簡単な構造を持つ記述名の入力解析

1Aa－8

古宇田 フミ子
fumiko@mtl.t.u-tokyo.ac.jp
東京大学 工学部*

1 はじめに

記述名は、対象の属性や動作、等の説明をすることで対象を識別しようとする名前である。

計算機資源の識別子が分からない時、代わりにその記述名を利用者が能動的に用いることができれば、資源の利用や情報検索等に役立つであろう。

このような記述名管理では、通常の識別子の名前管理と異なり、先ず、利用者インタフェースにおける記述名の表現法やその入力解析法が問題となる。これまでに、筆者らは簡単な構造を持つ利用者向き記述名表現法を提案してきた[F94]。ここでは、利用者により記述された記述名の入力解析法について、構成法を検討し、実装を試みたので、これについて報告する。

2 記述名に関する前提

提案した記述名表現法は幾つかの記述側面を持つ。例えば、属性機能(attribute function)は対象を属性の面から捉え、提供機能は(providing function)は対象の持つ動作から捉えた記述である。

このような記述名表現は、図1と図2のように表される。この記述法に基づくと、例えば、「psファイルをコピーする」は(copy p(Affected: file of:ps))となる。

2.1 記述表現法の構造概略

記述名は文の意味要素の枠組を取り出した構造を持つ。この構造は意味の区切りを表す記号とそれに対応した内容を表す単語(term)の(集合)からなる。

プリミティブな意味の区切り記号は、「予約語:」の形で示され、ofで始まる句を表すof:, 形容詞的な修飾句を表すattr:, 句や節の役割を表すrolename (Partname, Circname)、等がある。複合構造の意味の区切り記号は、(•), P(•), p(•), c(•), e(•), id(•), us(•)、であり、区切りは、(と), 接続の記号には、and は空白、or は | が用いられる。

```
SFe ::= clause | PA | id | us | Env    /* 記述側面 */
clause ::= {( ST [PA] [CR] )}          /* 記述側面 提供機能 */
PAe ::=  p([Al][Partname] Ph [Attr:Md] [of:Ph] [(Partname clause)] ) |
     P([Al][Agentive:] Ph [Attr:Md] [of:Ph] [(Partname clause)] ) |
     p(Partname clause) | P([Agentive:] clause) /* 記述側面 属性機能 */
CRe ::=  c([Circname][Prep] Ph [Attr:Md] [of:Ph] [(Partname clause)] )
       c( Circname[Prep] clause )  /* 記述要素 */
id( Env: Ph )   /* 識別的記述側面: 識別子 */
us( Env: Cmd )  /* 識別的記述側面: 使用法 */
Env ::=  e( noun, Value ) | Env  Env /*  環境的記述側面    */
```

図 1: Basic Components of Descriptive Aspects

```
P ::= Pe | Pa | Po | Pc
Pe ::= a Primitive, a Descriptive Element or a Descriptive Aspect
Pa ::= Pe Pe |  Pa Pe  |  Pe Pa
Po ::= { Pe | Pe }|{ Po | Pe }|{ Pe | Po }
Pc ::= {( Pa ) | Pe } | ( Po ) Pe | { Pe | ( Pa )} | Pe ( Po )
       | ( Pc ) ( Pc ) | {( Pc ) | ( Pc )}
```

図 2: Compound Components of Descriptive Aspects, Elements or Primitives

記述側面の構成要素は図1のような成分を持ち、省略可能である。これらが図2のようにandとorの演算で合成される。

3 入力解析

ここで目的とする入力解析は、前章で示された構造を持つ記述表現法で表された記述名から必要な単語を取り出し、対応する管理テーブルに書き込む処理を行なうことにある。

入力解析には、比較的広く用いられているyaccとlexを用いた[S88]。

3.1 字句解析と構文解析の問題点

入力文字列で、意味を持つ最小単位としてのトークン(token)をどう決めるかが問題となる。

*Syntax analysis of the simply structured descriptive names
Fumiko Kouda
Department of Information and Communication Engineering,
Faculty of Engineering,
University of Tokyo
3-1 Hongo 7-chome, Bunkyo-ku, Tokyo 113, Japan

記述名表現では、意味の区切り記号で構造が分かるようになっている。プリミティブな区切り記号は単語の前だけに置かれるが、複合構造の区切り記号は左括弧と右括弧に分かれて置かれる。 and と or の演算も同様に括弧が用いられる。意味の区切り記号から記述の構造が分かるので、意味の区切り記号にトークンを割り当てることは妥当と考えられる。単語も区別する必要があるので、単語には一つのトークンを割り当てることにする。

ところが、トークンは単に形の違いを見るものなので、括弧の対応関係等、記述構造の意味の関係は何も示されない。そのため、トークンだけでは、記述構造の関係が掴めなくなる。

また、yacc は LR(1) 文法 (上向き構文解析) で、先読みが一つなので、入力単語の状態を細かく区分けしないと、yacc の構文解析で、トークンの並びに duplicate が生ずることも起こる。

上述の問題を避けるため、以下のような解決策を採用した。

3.2 字句解析と構文解析の方針

lex と yacc は構文 (Syntax) 解析に限定し、トークンの種類を増やさない。この方針で、トークンは、意味の区切り記号、構造の区切り記号、明らかに品詞の違いが分かる単語、それぞれに表 1のように割り振った。yacc では、繰り返し構造の中でトークン一つだけを選択する形式にし、先読み二重化問題が生じないようにした。

これだけでは、記述表現の解析はできない。トークン毎に分けられた yacc の各処理で、記述構造の意味に対応したステートと記述の複合構造に対応したスタックを、新たに導入した。

ステートは、表す意味に応じて図 3のように分けられる。異なる記述要素にも同じ形態の記述成分が含まれ得るので、yacc の中で異なるトークンの処理に同じステートが使用され得る。例えば、トークン CRM とトークン VRBS には、ステート EndofSF, FALSE, AndOr, PatE, CrE, StE, St, cratcla, crecl, 等が共通に用いられる。

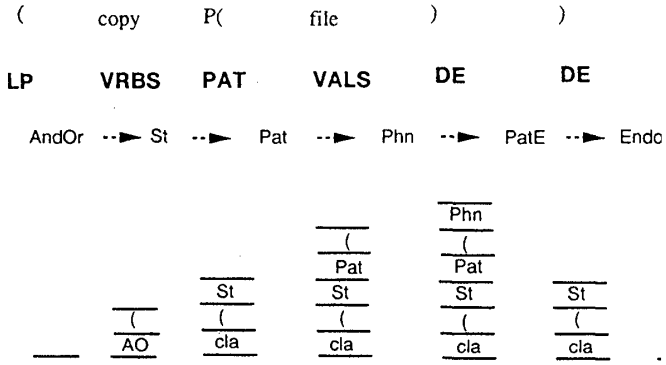
スタックは、トークンの入力時に記述の複合構造を記憶しておくものとして、以下のように用いた。

プリミティブな意味の区切り記号に対応するトークンが来た時は、それに対するステートをスタックに 1段だけ積む。これに対応する単語が終わった時に、スタックから 1段取り出す。

複合構造の意味の区切り記号の場合、「記号 (」に対するトークンには、スタックを 2段積み、「)」に対するトークンには、スタックから 2段取り出す。再帰構造など、一部の処理では、スタックを一段ずつ積んで処理を進めることがあるので、括弧に係わるスタックは 2段で一組とした。

4 例題

(copy p(file)) の場合についてスタックとステートの様子を示す。



5 おわりに

対象の動作や属性を記述できる簡単な構造を持つ記述名表現の入力解析の構成法について述べた。入力解析には、構文解析として、yacc と lex を用いた。入力された記述の意味構造の解析には、トークンだけでは不十分なので、スタックとステートを新たに、導入した。このことにより、少ないトークン数で、複合構造を持つ記述名の入力解析が可能となった。

参考文献

[S88] SUN Programming Utilities and Libraries lex, yacc pp.205 - 267, 1988

[F94] F. Kouda A new notation of user-friendly descriptive names, pp.493-498, ICON-9, 1994

```
(1) 未定状態 ( undefined )
  FALSE  AndOr
(2) その状態に入った ( created )
 a) 記述側面に
  Pat Ps Cr Idt Usg Envm  cla EvId EvUs
 b) 記述要素に
  Pat Ps St Adj ofp pht phop phv CrAdj Crofp
  crpht crphop crphv  Al prep cmd Idw
 c) 再帰状態に
   recla  atcla  crecl  cratcla
 d) これまでの入力値からだけでは入力されたものが
 一意に定まらない。(次の入力で決まる)
   Rl  phpcl  crphrec  crole
(3) その状態が終わった。 ( terminated )
 a) 記述側面が
   EndofSF     EvIE  EvUE
 b) 記述要素が
   StE  PatE  CrE  Patphr  crelm
```

図 3: State の種類

表 1: Input characters and corresponding tokens for the parsi

記述側面	P(	p(	id(	us(	e(	(	)
Token	PSUB	PAT	ID	US	EVT	LP	DE
記述要素	Attr:	of:	冠詞	Partname	Circname	c(	前置詞
Token	ATR	OFP	AL	PROLE	CROLE	CRM	PRP
値, 区切	動詞	Type	Value	操作	:	|	...
Token	VRBS	TPS	VALS	OPSG	CL	OR	END