コンピュータと教育　26－4
(1993. 4. 10)

UNIX コマンドの学習を支援するシステム

— エージェントモデルからのコマンド列生成とその検証 —

高橋 昌威　土屋 繁樹　伊丹 誠　伊藤 紘二

東京理科大学 基礎工学部

〒278 千葉県野田市山崎 2641
東京理科大学野田校舎

TEL:0471-24-1501 (内線 4226)

あらまし　本論文では，我々が開発中の UNIX コマンドの学習を支援するシステムにおいて，学習者が希望するタスクから，それを実現できるようなコマンド列を生成する仕組みを付け加えることを提案している．その目的のために，我々はエージェントモデル [1] を中間媒体に使用する．具体的には，システムは学習者に，タスクを実行する際の初期状態と最終状態をわかる範囲で入力させる．するとシステムは，その最終状態を実現するようなエージェントをできる限り検索する．そうして集められたエージェントを適当に並べ，さらに三角表 [6] を用いて，エージェントの作用による状態の推移が整合するように別のエージェントを補填していく．次に，コマンド列を解析してエージェント列に変換するプログラム [2] を逆に用いて，このエージェント列の候補を与えることにより，コマンド列の候補を生成することができる．それが失敗すれば，そのエージェント列がコマンド列では表現不可能であることを示す．これら一連のどのフェーズでも，学習者はシミュレーションを通じて，システムと対話することができ，自分の記述を修正したり，捕捉したりすることができる．

和文キーワード　UNIX コマンド，対話的学習環境，コマンド合成，エージェントモデル，行動計画問題，三角表

A Computer Assistant for Learning UNIX Command

— Synthesiging Candidate Commands from Specifications Using Agent Models as the Intermediary —

Masatake TAKAHASHI　Shigeki TSUCHIYA　Makoto ITAMI　Kohji ITOH

Department of Applied Electronics, Science University of Tokyo

2641 Ymazaki, Noda, Chiba 278, Japan

TEL:0471-24-1501 (ext.4226)

Abstract The paper is an interim report on our efforts in constructing a computer assistant that helps novice but ambitious learners of UNIX commands to acquire the ability to specify the task and compose UNIX commands accomplishing the task. For that purpose we propose to use our agent models as the intermediary. To be specific, the system lets the user input the initial and final states of the task to be carried out as far as he recognizes them. The system collects such agents as will attain the final states. Through arranging them as well as inserting between them other agents necessitated to realize the preconditions of the agents employed. The idea come from the triangle table used in action planning.

Nextly essentially the same program as is used in parsing command sequences into agent sequence can be run to generate candidate command sequence implying the given agent sequences. The failure of generation serves to filter out the infeasible agent sequences. At any phase of the process the learner is requested to interact with the system via selection simulation in order to amend, complement and refine his specification.

英文 key words　UNIX Commands, interactive learning environment, command synthesis, agent model, action planning

1 はじめに

我々が試作中の，UNIX コマンドの学習を支援するシステムでは，

1. コマンド列の動作を理解する
2. タスクの仕様を決め，それを実行するコマンド列を作る

の2つのフェーズに対する支援を目指しており，この目的を達成するために，エージェントモデル [1] を用いている．エージェントモデルとは，コマンドのする仕事よりももっとプリミティブな仕事をするモデルである．1. のフェーズの支援では，そのコマンド列をエージェントの系列に変換し [2]，それをシミュレーションして見せる [3] ことによって行なう．また，2. のフェーズの支援としては，システムは，学習者の希望するタスクをメニューによって選択させて，そのタスクを実現するようなコマンド列をいくつか提示することができる．しかし 現在のシステムでは，タスクとコマンド列とを直接結び付けてしまっているので，オプションや引数の様々な組合せや，いくつかのコマンドやリダイレクションなどが組み合わさったようなコマンド列に対しては対応できていなかった．そこで本論文では，タスクを直接コマンド列と結び付けるようなことはしないで，まずタスクからそのタスクを実現するエージェントの系列の候補を生成し，そのエージェント列の候補をもとにコマンド列の候補を生成する手法を提案する．このようにエージェントを媒体にすることによって，様々なオプションや引数の組合せ，複数のコマンドやリダイレクトなどを含むコマンド列の候補の生成が可能になる．システムは，この方法を用いて 2. のフェーズにおける高度な支援を実現できる．

2 エージェントモデルの例

ここでは，エージェントの例をいくつか紹介する．エージェントは

エージェント名 (引数)

という形式をとり，前提条件，追加リスト，削除リストなどを保持している．一般的には，図1のように表せる．

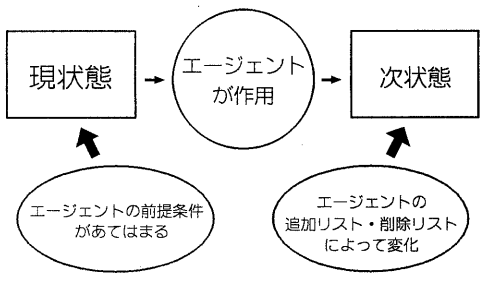

図1　エージェントモデル

エージェントは，作用する現在の状態と自分の前提条件と照らし合わせ，それが整合していればエージェントが作用し，その結果として追加リスト，削除リストによる状態遷移が生じる．この次状態は，そのまま次に作用するエージェントの前状態へとつながる．
引数には，エージェントが扱う対象が入る．

例として，

ファイルの内容転送 (F1,F2)
　前提条件　：ファイル (F1), ファイル_ 内容 (F1,F1c)
　　ファイル_ mode(F1,read_ok)
　　ファイル (F2)
　　ファイル_ mode(F2,write_ok)
　追加リスト：ファイル_ 内容 (F2,F1c)
　削除リスト：なし

といったエージェントがあるが，これはファイル F1 の内容をファイル F2 に転送するエージェントで，前提条件は，ファイル F1 が存在し，その内容は F1c であり，F1 は読み込み可能，ファイル F2 が存在し，それは書き込み可能，となる．そしてエージェント実行後，ファイル F2 の内容が F1c，という状態が追加されることを表している．
他にも，以下のようなエージェントがある．

- ファイルを作る (F)
  前提条件　：ファイル (F)
  追加リスト：ファイル (F)
  削除リスト：[]

- 標準出力つなぎ換え (F)
  前提条件　：標準出力_ 出力先 (stdout,crt)
  　　ファイル (F)
  追加リスト：標準出力_ 出力先 (stdout,F)
  削除リスト：[]

- 標準出力へ出力 (F,C)
  前提条件　：標準出力_ 出力先 (stdout,F)
  　　ファイル (F)
  　　バッファ_ 内容 (B,C)
  追加リスト：ファイル_ 内容 (F,C)
  削除リスト：[]

- 文字列の差をとる (X,Y,DXY)
  前提条件　：文字列 (X), 文字列 (Y)
  　　バッファ (N)
  　　バッファ_ 内容 (BX,X)
  　　バッファ_ 内容 (BY,Y)
  追加リスト：文字列の差 (X,Y,DXY), 文字列 (DXY)
  　　バッファ_ 内容 (N+1,DXY)
  　　バッファ (N+1)
  削除リスト：バッファ (N)

- ファイルを読む (F,C)
  前提条件　：バッファ (N)
  　　ファイル_ 内容 (F,C)
  　　ファイル_ mode(F,read_ok)
  追加リスト：バッファ_ 内容 (N+1,C)
  　　バッファ (N+1)
  削除リスト：バッファ (N)

3　エージェントモデルの拡張

前節のような従来のエージェントモデルでは，1つのコマンド列に対するエージェント動作の進行は分岐を持たなかった．しかし，実際のコマンドの実行中には，ユーザにyes/noを聞いたり，システムの状態が変化したりするとその後の処理が変わってしまうことがある．そのような場合にも対応できるように，エージェントスクリプトを定義する．これは，いくつかのエージェントが組み合わされたもので，条件によって実行されるエージェントが変化するようになっている．これは，次のように表すことができる．

```
<エージェントスクリプト> ::=
  <エージェント>,<エージェントスクリプト> |
  [cond,[ <条件分岐スクリプト> ]],
              <エージェントスクリプト> |
   nil

<条件分岐スクリプト> ::=
    [<条件>,<エージェントスクリプト>] |
    [<条件>,<エージェントスクリプト>],
                    <条件分岐スクリプト>
```

上記のように，エージェントスクリプト，条件分岐スクリプトは再帰的に呼び出すこともできるので，ユーザの状態やシステムの状態などがある条件を満たすまで，エージェントの実行を繰り返すといったことが可能になる．

4　コマンド列の生成

学習者の希望のタスクを実現するコマンド列を生成するには，まず，そのコマンド列を実行した時に状態がどう変化するかを学習者が指定することから始まる．つまり，初期状態と最終状態を指定する（この時，完全な状態記述をする必要はなく，不完全な記述でもシステムは候補の生成過程で，学習者と対話しながらその部分を埋めていく）．たとえば，

2つのテキストファイル (f1,f2) の内容の違いを調べ，その結果を別のファイル (f3) に出す．

というタスクを学習者が考えたとすると，学習者は

初期条件
テキストファイル (f1), テキストファイル_ 内容 (f1,f1c)
テキストファイル (f2), テキストファイル_ 内容 (f2,f2c)
最終状態
テキストファイル (f3), テキストファイル_ 内容 (f3,fd)
文字列の差 (f1c,f2c,fd)

といった指定を行なう．次にシステムは最終状態を参考に，そのような状態に遷移することが可能なエージェントを探せるだけ探し，それらのリストを生成する．上記の例では，さきほどのエージェントのデータより，

```
[文字列の差をとる (f1c,f2c,fd),
 [標準出力へ出力 (f3,fd),
  ファイルの内容転送 (F1,f3) ],
 ファイルを作る (f3)]
```

というリストができる．ここで，1つの状態情報から複数のエージェントの候補ができる場合，それらを1つのリストとしてまとめておき，そのリストからは1度に1つの候補しか出せないようにしておく．この場合，「標準出力へ出力 (f3,fd)」と「ファイルの内容転送 (F1,f3)」というエージェントは，最終状態の状態情報「テキストファイル_ 内容 (f3,fd)」を追加リストとして持っているため，リストにしてどちらか片方だけがエージェント列の候補にでてくるようにしている．

次に，そうしてできたエージェントのリストの中で可能な組合せを作り出す．例えば，

```
[ 文字列の差をとる (f1c,f2c,fd),
  ファイルの内容転送 (F1,f3) ,
  ファイルを作る (f3) ]

    エージェントリスト1

[ ファイルを作る (f3),
  文字列の差をとる (f1c,f2c,fd),
  標準出力へ出力 (f3,fd) ]

    エージェントリスト2
```

などのリストが生成される．これを初期リストとして，三角表 [6] を使いながらエージェント列の完全な候補を生成する．エージェントリスト2を初期リストとすると，三角表の最初の状態は下の図のようになる．

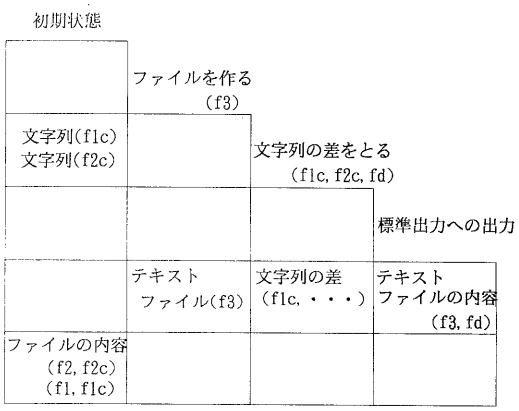

図2　　最初の三角表

図2において，エージェントの真横に並ぶ枠にはそのエージェントの前提条件，下に並ぶ枠にはエージェント実行後の追加リスト，削除リストの作用によって変化した状態が記述される．この三角表の状態では，エージェントの前提条件が満たされているのは「ファイルを作る (f3)」だけである．これでは他のエージェントは実行できないことになるので，それらエージェントの前提条件が次状態として現れるようなエージェントを，そのエージェントの前のどこかに挿入する．足りない条件がなくなるまで，エージェントの挿入を繰り返し行なう．ここで失敗すればシステムは次のエージェントの候補を試すが，成功した場合，図3のような三角表ができる．

図3において，一番下の枠は使われなかった条件（状態）が入っている．

結局，できあがったエージェント列の候補は，

```
[ ファイルを作る (f3),
  標準出力つなぎ換え (f3),
  ファイルを読む (f2,f2c),
  ファイルを読む (f1,f1c),
  文字列の差をとる (f1c,f2c,fd),
  標準出力へ出力する (f3,fd)  ]
```

となる．しかし，このようにしてエージェント列を生成しても，必ずしもそれが現実のコマンド列に結び付くとは限らない．よって，システムはエージェント列の候補をできるだけ多く生成しておく．

エージェント列からコマンド列を生成するには，コマンド列解析部 [2] で使われている仕組みを逆に利用することによっておこなう．解析の時はコマンド列を与えてエージェント列を生成していたが，今度は逆にエージェント列を与えてコマンド列を生成させることになる．ここでエージェント列の候補がコマンド列で実現不可能ならば，システムは次のエージェント列の候補を試していく．こうして，あるエージェント列の候補が現実のコマンド列で表せれば，コマンド列が生成できたことになる．この例ではコマンド列生成が成功し，結局，

```
diff f1 f2 > f3
```

というコマンド列が結果として得られる．

このコマンド列をシミュレートすることも可能であり，シミュレートする時はコマンド列に対応するエージェント列をシミュレーション部に送ればよい．このシミュレーションが，学習者の希望するタスクと違っていた場合，システムはエージェント列の新しい候補を持ってきて，同様なコマンド列生成を繰り返す．また，同じタスクをする別のコマンド列がないだろうか，という学習者の問いかけにも，エージェント列からコマンド列を生成する段階で別解を求めるようにシステムに指示すれば可能である．

5 おわりに

今回述べてきたことは，まだ構想の段階であり，今後はこれを実現していかなくてはならない．また，このままでは3節で述べた，拡張されたエージェントをそのまま適用できない．これは，条件によって，作用するエージェントが変化するためで，この場合でも，学習者が事例として与えたエージェント列を，可能なバリエーションの1つとして含むコマンドを提示することはできる．

参考文献

[1] 高橋昌威, 伊丹誠, 伊藤紘二：UNIX コマンドの学習を支援するシステム — エージェントモデルによる試み —, 電子情報通信学会技術研究報告, 教育工学, ET92-16,(1992).

[2] 高橋昌威, 伊丹誠, 伊藤紘二：UNIX コマンドの学習を支援するシステム — エージェントモデルによるコマンド列理解 —, CAI 学会全国大会資料,(1992).

[3] 土屋繁樹, 高橋昌威, 伊丹誠, 伊藤紘二：UNIX コマンドの学習を支援するシステム — エージェントモデルによるコマンド列のシミュレーション —, 電子情報通信学会春季全国大会,A-297,(1993).

[4] 伊藤紘二、伊丹誠、宮本健：学習支援環境 CAFEKS における問題解決支援機構の試作, 電子情報通信学会論文誌,Vol.J75-A,No.2,(1992).

[5] 舟本奨：UNIX ハンドブック, ナツメ社,(1990).

[6] NILS J. NILSSON:PRINCIPLES OF ARTIFICIAL INTELLIGENCE,Tioga Publishing,(1980).

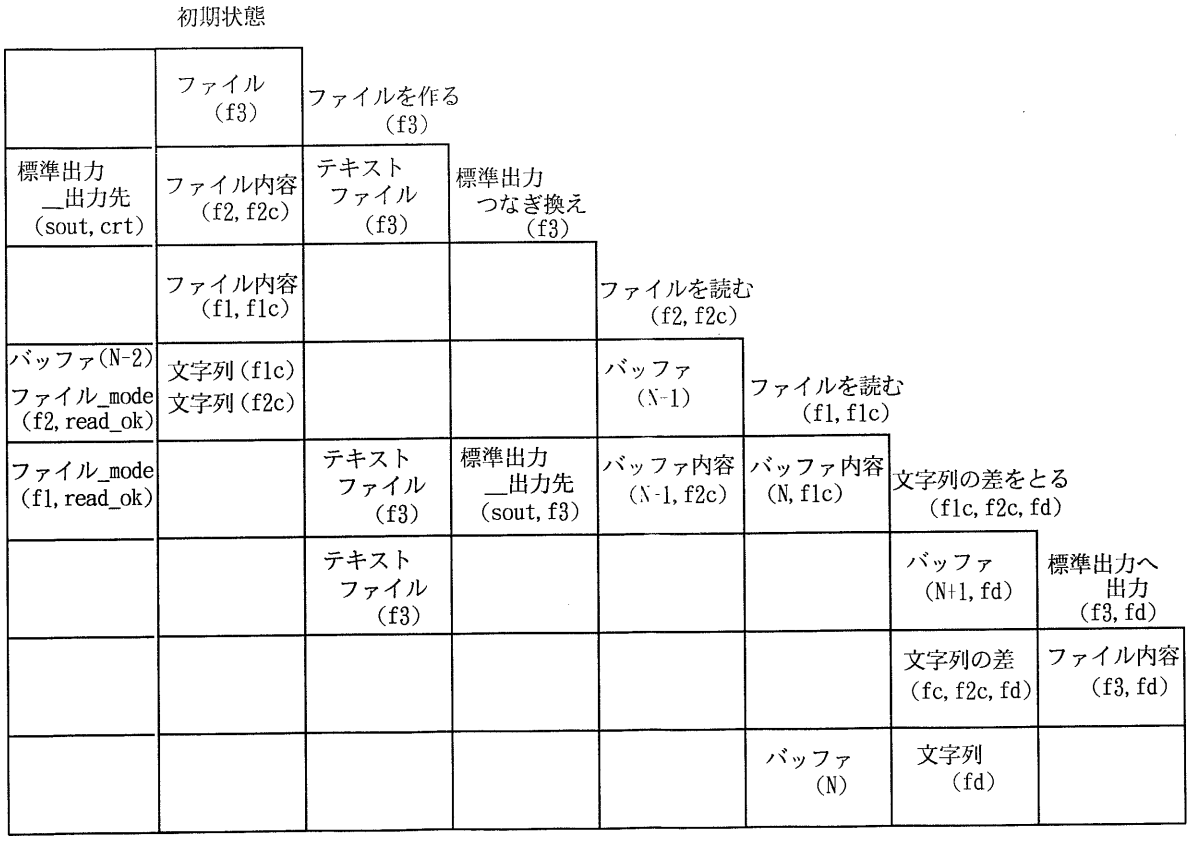
初期状態
ファイル
(f3)
ファイルを作る
(f3)
標準出力
_出力先
(sout, crt)
ファイル内容
(f2, f2c)
テキスト
ファイル
(f3)
標準出力
つなぎ換え
(f3)
ファイル内容
(f1, f1c)
ファイルを読む
(f2, f2c)
バッファ(N-2)
ファイル_mode
(f2, read_ok)
文字列(f1c)
文字列(f2c)
バッファ
(N-1)
ファイルを読む
(f1, f1c)
ファイル_mode
(f1, read_ok)
テキスト
ファイル
(f3)
標準出力
_出力先
(sout, f3)
バッファ内容
(N-1, f2c)
バッファ内容
(N, f1c)
文字列の差をとる
(f1c, f2c, fd)
テキスト
ファイル
(f3)
バッファ
(N+1, fd)
標準出力へ
出力
(f3, fd)
文字列の差
(fc, f2c, fd)
ファイル内容
(f3, fd)
バッファ
(N)
文字列
(fd)

図3　完成した三角表