モデル記述言語CASTにおける*defList*の定義

Definition of *defList* in Model Description Language, CAST

旭　　貴朗
（Takao ASAHI）

『経 営 論 集』84 号（2014年11月）抜 刷

モデル記述言語 CAST における *defList* の定義

Definition of *defList* in Model Description Language, CAST

旭　　貴朗

1　はじめに

本稿では，モデル記述言語 CAST におけるリスト作成関数 defList の新しい定義を提案する．先行研究とは異なる新しい提案をする理由は，述語の真偽を判定するための内部処理動作が実行に関与し，意図した動作と異なる動きをするからである．関数 defList は多数要素をもつシステムのモデル記述に不可欠なものであるため，これを数理的に定式化し正しく利用できるようになる必要がある．また高度な利用をするために内部処理動作について確認しておく必要がある．

筆者らはモデル理論アプローチによるシステム開発を提案している（高原ほか 2007）．モデル理論アプローチではモデル記述言語 CAST（旭ほか 2008）を用いてシステムモデルを作成し，開発実行環境 MTA-SDK あるいは Simcast でコンパイルし実行する（旭 2009）．したがって関数 defList を正しく利用するためには，集合や論理とは異なる，内部処理動作について理解しておくことが必要である（第 2 節）．第 3 節では，関数の働きをする述語について述べ，第 4 節で defList の新しい定義を提案する．限定的ではあるが数理的定式化を付録に記載する．

モデル理論アプローチの現在の実行環境では，言語 CAST で書かれたすべての式を言語 extProlog のホーン節に変換することをコンパイルと呼んでいる（Takahara & Liu 2006）．すなわち extProlog という中間言語に変換してから各種のモデルを実行する．本来なら中間言語の処理動作の説明が必要であろうが，本稿では extProlog そのものについて知る必要はないので，言語 CAST のレベルで説明をおこなう．

2　複合述語の真偽判定動作

言語 CAST は論理式を使ってシステムモデルを記述する言語である．論理式は規則に従った述語の列からなる．ここでは，まず CAST の言語処理プログラムが複合述語をどのように真偽判定するかについて確認し，述語を並べる順序が重要であることに注意を促したい．

例えば，偶数を定義する述語 p(x)を言語 CAST で記述すると p(x) <-> x%2 = 0; と

なる. ただし記号 <-> は必要十分条件を表わすが, 右辺で左辺を定義するという意味でもある.また記号%は除算による余りを表わす. したがって, p(x)は x を 2 で割った余りが 0 のとき（すなわち x が偶数のとき）真となる述語である.

以下では,「与えられた x の値に対して, 述語の真偽を確認する処理動作」を**真偽判定**動作と呼ぶことにする. 例えば x=4 のとき述語 x%2=0 は真であるが, 処理プログラムがその真偽を確認する動作のことである. このとき,「x=4 のとき述語 x%2=0 の真偽判定をする」などと言うことにする. これは処理動作レベルの用語である.

2.1 命令語

言語 CAST には 2 種類の基本的な述語がある. ひとつは数学的な意味の述語で,

y=x+1, x%2=0, subset(x,y), member(x,A)

などである. これらは変数 x, y に値が代入されるまで真偽を判定できないものである. もうひとつは, コンピュータ特有の述語で, 代入と入出力（テキスト表示, グラフ表示）に関するものである.

y:=x+1, xwriteln(0,x), show1(x,"plot")

ただし xwriteln( )はウインドウにテキストを表示する述語である. また show1 はグラフを描画する述語である. これらを述語と呼んでいるが, 本来は命令語であると考えるべきであろう. 本稿では, 命令語とそれ以外の述語を区別して考える. 命令語は, 指定の形式の x の値が代入される限り**必ず実行され**, **エラーでない限り常に真であると判定される**ことに注意したい（エラーの場合はシステムが停止する）.

2.2 連言（and 結合）

複合述語 p(x) <-> p1(x), p2(x), p3(x)； は, 与えられた x の値に対して, 右辺の p1(x), p2(x), p3(x)のすべての述語が真であるとき, 全体として真である. ただし, 上記のコンマは and を表わしている. 処理プログラムは, 右辺の左端 p1(x)から順に真偽判定を行い, もしも偽である述語が見つかれば真偽判定動作を終了し, 左辺の p(x)は偽であると判定する.

注意すべきは, **右辺の中で偽である述語が見つかったときは**, **それ以降の述語に対して真偽判定を行なわない**ことである. 例えば, p1(x)が真で, p2(x)が偽ならば, p3(x)の真偽判定を行なわないことになる. このとき, もしも p3(x)が命令語 xwriteln(0,x)であったならば, その命令は実行されない(Dialog ウインドウに x の値を表示しない)ことになる. したがって, p(x)の真偽判定動作中に必ず表示させたいならば, 順序を入れ替え,

p(x) <-> xwriteln(0,x), p1(x), p2(x)；

と書くべきであることが分かる.

2.3 選言（or 結合）

複合述語 p(x) <-> p1(x) or p2(x) or p3(x); は, 与えられた x の値に対して, 右辺の p1(x), p2(x), p3(x)のどれかの述語が真であるとき, 全体として真である. 処理プログラムは, 右辺の p1(x)から順に真偽判定を行い, もしも真である述語が見つかれば真偽

判定動作を終了し，左辺の p(x)は真であると判定する．ここでも，**右辺の中で真である述語が見つかったときは，それ以降の述語に対して真偽判定を行なわない**ので，上記の and 結合と同じように順序が関係することになる．

2.4 否定（not）

複合述語 p(x) <-> not(p1(x))；は述語 p1 の否定である．p1(x)が真のとき p(x)は偽であり，p1(x)が偽のとき p(x)は真となる．

2.5 含意（->結合）

複合述語 p(x)<->(p1(x))->(p2(x))；は，与えられた x の値に対して，右辺の p1(x)が真であるにもかかわらず p2(x)が偽であるときだけ，全体として偽である．処理プログラムは次のように動作する．まず p1(x)の真偽判定を行う．その結果，
1）p1(x)が偽ならば，p2(x)の真偽判定は行なわない．そして，全体 p(x)は真であると判定する．
2）p1(x)が真ならば，p2(x)の真偽判定も行なう．そのとき，
もしも p2(x)が真ならば，全体 p(x)は真であると判定する．
もしも p2(x)が偽ならば，全体 p(x)は偽であると判定する．

特に，処理プログラムは **p1(x)が偽ならば p2(x)の真偽判定を行なわない**ことに注意すべきである．特別な場合として，含意 (p1(x))->(p2(x))における p2(x)が命令語の場合を考える．上述したように，命令語は真偽判定されたときに必ず命令を実行する．したがって，含意 (p1(x))->(p2(x)) は，前提 p1(x)が成り立つなら p2(x)が成り立つはずだという意味の「推論の規則」というよりも，条件 p1(x)が成り立つなら，命令 p2(x)を実行せよという意味の「(条件→実行) の規則」あるいは「式の実行手順」であると解釈することができる．

述語 p1(x)が真であれば p2(x)が実行され，
述語 p1(x)が偽であれば p2(x)が実行されない．

つまり，p1(x)の真偽と p2(x)の実行・非実行を対応させることができる．先行研究における半関数の定義方法はこの動作に依っている．

次に連言と含意の処理動作を比較してみよう．

```
p5(x) <-> p1(x), p2(x);
p6(x) <-> (p1(x)) -> (p2(x));
```

前述の 2.2 と 2.5 で述べたように，どちらも「p1(x)が真のとき p2(x)は真偽判定され，p1(x)が偽のとき p2(x)は真偽判定されない」ので，同じ動作をすることになる．しかしながら，述語 **p1(x)が偽のとき，p5(x)は偽であるが p6(x)は真である**．同じ動作をするにもかかわらず，p5(x)と p6(x)の真理値が異なるので注意が必要である．このことが影響するのは，右辺にさらに述語を加えたときである．次の 2 つの述語 p51(x), p61(x)を考える．

```
p51(x) <-> p1(x), p2(x), p3(x);
```

```
p61(x) <-> ( (p1(x)) -> (p2(x)) ), p3(x);
```

いま p1(x)が偽であるとすると，上記のように p5(x)は偽であるが p6(x)は真であるので，p51(x)の右辺の p3(x)は真偽判定されないが，p61(x)の右辺の p3(x)は真偽判定されるのである．したがって，もしも常に p3(x)の真偽判定を期待してシステム開発しているときには，連言 p51(x)よりも含意 p61(x)の方が適している．

3　関数を定める述語

関数をユーザが定義するとき，言語 CAST においては次のように関数宣言 func( ) を行なってから関数の定義を行なう．

```
func([f]);
f(x) = y <-> y := x*x;
```

上記は，x の 2 乗を y に代入する関数 f を定義している．ただし，記号「:=」と「*」は，それぞれ代入を表わす操作と乗算を表わす演算である．

しかしながら言語 CAST の特別な用法として，関数宣言をしなくても，述語に関数の働きをさせることができる。例えば述語 p1(y,x) <-> y := x*x; は 2 次関数 $y = x^2$ を定めており，述語 p1(y,3) の真偽判定をすれば y=9 が得られる（y に 9 が代入される）．実際，Simcast の開発実行環境で，

```
p1(y,x) <-> y := x*x;
preprocess( ) <->
                x:=3,
                p1(y,x),
                xwriteln(0,"y=",y);
```

を実行すると、Dialog ウインドウに y=9 が表示される。（変数 x に値を定めれば述語 p(y,x) を真とする y が定まるのは，言語 Prolog のユニフィケーション機能そのものである．）言語 CAST において，述語で関数を定義することの利点は，関数宣言 func() を行なう必要がないことである．

3.1　計算の方向

ただし，述語 p1(y,x) <-> y:=x*x; の変数 y に値を代入しても x は「ひとつ」に定まらず，p1(9,x)はエラーとなる．関数の値はひとつでなければならない．

数学的には p1(9,x)から x=±3 を導くことはできるだろうが，そのためには方程式 $x^2=9$ を解く必要がある．平方根の場合には，組込みの算術関数 sqrt があるので，言語 CAST のばあいは，p2(y,x) <-> x:=sqrt(y); などとしておけば，y から x を導くことができる．すなわち述語ができることは，組込みの算術関数を組み合わせてできる数式（初等関数）の変数に値を代入して計算することだけである[注 1]．要するに，述語を関数として利用するときには計算の方向に注意しなければならない．

3.2 半関数

関数的な述語を定義する方法としては，上記のように右辺を初等関数として定義すればよい．あるいは，

p3(y,x) <-> x>0 , y := x;

など条件付きの写像として定義することができる．しかしながら，この述語 p3(x,y)に x= -3 を代入しても y が定まらないことに注意すべきである．これは標準的な話題であるが，定義域を広くとるとき，p3(y,x)は半関数を定めている．

半関数の正確な定義は付録の定義 2 で与える．すなわち，もしも与えられた x に対して p(y,x)を真とする y が存在するならば，y の値がただ一つに定まることである．これにより付録の定義 3 の半関数 Fp(x)を定義することが可能となる．

4 リストを作成する関数 defList

言語 CAST における関数 defList(p(y,x,[補助変数]), member(x,Xs)) は，述語 p を利用してリストを作成する関数である。それぞれ，Xs をインデックスリスト，述語 p(y,x,[補助変数])をリスト構成条件と呼ぶ．リスト構成条件 p は，各インデックス x に対して並べるべき要素 y を関係づけるためのものである．

4.1 defList の動作

直感的にいえば，defList は，**インデックスリスト Xs の要素 x をリスト構成条件 p(y,x,[])に代入し真偽判定したとき，述語 p(x,y)を真とする最初に確定する y の値をリスト化する**関数である．

その処理動作は次のようになる．

1）Xs の要素 x を最初から順番に p(y,x,[])に代入し，p(y,x,[])の真偽判定を行なう．したがって，Xs の要素の数だけ真偽判定を行なう．

2）p(y,x,[])の真偽判定によって，p を真とする y が確定すればそれを記憶する．ただし，複数の y が p を真とする可能性があっても，真偽判定動作の順序によって最初に確定する y だけを記憶する．

3）p(y,x,[ ])の真偽判定によって，p が偽となるときは，y が確定してもそれを記憶しない．

4）以上の繰り返しが終了したとき，記憶した y を並べてリストを作成する．

例えば各変数の定義域が自然数であるとき，述語 p1(y,x,[ ]) <-> x%2=0, y:=x; は，x が偶数であるとき y(= x)が定まる半関数である（付録の例 3）．x が奇数のとき p1 は偽となり y は未定義となるが，x が偶数のとき p1 を真とする y は唯一つ定まる（それは最初に確定することと同じである）からである．実際，

```
p1(y,x,[ ]) <-> x%2=0, y:=x;
Ls:=defList(p1(y,x,[ ]),member(x, [1,2,3,4,5,6]));
```

を開発実行環境 simcast で実行すると，偶数を集めたリスト Ls = [2,4,6]が得られる．

処理動作は,

x=1 のとき，x%2=0 が偽なので，y は確定せず，p1(y,1,[ ])は偽である.
x=2 のとき，x%2=0 が真なので，y=x=2 が確定し, p1(y,2,[ ])は真である.
x=3 のとき，x%2=0 が偽なので，y は確定せず，p1(y,3,[ ])を偽である.
x=4 のとき，x%2=0 が真なので，y=x=4 が確定し, p1(y,4,[ ])は真である.
x=5 のとき，x%2=0 が偽なので，y は確定せず，p1(y,5,[ ])は偽である.
x=6 のとき，x%2=0 が真なので，y=x=6 が確定し, p1(y,6,[ ])は真である.

となり，確定した y を並べてリスト Ls=[2,4,6]を得ている．確かに，インデックスリスト Xs の先頭の要素 x から順に p1(y,x,[ ])に代入し，「述語 p1 を真とする最初に定まる y」をリスト化している．また，Xs = [1,2,3,4,5,6] の要素の数だけ真偽判定を行なっている.

すなわち，リスト構成条件 p が半関数（関数）である場合には，x に対して述語 p(y,x,[ ])から唯一つ定まる y の値をリスト化する．付録の定義 4 はこの場合を定式化している.

4.2 含意の場合

次に，リスト構成条件が含意 p(y,x,[ ]) <-> (A(x)) -> (y:=f(x)); である場合を考察する．ただし，A(x)は述語であり，f(x)は Xs を定義域とする関数である．述語 A(x)は複合述語であっても良い．述語 A(x)は x を制限しており，関数 y:=f(x) は y を確定するためのものである. 付録の例 5 にあるように, 論理学においては述語 p（を真とする集合）は実は関数でも半関数でもない．しかしながら，リスト作成関数 defList で使用することが可能である．述語 defList(p(y,x,[ ]), member(x,Xs)) の処理動作は次のようになる.

1）インデックスリスト Xs の要素 x を「Xs の先頭から順に代入して」，述語 p(y,x,[ ]) の真偽判定を行なう.
2）そのため，2.5 項で述べたように，まず A(x)の真偽判定を行なう．もしも，A(x)が真であると判定できた場合，命令 y:=f(x) が実行され，変数 y の値が確定するので，その y の値を記憶する．逆に A(x)が偽であるとき命令 y:=f(x) は実行されず，y は未定義となり，記憶されない.
3）上記 1）2）を繰り返し，繰り返しが終了したら，記憶した y の値を並べてリストにする.

処理動作をみると，A(x)の真偽と y の確定／不確定が 1 対 1 に対応することが分かる．したがって，述語 (A(x)) -> (y:=f(x)) を用いた defList は「A(x)を真とする x に対して定まる y = f(x) をリスト化している」ことになる．それは，付録の定義 4 で定めた使用法を逸脱しているが，本文 4.1 項で述べた処理動作に従っており，特別な使用方法であると考えたい.

5　おわりに

本稿では，言語 CAST におけるリスト作成関数 defList(p(y,x,[ ]), member(x,Xs)) の新しい定義を提案した（本文 4.1 項前半）．すなわち，defList は，インデックスリスト Xs の要素 x をリスト構成条件 p(y,x,[ ])に代入し真偽判定したとき，述語 p(x,y)を真とする最初に確定する y の値をリスト化する関数である．

また，リスト構成条件を半関数的述語に限定したときの数理的定義を与えた（付録の定義 4）．数理的定義と実際の処理動作は確かに同じである（4.1 項後半）．defList は半関数的述語 p(y,x)を真偽判定したとき定まる y をリスト化する関数である．コンピュータ言語が平易な数理的体系によって裏付けできることが重要である．これまでは正確な数理的定義がなかったので，これはモデル理論アプローチに対する貢献であろう．

しかしながら，言語 CAST の現在の実装では，新しい定義の処理動作の範囲内で，数理的定義を逸脱する使用も可能である（4.2 項）．実装にもとづく真偽判定動作が重要であり，述語を並べる順序（第 2 節）が影響している．そのため defList の技巧的な利用も考えることができる．詳細は技術資料（旭 2009 Web）を参照されたい．

付録　定義と命題

定義 1（直積，関係，関数）

1）集合 X, Y の要素の組(x,y)を集めた集合を X×Y と書き，直積と呼ぶ．

$X \times Y = \{(x,y) \mid x \in X, y \in Y\}$

2）直積 X×Y の任意の部分集合を X と Y の関係と呼ぶ．

3）関係 S⊆X×Y が次の条件

$(\forall x, y, y')((x,y) \in S \wedge (x,y') \in S \rightarrow y' = y)$　　--- 値の一意性

を満足するとき，関係 S を半関数と呼ぶ．

また，半関数 S がさらに次の条件

$(\forall x \in X)(\exists y \in Y)((x,y) \in S)$　　--- 値の存在性

を満足するとき，S を関数と呼ぶ．

任意の関係 S が関数（半関数）であるとき，$f(x)=y \Leftrightarrow (x,y) \in S$ と定義すると，f は通常の意味の関数（半関数）である．実際，S が関数のとき，値の存在性から $f(x) \in Y$ は存在し，しかも値の一意性から $y = f(x)$ はひとつに定まる．一方，S が半関数のときは値の存在性を満たすとは限らない，つまり，任意の $x \in X$ に対し，f(x) の値を Y の要素として定義されないこともある．本稿では S と f を同一視し，S が関数であるときは S:X→Y，また，S が半関数であるときは S:(X)→Y と表記する．

定義 2（述語を真とする集合 Sp）

与えられた述語 p(y,x) の各変数の定義域（domain）が Y, X であるとき，

1）関係 $Sp \subseteq X \times Y$ を $(x,y) \in Sp \Leftrightarrow p(y,x)$ すなわち $Sp = \{(x,y) \in X \times Y \mid p(y,x)\}$ と定義する．このとき，$Sp \subseteq X \times Y$ を「述語 p(y,x)を真とする集合」と呼ぶ．

2）述語 p(y,x)を真とする集合 Sp が半関数（関数）であるとき，Sp を「述語 p(y,x)か

ら導かれる半関数（関数）」と呼ぶ.

補題 1

述語 p(y,x)を真とする集合を Sp⊆X×Y とする.

1）Sp が半関数であるための必要十分条件は,

$(\forall x, y, y')(\ p(y,x)\wedge p(y',x) \rightarrow y' = y\ )$

2）Sp が関数であるための必要十分条件は,

$(\forall x, y, y')(\ p(y,x)\wedge p(y',x) \rightarrow y' = y\ )$

$(\forall x)(\exists y)(\ p(y,x)\ )$

証明は省略する.

【例 1】自然数上の述語 $p(y,x) \Leftrightarrow y = x\ \%\ 2$ を考える．ここに, % は除算の余りをとる関数である．任意の x に対して必ず 2 で割った余り y がひとつ定まるので，述語 p(y,x)を真とする集合 Sp は関数である（ Sp: N→N ）.

【例 2】任意の関数 f: X→Y，g: X→Y と任意の述語 A(x)および

$p(y,x) \Leftrightarrow (A(x)\rightarrow y=f(x))\ \text{otherwise}\ (y=g(x))$

を考える．任意の x∈X に対して y∈Y が 1 つ定まるので，p(y,x) を真とする集合 Sp は関数である.

【例 3】自然数上の述語 $p(y,x) \Leftrightarrow (x\,\%\,2=0)\wedge(y=x)$ を考える．述語 p は，x が偶数であり，かつ y が x と等しいとき真となる述語である．したがって次が成り立つ.

$(\forall x, y, y')(\ p(x,y)\wedge p(x,y') \Rightarrow x \text{は偶数} \Rightarrow y' = x = y\ )$

よって述語 p を真とする集合は

x が偶数のとき Sp(x) = x

x が奇数のとき Sp(x) = 未定義

を満足する半関数 Sp:(N)→N である.

【例 4】任意の関数 f: X→Y と任意の述語 A(x) が与えられたとき，次の複合述語は半関数である.

$p1(y,x) \Leftrightarrow A(x)\wedge(\ y = f(x)\ )$

実際，A(x)が真である x に対してだけ y が存在し，p1(y,x)が真であるとき必ず y=f(x)がひとつだけ定まるからである．また,

$p2(y,x) \Leftrightarrow A(x)\wedge(\ A(x) \rightarrow y = f(x)\ )$

と定義すると，ド・モルガンの法則により,

$p2(y,x) \Leftrightarrow A(x)\wedge(A(x)\rightarrow y=f(x)) \Leftrightarrow A(x)\wedge(\text{not}(A(x))\vee y=f(x))$

$\Leftrightarrow (A(x)\wedge\text{not}(A(x)))\vee(A(x)\wedge y=f(x)) \Leftrightarrow A(x)\wedge(y=f(x)) \Leftrightarrow p1(y,x)$

であるから，p2(y,x) ⇔ p1(y,x) となり，p2 も半関数である.

【例 5】 任意の関数 f: X→Y と任意の述語 A(x)および p(y,x) ⇔ (A(x)→y=f(x)) を考える．論理学における含意の定義より，A(x)が偽であるとき，y=f(x)の真偽に関わらず p(y,x)は真である．換言すると，たとえ p(y,x)が真であっても A(x)を偽とする x に対して複数の y が存在する．よって p(y,x)は半関数ではない（もちろん関数でもない）．

【例 6】述語 p(y,x) ⇔ (y = x)∨(y = -x) を考える．ゼロでない任意の x に対して異なる二つの y，y'が存在し p(y,x)と p(y',x)がともに真となるので，p(y,x)を真とする集合は関数でも半関数でもない．

ここで，コンピュータ言語の Lisp や Prolog で使われるリストについて考える．二つのリスト Xs = [x1, x2,…, xn]と Ys = [y1, y2,…, yn] が与えられたとき，任意の i に対し xi = yi であるとき，すなわち長さと各成分の値が等しいとき Xs = Ys である．したがって，以下では (x1, x2,…, xn) のような n 項組をリストと呼ぶことにする．また，リスト Xs = (x1, x2,…, xn) が与えられたとき，その要素 xi が特に Xs の i 番目の要素であることを明示するため，一般的ではないが xi∈Xs と書くことにする．ただし，リスト Xs の要素は重複を許すものとする（xi = xj も許容する）．

複数のリストを結合してひとつのリストにすることをリストの連接と呼び，(x1,x2,x3)・(x4,x5) = (x1, x2,x3,x4,x5) などと表記する．要素のないリストを空リストと呼び，()と書く．空リストに関しては (x1)・() = (x1)，()・(x2) = (x2) と定義する．

定義 3（半関数によって生成されるリスト）
半関数 F:(X)→Y が与えられているとき，X の要素からなる任意のリスト Xs = (x1, x2,…, xn) に対して新たなリスト (F(xi) | xi∈Xs) を次のように定義する。

(F(xi) | xi∈Xs) = (F(x1))・(F(x2))・…・(F(xn)).

もしも F(xi)が未定義ならば (F(xi)) が空リストとなるため，リスト Ys = (F(xi) | xi∈Xs)の長さは Xs の長さよりも短くなることもある．

定義 4（defList）
与えられた述語 p(y,x)の各変数の定義域（domain）が，それぞれ Y, X であり，述語 p を真とする集合が半関数 Sp:(X)→Y であるとする．このとき，X の要素からなるリスト Xs に対して, Y の要素からなるリストを作成する関数 defList を次のように定義する．

defList(p(yi,xi), xi∈Xs) = (Sp(xi) | xi∈Xs)

このとき，Xs をインデックスリスト，述語 p をリスト構成条件と呼ぶ．

直感的に言えば，defList は，「p(y,x)を真とする確定した y」，すなわち y=Sp(x)をリスト化する関数である．ただし，defList はリスト構成条件 p(y,x)を真とする集合が半関数（関数を含む）であるときにのみ使用するものとする．

前述の例 1 と例 2 のように，リスト構成条件 p を真とする集合 Sp が関数であるとき，defList によって生成されるのは，インデックスリスト Xs の要素を代入した Sp(xi) を並べたリストである．例えば例 1 のように p(y,x) ⇔ y = x % 2 であるとき，

defList(p(yi,xi), xi∈(1,2,3,4)) = (1)・(0)・(1)・(4)・(0) = (1,0,1,0)

である．

一方，リスト構成条件が半関数の場合には，値が定義されている場合の Sp(xi)を並べたリストが生成される．例えば例 3 のように，述語 p(y,x) ⇔ (x % 2 = 0)∧(y = x)から導かれる半関数 Sp:(N)→N は，x が偶数のとき Sp(x) = x であり，x が奇数のとき Sp(x) = 未定義 となる．このとき，

xi が偶数のとき yi は唯一確定し，(Sp(xi)) = (xi),

xi が奇数のとき yi は確定せず，　(Sp(xi)) = (),

となるから，例えばインデックスリストを Xs = (1,2,3,4,5,6) とすると，

defList(p(yi,xi), Xs) = ()・(2)・()・(4)・()・(6) = (2, 4, 6)

となる．

また例 4 により，任意の関数 f: X→Y と任意の述語 A(x)および

p1(y,x) ⇔ A(x)∧y=f(x)；

p2(y,x) ⇔ A(x)∧(A(x) → y=f(x))；

が与えられたとき，p2(y,x) ⇔ p1(y,x)であるから，X の要素からなる任意のリスト Xs = (x1, x2,…, xn) に対して次が成立つ．

defList(p1(yi,xi), xi∈Xs) = defList(p2(yi,xi), xi∈Xs) ．

【注】

[1] 数学的に解の公式があっても組込み述語がない場合には，工夫が必要である．たとえば 3 乗根を与える組込み述語は CAST にはないので、方程式を解くための問題解決システム（Solver）などが必要である．

【参考文献】

旭貴朗，高原康彦，中野文平ほか（2008）「経営情報システム開発のためのモデル記述言語 CAST」経営情報学会誌，Vol. 16，No. 4，pp.19-30.

旭貴朗（2009）「モデル理論アプローチによるシミュレーション --- 開発実行環境 Simcast」東洋大学経営論集 73 号，pp. 33-51.

旭貴朗（2009 Web）技術資料「言語 CAST で漸化式を解く」
http://www2.toyo.ac.jp/~asahi/research/simulation/docs/recurrence.html（2014.08.29）

旭貴朗（2011）「モデル理論アプローチによるシミュレーション --- 自律分散システムの開発手順」東洋大学経営論集 78 号，pp. 177-188.

高原康彦，齋藤敏雄，旭貴朗，柴直樹（2007）『形式手法　モデル理論アプローチ : 情報システム開発の基礎』日科技連出版

Y. Takahara & Yonmei Liu (2006), *Foundations and Applications of MIS*, Springer Verlarg