CN-005

学習用自作プログラミング言語の生成 AI による実行環境生成の試み
Generating an Execution Environment for an Educational Self-made Programming Language using Generative AI: A Preliminary Study

倉橋 農[1)]　越智 徹[2)]　尾崎 拓郎[3)]　島袋 舞子[4)]　今井 正文[5)]
Minori Kurahasi　Toru Ochi　Takuro Ozaki　Maiko Shimabuku　Masafumi Imai

1　はじめに

1.1　概要

筆者らは，プログラミング的思考を促す目的で，順次・分岐・反復を盛り込んだアンプラグド授業「ハンバーガー・ロボ」を実践してきた [1][2]．この教材は関数やリテラルを表すカードを並べ，プログラムを完成させ，教師扮するロボットがそれを実行して絵カードでハンバーガーを完成させるものである．アンプラグド教材を補うものとして，この学習用言語をソフトウェア上で実行する環境を，生成 AI を利用して構築した．この教材は HTML と JavaScript からなるが，そのコードはほとんど完全に生成 AI によるものである．

コード生成の際には，BNF の作成や AST の生成など形式的な指示を段階的に与えることで効率的かつ正確な環境構築が可能であったことを報告する．また，実行環境の生成 AI による開発の過程により学習者が何を学べるかについても議論する．

1.2　動機

生成 AI によるコードの生成は一般的なものとなりつつあり，多くの試行と報告が行われている [3]．本報告では，最低限の指示で全てのコードを生成 AI に書かせるというものであり，以下 3 点を確かめることが主な動機であった．

- 生成 AI の実力を試す
- 効率的な生成 AI への指示は何か
- 教育への応用可能性は何か

なお，本報告と類似した試行として，生成 AI に BNF を与えて実行環境を作った例がある [4]．筆者らは BNF を直接与えるのではなく，BNF の生成から生成 AI に行わせた．

2　アンプラグド教材

2.1　アンプラグド

イギリスでは，2014 年 9 月から公立小学校 1 年生（5 歳）から，新教科 Computing が実施されている．これ以前には，1995 年から教科 ICT が全学年で実施されていたが，コンピュータ・サイエンスの内容が大幅に取り入れられ，ICT が Computing に置き換えられた．Computing では，教科書や教材，カリキュラムが特に定められておらず，学校の裁量で自由に決定できる．そのため，様々な教材が提供されているが，Rising STARTS が発行する “Switched on Computing” シリーズが比較的多くの学校で導入されている．この教材は，Computing を推進するための機関 CAS (Computing At School) や Naace (The National Association for all those interested in technology in Education) らと共同で開発され，様々な教育リソースも提供されている．

本報告の提案する授業は前述の Switched on Computing [6, p. 25] に着想を得たものであるが，本節で詳述するように，オリジナルの案とは大きく異なっている．本提案では題材をハンバーガーに変更しており，以後これを「ハンバーガー・ロボ」と呼び区別する．これは教師扮するハンバーガー・ロボは，受講者の口頭の指示または単語カード（図 1）を配したスクリプト・シート（図 2）により，食材の絵カード（図 3）を組み合わせて，メニュー（図 4）やリクエストに応じたハンバーガーを完成させると言うものである．

ハンバーガー・ロボへの指示と実行は，単純化されたプログラム・コードとその実行を可視化したモデルと考えることができる．このロボットに指示を与えハンバーガーを作成する過程で受講者は，厳密で曖昧性のない指示，対話モード／スクリプトによる実行，順次・反復・分岐，ブロック，要件定義や仕様の策定といった，プログラミングの重要な概念を理解可能となっている．

2.2　テキスト・プログラミングへ

前節までで紹介したアンプラグド授業に用いる教材には，1) カードの枚数に制限がある，2) 作成したスクリプトの答え合わせに手間がかかるなどの問題点があった．そこで筆者らはこのアンプラグド教材「ハンバーガー・ロボ」を補完する Scratch の教材を開発した [5].

本報告で用いるプログラミング言語を，「ハンバーガー言語」と呼ぶことにする．筆者らは，アンプラグド教材から一般的なテキスト・プログラミングへの橋渡しとして，ウェブ・ブラウザ上で動作するハンバーガー言語の実行環境を構築した．これはアンプラグド教材を再現し，ハンバーガー言語をテキストとして入力，命令を出してハンバーガーを完成させるものである．これにより，学習者は自然な形でテキストを用いたプログラミングを体験することができる．また，本実行環境は完成したハンバーガーを表示するのみならず，ハンバーガー言語の構文解析結果も表示することができる．初学者には不要なので隠したバージョンを利用してもらうが，プログラミングについて高度な理解を促したい，大人向けの授業では有効と考えられる．

3　完成した実行環境

3.1　実装に用いた言語

本実行環境の開発には，HTML と JavaScript を用いた．特に初等教育で用いられることを想定し，端末を選ばずウェブ・ブラウザだけでの動作が可能であることを重視した結果である．

3.2　ハンバーガー言語の仕様

ハンバーガー言語の仕様は，アンプラグドのそれに準ずる．Scratch 教材では，Scratch の仕様上，構文に若干

1）　羽衣国際大学，mkurahasi@hagoromo.ac.jp
2）　大阪工業大学
3）　大阪教育大学
4）　沖縄国際大学
5）　豊橋創造大学

したのパン	うえのパン
おにく	レタス
を とって	を おいて
⬇ここから	⬆ここまで
回	して

図 1　単語カード

図 2　スクリプト・シート

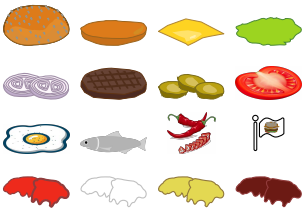


図 3　食材カード

図 4　メニュー

の変更を加えたが，本実行環境においてはアンプラグド教材とほとんど同一である．以下に仕様を記述する．

3.3　ハンバーガー言語の仕様

3.3.1　語彙

語彙を 語彙，引数を＜引数＞として表し，対で利用する語彙は「＝」で結んでいる．アンプラグド版で用いていた開始命令「ハンバーガーつくろう！」は利用していない．

関数

＜材料＞ をとって

＜材料/ソース＞ をとって

＜ソース＞ をかけて

材料　うえのパン，おにく，トマト，…

ソース　ケチャップ，カラシ，マヨネーズ，…

数　1，2，3，…

3.3.2　制御構造

ブロック　↓ここから＝↑ここまで

反復　＜ブロック＞＜数＞ 回 ＝ して

分岐　＜条件＞ なら ＝＜ブロック＞＝ して

3.3.3　条件

分岐「 なら ＝＜ブロック＞＝ して 」の条件として，「おとな」「こども」が用意されている．これは実際のインターフェース上ではラジオボタンでの選択式となっている．

3.4　インタフェース

3.4.1　コード入力

図 5 は，実行環境のコード入力欄と分岐の条件「おとな」「こども」の選択部分の画像である．コード入力欄下の「実行」を押すと，ハンバーガー作成の簡単なアニメーションが表示され，構文木とハンバーガーを表す JSON を表示することもできる．

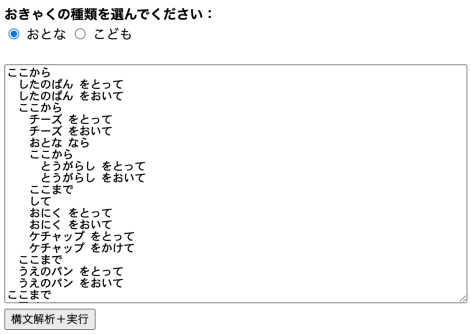


図 5　コード入力欄

3.4.2　実行結果

「実行結果」欄に，コードの実行結果が表示される．正常実行された場合はその旨，エラーが生じた場合はエラーメッセージが表示れる．

3.4.3　ハンバーガーのアニメーション

本実行環境では，ハンバーガーの材料が上から落ちてくるアニメーションを用いて，ハンバーガーを完成させる．図 6 は，完成したハンバーガーの画像である．

図 6　完成したハンバーガー

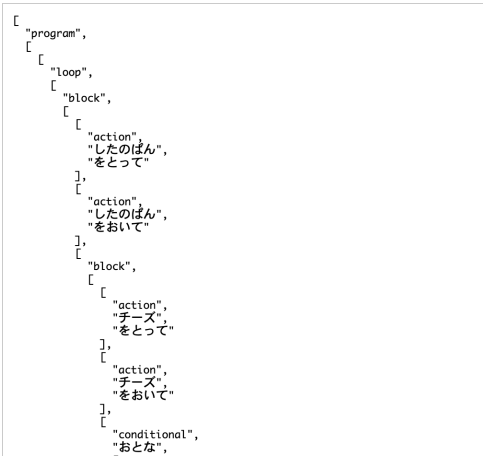
3.4.4　構文木の表示

図 7 は，構文解析の結果を構文木として表示した画面である．初学者向けにはこれを表示しないことが可能となっている．

3.4.5　完成したハンバーガーのリスト

図 8 は，完成したハンバーガーを JSON として表示した画面である．これはハンバーガーが内部的にリストとして扱われていることを意識させる意図で用意されているが，初学者向けには表示しないことが可能である．

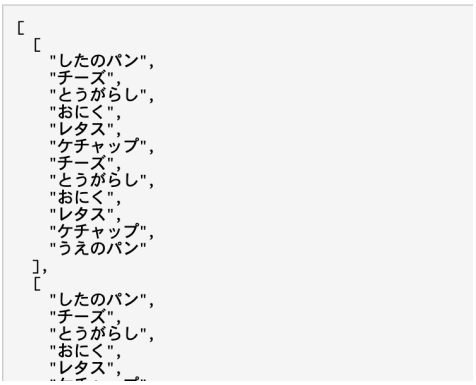
構文木（AST）



```
[
  "program",
  [
    [
      "loop",
      [
        "block",
        [
          [
            "action",
            "したのぱん",
            "をとって"
          ],
          [
            "action",
            "したのぱん",
            "をおいて"
          ],
          [
            "block",
            [
              [
                "action",
                "チーズ",
                "をとって"
              ],
              [
                "action",
                "チーズ",
                "をおいて"
              ],
              [
                "conditional",
                "おとな",
```

図 7　構文木

完成したハンバーガーの JSON



```
[
  [
    "したのパン",
    "チーズ",
    "とうがらし",
    "おにく",
    "レタス",
    "ケチャップ",
    "チーズ",
    "とうがらし",
    "おにく",
    "レタス",
    "ケチャップ",
    "うえのパン"
  ],
  [
    "したのパン",
    "チーズ",
    "とうがらし",
    "おにく",
    "レタス",
```

図 8　JSON 形式のハンバーガー

4　コードの生成

4.1　環境と方針

本報告で用いた生成 AI は ChatGPT，モデルは 4o である．実行環境の開発言語は JavaScript であるが，開発の初期段階では Python でのコーディングを指示している．

本試行では 1) 人間はコードを書かない，2) ハンバーガー言語のコードとその出力の具体例をあげる，3) 形式的な段階を踏むよう指示を与える，の 3 点を基本的な方針とした．

3) の「形式的な指示」とは，主に BNF, AST の作成，構文解析・意味解析の区別，実行結果の JSON による表示である．

4.2　指示と結果

以下は，最初に与えたプロンプトの一部分である．この直前に，3.3 節に述べたハンバーガー言語の仕様を与えている．最初期に基本的な動作を確認するまでは Python での開発を行い，その後 JavaScript に切り替えている．これは，筆者らが比較的 Python に明るいために Python での実装を行っていたが，実行できる端末を広げること，アニメーションでの表現を加えることを目的として，JavaScript へと言語を切り換えた．

> # あなたへの指示
>
> - まずは，このハンバーガー言語の文法を BNF 記法で書き表してください．
> - 次に，ハンバーガー言語のパーサーを Python で実装してください．
> - 不明な点，足りない情報があれば必ず確認してから作業してください．

このプロンプトの後は，仕様に合致しない部分とコードエラーの修正を繰り返し指示した．この際，仕様に合致しない点についての指示は，基本的に入力となるハンバーガー言語のコードと出力結果（完成するハンバーガー）を与えることで行った．例えば，以下のようなプロンプトである．

> 最外のアクション列が実行されていません．以下のコードで，「したのパン」のみからなるハンバーガーができませんでした．

次のように明示的に仕様からの乖離部分の修正を指示した場合もある．

> 反復のネストを実装して下さい．

さらに，適宜 BNF の再提示の指示，構文解析・意味解析の区別の指示で明らかな作業の前進が見られた．例えば，以下のようなプロンプトである．

> この BNF だと分岐は例えば以下のように，「ここまで」を 2 回書く必要があるように思えますが，正しい理解ですか．

> 以下のように，「ケチャップ」をとってから「マスタード」をかけてもエラーが出ないですね．「手に持っているのはマスタードです」とか言って欲しいのですが，これは構文解析上の問題でしょうか，それとも意味の問題ですか．

以下は，JavaScript コードの完成時に最終的に提示させた BNF である．

```
<プログラム>::=<文>*
<文>::=<反復>
        |<分岐>
        |<ブロック>
        |<アクション>
<反復>::=<ブロック><回数>回 して
<分岐>::=<条件>なら<ブロック>して
<ブロック>::=ここから<文>*ここまで
<アクション>::=<材料名>をとって
                |<材料名>をおいて
                |<ソース名>をとって
                |<ソース名>をかけて
<条件>::=おとな| こども
<材料名>::="したのパン"|"うえのパン"|"おにく"|"トマト"
          |"レタス"|"チーズ"|"たまねぎ"|"ピクルス"...
<ソース名>::="ケチャップ"|"マスタード"|"マヨネーズ"...
<回数>::=正の整数（例: 2, 3, 10）
```

5　実践で得た知見

5.1　すぐ忘れる

この実践を通じて，筆者らは特に感じたのは，以下の 2 点である．

- 「をおいて」「をかけて」の区別が忘れさられていた
- 存在しない材料が許容される，これも忘れ去られていた

形式的な指示を改めて提示することによって，問題の所在がどこにあったかの確認と，軌道修正ができる．例えば，このような場合であれば，以前に意味解析をしていたことを指摘すると話が早い．

5.2 細かく段階を踏む

当初，BNF や AST から実行環境の生成を一度の指示で行おうとしたが，実際はより細かく段階を分けて指示する必要があった．最終的な指示は，「反復 → ブロック → 複数のハンバーガー → 入れ子 → 分岐 → 反復と分岐の入れ子」といった順番で行なっている．ここでも，形式的な指示を確認しながら仕様と合致していることを確認して次に進むと効率が良い．

5.3 機能追加の提案は無視する

生成過程において，対話がひと段落するたびに機能の追加を提案してくる．それまでの指示が完了しているか確認すること，事前に予定した開発段階を踏むことを優先すべきである．

6 おわりに

本報告では，学習用言語をソフトウェア上で実行する環境を生成 AI を利用して構築した．コード生成の際には，BNF の作成や AST の生成など形式的な指示を段階的に与えることで効率的かつ正確な構築が可能であった．

また，本実行環境は基本的にプログラミングの初学者を対象としたものであるが，実行環境の生成 AI による開発の過程を観察することで，AST, BNF など言語処理系についての基本的な概念の獲得に寄与することが可能であろう．

今後の展望としては，本報告で提案した手順で実行環境の作成を再度行い，効果を確かめること，他のモデルでも同様の傾向が見られるか確認することが挙げられる．

謝辞

本報告は JSPS 科研費 24K06344 の助成を受けたものである．

参考文献

[1] 倉橋農，越智徹，尾崎拓郎，島袋舞子：子ども向け授業にリンクした保護者・教師向けプログラミングコースの検討，情報処理学会研究報告 2019, Vol. 2019-CE-150, No. 9, 1–5, 2019.

[2] 倉橋農，越智徹，尾崎拓郎，島袋舞子：小学生向けアンプラグド・プログラミング入門授業「ハンバーガー・ロボ」の提案と実践，情報教育シンポジウム論文集 2019 pp. 299–304, 2019.

[3] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, Charles Sutton, Program Synthesis with Large Language Models, arXiv:2108.07732, 2021

[4]「ChatGPT (Advanced Data Analysis) で BNF 式からパーサーを生成する」https://acro-engineer.hatenablog.com/entry/2023/09/22/150000

[5] 倉橋農，尾崎拓郎，越智徹，島袋舞子，今井正文：アンプラグド・プログラミング教材「ハンバーガー・ロボ」の補完としての Scratch 教材の開発，情報処理学会研究報告，Vol. 2024-CE-177, No.22, pp. 1–6, 2024.

[6] Miles Berry, “Switched on Computing Year 1” (2nd Edition), Rising Stars, 2014.