

Blockly によるプログラミング学習支援環境の多 言語対応の研究



香川大学工学部電子・情報工学科	
卒 業 論 文	
卒 業 年 度	平成 29 年度 (2017 年度)
指 導 教 員	香 川 考 司

香川大学工学部 電子・情報工学科

佐野 裕也

平成 29 年 2 月 00 日

This is the English Title.

Abstract This is the abstract. This is the abstract. This is the abstract. This is the abstract.
This is the abstract. This is the abstract. This is the abstract. This is the abstract. This is the
abstract. This is the abstract. This is the abstract. This is the abstract. This is the abstract.
This is the abstract. This is the abstract. This is the abstract. This is the abstract. This is the
abstract. This is the abstract. This is the abstract. This is the abstract. This is the abstract.
This is the abstract. This is the abstract. This is the abstract.

あらまし プログラミング初心者が新たなプログラミング言語を学習するとき、プログラミングの基礎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。これを解決するために尾崎の研究では、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 C 言語に対応させた。また山形の研究では、Blockly に練習問題を提示する機能を実装した。本研究では、Blockly を多言語対応し、ブロックの動的変形の機能を増やして、プログラミング初心者の学習の幅を広げることを目的としている。

キーワード プログラミング学習, C 言語, JavaScript, Haskell, Flex, Web ベース

目次

1	はじめに	1
1.1	はじめに	1
1.2	Web ベースの利点	2
1.3	システムに求められる要件	3
2	Blockly	4
2.1	全体の主なファイル構成	4
2.2	システム構成	5
2.2.1	ワークスペース部	5
2.2.2	ブロックメニュー部	7
2.2.3	ソースコード部	7
2.2.4	XML コード部	8
2.3	オプション機能	8
2.4	Mutator 機能	10
3	実装	11
3.1	多言語化	11
3.1.1	対応する言語の種類	11
3.1.2	システムの WEB ページ	12
3.2	動的変形機能	12
3.3	新たに作成したブロック	13
3.3.1	Blockly for C	13
3.3.2	Blockly for Flex	14
3.3.3	Blockly for Haskell	15
3.4	サンプルボタン機能	18
4	評価	19
4.1	項目	19
4.2	結果	19
5	おわりに	20
5.1	まとめ	20
5.2	今後の課題	20
5.3	ビジュアルプログラミング言語 Processing	20

謝辞	22
参考文献	23
付録 A プログラムの全ソース	24
A.1 ファイル名	24

第 1 章


はじめに

1.1 はじめに

プログラミング学習者は、以下の 2 つを同時に学ばなければならない。

- プログラミングの基礎概念
- 各プログラミング言語の文法

これらを学ぶことは、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 ベースのビジュアルプログラミング言語である Blockly  を用いる。

Blockly 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る。図 1 のようなブロックを繋ぎ合わ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、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。

尾崎の研究は、Blockly を C 言語、Flex 言語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。このシステムによって文法を意識せずにプログラミングを学ぶことができる。しかし、大学の講義で学習する言語に対応しきれていない。システム自体が便利であっても、大学の講義で学習する言語に対応していないと、大学生はこのシステムを利用することができない。また、ブロックの形を動的に変形することができないため、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約されてしまうことになる。尾崎の研究で実装したシステムのイメージを図 1.1 に示す。

本研究では Blockly の多言語化を目標とする。多言語化によって、より Blockly で学習支援できる範囲が広がり、学習者が自分に適した言語を選ぶことができる。しかし、必要となるブロックの種類や形状は増加するので、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。

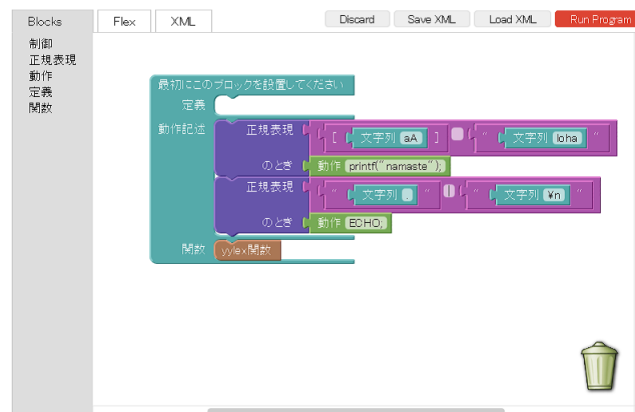


図 1.1: 尾崎の研究で実装したシステム

1.2 Web ベースの利点

本研究で開発したシステムは、Web ベースシステムである。Web ページすることにより、学習者に次のような利点があると考えられる。

- システムのインストールが不要
- Web ブラウザがあるば実現可能
- 常に最新版のシステムが可能

また、教員側にも次のような利点がある。

- システムのカスタマイズが容易
- システムの不具合をすぐに修正可能

- 導入に関する指導が不要

サーバ側にアプリケーションを設置することで、クライアント側ではアプリケーションを導入する必要がなく、Web ブラウザがあれば学習者にとって、簡単にそして常に最新のシステムが利用可能である。教員にとってもシステムのカスタマイズが容易で、不具合もすぐに修正できるので、システムの運営がとてもしやすくなる。

1.3 システムに求められる要件

先行研究の問題点より、本システムに求められる要件は以下の 3 つである。

- プログラミング学習者の負担を軽減させる
- 大学の講義で学習する言語に対応させるための多言語化
- ブロックの動的変形を拡張

既存の Blockly の対応言語が、大学の講義で学習する言語に対応しきれていないという問題点を踏まえて、システムの多言語化が必要である。システムの多言語を行うことで必要となるブロックの種類が増加する。また、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約されてしまう。これらに対応するために、ブロックの動的変形機能を拡張する。その際に、プログラミング学習者の負担を軽減させることが目的なので、拡張する機能を複雑にしてはいけない。

本論文では、第 2 章で Blockly について、第 3 章で実装したシステム、第 4 章でシステムの評価、第 5 章でまとめ、今後の課題について述べる。

第 2 章

Blockly



2.1 全体の主なファイル構成

- Library フォルダ

システム自体の定義を行うプログラムで構成されている。また、blockly フォルダには、Blockly の本体である blockly_compressed.js ファイルが存在する。いずれも Google が用意した元からあるファイルである。

- js フォルダ

こちらもシステム自体の定義を行うプログラムで構成されている。システムの拡張を行いたい場合はこちらから行う。

- MyBlock フォルダ

このフォルダの直下にブロックの種類ごとのファイルがあり、ブロックの形状を定義している。いずれも Google が用意した元からあるファイルであり、これらのプログラムはビルドされ、Library フォルダの blocks_compressed.js ファイルに圧縮されている。そのため、これらのファイルを書き換えてもシステムが変更されることはない。

- MyBlock2 フォルダ

このフォルダの直下にブロックの種類ごとのファイルがあり、こちらもブロックの形状を定義している。このフォルダの中のファイルは私自身が作成したもので、新たなブロックを定義したり、ブロックの形状を変更したいときはこちらにプログラムを記述する。

- Mygenerators フォルダ

このフォルダの直下に各プログラミング言語のフォルダがある。それらのフォルダの中にブロックの種類ごとのファイルがあり、各ブロックを配置したときに出力されるソースコードの内容が書かれている。

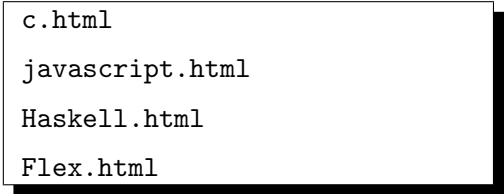
- html フォルダ

Web ページの仕様を定義するプログラムで構成されている。

- index.html ファイル

私が作成したデモページのホーム画面の仕様を定義しているファイルである。

- デモページのファイル群



```
c.html
javascript.html
Haskell.html
Flex.html
```

これらのプログラムは、デモページを構成している html ファイルである。

2.2 システム構成

Blockly は、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 2.1 に Blockly のイメージを示す。デフォルトでは、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。ブロックメニュー部は常に左部に表示されている。ここでは、それぞれのコンポーネントについて説明する。

2.2.1 ワークスペース部

ワークスペース部は、ブロックを使ってプログラミングを行うスペースである。図 2.2 にワークスペース部のイメージを示す。ユーザは、ブロック部に存在するブロックをワークスペース部にドラッグ&ドロップで自由につなぎ合わせることでプログラミングを行う。ワークスペース部の右下にはゴミ箱アイコンが存在する。ワークスペース部に設置したブロックをこのアイコンにドラッグすると、ブロックを削除することができる。その上に拡大縮小アイコン、さらにその上に中央に寄せるアイコンが表示されている。

また、ワークスペース部に設置したブロックを右クリックすることで、以下のような動作が行える。



図 2.1: Blockly

- ブロックの複製
- コメントの追加
- ブロックの接続表現の切り替え
- ブロックの表現の簡略化
- ブロックの透明化
- ブロックの削除
- ブロックの動作を説明する Web ページを表示

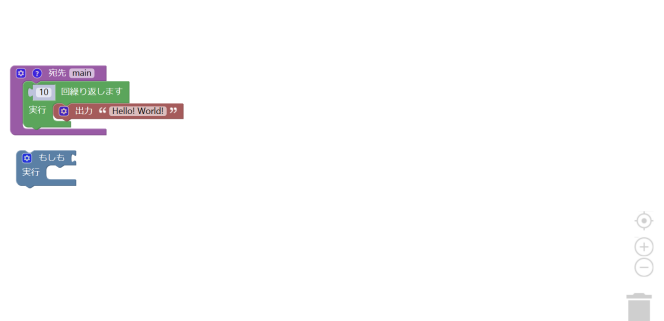


図 2.2: ワークスペース部

2.2.2 ブロックメニュー部

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ある。図 2.3 にブロックメニュー部のイメージを示す。ブロックは、論理、数、リストなどのカテゴリーにそれぞれ格納され、カテゴリーをクリックすると、そのカテゴリーに格納されたブロックが表示される。図 2.4 にカテゴリーをクリックしたときのイメージを示す。ユーザは、このブロックメニューから新たに必要なブロックを選択して使用する。



図 2.3: ブロックメニュー部

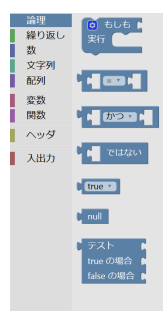


図 2.4: カテゴリーをクリックしたとき

2.2.3 ソースコード部

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る。図 2.5 にソースコード部のイメージを示す。タブ名は各プログラミング言語の名称になる。ワークスペース部でブロックを組み合わせて作成したプログラムが、リアルタイムにソースコードに変換され、このソースコード部でいつでも確認することができる。この機能によって、Blockly でプログラミングを学んだ学習者がソースコードを記述できるように支援する働きを持つ。システムの拡張の際に新たなブロックを実装した場合は、新たなブロックの定義と共に新たにソースコードも定義しなければならない。



図 2.5: ソースコード部

2.2.4 XML コード部

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。図 2.6 に XML コード部のイメージを示す。ワークスペースで組み合わせたブロックの構造が XML 形式で出力され、その出力された XML コードをセーブ、ロードすることができる。この機能によって、組み合わせたブロックを保存したり、他の学習者や教員が組み合わせたブロックを自分のワークスペース上に再現することができる。

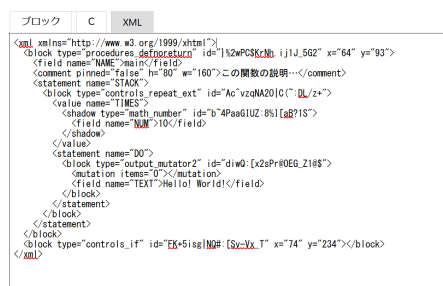


図 2.6: XML コード部

2.3 オプション機能

Blockly には、オプション機能が充実している。図 2.7 にオプション機能のイメージを示す。これは、システム WEB ページの右部にあるものである。図の上部 (システム WEB ページ上では右上部) に 5 つのボタンが表示されてある。左から、ホーム、ヘルプ、読み込み、保存、リセットである。図の下部 (システム WEB ページ上では右下部) に 4 つもアイコンがある。下から、削除、縮小、拡大、中央に寄せるである。

- ホームボタン

ホームボタンは、私が開発したデモページのホームページへ移動できるようになっている。これは、私が新たに実装したボタンである。そしてオプション機能で以後説明するシステムは、Google が用意した既存にある Blockly のシステムである。



図 2.7: オプション機能

- ヘルプボタン

ヘルプボタンは、このシステムを初めて使う人向けの機能である。ボタンをクリックすると、順追ってシステムの各機能を説明してくれるようになっている。

- 読み込みボタン

読み込みボタンは、過去に Blockly で保存したプログラムのデータを読み込み、ワークスペース部に出力できる機能である。前回に中断したプログラムの続きを行うことができる。

- 保存ボタン

保存ボタンは、ワークスペース部で組み立てたプログラムを保存することができる。保存形式は、XML である。プログラムを保存しておけば、次回 Blockly を立ち上げたときに、プログラムの続きから行うことができる。

- リセットボタン

リセットボタンは、ワークスペース部で組み立てたプログラムをリセットして何もない状態から再開できるボタンである。このボタンをクリックする前に、組み立てたプログラムを一旦、保存しておくことを推奨する。

- 削除アイコン

削除アイコンは、ワークスペース部にあるブロックを削除できる機能である。操作方法として、削除したいブロックをドラッグして削除アイコン上でドロップすることで削除できる。また、同様の機能として、削除したいブロック上で右クリックした際のコンテキストメニューにも削除がある。

- 拡大・縮小アイコン

拡大・縮小アイコンは、組み立てたプログラム全体を拡大・縮小することができる機能である。大規模なプログラミングを組み立てるときにはなくてはならない機能である。

- 中央に寄せるアイコン

このアイコンは、ワークスペース部で組み立てたプログラムを中央に寄せることができる機能である。

2.4 Mutator 機能

Blockly の機能に Mutator がある。ここでは、その Mutator という機能について説明する。

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 の用意されてる機能の中で唯一、用意されている動的変形である。図 2.8 に Mutator のイメージを示す。Mutator の機能が使えるブロックには、左上に歯車のマークがある、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。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 2.9 が、そのときのイメージである。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。実装で動的変形機能の拡張を行うときは、この機能を参考にして新たな動的変形機能を実装する。



図 2.8: 動的変形前の Mutator ブロック

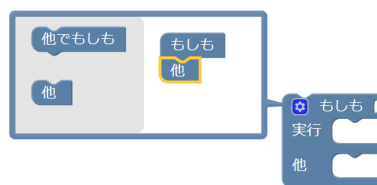


図 2.9: 動的変形後の Mutator ブロック

第 3 章

実装

本研究では、Blockly で多言語化を行うために、以下のような実装を行った。

3.1 多言語化

3.1.1 対応する言語の種類

まず、Blockly の多言語化を行うために、対応するプログラミング言語の種類を決めなければならない。本システムでは、4 種類のプログラミング言語に対応している。それが以下の 4 種類のシステムである。

- Blockly for C
- Blockly for JavaScript
- Blockly for Haskell
- Blockly for Flex

JavaScript 言語は、Blockly の既存のシステムに用意された言語である。また、C 言語では先行研究で尾崎と山形が、Flex 言語では、尾崎が開発したシステムを引き継いで実装を行った。Haskell 言語では、私が提案した言語で Blockly で初めて導入されるプログラミング言語である。



3.1.2 システムの WEB ページ

Blockly の既存のシステムでは、1 つの WEB ページで Blockly の対応する言語をサポートしている。1 つの WEB ページに対応言語名が書かれた複数のソースコードタブが存在し、ワークスペースでブロックを組み立てた後に、ソースコードを表示したい言語のタブを押してソースコードを確認するようになっている。しかし、どのソースコードタブを押してもブロックカテゴリーはすべて同じなので、Blockly に対応するすべての言語に共通できるブロックしか作成することができない。

本システムでは、多言語化を行うために 1 種類の言語に、1 つの WEB ページを用意した。また、私が開発した WEB ページにスムーズにアクセスできるように、それぞれの言語のシステムのリンク先を貼り付けたデモページの索引ページも作成した。索引ページのイメージが図 3.1 である。

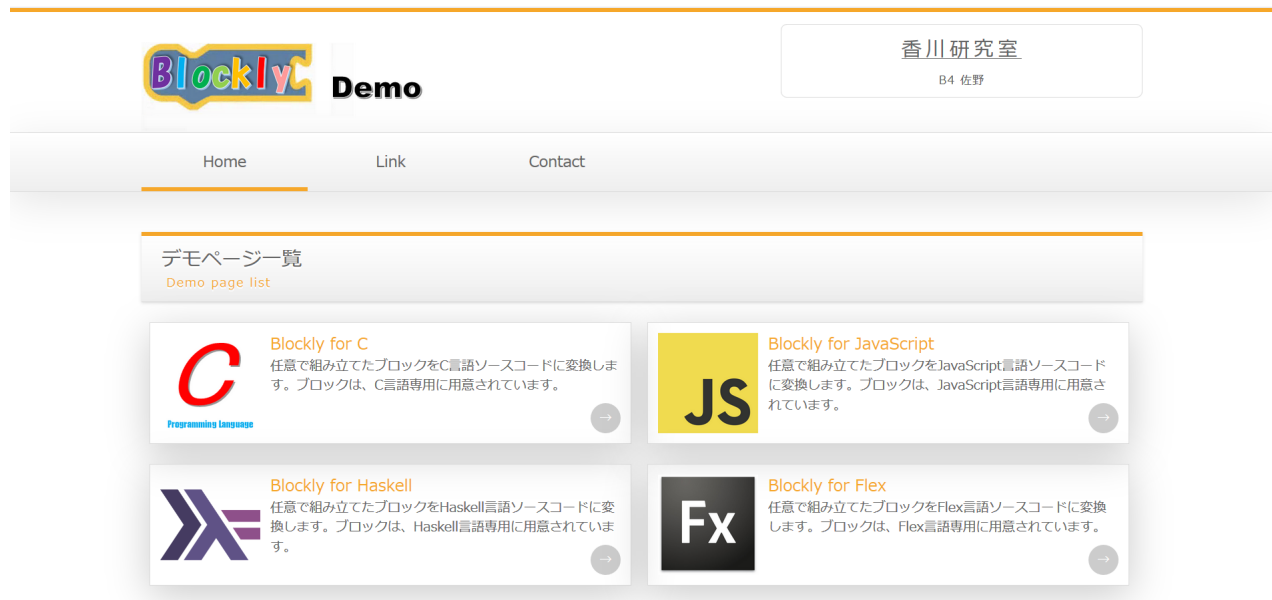


図 3.1: 索引ページ

3.2 動的変形機能

Mutator は、Blockly の既存に用意されている機能だが、本システムで新たに実装した動的変形機能を具体的なブロックを例に挙げて説明する。

本システムで新たに開発した動的変形機能は、出力ブロックに実装されている。その出力ブロックのイメージは、図 3.2 である。このブロックは、Blockly for C で用意されている。Mutator 機

能では、ブロックの左上に歯車マークがあり動的変形機能が実装されていることが分かるが、このブロックは歯車マークの代わりになるものがないので、動的変形機能が実装されていることが分からない。しかし、このブロックの入力フォームに図 3.3 のような文字列を入力すると、動的変形機能が実装されていることが分かる。これは、入力フォームで”%”の数を自動検出して、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。この機能は、Mutator 機能と違って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いるので、プログラミング学習者の負担を軽減させることができています。



図 3.2: 動的変形前の出力ブロック

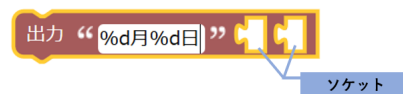


図 3.3: 動的変形後の出力ブロック

3.3 新たに作成したブロック

ここでは、本システムで新たに実装したブロックについて分類ごとに分けて説明する。

3.3.1 Blockly for C

本小節では、Blockly for C で新たに実装したブロックについて説明する。

- 入力・出力ブロック

プログラムにおける入力、出力の処理を表すブロックである。ブロックカテゴリーに格納されている状態では入力フォームのみだが、動的変形機能によって入力フォームに入力された”%”の数を自動検出してソケットを増やしていく。以下の図 3.4 に、入力・出力ブロックを示す。

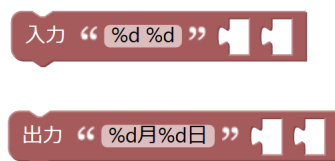


図 3.4: 入力・出力ブロック

- ヘッダブロック群

C 言語のヘッダファイルを表すブロック群である。Blockly for C のブロックカテゴリーのヘッダに格納されている。以下の図 3.5 に、ヘッダブロック群を示す。

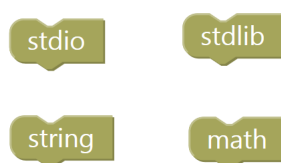


図 3.5: ヘッダブロック群

3.3.2 Blockly for Flex

本小節では、Blockly for Flex で新たに実装したブロックについて説明する。

- 正規表現のブロック群



Blockly for Flex 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。ソケットを使うことで、複数の文字範囲を正規表現で表せるようになっているが、2 個が限界である。動的変形機能を使って 3 個以上の文字範囲が実装できるように開発中である。ブロックの表現は、実際の Flex の文法の記述形式としている。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。以下の図 3.6 に、正規表現のブロック群を示す。

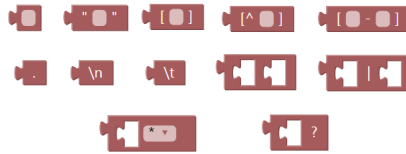


図 3.6: 正規表現ブロック群

3.3.3 Blockly for Haskell



本小節では、Blockly for Haskell で新たに実装したブロックについて説明する。

- もしも実行ブロック

もしも実行ブロックは、他の言語のシステムでも既存に用意されてるブロックだが、Blockly for Haskell では、Haskell の文法の性質上、違うので新たに用意した。C 言語や JavaScript 言語では、“if”文のあとに“if else”文や“else”文がくるので Mutator 機能が必要になるが、Haskell 言語は、“if then else”文しかないので、ブロックを変形させる必要がない。よって、Mutator 機能を除去した。また、“if then else”文の“then”と“else”のあとには、式がくるので、ブロックのつなぎ目の形を変形させた。以下の図 3.7 に、Blockly for Haskell で新に実装したもしも実行ブロックを示す。

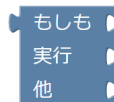


図 3.7: もしも実行ブロック

- 論理ブロック群

Blockly for Haskell でブロックカテゴリーの論理で新たにブロックを実装した。main ブロック、do ブロック、let ブロックである。以下の図 3.8 に、Blockly for Haskell の実装した論理ブロック群を示す。

- リスト内包表記ブロック

Blockly for Haskell のリストカテゴリーで新たに内包表記ブロックを実装した。以下の図 3.9 に、内包表記ブロックを示す。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、

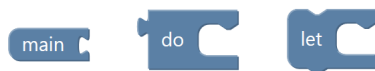


図 3.8: Haskell の論理ブロック群

内包表記と書かれた右のソケットに式を挿入式し、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。

[式 | 限定式, ... , 限定式]

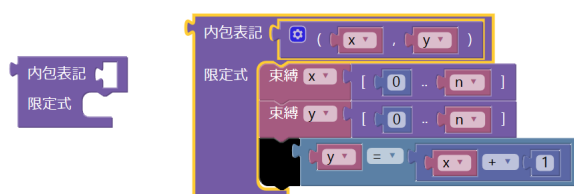


図 3.9: リスト内包表記ブロック

● 関数ブロック

Blockly for Haskell のリストカテゴリで新たに関数ブロックを実装した。以下の図 3.10 に、関数ブロックの何も接続していない初期状態と関数ブロックの小窓の中身を示す。関数ブロックでは、Mutator 機能でブロックを動的に変形させることができる。この Mutator 機能で関数の仮引数を自由に設定することができ、~~その設定方法は~~小窓のなかの型ブロックを引数コンテナに組み立てることで仮引数を調整できる。その際に、小窓の中の型ブロックには入力フォームがあり、直接入力することで仮引数の型を設定できる。また、引数コンテナに接続された一番下の型ブロックは、関数ブロックの返却値となり、関数ブロックを動的変形に変形しない仕様となっている。

● 関数呼び出しブロック

関数呼び出しブロックは、関数ブロックを右クリックしてコンテキストメニューで取り出すことができる。このブロックのソケットは、実引数を表し変数ブロックや数ブロックを挿入する。関数呼び出しブロックは、ブロックカテゴリ部にはない唯一のブロックである。関数ブロックの仮引数の数と関数呼び出しブロックの実引数のソケットの数は動的に対応でき



図 3.10: 関数ブロックの初期状態と小窓の中身

るようになっている。以下の図 3.11 に関数呼び出しブロックを示す。

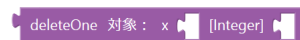


図 3.11: 関数呼び出しブロック

- 関数ブロックと関数呼び出しブロックの接続例

図 3.12 に、関数ブロックと関数呼び出しブロックの接続例を示す。図の関数ブロックでは、小窓のなかの引数コンテナに型ブロックが 3 個接続されている。そして、本体の関数ブロックには、仮引数の接続部が 2 か所ある。これは、小窓のなかの引数コンテナに接続されている一番下の型ブロックが、関数の返却値を示しているためである。このブロック状態でのソースコードは以下のようになっている。

```
deleteOne n (x:xs) = if n == x then xs else (x:deleteOne n xs)
```

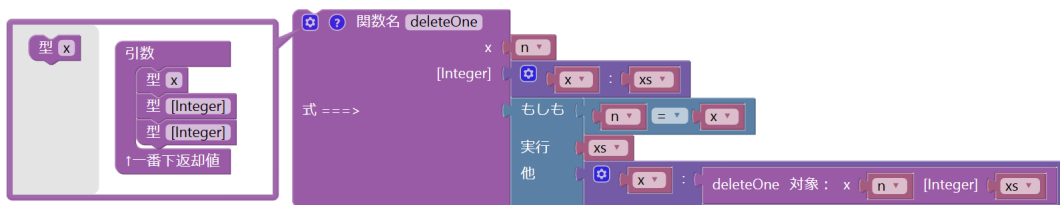


図 3.12: 関数ブロックと関数呼び出しブロックの接続例

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 3.13 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。

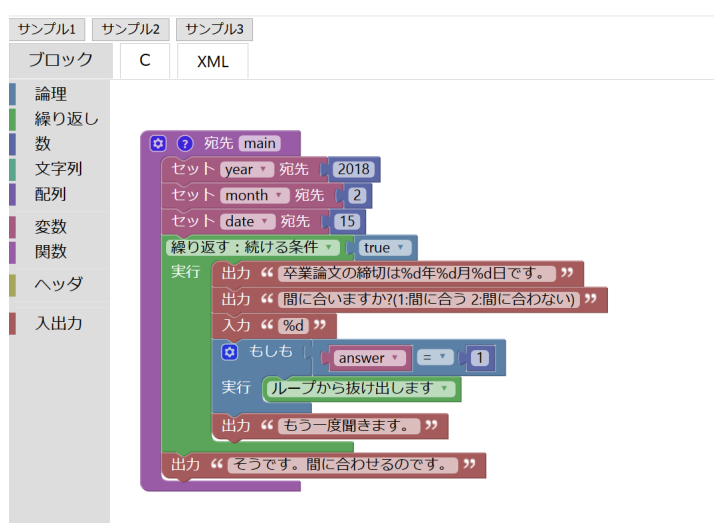


図 3.13: サンプルボタンを押したときのワークスペース部

第 4 章

評価

香川研究室の学部生 3 名と院生 2 名を対象に、本システムの評価を行った。本章では、システム評価について述べる。

4.1 項目

実際にシステムを使用してもらい、その後以下の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（無ければ書かなくてよい）

4.2 結果

序論を書いてください。

文字だけで埋めるとこのような感じになります。

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0

第 5 章

おわりに

5.1 まとめ

大学の講義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 の多言語対応を行った。その際に、ブロックの種類が多くなりすぎないように Mutator 以外の動的変形の機能の拡張も行った。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、拡張した機能もシンプルで、学習者の負担も増やさないようにしている。これらの特徴は、第 1 章で述べたシステムに求められる要件を満たしているといえる。

本研究で実装したシステムのイメージを図 fig: に示す。

5.2 今後の課題

序論を書いてください。

5.3 ビジュアルプログラミング言語 Processing

こちらも Blockly と同様に、グラフィック系に特化しており、プログラミング初心者扱いやすいものとなっている。ブロックのような視覚的なものでプログラミングを行い、随時ソースコードと実行結果が出力されるようになっている。ソースコードの文法は、Java と非常に似た独自の文法を採用している。学習者の対象がアート系のプログラミング初心者で、Blockly よりも学習者のターゲットが狭い。

このシステムの特徴として、ライブプログラミングという機能を導入している。通常ならば、プログラムを書く コンパイルする 実行する、という手順を踏んでプログラミングを行う。ライ

プログラミングでは、プログラムの実行中でもソースコードの変更を許容し、ソースコードの編集がリアルタイムに実行結果となってフィードバックされる。

しかし、このシステムの問題点として、ブロックの種類がとても少なく、ブロック自体を変形させる機能がないので、学習できる範囲がとても限定的ということである。

謝辞

本研究においてご指導を承りました香川考司先生に心からの感謝の意を表します。
また、システムの評価に協力して頂いた、香川研究室の皆様に深く感謝します。

参考文献

- [1] 著者, “論文タイトル,” 出典, ページ, 年.
- [2] 著者, “本タイトル,” 出版社, (通巻,) 年.

付録 A

プログラムの全ソース

A.1 ファイル名

% ソースの実体