可能項であるとき，$s+t$, $s-t$, $s \times t$, s/t はそれぞれ定値可能項である．

定義 2.5（項）

(1) 定値可能項は項である．

(2) 型 0 の質変数 νx は項である．

(3) g, h が項であるとき，$g+h$, $g-h$ は項である．

(4) g が項であり，s が定値可能項であるとき，$s \times g$, $g \times s$, g/s はそれぞれ項である．

定義 2.6（範囲指定式，ν 行為）

(1) g, h が項であるとき，$Z(g)$, $g = h$, $g < h$, $g \leq h$ はそれぞれ ν 行為である．

(2) p が型 $[\underbrace{0,\ldots,0}_{n}]$ のプログラム変数であり，$g_1,\ldots,g_n$ が項であるとき，$p(g_1,\ldots,g_n)$ は ν 行為である．

(3) νp が型 $[\underbrace{0,\ldots,0}_{n}]$ の質変数であり，$g_1,\ldots,g_n$ が項であるとき，$\nu p(g_1,\ldots,g_n)$ は ν 行為である．

(4) A, B がそれぞれ ν 行為であるとき，$\neg A$, $A \supset B$, $A \equiv B$, $A \vee B$, $A \wedge B$ はそれぞれ ν 行為である．

(5) $\underline{x}$ が型 0 の束縛変数の並びであり，$\underline{\nu y}$ が型 0 の質変数の並び，$A[\underline{\nu y}]$ が ν 行為であるとき，$\{\underline{x}\}A[\underline{x}]$ を型 $[0,\ldots,0]$ のアブストラクトと呼ぶ．このとき $A[\underline{x}]$ には質変数が現れないものとする．

(6) $\{\underline{x}\}A[\underline{x}]$ が型 $[0,\ldots,0]$ のアブストラクトであり，$g_1,\ldots,g_n$ が項であるとき，$(\{\underline{x}\}A[\underline{x}])(g_1,\ldots,g_n)$ は ν 行為である．

(7) x が型 0 の束縛変数であるとき，次の 3 つの形の素論理式それぞれ少なくとも 1 つずつと，ν 行為との論理積を x の範囲指定式と呼ぶ．

(a) $Z(x)$ または，$N(x)$

(b) $s < x$ または，$s \leq x$

(c) $x < s$ または，$x \leq s$

ここで，s は x が現れない定値可能項である．

(8) x が型 0 の束縛変数であり，$s_1,\ldots,s_n (1 \leq n)$ が x の現れない定値可能項であるとき，$x = s_1 \vee \cdots \vee x = s_n$ と ν 行為との論理積は x の範囲指定式である．

(9) y が型 0 の自由変数であり，A が x の範囲指定式，$B[y]$ が ν 行為であるとき，$\exists x(A \wedge B[x])$, $\forall x(A \supset B[x])$ は ν 行為である．

以上 (1), (3) で定義された ν 行為および，$\neg Z(t)$, $\neg p(t_1,\ldots,t_n)$ の形の ν 行為を素行為と呼ぶ．

意味的には関数記号，述語記号等の解釈は通常の有理数論に従う．定義 2.6 (6) の $(\{\underline{x}\}A[\underline{x}])(g_1,\ldots,g_n)$ は $A[g_1,\ldots,g_n]$ を意味する．p が型 $[0,\ldots,0]$ の自由変数のとき，p への割当を $\{\underline{x}\}A[\underline{x}]$ とすると，$p(g_1,\ldots,g_n)$ の解釈は $A[g_1,\ldots,g_n]$ の解釈に等しい．

以上の定義で十分なのであるが，プログラムの作成を容易にするために次の定義を加える．

(1) array $[\underbrace{0,\ldots,0}_{n}]$ of 0 は型である．

これはその型の変数の割当を制限するものである．この型を配列型と呼ぶ．この型は $[\underbrace{0,\ldots,0}_{n},0]$ であり，その対象は通常のプログラム言語の配列に対応する．配列型の対象 p は以下で示される一意性を満たすこととする．

$$\forall \underline{x}, u, v(p(x_1,\ldots,x_n,u) \wedge p(x_1,\ldots,x_n,v) \supset u = v)$$

(2) プログラム変数 p の型が array $[\underbrace{0,\ldots,0}_{n}]$ of 0 であり，$t_1,\ldots,t_n$ がそれぞれ定値可能項，$g_1,\ldots,g_n$ がそれぞれ項であるとき，$p(t_1,\ldots,t_n)$ は定値可能項，$p(g_1,\ldots,g_n)$ は項である．

(3) 質変数 νp の型が array $[\underbrace{0,\ldots,0}_{n}]$ of 0 であり，$g_1,\ldots,g_n$ がそれぞれ項であるとき，$\nu p(g_1,\ldots,g_n)$ は項である．

p が型 array $[\underbrace{0,\ldots,0}_{n}]$ of 0 の自由変数であり，p への割当が $\{\underline{x}\}A[\underline{x}]$ であるとき，$p(t_1,\ldots,t_n)$ の解釈は $A[t_1,\ldots,t_n,a]$ である a の解釈に等しい．

以下でアブストラクト $\{x_1,\ldots,x_n\}(\bigvee_{i=1}^{m}\bigwedge_{j=1}^{n} x_j = c_{ij})$ を

$$\{\langle c_{11},\ldots,c_{1n}\rangle,\ldots,\langle c_{m1},\ldots,c_{mn}\rangle\}$$

と略記する．

定義 2.7（プログラム） A が ν 行為であるとき，A および etern A は BQL のプログラムである．

単なる A は A の基本解釈での解釈の指定を表し，etern A は A の永久解釈での解釈を示す．

3. 解釈アルゴリズム

解釈系の説明をするにあたり，たとえば ν 行為のある部分を解釈して得られる結果などと現在の各変数

への割当との混同を避けるため，解釈系の計算結果を新しい割当と呼ぶ．ν 行為に出現する質変数の数は有限であり，ν 行為に質変数として現れない変数の値は変えないものとする．解釈は値の変化する有限個の変数それぞれの変化後の値を求めればよい．よって新しい割当は，行動不能を示す $\perp$ または $\sigma(x)=a$ なる変数 x と値 a の組の有限集合で表す．

この定義より，たとえば空集合 $\emptyset$ は現在の割当と等しい新しい割当を示す．

また，$\underline{x}$，$\underline{x'}$ がそれぞれ自由変数の並びであり，$y_1,\ldots,y_n$ が束縛変数であるとき，ν 行為 $A[\underline{x},\underline{x'}]$ 中の項 $t[\underline{x},\underline{x'}]$ に対応する，ν 行為 $Q_1y_1\cdots Q_ny_nA[\underline{x},y_1,\ldots,y_n]$ 中の表現 $t[\underline{x},\underline{y}]$ も項と見なす．(ここで Q_i は量記号を表す．)

解釈系は ν 行為に対し使用者の指示にしたがって基本解釈か，永久解釈かのどちらかを実行する．永久解釈は基本解釈を行動不能になるまで繰り返すものであるので，以下では基本解釈の解釈アルゴリズムを述べる．

基本解釈では ν 行為 $A[\nu\underline{x},\underline{x}]$ が行動可能であるとき次の条件を満たす新しい割当 $\{\underline{\langle x,a\rangle}\}$ を1つ求める．

$$\sigma\models A[\underline{a},\underline{x}],$$

ここで σ は現在の割当である．行動不能であるとき解釈結果は $\perp$ である．

解釈アルゴリズムは次の2つの手続きからなる．

(1) 行為を標準形に変換する手続き（簡単化を行う手続き）

(2) 標準形の ν 行為の解釈手続き

3.1 ν 行為の標準形への変換アルゴリズム

この手続きは，よく知られている同値変形によって，ν 行為の形を簡単化し，この後の解釈アルゴリズムを簡潔にし，また解釈をより早くするためのものである．

定義 3.1 (標準形)

(1) $a_1,\ldots,a_n,a_{n+1}$ が定数記号と束縛変数だけからなる項であり，$\nu x_1,\ldots,\nu x_n$ が相異なる型0の質変数であるとき $a_1\times\nu x_1+\cdots+a_n\times\nu x_n+a_{n+1}$ という形の項 $(0\le n)$ は標準形であるという．このとき a_{n+1} を定数部と呼ぶ．

(2) t が定数部を含まない標準形の項，c が束縛変数と定数記号からなる項であるとき，素行為 $t'=c$，$t'<c$ はそれぞれ標準形である．

(3) t が標準形の項であるとき $Z(t),\neg Z(t)$ はそれぞれ標準形である．以上(2)および(3)で定義される標準形は少なくとも1つの束縛変数か，質変数の出現を含むものとする．

(4) $t_1,\ldots,t_n$ が標準形の項，質変数 νp の型が $\underbrace{[0,\ldots,0]}_{n}$ であるとき，$\nu p(t_1,\ldots,t_n),\neg\nu p(t_1,\ldots,t_n)$ はそれぞれ標準形である．

(5) ν 行為 $\exists xA[x]$，$\forall xA[x]$ は $A[x]$ が標準形であるときそれぞれ標準形である．

(6) $A_1,\ldots,A_n$ が標準形の ν 行為であるとき，$(\bigvee A_1,\ldots,A_n)$ は標準形の ν 行為である．ただしこのとき $A_1,\ldots,A_n$ はこの形はしていないものとする．

(7) $A_1,\ldots,A_n$ が標準形の ν 行為であるとき，$(\bigwedge A_1,\ldots,A_n)$ は標準形の ν 行為である．ただしこのとき $A_1,\ldots,A_n$ はこの形はしていないものとする．

以上で定義された形に加えて，単なる $\perp,\top$ も標準形の ν 行為とする．ただし，たとえば $(\wedge\perp,A)$ は標準形でない．

以下の補題，定理は明らかである．

補題 3.1 任意の項 $t[\nu\underline{x}]$ に対し，

$$\forall\underline{x}t[\underline{x}]=s[\underline{x}]$$

となるような標準形の項 $s[\nu\underline{x}]$ が存在する．

定理 3.1 任意の ν 行為 $A[\underline{\nu x}]$ に対し，

$$\forall\underline{x}(A[\underline{x}]\equiv B[\underline{x}])$$

となるような標準形の ν 行為 $B[\underline{\nu x}]$ が存在する．

任意の ν 行為に対して定理 3.1 を満たす標準形の ν 行為を求める手続きはよく知られているのでここでは述べない．

3.2 標準形の ν 行為の解釈アルゴリズム

以下で（新しい）割当を代入するとは，表現 $E[\underline{\nu x}]$ と新しい割当 $\{\underline{\langle x,a\rangle}\}$ に対して，

- E が ν 行為の場合，表現 $E[\underline{a}]$ と同値な，
- E が項の場合，表現 $E[\underline{a}]$ と等しい

標準形をそれぞれ得ることを指す．

定義 3.2 解釈アルゴリズム I が ν 行為 $A[\underline{x},\nu\underline{x}]$ を解釈するとは，$\sigma\not\models\exists\underline{y}A[\underline{x},\underline{y}]$ のとき

$$I(A)=\perp$$

となり，$\sigma\models\exists\underline{y}A[\underline{x},\underline{y}]$ のとき

$$I(A)=\{\underline{\langle x,a\rangle}\}\text{ かつ }\sigma\models A[\underline{x},\underline{a}]$$

であることをいう．ここで σ は現在の割当を表す．

以下解釈手続きを次のように ν 行為の形が特殊な場合からより一般的な場合へと順に説明していく．

(1) 束縛変数が現れない場合

(a) 高階の変数が出現しない場合

(i) $Z(\nu x_1)\wedge\cdots\wedge Z(\nu x_n)\wedge Z(g_1)\wedge\cdots\wedge Z(g_l)\wedge B$ という形の ν 行為で B は等式と不等式の連言，$g_1,\ldots,g_n$ および B には

$\nu x_1, \ldots, \nu x_n$ 以外の質変数は出現しない場合.(条件 1)

(ii) 上の条件に加えて $\nu x_1, \ldots, \nu x_n$ 以外の質変数も現れる場合.(条件 2)

(iii) さらに $\neg Z(f_i)$ という形の ν 行為の連言も加わる場合.(条件 3)

(b) 高階の質変数が現れる場合

(2) 束縛変数が現れる場合

以下,アルゴリズムの説明の中で ν 行為 A のある部分や A に現れる質変数を選ぶ部分があるがこれは選び方を決めておき A の形によって一意に決まるものとする.

3.2.1 束縛変数の現れない場合

3.2.1.1 高階の変数の現れない場合

A を条件 1 を満たす ν 行為とする.A の解釈結果 $C(A)$ を以下のように定義する.

定義 3.3 (手続き C)

- 質変数が現れない場合.このとき A の真偽を定めることができる.偽であれば $C(A) = \bot$ とし,真のとき $C(A) = \emptyset$ とする.
- $k+1$ 個ある場合
  - 等式がある場合
    (1) 等式を 1 つ選び,その中に現れる質変数を 1 つ選んでこれについて解く.この結果を $\nu x = t$ とする.
    (2) $C(A_{\nu x}[t]) = \bot$ のとき $C(A) = \bot$, $C(A_{\nu x}[t]) = \sigma$ のとき $C(A) = \sigma \cup \{\langle x, t'\rangle\}$ とする.t' は項 t に割当 σ を代入したものである.
  - 等式がない場合
    (1) 不等式中に現れるすべての有理数の分母の最小公倍数を両辺に掛け,係数をすべて整数にする.
    (2) 質変数 νx を 1 つ選び,不等式および $Z(t)$ 中の νx の係数の分子の最小公倍数 N を求める.
    (3) すべての νx の係数が N になるように適当な数を不等式の両辺および $Z(t)$ 中の係数の分母分子に掛ける.この結果を A' とする.
    (4) 不等式を $N \times \nu x$ について解く.この結果を $l_1 < N \times \nu x, \ldots, l_o < N \times \nu x, N \times \nu x < h_1, \ldots, N \times \nu x < h_p$ とする.
    (5) Z の式すべての質変数の係数の分母と定数の分母および N との最小公倍数を M とする.
    (a) $o = 0$ または $p = 0$ の ($l_j < N \times \nu x$, $N \times \nu x < h_k$ のどちらかがない) 場合
    νx の現れない式から他の質変数の新しい割当を求める.これが行動不能であれば A の解釈も行動不能とする.
    新しい割当 σ が得られればこれを l_j, h_k 中の質変数に代入し,$o = 0$ のとき h_k の最小値より小さい M の倍数,$p = 0$ のとき l_j の最大値より大きい M の倍数を νx の新しい割当とする.
    (b) 両方ともある場合
    $l_j (1 \leq j \leq o)$ を順に選び d の値を 1 から M まで順に増やして,$A'_{N\nu x}[l_j + d] \wedge Z((l_j + d)/M)$ の解釈を行う.この結果がすべて行動不能 $\bot$ であれば $C(A) = \bot$.新しい割当 σ が得られた場合,$(l_j + d)/N$ に新しい割当を代入し,その結果の a と x の組を σ に加えたもの $\{\langle x, a\rangle\} \cup \sigma$ が A の解釈結果である.

補題 3.2 解釈アルゴリズム C は条件 1 を満たす ν 行為を解釈する.

証明は質変数の数の上の帰納法を用いる.(この証明の基本的な部分は Cooper の決定手続き[2]による.)

与えられた ν 行為 $A[\nu x_1, \ldots, \nu x_n]$ に対し,選んだ質変数を νx_i とする.このアルゴリズムでは最後の代入以外は同値変形であり,代入した結果に対して,等式 $\nu x_i = t$ がある場合は

$$\forall \underline{x}(\exists y A[x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n] \equiv A[x_1, \ldots, x_{i-1}, t, x_{i+1}, \ldots, x_n]),$$

等式がない場合は

$$\forall \underline{x} \left(\exists y A[x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n] \equiv \bigvee_{j=1}^{o} \bigvee_{d=1}^{M} \left(A\left[x_1, \ldots, x_{i-1}, \frac{l_j + d}{N}, x_{i+1}, \ldots, x_n\right] \wedge Z\left(\frac{l_j + d}{M}\right)\right)\right)$$

がそれぞれ成り立っている. □

条件 2 を満たす ν 行為 A の解釈結果 $CN(A)$ を以下のように定義する.

定義 3.4 (**手続き CN**) A 中の Z の式中に現れない ($\nu x_1, \ldots, \nu x_n$ 以外の) 質変数の数の上の帰納法.

(1) 0 個の場合
$CN(A) = C(A)$.

(2) $k+1$ 個の場合. $\nu x_1, \ldots, \nu x_n$ 以外の質変数 νx_i を 1 つ選ぶ.

(a) A 中に νx_i が出現する等式があるときこれを解いた結果を $\nu x_i = s$ とする.
$CN(A_{\nu x_i}[s]) = \perp$ の場合, $CN(A) = \perp$.
$CN(A_{\nu x_i}[s]) = \sigma$ の場合, 項 s に σ を代入して得られた項を s' とする.
$CN(A) = (\sigma \cup \{\langle x_i, s' \rangle\})$.

(b) 等式がない場合
νx_i が現れる不等式を νx_i に関して解く. この結果を $l_1 < \nu x_i, \ldots, l_o < \nu x_i$, $\nu x_i < h_1, \ldots, \nu x_i < h_p$ とする. A 中の νx_i が現れない素行為の連言 (Z の式を含む) を B とする.

$$C \equiv \bigwedge_{j=1}^{o} \bigwedge_{k=1}^{p} l_j < h_k \wedge B$$

の解釈を行う.
$CN(C) = \perp$ であれば $CN(A) = \perp$.
$CN(C) = \sigma$ の場合, 各 l_j, h_k 中の質変数にこの新しい割当を代入し, この結果を l'_j, h'_k とする. $CN(A) = \sigma \cup \{\langle x_i, (max\{l'_j\} + min\{h'_k\})/2 \rangle\}$

定理 3.2 CN は条件 2 を満たす ν 行為を解釈する.

証明は Z の式中に現れない ($\nu x_1, \ldots, \nu x_n$ 以外の) 質変数の数の上の帰納法によれば容易.

定義 3.5 (**手続き P**) A 中の $Z(t_i)$, $\neg Z(s_j)$ の出現に対し, 新しい質変数 νx_{t_i}, νx_{s_j} を作り, $Z(t_i)$, $\neg Z(s_j)$ をそれぞれ $Z(\nu x_{t_i}) \wedge \nu x_{t_i} = t_i$, $Z(\nu x_{s_j}) \wedge \nu x_{s_j} < s_j \wedge s_j < \nu x_{s_j} + 1$ を標準形にしたもので置き換えた ν 行為を B とする. $CN(B) = \perp$ であれば A の解釈結果も $\perp$. $CN(B) = \sigma$ の場合, σ から今作った新しい質変数への割当を除く.

系 3.1 手続き P は条件 3 を満たす ν 行為を解釈する.

証明

$\neg Z(t) \equiv \exists x(Z(x) \wedge x < t < x+1)$ と上の定理 3.2 より明らか.

3.2.1.2 高階の質変数の現れる場合

定義 3.6 (**条件集合, 条件式**) 高階の質変数からなる素行為の集合

$$\left\{ \begin{array}{c} \nu p(t_{11}, \ldots, t_{1n}), \ldots, \nu p(t_{m1}, \ldots, t_{mn}), \\ \neg \nu p(s_{11}, \ldots, s_{1n}), \ldots, \neg \nu p(s_{l1}, \ldots, s_{ln}) \end{array} \right\}$$

を高階の質変数 νp の条件集合と呼び, 条件集合に対して次のような ν 行為を質変数 νp の条件式と呼ぶ.

$$\bigwedge_{1 \le i \le m} \bigwedge_{1 \le j \le l} \left(\bigvee_{1 \le k \le n} t_{ik} \neq s_{jk} \right)$$

定理 3.3 $S[\nu p, \nu \underline{x}]$ を高階の質変数 νp に対する条件集合とし, $A[\nu \underline{x}]$ を νp の S に対する条件式とする. このとき,

$$\models \exists \underline{y} A[\underline{y}] \text{ iff } \models \exists V, \underline{y} \left(\bigwedge S[V, \underline{y}] \right)$$

が成り立ち, S に対する条件式が行動可能であるとき, その任意の新しい割当から $\bigwedge S$ の新しい割当を求めることができる.

証明

条件式 A が行動可能 ($\models \exists y A[y]$) の場合,

条件式中の項 $t_{11}, \ldots, t_{mn}, s_{11}, \ldots, s_{ln}$ に A の解釈によって得られる新しい割当 σ を代入したものを $t'_{11} \ldots, t'_{mn}, s'_{11}, \ldots, s'_{ln}$ とする. このとき Q を

$$\{\underline{x}\} \left(\bigvee_{i=1}^{m} \left(\bigwedge_{j=1}^{n} x_j = t'_{ij} \right) \right)$$

とすると,

$$\models \bigwedge_{i=1}^{m} Q\left(t'_{i1}, \ldots, t'_{in}\right) \wedge \bigwedge_{j=1}^{l} \neg \left(Q\left(s'_{j1}, \ldots, s'_{jn}\right) \right)$$

であり,

$$\bigwedge_{i=1}^{m} Q\left(t'_{i1}, \ldots, t'_{in}\right) \wedge \bigwedge_{j=1}^{l} \neg \left(Q\left(s'_{j1}, \ldots, s'_{jn}\right) \right)$$

$$\Rightarrow \exists \underline{y} \left(\bigwedge_{i=1}^{m} Q\left(t_{i1}[\underline{y}], \ldots, t_{in}[\underline{y}]\right) \wedge \bigwedge_{j=1}^{l} \neg \left(Q\left(s_{j1}[\underline{y}], \ldots, s_{jn}[\underline{y}]\right) \right) \right)$$

$$\Rightarrow \exists V, \underline{y} \left(\bigwedge_{i=1}^{m} V\left(t_{i1}[\underline{y}], \ldots, t_{in}[\underline{y}]\right) \wedge \bigwedge_{j=1}^{l} \neg \left(V\left(s_{j1}[\underline{y}], \ldots, s_{jn}[\underline{y}]\right) \right) \right)$$

$$\Leftrightarrow \exists V, \underline{y} \bigwedge S[V, y]$$

と導けるので，新しい割当として，$\sigma \cup \{\langle p, Q\rangle\}$ をとればよい．

A が行動不能の場合．任意の新しい割当に対しある $i, j (1 \leq i \leq m,\ 1 \leq j \leq l)$ が存在して $\bigwedge_{1 \leq k \leq n} t'_{ik} = s'_{jk}$ となる．このとき $\bigwedge S$ より $\nu p(t'_{i1}, \ldots, t'_{in}) \wedge \neg \nu p(s'_{j1}, \ldots, s'_{jn})$ が導かれるがこれを満たす新しい割当は存在しない． □

この証明で条件式の解釈結果 σ から高階の変数への割当 $\{\langle p, Q\rangle\}$ を求める手続きを $H_p(\sigma)$ とする．

3.2.2 束縛変数の現れる場合

以下では新しい割当の集合，標準形の素行為の集合，標準形の ν 行為のリストの集合をそれぞれ Σ, ACT, $ACTL$ で表す．

T を $ACTL \times \mathcal{P}(ACT)$ から $\Sigma \cup \{\bot\}$ への関数とする．$T(AL, As)$ を以下のように定義する

- $AL = NIL$ の場合
  - 素行為の集合 As 中に高階の質変数からなる素行為がある場合．As 中の高階の質変数からなる素行為を各質変数ごとに分ける．それらの集合を As から取り除いたものを As' とする．また各高階の質変数ごとの集合を各々の条件集合と見なして条件式を作る．各条件式の標準形のリストを AL' とする．$T(AL', As') = \bot$ の場合，
    $$T(NIL, As) = \bot.$$
    それ以外のとき
    $$T(NIL, As) = T(AL', As') \cup \bigcup_p H_p(T(AL', As')).$$
  - As 中に高階の質変数からなる素行為が無い場合．
    $$T(NIL, As) = P(As).$$
- AL が NIL でない場合．
  標準形の AL の先頭の要素 A の構造に関する帰納法により定義する．リスト AL の A の後ろのリストを AL' で表す．
  - A が素行為のとき
    $$T([A|AL'], As) = T(AL', As \cup \{A\})$$
  - $A \equiv \bigwedge A_1, \ldots, A_n$ のとき
    $$T\left(\left[\left(\bigwedge A_1, \ldots, A_n\right)|AL'\right], As\right) = T([A_1, \ldots, A_n|AL'], As)$$
  - $A \equiv \bigvee A_1, \ldots, A_n$ のとき
    (1) $T([A_i|AL'], As) = \bot\ (0 \leq i \leq k-1)$ で $T([A_k|AL'], As) \neq \bot$ ならば
    $$T\left(\left[\left(\bigvee A_1, \ldots, A_n\right)|AL'\right], As\right) = T([A_k|AL'], As).$$
    (2) $T([A_i|AL'], As) = \bot\ (0 \leq i \leq n)$ ならば
    $$T\left(\left[\left(\bigvee A_1, \ldots, A_n\right)|AL'\right], As\right) = \bot.$$
  - $A \equiv \exists x B[x]$ のとき範囲指定式から x のとりうる値を求め，それらを $a_1, \ldots, a_n$ とする．($i < j$ のとき $a_i < a_j$)
    (1) $T([B[a_i]|AL'], As) = \bot\ (0 \leq i \leq k-1)$ かつ $T([B[a_k]|AL'], As) \neq \bot$ ならば
    $$T([\exists x B[x]|AL'], As) = T([B[a_k]|AL'], As).$$
    (2) $T([B[a_i]|AL'], As) = \bot\ (0 \leq i \leq n)$ のとき
    $$T([\exists x B[x]|AL'], As) = \bot.$$
  - $A \equiv \forall x B[x]$ のとき範囲指定式から x のとりうる値を求め，それらを $a_1, \ldots, a_n$ とする．($i < j$ のとき $a_i < a_j$)
    $$T([\forall x B[x]|AL'], As) = T([B[a_1], \ldots, B[a_n]|AL'], As)$$

定理 3.4 $A[\nu \underline{x}]$ を標準形の ν 行為とする．
$\models \exists \underline{y} A[\underline{y}]$ のとき $T([A], \emptyset) = \sigma \neq \bot$ となり，σ を A に代入した結果を A' とすると $\models A'$ となる．
また，$\models \neg \exists \underline{y} A[\underline{y}]$ であれば $T([A], \emptyset) = \bot$ となる．

証明

$T(AL, As)$ の計算結果が AL 中の ν 行為と As 中の ν 行為との論理積の解釈結果であることを示せば十分．リスト AL の長さに関する帰納法．AL が NIL で高階の質変数を含まないときは系 3.1 より明らか．高階の質変数があるときには，定理 3.3 より NIL 以外の場合で AL, As に高階の質変数を含んでいない場合を示せばばよい．

AL が NIL でない場合は AL の先頭の ν 行為 A の構造に関する帰納法により明らか． □

4. 実 行 例

この節では Macintosh 上の Allegro Common LISP で制作され，SPARC Station 上の GCL に移植された解釈系の実行例について述べる．

例 1　バスの 20 円券と 30 円券がそれぞれ 5 枚ずつあるとする．学園並木という停留所から荒川沖駅ま

でバス代が230円かかる．学園並木から荒川沖駅までに払うべき20円券の枚数 x と30円券の枚数 y をそれぞれ求める．与えられた条件を ν 行為で表現する

$$\nu x \leq 5 \wedge \nu y \leq 5 \wedge N(\nu x) \wedge N(\nu y)$$
$$20 \times \nu x + 30 \times \nu y = 230$$

これを解釈すると解 $\nu x = 4$, $\nu y = 5$ が得られる．

例2　次のようなパズルを解く．このパズルは参考文献1)より引用したものである．

この前の学期にジャスミン，カーク，ラホーマ，ミッチの4人は全員，英語，数学，歴史の科目を選択しました．そして成績はどれもAかBかCでDだった人はいません．以下の一部書き込まれた表をあなたは完成できますか．

(1) ミッチの成績にはCが1つありました．
(2) ラホーマの数学の成績は，カークの英語の成績と同じです．
(3) 歴史の成績がBだった人はいません．
(4) 3科目すべてAだった人が1人います．
(5) 英語でAだったのは2人，数学でAだったのも2人，歴史でAだったのも2人です．
(6) ジャスミンとカークの歴史の成績は同じです．
(7) 「1つの科目でAで，もう1つでB，残りでCだった人」が少なくとも1人います．

	英語	数学	歴史
ジャスミン		A	
カーク		A	
ラホーマ			C
ミッチ	B		

BQLでこの問題を解くために，ジャスミン，カーク，ラホーマ，ミッチをそれぞれ0, 1, 2, 3で，英語，数学，歴史をそれぞれ0, 1, 2で，成績のA, B, Cをそれぞれ0, 1, 2で，表すこととする．型 $array[0,0]$ of 0 の変数 s を用い，$s(x,y) = z$ で人 x の科目 y の成績が z であることを表すことにする．$person(x)$, $subject(x)$ をそれぞれ次のようにマクロ定義する．

$$person(x) \equiv Z(x) \wedge 0 \leq x \leq 3.$$
$$subject(x) \equiv Z(x) \wedge 0 \leq x \leq 2.$$

$\nu s(x,y)$ が成績を表すので

$$\forall x(person(x) \supset \forall y(subject(y) \supset Z(\nu s(x,y)) \wedge 0 \leq \nu s(x,y) \wedge \nu s(x,y) < 3)).$$

表の条件は次のようになる．

$$\nu s(0,1) = 0 \wedge \nu s(1,1) = 0 \wedge \nu s(2,2) = 2 \wedge \nu s(3,0) = 1.$$

そして，上の条件文はそれぞれ以下のように表せる．

$$\exists x(subject(x) \wedge \nu s(3,x) = 2 \wedge \forall y(subject(y) \wedge \nu s(3,y) = 2 \supset x = y)). \quad (1)$$
$$\nu s(2,1) = \nu s(1,0). \quad (2)$$
$$\forall x(person(x) \supset \nu s(x,2) \neq 1). \quad (3)$$
$$\exists x(person(x) \wedge \forall y(subject(y) \supset \nu s(x,y) = 0)). \quad (4)$$
$$\forall w(subject(w) \supset \exists x1(person(x1) \wedge \exists x2(person(x2) \wedge \exists x3(person(x3) \wedge \exists x4(person(x4) \wedge x1 \neq x2 \wedge x1 \neq x3 \wedge x1 \neq x4 \wedge x2 \neq x3 \wedge x2 \neq x4 \wedge x3 \neq x4 \wedge \nu s(x1,w) = 0 \wedge \nu s(x2,w) = 0 \wedge \nu s(x3,w) \neq 0 \wedge \nu s(x4,w) \neq 0))))). \quad (5)$$
$$\nu s(0,2) = \nu s(1,2). \quad (6)$$
$$\exists x(person(x) \wedge \exists s0(subject(s0) \wedge \exists s1(subject(s1) \wedge \exists s2(subject(s2) \wedge \nu s(x,s0) = 0 \wedge \nu s(x,s1) = 1 \wedge \nu s(x,s2) = 2)))). \quad (7)$$

表を完成するためには，以上の ν 行為の連言を解釈すればよい．

この結果 s への割当として

$$\left\{ \begin{array}{llll} \langle 0,1,0\rangle, & \langle 1,1,0\rangle, & \langle 2,2,2\rangle, & \langle 3,0,1\rangle, \\ \langle 1,0,1\rangle, & \langle 1,2,0\rangle, & \langle 2,0,0\rangle, & \langle 2,1,1\rangle, \\ \langle 0,0,0\rangle, & \langle 0,2,0\rangle, & \langle 3,2,2\rangle, & \langle 3,1,1\rangle \end{array} \right\}$$

を得る．

この ν 行為の表現は先に述べたパズルの条件を素直に表現したものといえよう．この中にいくつか，存在量記号 ヨ で束縛された形で表現されるものがあるが，このような形の表現はPrologなどではそのままの形では表せない．何らかの仕掛けが必要となってプログラムが複雑になったり，人間がある程度の前処理をした形となる．BQLでは否定記号，あるいは存在量記号を本来の意味で使用できる．

5. おわりに

ν 行為の解釈のための言語BQLとその解釈系のアルゴリズムを述べた．

BQLの ν 行為は以下の制限を満たすものである．

(1) 質変数の型は0または $[0,\ldots,0]$ に限る．
(2) 項 $t \times u$ 中の t, u ともには質変数が出現しない．
(3) 項 t/u は u 中に質変数を含まない．
(4) 量記号はそれが束縛する束縛変数の値のとりうる範囲を範囲指定式で明示することにする．すなわち量記号が現れる部分は，A が x に対する範囲指定式，B が任意の解釈可能形の ν 行

為であるとき，$\forall x(A \supset B)$ および $\exists x(A \wedge B)$ に限る．

以上の制限について，BQL は計算機で解釈できる ν 行為の最も基本的なクラスであると考えている．これは先にも述べたように本研究のもう 1 つの目的が検証などの目的で ν 行為の解釈を計算機で行うことであり，次に示すとおり，通常のプログラム言語による表現を BQL である程度表すことができる．

Pascal 様の言語で書かれたプログラム P に対して，永久解釈を行ったとき最終的に得られる値の割当が P が停止したときの P 中の各変数の値と同値となる BQL のプログラムは，たとえば以下のように定義できる[13]．

型 0 の変数 l を導入し，これをラベルとして使用する．(P, l_a, l_b) はラベル l が定数 l_a のとき P と（上で述べた意味での）同様の動作をし，終了後ラベルの値を定数 l_b と $(l_a < l_b)$ する ν 行為を表す．

$$(\texttt{x := } E, l_a, l_b) \equiv$$
$$l = l_a \wedge \nu x = E \wedge \nu l = l_b.$$
$$(\texttt{P; Q}, l_a, l_b) \equiv$$
$$(\texttt{P}, l_a, (l_a + l_b)/2) \vee (\texttt{Q}, (l_a + l_b)/2, l_b).$$
$$(\texttt{if } A \texttt{ then P else Q}, l_a, l_b) \equiv$$
$$(A \supset (\texttt{P}, l_a, l_b)) \wedge (\neg A \supset (\texttt{Q}, l_a, l_b)).$$
$$(\texttt{while } A \texttt{ do P}, l_a, l_b) \equiv$$
$$l = l_a \wedge \left(A \supset \nu l = \frac{l_a + l_b}{3}\right)$$
$$\wedge (\neg A \supset \nu l = l_b)$$
$$\vee \left(\texttt{P}, \frac{l_a + l_b}{3}, \frac{2 \times (l_a + l_b)}{3}\right)$$
$$\vee l = \frac{2 \times (l_a + l_b)}{3} \wedge \nu l = l_a.$$

以上より，BQL を用いて if 文や，while 文などからなる Pascal 様言語で書かれたプログラムに対応する ν 行為の解釈を行うことができる．

解釈系は SPARC Station の GNU Common Lisp 上で動いている．前節の例を実行するために他のジョブが動いていない状態での実時間それぞれ約 0.1 秒，10 分を要した（コンパイルは行っていない）．これは通常の論理型言語より一般的な形のものを解釈しているのでやむを得ないと考えられる．また，高速化の手段としては範囲指定式の表現を限定する（特に出現位置について），内部表現を見直すなどが考えられる．しかし，研究の方向としてそのような高速化よりも解釈を行える範囲を広げる方に重きを置きたいと考えた．特に量記号に対する制限をなくし，有限の範囲ではなく有理数全体を動く束縛変数を扱えるよう検討していきたい．そして，これが解決されたシステムが今後の核となるシステムと思われるので，それが完成した後，高速化を考えたい．

謝辞

本研究を進めるにあたり，適切なご指導，ご助言を下さった，筑波大学教授五十嵐滋先生，筑波大学講師水谷哲也先生にお礼申し上げます．また，実際のプログラミングに助力してくれた，工業技術院機械技術研究所の富田康治氏に感謝します．研究の議論に参加し，多大な意見を下さった高鉄軍氏に感謝します．

参考文献

1) Bantam Doubleday Dell Magazines: *Logic Puzzles*, Bantam Doubleday Dell Magazines (1993). 小野田博一（編訳）：論理パズル 101，ブルーバックス，講談社 (1993).
2) Cooper, D.C.: Theorem Proving in Arithmetic without Multiplicaton, *Machine Intelligence*, Vol.7, pp.91–99 (1972).
3) Igarashi, S.: The ν-Conversion and an Analytic Semantics, *Inf. Proc. 83*, Mason, R.E.A. (ed.), Elsevier Science Publishers (North-Holland), IFIP, pp.657–668 (1983).
4) Igarashi, S., Mizutani, T. and Tsuji, T.: An Analytical Semantics of Parallel Program Processes Represented by ν-conversion, *Tensor, N.S.*, Vol.45, pp.222–228 (1987).
5) Igarashi, S., Mizutani, T., Tsuji, T. and Hosono, C.: On Locomorphism in Analytical Equivalence Theory, *Logic, Language and Computation: Festschrift in Honor of Satoru Takasu, Lecture Notes in Computer Science*, Vol.792, pp.173–187 (1994).
6) Igarashi, S., Mizutani, T. and Tsuji, T.: Specifications of Parallel Program Processes in Analytical Semantics, *Tensor, N.S.*, Vol.45, pp.240–244 (1987).
7) 五十嵐滋：計算機と論理，数理科学，No.227, pp.52–59 (1982).
8) 池田靖雄，辻　尚史：ν-インタプリタの設計・試作，日本ソフトウェア科学会第 5 回大会論文集，pp.353–356 (1988).
9) 池田靖雄：高階論理型プログラム言語 NU の処理系に関する研究，筑波大学大学院工学研究科修士論文 (1989).
10) Manna, Z., 五十嵐滋（訳）：プログラムの理論，p.236, 日本コンピュータ協会 (1975).
11) 水谷哲也，細野千春，五十嵐滋：ν-定義可能行為によるプログラムの検証，コンピュータソフトウェア，Vol.2, No.3, pp.47–56 (1985).
12) Takeuti, G.: *Two Applications of Logic to Mathematics*, Iwanami Shoten, Princeton University Press (1987).

13) Tsuji, T.: The Language NU, to appear.

(平成 7 年 4 月 12 日受付)
(平成 8 年 2 月 7 日採録)

細野 千春（正会員）

1948 年生．1973 年京都大学理学部卒業．1976 年同大学大学院退学．1976 年長崎大学医学部原爆被災学術資料センター助手．1978 年筑波大学電子情報工学系助手．1989 年同講師．1993 年同助教授．理学博士．プログラム理論，定理自動証明に興味を持つ．日本ソフトウェア科学会，日本数学会会員．

池田 靖雄（正会員）

1963 年生．1992 年筑波大学大学院博士課程工学研究科満期退学．工学博士．同年より埼玉短期大学情報処理学科講師．論理プログラム，プログラム基礎論等に興味を持つ．ソフトウェア科学会会員

辻 尚史（正会員）

1945 年生．東京大学大学院．工学博士．1973 年東京工業大学助手（理学部）．1977 年筑波大学助教授（電子・情報工学系）．1995 年千葉大学教授（理学部）．プログラム基礎論．「FORTRAN の実際」（共著，サイエンス社）．日本ソフトウェア科学会，日本数学会会員．