自然言語のテストシナリオから保守性の高いテストスクリプトを自動生成する手法 ～Webエージェント技術を活用したアプローチ～

NTT 株式会社
コンピュータ&データサイエンス 研究所
切貫 弘之

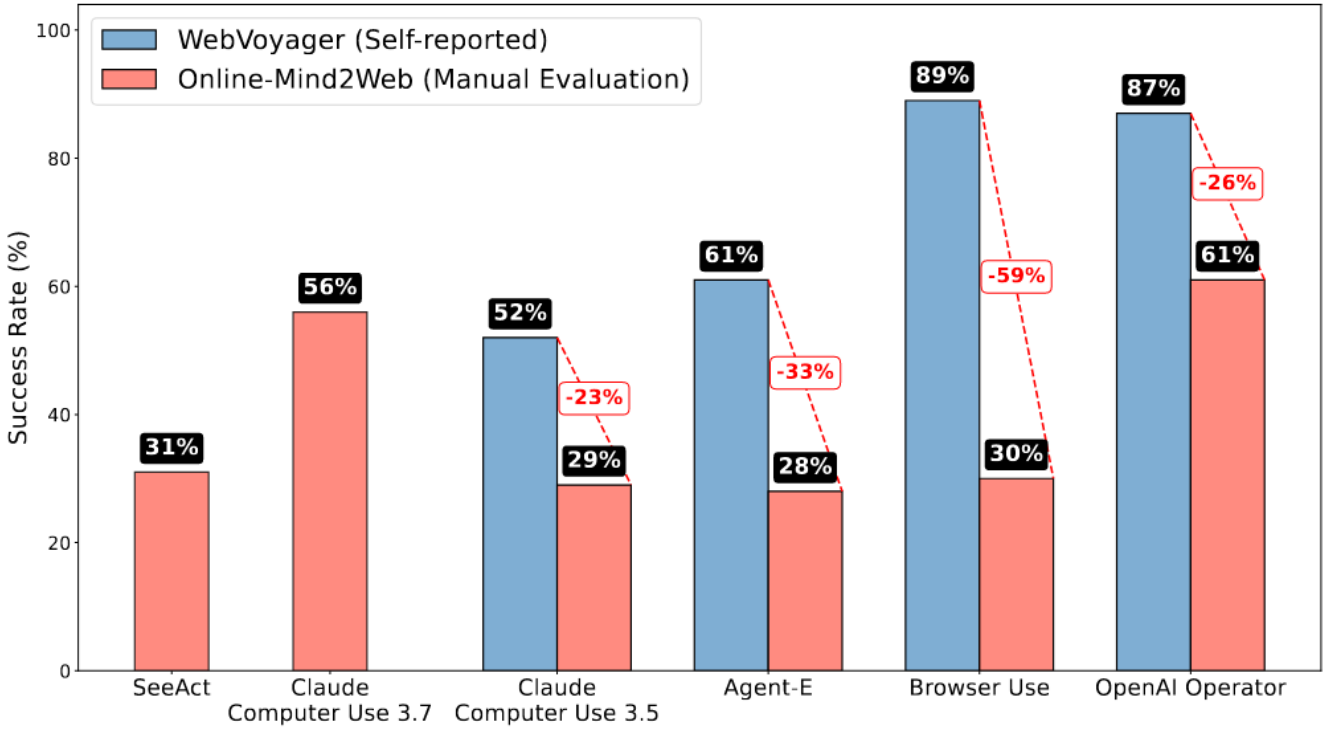
2025/09/25

Webエージェント

Browser-useをはじめとするLLMベースのWebエージェントの登場によって、
自然言語の指示を基にしたE2Eテストの自動化が現実的になりつつある

しかし、
現状のWebエージェントの性能は、信頼できる回帰テストを実行するには不十分

エージェントにテストを実行させるのではなく、エージェントの振る舞いからテストスクリプトを生成する



“An Illusion of Progress? Assessing the Current State of Web Agents.” [Tianci et al. Arxiv’25]

Webエージェントによるスクリプト生成の課題

- **複雑なWebページにおける操作対象の特定が困難**
  - 画面全体のスクリーンショットやDOMをLLMに与える手法では不十分
- **エージェントの行動を単純にスクリプト化するだけでは保守性や再利用性に乏しいテストスクリプトになる**
  - 従来のレコード＆リプレイツールと同様の問題
  - 無駄な操作が含まれる可能性がある

アイデア:

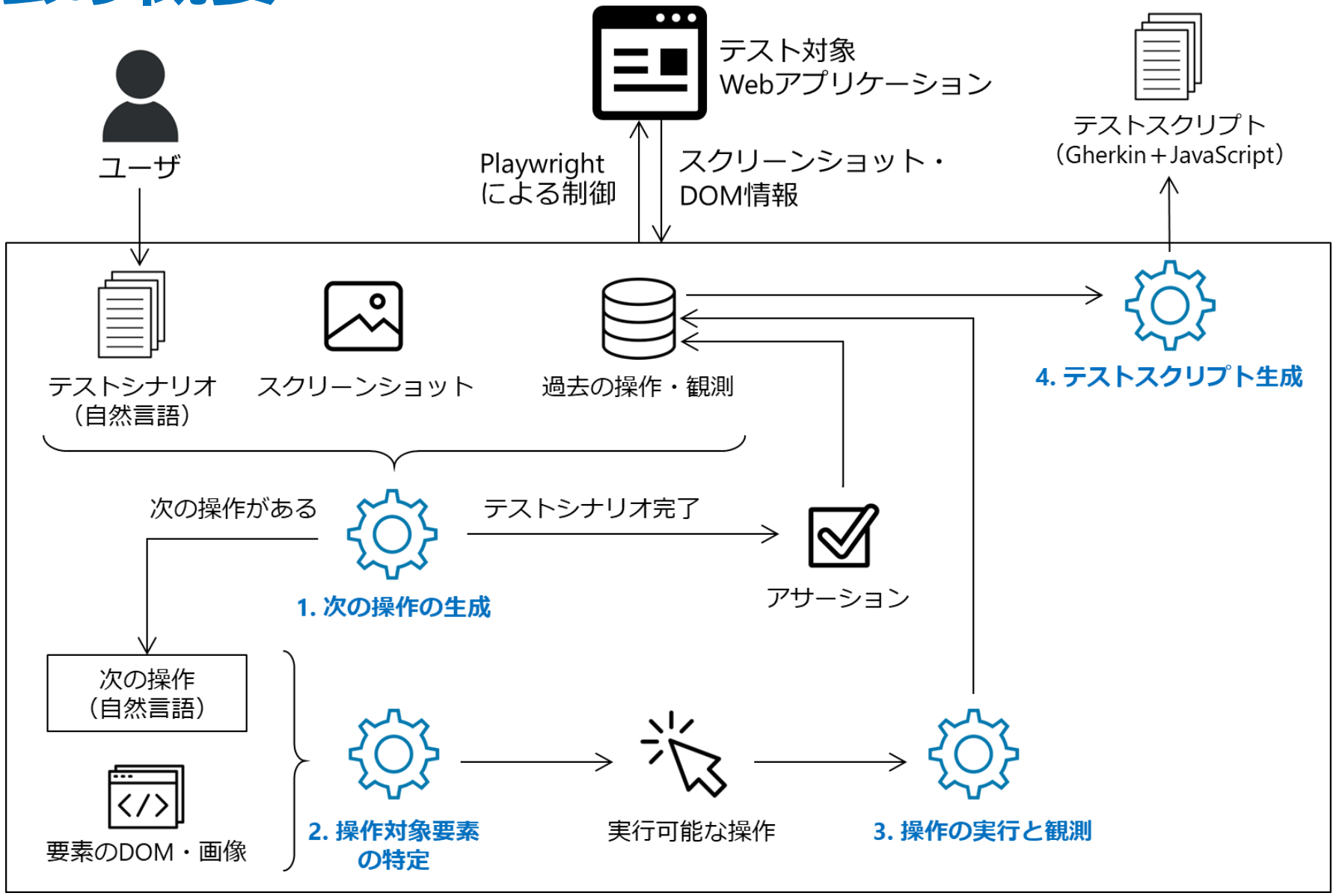
Open AI gpt-5など

マルチモーダルLLMを用いて、自然言語によるテストシナリオをGherkin形式のテストケースおよび実行可能なJavaScriptのテストステップ定義へ変換する

ポイント:

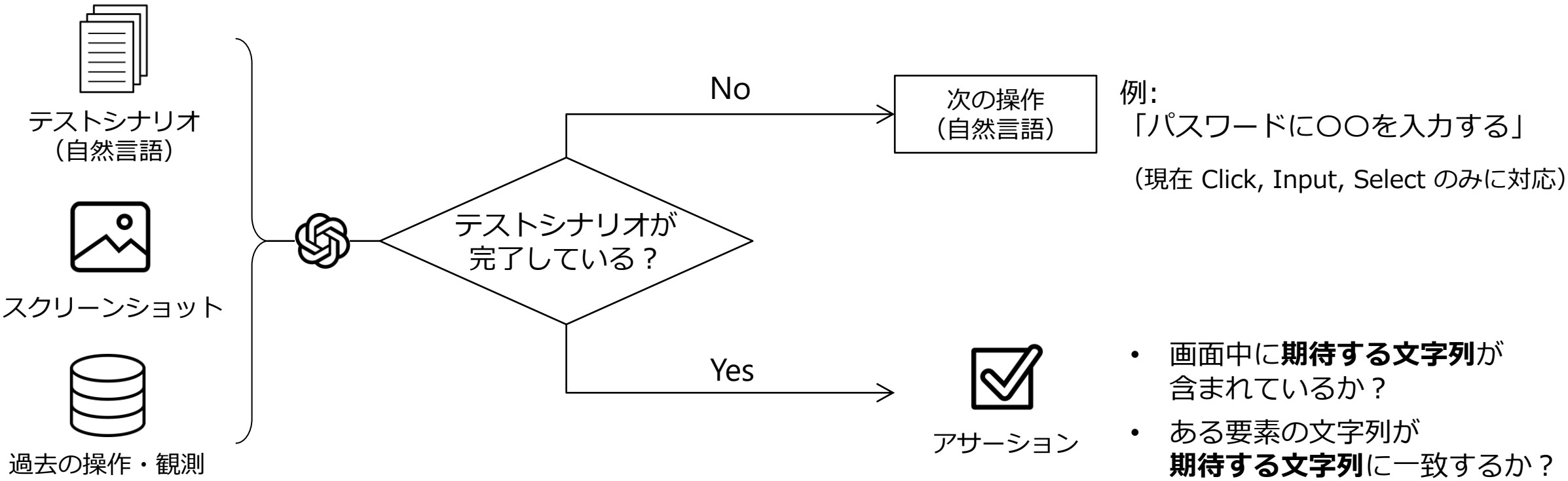
- DOMツリーを探索しながら操作対象を特定する新しいアルゴリズム
- テストスクリプトを複数の方法で抽象化し、保守性を高める

提案手法の概要



フェーズ1: 次の操作の生成

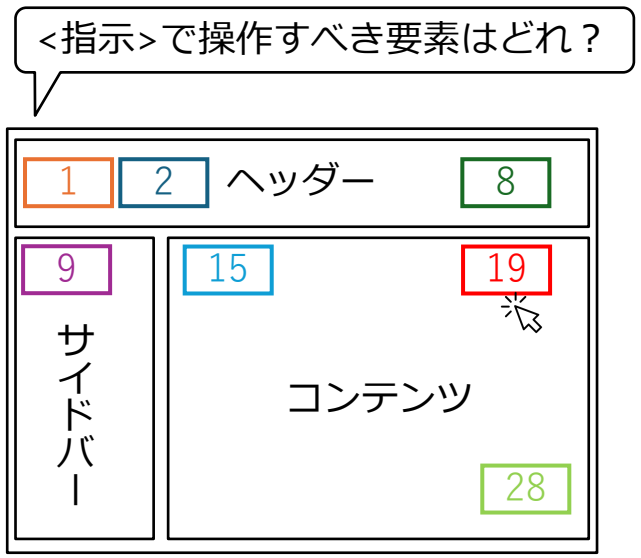
操作生成は既存研究 SeeAct [Zheng et al. Arxiv'24] のアプローチに加え、
状態の観察による失敗からのリカバリー機構を導入



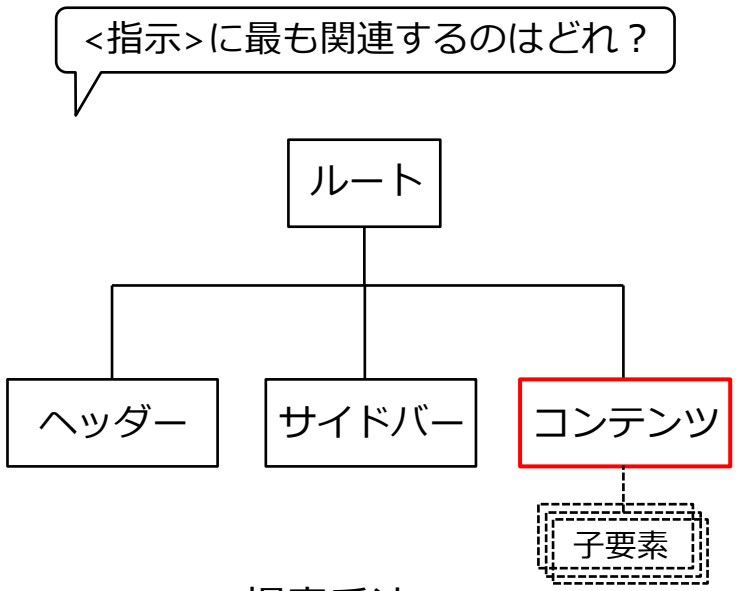
フェーズ2: 操作対象要素の特定

Webエージェントでよく用いられる **Set-of-Mark Prompting (SoM)** [Yang et al. arxiv'23]
は操作対象候補が多い画面や大きい画面では精度が下がる恐れがある

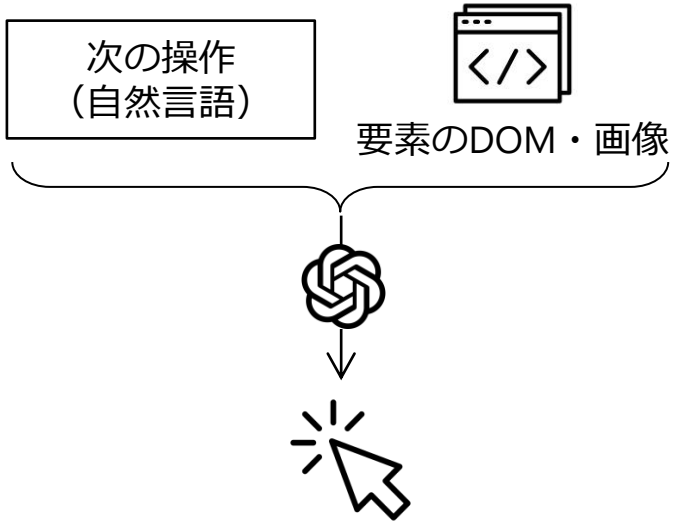
➡ DOMツリーを探索して徐々に探索範囲を絞り込む手法
Hierarchical Tree Exploration (HTE) を提案



Set-of-Mark Prompting



提案手法 (HTE)



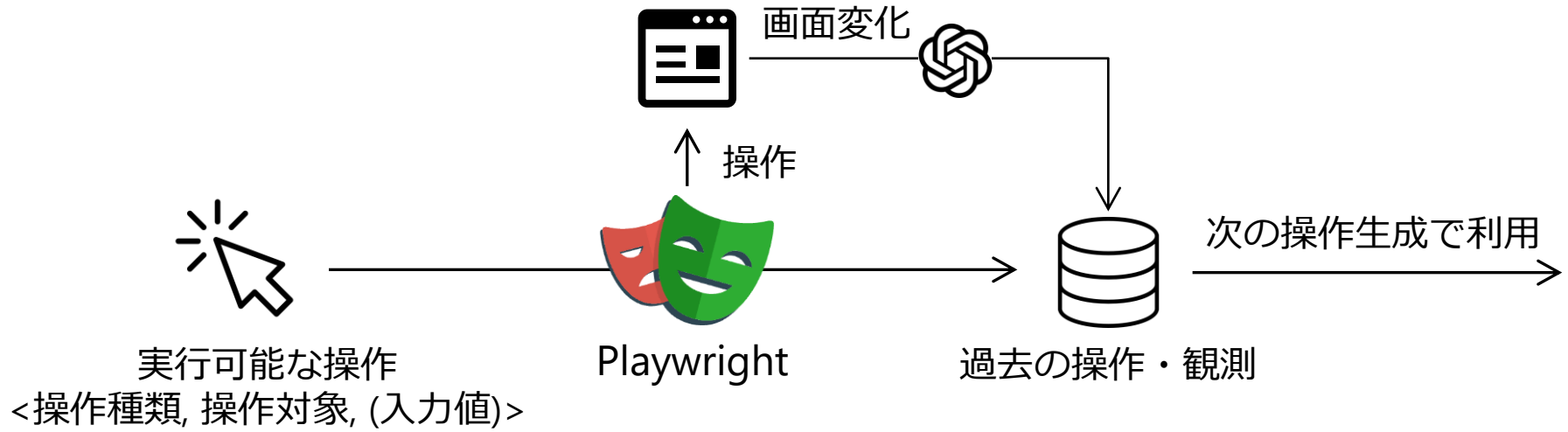
実行可能な操作

フェーズ3: 操作の実行と観察

フェーズ2で得られた操作をPlaywrightを介して実行する

- 行おうとした操作がPlaywrightのエラーで失敗した場合、その理由を分析して、操作を再生成する。
  例）直接入力が禁止された日付フォームに入力しようとしてエラーが起きたので、次はクリックする。
- 操作自体には成功したにもかかわらず、テストの手順として不適切な場合は、
  操作後の観測により軌道修正してくれることに期待する。

画面遷移や画面状態の変化があった場合、それをLLMで要約し、操作と合わせて記録する



フェーズ4: テストスクリプト生成

記録された操作手順をテストスクリプトに変換する

テストスクリプト生成の過程で以下の3つの抽象化を行う

- **入力値パラメータ化（with LLM）**
  - 入力値の違いを吸収するために操作の具体値の部分をプレースホルダーに変換する
  - 入力値と処理を分離することで、データ駆動テストが可能になる
- **シナリオ抽象化（with LLM）**
  - 実行された操作列を、ログインやユーザ登録といった高レベルなテストステップへとまとめる
  - 失敗した操作、無駄な操作を削除する
- **テストステップ再利用**
  - シナリオ抽象化の際、パラメータ化されたテストステップが既存のテストステップと一致する場合は、テストステップを新たに定義せず、既存テストステップを利用する

自動生成されたテストスクリプトの例

Gherkin形式のテストケース

```
Scenario Outline: Verify Adding a Pet Visit for an Owner and Confirming the Visit Details Displayed
  Given Open the page
  When Search for an owner by entering <lastName> in the last name field and submitting the search
  When Add a visit to the pet by clicking 'Add Visit', entering <description>, and submitting the form
  Then Assert that the string "<assertString>" exists within the page.
  Examples:
  | lastName | description      | assertString |
  | Franklin | Routine checkup | Routine checkup |
```

テストステップに対応するJavaScriptコード

```
When("Add a visit to the pet by clicking 'Add Visit', entering {}, and submitting the form", async function
(description) {
  await this.page.locator("//body[1]/div[1]/div[1]/table[2]/tbody[1]/tr[1]/td[1]
    /table[1]/tbody[1]/tr[1]/td[2]/a[1]").click();
      // Click the "Add Visit" link in the "Pets and Visits" section.
  await this.page.locator("#description").fill(description);
      // Type <description> into the "Description" input field.
  await this.page.locator("//body[1]/div[1]/div[1]/form[1]/div[2]/div[1]/ button[1]").click();
      // Click the "Add Visit" button located below the "Description" input field.
});
```

評価｜ケーススタディ

Spring PetClinic サンプルアプリケーションを対象に以下を評価

- **提案手法が自然言語シナリオから実行可能なテストスクリプトを生成できるか**
- **生成されたスクリプトがどの程度テスト手順を抽象化できるか**

2名の開発者がテスト手順は書かずに「何をテストするか」のみを記述した
21件のテストシナリオを作成

これらのテストシナリオを入力とし、
有効なテストスクリプトが得られるまで提案手法を最大3回適用

結果｜ケーススタディ

全てのテストシナリオに対し、実行可能なテストスクリプトの生成に成功

画面	#	テストシナリオ（概要）	操作数	ステップ数
オーナー検索ページ	1	オーナー検索で1件ヒットした場合、オーナーページが表示される。	3	1
	2	オーナー検索で2件以上ヒットした場合、オーナー検索結果ページに一覧表示される。	3	1
	3	オーナー検索で何も入力しなかった場合、全オーナーが検索結果ページに表示される。	2	1
	4	オーナー追加ページへ移動する。	2	1
オーナー検索結果ページ	5	オーナー名をクリックしてオーナーページへ移動する。	3	1
オーナーページ	6	ペット追加ページへ移動する。	4	1
	7	ペット編集ページへ移動する。	6	2
	8	オーナー編集ページへ移動する。	4	1
	9	訪問データ追加ページへ移動する。	4	2
オーナー追加/編集ページ	10	入力欄を埋めてオーナーを追加する。	8	1
	11	入力欄が空欄のままオーナーを追加する。	3	2
	12	入力欄を埋めてオーナーを編集する。	7	2
	13	入力欄が空欄のままオーナーを編集する。	7	3
ペット追加/編集ページ	14	入力欄を埋めてペットを追加する。	9	2
	15	入力欄が空欄のままペットを追加する。	5	3
	16	入力欄を埋めてペットを編集する。	6	2
	17	入力欄が空欄のままペットを編集する。	6	3
訪問データ追加ページ	18	入力欄を埋めて訪問データを追加する。	6	2
	19	入力欄が空欄のまま訪問データを追加する。	5	2
ヘッダ	20	獣医一覧ページへ移動する。	1	1
	21	エラーサンプルページへ移動する。	1	1

全操作数

95

⇩ 再利用

JSコード上の操作数

69

⇩ 抽象化

テストステップ数

35

評価｜Web要素特定戦略の有効性

提案手法HTEの要素特定能力をSoMと比較
ケーススタディとは異なり、実在の複雑なWebサイトで評価

対象:

Mind2Webデータセット[Deng et al. arxiv'23]のオンライン評価用サブセットを利用
全90タスクから対象Webサイトがユニークになるように51タスクを抽出

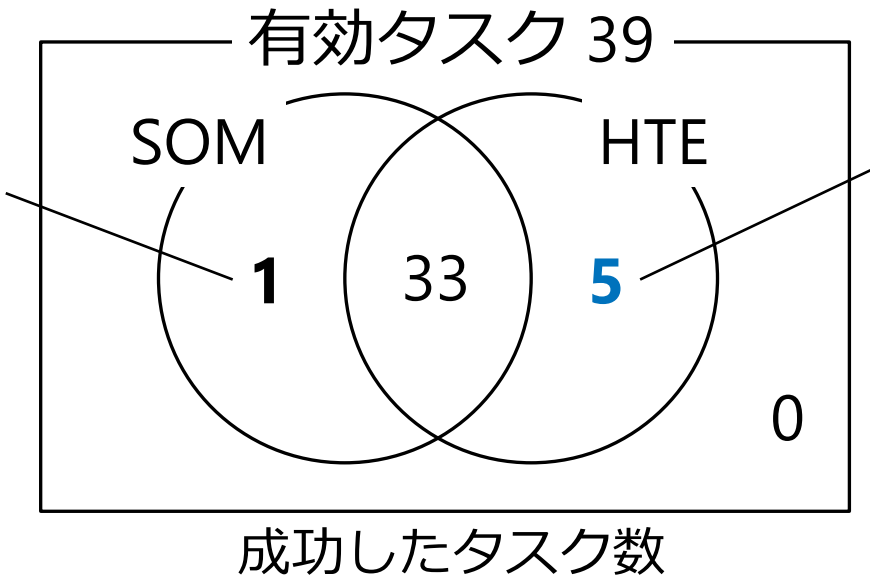
実験方法:

- 要素特定能力のみを測るため、タスクの最初の操作が正しく行えた件数をカウント
  （操作生成フェーズで間違うと、その後のプロセスが無意味になってしまう）
- 操作生成フェーズはHTEとSoMともに提案手法を採用
- 実験は各タスクに対し最大3回行い、1回でも成功すればOKとした

結果｜Web要素特定戦略の有効性

51件中12件は、操作生成時点の失敗もしくは実装上の問題（Shadow DOM・iframe未対応など）で操作対象要素特定フェーズまで到達できなかったため除外

HTEにおいて、Web要素画像の取得に失敗



HTEでのみ成功

要素の多い画面や複雑な画面において、HTEが有効である可能性を示唆

現状の課題

シナリオ抽象化における一貫性の欠如

例えば、操作列 [a, b, c] で構成される2つのテストケースT_1, T_2が、それぞれ T_1 = [[a], [b, c]], T_2 = [[a, b], [c]] というテストステップにまとめられた場合、類似するテストステップ {[a], [a, b], [b, c], [c]} が生成され、生成されるテストスクリプトが冗長になってしまう

➡ **シナリオ抽象化時に生成済みのテストステップを参照することで、一貫性を持たせる**

シナリオ実行の安定性の欠如

失敗からのリカバリー機構を導入しているが、複雑なアプリにおいては次に行うべき操作の生成に失敗する場合がある

➡ **既存手法のより高度なプロンプティング技術や機械学習技術と組み合わせる**

まとめ

Webエージェント技術を活用し、自然言語のテストシナリオから
テストスクリプトを生成する手法を提案した

ポイント:

- DOMツリーを探索しながら操作対象を特定する新しいアルゴリズム
- テストスクリプトを複数の方法で抽象化し、保守性を高める

手法の効果:

- テストスクリプト実装が容易になり、E2Eテスト自動化のハードルを下げる
- 生成されたテストケースが自然言語で表現されている上、テストステップが共通化されているため、保守性しやすい

今後の展望:

アプリケーションを自動的に探索し、LLMにテストシナリオを生成させることで、
テスト設計の負担を減らし、テストスクリプト生成の全自動化を目指す