Blockly を用いた多言語プログラミング学習支援環境の構築

佐野裕也[*1]　香川考司[*2]
*1 香川大学大学院工学研究科
*2 香川大学創造工学部

Construction of multi-language programming Learning Environment using Blockly

YUYA SANO[*1]　KOJI KAGAWA[*2]
*1 Graduate School of Engineering, Kagawa University
*2 Faculty of Engineering and Design, Kagawa University

概要: 学習者が第 2, 第 2 のプログラミング言語を学習するとき, 基礎概念と文法を短時間に学習しなければならない. これは, 学習者にとって大きな負担である. これを解決するために本研究ではブロック方式の初学者用プログラミング環境である Blockly を採用し, Blockly のブロックの動的変形機能を利用して, 手続き型以外の言語に対するプログラミング学習支援環境の構築を試みる.

キーワード: プログラミング学習, Blockly, 学習支援, Web ベース

1. はじめに

プログラミング学習者は, プログラミングの概念と言語の文法を同時に学習しなければならない. これは, 学習者にとって大きな負担である. この負担を軽減するために, 文法を意識せずにプログラミングができる学習環境が必要である. これを解決するために, Web ベースグラフィカルプログラミングエディタである Blockly を用いる.

尾崎の研究[(1)]は, Blockly を C 言語, Flex 言語（字句解析器生成系、Adobe Flex とは異なる）に対応させたもので, システムの対象者がプログラミング入門者である. これにより文法を意識せずにプログラミンングを学ぶことができる. しかし, 大学の授業で学習する言語に対応しきれていない. また, 動的に変更できるブロックが限られているため, 柔軟性のあるプログラム言語をブロックの形状によって制約されてしまうことになる.

著者らの所属する大学と大学院の授業で学習するプログラミング言語には, C, Java, JavaScript, PHP, Haskell, Bison/Flex などがあるが, いずれの言語もそれぞれの固有の文法を持っており, その文法を踏まえてテキストエディタからソースコードを記述するまで理解するにはかなりの時間と負担がかかる.

本研究では Blockly をこれらの言語へ対応することを目標とする。これにより Blockly で学習支援できる範囲が広がり, 学習者が第 2, 第 3 の言語を迅速に学習することができる. しかし, これらの言語では式の入れ子が深くなりがちなので, ブロックの形を動的に変形させることも考えなければならない.

2. Blockly とシステム構成

2.1 Blockly について

Blockly[(2)]とは, Google で開発されているグラフィカルなプログラミングエディタであ. 図 1 のようなブロックを繋ぎ合せることでプログラミングを行う. このため構文エラーに悩まされず, 直感的にプログラミングをすることができる. JavaScript で記述されており, ドキュメントも豊富に用意されているためカスタマイズが容易である. また, Blockly は Web ベースのアプリケーションであるため, 学習者側の導入の作業が不要である.さらに、Blockly で作成したプログラムは, JavaScript, Dart, Python, Lua, PHP の 5 種類のコードに

変換して出力することができる.

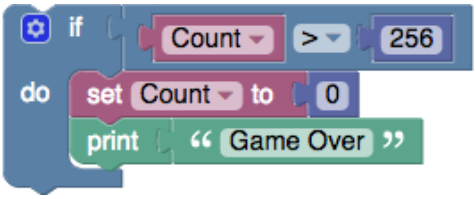

図 1. Blockly のブロック

2.2 システム構成

Blockly は，通常，ワークスペース部，ブロックメニュー部，ソースコード部， XML コード部の 4 つのコンポーネントによって構成される．図 2 に Blockly のスクリーンショットを示す.初期状態では，ワークスペース部が表示されており，ソースコードタブを押すとソースコード部のコンポーネントに切り替わり，XML コードタブを押すと XML コード部のコンポーネントに切り替わる．ソースコード部は，作成したプログラムのソースコードを表示するスペースであり，XML コード部は，作成したプログラムの XML コードを表示するスペースである．ブロックメニュー部は，定義されたブロックが存在するスペースで，常に左部に表示されている．

図 2. Blockly のスクリーンショット

2.3 Mutator 機能

Blockly の機能にMutator がある. Mutator は,ブロックの動的変形を行う機能である．この機能は, Blockly が用意している機能の中で唯一の動的変形である．Mutator の機能が使えるブロックには，左上に歯車がある場合がある．その歯車のマークを押すと，その近くに吹き出しの形をした小窓が現れる．小窓の左半分はブロックメニュー部，右半分はワークスペース部となっている．図3にこの時のスクリーンショットを示す．小窓の中のコンポーネントは，このブロック限定のものである．小窓のブロックメニュー部から拡張したいブロックを取り出し，小窓のワークスペース部の既存のブロックに取り付ける．すると，吹き出し元のブロックが変形される．図4が，そのときのスクリーンショットである. Mutator機能によって，ブロックの形をカスタマイズでき，ブロックメニュー部で用意されるブロックの種類を大幅に減らすことができる.

図3. 動的変形前のMutatorブロック

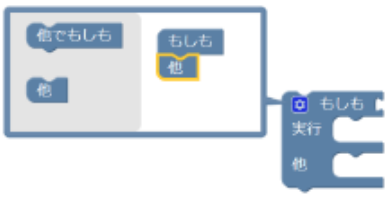

図4. 動的変形後のMutatorブロック

2.4 Blockly の対象言語と新たに導入する言語の違い

2.4.1 flex の正規表現の場合

算譜 1 は, flex の正規表現の例である．正規表現には特別な意味を持つ文字があり，適したブロックが Blockly には存在しないので,特殊文字に対して新たにブロックを用意する必要がある.

```
\"(w|\\["\\w])*\"
```

算譜 1. flex の正規表現のコード

2.4.2 Haskell の内包表記の場合

算譜 2 は, Haskell のリスト内包表記の例である．リスト内包表記では，右側に変数を限定する式を記述し，左側にその変数を使った式を記述する．このリストの表し方は, Blockly で既存に用意されているリストブロックでは表すことができない．そこで，リスト内包表記を表すことができる新たなブロックを用意する必要がある.

```
[ (x, y) |  x <- [0..n], y <- [x..n] ]
```

算譜 2. Haskell のリスト内包表記のコード

3. 実装

本研究では, Blockly を C, Haskell, flex に対応させるために機能の拡張や新たなブロックの定義を行った. 本章では、これらについて詳しく説明する.

3.1 新たに実装した動的変形機能

本システムで新たに実装した動的変形機能をいくつかのブロックで実装した.

3.1.1 C 言語の printf ブロック

従来の Blockly 出力用ブロックには、C 言語の printf のように文字列中に式を埋め込む機能がなかった.

そこで, C 言語の printf ブロックで, 新たに動的変形機能を実装し, 複数の式を文字列中に埋め込んで出力できるようにした. C 言語システムの入出力ブロックカテゴリで用意されている. そのブロックのイメージは, 図 5 である.

入力フォームで"%"の数を検出して, その数だけソケットの数を動的変形機能で増やしている. ソケットとは, 変数ブロックや数ブロックを挿入できる穴である. 検出のタイミングは, 入力フォームの中身に変化があるごとに行われる. この機能は, いちいち歯車マークをクリックして別途でブロックを組み立てる必要もないので, 動的変形の手間を省きかつ操作もシンプルになって, プログラミング学習者の負担を軽減させることができる.

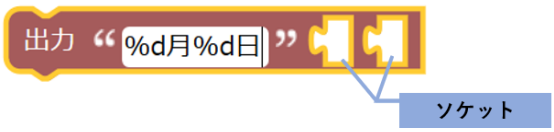

図 5. 動的変形後の printf ブロック

3.1.2 flex の文字クラスブロック

文字クラスブロックにも新たな動的変形機能が実装されており, flex システムの正規表現ブロックカテゴリで用意されている. そのブロックのイメージは, 図 6 である.

文字クラスブロックでは, Mutator で 3 種類の動的変形を行うことができる. Mutator の子ブロックの範囲ブロックを繋げると, 2 つの入力フォームが出現し, 1 組の文字範囲を入力できるようになる. 文字ブロックは, 1 つの入力フォームが出現し, 任意の 1 文字を入力できる. 特殊文字では、選択フォームが出現し, 5 種類の特殊文字を選択できるようになっている.

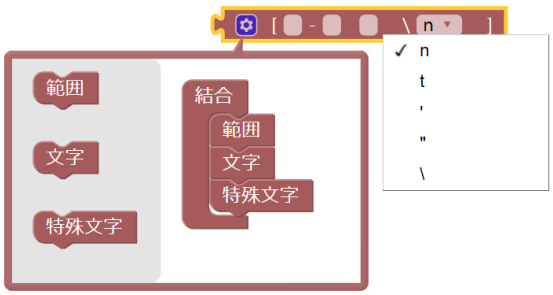

図 6. 文字列任意ブロック

3.2 新たに実装したブロック

本小節では, 新たに実装したブロックについて説明する.

3.2.1 Haskell のリスト内包表記ブロック

Haskell 言語システムのリストカテゴリで新たにリスト内包表記ブロックを実装した. リスト内包表記ブロックのイメージは, 図 7 である.

図の左が, リスト内包表記ブロックに何も接続していない初期状態で, 右がリスト内包表記ブロックの接続例である. リスト内包表記のソースコードは, リストの中に 1 つの式と複数の限定式で記述される. リスト内包表記ブロックでは, 内包表記と書かれた右のソケットに式を挿入し, 限定式と書かれたところの右に複数の限定式を表すブロックを挿入する仕様となっている.

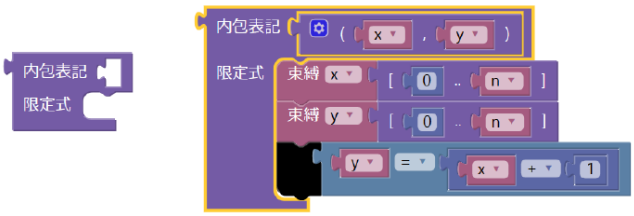

図 7. リスト内包表記ブロック

3.2.2 flex の正規表現ブロック群

flex システムでは, 先行研究で尾崎が実装したブロックを参考にしながら正規表現のブロックを一新した. 図 8 の連接ブロックでは, 動的変形機能でソケットの数を増やして, 複数の正規表現を表すことができる. 図 9 の繰り返しブロックでは, 選択フォームで繰り返しの意味を示す演算子を選択できるようになっており, 正規表現の繰り返しを表すことができる. ブロックの表現は, 実際の flex のキーワードや演算子を使用した形式としている.

これでユーザは，構文エラーに悩まされることなくプログラミングを行うことができる．

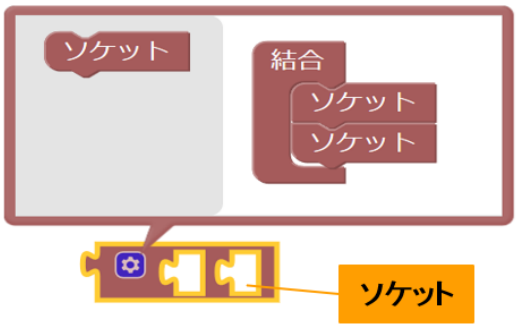

図 8. 連接ブロック

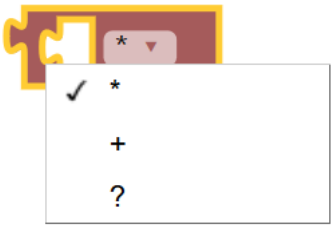

図 9. 繰り返しブロック

3.3 新たに実装した表記切り替え機能

Flex 言語のシステムに新たに表記切り替え機能を実装した．表記切り替え機能のイメージは，図 10 に示す．切り替えは，自然言語を用いた表記（ここでは日本語表記と呼ぶ）とプログラミング言語の演算子やキーワードを用いた表記（ここでは flex 表記と呼ぶ）の 2 種類があり，選択フォームで切り替えを行う．

例えば，ある正規表現を Blockly のブロックを使って表現したいときに，日本語表記の場合は図 11 のようになる．初心者にとっては意味は分かりやすいが，これでは，システムの画面に入りきらず，右が見切れてしまっている．そこで，画面内にブロックを収めるために，flex 表記を実装した．そのイメージが図 12 である．

この実装によって，それぞれのブロックパーツの概念とプログラミング言語での構文や演算子との対応も理解できるようになっている．

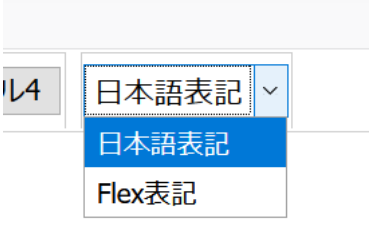

図 10. 表記切り替え機能

図 11. ある正規表現の日本語表記

図 12. ある正規表現の Flex 表記

3.4 サンプルボタン機能

Blockly を初めて使う人にとって，ワークスペース部が何も無い状態でブロックを一から組み立てて，実行できるプログラムを完成させることは難しい．そこで，このシステムの初心者が利用しやすくするためにサンプルボタンを実装した．図 13 は、サンプルボタンを押したときのワークスペース部のイメージである．WEB ページの上部にサンプルボタンを複数個用意した．このサンプルボタンを押すことによって，システム開発者側が用意したブロックを一瞬でワークスペース部に表示される．システム初心者は，その完成されたブロックのプログラムをアレンジしながらシステムを理解することができる．

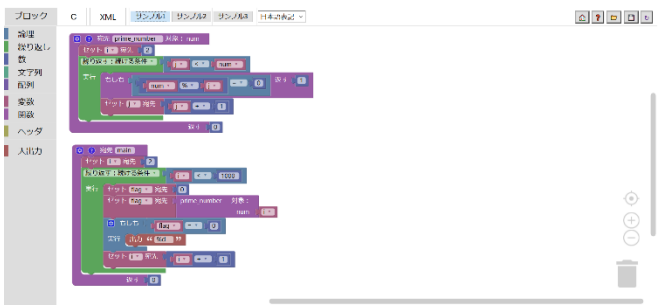

図 13. サンプルボタンを押したときのワークスペース部

3.4.1 問題点

サンプルボタンを実装したときに問題が発生した. 動的変形機能付きのprintfブロックがサンプルボタンを押したときに初期状態のままで出力される. つまり, ソケットがない状態でブロックが出力される. 入出力ブロックの動的変形機能は, 入力フォームの%の数を検出してソケットの数を増やすのだが, その検出のタイミングは入力フォームに何か変化があるときである. サンプルボタンを押して, 入出力ブロックを表示させると入力フォームの変化するタイミングがないため, ソケットも初期状態のままである.

3.4.2 改善点

サンプルボタンの問題点を改善するために, JavaScript の LocalStorage[3] という API を利用した. LocalStorage とは, データをブラウザ側に保存する仕組みである. 同様の有名な API に Cookie があるが, Cookie には保存の有効期限があるのに対し, LocalStorage には有効期限がない. また, Cookie よりもより多くのデータ量を保存できる.

4. 評価

情報系学科の学生5名を対象に, 実際にC言語のシステムとHaskell言語のシステムを使用してもらい, その後以下の評価項目に自由に回答する形式で行った.

- 操作方法は直感的に分かったか
- 使ったブロックとそのブロックの評価
- 欲しい機能やブロック
- 不具合報告

「操作方法は直感的に分かったか」という項目では, C言語のシステムで「サンプルボタンを押すと完成されたプログラムをカスタマイズすることができ, 操作方法が直感的に分かった」など, 肯定的な回答が見られた. 一方, Haskell言語のシステムでは「どういう風につなげて良いかとかどう関数を作ったらいいかなどが分からなかった」などといった否定的な回答が多く見られた.

「欲しいブロックとそのブロックの評価」という項目では, 「%で出力変数を動的に変更できるのは良いと思った」といった回答が得られ, printfブロックの評判は良かった.

「欲しい機能やブロック」の項目では, 「C 言語の実数の変数ブロック」, 「C 言語のビット演算ブロック」, 「Haskell の関数ブロックの改良」といった数々の回答が得られた.

5. まとめ

大学と大学院の授業で学習する言語に対応させるために, Web ベースグラフィカルプログラミングエディタ Blockly の flex と Haskell への対応を行った. その際に、ブロックの種類が多くなりすぎないように動的変形を利用した拡張を行った. この拡張によって, 柔軟性のあるプログラミング言語をブロックの形状によって制限されないようになっている. また, 拡張した機能も学習者の負担を増やさないようにしている. しかし, 実際に授業で使用してもらっていないので, 今回の評価で得られたフィードバックをもとにシステムを改良し, 授業で効果を確認する必要がある.

6. 今後の課題

大学の授業で効果を確認できるようなシステムを目指すために, 以下に本システムの課題を述べる.

6.1 Blockly の対応言語を増やす

本システムで対応させた言語は, C, Haskell, flex である. 本システムの対応言語が本大学の授業で学ぶ言語にすべて対応できているとはいえない. そこで, 次の対応言語として, Java, Rubyの実装が必要である.

6.2 C 言語のシステムの配列に対応させる

本システムの C 言語システムには, 配列に対応させることができなかった. JavaScript 言語システムでは, 既存のシステムでリストが対応しており, Haskell 言語システムでは, リストを本システムで初めて実装した. これらを参考にしながら, C 言語の配列に対応させる必要がある.

6.3 Haskell 言語のシステムの対応構文を増やす

Haskell言語のシステムは, 著者らの所属する大学・大学院の授業に対応したシステムにするために大学院での授業「プログラミング・パラダイム」の内容に沿って作成した. しかし, 授業の内容すべて対応させ

ることはできず，本システムの対応する構文は限定的なものとなってしまった．そこで，do式，let式，case式，関数のカリー化にも対応させる必要がある．

6.4　ブロック接続部の改善

flex言語システムで接続できないブロックがある．内包表記ブロックの限定式のところで，直接式ブロックを接続しようとしても，接続の形状的に接続することができない．そこで，急遽，接続できないときのために作られたブロックを用意した．以下の図14の黒色のブロックが急遽，用意したブロックである．このブロックがワークスペース部にある状態で，XMLコードタブを押しワークスペース部に戻るとエラーでプログラムが出力できなくなる．この問題を解決するためにも，ブロックの接続部の改善を検討しなければならない．

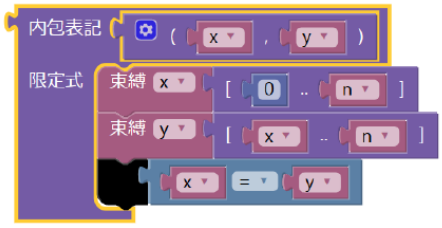

図14. 接続できないときのために作られたブロックのプログラム例

謝辞

システムの評価に協力して頂いた，大橋健太さん，仁科陽彦さん，朝野有也さん，伊藤拓海さん，太田圭祐さんに深く感謝します．

本研究はJSPS科研費15K01075の助成を受けたものである．

参　考　文　献

(1) 尾崎陽一・香川考司,“Webベースグラフィカルプログラミングエディタを用いたFlexプログラミング環境の開発”，第38回全国大会（JSiSE2013）講演論文集, TF1-1

(2) Fraser, N. “Google Blockly” – a visual programming editor. https://developers.google.com/blockly/, (参照 2018-09-18).

(3) HTML standard, “LocalStorage”, https://html.spec.whatwg.org/multipage/webstorage.html, (参照 2018-09-18)