2005－EVA－12 (4)
2005／3／23

プログラミング演習におけるプログラムの評価についての一考察

笹倉 万里子
岡山大学大学院自然科学研究科

概要

本大学では学部生がコンパイラを作成するという演習を行っている．この実験では，あらかじめプログラムの枠組を例として与え，学生はそれを参考にしながらプログラムを作成する．本研究では，学生が作成したプログラムを視覚化することによってプログラムを評価することを試み考察を行う．

An evaluation for programs written by students

Mariko Sasakura
Department of Intelligence Computing and System,
Graduate School of Natural Science and Technology, Okayama University

Abstract

In Okayama University, we have an exercise program of implementing a compiler. For the exercise, we prepare a frame of a compiler, and students write a compiler referring to the frame. In this paper, we try to evaluate the programs written by students with figures that visualize the programs.

1 はじめに

プログラムがよいかどうかを評価したいというのは，プログラミングが多数の人によって行われるようになって以来普遍的な要求の一つであろう．しかし，今もってプログラムを評価する決定的な方法はない．

プログラムの何をもってよいと評価するかには様々な基準があるだろう．プログラムが仕様と合っているか否か，バグがないか否かは，実用的なプログラムの作成においては非常に重要な点である．しかし，仮に仕様に合っていて重要なバグが存在しないことが確認できたプログラムであっても，実装の善し悪しを論ずる場合がある．例えば，

- プログラムの読みやすさ
- プログラムの実行速度

などでプログラムを評価することがある．

一般的にプログラムは，

- 一つのことを実現する方法が無数にあること
- 一般にプログラムはサイズが大きく全部を検査することは不可能に近いこと

という特徴がある．従って評価も容易ではない．

しかし，だからといってプログラムの評価をしなくていいというわけにはいかない．例えばソフトウェアを納入する時に複数の候補を評価してどれを選ぶか決めなければならない場合がある．プログラミング能力を計るためにプログラムを評価するという場合もある．プログラミング教育においては学生の書いたプログラムをなんらかの方法で評価しなければならない．

本稿では，学生のプログラミング演習において書かれたプログラムの評価について考察する．ここで対象とする演習では学生がコンパイラを作成する．本研究では，そのコンパイラが生成した目的コードの実行時間とプログラムの構造の間に相関があるかどうかを調べることを目的とする．

2節で対象とするプログラミング演習の概要を説明する．3節でプログラムをどのようにモデル化し評価するかを述べる．4節で評価の結果と考察につ

いて言及し，5節で結論を述べる．

2 学生実験概要

今回の評価対象であるプログラムは岡山大学工学部情報工学科3年生に対して行われている情報工学実験のコンパイラ作成実験（以下コンパイラ作成実験）において作成されたものである．本節ではこの実験の概要を簡単に説明する．

コンパイラ作成実験において学生は次のことを行う．

- コンパイラの対象言語の設計
- コンパイラの実装
  - 字句解析部
  - 構文解析部
  - コード生成部

コンパイラの出力する目的コードは実験用に設計されたアセンブラ言語である．

対象言語の設計では，言語としての基本機能すなわち算術式の計算，条件分岐，ループができることを条件にどのような言語を設計してもよいとしている．しかし，学生は2年次までの演習をほとんどC言語で行っているため，実際にはほとんどの学生がC言語のサブセットのような言語を設計する．

コンパイラの字句解析部，構文解析部の作成では，lexやyaccのようなツールを使わずに作成する．実際には，プログラムの例を示して作成方法を説明するため，ほとんどの学生が，ファイル構成，主要な関数の構成，主要な関数の名前については同じものを用いている．

一方，コード生成に関しては，概念的な説明だけをして具体的な例を示さないので，学生によってその実装はまちまちになっていると考えられる．学生にはコンパイラが出力する目的コードが効率のよいものである方が高い評価を得られると伝え，最適化をすることを推奨している．しかし，あまり具体的な最適化の方法については説明せず学生の能力で最適化を行うよう促している．なお，このコンパイラでは入出力関数は実装しない．

実験の評価のために，課題を設定し，それを実現する目的コードの実行に要するステップ数をエミュレータを用いて計る．今年度はエラトステネスのふるいのアルゴリズムによって1000までの素数を調べることを課題とした．

コンパイラの実装言語は特に規定していない．しかし，説明の例はC言語で記述したものを用いており，学生全員がC言語による実装を行っている．

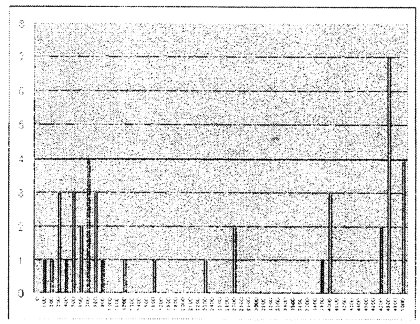

図1: 課題の結果

3 プログラムの評価方法

図3に課題の結果を示す．横軸はステップ数を要素数である1000で割った値を100間隔で取ったもの（以降これを実行性能と呼ぶ），縦軸が人数である．これをみると，だいたい100～1000のところに分布のピークがあり，4000以上のところにもう一つのピークがありそうに見える．100～1000をAグループ，1000以降をBグループと呼ぶことにする．この両者でプログラムに違いがあるかどうかを見る．

プログラムの比較は次のようにして行う．まず，コンパイラのソースプログラムをブロックで分けて木構造で表す．以降これをプログラムの構造と呼ぶことにする．ここではブロック構造として{から}までを一つのブロックとして，それを一つのノードとする．木構造のエッジはブロック同士の包含関係を示す．すなわち，ブロック1からブロック2に枝がある場合にはブロック1の中にブロック2があることを示す．このプログラムの構造を示す図でプログラムの比較を行い，AグループとBグループの間に違いがあるかどうかを見る．

プログラムは人間の思考形態を表したものであると考えられる．すなわち，論理的思考のできる学生の書くプログラムは論理的にきれいなものになり，論理的思考に慣れていない学生の書くプログラムは論理的にきれいなものにならないと予想できる．また，論理的な思考のできる学生の書くプログラムの方がよいプログラムになる可能性が高いと予想できる．論理的思考とは，しばしばきれいな木で表すことが可能なものである．ここでは，次のようなものをきれいな木とすることにする．

- 階層の深さが適当である．
- 木の複雑さ，すなわちノードの数が適当である．

適当であるかどうかは書こうとしているプログラムの課題の複雑さに依存するので一般的に示すことは

難しい．そこで，今回はAグループとBグループのプログラムを比較し，グループごとにプログラムの構造に特徴が見られるかどうかでこの仮説を検証する．

コンパイラのプログラムは4つのファイルに分かれている学生が多い．これは例で示したプログラムがそうなっているからである．その四つとは

main.c プログラムの main が書かれているもの

lex.c 字句解析を行う部分

oparser.c 算術式の構文解析を行いコードを生成する部分

parse.c 算術式以外の部分の構文解析を行いコード生成する部分

である．ファイルの分け方を特に指定しているわけではないので，5つ以上のファイルに分かれている学生もいる．しかし，5つ以上のファイルに分かれているものと4つのファイルに分かれているものとの間で比較をするには木同士の対応関係が明確ではないので，ファイルが4つにわかれているものだけを対象として比較する．

4 評価結果と考察

AグループとBグループから4つのソースファイルでコンパイラを構成している学生を4人ずつ抽出してそのプログラムの構造を表示した．それぞれの学生のコンパイラが出したコードの実行性能を表にして示す．図2，図3，図4はそれぞれ学生の書いた lex.c，oparser.c，parse.c のプログラムの構造を示したものである．main.c に関してはどの学生のプログラムの構造も大差ないので図は省略する．なお、これらの図は GraphViz[1][2] を用いて作成し，ZGRViewer[1] で表示したものである。

図から読みとれる各ソースファイルごとの特徴を述べる．

main.c もともと提示している例が関数 main と 関数 usage のみが存在するファイルとなっており，どの学生も大差ない．

lex.c AグループとBグループであまり差がないが，Aグループの方がやや階層が深い木となる傾向が見られる．

oparser.c Aグループの方がBグループより木が単純で見やすい，つまり，図にした時に一つ一つのノードが大きく表示される傾向が見られる．

[1] http://zvtm.sourceforge.net/zgrviewer.html

表 1: 評価に用いた学生プログラムの実行性能

学生	実行性能
a_1	353
a_2	554
a_3	583
a_4	729
b_1	2318
b_2	4021
b_3	4780
b_4	4827

parse.c あきらかにAグループの方がBグループより複雑である．これはAグループではコードの性能をあげるための工夫を行っているのに対し，Bグループではそれが行われていないためと推測できる．

5 おわりに

プログラムの評価の一例として，岡山大学工学部情報工学科で行っている演習で書かれたプログラムを木構造として表示し，性能のよいプログラムとあまりよくないプログラムの間でその構造に違いがあるかを比較した．

本稿で示した範囲では，対象としたプログラムの数も少なく，また，比較も木構造の図を目で比較するというものだったので，違いがあるかどうかを判断するのは早計であると思われる．今後より多くのプログラムを対象とし，また，木の比較を定量的に行うことによって違いがあるのかどうかをより明確にすることが可能であると考えられる．

参考文献

[1] J. Ellson, E. Gansner, L. Koutsofios, S.C. North, and G. Woodhull, *Graphviz - open source graph drawing tools*, Graph Drawing. 9th International Symposium, GD 2001. Revised Papers (Lecture Notes in Computer Science Vol.2265), pp.483-4, 2002.

[2] E.R. Gansner, and S.C. North, *An open graph visualization system and its applications to software engineering*. Softw. - Pract. Exp. pp.1203-33, vol.30, no.11, 2000.

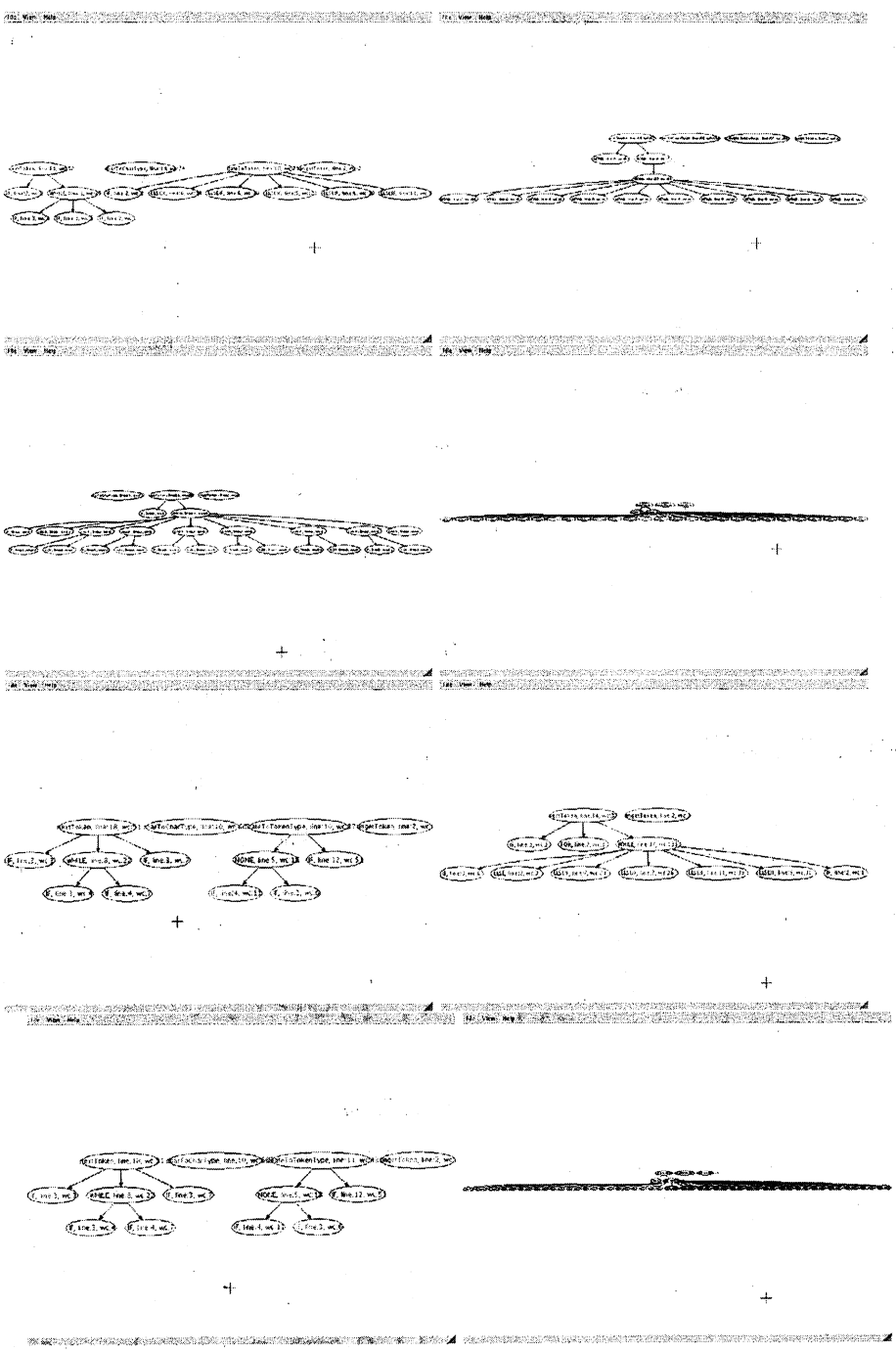

図 2: lex.c のプログラムの視覚化．上 4 つが A グループ下 4 つが B グループのもの．

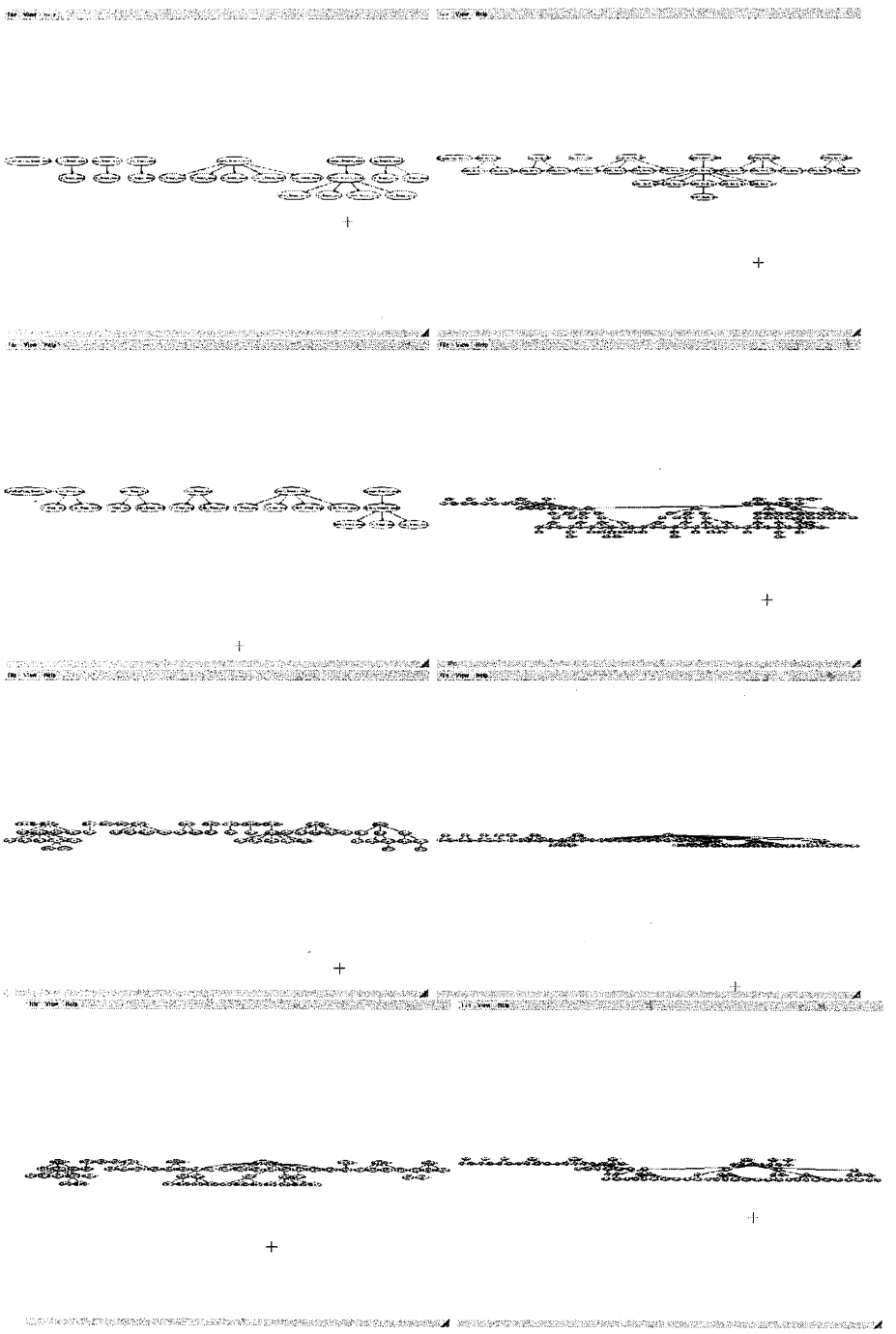

図 3: oparser.c のプログラムの視覚化．上 4 つが A グループ下 4 つが B グループのもの．

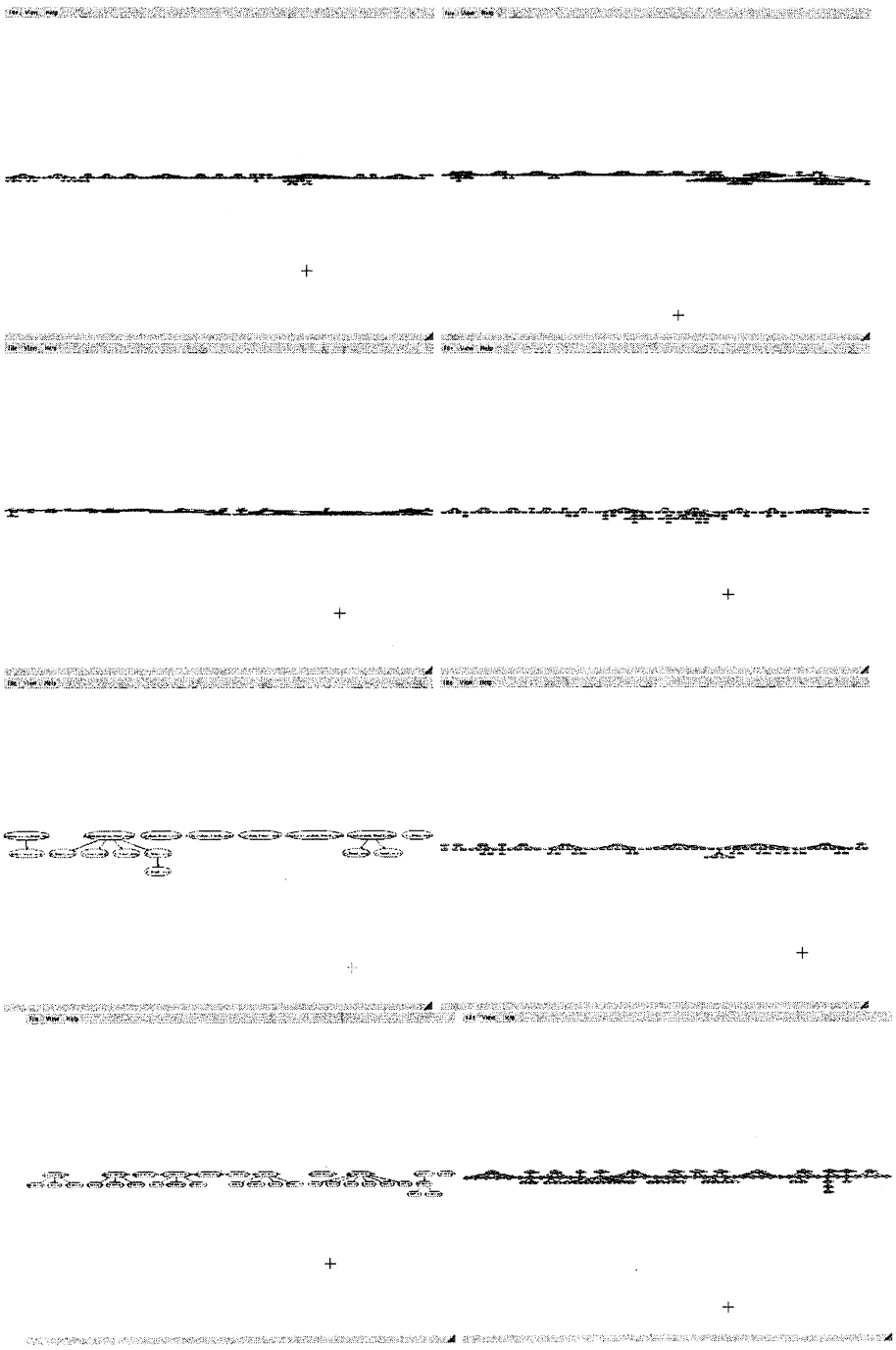

図 4: parse.c のプログラムの視覚化．上 4 つが A グループ下 4 つが B グループのもの．