ブロックベースの視覚的プログラミング言語のための検索手法
A Search Method for a Block-Based Visual Programming Language

川村　大樹
Hiroki Kawamura
法政大学情報科学部ディジタルメディア学科
E-mail: hiroki.kawamura.7u@stu.hosei.ac.jp

Abstract

Block-based visual programming languages are easy for inexperienced programmers to use because blocks display their meanings in natural languages and enable programs with no grammatical errors. Users of block-based languages can make programs only with easy operations like drag and drop, and therefore block-based languages are suitable for learners of programming. On the other hand, because of blocks, they need larger space for programs than text-based languages. Therefore, programs in block-based languages often occupy large workspace, and spread not only vertically but also horizontally unlike text-based languages. These things make it difficult for users to find blocks. This paper proposes a search method for a block-based visual programming language. The method first combines blocks in a search box. Then it searches the workspace for the same combination of blocks by using the information in the search box, and it visualizes the results of the search. This method was incorporated in a programing environment implemented in HTML, JavaScript, and CSS as a Web application. This paper shows the results of experiments on the addition of a keyboard operation to a calculator program initially operated by buttons.

1. はじめに

Scratch [1]などのブロックベースの視覚的言語で，ブロックはそのはたらきを自然言語で示し，ブロックで記述されたプログラムは文法エラーを生じない．このため，ブロックベースの言語はプログラミング未経験者でも理解しやすい．また，ドラッグ&ドロップなどの簡単な操作だけでプログラミングできるので，プログラミングを学習するのに適している．

しかし，ブロックベースの視覚的言語はブロックを組み合わせてプログラムを記述するため，テキストベースの言語で同じプログラムを記述するときよりも広いスペースを使用する．また，大きいプログラムを作ろうとするとスクリプトが大きくなり，数も多くなる．さらに，テキストベースの言語と違ってワークスペースが縦だけでなく横にも広がるため，ブロックの位置を特定するのはテキストベースの場合よりも難しい．

本論文は，ブロックベースの視覚的言語でユーザが大きいプログラムの中から特定の部分を探す作業を容易にすることを目的として，ブロックを用いた検索手法を提案する．本手法の特徴は，ワークスペース上と同じように検索ボックス内でブロックを組み合わせることである．検索ボックス内の情報を元にワークスペース上から同じ組み合わせのブロックを検索し，ユーザに結果を提示する．本論文は，マウスでボタンをクリックして入力を行う電卓を開発し，その電卓にキーボード入力機能を追加する実験を行い，結果を提示する．

2. 関連研究

Scratch [1]は 8～16 歳の子供を対象に開発された，有名なブロックベースの視覚的言語であるが，子供だけでなく幅広い年齢層のユーザに利用されている．Scratch には大きいプログラムを扱うための様々な機能がある．例えば，スクリプトがワークスペースの右側や下側にはみ出した場合，自動でワークスペースを拡張する．また，ワークスペースには zoom 機能があり，ワークスペースが大きくなってしまった場合でも，ワークスペース内の表示範囲を広くすることでブロックの位置を探すことができる．clean up 機能はワークスペース上のスクリプトを自動で整列するが，スクリプトの先頭のブロックやスクリプトの大きさごとに整列しているわけではないので，ブロックの位置を特定することはできない．

Simulink や DRAKON [2]はオブジェクトを矢印で繋ぐことでプログラムを開発する，データフロー型のビジュアルプログラミング言語である．これらの開発環境では，テキストを使ってオブジェクトを検索することができる．

ブロックベースの視覚的言語である Blockly [3]には，複数のブロックをまとめて 1 つのブロックとして表示する collapse block 機能がある．この機能を使用することで，スクリプトを小さくして，ワークスペースを広く使うことができる．

3. 本研究で使用する視覚的言語

本研究では，栗原らが開発したブロックベースの視覚的言語 [4]を拡張する．本言語の開発環境は Web アプリケーションとして提供され，HTML，JavaScript，CSS で記述されている．

ブロックの形状は 4 つある．スタートブロック (図 1(a))は下にのみコネクタを持ち，関数定義に変換される．コマンドブロック(図 1(b))は上下にコネクタを持ち，関

Supervisor: Prof. Hiroshi Hosobe

数呼び出しや代入などの式を表す．ルールブロック(図 1(c))は上下と内部にコネクタを持ち，for 文などを表す．アウトプットブロック(図 1(d), (e))はコネクタを持たず，他のブロックのソケットに組み込むことで変数や定数，算術式などに変換される．アウトプットブロックとソケットには真偽値型と数値型の 2 つの形状があり，ブロックとソケットの形状が一致する場合にのみ組み込むことができる．また，ソケットにはアウトプットブロックを組み込むだけでなく，直接値を入力することもできる．

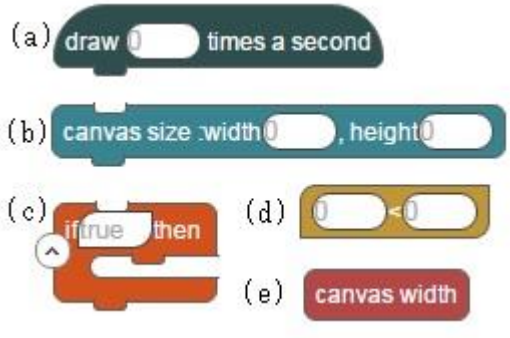

図 1　（a）スタートブロック，（b）コマンドブロック，（c）ルールブロック，（d）真偽値型のアウトプットブロック，（e）数値型のアウトプットブロック．

4. 提案手法

本研究で提案する手法において，検索すべきブロックの組み合わせがワークスペース上と同じように検索ボックス内に指定されると，同じ組み合わせのブロックがワークスペース上から探し出され，一致したものがユーザに提示される．これを実現するために，ブロックの組み合わせ方を判別するためのアルゴリズムが必要となる．

4.1. 判別アルゴリズム

検索ボックス内のブロックの組み合わせ方は以下の 4 つの工程で判別する．

1. ブロックの種類を特定する．
2. ブロックがソケットを持っている場合，そのブロックの持つ全てのソケットについて，ブロックが組み込まれているか，入力値があるか調べる．
3. ブロックが blocktail 型のコネクタ(図 2)を持っている場合，ブロックが組み込まれているか調べる．
4. ブロックが connect 型のコネクタ(図 2)を持っている場合，connect にブロックが組み込まれているか調べる．

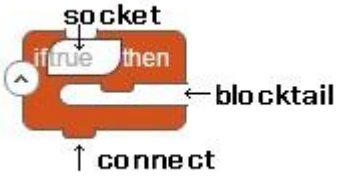

図 2　コネクタの種類

ただし，2〜4 の工程で組み込まれているブロックを発見した場合には，この 4 つの工程を発見したブロックにも再帰的に行う．

この 4 つの工程で得られた情報を元に，以下の工程でワークスペース上のスクリプトから検索対象のブロックの組み合わせが存在するか調べる．

1. 全てのブロックに対して 1 つずつ情報と一致する組み合わせであるか調べる．一致する組み合わせでなかった時点で次のブロックを調べる．
2. 一致したブロックを検索結果として提示する．

4.2. 例

図 3 に示す組み合わせのブロックを検索する場合，まず検索ボックス内のブロックの組み合わせ方を特定するための 4 つの工程が行われる．

1. ブロックの種類は repeat ブロック．
2. ソケットにブロック，入力値はない．
3. blocktail 型のコネクタにブロックがあるので，そのブロックに対して 4 つの工程を行う．これによって，ソケットに canvas width ブロックと入力値 height を持つ draw line ブロックが得られる．
4. connect 型のコネクタにブロックがあるので，そのブロックに対して 4 つの工程を行う．これによって，ソケットに入力値を持つ line color ブロックがあることが得られる．

これらの工程で以下の情報が得られ，ワークスペース上でこの情報と一致するブロックを検索結果として提示する．

1. repeat ブロック
2. repeat ブロックの blocktail 型のコネクタに Draw line ブロック
3. draw line ブロックの 1 つ目のソケットに Canvas width ブロック
4. draw line ブロックの 2 つ目のソケットに入力値 height
5. repeat ブロックの connect 型のコネクタに line color ブロック

図 3　検索の例

5. 実装

本研究の提案手法の実装に使用したプログラミング言語は JavaScript で，実行環境は Google Chrome を用いる．

5.1. ブロックの HTML 構造

本開発環境のブロックは HTML で記述されている．図 4 は if ブロックの HTML 構造の例であり，複数のクラスから成り立っている．

ブロックの形状を決めているクラスは第 1 行の最後のクラスであり，図 1(a)の形状は stage クラス，図 1(b)の形状は command クラス，図 1(c)の形状は rule クラス，図 1(d)の形状は expression クラス，図 1(e)の形状は condition クラスである．if ブロックは図 1(c)の形状であるので第 1 行の最後のクラスは rule クラスになっている．

ブロック同士の組み合せ方は socket 型，blocktail 型，connect 型 の 3 種類である．図 4 を例に説明すると，socket 型のコネクタを示すクラスは，第 8 行の socket クラスで，第 9 行の input 要素の後に組み込まれるブロックの HTML 要素が追加される．また，第 8 行の bool クラスは真偽値型のソケットの形状を示しており，数値型のソケットの場合は expr クラスである．blocktail 型のコネクタを示すクラスは，第 15 行の blocktail クラスで，第 16 行の join クラスの後に組み合わせるブロックの HTML 要素が追加される．connect 型のコネクタを示すクラスは第 19 行のクラスで，第 20 行の join クラスの後に組み合わせるブロックの HTML 要素が追加される．

```
1  <span class="script wrapper rule">          // ブロックの形状
2    <span class="script-block">
3      <b class="slot"></b>
4      <span class="blockhead">
5        <span class="script-label">
6          <span class="rule">                 // ブロックの形状
7            if                                // ラベル
8            <span class="bool socket">        // socketを持つ場合
9              <input type="text" class="$1" size="1" placeholder="true">
10           </span>
11           then                              // ラベル
12         </span>
13       </span>
14     </span>
15     <span class="blocktail">                // ルールブロックの場合
16       <b class="join"></b>
17     </span>
18   </span>
19   <span class="connect">                    // コネクタ
20     <b class="join ujoin"></b>
21   </span>
22 </span>
```

図 4　if ブロックの HTML 構造の例

5.2. ブロックを特定するための HTML 要素の追加

ブロックが組み合わされることで HTML 構造が複雑になるので，図 4 の HTML 要素のみでブロックを特定するのは困難である．そのため，ブロックを特定するための HTML 要素 name を第 1 行の span 要素に追加した．name 要素を追加した第 1 行は以下のようになる．

これにより，コネクタに他のブロックが組み込まれていても，ブロックが組み込まれていないものと同一のブロックとして容易に判別できる．

5.3. 判別方法

コネクタにブロックが組み込まれている場合，そのコネクタのクラスの直下に組み込まれたブロックの HTML 要素が組み込まれる．そのため，コネクタのクラスの直下に script クラスが存在するかを調べることで，4.1 節の 4 つの工程の 2〜4 の工程でコネクタにブロックが組み込まれているかを調べることができる．

6. 実験

本論文で提案した検索手法に関して実験を行った．本実験では，はじめに，マウスでボタンをクリックすることで入力を行う電卓を作り，その後，その電卓にキーボードからも入力できるように機能を加えた．

本開発環境のブロックは Processing [5]のコードに変換される．Processing は Java をベースにして，グラフィック機能に特化した言語である．簡単なコードだけですぐに視覚的なフィードバックを得られることからプログラミング初心者がプログラミングを学習するのに適している．

本実験で最初に作成したプログラムは大きく 3 つの部分に分けることができる．1 つ目は，プログラムを実行したときにはじめの 1 回だけ実行されるスクリプトで，電卓のボタンや表示画面を表示する．2 つ目は，一定間隔ごとに繰り返し実行されるスクリプトで，演算結果を表示する画面を更新する．3 つ目は，ウィンドウの中がクリックされたときに実行されるスクリプトで，電卓のボタンがクリックされたときにそのボタンに応じた処理を行う．

次に，キーボードからも数値を入力できるように，最初のプログラムに機能を加えた．最初のプログラムでも同様の組み合わせのブロックを使用していたため，検索ボックスを使って検索した(図 5) .

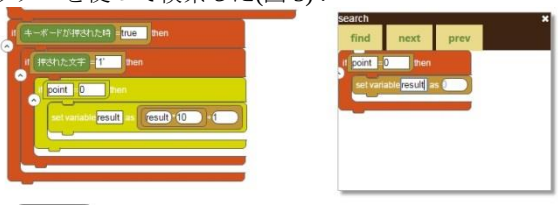

図 5　検索結果 1

これによって，数値のボタンがクリックされたときの入力を処理するスクリプトで同じ組み合わせのブロックを発見した(図 6).

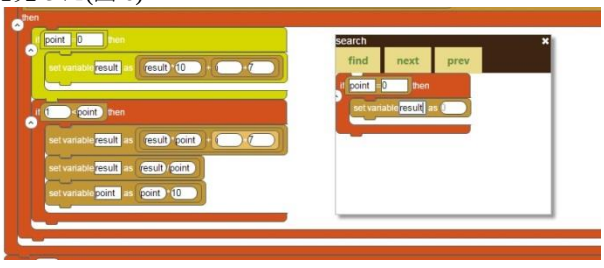

図 6　検索結果 2

発見した組み合わせのブロックをまとめるために，マクロブロックで定義した．マクロブロックは関数呼び出しに，マクロ定義ブロックはメソッドに変換される．図 7 のマクロブロックは以下のように変換される．

pressedNumber(0);

マクロ定義ブロックは以下のように変換される．

```
void pressedNumber(float num){
        …
)
```

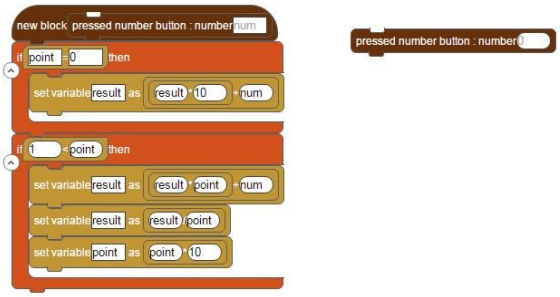

図 7　左 : マクロ定義スクリプト / 右 : マクロブロック

また，キーボードから演算記号を入力した場合の処理を作ったときにも，検索ボックスを使って同じ組み合わせのブロックをそれぞれ発見できたので，その組み合わせをまとめるマクロブロックを定義した．

また，repeat ブロックを検索したところ，同じ回数繰り返している repeat ブロックを複数使用している箇所を見つけることができたので，1 つにまとめてより簡潔にした(図 8).

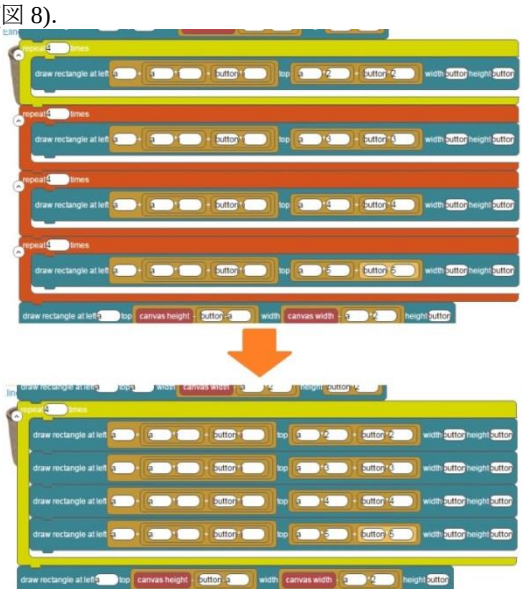

図 8　検索結果 3

7. 議論

本提案手法では 1 つのブロックを検索対象として検索したときに多くのブロックが検索結果として提示された場合には，検索ボックス内のブロックのソケットに入力値を加えたり，コネクタにブロックを加えたりすることで検索結果を絞ることができる．また，本提案手法ではテキストによる検索方法と違い変数名で検索することができないので，定義されている変数がどこに使われているかを探すことはできない．

テキストベースの言語では，“＋”や“＝”などの文字は複数の用途で使用されているために上手く検索することができない．一方，本論文の提案手法では，これらの文字であっても検索することができる．

条件式や複数のコマンドブロックを組み合わせる場合など，ブロックの組み合わせ順序によって意味が変わらないものについては，本提案手法では適切に検索することができない．例えば条件式において“＋”や“＝”等のブロックを使用する場合，そのブロックの持つ 2 つのソケットの内容を入れ替えたとしても処理としては同じ意味を表す．しかし，本提案手法の 4.1 節で得られる情報ではソケットの位置も検索の条件になってしまうため，異なる意味を持つブロックの組み合わせであると判断される．

図 9　本提案手法において同一とみなされない例

本提案手法は，プログラムが大きい場合に有効であると考えられる．例えば Scratch ではブロックの種類が多く，スクリプトの数が増えやすいために，本提案手法が有効に働く可能性がある．

8. おわりに

本論文ではブロックベースの視覚的プログラミング言語において，大きいプログラムの中から特定のブロック組み合わせを発見するためにブロックを用いる検索手法を提案した．さらに，マウスでボタンをクリックして入力を行う電卓を開発し，その電卓にキーボード入力機能を追加する実験を行った．実験の結果，同じ組み合わせのブロックを使用している部分を検索手法によって発見できることが示された．

文　献

[1] M. Resnick et al., "Scratch: Programming for all," in *Communications of the ACM*, 52(11):60–67, 2009.

[2] "DRAKON," [Online].
Available: http://drakon-editor.sourceforge.net/.

[3] "Blockly: Web-Based Visual Programming Editor," Google, [Online].
Available: https://code.google.com/p/blockly/.

[4] A. Kurihara, A. Sasaki, K. Wakita and H. Hosobe, "A Programming Environment for Visual Block-Based Domain-Specific Languages," *Proc. SCSE, Procedia Computer Science*, vol. 62, pp. 287-296, 2015.

[5] C. Reas and B. Fry, "Processing," [Online].
Available: https://processing.org/.