Vol.2016
2017/2/10

自律走行ロボットを題材とした プログラミング初学者向け教育環境の提案

佐藤真[†]　　武田敦志[†]

高等教育機関でのプログラミング教育では，学習者はアルゴリズムやプログラミング言語の文法などの基礎的要素を理解した上でプログラムを作成する．一方，小学校などの初等教育でプログラミング教育を行う場合，アルゴリズムやプログラミング言語の文法に対する学習者の理解度が低いため，従来のプログラミング言語を用いたプログラミング教育を行うことは困難である．そこで，本稿では，アルゴリズムやプログラミング言語の文法などの基礎的要素の理解を必要としないプログラミング教育環境を提案する．この教育環境を用いることにより，問題分割などの論理的思考力に焦点をあてたプログラミング教育が可能となると期待できる．

Programming Education Environment of Autonomous Robot Control for Beginners

Makoto Sato[†]　and　Atsushi Takeda[†]

In programming education system at higher education institutions, students have to understand algorithms, data structures and grammars of programming language before they make a completed program. However, in elementary school education, it is difficult to teach programming techniques by using programming language because it is hard for elementary school students to understand algorithms or grammars of programming language. Therefore, we propose a new programming education environment where students can accomplish a program without knowledge of algorithms or grammars. By using the proposed environment, teachers achieve an effective programming education which focuses on logical thinking such as problem reduction.

1. はじめに

現在，大学などの高等教育機関においてプログラミング教育が行われている．これらのプログラミング教育の多くは，C 言語や Java などのプログラミング言語を用いて実用的なプログラムを作成することを目的としている．一方，2020 年度から小学校などの初等教育におけるプログラミング教育の必須化が検討されている．正しく動作するプログラムを作成するためには，実行したい動作を明確に表現する必要がある．そのため，プログラミング教育を通じて，物事を分析して整理する「論理的思考力」が養われると考えられている[1].

C 言語や Java などのプログラミング言語を用いることで実用的なプログラムを作成できる．しかし，これら言語を用いてプログラミングを作成するためには，目的の動作を行うために必要となるアルゴリズムやデータ構造を整理し，これらを正しい文法で正確に記述する必要がある．そのため，プログラミング初学者に対してプログラミング言語を用いたプログラミング教育を行う場合，アルゴリズムやプログラミング言語の文法に対する理解度が低いために効果的な教育を行うことが難しいという問題がある．特に，小学校などの初等教育では，学習意欲の低い生徒も存在するため，アルゴリズムや文法の理解を前提としたプログラミング教育は難しいと予想される．そこで，この問題を解決するため，プログラミング言語の文法を把握しなくてもプログラムを作成できる Scratch [2]などのプログラミング教育環境が提案されてきた．これらの教育環境では，プログラムの論理構造をブロック図として表現できるため，プログラムを作成するためにプログラミング言語の文法を理解する必要がない．そのため，小学校などの初等教育の生徒に対しても効果的なプログラミング教育が可能となる．しかし，これらの教育環境では，分岐処理や繰り返し処理などの基本的な論理構造のみの組み合わせとしてプログラムを作成するため，簡単なゲームや単純な数値処理しか開発できないという問題がある．一方，現在の社会で求められる「論理的思考力」とは，複雑な問題を分析し，その問題をより単純な複数の問題に分割し，分割した各問題の解決方法を考案する能力である．上記のプログラミング教育環境では，学習者は基本的な論理構造を理解できるため，問題の解決方法を考案する能力を養うことができる．しかし，これらの教育環境では単純なプログラムしか作れないため，複雑な問題をより単純な複数の問題に分割する「問題分割能力」を養うことは難しい．

そこで，本稿では，学習者の「問題分割能力」の向上に着目したプログラミング教育環境を提案する．提案する教育環境では，処理の連鎖とパラメータの設定のみで自動走行ロボットの制御プログラムを作成できるようになっており，学習者はアルゴリズムやプログラミング言語の文法を理解していなくてもロボットを走行させるための

[†] 東北学院大学教養学部情報科学科
Department of Information Science, Tohoku Gakuin University

プログラムを作成できる．また，複雑なロボットの動作をより単純な動作に分割し，これらの単純な各動作のパラメータと実行順序を指定することにより，複雑なロボット制御プログラムを作成することも可能である．これにより，「コーディング能力」や「アルゴリズムの理解度」の低い学習者であってもロボットを自分の設定したとおりに動かすことができるようになり，学習者のモチベーションを損なわずに「問題分割能力」を養うことができ，論理的思考力を養うことができると考えられる．

2. 問題分割に着目したプログラミング教材

まず，本研究では従来の教育手順を逆にして，本来の目的である論理的思考力を養うための「問題分割能力」を養うことを第一とする（図 1　提案するシステム）．これにより，従来の問題点であった「コーディング」と「アルゴリズムの理解」におけるモチベーションの低下を避けることができる．また，問題分割能力を先に養うことで，複雑な問題と直面したときに，その問題を分割することができること，さらにそのひとつひとつは単純なアルゴリズムで解決できることを理解することができる．

次に，提案するシステムを実行するにあたり，制御プログラムについて考える．一般的なロボット制御プログラムは「行動選択」・「行動実行」・「終了条件の確認」に 3 つを繰り返すことでロボットの動作を制御している．「行動選択」では，次の行動を判断するためにまわりの状況を確認し，それに応じた行動を選択する．「行動実行」では，まわりの状況に応じて選択された行動を実行する．その際に，パラメータを状況に応じて変更することで，ロボットに適切な行動を指示することができる．「終了条件の確認」では，行動を実行しながらまわりの状況を確認して行動終了の判断をする．まわりの状況が終了条件に達するまではその前に選択された行動を繰り返し実行し，終了条件に達すると行動を終了して次の行動の選択へ移る．これらの手順を繰り返し行うことで，ロボットはまわりの状況に応じた行動を実行できる（図 2　制御プログラムの基本構造）．

3. 自律走行ロボットを題材とした初学者向けプログラミング教育環境

本稿ではロボット制御プログラムの基本構造が一定であることに着目した．プログラムの型をあらかじめ定めておくことにより，学習者がパラメータのみを制御して，それに応じたプログラムを自動で編集し出力させることができるシステムを提案する．この提案システムでは学習者がパラメータのみを制御するため，本来プログラムを完成させるために必要な「コーディング」や「アルゴリズムの理解」といった学習状況に捉われずに，実際にロボットを動作できるようになる．結果として，初学者がプログラミングに対し壁を感じることなく，目的を達成するために必要な要素を考えることができる．

例として，走行ロボットの制御について考える．今回用いる走行ロボットのパラメータの種類は図 3 のとおりである（図 3　走行ロボットのパラメータの種類）．これらの中からそれぞれパラメータを選択し，ロボットはそれに応じた動作をする．提案システムではこれらのパラメータを選択・入力するだけで，それに応じたプログラムを出力することができる．今回提案するシステムでは，パラメータのみを制御して動作させることを目的としている．次の図 4 は走行ロボットが「5 秒間直進して，その後停止する」プログラムでの例である（図 4　提案するシステムの利点）．これにより従来のブロック型に必要だった繰り返し処理を省くことができ，「アルゴリズムの理解」に関わらず目的に沿った動作をさせることが可能となる．

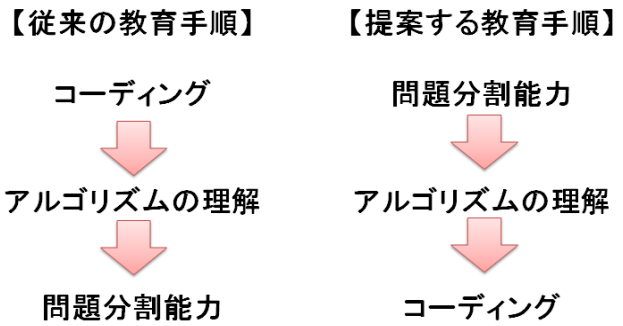

図 1　提案するシステム

図 2　制御プログラムの基本構造

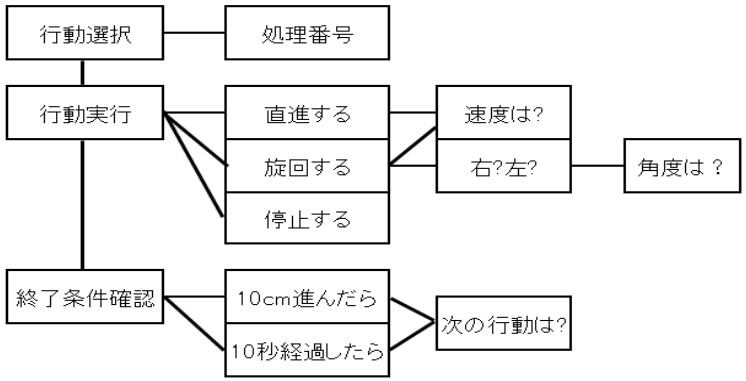

図 3　走行ロボットのパラメータの種類

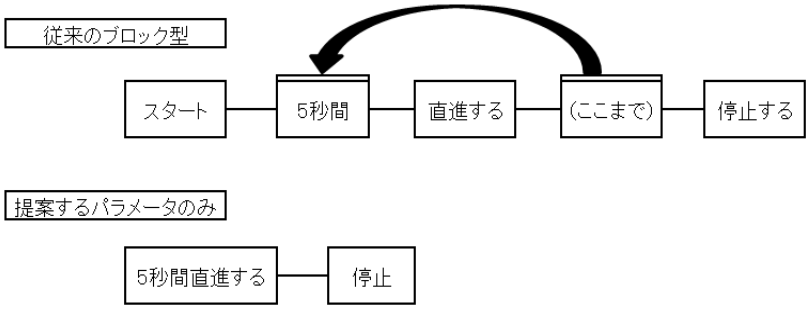

図 4　提案するシステムの利点

4. 実装と動作検証

4.1 提案システムの実装

Microsoft Excel 2010 上で作成されたシートにパラメータを入力し，csv ファイルとして出力されたそのデータを Eclipse で組んだ Java アプリケーションで受け取り，パラメータに合わせて選択された cpp ファイルとして出力することができる仕組みを実装した．また，作成された cpp ファイルをコンパイルすることで Toppers/EV3RT [3]上で動作する実行ファイルを作成する．このファイルを LEGO Mindstorms EV3 [4]で実行することにより，自律動作可能なロボットを動作させることができる．

4.2 提案システムの動作手順

提案システムでは以下の 3 つの行程により実際にロボットを動作させるプログラムを出力することができる（図 5　提案するシステムの動作手順）．これにより，「Excel 上でのパラメータの操作」，「Java アプリケーション上での操作」，「コンソール上での変換」の 3 つの行程でロボットの実働プログラムを出力することができるため，コーディングやアルゴリズムの理解といった学習段階に関係せず，目的に合わせて選択した動作を確認することができる．

4.3 提案システムの動作検証

例として，走行ロボットが「5 秒間直進し，2 秒間旋回することを繰り返す」プログラムを作成する．この場合の制御構造と操作するパラメータは図 5 のようになる（図 6　例題のパラメータ）．このパラメータをマクロにより csv ファイルとして出力し，さらにその csv ファイルを Java アプリケーションで読み込み，パラメータを変更した走行ロボットのプログラムファイルとして出力する（図 7　例題より出力した「1（直進）」のプログラムの一例）．実際にそのプログラムファイルを実働ファイルへ変換し，Mindstorms EV3 で動作確認したところ，変更したパラメータ通りの動作が確認できた．また，さらに一部のパラメータのみを変更して同様に動作検証したところ，選択したとおりの動作が確認できた．

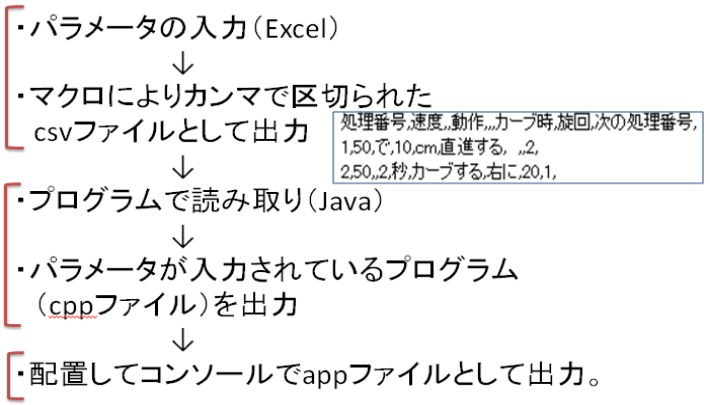

図 5　提案するシステムの動作手順

行動選択	0(開始)	1(直進)	2(カーブ)
行動実行		直進する	右に旋回する
終了条件確認	ボタンが押されたら 1に移る	10cm移動したら 2へ移る	2秒経過したら 1へ移る

処理番号	速度	動作		カーブ時		旋回	次の処理番号
1	50	10	cm	直進する			2
2	25	2	秒	カーブする	右に	20	1
3			cm	直進する			
4			cm	直進する			
5			cm	直進する			

図 6　例題のパラメータ

```
void s02Rule::run(FilterManager* filterManager, DeviceManager* deviceManager)
{
        float speed = 25;
        float turn = 20;
        float gyro = deviceManager->getSensor()->getGyroRate();
        float left_count = deviceManager->getLeftMotor()->getCount();
        float right_count = deviceManager->getRightMotor()->getCount();
        float battery = deviceManager->getBody()->getBatteryVoltage();
        signed char left_pwm;
        signed char right_pwm;
        balance_control(speed, turn, gyro, 0, left_count, right_count, battery, &left_pwm, &
        deviceManager->getLeftMotor()->setPower(left_pwm);
        deviceManager->getRightMotor()->setPower(right_pwm);
}

int32_t s02Rule::next(FilterManager* filterManager, DeviceManager* deviceManager)
{
        get_tim(&endTime2);
        double margin = (double)(endTime2 - startTime2) / CLOCKS_PER_SEC;
        if(margin > 20.0){
                return s03Rule::NUMBER;
        }
        else{
                return s02Rule::NUMBER;
        }
```

図 7　例題より出力した「1（直進）」のプログラムの一例

5. まとめ

本稿では，従来のプログラミング教育の問題点を挙げ，その問題を解決するためのシステムを提案した．この提案システムでは，「コーディング」や「アルゴリズムの理解」を必要とせずに，パラメータのみを制御することで実際にロボットを動作させることができる．したがって，従来のプログラミング教育で問題となっていた，「コーディング」及び「アルゴリズムの理解」を学習する過程における初学者のモチベーションの低下等の問題を解消し，初学者が本来の目的である「論理的思考力」を初めに養うことができるようになると考えられる．

参考文献

1) 文科省，プログラミング教育実践ガイド, 2014.
2) Scratch, https://scratch.mit.edu/
3) Toppers/EV3RT, http://dev.toppers.jp/trac_user/ev3pf/wiki/WhatsEV3RT
4) LEGO Mindstorms EV, https://www.lego.com/ja-jp/mindstorms