OCaml Blockly のキー操作の実装

高越 莉菜　(指導教員：浅井 健一)

1　はじめに

現在、本学の関数型言語の授業では、OCaml のブロックプログラミング環境である OCaml Blockly が使用されている。これは、先行研究 [2] にて Google が提供する Blockly [1] を参考にして作成されたもので、関数型言語の初学者が陥りやすい型エラーや構文エラーが起こらないよう、ユーザインタフェースで制限する。現段階では授業ではマウス操作のみが可能であり、簡単な操作には適しているが、複雑なコードを生成しようとするとマウスでの操作が面倒になる。また、undo 機能に対応しておらず作業を戻したい時に手間がかかる。さらに、使いこなせるようになってきて型エラーや構文エラーが起きなくなってきて必要性を感じなくなってきた人は OCaml Blockly を使わなくなる。

本研究では、このような人々にも使い続けてもらうために、便利な機能拡張をしようと考え、undo 機能とキー操作を実装させた。

2　OCaml Blockly

OCaml Blockly は、関数型言語である OCaml のビジュアルプログラミング環境である。具体的には図 1 のように、OCaml の構文がブロックで表現されており、型ごとにコネクタの形が変えられている。型が違うブロックをくっつけようとしてもくっつかない仕組みになっている。また、

- 穴が見えやすく、構文エラーに陥ることなくブロックが組める
- 型の違うところにはブロックが接続できないため型エラーが起きない
- 変数名の変更が楽

などのメリットもある。

3　各機能の問題と改良点

本研究では、以下の 2 つの機能を改良した

1. Undo 機能
2. キー操作でブロック組立

改良点の説明に先立ち、OCaml Blockly において主要となる用語の説明を以下にする。

コネクタ　ブロックとブロックを接続する切り欠きを表すオブジェクト。キー操作においては矢印キーでコネクタがハイライトされる。

ワークスペース　（以下、WS）ブロックの組み立てやドラッグ移動などを自由に行うことのできる空間のこと。各ブロッ クは必ず 1 つの WS に属している。

メインワークスペース　WS の中で一番根本にある WS のこと。

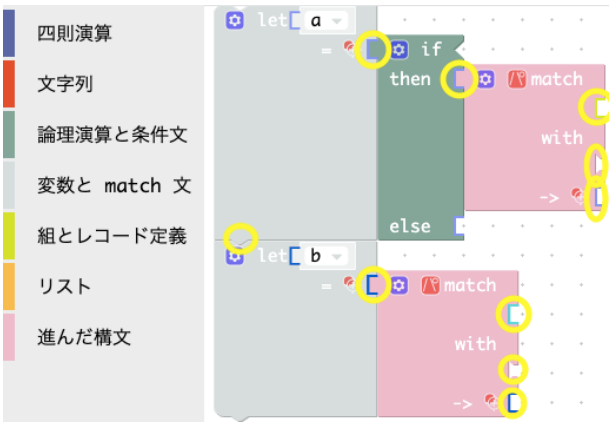

図 1: OCaml Blockly のページ．黄色で囲まれているのがコネクタで, ブロックが組み立てられていってコネクタにつながる型が決まると形が変わる。左側のサイドメニューを開くと、定義されたブロックが並んでいる。

3.1　Undo 機能

先行研究の OCaml Blockly では undo 機能が完成されていなかったため、その実装を行った。

問題点としては、

- それぞれの WS が独立しており、WS をまたぐ操作をするときのためにグローバルな場所にデータ構造を作る必要がある。
- undo できるのがブロックの出現と消失の時のみであり、対応していない動作が多い。

が挙げられた。

また、エディタの undo は、undo 操作そのものを undo できるかで違いがあるが、検討の末 OCaml Blockly では undo 操作の undo は redo 機能によって行うことにした。また、undo 操作をした後に新しい動作を行なった場合は、それ以前に undo した操作を redo することはできなくした。以下、改良点である。

- Undo は control + z , Redo は control + shift + z で実装。
- 人が行う 1 回の動作（マウス操作でいうとブロックを出す、動かす、消すなどの一回の操作）を Undo スタックに格納する。WS 間の移動も可。

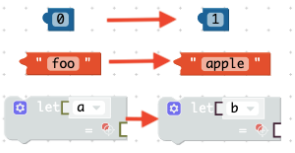

図 2: 数字、文字列、変数 (関数) 名の変更

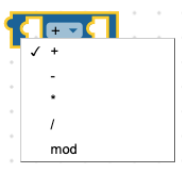

図 3: ドロップダウン

- 図 2 のように数字や文字列、変数 (関数) 名などの変更もキー操作で undo , redo を可能にした。
- 図 3 のようなドロップダウンの変更も可能にした。

3.2 キー操作でブロック組立

OCaml Blockly は元々基本的にマウス操作のみしか対応していなかったため、キー操作の実装はほぼ一からの製作だった。先行研究の時点でキー操作での Delete やコピー＆ペーストは使用可能でキー操作の関数自体は存在していたため、関数内を改良した。

どのようにキー操作を行えるようにしたかを以下に示す。

定義ブロックの出現 WS 上に何もない状態で “L” キーを押すと、let ブロックが左上に出現する。また、“T” キーを押すと、type 文のレコード定義ブロックが左上に出現する。

図 4: 定義ブロック. 定義ブロックはコネクタのハイライトがなくてもブロックを出現させることができる。

コネクタのハイライト 矢印キーを押すと、該当のコネクタが黄色くハイライトされる。→ (右矢印キー) でブロック横のコネクタ、↓ (下矢印キー) でブロック下のコネクタなど。

図 5: コネクタのハイライト. 左のブロックは→ (右矢印キー) を押した後の状態、右のブロックは↓ (下矢印キー) を押した後の状態。

ブロックの結合、出現 コネクタがハイライトされている状態で該当のキーを押すと、ハイライトされている場所にブロックが結合される。型が合わない場合はブロックは出現しない。

図 6: ブロックの結合. let ブロックの右側がハイライトされている状態で “M” キーを押した後の状態。

ハイライトの移動 ハイライトされている状態で矢印キーを押すとハイライトの場所が移動する。左右の結合先のブロックに移動したい場合、 shift + 左右キー でブロック間を移動できる。上下はそのまま移動可能。

ブロックの削除 各ブロックでコネクタのハイライトが順に下までハイライトされ、次に→ (右矢印キー) を押すとブロック全体がハイライトされる。その状態で Delete キーを押すとブロックが削除される。ブロックが削除された後、矢印キーを押すと、削除されたブロックが繋がっていたブロックのコネクタがハイライトされる。

図 7: ブロックの削除. if ブロック内でハイライトの移動をしていった状態。この後 Delete キーを押すと if ブロックは削除される。

3.3 プログラムの変更点

これらの改良にあたって、どのようにプログラムを変更したかを簡単に説明する。

Undo 機能 それぞれの WS にはすでに undo スタックと redo スタックが格納される場所があったので、グローバルにそれぞれの WS を一つにまとめるスタックを作った。よって WS をまたぐ undo が可能になった。これに、undo 機能が未対応のところは該当の関数内にスタックへ push する関数を呼び出すようにすることで undo が可能になった。

キー操作 ハイライトに関しては、各ブロックは自分のコネクタのリストを持っていたので、グローバルに現在選択されているブロックとコネクタを保持し、リストを順に回る形で実装した。また、キー操作の関数内に該当のキーが押されたらブロックを出現させるというプログラムをブロック数だけ作ることで様々なブロックの出現が可能になった。

4 まとめ

OCaml Blockly の機能拡張によって操作が楽になり、ブロック作成の時間も短縮されるようになった。まだ対応できていないものもいくつか残っているが、単純なブロックはキー操作一つで出現させることが可能になった。

ただ、操作テストをしていて細かい部分やまだ対応できていない操作がいくつか見えてきた。例えば、

- undo できなくなる場所がある
- メイン WS のみでしかキー操作できない

などが挙げられる。今後はそういった部分の修正、改良を重ねてより複雑なコードも早く簡単な操作を可能にしていく。

参考文献

[1] Google Blockly.
https://developers.google.com/blockly/

[2] 松本晴香, 浅井健一. Blockly をベースにした OCaml ビジュアルプログラミングエディタ, 第 21 回プログラミングおよびプログラミング言語ワークショップ論文集, 15pages, 2019