ＵＮＩＸ日本語インタフェースシステムの構築

７Ｊ－７　―意味構造表現とコマンド結合―

矢野敏之　西島恵介　藤田米春
大分大学工学部

１．はじめに

現在，日本語で入力されたユーザの要求からUNIXコマンド列を生成する，UNIX日本語インタフェースシステムを構築している．

ユーザの要求は，構文・意味解析により，このシステムで処理しやすい形式である意味構造表現に変換される．そして，システム内のコマンド辞書と生成プランを用いながらコマンドを生成する．

ユーザの要求は計算機からみて複雑なものが多いため，複数のUNIXコマンドを組み合わせて実現しなければならない場合が必ず生ずる．これは従来の自然言語インタフェースやアイコン操作によるインタフェースでは実現が困難であった機能であり，それらを実現するために効率的な意味構造表現の定義と，awk や shell script を用いたコマンド加工などの自動プログラミング手法の応用が必要である．本稿ではこれらについて検討する．

２．意味構造表現

意味構造表現は，本システム内の統一した内部表現であり，ユーザ要求の記述や後述するコマンド辞書を記述するために用いられる．構成要素は，状態記述述語，動作手続き，関数，制約語，ファイル構造記述語である．それぞれの構成要素の表現例を図１に示す．

例として，「名前がxではじまるファイルの属性を標準出力に表示する」の意味構造表現は，動作手続きout，関数atr，ファイル構造記述語file，制約語beginを用いて，

```
out(atr(file([name:begin(x)])),stdout)
```

と記述できる．

３．コマンド辞書

コマンドの動作を表現するためにコマンド辞書項目を記述する．UNIXコマンドはその引数の違いにより大きく機能が異なる場合が多いので機能一つひとつについてコマンド辞書を用意する．

コマンド辞書は項目として，前提条件，入出力構造，コマンド動作，コマンド構文などをもつ．実際には，ユーザ要求とコマンド動作がマッチするとき，そのコマンドが候補コマンドとして選択され，前提条件も満たす場合，正式に生成コマンドとなる．

「ファイルの内容を出力する」UNIXコマンド'cat A'のコマンド辞書項目を例としてあげる．ファイルの内容を出力するにはファイルが存在しなければならないので，前提条件は，

```
exist(file[name:A])
```

ファイルの内容を出力するという動作を動作手続きを用いて記述すると，コマンド動作は，

```
out(content(file([name:A])),stdout)
```

となる．コマンド辞書の記述例を図２に示す．

４．生成プラン

ユーザ要求を満たすコマンドの生成方法があらかじめわかっている場合，コマンド列の生成プランとして以下の項目を記述する．

■マッチングパターン：その生成プランを選択するためにマッチングをとる意味構造表現パターン．
■前提条件：生成プランを選択するための前提条件．
■処理：パターンに応じたコマンド生成処理．
■変換パターン：生成プランを選択した結果，ここに記述された意味構造表現パターンに変換される．
■生成パターン：その生成プランを選択した場合に生成するUNIXコマンドの構文．

「処理」の項目においてコマンドを生成するのに必要な知識，規則を記述する．sortコマンドを例にとると，ソートのオーダやキーとする対象に応じてコマンドのオプションを計算する規則を記述しておく．生成プランの記述例を図３に示す．

５．コマンド結合戦略

構文・意味解析することによってユーザの要求文は意味構造表現に変換され，それらをもとにコマンド検索をする．コマンド検索の方法として以下の３つの戦略を用いる．

(1)コマンド辞書との完全マッチング

ユーザ要求の意味構造表現とコマンド辞書のコマ

Construction of a Japanese Interface for UNIX
Toshiyuki Yano, Keisuke Nishijima,
Yoneharu Fujita
Oita University

ンド動作項目が完全にマッチングし，さらに，システムが前提条件を満たす場合，コマンド構文項目をもとに実際のＵＮＩＸコマンド列を生成する．

(2)生成プランの選択

ユーザ要求の意味構造表現が生成プランのマッチングパターンに適合する場合，その生成プランを選択する．そして，処理項目に記述されている規則をもとにコマンドを生成し，変換パターンにしたがって変換された意味構造表現についてコマンド検索を繰り返す．

(3)入出力構造の組み合わせ

上記(1),(2)に失敗した場合，コマンド辞書の入出力構造の記述や生成プランの変換パターンを組み合わせ，ユーザの要求を満たすことができるコマンド列を生成する．

これらの戦略を組み合わせて用いることによりユーザの要求を満たすUNIXコマンド列を生成することができる．管理者がコマンド辞書または生成プランを必要なとき必要なだけ追加することにより処理の幅が広がると考えられる．

６．コマンド列生成例

ユーザが入力した要求文「ファイルaの内容を逆順にソートして見せて下さい」は構文・意味解析により意味構造表現，

out(sort(content(file[name:a]),zyx,''),stdout)

に変換される．これをコマンド結合戦略を用いてコマンド検索を行う．まず，５の(1)の戦略を実行するがこのような動作をするコマンドが辞書に登録されていないので失敗する．よって，(2)の戦略を実行する．ここで，生成プランsort_planのマッチングパターンとマッチするのでこのプランを選択し，オプションの計算を行いsortコマンドを生成する．さらに，生成プランの変換パターンによって

out(content(file[name:a]),stdout)

のように変換され（図３参照），これについてコマンド検索を行う．検索は(1)の戦略によりコマンド辞書cat_1に完全マッチしてcatコマンドが生成される（図２参照）．これらの処理の結果UNIXコマンド列 cat a | sort -r が生成される．

７．おわりに

本稿ではユーザ要求の意味構造表現から生成プラン，コマンド辞書を組み合わせながらUNIXコマンド列を生成する方式について述べた．この方式により，従来のインタフェースでは実現が困難であった複雑な要求を処理することが可能となった．現在は，ユーザ要求の対象をファイル操作に限定しているので，その他のユーザ要求に対応していく場合の戦略の検討が今後必要である．

参考文献

[1]久保他,"UNIX日本語インタフェースシステムの構築-コマンド辞書の記述とコマンド選択-",平成4年度電関九支連大,No.1207

[2]西島他,"UNIX日本語インタフェースシステムの構築-要求文からの意味構造の抽出-",平成4年度電関九支連大,No.1208

状態記述述語	exist(<object>) not_exist(<object>)
動作手続き	out(<object>,<place>) make(<object>) delete(<object>) change(<source>,<destination>)
関数	sort(<object>,<order>,<key>) number_of_line(<object>)
制約語	begin(<regular exp.>) greater_than(<regular exp.>)
ファイル構造記述語	file([name:A],[place:P])

図１　意味構造表現

```
command(
 name:cat_1
 precondition:exist(file[name:A])
 input_structure:
 action:out(content(file([name:A])),stdout)
 output_structure:text([*])
 syntax:cat A
)
```

図２　コマンド辞書の記述例

```
plan(
 name:sort_plan
 m_pattern:out(sort(X,Order,Key),stdout)
 precondition:
 process:order(Order,Op1)
         key(Key,Op2)
 tra_pattern:out(X,stdout)
 gen_pattern: | sort Op1 Op2
)
```

図３　生成プランの記述例