C-05

Rust の所有権に関する学習支援システムの構築
Building a Learning Support System for Rust Ownership

林 亮汰†　　村川 猛彦 †
Ryota Hayashi　　Takehiko Murakawa

1. はじめに

近年，社会の IT 化は急速に進んでおり，その基盤となるのがプログラミング技術である．プログラミングとは，コンピュータに指示を与えるために，特定のプログラミング言語を用いてコードを記述することを指す．現在，世界には 200 種類以上のプログラミング言語が存在するといわれており，それぞれに異なる特徴や用途がある．和歌山大学システム工学部では，C や Python，Java といったプログラミング言語を学習する科目が開講されている．

Rust は，メモリ管理の安全性やパフォーマンスの高さから，開発者に高い人気を誇るプログラミング言語である．この言語は，システムプログラミングや Web アセンブリなど，多様な分野で活用されている．しかし，その習得難易度の高さや初学者向けの教材の不足がその普及を妨げている．特に Rust 特有の概念である“所有権”は他の言語を習得している人にとっても課題となっている．

また最近では，様々な生成 AI が手軽に利用可能となっている．生成 AI はテキストや画像，プログラムコードといった多種多様なコンテンツの生成に優れており，様々な用途で利用される．ChatGPT はその自然な対話や高性能な生成を行えることから特に注目されている．

本研究では Rust の所有権に関する学習支援のための Web アプリケーションを試作した．多肢選択式で，エラーを解消する方法を選ぶ．作成した 4 問のうち 3 問において ChatGPT を活用している．

2. プログラミング学習とその支援

2.1 プログラミング言語 Rust およびその学習

プログラミング言語 Rust[1]は，メモリ管理の安全性やパフォーマンスの高さから，開発者に人気が高いプログラミング言語である．Rust はその堅牢なメモリ安全性や高効率な並行処理から評価されており，それらの機能は Rust 特有の概念である所有権からなっている．

Rust は現在，さまざまなソフトウェアで活用されている．アプリケーションとしては，Transport Layer Security (TLS) の Rust 実装である Rustls，grep よりも高速なテキスト検索ツールの ripgrep，Adobe Flash Player の代替として開発されたエミュレーターの Ruffle などがある．Windows10，Linux，Android といった基本ソフトウェアでも，Rust が採用されている．

所有権は Rust 独自のメモリ管理方法である．他言語では Java や Python などのようにメモリの管理にガベージコレクションを採用しているか，C や C++のようにプログラマが明示的にメモリの確保および解放を行う．しかしながら，ガベージコレクションは実行時のオーバーヘッドがかかり，プログラムの実行速度を低下させてしまうことがある．またプログラマ自身がメモリを管理する方法だとヒューマンエラーを完全に防ぐことはできないため，メモリの二重解放，ダングリングポインタといった想定外のバグが発生しうる．それに対し，Rust では所有権システムに違反したプログラムコードはコンパイル時にエラーを出力するので，意図せぬバグを未然に発見することができる．例えば図 1 に示したコードは所有権システムに基づきエラーとなる．

```
1    fn main(){
2        let a = String::from("hoge");
3        let b = a;
4        println!("{}", a);
5    }
```

図 1: 所有権システムが原因でエラーとなるコードの例

このプログラムコードは変数 a に文字列 ”hoge” を格納し，それを出力しようとするものである．所有権とは値を所持する権利であり，原則として 1 つの変数でのみ保持される．上記のコードでは 2 行目で変数 a が文字列 ”hoge” に対する所有権を持ち，3 行目で変数 a の持つ所有権が変数 b に移動する．これにより，変数 a は所有権を失う．そして変数 a の内容を出力しようとする 4 行目でコンパイルエラーが発生する．

2.2 プログラミング学習支援システム

現在多くの大学で，プログラミング教育が行われているが，プログラミングに対する関心，躓きやすい箇所，学習速度などは学生によって異なることから，学部・学科などで提供されている講義や演習のみでプログラミングを習得するのは容易でない．このような背景のもと，筆者らの研究室では，プログラミング学習支援システムを Web アプリケーションとして開発し，授業中または授業時間外に利用できるよう提供してきた[2][3]．プログラミング学習支援システムにおける解答の方法として，目的となる機能を見てゼロからコードとして実装する記述形式[4]，完成したコードの一部だけが空欄となっており，その空欄部分に適切なコードを入れる空欄補充形式[5]，適切に動作するためのコードを選択肢の中から選ぶ多肢選択式[6]を挙げることができる．記述形式では実際にすべてのコードを記述するので，その言語についての包括的な知識が必要となる．様々な知識を得られるので，その言語への理解を効率的に進められることが期待されるが，学び始めの言語の学習には適さない．空欄補充形式や多肢選択式では記述形式ほど多くの知識を求められないので，様々な学習者に適している．また間違えた問題の理解を自力で行いやすいことや，問題を解いて理解するまでの時間が短くなることから，学習者の学習意欲の低下を防ぐことができる．

Rust を問題形式で学習する Web アプリケーションとして Rustlings[8]が挙げられる．Rustlings では，Rust の公式ドキュメントである The Rust Programming Language に対応した

†和歌山大学, Wakayama University

演習問題が提供されるため，テキストの読み進めと合わせて演習問題が進められる．しかし，Rust に関するこのような教材は他の言語に比べて少ない．さらに習得の難しい所有権に絞って学習できるものとなるとより一層限られる．本研究では，問題の数が限られていること，問題作成にかかる人的コストを，生成 AI を用いた問題の生成によって解決できるのではないかと考えた．

2.3 生成 AI

生成 AI とは，ユーザの指示に従って文章や画像，音楽などのコンテンツを生成することのできる人工知能である．またユーザの与える指示をプロンプトと呼ぶ．

代表的な生成 AI として，OpenAI 社の開発した ChatGPT，Google 社の開発した Gemini，Microsoft 社の開発した Copilot などがある．いずれも，ブラウザ上で対話的に利用するものであるが，GitHub Copilot のように，対応するコードエディタに，専用のプラグインをインストールすることで使うというものもある．また八木らは，Gemini を使用して複数科目で利用可能な対話型質問支援システムを開発している[8]．

本研究では 3 章で述べる通り，Rust のソースコードを含む多肢選択式問題の生成に ChatGPT を使用している．

2.4 関連研究

Almeida ら[9]は，Rust の所有権と借用の概念の可視化を補助するツールである RustViz を開発した．RustViz を用いたチュートリアルに対するアンケートを取った結果，RustViz で作成された可視化図が「役立った」「とても役立った」と答えた参加者は 90%を超えていた．RustViz を使用しなかった学習者との比較が行えなかったこと，コンパイルの通らないコードに対する可視化が行えないことが課題となった．

竹重[10]は，Rust の所有権概念を他言語に移植し，Rust の文法や開発環境と切り分けて所有権概念を学習できるようにすることで，Rust の学習効率の向上を目指した．被験者実験の結果，所有権理解の促進に寄与することが認められた．

3. Rust 学習支援システム

3.1 開発方針

本研究で開発した，Rust の所有権システムを対象とした学習支援システムについて説明する．このシステムでは学習者が多肢選択式の問題に解答することで学習を進める，問題数は 4 問であり，そのうち 3 問は ChatGPT を用いて生成したもの，1 問は人手で作成したものである．

3.2 システム構成

本システムは，Web アプリケーションとして実装した．サーバ構築やルーティングなどのバックエンド部分は，Rust の Web アプリケーションフレームワークの一つである axum を用いて行い，画面表示や画面の動作を行うフロントエンド部分は JavaScript を埋め込んだ HTML で記述した．

データの保存や検索，更新を効率的に行う方法として，データベースが広く用いられる．特に大規模な Web アプリケーションではデータベースにユーザ情報をはじめとするデータを保存することがあるが，本システムは小規模なもので，データを更新することがないのでデータベースは利用していない．

3.3 システム画面と機能

作成したシステムの問題出題画面を図 2 に示す．これは ChatGPT を用いて生成した問題である．

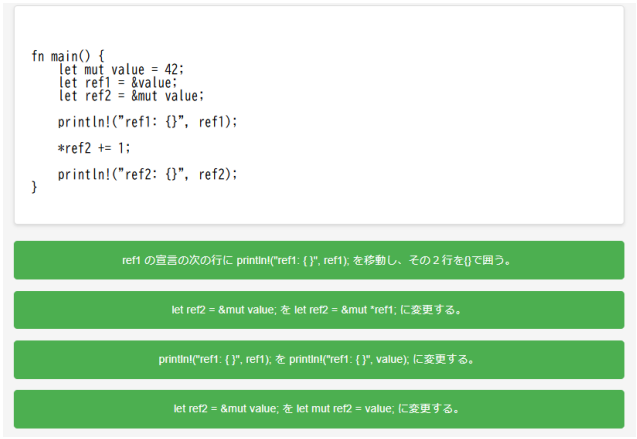

図 2: システムの出題画面

学習者が問題の出題画面で適切だと思う選択肢を選ぶと，解答表示画面へ遷移する．解答表示画面では選んだ選択肢に応じて正解不正解が表示される．また各問題において適切な選択肢は一つのみである．

図 2 に表示した問題について出題意図，正解・不正解およびその理由を示す．この問題は同じ参照先への不変参照と可変参照が共存できないことを理解しているか問う問題である．main 関数内の 2 行目で value に対しての不変参照を行っており，3 行目でも value に対しての可変参照を行っている．これらが原因で，このコードを実行するとコンパイルエラーが発生する．正答は 1 番目の選択肢「ref1 の宣言の次の行に println!("ref1:{}", ref1);を移動し、その 2 行を{}で囲う。」であり，この通りに修正して実行すると，「ref1:42」および「ref2:43」を出力する．

2 番目の選択肢「let ref2 = &mut value;を let ref2 = &mut *ref1;に変更する。」の通りに変更したコードは，不変参照として定義した ref1 に対して可変参照でアクセスしようとしているためコンパイルエラーが発生する．

3 番目の選択肢「println!("ref1:{}", ref1); を println!("ref1:{}", value);に変更する。」の通りに変更したコードでは，main 関数内の 2 行目，3 行目で行われる，value に対しての不変参照と可変参照が解消されていないためコンパイルエラーが発生する．

4 番目の選択肢「let ref2 = &mut value;を let mut ref2 = value;に変更する。」の通りに変更したコードでは，ref2 は value を含めどの変数も参照していないにもかかわらず，main 関数内で 5 行目の*ref2 という使い方をしてしまったためにコンパイルエラーが発生する．

人手で作成し，本システムで解答可能にした問題について，提示する（エラーが発生する）コード，正解の選択肢，および 3 つの誤選択肢を，表 1 に示す．

表 1: 人手で作成した問題

コード	fn main(){
	let s1 = String::from("hello");
	let s2 = takes_and_gives_back(s1);
	println!("{}", s1);
	println!("{}", s2);
	}
	fn takes_and_gives_back(some_string: String) -> String{
	some_string
	}

正解	let s2 = takes_and_gives_back(s1); を let s2 = takes_and_gives_back(s1.clone( ));とする
誤り	fn takes_and_gives_back(some_string: String) -> String の戻り値 String を&String とし、関数内の some_string を&some_string とする
誤り	fn takes_and_gives_back(some_string: String) -> String の引数 some_string: String を some_string: &String とし，lets2 = takes_and_gives_back(s1); の引数 s1 を&s1 とする
誤り	println!("{}", s1);を println!("{}", &s1);とする

学習に適した問題の生成のためにプロンプトの改善を行い，プロンプトの改善による問題生成の有用性を調べた．プロンプトの改善によって，多様な問題が生成され，方針に合う問題がプロンプト改善前より生成されやすくなることが確認できた．このことから学習支援システムの開発において生成 AI の活用は効果的であると考える．

4. 考察

本研究では，ChatGPT を利用した Rust の学習に役立つ問題の生成，生成された問題を利用した学習支援システムの構築を行った．

一方で作成したシステムにはいくつかの課題も見られた．第一に，利用者評価が行えなかった点が挙げられる．その主な要因として，Rust を学び始めた人にとって学習難易度が高く，十分な利用者が集められなかったことが挙げられる．今後はより幅広い学習者に合わせた問題とするために，難易度設計を見直す必要がある．

本システムは Rust の所有権にのみ焦点を当てたものであったため Rust の他の概念や文法の理解を促進できなかった．他に Rust に関して初学者が理解すべき事項としてライフタイムがある．これにより，参照が有効である期間をコンパイル時に検査することができる．他のプログラミング言語で使われるスコープは変数が使える範囲を示すものであるため，参照の有効性を確かめられるライフタイムは Rust 独自の概念と言える．Coblenz らの調査[11]では，Rust を学習する際に，所有権と借用に加えてライフタイムの概念が大きな課題となることが明らかになっている．そこで，ライフタイムの学習ができるシステムへの拡張が求められる．

Rust のライフタイムの学習支援を行う先行研究として，Blaser[12]はライフタイムチェックに関するエラーメッセージが Rust の初心者にとって理解しがたいという点に着目し，エラーメッセージに対してステップバイステップの説明を自動生成するツール Rust Life を開発した．本ツールは多くの検証用コードに対して適切に機能したが，いくつかの制約も確認された．

5. おわりに

本研究では，Rust の所有権の理解に役立つ問題を生成し，生成した問題を利用した学習支援システムの構築を行った．

今後の課題としては，Rust の所有権にとどまらないより包括的な学習支援を行えるシステムの構築や，Rust の初学者を含め，多様な学習者が利用可能な学習支援システムの構築を行う必要がある．

謝辞

本研究は JSPS 科研費 22K12291 の助成を受けたものです．

参考文献

[1] 佐藤昭文:Web アプリ開発で学ぶ Rust 言語入門, 秀和システム (2022).

[2] Murakawa, T.: LbTyping: A Web Application for Programming Learning by Typing, The Fourteenth International Conference on Information, Intelligence, Systems and Applications (IISA 2023), Volos, Greece, 4 pages (2023).

[3] 田中和季, 村川猛彦: プログラミング初学者を対象としたオンラインジャッジシステム,第 23 回情報科学技術フォーラム (FIT2024), 第 4 分冊, pp.73-80 (2024).

[4] 野上裕二, 納富一宏: プログラミング学習支援における問題自動生成に関する基礎的検討, 第 16 回情報科学技術フォーラム(FIT2017), 第 3 分冊, pp.465-466 (2017).

[5] 坂手柊斗, 重松大志, 洲濱拓弥, 松本慎平: 空欄補填による C プログラミング学習課題の Web システムの開発, 教育システム情報学会 2022 年度学生研究発表会 中国地区, pp.213-124 (2023).

[6] Sarsa, S., Denny, P., Hellas, A., and Leinonen, J.: Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models, Proceedings of the 2022 ACM Conference on International Computing Education Research (ICER '22), Vol.1, pp.27-43 (2022).

[7] Rustlings. https://rustlings.cool (2025 年 7 月 25 日参照)

[8] 八木俊磨, 漆原宏丞, 鈴木達也, 島袋舞子, 荒木千秋, 兼宗進: 複数科目で利用可能な対話型質問支援システム「AI Parrot」の開発: 2024 年度情報処理学会関西支部支部大会, D-01 (2024).

[9] Almeida, M., Cole, G., Du, K., Luo, G., Pan, S., Pan, Y., Qiu, K, Reddy, V., Zhang, H., Zhu, Y., and Omar, C., RustViz: Interactively Visualizing Ownership and Borrowing, 2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Rome, Italy, pp.1-10 (2022).

[10] 竹重拓輝, 柗本真佑, 楠本真二: Java への所有権システム移植による Rust 学習支援, 電子情報通信学会技術研究報告, Vol.123, No.335, SS2023-31, pp.1-6 (2024).

[11] Coblenz, M., Mazurek, M. L., and Hicks, M., Garbage Collection Makes Rust Easier to Use: A Randomized Controlled Trial of the Bronze Garbage Collector, 2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE), Pittsburgh, PA, USA, pp.1021-1032 (2022).

[12] Blaser, D., Simple Explanation of Complex Lifetime Errors in Rust, ETH Zurich (2019).