

対話型UNIX支援システムの機能と構成

5S-2

樋口 英幸

伊藤 昭

郵政省通信総合研究所

1 はじめに

計算機のユーザインターフェースを改善する研究は、いろいろなアプローチで進められてきている。これらの流れのうち、大きなものとしては、日常生活とのアナログをもとに視覚的なコマンドインタプリタを実現するものと、知的ヘルプ機能を備えたコマンドインタプリタを実現するものがあげられる。前者と後者は互いに対立する概念ではないので、その双方を取り入れることも可能ではある。しかし、現状を見ると、前者は一般的にユーザに分かりやすいものの、メニューなどにより選択肢の提示を行うことで、コマンド言語の表現力を制限している。一方、後者では表現力は大きいものの、ユーザの意図の抽出が難しいことが問題点となっている。我々は、後者のアプローチを取り、UNIXの熟練者が初心者からの相談を受けたときの対処法をモデルとした UNIX-wizard[1]を提案した。本報告では、[1]の持つ具体的機能及び構成法を述べる。

2 対話のモデリング

ユーザと計算機の相互作用のあり方については色々議論が分かれるものの、一旦計算機の前に座ったユーザは、計算機をあたかも一個の独立した人格のように擬人化して扱い始めることが多い。この環境の中にヘルプシステムを導入する場合、UC[2]などでは、対象となる計算機とヘルプシステムは独立した存在という扱いになっている。しかし、ユーザ側からみれば、同じ計算機の上のプログラムに対して、「なぜ、前からコマンド言語を通じて計算機と対話していたのに、再びヘルプシステムに同じことを説明し直さなければならないのか」と考えるのは極く自然である。したがって本システムでは、コマンドインタプリタとヘルプシステムがコンテキストを共有するアプローチをとり、コマンド入力履歴から推論可能なことをユーザに再び入力させることのない「協調的対応」をすることを主眼としている。

3 本システムの機能

本システムには、大きく分けて次の3つの機能がある。

1) UNIXコマンドに対する一般的質問応答

"How can I delete a file?" といったコマンドに対する一般的質問に答える機能である。

2) コマンド入出力のコンテキストを持込んだ質問応答

これは、ユーザの質問中における代名詞の指示対象や、省略された背景情報をコマンド入出力から補って応答する機能である。例えば、

```
% rm ~/tmp
rm: /usr/foo/tmp is a directory
% #Why wasn't it deleted? ("#"は対話開始記号)
> "rm" can't remove directories without "-r"
options.
```

という動作をする。"it"は"delete"があることから直前のコマンドの引数であることがわかる。応答部分は、[3]のsuggestive indirect responseの観点で見ると、"without"以下の部分を省いた場合に比べ協調的である。しかし、この段階で、どうやったら~/tmpが消えるか、どんな副作用が起きるか、といったことには直接言及しない。一問一答を短くすることも「協調的動作」の要素である。もし、次のユーザの質問が1)に属するものならば、それに対応する応答がなされる。また、"How can I delete it?"とユーザが質問すれば、システムは関係するディレクトリを実際に調べて応答する。

3) コマンド入力列の一部の挙動がユーザの考えと違っていたことで起こる質問応答

これは、例えばユーザの入力ミス、コマンド動作の誤解、勘違いなどによって起こる。例えば、

```
% vi foo.c (A)
% cd (B)
% cp foo.c bar.c (C)
cp: foo.c: No such file or directory
% #Isn't foo.c here?
```

というような場合である。この対話部分のみでは些末な例であるが、(A), (B), (C)の間に他のコマンドが入った場合、このようなことは実際に起こり得る。本システムでは入力履歴をたどることでユーザの質問の意図を検出す

る。

4 本システムの構成

本システムの構成は図1のようになっている。ここでは、構成要素のいくつかについて述べる。

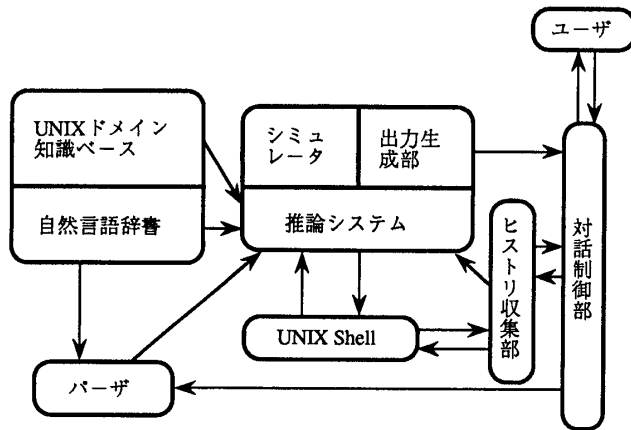


図1 UNIX-wizardのシステム構成概念図

・ コマンド入出力履歴収集部

コマンド入出力履歴収集部では、ユーザのコマンド入力とコマンドの実行結果の出力の両方を取り込む。収集された入力コマンドは、解析時には2種類の表現レベルで扱われる。一つはコマンド入力そのものである。もう一つはSchankのCD表現([4])にあたり、計算機における操作・概念を基本動作で表現したものである。ファイルシステム領域に対しては表1に示した5つの基本動作を用いる。異なるコマンド列でも同じ効果をもたらす場合があるため、ユーザにとって同等の表現は同一の表現となるようにする。これによって、ユーザとの意図のレベルでの対話を可能にする。

・ UNIXドメイン知識部

UNIXドメイン知識には表2に示した4つの種類の知識が必要である。それぞれの知識はフレーム形式で表現される。CLOS[5]を記述言語としているので、知識の階層化は自然な形で導入できる。ヒューリスティックスに属する知識には、ユーザの意図の検出をより効果的にするための計算機操作の標準的手順(Schankのスクリプト[4]に相当)や、OSの構成とは直接の関係はないものの、UNIX独特の習慣として必要な知識が含まれる。

・ 推論システム部

推論システム部は基本的にプロダクションシステムで構成される。また、副作用を与えずにUNIXの実際の状態(ファイルのアクセス制御など)を把握できる場合は、適当なコマ

ンドを発して調べる。副作用がある場合は内部シミュレータを用いて何が起きるかを予測する。

表1 ファイルドメインにおける計算機の基本動作

CREATE	ファイル、ディレクトリの作成、画面への表示、プリンタへの出力など
DELETE	ファイル、ディレクトリなどを消去
MODIFY	ファイル、ディレクトリの内容や属性の変更
CHECK	ファイル、ディレクトリなどの状態の問合せ

表2 UNIXドメイン知識のデータの類別

システム構成に対する知識(例 ディレクトリ)
コマンドに対する知識
ヒューリスティックス(標準的作業手順等も含む)
自然言語用辞書

5 おわりに

知的UNIXヘルプシステムUNIX-wizardについてその機能と構成について述べた。現在この設計にもとづいてプロトタイプシステムを実装中である。

参考文献

- [1] 伊藤, 樋口(1990):対話型UNIX支援システム, '90信学会春全大, D-208, p. 209.
- [2] R. Wilensky, Y. Arens and D. chin(1984): Talking to UNIX in English: An Overview of UC, *CACM*, Vol.127, No. 6, pp. 574-592.
- [3] S. J. Kaplan(1983):Cooperative Responses from a Portable Natural Language Database Query System, in *Computational Models of Discourse*, ed. M. Brady and R. Berwick, The MIT Press, pp. 167-208.
- [4] R. C. Schank and R. P. Ableson(1977): *Scripts, plans, goals, and understanding*, Lawrence Erlbaum Associates.
- [5] D. G. Bobrow, L. G. DeMichiel, R. P. Gabriel, S. E. Keene, G. Kiczales and D. A. Moon(1988): *X3J13 standards committee documents 88-002R*.