論文

初級プログラミング教育における支援システムに関する研究
～因果マップエディタ～
Design and Implementation of Support System for Beginner's Programming Education -The Cause-Effect Map Editor-

堀内　幸造[1)]　Kozo Horiuchi
長田　一興[2)]　Kazuoki Osada

We aimed at the construction of the intellectual CAI system in the beginner's programming education and study about the definition of program specification[1)2)3)]. We expect it is mechanically decidable that students' programs are correct or not and which advices must be given when they are incorrect by comparing them to it. But it is insufficient if the specification is only comprised of the solution to realize our expectation. It must be associated with the common sense and knowledge to be used to build the program to give relevant advices[4)]. To express the common sense and knowledge in addition to the solution, we proposed a kind of design diagram combining the flow-chart and the data flow diagram, the Cause-Effect Map[1)2)]. Conventionally the flow-chart is drawn for human programmers to understand what and how to program and it is not directly machine-interpretable. By combining the data-flow diagram with it, the objects of each process can be explicitly expressed, but rigidness among the order of execution of processes still remains in the flow-chart.

By contrast, processes in the Cause-Effect Map are loosely coupled by just specifying their start conditions and resulted state changes of data which reflect some common sense or knowledge. In this paper we fully explain both of them and consider an experimental software to generate a program from the Cause-Effect Map.

キーワード：プログラムの自動生成, 知的CAI, 因果マップ
Keywords： automatic generation of programs, intellectual CAI, Cause-Effect Map

1　はじめに

我々は初級のプログラミング教育における知的CAIシステムの構築を目指し、プログラム課題の仕様化に関する研究を行ってきた[1) 2) 3)]。これら仕様が与えられれば、学習者が作成したプログラム(ソースコード)が課題の仕様を満たしているか判断できるようになり、学習者に適切な教授を行うことができる。しかし、単純に仕様化した課題と照らし合わせるだけでは有用なCAIシステムとしては不十分である。そこで、学習者がプログラミングという行為をどのように据え、思考し解を得ているか「プログラミングにおける常識と知識」という観点から考察してきた[4)]。これらプログラム課題の仕様化とプログラミングにおける常識と知識を表現する手法として我々は「因果マップ」を提唱し、この因果マップを描くことにより学習者への効果的な教授を実現することを目標とする。

プログラムを作成するために従来から利用されている図にフローチャートがあるが、フローチャートはあくまでも１つの解を表現する図であり、解の多様性には対応できない。そこでフローチャートをベースに個々のデータの状態や処理の前後関係を考慮し、各処理の間に自由な繋がりを持たせた図が因果マップである。同種の図にデータフロー図があるが、因果マップはデータの状態を3つの状態に分けて表現し、それらの状態によって多様な繋がりを持つことができる点で大きく異なっている。そのため学習者によって異なる多様な解にも対応でき、描かれた図は課題の仕様を可視化したものとなる。またデータの状態を見ながら処理が行われる過程が見えるためプログラムの流れの理解が容易になり、プログラミング教育においてはとても有用である。

溝渕らによる研究[6)]に因果マップによく似た図を利用し、プログラムを解析、依存関係を表現しているが、これらの図はプログラム文を主体に表現している。一方因果マップはプログラム文を処理=タスクとして表現し、さらに変数＝データを追加したことで、データ間の依存関係をより深く表現している。加えて、因果マップには起動条件が存在するのでプログラム課題の仕様を詳しく表現できる点で優

1)　産業技術研究科電子情報工学専攻研究生　horiuchi@olab.elec.fuk.kindai.ac.jp
2)　産業理工学部情報学科教授　osada@fuk.kindai.ac.jp

表1　処理30 種一覧

インクリメント	インスタンスの生成
ゲッターの呼び出し	スレッドの終了を待機
スレッド一時停止	スレッド起動
セッター呼び出し	その他条件ブロックの開始
デクリメント	メソッドを終了
メソッド呼び出し	繰り返しブロック
繰り返しブロックの開始	繰り返しブロックの終了
繰り返しを終了する	計算する
条件ブロック	条件ブロックの開始
条件ブロックの終了	親のコンストラクタ呼び出し
親のメソッド呼び出し	他のコンストラクタ呼び出し
代入する	入力する
配列呼び出し	表示する
例外を受ける	例外を受けるの開始
例外を受けるの終了	例外を投げる

れている。

これまでは「因果マップ」の仕様や描き方を中心に解説してきた[5]が、本文では具体的に因果マップを描くソフトウェアとして作成した「因果マップエディタ」について解説し、実際に描いた因果マップから正しいプログラムが得られることを確認する。

2　因果マップの処理

これまでの研究から現在想定している処理30種を表1に掲載する。

これら30種の処理の内、現在の因果マップエディタでは「表示する」「入力する」「代入する」「計算する」の4種類について動作している。よって本文では、本学部情報学科「オブジェクト指向プログラミング」の課題の中から課題2-4について検証することにする。課題2-4の内容は次のとおりである。

- 問題文

  英語、数学、国語の科目の点数と入力し、合計点と平均点を出力するプログラムを作成しなさい。

- 実行例

  英語の点を入力してください：52
  数学の点を入力してください：68
  国語の点を入力してください：76
  3科目の合計点は196点です。
  3科目の平均点は65.3点です。

課題2-4を手描きの因果マップで描くと図1のようになる。

この図を実際に因果マップエディタを用いて描き、最終的に課題の解を得るまでの流れについて解説する。

3　因果マップエディタの使い方

3.1　画面と機能

因果マップエディタは、上部にいくつかのボタンが並ぶツールバーと因果マップを描画する描画エリアから構成されている。ツールバー内のボタンは左から

- 選択
- ダイレクト選択
- 消しゴム
- プロパティ
- 図形追加
- グリッド
- タブの追加
- 生成

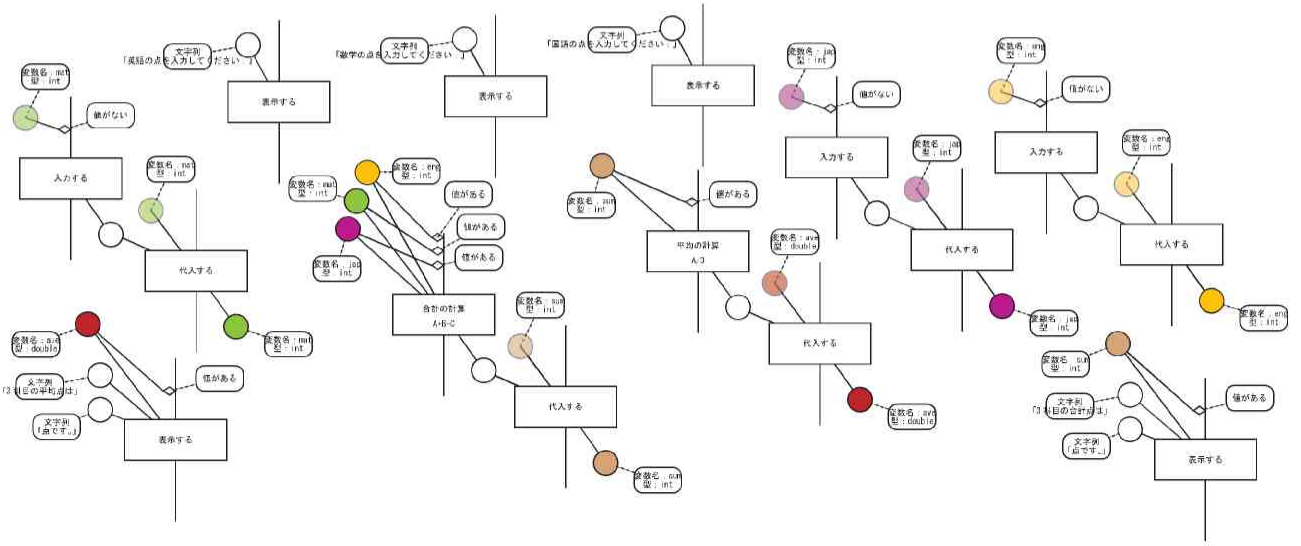

図 1　課題2-4 3教科の合計と平均 – 因果マップ

となっている。

選択は、タスクをひとつのまとまりとして選択し、全体を一度に移動させることができる。

ダイレクト選択は、タスクを構成する個々の図形、例えば入力の変数のみを選択し、個別に移動させることができる。

消しゴムは、選択したタスクを削除することができる。プロパティは、選択したタスクの情報を再編集することができる。

図形追加は、新しいタスクを追加することができる。グリッドは、画面上にグリッド線を表示し、図形を移動する際にグリッドにスナップさせ整列することができる。押すごとにグリッドのONとOFFを切り替えることができる。

タブの追加は、新しい描画エリアを追加することができる。まだ実装出来ていないが、新しいクラスの新しいメソッドを描く際にタブを追加して個別に描くことを想定している。

生成は、画面に描かれた因果マップに基づいてプログラムを生成する。

3.2 変数の追加

まず、図形追加ボタンを押し追加したい図形の詳細情報を入力する。図形追加ボタンを押すと図2のような画面になる。

上部にある「処理」で必要な処理を選択する。処理によって設定できる項目が異なるので、処理を変える毎に各項目が変化する。図2は初めて追加を行った時の図なので、入力のリストや起動条件のリストには何も変数が存在しない。変数を追加するには、画面下部の「変数の追加」ボタンを押す。変数の追加ボタンを押すと図3のような画面になる。

変数名や型、値に必要な情報を入力し「OK」ボタンを押すと、図4のように、各リストへ追加した変数が反映される。

3.3 文字列の追加

次に、処理「表示する」のように、入力に変数だけではなく文字列が必要な場合には変数の追加で図5のように、型を「STRING_V」とし表示させたい文字列を値に入力する。

この設定で変数を追加すると図6のように、各リストへ文字列がそのまま表示された状態で反映される。入力にここで追加した文字列「英語の点を入力してください：」を

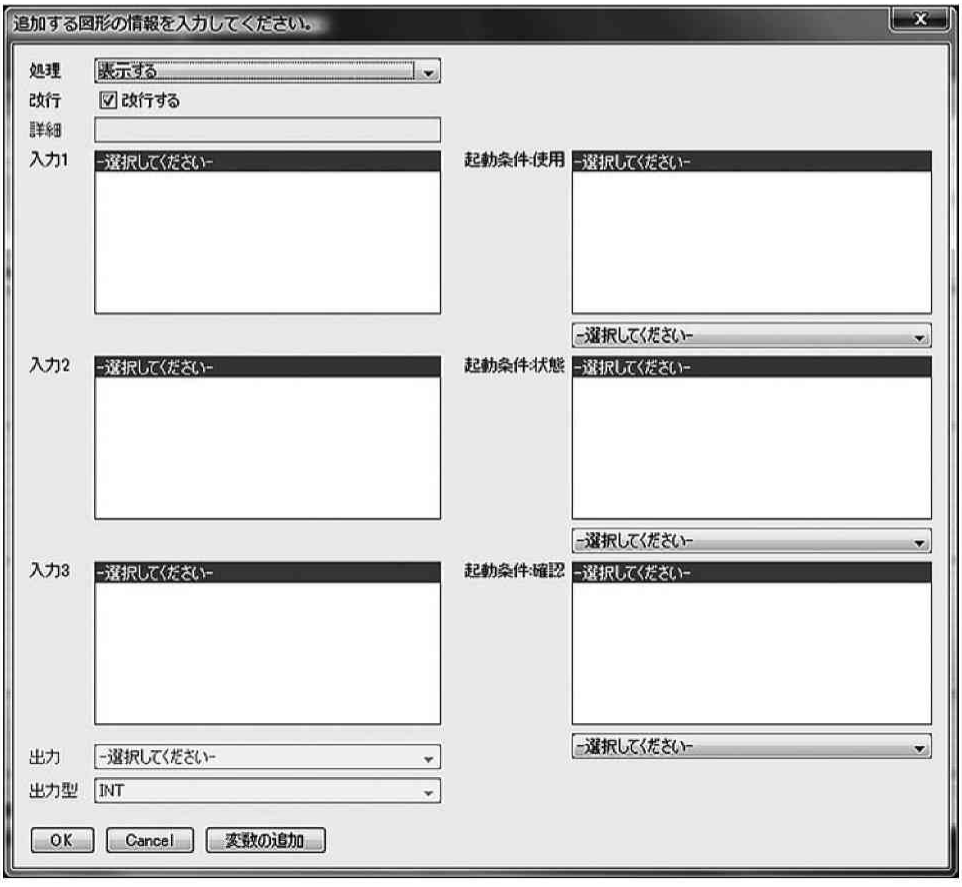

図 2　タスクの追加画面

図 3　変数の追加画面

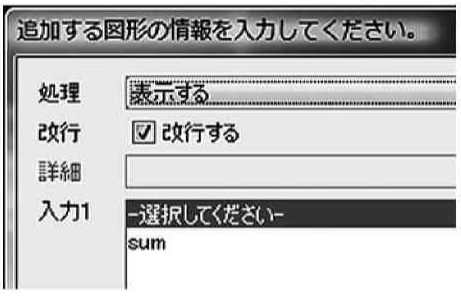

図 4　タスクの追加 - 変数の追加の反映

図 5　変数の追加 - 文字列

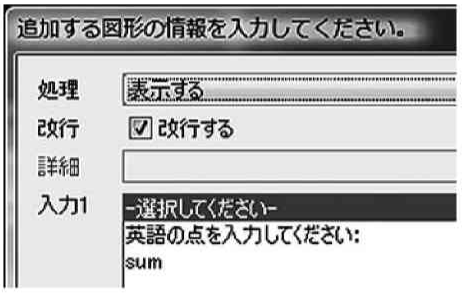

図 6　タスクの追加 - 文字列の追加の反映

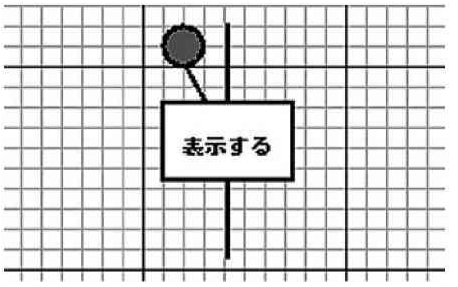

図 7　処理「表示する」の追加

図 8　関連付けのあるタスクの追加

図 9　処理「代入する」の追加

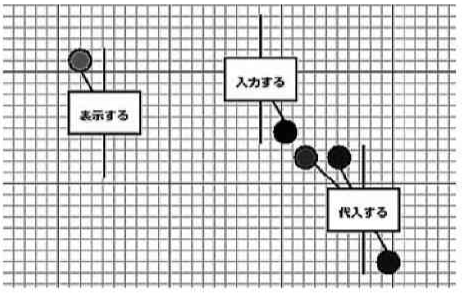

図 10　処理「入力する」と「代入する」の追加

選択し「OK」ボタンを押すと図7のようにタスクが描かれる。

3.4　関連付けのあるタスク

処理「入力する」と「代入する」、処理「計算する」と「代入する」のように2つのタスクが連続して必要なタスクの場合、出力の指定の仕方が少し異なる。図8のように「出力」ではなく「出力型」を指定するようになり、ここで処理がどのような型のデータを出力するかを選択する。

すると、図9のように処理「代入する」しか処理のリストに無い画面が表示され、さらに入力のリストに「INT」という表示が追加されている。

この「INT」は、関連付けのあるタスクが出力するデータを表しており、例えば処理「入力する」の場合、ユーザが入力したデータということになる。ここで、入力1のリストで変数sum、入力2のリストで「INT」を選択し「OK」ボタンを押すと図10のように入力したデータが変数sumへ代入

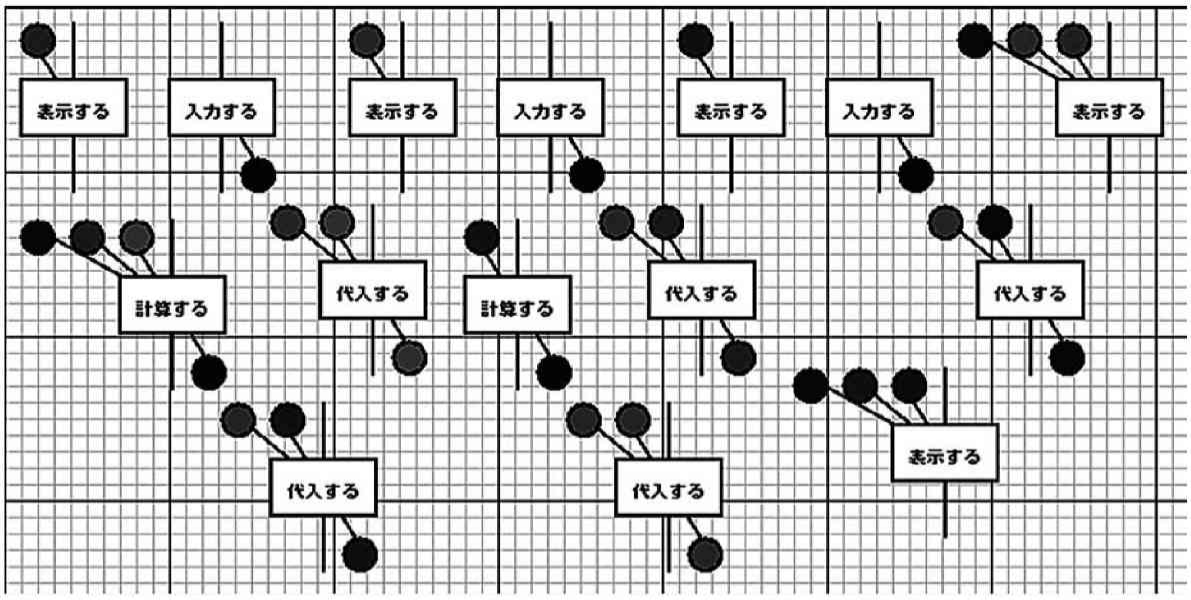

図12　課題2-4 3教科の合計と平均 - 因果マップエディタ版 因果マップ

図11　起動条件の設定

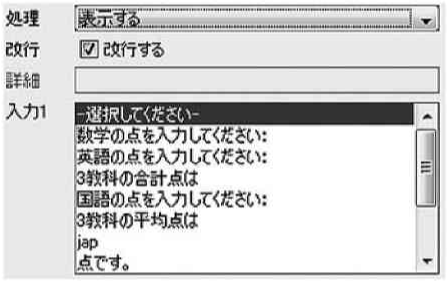

図13　課題2-4 3教科の合計と平均 - 追加した変数

される2つのタスクから成る”関連付けのあるタスク”として追加される。

3.5　起動条件の設定

変数を追加すると入出力のリスト以外に起動条件の各リストへも変数が追加される。図11のように追加するタスクに対して、必要な起動条件を選択することで設定できる。

現在のソフトウェアでは起動条件は描かれていないが、生成時の条件としては設定されている。今後図として描画する部分も対応予定である。

3.6　因果マップを描く

これらの操作を行い、図1を因果マップエディタで描いたものが図12になる。

図13に追加した変数や文字列のリストを示す。

細かい見た目については多少異なるが、起動条件なども含め図と同じ意味の図となっている。

3.7　プログラムの生成

図が完成したのち、ツールバー右にある「生成」ボタンを押すと図14のような画面に生成されたプログラムが表示される。

これにより、描いた因果マップから課題の解に当たるプログラムを得られることが確認できた。

4　まとめ

我々は初等のプログラミング教育における知的CAIシステムの構築を目指し、プログラム課題の仕様化とプログラミングにおける常識と知識を表現する方法として因果マップを提唱してきた。因果マップは複数のタスクから成り、タスクは複数の入力と出力、一つの処理という基本構成をベースに、柔軟な繋がりを持たせるためタスク後に起きるデータ状態の変化、タスク同士を繋ぐ根拠となる起動条件を同時に描ことができる。

これまで因果マップはすべて手で描き、絵に相当するプログラムを自力で記述し、生成エンジンへ引き渡し確認し

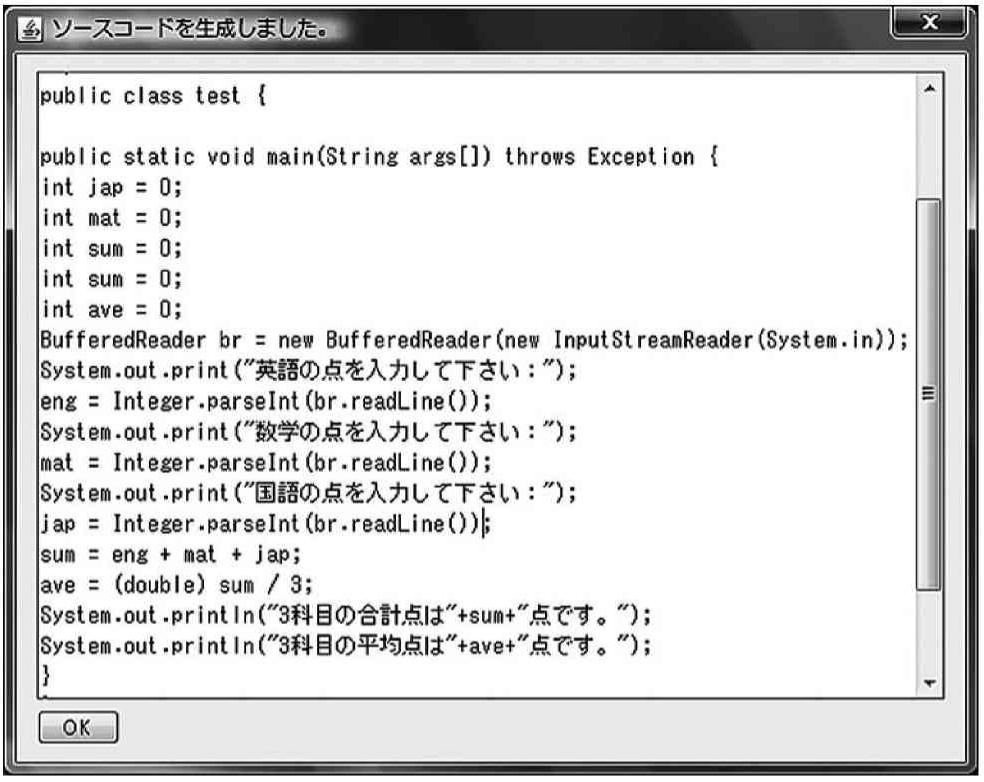

図14　課題2-4 3教科の合計と平均 - 生成したソース

ていたが、今回作成した因果マップエディタにより手軽に描くことができるようになり、また処理による設定項目の違いなどによる図の間違いを因果マップエディタ上で制御することで記述ミスがなくなった。これにより描く手間が大幅に減り、どのようなプログラムが生成されるか確認しながら描くことができるようになった。

今後は、すでに定義済みの処理について描けるタスクの種類を充実させていく予定である。

参考文献

1) 堀内幸造, 長田一興
モバイルエージェント技術を用いた教育支援システム- 常駐アプリケーションによるキーボードデータの収集, 近畿大学産業理工学部研究報告、 Vol.02, pp. 1-7, 2004.

2) 堀内幸造, 長田一興
初級プログラミング教育における支援システムに関する研究～因果マップを利用する仕様の定義～
近畿大学産業理工学部研究報告, Vol.03, pp. 21-26, 2005.

3) 堀内幸造, 長田一興
初級プログラミング教育における支援システムに関する研究～因果マップを利用したプロトタイプについての考察～
近畿大学産業理工学部研究報告, Vol.04, pp. 27-32, 2006.

4) 堀内幸造, 長田一興
プログラミングにおけるものの見方と常識について
近畿大学産業理工学部研究報告, Vol.05, pp. 45-49, 2006.

5) 堀内幸造, 長田一興
初級プログラミング教育における支援システムに関する研究～因果マップからJava ソースの自動生成について～, 近畿大学産業理工学部研究報告, Vol.11, pp. 24-30, 2009.

6) 溝淵裕司, 中谷俊晴, 佐々政孝
コンパイラ・インフラストラクチャを用いた静的プログラムスライシングツール, 日本ソフトウェア科学会第20 回大会論文集, 2B-3, 2003.