E-006

文書構築のためのツールとしてのProlog
Prolog as a Tool for Document Construction

花川 賢治[†] 天笠 俊之[†] 波多野 賢治[†] 宮崎 純[†] 植村 俊亮[†]
Kenji Hanakawa Toshiyuki Amagasa Kenji Hatano Jun Miyazaki Shunsuke Uemura

1. はじめに

本研究は我々が行ってきた動的文書システムの研究 [1] の延長線上にある．動的文書システムとは，文書を小さな文書部品に分解して保存しておき，要求されたときにユーザが指定した内容と構造を持つ文書を作成するシステムである．この研究では,「文書」とは自然言語で書かれた事実についての記述であるとし,「文書部品」とは,それだけを読んで意味がわかるように加工した文書の断片であるとした．文書部品は独立性が高く可搬性に優れるので共有と再利用に適する．そして，文書部品による共有と再利用は，文書データを扱うシステムの冗長性を低下させ効率を向上させるのに役立つ．動的文書システムの一つの応用として,「本」よりも小さな文書部品を情報流通の基本単位とすることにより，著者の協調的な書く行為 (Socaial Writing) と利用者の多様な個別の要求に適合した読む行為 (Individual Reading) を支援するシステムを提案した [2].

本研究では，対象とする文書を構造化文書に限定して，構造化文書が持つ情報を Prolog の事実に分解・保存し，それに対し Prolog の質問を行うことで，ユーザが指定した構造で構造化文書を再構築する方法を提案する．また，Prolog の処理系にわずかな拡張を加えることで実装を行う．もともと宣言型言語である Prolog は，記述が簡潔であることが知られている．それに加えて，一つの Prolog の質問で，事実を検索することと，得た事実を組織化することを，同時に指定することができため，非常に直感的で簡単な入力で目的とする文書を得ることができる．

以下では，まず，Prolog の質問から文書の木構造を構成する基本的なメカニズムについて述べ，その次に，実用性を持たせるために導入した書式付き出力とマクロ定義の機能について述べる．さらに，ここで述べた方法の有効性を示すために，レストランガイドの例題を示す.

2. Prologによる文書の構成処理

2.1 質問の論理構造と木との対応付け

本研究では，以下のような文書の木構造の意味論を採用する [3]:

文書の木構造のノードは述語を意味し，根からあるノードへのパスはその上にあるノードの論理積が真であることを意味する.

したがって根から葉に至るパスは，質問全体が真であることを意味する．Prolog の質問は，述語，AND 演算子（,），OR 演算子（ ; ）から構成されるが，この意味論に従えば，それぞれは，ノード，階層化，枝分かれに対応させることができる．

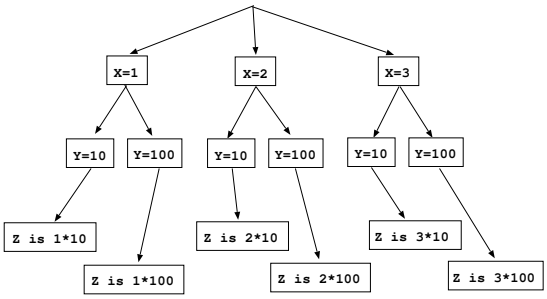

図 1: (X=1;X=2;X=3),(Y=10;Y=100),Z is X * Y の木構造

具体例として，以下の質問を上に述べた対応付けで木に変換すると，図 1 のようになる．

```
?- (X=1;X=2;X=3),(Y=10;Y=100),Z is X * Y.
```

Prolog のインタプリタはトップレベルで失敗すると，質問全体を再試行するが，その結果得られる複数の解のそれぞれが葉に至るパスに対応する．これは枝分かれのない枝が根に接続されている木構造に対応し，Prolog の質問を論理的に等価な積和標準形に変えたものと一致する．積和標準形の木構造から最初に与えられた質問の論理構造を直接反映した木構造へ変換することは容易である．つまり，Prolog 処理系に質問を再試行させ得られた結果から質問に対応した木構造を生成することができる．

2.2 Prolog インタプリタの拡張

図 2 に示すような，質問を与えるとパスを返す規則 call_ を定義する．ただし，call_の基本的な部分は最初の 3 個の規則で，最後の規則は後で述べるマクロ機能のための拡張である．call_ は Prolog の組み込み述語 call と同様に第 1 引数に与えられた質問に答える．それに加えて，成功した述語のリスト（パス）を第 2 引数に返す．

call_は再試行するたびに異なったパスを第 2 引数に返す．多くの Prolog 処理系にある組み込み述語 findall を用い，call_が失敗するまで再試行させることで全解を求め，それに対応したパスのリストを求める．パスのリストは前の方（根に近い方）の要素が共通なパスが連続する．共通要素部分に対応した親ノードを生成し，それを持つパスの残りの部分をその子とすることで，木構造を生成することができる．

3. 書式付き出力とマクロ定義

3.1 書式付き出力

文書の出力処理は，木構造に対して，すべてのノードを根を出発点とする深さ優先探索のアルゴリズムで訪問し，訪問したノードを何らかの方法で文字列に変換して出力することであるが，ノードは述語であるので，Prolog

[†] 奈良先端科学技術大学院大学 情報科学研究科

```
call_((X;Y), Path):- !,
        (call_(X, Path); call_(Y, Path)).
call_((X,Y), Path):- !,
        (call_(X,Path1), call_(Y,Path2)),
        append(Path1, Path2, Path).
call_(X, [X]):-
        call(X).
call_(Name, Path):-
        ::(Name, Definition),!,
        call_(Difinition, Path).
```

図 2: 規則 call_の定義

の write などで直接出力したのでは文書の体裁にはならない．そこで，書式を定義し，それに従い出力を行う機能を追加する．述語に書式を付加した構文は以下のようになる．

述語:始点書式リスト:終点書式リスト

書式リストの要素は，アトム，数，変数であり，連結して出力される．始点書式リストはノードを訪問したときに，終点書式リストはノードを去るときに出力される．終点書式リスト，あるいは両方を省略することができる．

3.2 マクロ定義

質問の部分を再利用するために以下の構文を持つマクロ定義が扱えるように拡張を行った．call_の最後の規則がその部分である (図 2).

マクロ名 (引数, ...) :: マクロの本体

マクロは述語と同様に引数を与えて呼び出すことができる．マクロが呼ばれると，:: の右辺のマクロ本体が質問される．マクロ本体は述語と論理演算子から構成される．マクロ本体にマクロ呼び出しを含めること，つまりマクロの入れ子も可能である．

4. ケーススタディ

このシステムの有効性を確かめるために、いくつもの応用例を試した。そのうちの一つを紹介する。*http://gourmet.yahoo.co.jp/* は WWW 上のオンラインのレストランガイドである．このサイトにはレストランごとの HTML ファイルが置かれている．HTML ファイルには，カテゴリ，所在地，住所，電話番号，最寄り駅，営業時間，主なメニューなどの項目が含まれている．それぞれに *category*, *location*, *address*, *tel*, *station*, *open*, *menu* などの述語を定義した．すべての述語は 2 個の引数を持ち，第 1 引数はレストランの名前，第 2 引数は属性値である．そのほかに区と街を関係付ける *ward_town* も定義した．これらの述語による事実データベースは、HTML ファイルから簡単な Perl スクリプトを用いて作成した．

以下の要求を満たすレストランのガイドを作成するための質問について考える．

1. 属性による分類を指定する．
   ある区を指定して，その区内の街ごとにレストランを表にまとめる．
2. 出力するフォーマットを指定する．
   レストラン名，住所，メニュー (箇条書き) を出力する．
3. 選択する条件を指定する．
   イタリア料理またはフランス料理のレストランだけを選ぶ．

まず，最初の要求は， *ward_town* と *sort_by_town* マクロにより実現する．*sort_by_town* マクロでは，街に対応した行を作成し，その中にレストランを格納する入れ子の表を格納している．

次のフォーマットの要求は *page* マクロで実現する．*page* マクロの引数には具体的なレストランの名前がバインドされ呼び出される．*page* マクロの最初の true はレストランの名前を出力する目的のためにあり，次の *address* でそのレストランの住所を得て，br タグを加えて出力している．その次の true はメニューの項目を箇条書きにするための ul タグを出力するためのものである．最後の menu の行でメニュー項目に li タグを付加して出力する．

最後のレストランを探す条件は，質問の中間に置かれた `(category(R,'イタリア料理'):[];category(R,'フランス料理'):[])` により与えている．これによりカテゴリがイタリア料理かフランス料理のレストランだけが出力される．

```
sort_by_town(T,R)::
  true:['<tr><td>',T,'</td><td><table border=1>']
      :['</table></td></tr>'],
  location(R,T):['<tr><td>']:['</td></tr>'].

page(R)::
  true:[R,'<br>'],
  (address(R,Adr):[' 住所:', Adr, '<br>'];
   true:[' メニュー:<ul>']:['</ul>'],
   menu(R,Menu):['<li>', Menu, '</li>']).

?- ward_town(' 中央区',T):[],
   sort_by_town(T,R),
   (category(R,' イタリア料理'):[];
    category(R,' フランス料理'):[]),
   page(R).
```

5. おわりに

本発表では，構造化文書を Prolog の事実データベースに変換し，それに対し Prolog の質問を行うことで文書を再構成する方法を提案した．また，実際に具体的な例題を試し，非常に簡潔な質問の入力で，意図した構造を持った複雑な書式の出力が得られることを示した．今後の課題は，より厳密な評価と，XML のような他のアプローチとの比較である．

参考文献

[1] K. Hanakawa, H. Takeda, T. Nishida: Construction of a Dynamic Document Using Context-Free Pieces. Proc. SEKE'99. (1999) 212–216

[2] K. Hanakawa, M. Yoshikawa and S. Uemura: Social Writing and Individual Reading with Dynamic Documents. Proc. KES2001 (2001) 1600–1606.

[3] K. Hanakawa, T. Amagasa, K. Hatano, J. Miyazaki, S. Uemura: A Predicate Based Query Language for User-Specified Document Organization. SCI 2004. (2004) (to appear)