コンピュータと教育　15－4
(1991. 4. 26)

ＵＮＩＸコマンドの学習を支援するシステム

赤井　隆志　　伊丹　誠　　伊藤　紘二

東京理科大学基礎工学部

あらまし　ＵＮＩＸは便利なＯＳとして広く普及し今日に至っているが、豊富な機能のために初心者にはとっかかりにくいものとなっている。そこで、水準の異なる学習者が独習をする際に、これを支援するようなシステムの開発が望まれる。本システムでは手始めに初心者を対象とし、コマンドの使用例を検索させることにより、ＵＮＩＸコマンドの使用法をガイドすることを目指すシステムを構築した。支援の概要としてはまず、ユーザにプリミティブのメニュから問い合わせとしてのゴールを入力させる。システムは、それとマッチする手がかり表現を検索し、対応したコマンド使用例を提示、ユーザに選ばせる。ユーザは、そのコマンド列の要素の学習や、それを説明するシェルスクリプトの実行による学習を行うことができる。

A Computer Assistant for Learning UNIX command

Takashi AKAI　　Makoto ITAMI　　Kohji ITOH

Department of Applied Electronics, Science University of Tokyo

Yamazaki 2641, Noda city, CHIBA 278

Abstract　UNIX operating system is widely used because of its capability and versatility. It is however difficult for a begginer to use UNIX or to find out what it is and how it works. This motivated our work on a computer assistant to help for self learning of UNIX command. The serport describes the prototype we developed to guide beginners through examples in mastering UNIX. To each of the examples embedded is an indexical description consisting of expressions from a set of primitives. When the user composes a goal description using the same set of primitives, the system searches for such inderical descriptions as match the goal description. He/she, selectingan example from those with the retrieved indexical discriptions, is assisted to study on command line elements or to see how the example works on the displayed execution of a shellscript.

1. はじめに

今日、個人が変化の激しい社会の中で生活していくにあたって、新しい知識や技量を短期間で多く獲得していく必要性が生じてきた。ただし、この知識とは学習者にとって当然目的、興味、水準が異なったものとなっている。

特に情報処理教育においては、学習者に演習を課することが重要であるが学習者の水準が多様であるため、個人指導を行なうのが望ましい。しかし、教員には学習者を個人的に指導するだけの余裕がないのが普通であり従って、水準の異なる学習者が独習するときに、これを支援できるようなシステム環境が望まれる。本稿では講義形式による一応の知識は得ているという前提の下に、ＵＮＩＸコマンドの使用法という分野におけるこのようなシステム構築の試みについて報告する。

2. 構想

2．1 ＵＮＩＸコマンド学習支援システム

ＵＮＩＸは便利なＯＳとして広く普及し今日に至っているが、豊富な機能の故にユーザにとって決して簡単なものとは言えない。本システムでは、ビギナあるいはマスタユーザを対象とし、コマンドの使用例を検索させることにより、ＵＮＩＸコマンドの使用法をガイドすることを目指すものである。

すでにビギナの問い合わせに対して、ＵＮＩＸのコマンドの使い方を答えるシステムとしてＵＣ[9]がある。これに対して我々は「事例の説明によって問題解決方略を獲得させる」事を学習の方法とするシステムの構築を目標としており、今回はそのようなシステムのベースとなる仕組みを実現した。

実際には、ユーザが行ないたい操作を実現するコマンドの書き方を調べるという問題に対して、コマンドの使用例、実際にそれを使用したシェルスクリプトの実行、関連コマンドを検索、提示し、そういった過程を繰り返していくことによってＵＮＩＸコマンドの学習を支援していくシステムとなっている。また、単にＵＮＩＸコマンドを学習するだけではなく、このシステムの行なうコマンドの分類を通して、ＵＮＩＸの考え方、概念をつかむといった狙いもある。

2．2 コマンド使用例の検索における知識ベースの利用

通常ユーザが、何もない状態から検索の要求を提示する場合、不完全なものしか提示できない場合が多い。その原因として、

・理解が断片的である。
・用語を知らない。
・ユーザによって、検索しようとしている知識の理解の仕方が違う。

などが挙げられる。一方、異なる使用例の間には、課題や構造の上で関連があり、この関連を辿ることが、汎用性のある方略と概念を摑む上で重要である。

情報単位の間の関連を表現する方法としてハイパーメディアで見られるように、情報単位をノードとし、各ノードの間にリンクを張る方法がある。このようなリンクをたどって必要な情報を引き出すことも可能となる。ただし、この方法では以下のような欠点が含まれている。

・不完全な検索要求に応ずる仕組みがない。
・ノード数が膨大になると、リンクをたどって情報を捜すことが難しい。
・ノード間の関係をリンクで表わそうとすると、組合せ的爆発は避けられない。
・２つのノードの間の関係のみに基づいて作られるリンクでは、ノードがグループとしてもっている様々な型の階層的並びに構造的関係を抑えることができない。
・検索の要となるノードの名称が分からない。
・これらのことより、ユーザの自由な発想が奪われる結果となる。

そこでこの問題を解決するために、ユーザの水準に応じて、説明を要しない基本的な概念－プリミティブと称する－を組み合わせた手がかり表現を使用例の各々の背後に置く方法が考えられる。この方法を用いると、プリミティブは多くの概念に共有されるため、異なる使用例どうしをいろいろな視点から結びつける手段となる。ユーザに同じプリミティブを使って検索要求を入力させることによって検索が容易になる。プリミティブは数が少ないのでこのような基準化が容易であるといえる。ただし、そのようなプリミティブを直接用いて検索要求を表現しきることは難しい。

そこで、ユーザとプリミティブによる表現と

のインタフェースをとってやることや、複数の候補となる使用例を取り上げ、再びユーザに問い合わせるといった方法がある。前者は、今回のシステムの場合、アニメーションを用いる方法や自然言語を用いる方法が考えられる。後者の場合、今回のシステムでは、（３．２．１）で示すように実現した。

一般に、できるだけプリミティブなレベルの概念を用いて知識を表現すると知識の利用に汎用性をもたせることが可能となる。そうして初めて、様々な問題に対して、柔軟な知識適用が可能となる。また、上述のような基準によって知識の追加や改訂も容易に行なうことができるという利点がある。手がかり表現の変更は、他の表現になんらの変更も必要としない。

２．３　ユーザ・インタフェース

システムが検索を行なう場合、その働きは「検索要求内のプリミティブを含む（検索要求内のプリミティブを完全にマッチする）知識を捜し出してくる」ということができる。ただし、ここでの「検索要求」とはユーザが自由に設定するものである。そのため、当然曖昧な表現などが含まれた検索要求になる可能性がある。ここで、そのような場合でも知識を検索するようにすると、設定いかんではほとんど知識を引いてくることができない場合や、逆に非常に多すぎる知識を引いてきてしまうことがある。その結果、知識検索に負担がかかり、組合せ的爆発を起こす原因となる。その為、適切なユーザとのインタフェースを実現する必要がある。今回の場合、ＵＮＩＸコマンドの中での表現に限って手がかりプリミティブを作っているため、かなり限定されたものとなっている。つまり、アニメーションやメニューなどを使って学習者と対話しながらシステム内のオブジェクトに対し適当なメッセージを送れるようなインタフェースを、インタフェース・オブジェクトとして実現できれば無理なくシステムに組み込むことができると考えられる。特にアニメーションなどを用いることによって、ユーザに視覚的に訴えることができ、非常に有用なものになると思われる。

また、インタフェースをユーザと知識との受渡しという点を注目して考えた場合、自然言語を用いることが適していると考えられる。自然言語を用いると、ユーザの表現の自由度が増すであろう。ただし、その中から目的の情報を的確につかむには、構文解析などができた上で学習者との対話が必要不可欠なものとなると考えられる。

さらに、水準の違ったユーザに対してどのようなインタフェースを使ったらいいかという問題がある。これは、ビギナ用とマスタ用のインタフェース２種類を持っていることで解決できる。つまりそれぞれ、日常表現手がかりと体型的手がかりを持つのである。このシステムでの例を挙げると、ビギナは、ＵＮＩＸコマンドが実行する情報操作を日常表現で比喩的に表わしたもの－メタファによる表現と称する－を使い、マスタは本来のコマンド体型の原理に沿って表現したものを使う。ビギナには、はじめビギナ用の日常表現手がかりで問い合わせを行ない、検索されたコマンド使用例の体型的手がかりの一部を問い合わせとして使って類比な例をたどり、これを日常表現手がかりで解読することにより、漸次体型を習得してゆくことができよう。このことによって違った視点、つまり本来の視点からみることによって、学習の手助けになると思われる。

３．　実現

３．１　支援の概要

本システムは、知識となる部分をPrologで構築し、全体の制御をCを用いて行なった。

Prologは、マッチング機構などデータベースの構築に適している。全体のシステムの流れとしては、図１に示すようになっている。

３．２　システムの構成

３．２．１　メタファゴールの入力

本来、前で述べたようにアニメーションや自然言語を用いることが望ましいと思われるが、今回試作したものは内部表現がむき出しになった初歩的なものに留まっている。

このシステムでは、実際に知識表現の中で使用しているすべての手がかりプリミティブを表示し、その下にユーザが選んだプリミティブを入力するエディタがある形となっている。ユーザは、目的のコマンドの動作を考え、上に示さ

れている手がかりプリミティブを組合せて問い合わせ表現を作る形になる。手がかりプリミティブは、４種類に分かれていて、それぞれ precond,action,add,delete となっている。precond は初期状態，add は追加リスト，delete は削除リストに対応しており、action がそのコマンドの動作を示したものとなっている。ここでのプリミティブはＵＮＩＸが行なう情報操作を日常的な表現で表わしている。前で述べたよう、ユーザは完全なものを入力する必要はなく、考えられるプリミティブのみ入力するだけでよい。

３．２．２　コマンド列使用例の検索・提示・ユーザによる選択

手がかりプリミティブを得たシステムは、prologで書かれた知識ベースへ行き、マッチングを行なう。現在のシステムではコマンド使用例の知識が１００程度のため、ユーザの提示したプリミティブを含む知識をすべて取ってきている。しかし、将来的に知識が増えた場合の事を考えて、マッチングした割合の多いものを取ってくるというようにした方が適切であると思われる。

ここで、ユーザの提示したプリミティブをすべて満たす知識がなかった場合は、条件を緩めて（一部のプリミティブを削って）探索を行なっている。

システムが提示した候補のメニュからユーザが選択することにより、ここで初めて、ユーザの要求に見合ったコマンド使用例が明らかになり、ユーザが学ぶべき問題が提示されたこととなる。

３．２．３　分析学習

ＵＮＩＸコマンドを考えた場合、いわゆるコマンドと言われるものや、そのコマンドを制御する働きを持っているリダイレクション、パイプライン、メタキャラクタ、バックグラウンド・ジョブ、コマンドセパレータなどのものがある。特に、後者を理解することはＵＮＩＸシステムの概念を理解するのに非常に有効であると考えられる。

このシステムでは、コマンド列を一つ一つの要素に分解し、ユーザに分からないところを選ばせる。そして同類の学習へ行き、同じ要素を持つコマンド列をメニュとして提示し、コマンド列使用例メニュからの学習を行なう形にしてある。その方法として、同じ構造の例を複数示すことによってそのコマンドの働きを理解させようとしている。つまり副問題ここで生成し、解決する仕組みになっている。ただし、副問題を生成する必要のないユーザは次の過程に進むことができるようになっている。

３．２．４　シェルスクリプト実行表示からの学習

実際の体験を通すと知識となりやすいことはすでに多くの例で報告されている。シェルスクリプトをユーザに見せて実際に実行させ（図３）、そのコマンド列を実行させると何が起こるのかを体験させ、どういう意味を持ったコマンドなのかを分からせることがここでの目的である。またそのときに、シェルスクリプトの実行結果の表示の中に分からないコマンドがあった場合は副問題の解決として、同類の学習を行なうことになる。ここでは、シェルスクリプトを動かすための専用のディレクトリを用意しておき何回シェルスクリプトを動かしても動かす前の状態を保つようにしてある。そのディレクトリの構造はユーザに分かるようシェルスクリプトの実行結果の中で示され、自分がどのような状態でそのコマンドを行なっているかが実行しながら分かるようになっている。またそのことによって、コマンドの動作の理解を助ける役目を果たす。

３．３　手がかり表現と知識ベース

手がかり表現は、'cat < f1 > f2' の例を示すと図２に示すような構成になっている。検索はprologのマッチングを用いている。

この例では、６つの手がかりプリミティブで構成されている。①の index ではコマンド列とその id が示されている。②以降の index の中での第一引き数は、インタフェースで述べた precond,action,add,delete に対応している。第二引き数は手がかりプリミティブで、その対象となるものを変数で表わしている。第三引き数でその対象となるものの対応を示している。つまり、ここで示された Z1 から Z4 はそれぞ

れ同じものでなくてはならないように定義されている。そして最後の引き数では、id が示されている。

それぞれの意味として、

①事前に入れ物１（ｆｉｌｅ１）が存在する。
②入れ物２（ｆｉｌｅ２）をつくる。
③入れ物２が存在する。
④入れ物１の内容を
⑤入れ物２へ移す。
⑥最後に入れ物２ができている。

となっている。

また、この検索を行なうときに知識が増えた場合に汎化、特化といった考え方を取り入れる必要がある。ＵＮＩＸでは例えば、ファイルなどに利用する事ができる。ファイルで行えるコマンドは、文章ファイルでも行えるし、もちろんプログラムファイルでも行なうことができる。ただし、文章ファイルやプログラムファイルでできても、ファイルでできるとは限らない。このような考え方を知識ベースとして表現し、これを用いることによって、ユーザの問い合わせ表現の曖昧さを解消する手段となる。

4．　結論

まず学習者の問題解決という点からみた場合、副問題の生成が挙げられる。このシステムでは、スクリプト実行表示からの学習と分析学習の部分でそれを行なっている。特にこのシステムでは、学習支援の対象がＵＮＩＸの概念もしくはＵＮＩＸコマンドであったため、コマンド列を分解して構造と要素とに分けたことは非常に有効なものであった。ただし、スクリプト実行表示からの学習での副問題に関しては、まだ改良の余地がある。

このとき、学習者の記憶容量が問題となってくると思われる。そのため解決過程を制御するシステムが必要となってくるであろう。それは複数の副問題が生じた場合、まだ解決していない副問題と解決した副問題とを制御するものである。さらにそれをユーザに分かるように、視覚的に訴える機能が必要である。

また解決過程の履歴を随時提示する仕組みによって、ユーザは迷ったとき、戻るべき所を知ることができる。

経験という視点からみたときのシェルスクリプトも、適当な働きをしていたと思われる。今回は実現していないが、シェルスクリプトとともにテキストによる説明を持たせる方法もよいと思われる。コマンドによっては当然シェルスクリプトだけでは適切ではないものもある。そのような点の解決になると思われる。

手がかり表現であるが、プリミティブな概念を用いたため柔軟な探索が行えることが実証された。ただし、使用例を増やしていく場合、知識のネットワーク化ということを実現しないと、探索に負担がかかるようになる。探索が高速化されればプリミティブの工夫によって、より深いＵＮＩＸの概念が表現できるであろう。

最後に、インタフェースの部分であるが、このシステムではまだ多くの問題点を残している。解決方法としては前で述べた通りである。インタフェースの解決は今後の課題である。

参考文献

[1]伊藤 紘二，”知識処理”，電子情報通信会誌，71，4，pp.359-366，1988

[2]Etienne Wenger，"ARTIFICIAL INTELLIGENCE and TUTORING SYSTEMS"，Chapter 11，13，Morgan Kaufmann

[3]伊藤 紘二，”知的ＣＡＩシステム探訪”，情報処理，29，11，pp.1283-1293，1988

[4]木内 伊都子，畠山 敦，大木 優，藤澤 浩道，”知的検索を目指したＣｏｎｃｅｐｔ Ｂｒｏｗｓｅｒ”，情報学基礎，13，4，1989

[5]鈴木 宏昭，”人間における知識の利用と獲得”，ＣＡＩ学会－「夏の学校」－論文集

[6]伊藤 紘二，”パーソナル知識ファイルシステム”，電子情報通信学会，研究会技術報告，ET88，1，pp11-16，1988

[7]鈴木 昭二，伊藤 紘二，”学習者の問題解決を支援する知識ベースシステム”，情報処理学会知識工学と人工知能研究会報告，56，11，pp.81-88

[8]Brian W.Kernighan，Rob Pike，石田 晴久訳，”ＵＮＩＸ プログラミング環境”，アスキー出版

[9]Robert Wilensky，Yigal Arens，and David Chin，"Talking to UNIX in English:An Overview of UC"，Communications of the ACM，27，6，1984

```
メタファからの学習：－
        メタファゴールを入力させ、使用例の＜メニュ＞を作る，
        コマンド使用例メニュからの学習（＜メニュ＞）．
コマンド使用例メニュからの学習（＜メニュ＞）：－
        ＜メニュ＞を提示し、＜使用例＞を選ばせる，
        コマンド使用例の学習（＜使用例＞），
         select｛コマンド使用例メニュからの学習（＜メニュ＞）；終了｝．
コマンド使用例の学習（＜使用例＞）：－
        select｛使ってみる学習（＜使用例＞）；分析学習（＜使用例＞）；メタファからの学習｝．
使ってみる学習（＜使用例＞）：－
        シェルスクリプト起動、＜表示＞を与える，
        select｛（スクリプト実行表示からの学習（＜表示＞）；終了｝．
スクリプト実行表示からの学習（＜表示＞）：－
        ＜表示＞の中の＜分析箇所＞を選ばせる，
        select｛スクリプト実行表示からの学習（＜表示＞）；終了｝．
分析学習（＜使用例＞）：－
        ＜使用例＞から＜分析箇所＞を選ばせる，
        ＜分析箇所＞が引き数なら、同類の学習（＜分析箇所＞），
        select｛分析学習（＜使用例＞）；終了｝．
同類の学習（＜分析箇所＞）：－
        ＜分析箇所＞と同じコマンドあるいは構造の使用例の＜メニュ＞を作る，
        コマンンド使用例メニュからの学習（＜メニュ＞）．
```

図１　ＵＮＩＸコマンドの学習の支援

```
index('cat < f1 > f2', 41).
index(precond, [box, Z1, box, Z1],          [[Z1, a1], [Z1, a1]], 41).   ①
index(action , [make, Z2, object, Z3],      [[Z2, a2], [Z3, a3]], 41).   ②
index(action , [box, Z3, box, Z3],          [[Z3, a3], [Z3, a3]], 41).   ③
index(action , [copycontent, Z4, from, Z1],[[Z4, a4], [Z1, a1]], 41).   ④
index(action , [copycontent, Z4, to, Z3],   [[Z4, a4], [Z3, a3]], 41).   ⑤
index(add    , [box, Z3, box, Z3],          [[Z3, a3], [Z3, a3]], 41).   ⑥
```

図２　手がかり表現の例（'cat < f1 > f2'）

```
#! /bin/csh
cd sample                  ．．．．専用ディレクトリへの移行
echo '¥n>ls -l'            ．．．．ユーザに何のコマンドを動かすかを示す
 set pay=$<                ．．．．ユーザはリタンキーを押すことによって次のコマンドに移る．
 ls -l
 echo '¥n>cat text1'
 set pay=$<
 cat text1
echo '¥n>cat < text1 > text2'
 set pay=$<
 cat < text1 > text2
echo '¥n>ls -l'
 set pay=$<
 ls -l
echo '¥n>cat text2'
 set pay=$<
 cat text2
rm text2                   ・・・・元の状態に戻す．
echo '¥n>'
 set pay=$<
```

図３　シェルスクリプトの例（'cat < f1 > f2'）