知識工学と人工知能 51－1
（1987. 3. 12）

文字列領域のロジック・プログラム

赤間 清

（北海道大学 文学部 行動科学科）

項が文字列（あるいは単語列）であるようなホーン節からなるロジック・プログラムについて述べる．ＣＦＧと prolog の親密性はよく知られているが，本論文で導入する「文字列領域のロジック・プログラム」はそのもっとも明快な基礎付けの場を提供する．それは，ＣＦＧや翻訳文法を含む表現の体系であり，帰納的学習システムＬＳ／１の主要な知識表現の基礎を与えている．一般の場合，あるいは，質問応答システムに使われる場合の応答文探索（推論）の方法に関して， prolog や BUP との関連にもふれる．

Logic Programs in Symbol String Domain

Kiyoshi Akama
(Hokkaido University, Sapporo-shi, 060, Japan)

This paper describes logic programs which consist of Horn clauses whose terms are symbol strings. It is well known that CFG and Prolog are closely related. Logic Programs in Symbol String Domain introduced in this paper provides more suitable basis to explain the relation. Their expressive power as the representation frameworks subsumes the ones of CFG and Translation Grammar and are used as the main knowledge representation methods for an inductive learning system named LS/1. This paper also discusses the inference technique for generating answers to given questions, and its relation with Prolog and BUP.

1．　まえがき

項が文字列（あるいは単語列）であるようなホーン節からなるロジック・プログラムについて述べる．それは，ＣＦＧや翻訳文法を含む表現の体系であり，帰納的学習システムＬＳ／１の主要な知識表現の基礎を与えている．ＣＦＧと prolog の親密性はよく知られているが，本論文で導入する「文字列領域のロジック・プログラム」はそれよりも明快な基礎付けの場を提供する．一般の場合，あるいは，質問応答システムに使われる場合の応答文探索（推論）の方法に関して， prolog や BUP との関連にもふれる．

都合により，記述は例を中心とした粗いスケッチに留める．厳密な記述は別の機会に譲る．

2．　文字列領域のホーン節

Ａを空でない集合とする．Ａの元を文字，Ａをアルファベットと呼ぶ．Ａ上の文字列とは，Ａの任意の元を（０個以上）有限個並べたものである．各文字列における文字の出現回数をその文字列の長さという．長さ０の文字列をλで表わす．Ａ上の文字列全体の集合をA^*で表わし，Ａ上の長さ１以上の文字列全体の集合をA^+で表わす．簡単のため，Ａの元をそのまま，Ａ上の長さ１の文字列とみなすことがある．

Ｃ，Ｖ，Ｗを，空でないたがいに排反な集合とし，それぞれ，定数集合，１文字変数集合，ｎ文字変数集合と呼ぶ．それらの元は，それぞれ，定数，１文字変数，ｎ文字変数と呼ばれる．変数とは，１文字変数またはｎ文字変数のことである．我々が考察の対象とするのは，Ｃ∪Ｖ∪Ｗ上の文字列である．以後本論文では，たんに文字列というときには，Ｃ∪Ｖ∪Ｗ上の文字列を指すものとする．そのうちＣ上の文字列は，定文字列と呼んで区別する．

ｐがｎ項述語で，$t_1,\ldots,t_n$ が文字列のとき，$p(t_1,\ldots,t_n)$ は原始論理式(atom) と呼ばれる．文字列領域の program clause とは，

$$A \leftarrow B_1, \cdots, B_n$$

の形の式である．ここで，ｎは負でない整数であり，Ａ，B_1，…，B_n は atom である．Ａは head，B_1，…，B_n は body と呼ばれる．文字列領域の logic programとは，文字列領域の program clause の有限集合である．

代入(substitution)θとは，変数と文字列のペアの有限集合のことである．

$$\theta = \{ X1/S1, \cdots, Xn/Sn \}$$

ここで，Xiが１文字変数ならば，Siは定数または１文字変数でなければならない．代入θの元 Xi/Si を Xi の束縛と呼ぶ．すべての Si が定文字列のとき，θは定代入(ground substitution)と呼ばれる．

Ｓを文字列，$\theta = \{ X1/S1, \cdots, Xn/Sn \}$ を代入とする．Ｓのθによる例 (instance) とは，Ｓにおける変数 Xi のすべての出現を同時に文字列 Si に置き換えて得られる文字列である．Ｓのθによる例を $S\theta$ で表わす．

代入θ，σに対して，$\theta \geqq \sigma$とは，$\sigma = \theta\pi$を満たす代入πが存在することである．$\geqq$は反射律，推移律を満たす．$\theta \sim \sigma$を$\theta \geqq \sigma$かつ$\sigma \geqq \theta$で定義すれば$\sim$は同値関係であり，$\sim$で商をとることにより$\geqq/\sim$は反順序となる．今後$\sim$による商集合のうえでしばしば議論し，$\geqq/\sim$を単に$\geqq$と書くことがある．

２つの文字列S_1，S_2の unifierθ とは，$S_1\theta = S_2\theta$ を満たす代入θのことである．S_1，S_2の unfier θ が，順序$\geqq$について極大であるとき，極大 unifier と呼ばれる．S_1，S_2が与えられて，その極大 unifier （の代表元）を求めることを，マッチングまたはユニフィケーションと呼ぶ．

表現を明確にし，推論のプログラムを与えるのが便利なように，文字列や原始論理式を表現する方法を次のように指定する．文字列に出現する文字はアトムで，文字列はアトムの「リスト」で表現する．定数と１文字変数とｎ文字変数は，アトムの表現において次のように区別される．

ｎ文字変数－－－＄から始まるアトム
１文字変数－－－％から始まるアトム
定数　　－－－それ以外のアトム

例えば，

Ｐ１＝"だちょうは飛ぶか飛ばないか"
Ｐ２＝"＄は飛ぶか飛ばないか"

は（漢字やかな文字が１文字でアトムになると仮定して），それぞれ，

Ｑ１＝［だ　ち　ょ　う　は　飛　ぶ　か　飛　ば　な　い　か］
Ｑ２＝［＄　は　飛　ぶ　か　飛　ば　な　い　か］

と表現される．

このような表現を採用すると，対象としていわゆる文字列以外に，例えば単語列も容易に扱い得る．単語列を扱う例を次に挙げる．

Ｑ３＝［WHAT COLOR IS SNOW］
Ｑ４＝［IT IS WHITE］

文字列領域の unification では極大 unifier が複数個（有限個または無限個）ありうる．例えば，パターンが，

Ｑ８＝（［％１　は　飛　ぶ　か］［飛　び　ま　せ　ん］）
Ｑ９＝（［＄１　は　＄２　か］［＄３　ま　せ　ん］）

のとき，極大 unifier（の代表元）は，

％１　／　［＄１］
＄２　／　［飛　ぶ］
＄３　／　［飛　び］

だけである．無限の解の存在するのは，例えば，［１　＄］と［＄　１］を unify するときで，この解は，

＄／［］，＄／［１］，＄／［１　１］，＄／［１　１　１］　…

となる．

３．　文字列領域のロジック・プログラムのセマンティクス

Ｃ∪Ｖ∪Ｗ上の文字列領域のロジック・プログラムＰのセマンティクスを次のように定める．Ｐに出現する述語全体の集合をＶとする．Ｃ上の文字列の組全体の集合のベキ集合，すなわち，powerset(∪［Ｃ*］^{n})をＳとする．Ｘを，ＶからＳへの写像ｘの空間とする．Ｘの任意の元ｘ：Ｖ→Ｓ は，各述語にＣ上の文字列の組の集合を対応させる．プログラムＰによってＸ上の変換Ｆが，標準的な prolog の場合と同様に定められる．文字列領域のロジック・プログラムの意味は，

Φ，Ｆ（Φ），Ｆ^{2}（Φ），Ｆ^{3}（Φ），…

の極限によって定められる．それは，方程式　ｘ＝Ｆ（ｘ）　の最小解であり，各述語にＣ上の文字列の組の集合を対応させる．

４．　文字列領域の推論システム

文字列領域のロジック・プログラムのための推論システムを prolog 風に書くと，

```
demo(pred(t1,t2,...,tn), Sub1, Sub3) ←
        clause(pred(s1,s2,...,sn), Body),
        matchs((t1,t2,...,tn),(s1,s2,...,sn),Sub1,Sub2),
        demos(Body, Sub2, Sub3).
demo((A, B), Sub1, Sub3) ← demo(A, Sub1, Sub2), demo(B, Sub2, Sub3).
demo(true, Sub, Sub).
```

となる．ここの clause は文字列マッチをせずに述語名だけみてクローズを持って来る．matchs は文字列のｎ項組のマッチングをおこない代入を更新する．matchs は複数回成功する可能性があることに注意せよ．推論を起動するには，

```
demo(pred(t1,t2,...,tn), φ, Sub)
```

とすればよい．Sub に answer substitution が返る．

５．　文の集合の表現

すべての文脈自由文法（CFG）をロジック・プログラムで表現できる．例えばcfg

```
A → r A s B
A → p q
B → k l m
```

は，

```
a([r $a s $b]) ← a([$a]), b([$b]).
a([p q]).
b([k l m]).
```

と書ける．Ａの生成する言語は，１引数述語 a を満足する文字列に対応する．それらは，明らかに，｛ （ｒ）ⁿｐ ｑ （ｓ ｋ ｌ ｍ）ⁿ　｜　ｎ∈Ｎ ｝である．

１引数述語が，文脈自由文法ではない言語を表現する場合が存在する．例えば，

```
u([$1 $2 $3]) ← z([$1], [$2], [$3]).
z([a $1], [b $2], [c $3]) ← z([$1], [$2], [$3]).
z([a], [b], [c]).
```

はその例である．１引数述語 u を満足する文字列全体の集合は，

｛ ａⁿｂⁿｃⁿ　｜　ｎ∈Ｎ ｝

であるが，これは文脈自由言語ではない．

６．　文のペアの集合の表現

６．１　質問応答の対応の表現

２項述語を満足する文字列全体の集合は，文のペアの集合を表現する．たとえば，

```
qa([$1 NO IRO HA NANI], [$2 DESU]) ← color([$1], [$2]).
qa([$3 I MONO HA NANI], [$4 DA]) ← color([$4], [$3]).
color([YUKI], [SIRO]).
color([SATOU], [SIRO]).
color([RINGO], [AKA]).
```

は，次の集合を表現している．

```
｛ ([SATOU NO IRO HA NANI], [SIRO DESU]),
  ([YUKI NO IRO HA NANI], [SIRO DESU]),
  ([RINGO NO IRO HA NANI], [AKA DESU]),
  ([SIRO I MONO HA NANI], [YUKI DA]),
  ([SIRO I MONO HA NANI], [SATOU DA]),
```

([AKA I MONO HA NANI], [RINGO DA]) }

これは質問文と応答文のペアの集合の表現にも使うことができる．

６．２　対訳集合の表現

つぎの例は，和文英訳の対訳集合である．

```
{ ([WATASI HA JIRO DA], [I am Jiro])
  ([WATASI HA TARO DA], [I am Taro])
  ([ANATA HA JIRO DA], [you are Jiro])
  ([ANATA HA TARO DA], [you are Taro])
  ([WATASI HA SHOUNEN DA], [I am a boy])
  ([ANATA HA SHONEN DA], [you are a boy]) }
```

これを表現するには，ロジック・プログラム：

```
trans([WATASI HA $1 DA], [I am $2]) ← name([$1], [$2]).
trans([WATASI HA $1 DA], [I am $2]) ← class([$1], [$2]).
trans([ANATA HA $1 DA], [you are $2]) ← name([$1], [$2]).
trans([ANATA HA $1 DA], [you are $2]) ← class([$1], [$2]).
name([TARO], [Taro]).
name([JIRO], [Jiro]).
class([SHOUNEN], [a boy]).
```

の，述語 trans を用いればよい．

佐藤らは，翻訳の学習のための文法として，翻訳文法[8]と呼ばれる知識表現を提案している．これは文脈自由文法（ＣＦＧ）を２つペアにしたような形の文法であり，対訳集合を表現できる．翻訳文法での非終端記号の役割は，ロジック・プログラムでは，大域的な意味を持つ述語と局所的な意味しか持たない変数の２つが分担して担っている．上の例の場合，translate には，開始記号が対応する．

７．　質問応答のための推論システム

７．１　質問応答のための推論システム

４節のシステムを質問応答のために特殊化すると次のようになる．

```
question-answer() ← read(Q),
                    demo(qa(Q, R), φ, Sub),
                    apply(R,Sub,A),
                    print(A),
                    more(no).
```

７．２　クローズの制限

これは，前のプログラムの matchs を呼出すので，効率が悪い．文字列のマッチングはＳ式のマッチングに比してかなり複雑である．マッチングの解が一般には一意には定まらないばかりか，解が無限個存在する場合もある．マッチングのアルゴリズムの複雑さは応答探索の速度を低下させる要因となる．しかし質問応答の場合は，プログラムを構成するクローズに制限を課すことによって，必要な文字列マッチングをより簡単な場合だけに留めることができ，応答生成を高速化できる．

ロジック・プログラムを構成するプログラム・クローズに対して，

(1)　「bodyに存在する変数は必ずheadに存在する」

(2) 「bodyの各述語に出現する定文字列でない引数は，互いに異なる変数（変数1個からなるストリング）だけである」

という制限を加える．この制限は例えば，

```
e([a b c], [de]).
p([$1 + $2 = Q]) ← q([$1], [$2], [y o s h i]), r([$1]).
```

などのクローズを許すが，

```
e([a $b $c], [de]).
p([$1]) ← q([$1], [h u b s], [$1]).
```

などは許さない．

7．3 制限されたプログラムでの応答探索

前節のように制限されたクローズだけを持つプログラムをＰとする．各引数が，定数ストリング（定数だけからなるストリング）または変数（変数1個からなるストリング）であり，各変数はたがいに異なるような問い合せ $p(t_1,...,t_n)$ のクラスＫを考える．demo に対するクラスＫの問い合せが，トップ・ダウンの推論によって作り出す問い合せと，そのために必要なユニフィケーションのクラスについて考える．

例を挙げる．問い合せＱ１を p([a a b c c], [$1], [$2])，クローズＣ１を

```
p([$3 b $4], [$3 $4 $5], [$4]) ← q([$3], [$4], [$5]), r([$5]).
```

とする．Ｑ１はＫに属しＣ１は上記の制限を満たす．両者の head を unify するには

```
[a a b c c] = [$3 b $4],
[$1] = [$3 $4 $5],
[$2] = [$4]
```

の３つの条件が要請される．ユニフィケーションは２種類に分けられる．一方は，問い合せ側に定文字列がある場合であり，他方は問い合せ側に変数がある場合である．前者より，そこに出現する head 側の変数は定文字列に束縛される：

```
$3/[a a], $4/[c c]
```

後者より，問い合せ側の変数はそれぞれ文字列に束縛される：

```
$1/[a a c c $5], $2/[c c]
```

となり，新たな問い合せ，q([a a], [c c], [$5]) が得られる．これはやはりクラスＫに属する．q([a a], [c c], [$5]) の答えが返って来た時に $5 が定文字列に具体化されていれば，つぎの問い合せ r([$5])は，やはりクラスＫに属することになる．

一般に次のことがいえる．クラスＫの問い合せが，トップ・ダウンの推論によって作り出す問い合せは，すべてクラスＫに属し，すべての変数は定文字列に具体化される．したがって，この場合，上記の２種の簡単なユニフィケーションだけをやれば十分であり，応答生成は一般の場合に比べて著しく高速化できる．

8． ボトム・アップ・パージング

文字列領域のロジック・プログラムの表現に十分制限をつけ，ＣＦＧに対応する質問応答の範囲に抑えた場合，当然ながらＣＦＧに対する技術が流用できる．ここではボトム・アップ・パーザー BUP を応用した例を述べる．まずロジック・プログラムによる翻訳の知識が次のように与えられたとする．ただし，i06 などはクローズを assert するとき自動的に割り付けられた id 番号である．

```
;----------------------> trans.test <-----------------

i06  (rule (v-root (does $1 $2 :?) ($3 ha $4 desuka))
           (np     ($1) ($3))
           (gvp    ($2) ($4)))
i07  (rule (v-root ($1 $2) ($3 ga $4))
           (np     ($1) ($3))
           (vp     ($2) ($4)))
i08  (rule (np  ($1 $2) ($3 $4))
           (det ($1) ($3))
           (n   ($2) ($4)))
i09  (rule (vp  ($1 $2) ($4 wo $3))
           (vt  ($1) ($3))
           (np  ($2) ($4)))
i10  (rule (gvp  ($1 $2) ($4 wo $3))
           (gvt  ($1) ($3))
           (np   ($2) ($4)))
i11  (rule (n (boy)  (shounen)))
i12  (rule (n (girl) (shoujo)))
i13  (rule (det (a) (hitorino)))
i14  (rule (det (the) (sono)))
i15  (rule (vt (likes) (sukidesu)))
i16  (rule (vt (loves) (aisuru)))
i17  (rule (gvt (like) (suki)))
i18  (rule (gvt (love) (aisi)))

;----------------------> trans.test <-----------------
```

このとき，

```
   (translate)
```

によってシステムが起動される．さらに，

```
     E>(a girl loves the boy)
```

と入力して，その和訳を聞くと，システムは，入力文の構文解析木：

```
     S>(i07 (i08 (i13) (i12)) (i09 (i16) (i08 (i14) (i11))))
```

をもとめ，そのあとその木をつぎつぎに適用して，和訳を生成する．その過程は，

```
        (det ((a) (hitorino)))
        (n ((girl) (shoujo)))
        (np ((a girl) (hitorino shoujo)))
        (vt ((loves) (aisuru)))
        (det ((the) (sono)))
        (n ((boy) (shounen)))
        (np ((the boy) (sono shounen)))
        (vp ((loves the boy) (sono shounen wo aisuru)))
     J>(hitorino shoujo ga sono shounen wo aisuru)
```

である．拡張 prolog ：ＰＡＬによるプログラムのトップ・レベルは，

```
;-------------------------------------------------------------> translate
(define translate
  (() (loop (rind "E>" *E) (if (= *E ()) (exit))
            (goal v-root *S  *E ()) (princ "S>") (print *S)
            (apply-rule *S (p v-root (*E *J)))
            (terpri) (princ "J>") (print *J) (terpri))))
;-------------------------------------------------------------> translate
```

である．goal が BUP と同様のボトム・アップ・パージングを行い，できあがった構文解析木を利用して apply-rule が応答を生成する．

これらを文字列領域のロジック・プログラムの全体に拡張する方法については，別の論文で議論する．

９．　むすび

文字列領域のホーン節からなるロジック・プログラムについて非常に粗くスケッチした．これらは，帰納的学習システムの研究において，その知識表現である標号網についての知見をロジック・プログラミングの枠組みの言葉で書き直したものである．詳細や拡張については別途報告する．

参考文献

［１］赤間　清，市川　惇信：学習システムのための記号列集合の表現，電子通信学会論文誌 Vol.61-D　No.9 P649-656 (1978)

［２］赤間　清：　連立方程式に基づくネットワークを用いた表現のシステムとその重要なサブクラス，HBSR M 3 1983　北海道大学

［３］赤間　清：　ベキ集合上への自然な拡張によって得られる写像を用いた標号網のクラス　HBSR M 6 1983　北海道大学

［４］赤間　清：文字列領域における一般化と統一化のアルゴリズム，情報処理学会，知識工学と人工知能研究会資料，45-6，(1986)

［５］赤間　清：最良優先探索ＰＲＯＬＯＧ，情報処理学会，知識工学と人工知能研究会資料，47-8,p57-64 (1986)

［６］小長谷明彦，梅村讓：Shape Upの文字列照合アルゴリズムについて，情報処理学会，記号処理研究会資料，24-7 (1983)

［７］佐藤　理史，長尾　真：文法推論に基づいた翻訳文法の学習方式，シンポジウム「学習の諸問題」報告論文集　p132-142 (1986)