Jackson木の詳細化によるアルゴリズム設計支援システム
A System to Support Algorithm Design of a Program by Refinement of Jackson Tree

草野翔† 櫻井孝平‡ 古宮誠一‡
Sho Kusano† Kohei Sakurai‡ Seiichi Komiya‡
† 芝浦工業大学　工学部
‡ 芝浦工業大学大学大学院　大学院工学研究科
† College of Engineering, Shibaura Institute of Technology
‡ Graduate School of Engineering, Graduate School of Shibaura Institute of Technology

要旨
プログラミングに際し、アルゴリズムの詳細化にはフローチャートなどのプログラム図が有効であることが知られている。しかし、これらの詳細化ではプログラムに対して直接的な設計なので負担が大きく、ボトムアップ向けの設計であるために詳細化に向かないという点と、プログラムの構造化が保証されないという問題があった。そこで Jackson 木を用いると、トップダウン設計ができ、プログラムの構造化が保証される。さらに、アルゴリズム設計手法として JSP 法を用いることで、入出力のデータ項目に基づき間接的に設計できるため、前者に比べ負担が少ない。このため、Jackson 木作成用の図式エディタ機能を持ったアルゴリズム設計支援システムを開発し、提案する。

1. はじめに

ソフトウェア開発に必要なプログラミング技術を持った人材の育成のため、大学などの教育機関ではプログラミング教育が行われている。基本的な文法規則からアルゴリズムの実装、ソフトウェア開発の演習に渡って高度なプログラミング技術を持った人材の育成を目指しているが、現状では文法規則まで理解したとしてもアルゴリズムの実装で躓く学習者が多い。

学習者はアルゴリズムの実装に際しては構造化プログラミングの習得が要求される。構造化プログラミングを理解することで制御構造が見えるようになり、アルゴリズムを正しく実装できる。現在では、ほとんどのプログラミング言語が構造化プログラミングをサポートしており、教育現場の現状としても、アルゴリズムの実装と構造化プログラミングを区別なく教えることが多いので、学習者は構造化プログラミングに慣れないまま学習を進められてしまいプログラム構造の理解に悩んでいる。

本研究は、学習者が正しく構造化プログラミングを行ったアルゴリズムの実装を可能にするため、JSP 法[1]を用いた設計をプログラミング教育に導入する。JSP 法による設計支援のために Jackson 木を用いてアルゴリズムを表現する図式エディタを開発する。

2. 問題点

プログラミング初学者がプログラムを正しく実装することは難しい。そのため、与えられた仕様を実現するプログラムの全体像をイメージできるようになる必要がある。このイメージを視覚化し、アルゴリズムの詳細化を兼ね備えるものとして、フローチャートなどのプログラム図を用いることが有効であることが知られている。

一方フローチャートの利用には２つの問題点がある。

１つ目は、フローチャートを使った設計はプログラムの実装に対しての直接的かつボトムアップ向けの設計であり、仕様を実現するプログラムの実装に対応する図を記述することが、プログラム初学者には負担が大きい。加えて、作成した図を見返したときに、ある程度の図の流れを見ないと、どのような処理を行っているのかを理解しにくいか理解できない可能性がある。

２つ目は、フローチャートでは構造化プログラミングが保証されないという点である。多くのプログラミング言語では構造化プログラミングをサポートしており、goto 文による大域的なジャンプ命令を使

ったプログラムは実装できない。しかし、初学者によるフローチャートを使った設計は goto 文による実装に対応することが多い。

続く 2.1 節では構造化プログラミングの習得について、より詳細に説明する。

2.1. 構造化プログラミング

プログラミングの学習手順として、プログラミング言語の文法を学んだ後、まず構造化プログラミングを習得する必要がある。構造化プログラミング[2]は 1967 年ダイクストラにより提唱され、以下の 3 つの制御構造のみを用いてプログラミングを行うというものである。

- 連接(concatenation)
- 反復(repetition)
- 選択(selection)

この構造を用いてプログラム全体をいくつかの手続きに分け段階的に分割する。

しかし、構造化プログラミングだけでは制限が強すぎることが知られている。原則として構造化プログラミングを行い、一部の非構造化プログラミングによって簡単に表現できるロジックや、例外処理を非構造化プログラミングとして実装することが一般的である。

本研究では上記に示した学習の流れの一環として、プログラミングの際は構造化プログラミングのみを行うことを前提とする。

3. 提案

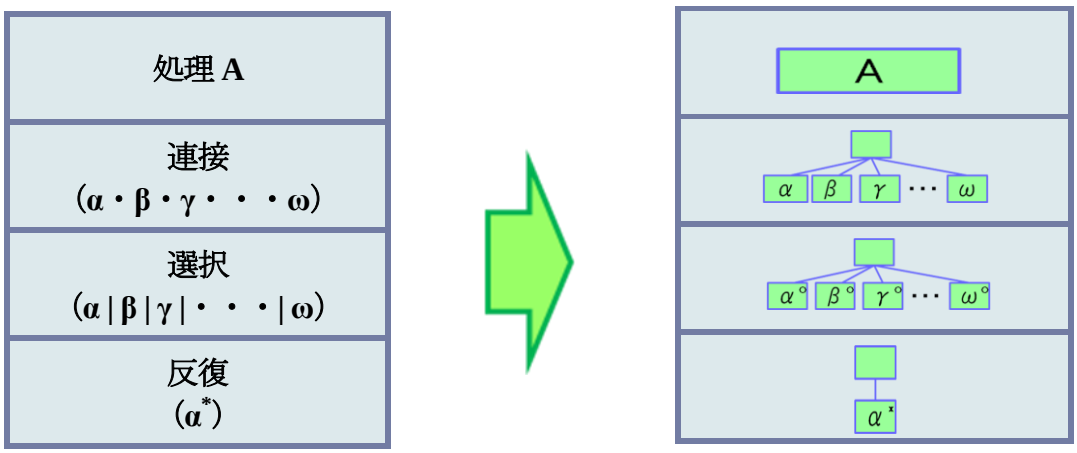
問題点の解決方法として、アルゴリズム設計手法として JSP 法[1]を用いることを提案する。JSP 法は Michael A. Jackson によって考案され、それに用いられる Jackson 木は正規表現を図で表したものである。アルゴリズム詳細化の際に連接・反復・選択の制御のみを用い、データ構造・プログラム構造を表すことができる。これにより、トップダウンによる段階的なアルゴリズム詳細化設計ができるようになり、Jackson 木によって表されたアルゴリズムは必ず構造化が保証される。また、JSP 法は入出力のデータ構造に着目して正規表現でモデル化するだけで，モデル化の対象となったデータを処理するプログラムのアルゴリズムが得られる。よって、プログラムに対して直接の設計ではなくデータ構造に着目した間接的な設計なのでアルゴリズム詳細化の負担は軽減される。

また、この Jackson 木を作成できる図式エディタとしての機能を持ち、いくつかの例題を組み込むことで JSP 法アルゴリズム設計の一部を支援し学習することのできるシステムを開発する。

3.1. Jackson 木

既存の Jackson 木の表記法を下に示す。

表 1　Jackson 木の表記法



3.2. JSP 法

3.2.1. JSP 法によるアルゴリズム設計法

JSP 法によるアルゴリズム設計は次のようにして導かれる

- 入力データ構造に沿って入力の木を作成する
- プログラム仕様に基づいて出力の木を作成する
- 必要な演算をリストアップする
- 最後に、入力の木と出力の木との対応付け、リストアップした演算を適切な箇所に当てはめて 1 つの木を作りプログラム構造を決定する

4. アルゴリズム設計支援システム

JSP 法は高度なアルゴリズム設計技術であるため、初学者には適さない。理由を以下に挙げる。

- 与えられた仕様を実現するのに必要な演算をリストアップするのが難しい
- 入出力の木の対応付けや、リストアップした演算を適切な個所に当てはめるのが難しい

そこで本研究では学習者にシステムによる支援をしながら JSP 法の一部の過程を行ってもらう。

4.1. システム概要

システムは図式エディタの機能を持ち、これにより Jackson 木の詳細化による構造化が保証されたアルゴリズム設計を行うことができる。また学習者には JSP 法の一環として入力の木と出力の木を作ってもらう。それ以外のプロセスはシステムが行う。学習者が入出力の木を作った後、それらを合わせて作られるプログラム構造の木をシステムが学習者に提供することで、学習者に入力の木と出力の木との関係を考察し、繰り返し学習してもらうことで JSP 法に慣れるようになることを目的とする。また、作成した Jackson 木は XML として保存可能である。

4.1.1. 支援内容

例題とともにシステムの支援内容を示す。詳細化支援として、表 2 ように入力の木の現段階に必要な詳細化に重要な文をハイライトしたり、図 1 のように入力データ構造を視覚的に表したりすることで支援を行う。学習者が図 2 で表される入力の木と図 3 で表される出力の木を完成させると、入力の木と出力の木と演算リストを合わせて作られた、仕様を実現するアルゴリズムを Jackson 木で表示する。(図 4)

表 2　例題 - ハイライトによる支援

文字列の入力解析を行う。入力は文字列として 1 回の標準入力とし文字列は 128 文字以下の半角英数字のアルファベットからなり、最後に.(ピリオド)が来る。 (1)　最初の a の前にある文字数を数える(カウント A) (2)　最初の a の後には b または c の文字の連続の入力があり b まとまりの数(カウント B)と c 単体の数を数える。(カウント C) (3)　文字列の解析後、それぞれのカウントを表示する。 例　入力が webbocgjabbbbccbbcccbbbbccccbb. の場合 a の前にある文字数(カウント A)　：8 a の後の b の文字の連続のまとまり(カウント B)：4 a の後の c の文字数(カウント C)　：9

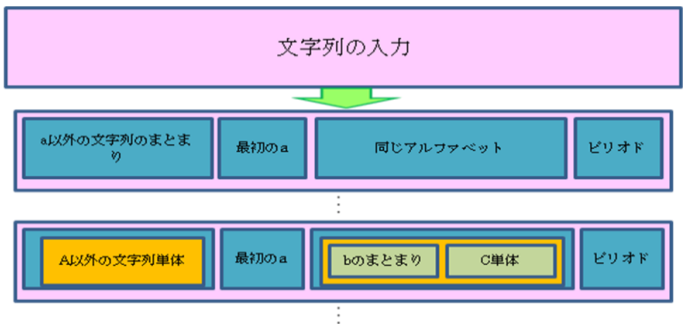


図 1　図による視覚的な支援

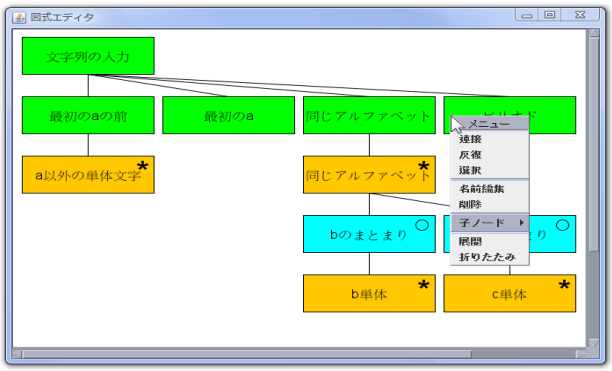


図 2　図式エディタ(入力の木)

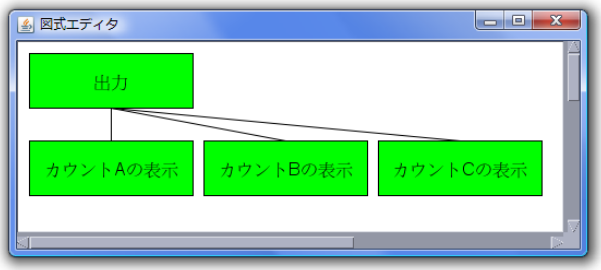


図 3　図式エディタ(出力の木)

表 2　要求された仕様に必要な演算のリスト(C 言語準拠)

```
1.  char array[128]
2.  int i = 0
3.  int countA = 0
4.  int countB = 0
5.  int countC = 0
6.  scanf("%s", array)
7.  countA++;
8.  countB++;
9.  countC++;
10. i++;
11. printf("countA = %d¥n", countA);
12. printf("countB = %d¥n", countB);
13. printf("countC = %d¥n", countC);
```

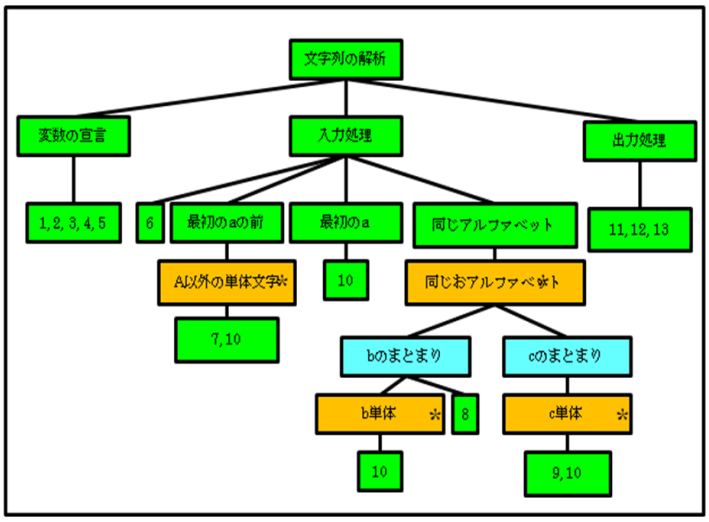


図 4　プログラム構造を表す Jackson 木

5. 評価方法

プログラミング言語の最低限度の文法知識を持つプログラミング初学者を２つのグループに分ける。例題を出題し、入力データ構造について片方のグループはフローチャートでの表し方を学習させ、もう片方のグループは Jackson 木での表し方とそれをフローチャートへ変換する方法を学習してもらう。その後、同じく入力データ構造表すという課題を与え、前者には直接フローチャートを書いてもらい、後者には一旦図式エディタを用いて Jackson 木を作成させてからフローチャートに変換してもらう。後者の方がより早く、正確に構造化されたフローチャートを書けることを確認する。

6. 今後の課題

本研究では JSP 法を一部のみしか支援を行わないので、より JSP 法を習得しやすくするために入力の木と出力の木の対応付けの支援など JSP 法に対して始めの過程から最後の過程までの支援を検討する必要がある。また、効率よく JSP 法を習得するためにどのような例題を組み込めばよいかを検討する必要がある。

参考文献

[1] Jackson A. M, “Principles of Programming Design”, Academic Press, 1975 (邦訳 鳥居宏次, 構造的プログラム設計の原理, 日本コンピュータ協会, 1980, pp.1-60) .
[2] O.-J.Dahl, E.W.Dijkstra, C.A.R.Hoare, "Structured Programming", Academic Press, 1972