実行履歴の構造化に基づくコマンド列の再実行支援に関する研究

2013SE075 加藤宗一郎
指導教員：吉田敦

1 はじめに

一般に，プログラムや文書を一度の編集のみで完成させることは困難であり，修正や確認などを何度も繰り返すことで完成に近づける．その際，ソースファイルへの編集内容を最終成果物に反映させるために，作業者 (以下ユーザとする) は同じコマンドを繰り返し実行する．作業は複数のコマンドで構成されることが多い．Makefile やシェルスクリプトなどを記述して実行することで，繰り返し作業を簡略化できるが，作業を明確に把握していない場合，試行錯誤しながら作業を進めるので事前にスクリプトを完成できない．実行履歴からコマンド列を指定して再実行できれば，ユーザは事前にスクリプトを記述する必要がない．しかし，現状のシェルでは，実行履歴から選択できるコマンドは 1 つのみで，複数のコマンドを指定しようとすれば，実行するコマンドの数だけ指定する必要がある．そこで，実行履歴から選択して実行できるコマンドの数を単一から複数に拡張する方法を考える．

複数のコマンドを再実行するためには，実行履歴から特定の範囲のコマンド列を選択できる必要がある．しかし，(1) 選択した範囲のコマンド列が再実行したいコマンド列であるとは限らず (2) ユーザが選択したいコマンド列が連続して出現しているとは限らないという問題点が挙げられる．

本研究では，実行履歴中からユーザが再実行したいコマンド列を簡潔に選択できるよう候補を生成する手法を提案し，その支援システムを実現する．

2 関連研究

先行研究 [1] では，コマンドの実行履歴に基づき，次に実行されるコマンドを確率的に予測する．先行研究 [2] では，コマンドの実行履歴とファイルの入出力の情報を用いて，コマンド間の関係を解決し，次に実行されるコマンドを予測する．先行研究 [3] では，機械学習を用いて習得したユーザの行動の傾向から，次に実行される操作を予測する．先行研究 [1], [2]，[3] は，予測する操作が単一であるという点で，本研究とは異なる．

3 実行履歴に基づく再実行の指定方法

3.1 シェルの実行履歴に基づく再実行方法

シェルにおける実行履歴とは，直前までに実行されたコマンドのリストを指し，コマンドとは，処理を行う際にユーザから与えられた命令を表す．以前に実行したコマンドを再度実行するには，実行履歴から条件を指定して実行したいコマンドを指定する．例えば，bash[5] では履歴展開機能を用いる．

3.2 コマンド列の再実行

実行履歴中からコマンド列を指定するためには，開始と終了の指定が必要である．しかし，様々な作業が混在する実行履歴からその範囲を見つけることは容易ではない．そこで，自動的に履歴を分割し，候補を生成することで支援する．一般に，1 つの作業では短い時間間隔でコマンドが実行されると想定されるので，時間に基づいたコマンド列の分割により，作業ごとに分割できると予想できる．129 個のコマンドで構成される実際の実行履歴で時間間隔を調べたところ，作業間の最短時間は 13 秒であった．このことから，作業時間に基づいて分割できると想定する．

3.3 コマンド列の再構成

ユーザが繰り返したい作業が，常に連続して実行されているとは限らない．(1) 作業内容の決定時に参照したコマンドの実行，(2) 作業決定時の思考時間による作業の分割，(3) データ加工作業時の中間状態の確認により作業が分断されることがある．よって，実行に不要なコマンドの除去や，複数のコマンド列の結合によってコマンド列を再構成する必要がある．

コマンド間に依存関係とは，あるコマンド c_1 の実行によって出力されたファイルが異なるコマンド c_0 の実行によって読み込まれているとき，c_0 は c_1 に依存していると定義する．コマンド間の依存関係を解析し，プログラム・スライシング [4] の手法を応用すれば，結合すべきコマンド列を構成できるが，コマンドの仕様情報なしに依存関係の解析は困難である．そこで，コマンドが入出力するファイルの情報を用いることで，コマンドの仕様を知らなくても依存関係の解析ができる．

データ変換処理のように，同じ種類の複数ファイルに同じ処理を適用することがある．これを支援するには，コマンドの入出力に用いられたファイル名が拡張子のみ異なる場合はデータ変換処理と識別し，ファイル名を置き換えた候補を生成する．

4 再実行コマンド列選択システム

4.1 システムの構成

選択支援システムは，図 1 のように，実行履歴を記録する機能と，実行履歴を確認する機能，およびコマンド列を自動実行する機能から構成される．

4.2 実行履歴記録ツール

実行履歴記録ツールは，シェルの拡張機能として trap コマンドとシェル変数 PROMPT_COMMAND を利用し，記録する機能を組み込む．記録する情報は実行されたコマンド

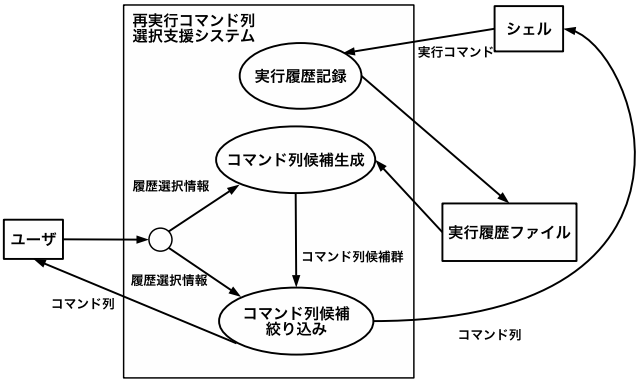

図 1　システムの全体像

とその開始と終了時刻，ファイルの更新履歴である．

4.3　コマンド列再実行ツール

コマンド列再実行ツールは，ユーザからの呼び出しによって起動され，コマンド列候補生成処理によってファイル出力をしないコマンドを除去したものや，ファイル名を置換処理したコマンド列といった候補を生成される．ユーザはオプションによってコマンド候補列の生成処理を指定できる．候補の生成処理は独立したプログラムとして実装しており，追加や変更が容易になっている．

生成されたコマンド列候補から絞り込み処理を行うためには，候補番号やファイル名などを用いて条件をユーザが指定する．

4.4　評価と考察

本システムの開発や，本論文の執筆などの作業を記録し，合計 129 行の実行履歴を得た．この履歴を用いてコマンド列を選択し，適切にコマンドリストから指定できるか確認した．それらの結果から，期待した動作との差が少ないコマンド列を出力した場合と，期待とは異なったコマンド列を出力した場合が存在した．図 2 は履歴をコマンド列に分割した際の表示結果の一部であり，5 個のコマンド候補列が約 0.4 秒で出力された．図 3 は図 2 中の 0 番目の作業についての候補列の生成結果である．図 4 は期待されるスクリプトを示し，図 5 は出力結果で，期待する動作との差分は `vim rep.tex` のみであった．

結果から，エディタの実行を除去できれば，期待する出力と一致した．一般に，コンパイルのようなデータの変換処理を自動実行する際，ソースファイルを編集したあとに自動実行するので，エディタの実行を除去できれば期待する結果との差分を小さくできると考えられる．エディタやコンパイルのようなコマンドを予め登録し，一般的な作業を登録することで，より精度の高い候補の生成と絞り込みができる．

5　おわりに

ユーザが再実行したいコマンド列を実行履歴中から簡潔に選択できる手法を提案し，その支援システムを実現した．

```
vim rep.tex
platex rep.tex
dvipdfmx rep.dvi
open rep.pdf
vim fig.tex
platex fig.tex
man dvips
dvips -E fig.dvi -o fig.eps
vim rep.tex
platex rep.tex
platex rep.tex
dvipdfmx rep.dvi
vim fig.tex
platex fig.tex
dvips -E fig.dvi -o fig.eps
vim rep.tex
platex rep.tex
platex rep.tex
dvipdfmx rep.dvi
vim fig.tex
tasks
again -n 0 -o rep.pdf
```

図 2　作業の区切りの一部

```
[#0-1]
--------------------
vim fig.tex
platex fig.tex
dvips -E fig.dvi -o fig.eps
vim rep.tex
platex rep.tex
platex rep.tex
dvipdfmx rep.dvi
--------------------
[#0-2]
--------------------
platex fig.tex
dvips -E fig.dvi -o fig.eps
vim rep.tex
platex rep.tex
platex rep.tex
dvipdfmx rep.dvi
--------------------
```

図 3　生成される候補

```
platex fig.tex
dvips -E fig.dvi -o fig.eps
platex rep.tex
platex rep.tex
dvipdfmx rep.dvi
```

図 4　期待されるスクリプト

```
platex fig.tex
dvips -E fig.dvi -o fig.eps
vim rep.tex
platex rep.tex
platex rep.tex
dvipdfmx rep.dvi
```

図 5　実際の出力結果

今後の課題は，候補列の生成と絞り込みの方法を拡張することで生成するコマンド列候補の精度の向上である．

参考文献

[1] B. Korvemaker, and R.Greiner, “Predicting UNIX Command Lines: Adjusting to User Patterns,” *Proc. of the 17th National Conf. on Artificial Intelligence and 12th Conf. on Innovative Applications of Artificial Intelligence*, pp. 230–235, 2000.

[2] K. Yoshida, “User Command Prediction by Graph-Based Induction,” *Tools with Artificial Intelligence*, pp. 732–735, 1994.

[3] J.Ruvini, and C.Dony, “APE: Learning User’s Habits to Automate Repetitive Tasks,” *Proc. of the 5th int. conf. on Intelligent user interfaces* , pp. 229–232, 2000

[4] 下村隆夫，“プログラムスライシング技術と応用，” 共立出版，1995

[5] Free Software Foundation, Inc., “*Bash Reference Manual,*” `https://www.gnu.org/software/bash/manual/bash.html#History-Interaction`