

RepairAgent: An Autonomous, LLM-Based Agent for Program Repair

Islem Bouzenia¹, Premkumar Devanbu², Michael Pradel¹

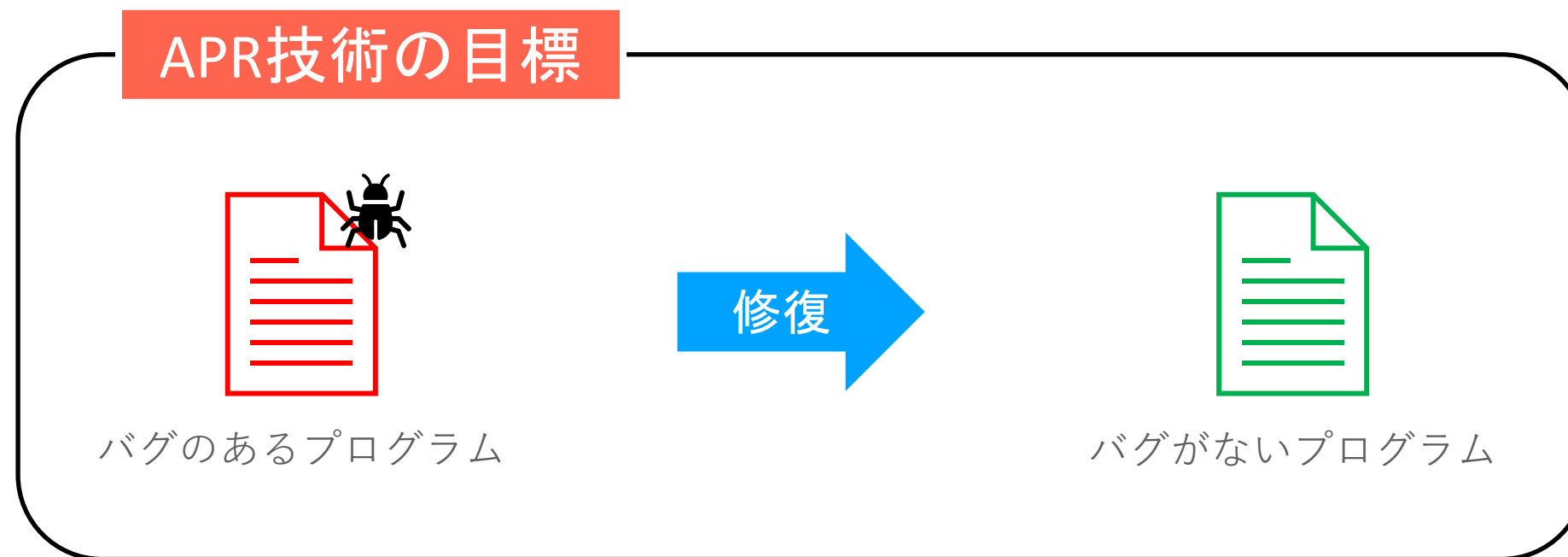
¹University of Stuttgart, Germany

²UC Davis, USA

背景：自動プログラム修復

近年、自動プログラム修復(APR)技術が活発に研究されている

APR技術の目標：バグのあるプログラムの自動修復



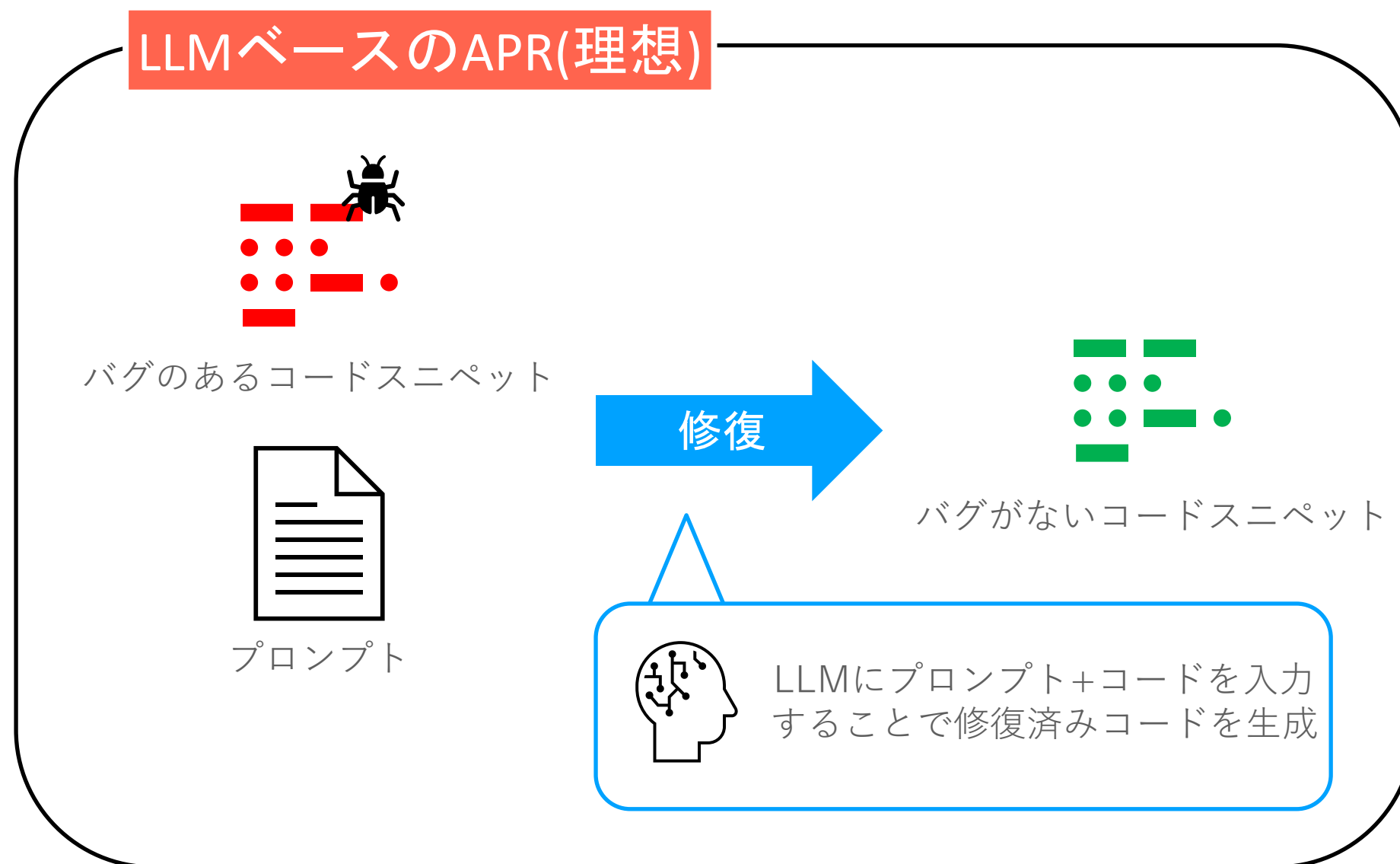
背景：APRの既存手法

これまで、APR実現のために様々な手法が提案された

- 検索ベース
- 制約ベース
- テンプレートベース
- ニューラル機械翻訳(NMT)ベース

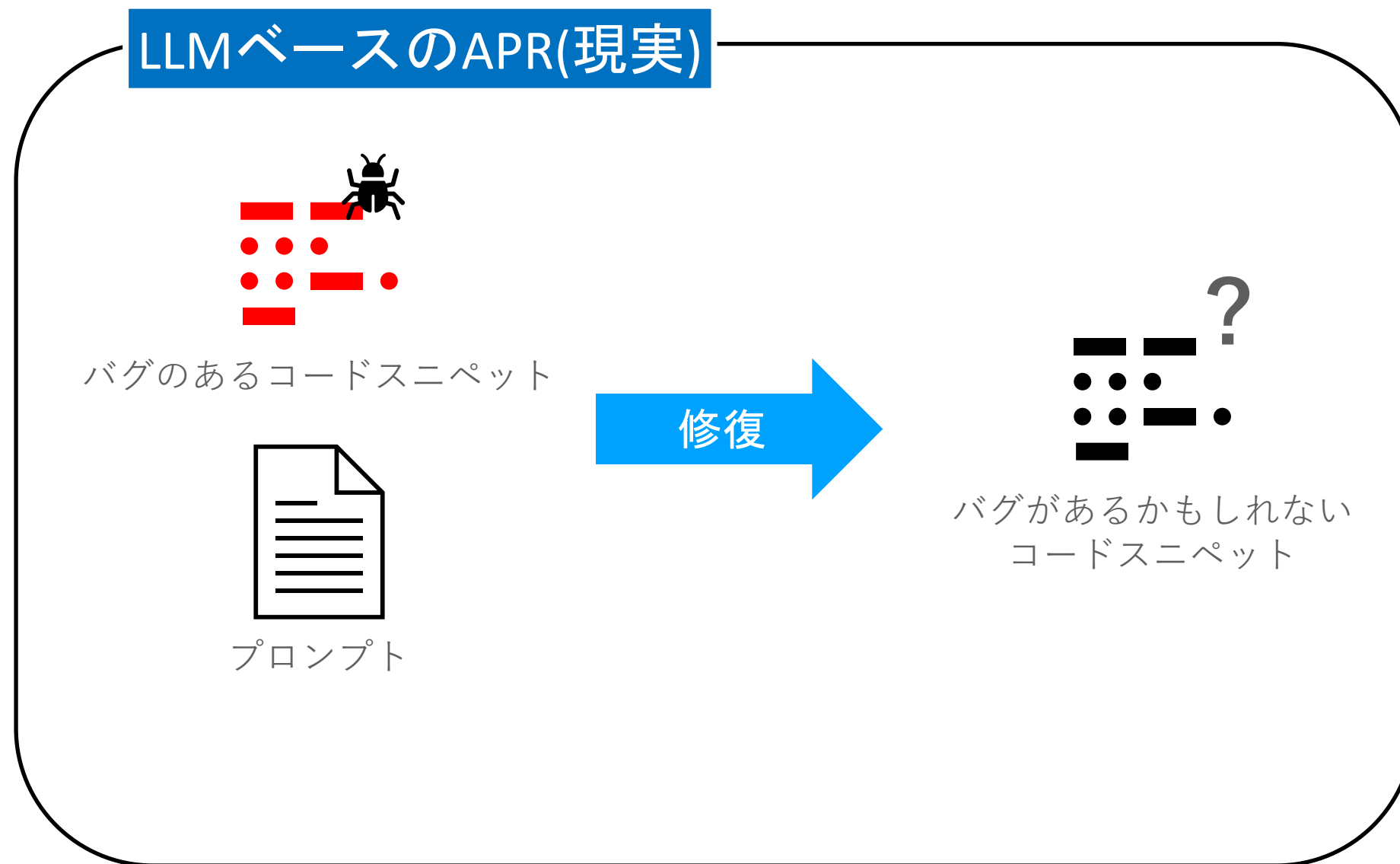
背景：LLMベースのAPRの理想

近年はLLMベースのAPR手法が主流となっている



背景：LLMベースのAPRの現実

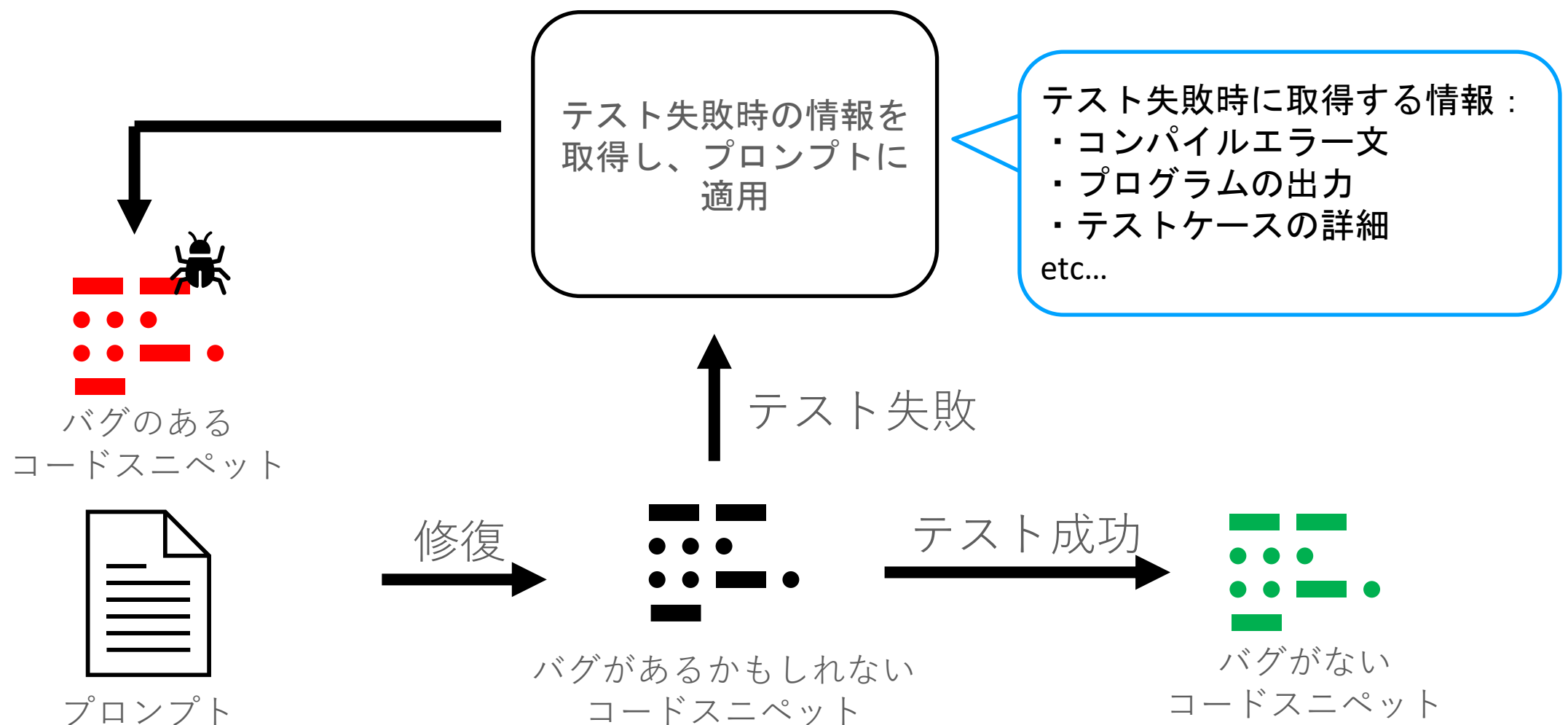
しかし、1回の修復では必ずしもバグが修復されとは限らない



背景：LLM+フィードバックループによるAPR

そこで、LLM+フィードバックループによるAPR手法 (ChatRepair等)が提案されている

LLM+フィードバックループによるAPR

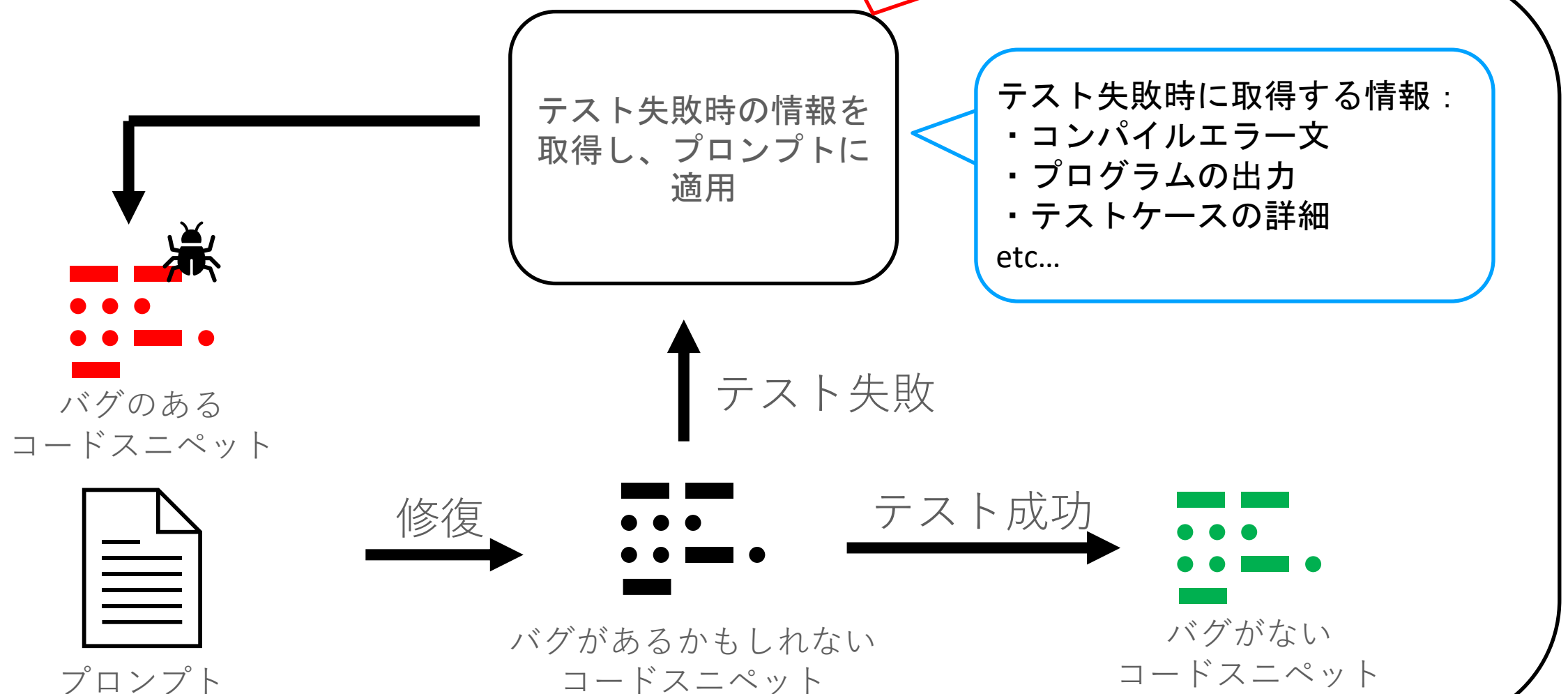


背景：LLM+フィードバックループによるAPR

そこで、LLM+フィードバックループによるAPR (ChatRepair等)が提案された

しかし、既存手法では「テスト失敗時に予め決められた情報しか取得できない」という問題がある

LLM+フィードバックループによるAPR



背景：LLMエージェント

LLMを中心に、自律的に行動するソフトウェアシステムを**LLMエージェント**という

- LLMエージェントは、以下2つの特徴を持つ技術を指す
 1. 目標を達成するためのアクションを、自律的に計画し、実行する
 2. LLMが実行するアクションに、外部ツールの呼び出しが含まれる

仮説：LLMエージェントを用いれば、自律的にテスト失敗時の情報収集を行えるAPR技術が実現できるのではないか

研究概要

目的：自律的にテスト失敗時の情報収集ができるAPR手法の確立

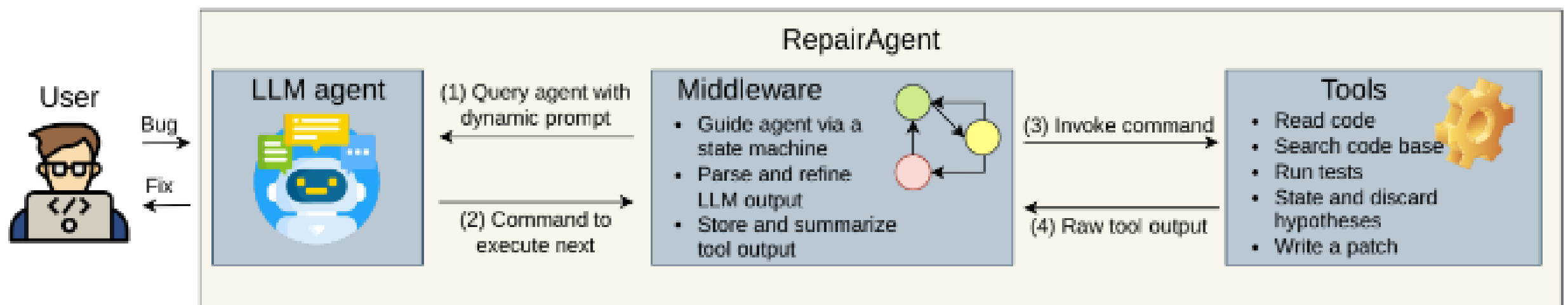
アプローチ：

- LLMエージェントを活用したAPR技術の実現及び評価

アプローチ：RepairAgent

図：提案手法の概要を示す図

- ミドルウェアの存在がキモ
- ファイル中のバグ発生箇所特定→バグ修復までの流れを(1)~(4)までを繰り返すことで実現
- APIの予算を使い切るまで実行され続ける

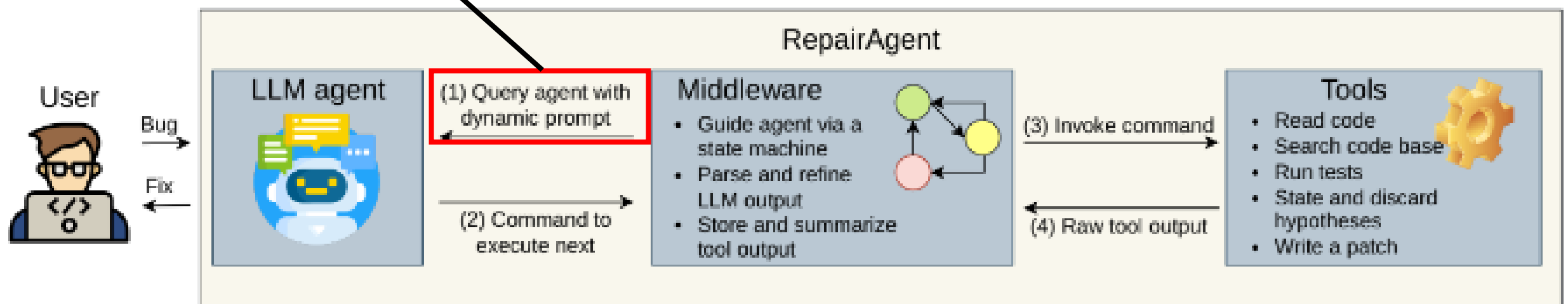


アプローチ：RepairAgent

図：提案手法の概要を示す図

- ミドルウェアの存在がキモ
- (1)~(4)までを繰り返し実行する
- APIのミスを使い切るまで実行され続ける

(1)LLMにプロンプトを入力する
※2回目以降はフィードバック
によってプロンプトの内容を
変えていく

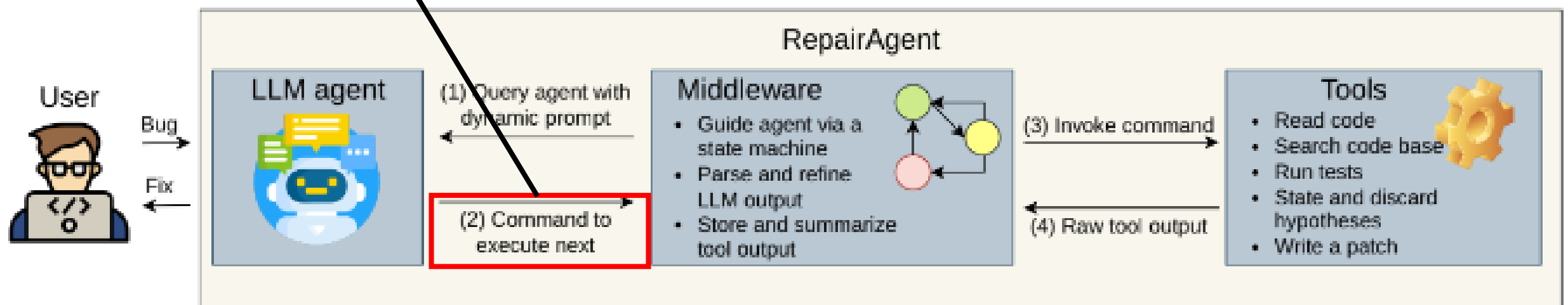


アプローチ：RepairAgent

図：提案手法の概要を示す図

- ミドルウェアの存在がキモ
- (1)~(4)までを繰り返し実行する
- APIの予算を使い切るまで実行され続ける

(2)次に実行するコマンドをLLMから受け取る

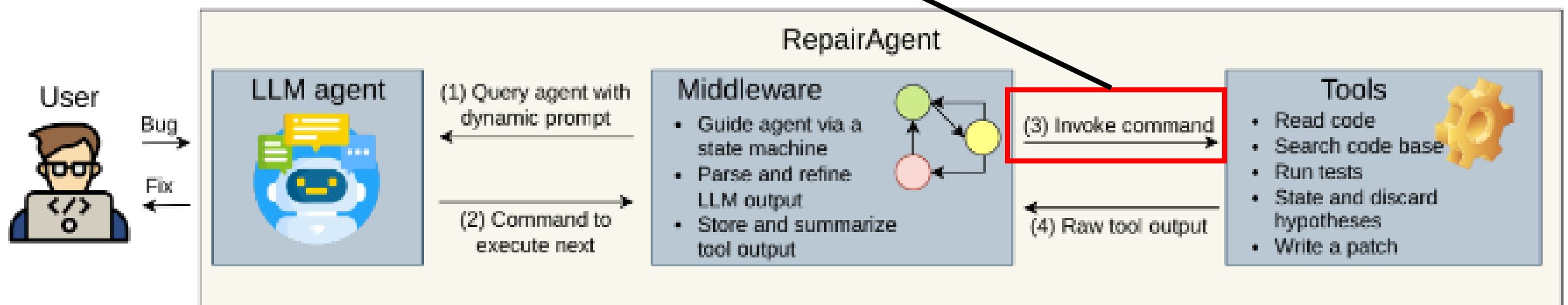


アプローチ：RepairAgent

図：提案手法の概要を示す図

- ミドルウェアの存在がキモ
- (1)~(4)までを繰り返し実行する
- APIの予算を使う

(3)コマンド(ツール)の実行を行う

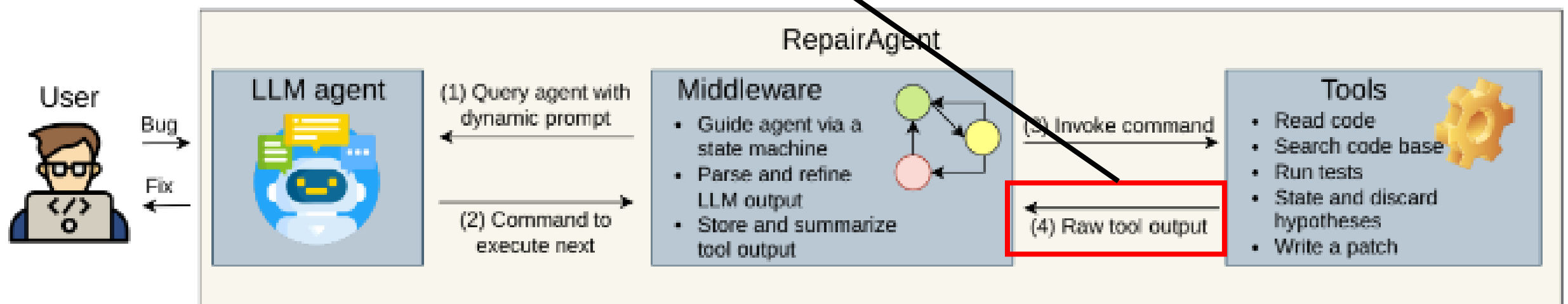


アプローチ：RepairAgent

図：提案手法の概要を示す図

- ミドルウェアの存在がキモ
- (1)~(4)までを繰り返し実行する
- APIの予算を使う

(4) コマンド(ツール)の実行結果を
ミドルウェアに出力



結果：実験設定

データセット：

- Defect4Jのversion1.2とversion2を合体させたデータセット

ベースライン

- ChatRepair(LLMを用いた反復的APRアプローチ)
- ITER(DNNを用いた反復的APRアプローチ)
- Self-APR(NN or DNNを用いた反復的APRアプローチ)

使用LLM

- GPT-3.5-0125

結果：RepairAgentは、実際のバグを修正するのに どれほど効果的か

表：Defect4Jデータセット
を使って、RepairAgentに
APRを行わせた結果

各項目の説明：

- Bugs: そのプロジェクトに含まれるバグの数
- Plausible: 実際の開発者の修正とは違うが、テストケースは通過する修正の数
- Correct：実際の開発者の修正と一致する修正の数

Project	Bugs	Plausible	Correct
Chart	26	14	11
Cli	39	9	8
Closure	174	27	27
Codec	18	10	9
Collections	4	1	1
Compress	47	10	10
Csv	16	6	6
Gson	18	3	3
JacksonCore	26	5	3
Jackson databind	112	18	11
JacksonXml	6	1	1
Jsoup	93	18	18
JXPath	22	0	0
Lang	63	17	17
Math	106	29	29
Mockito	38	6	6
Time	26	3	2
Total	835	186	164

結果：RepairAgentは、実際のバグを修正するのに どれほど効果的か

表：Defect4Jデータセット
を使って、RepairAgentに
APRを行わせた結果

各項目の説明：

- Bugs: そのプロジェクトに
含まれるバグの数

割合に換算すると・・・
Plausible: 22.3%
Correct: 19.6%

- Plausible: 修正と見做される修正の数
- Correct: 実際の開発者の修正
と一致する修正の数

Project	Bugs	Plausible	Correct
Chart	26	14	11
Cli	39	9	8
Closure	174	27	27
Codec	18	10	9
Collections	4	1	1
Compress	47	10	10
Csv	16	6	6
Gson	18	3	3
JacksonCore	26	5	3
Jackson databind	112	18	11
JacksonXml	6	1	1
Jsoup	93	18	18
JXPath	22	0	0
Lang	63	17	17
Math	106	29	29
Mockito	38	6	6
Time	26	3	2
Total	835	186	164

結果：RepairAgentは、実際のバグを修正するのに どれほど効果的か

表：Defect4Jデータセットを使って、RepairAgentと既存手法(ChatRepair等)でそれぞれAPRを行わせた結果

この表におけるスコアは実際の開発者の修正と一致する修正の数となっている

Project	RepairAgent	ChatRepair	ITER	SelfAPR
Chart	11	15	10	7
Cli	8	5	6	8
Closure	27	37	18	20
Codec	9	8	3	8
Collections	1	0	0	1
Compress	10	2	4	7
Csv	6	3	2	1
Gson	3	3	0	1
JacksonCore	3	3	3	3
Jackson databind	11	9	0	8
JacksonXml	1	1	0	1
Jsoup	18	14	0	6
JXPath	0	0	0	1
Lang	17	21	0	10
Math	29	32	0	22
Mockito	6	6	0	3
Time	2	3	2	3
Total	164	162	57	110

結果：RepairAgentは、実際のバグを修正するのに どれほど効果的か

表：Defect4Jデータセットを使って、RepairAgentと既存手法(ChatRepair等)でそれぞれAPRを行わせた結果

この表におけるスコアは
実際の開発者の修正と一
致する修正の数となっ
てい

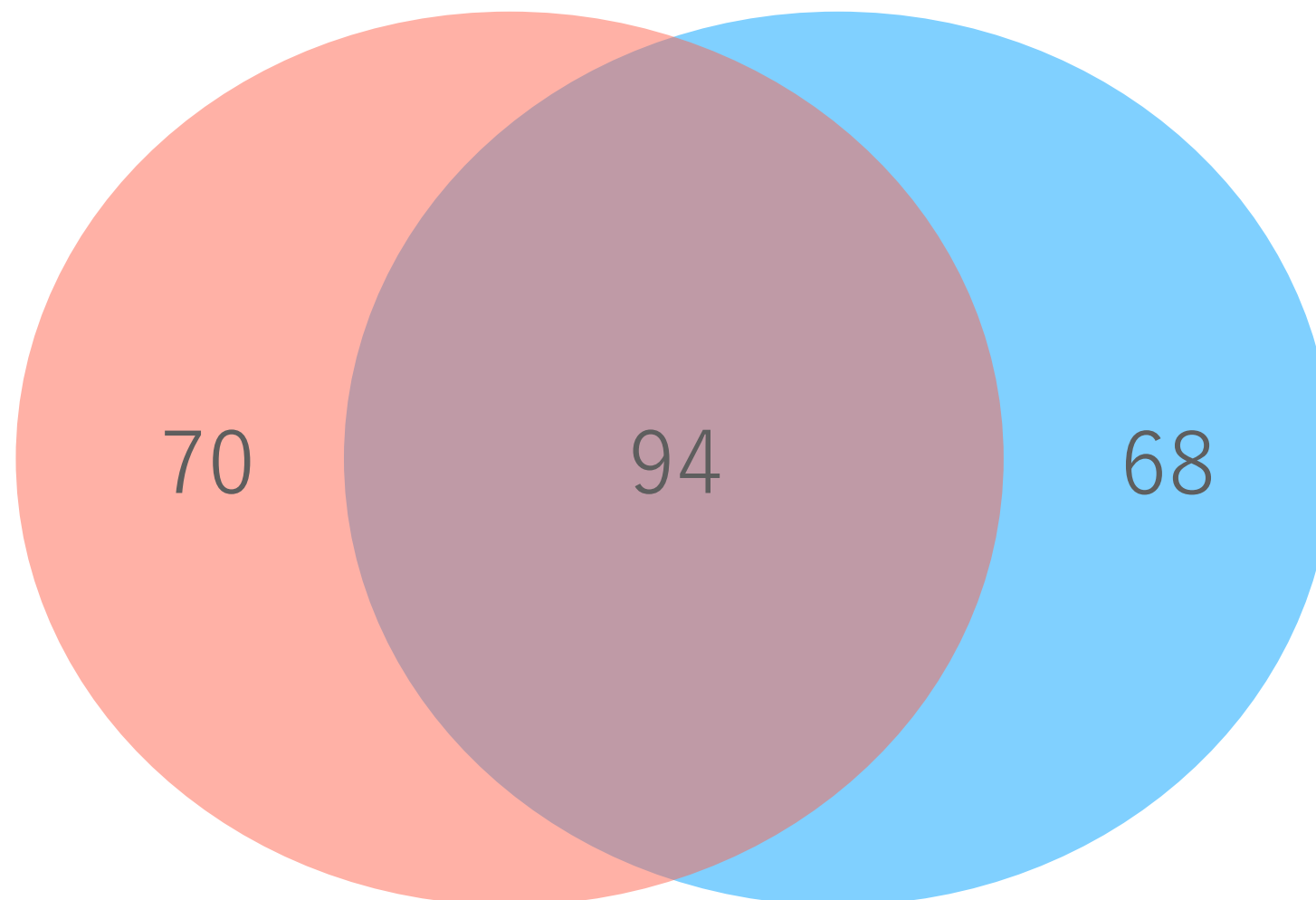
RepairAgentとChatRepairで正しく
修正できたバグの数はほぼ同じ

Project	RepairAgent	ChatRepair	ITER	SelfAPR
Chart	11	15	10	7
Cli	8	5	6	8
Closure	27	37	18	20
Codec	9	8	3	8
Collections	1	0	0	1
Compress	10	2	4	7
Csv	6	3	2	1
Gson	3	3	0	1
JacksonCore	3	3	3	3
Jackson databind	11	9	0	8
JacksonXml	1	1	0	1
Jsoup	18	14	0	6
JXPath	0	0	0	1
Lang	17	21	0	10
Math	29	32	0	22
Mockito	6	6	0	3
Time	2	3	2	3
Total	164	162	57	110

結果：RepairAgentは、実際のバグを修正するのに
どれほど効果的か

下図：Defect4jに含まれるバグについて、各手法で修復
できた数を表すベン図

- 赤：RepairAgent
- 青：ChatRepair



結果：RepairAgentは、実際のバグを修正するのに どれほど効果的か

で修復

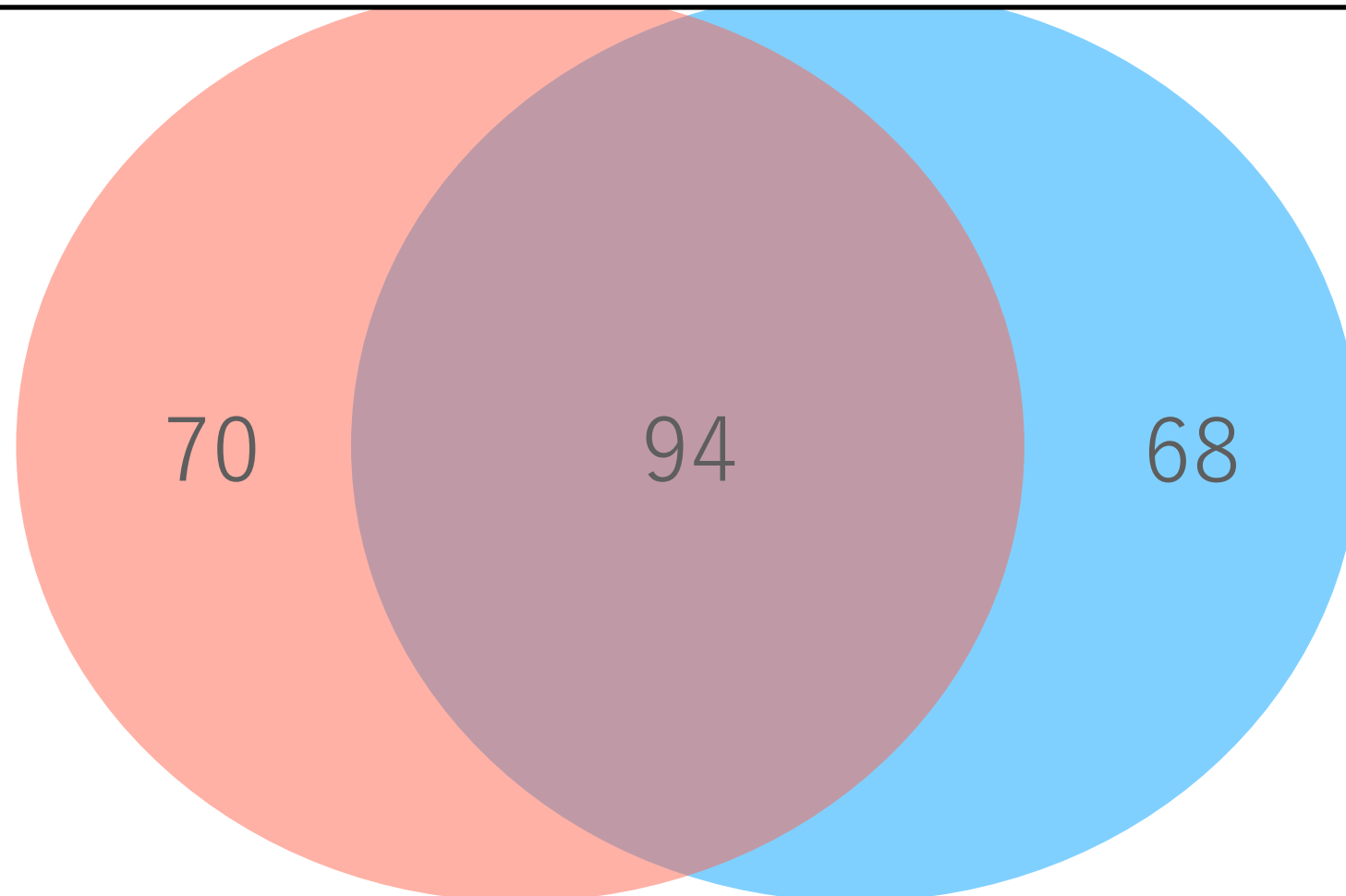
下図
でき

- 赤
- 青

RepairAgentのみ、ChatRepairのみで修復できたものがそれぞれ一定数
ある

↓

RepairAgentとChatRepairは同じようなバグしか修復できないというわけ
ではない



結論

- | 本研究では、LLMエージェントによるバグ回復のたえの先駆的な技術を示した。
- | 実験により、提案アプローチの有効性と可能性を検証した
- | LLMエージェント技術をさらに探究し、洗練するとより多様なバグへの一般化が可能になるだろう

所感

LLMエージェントを使って尚、修復できるバグの数が2割弱に収まるという事実が意外だった

- もちろん、使っているLLMのモデルが古いので、現在の我々の感覚とずれている可能性は十分あり得る

既存手法(ChatRepair)と比べても修正できたバグの数自体は変わらないことにも驚いた

- ChatRepairとRepairAgentそれぞれで固有に修復できるバグがある
事実は、LLMエージェントが既存手法の完全上位互換となりえないことの何よりの証拠