AI in Debugging and Code Security

By

Shoubhik Das

Agenda

- AI in debugging
  - Feature Extraction
  - Error Localization
  - Fix Suggestion
  - Pros and Cons
- AI in Code Security
  - Static Code analysis
  - Buggy code pattern similarity
  - Error localization
  - Security patch Suggestion
  - Pros and Cons

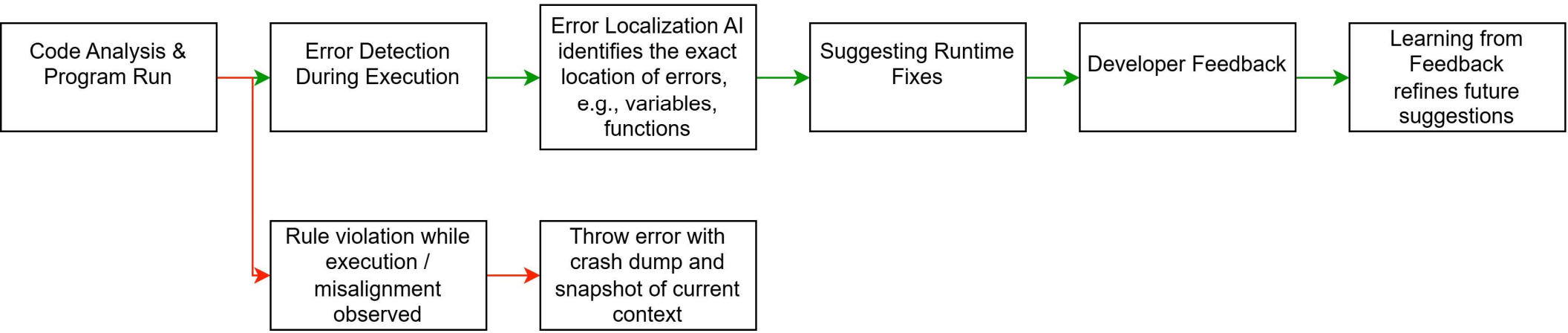
AI in Debugging

- Debugging deals with the run time or environment related bugs.
- AI agents for debugging integrates seamlessly with popular IDEs and popular debuggers as plugins. The running program has to be instrumented based on compiled or interpreted language.

How is AI-based Debugging different

- AI based agents have a learning-based approach, whereas agents are rule based.
- AI based agents are adaptable and have better learning curve whereas rule based ones are less contextual.
- Rule-based debuggers just propose the fix by using pattern recognition, while AI based proposes the fix along with the reason to do so.
- Eg for AI based debuggers: Deepcode, CodeQL
- Eg for rule based debuggers: gdb-peda, olly dbg

How is AI-based Debugging different



Feature Extraction

- AI-based agents involves extracting the features through learning, like previous bug patterns, data structures and so on which is pretty much adaptable, while rule based ones rely on hardcoded rules to do a formal testing which is a much rigid approach.
- AI-based agents uses anomaly detection for unusual program behavior, while rule based ones keep a track on execution traces to find any violations.

Error Localization

- AI-based agents understand the context of the error, while a rule based agent checks for model violations and/or misalignments
- AI-based agents uses contextual data and runtime feedback to triage to error locations, while rule-based ones use formal analysis and state transitions.
- AI-based agents localization may improve over a period of time, while rule-based ones localization needs to be manually found from stack trace and crash dumps with the corresponding context details.

Fix suggestion

- Based on previous encountered patterns the AI based agents suggests fixes like refactoring code, adding error handling, or correcting logic flaws, while rule based ones Fixes are based on violation resolution.
- AI Based suggestions are context aware and explain the need for fixes, while rule based suggestions are rigid and rules based and does not have any explanation for the same.
- Once if a patch is enforced by AI agent it is taken as a feedback and improves future suggestions, while rule based does not have a feedback loop.

Code vulnerable to integer overflow

```
#include<stdio.h>
int test(int i)
{
     arr[2147483648];
    if(i > 2147483648)
     {        printf(“Sorry limited buffer space\n");
     }
     else
     {
       memcpy(arr,some_buffer,strlen(some_buffer));
     }
}
int main(int argc, char const *argv[])
{
   test(2147483649);
   return 0;
}
```

Pros and cons of AI based debuggers

- Pros
  - Faster Bug Detection
  - Highly Scalable
  - Context aware suggestions
  - Adaptable
- Cons
  - Huge training data requirements
  - Black boxed in nature
  - False positives and negatives
  - Huge computation needs

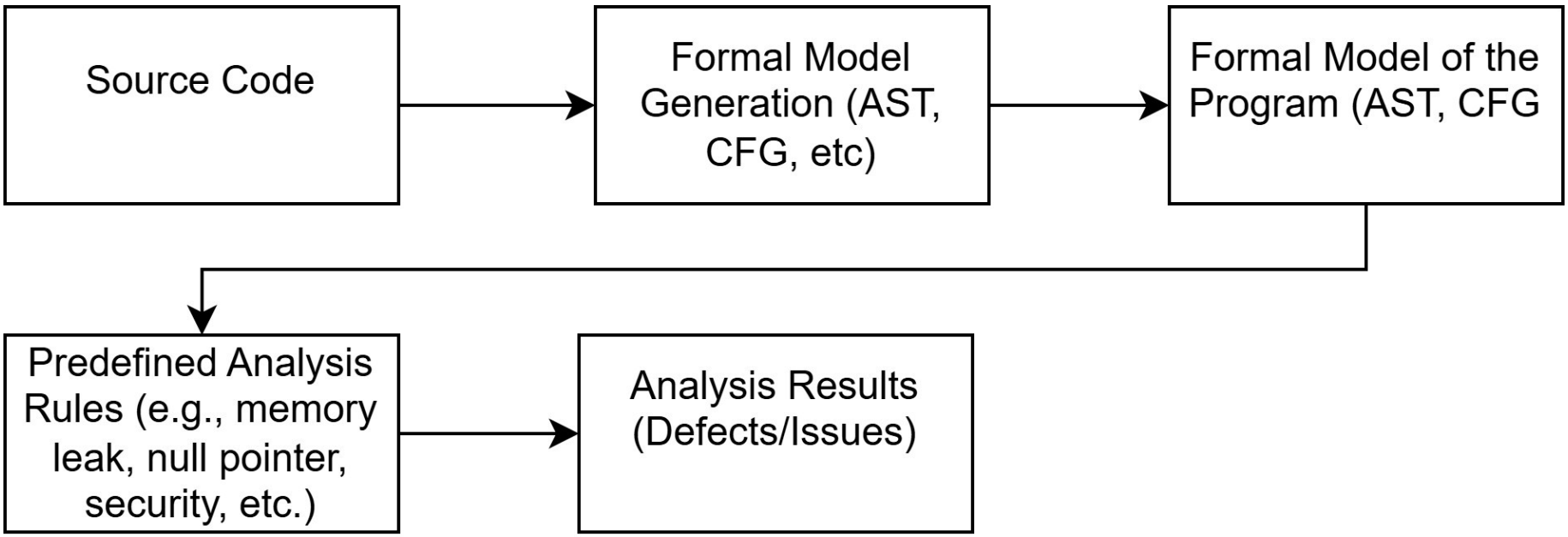
AI in Code Security

- It mainly deals with code walk-through, break-down of the entire program and analysis of any vulnerable code from a prebuilt codebase.
- It is a static procedure since the program code is analyzed by parsing the code without executing it.

Static Code Analysis

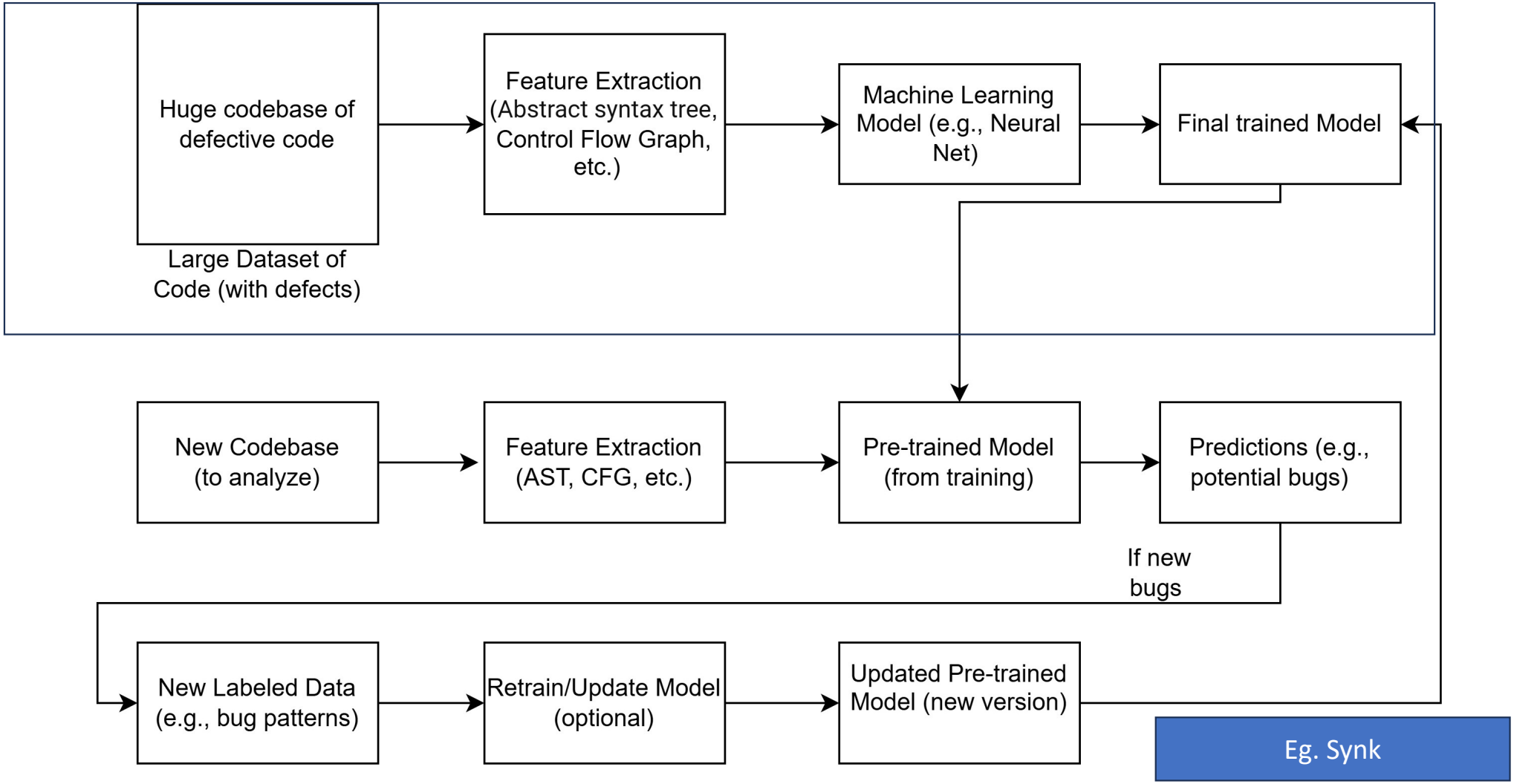
- This involves analysis of source code to identify potential security flaws which might be missed during manual review. This includes issues like improper input validation, insecure API calls, or use of weak encryption algorithms.

Training of rule based SAST



Eg. plugins for Visual Studio for various languages

Training of AI based SAST



Static Code Analysis

- From the previous slides we can conclude that rule based (Static Application Security Testing) SAST tools do more of a formal verification of the code by parsing it.
- Whereas an AI based SAST tool will perform a data driven approach to learn and predict bugs in the given code.

Static Code Analysis

```
int add(int a, int b) {
    int sum = a + b;
    return sum;
}
int divide(int a, int b) {
        return a / b;
}
main()
{
    divide(0);
}
```

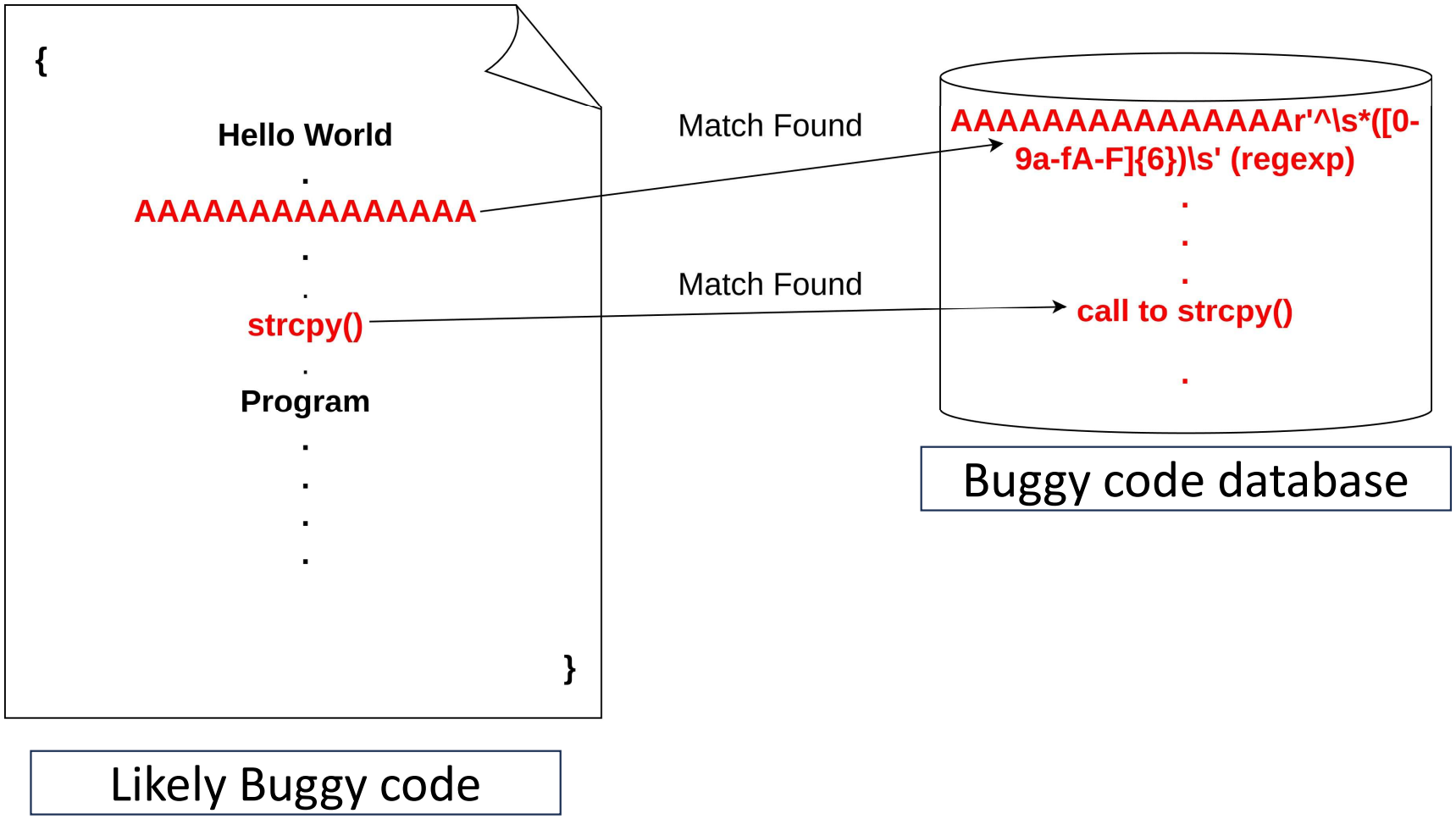
AI based SAST

rule based SAST

Buggy code pattern similarity

- This involves getting equipped with the bugs encountered across various developer communities across various projects and vast sources around the globe.
- This will help encountering the buggy code pattern with a corresponding mitigation technique.
- This will help in suggesting the user about the coding patterns that might lead to a bug, thus improving the coding practices.
- Eg. Checkmarx

Training Process



Error Localization

- Involves pinpointing the scope for the vulnerable code.
- The scope might extend from a method to a component or even to the entire system/application.
- This step is possible due to feature extraction which helps in providing a better code coverage and predicting the mitigations that is viable within the scope.
- Eg. Copilot

Security Patch Suggestion

- The model predicts fixes by pattern matching based on known pattern for common bugs that has been learnt at the time of the learning phase.
- However, it also uses context specific approaches to propose a new code as per the current context.
- Eg. Copilot

Pros and cons of AI based SAST

- Pros
  - Learning from data and evolving
  - More Scalable
  - Takes real time feedback
  - Broad issue coverage
- Cons
  - False negatives/Positives
  - Black Box in nature
  - Training dataset should be huge

List of tools used

- GDB-PEDA
- https://zzzcode.ai/
- Visual Studio Code with plugin
- GCC
- Copilot

Thank You