プラン認識におけるコマンド処理システム

3T－9

井上義史　永田守男
慶應義塾大学　理工学部

１．はじめに

コンピュータを使いやすいものにするには、その時々の意図をくみとりながら動くシステムが望ましい。しかし、一般的な意図理解に関する理論はその計算モデルも出来ていない状況である。[1]

そのような現状のなかで、本研究ではプラン認識の事例として、ＭＳ－ＤＯＳを使ったコマンド列を認識して対応する仕事をおこなうＵＮＩＸのコマンド列に変換するシステムを構築した。これを作成する過程と、このシステムを使った実験結果とを考察することにより、ここでの事例からプラン認識一般についてどのようなことが言えるかを考えた。

２．プラン認識とは

プラン認識の手法は、ある行動を観察して、プラン構成に関する観察者の知識と、その行為者がどのような目標をもっていそうかという観察者の信念とからプランを推定する方法である。行為間の関係を表現する一つの常識的な関係である「因果関係」を使って、既知の行為と状態からプランが構成できるが、プラン推定の方法は、そのようなプラン全体を探索範囲とする探索問題とみなせる。[1]-[3]

下図は、プラン認識プログラムを示したものである。

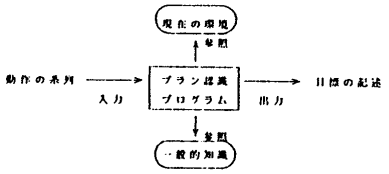

図　１　プラン認識プログラム

３．コマンド処理システム

３．１　システム構築にあたって

一般的なプラン認識では、観察者であるコンピュータに行為を入力することが難しいことが多い。ところが、ＯＳ上のコマンドは直接コンピュータに入力されるので、これから行為者のプランを認識することが可能であると考えられる。そこで、本システムでは、ＭＳ－ＤＯＳで入力したコマンド列から、そのユーザが何をしようとしているかのプランを推定し、その結果をＵＮＩＸのコマンド列で表現するものを考えた。つまり、先の図－１において動作の系列がＭＳ－ＤＯＳコマンド列で、目標の記述がＵＮＩＸコマンド列になっていると考えればよい。また、このような状況では、主にエディタを利用してのプログラム開発という行為にあるプランを考えればよいので、一般的なプラン認識よりも考えやすい。さらに、「コンピュータを使いやすいものにする」ためのプラン認識の事例の第一歩としてはふさわしいものである。

３．２　ＭＳ－ＤＯＳとＵＮＩＸのコマンド

ＭＳ－ＤＯＳとＵＮＩＸのコマンドを調べあげ、[4]-[7]その対応表を作ってみると、プラン認識まで行わなくとも変換できるものがあることがわかる。

しかし、その範囲でもＵＮＩＸの豊富なコマンドを活かして、より少ないコマンド列で答えることを試みた。そこで、考えたのが次の４点である。

```
① COPY f1 f2 , DEL f1
    -> cp f1 f2 , rm f1
    -> mv f1 f2
② DEL f1 ,RD f1
    -> rm f1 , rmdir f1
    -> rm -r f1
```

図－２　２対１対応

```
CD TJPK      -> pushd TJPK
 ...
 ...
CD YUSAYLOS -> pushd YUSAYLOS
 ...
CD TJPK      -> popd
(但し、CDとCDの間でカレントディレクトリの変更はないものとする。)
```

図－３　履歴機能

```
   ...                    ...
FL /c EXAM.FOR
MIFES EXAM.FOR    } fl /c EXAM.FOR || Lmmake -nv EXAM.FOR
FL /c EXAM.FOR
   ...                    ...
```

図－４　条件実行（再コンパイル）

```
   ...                    ...
FL /c EXAM1.FOR
MIFES EXAM2.FOR   } fl /c EXAM1.FOR
FL /c EXAM2.FOR   } Lmmake -nv EXAM2.FOR
FL EXAM1.OBJ EXAM2.OBJ  fl /c EXAM2.FOR
   ...                    RA fl EXAM1.OBJ EXAM2.OBJ
                          ...
```

図－５　条件実行（リンク）

図－２では、COPYとDELをmvに、DELとRDをrm -rに置き換えたわけである。この場合、ＭＳ－ＤＯＳ上で２つのコマンドが連続して現れていなければ必ずしも使えるわけではない。

ところが、離れている場合の方が使いがいのあるＵＮＩＸコマンドもある。図－３で示したpopd, pushdは、ヒストリ機能によってディレクトリをスタックに積んだり、そこから出すことができる。図－３のように、どこか別のディレクトリで仕事をして、再び元のディレクトリに戻るとき、これら２つのコマンドを使うと元のディレクトリを覚えていなくてもよい点で高級である。

３．３　プログラム開発におけるプラン

プログラム開発において、複数のコマンド列の組み合わせから、ユーザが作業中にどのような状態になっていて、そのプランがどのようになっているかわかる場合がある。例えば、FORTRANのプログラムを作っていて、図－４のように３つのコマンド列が現れたとする。EXAM.FORのプログラムをコンパイルし、そのプログラムを編集して再びコンパイルしている。ということは、初めのコンパイルが失敗し、デバッグするためにMIFESを立ち上げたと考えられる。そう考えると、ＵＮＩＸ上での例のようにOR条件演算子を使って、コンパイルが成功したら編集せず、失敗した場合に限ってエディタを立ち上げることにすればよいであろう。

A Command Management System based on Plan Recognition

Yosifumi INOUE, Morio NAGATA
KEIO University

同様に図－５は、EXAM1.FORのコンパイルが成功したので、EXAM2.FORというファイルを新しく編集してコンパイルした。そして最後に、この２つをリンクしたことがわかる。リンクはオブジェクトファイルがないと出来ないから、ＵＮＩＸ上での例のようにAND条件演算子を使って、コンパイルが成功したときに限ってリンクを行えばよいであろう。

３．４　プラン認識プログラムについて

図－２から図－５のような出力例を取り入れることによって、プランとしてよりふさわしいものになるが、これらの変換をもシステムに組み込むためには、実行過程における優先順位が必要になる。これは、ここまでの規準だけだと互いにコンフリクトを起こす恐れがあるからである。よって、図－１中の「プラン認識プログラム」の部分で、優先順位が与えられる。ただ、優先順位をどう決めても間違いにはならない。しかし、なるべく変換されたコマンド列の数を少なくしたいので、図－６のようにした。

図－６　優先順位

４．実験結果、検討

まず、本システムの制約条件は次の６項目である。

・ユーザは、ＭＳ－ＤＯＳはよく知っているが、ＵＮＩＸについては、loginとlogout程度しか知らない。
・コマンドは、ＭＳ－ＤＯＳ VER 3.3とＡＴ＆Ｔ社 Release 3.1のＣシェルに対応する。
・入力するコマンド列はタイプミス、及び文法上の誤りがないものとする。
・エディタを使う場合、ＭＳ－ＤＯＳではMIFES、ＵＮＩＸではEMACSに限る。
・使用可能なＭＳ－ＤＯＳの外部コマンドは、RENDIR,MORK,PRINTに限り、バッチコマンドは使わないものとする。
・ファイルやディレクトリを指定するとき、絶対パス名で表す。

実験はＮＥＣ　ＰＣ９８シリーズやＥＰＳＯＮ　ＰＣシリーズなどのパソコン上で行った。何人かの人たちにプロンプトが出た状態でいろいろ作業をしてもらった。システムがＬＩＳＰで書かれているため、作業が終わってからコマンド列を拾い出しＬＩＳＰを立ち上げ、関数MSDIXの引数にリストという形式で入れた。画面上では、図－７のような形で出力された。

```
(MSDIX '((CD ETC)(DIR)(CD GRP)(DIR)(CD ETC)(COPY GRP¥MAS
        GRP)(CD GRP)(DELETE MAS)(CD ETC)(REN MAS DOC))))
→ ((PUSHD ETC)(LS -ALF)(PUSHD GRP)(LS -ALF)(POPD)
   (CP GRP¥MAS GRP)(CD GRP)(RM MAS)(CD ETC)(MV MAS DOC))

(MSDIX '((PROMPT)))
→ ((UNSET PROMPT))

(MSDIX '((CC -C TEST1)(MIFES TEST1)(CC -C TEST1)
        (MIFES TEST2)(CC -C TEST2)(CC -O TEST1 TEST2)))
→ ((CC -C TEST1 ¥¥ TAMAGO -NW TEST1)(TAMAGO -NW TEST2)
   (CC -C TEST2 && CC -O TEST1 TEST2))
注）¥¥は||の意味として使った。
```

図－７　コマンド処理システムの出力例

実験した結果、ファイルやディレクトリ操作、プログラム開発など組み込んだ変換規則を用いたものは、正しく認識できていたことがわかったが、ワイルドカード使用やドライブ指定、子プロセスとしてCOMMAND.COMを起動した場合など、改良を要する問題点も明らかになった。

本システムを実用化するにはまだ変換規則が不足していたことがわかったが、目標の記述をＵＮＩＸ書式にしたためか変換規則が作りやすく、優先順位を決めるだけでＭＳ－ＤＯＳ書式の入力からプランが推定できた。しかし、一般的な人間の動作に関するプラン認識では変換規則が作りにくいことも多いし、優先順位だけでは認識できないかもしれない。ただ本システムのように、どのような形式で入力することになっているのか、又、どの範囲までなら処理できるかを明確にしておくことがどんな場合でも必要になると思われる。

５．結論

前章でも述べたように、現段階ではプランを認識するためには、入力形式が定まっていること、また知識ベースの中にある推論規則は多いほど好ましいが、実際にはその数に限界があると思われるので、どの範囲までが認識可能なのかを明確にすることが重要であろう。

＜参考文献＞

1) 飯田，相沢：意図の理解，情報処理，Vol 30, No.10, pp.1216-1223 (1989)
2) Schank,R.C. & Abelson,R.P. : Scripts,Plans, Goals,and Understanding, Lawrence Erlbaum (1977)
3) Schank,R.C. & Abelson,R.P. : SCRIPTS,PLANS, AND KNOWLEDGE, Proceedings of the 4th IJCAI, pp.151-157 (1975)
4) 浅賀，名取，中川：ＭＳ－ＤＯＳパーフェクトマスター（秀和システム）(1991)
5) 木江，本郷：ＭＳ－ＤＯＳユーザのためのＵＮＩＸ攻略法（日刊工業新聞社）(1990)
6) 坂本：たのしいＵＮＩＸ（アスキー出版局）(1990)
7) ＪＬＥリファレンス・マニュアルＩ　ユーザコマンド（ｓｕｎ）(1991)