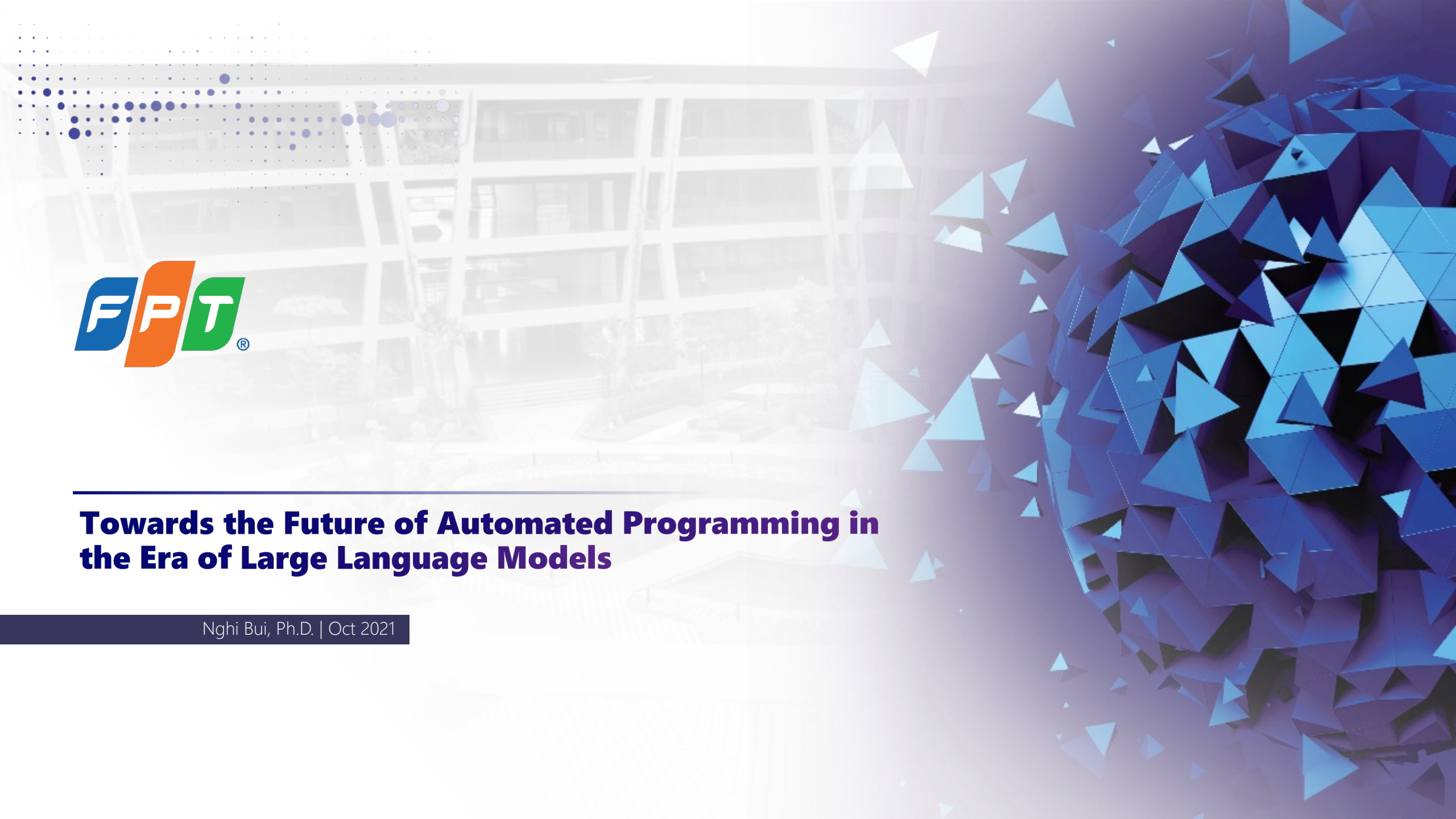


Towards the Future of Automated Programming in the Era of Large Language Models

Nghi Bui, Ph.D. | Oct 2021

About the Speaker

01/2020

SMU Singapore Management University — School of Computing and Information Systems

Ph.D. in Computer Science
School of Computing & Information Systems,
Singapore Management University

https://bdqnghi.github.io/

Topics:
- Combining static analysis with deep learning for automated program understanding

2019-2021

HUAWEI

Principal Research Scientist,
Software Trustworthy Lab - Huawei Research Center, Ireland

Lero — The Irish Software Research Centre

Topics: https://trusted-programming.github.io/
- AI-based C2Rust Migration.
- Program understanding
- Bug Detection & Repair

Now

FPT Software

Mentor/Principal Scientist,
AI Residency Program
FPT Software AI Center

SMU Singapore Management University — School of Computing and Information Systems

Research Manager/Adjunct Faculty
Intelligent Software Lab, SMU X Salesforce Research Asia

Software Lab, University of Stuttgart

Topics:
- Large language models of code.
- Code quality scanning.
- Code understanding & generation

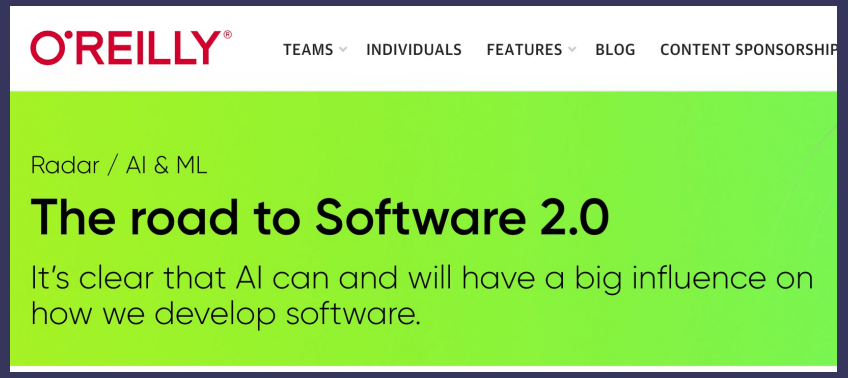
Carnegie Mellon University

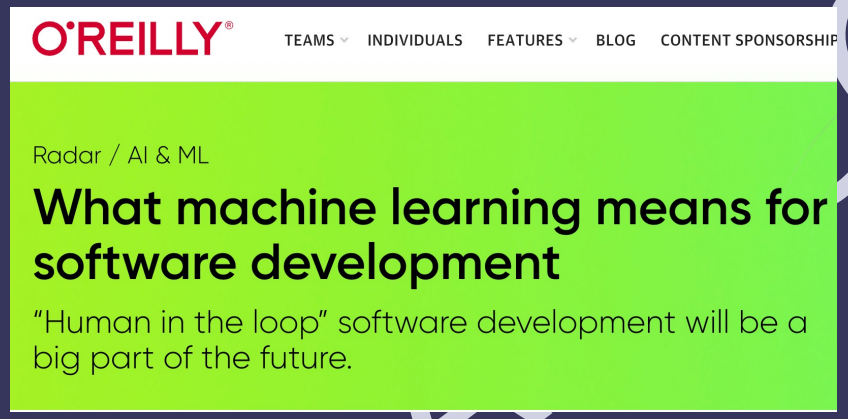
Salesforce Research , Asia

Ministry of Education SINGAPORE

Software (1.0) is eating the world, and now AI (Software 2.0) is eating software.

Andrej Karpathy, Director of AI at Tesla





The "classical stack" of **Software 1.0** is what we're all familiar with — it is written in languages such as Python, C++, etc. It consists of explicit instructions to the computer written by a programmer. By writing each line of code, the programmer identifies a specific point in program space with some desirable behavior.

In contrast, **Software 2.0** is written in much more abstract, human unfriendly language, such as the weights of a neural network. No human is involved in writing this code because there are a lot of weights (typical networks might have millions), and coding directly in weights is kind of hard.

Software 2.0

Intelligent Software

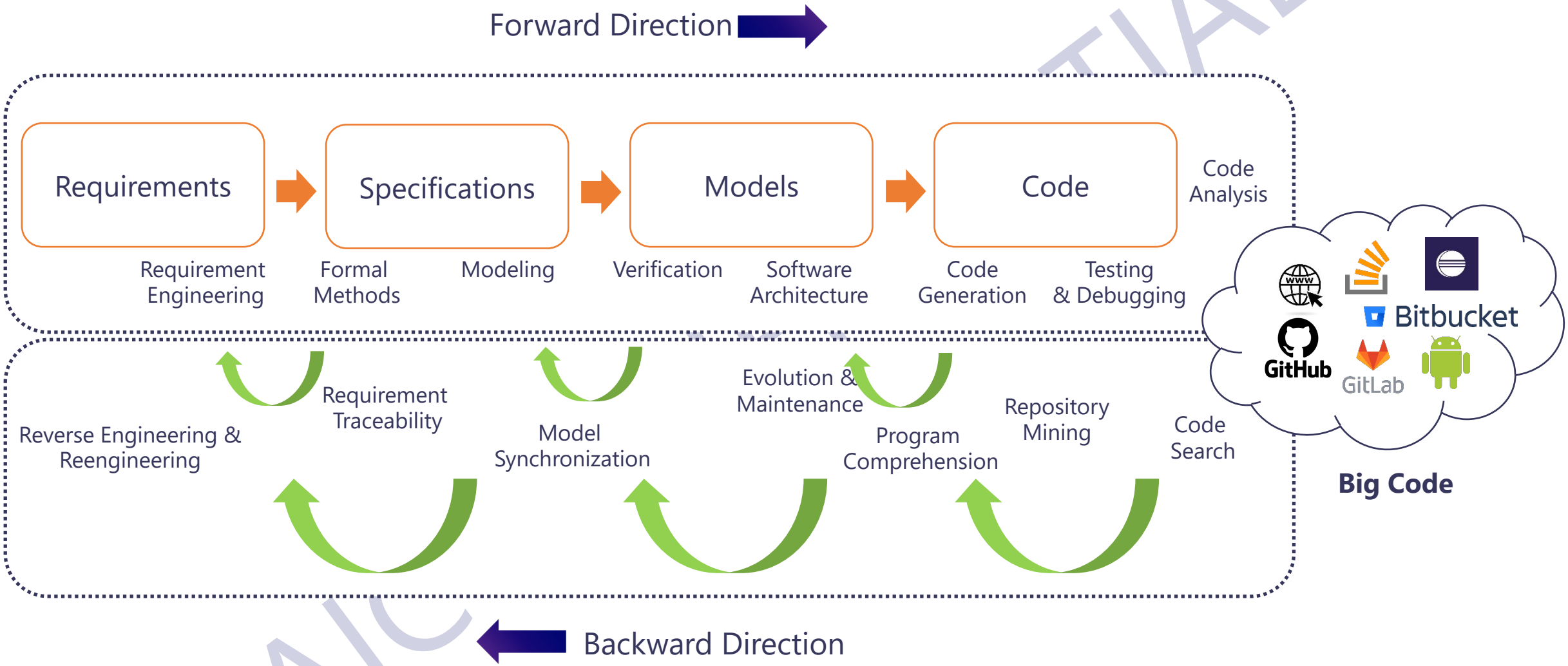
Machine Programming

AI4Code

Automated Programming

The Automation of Software Development

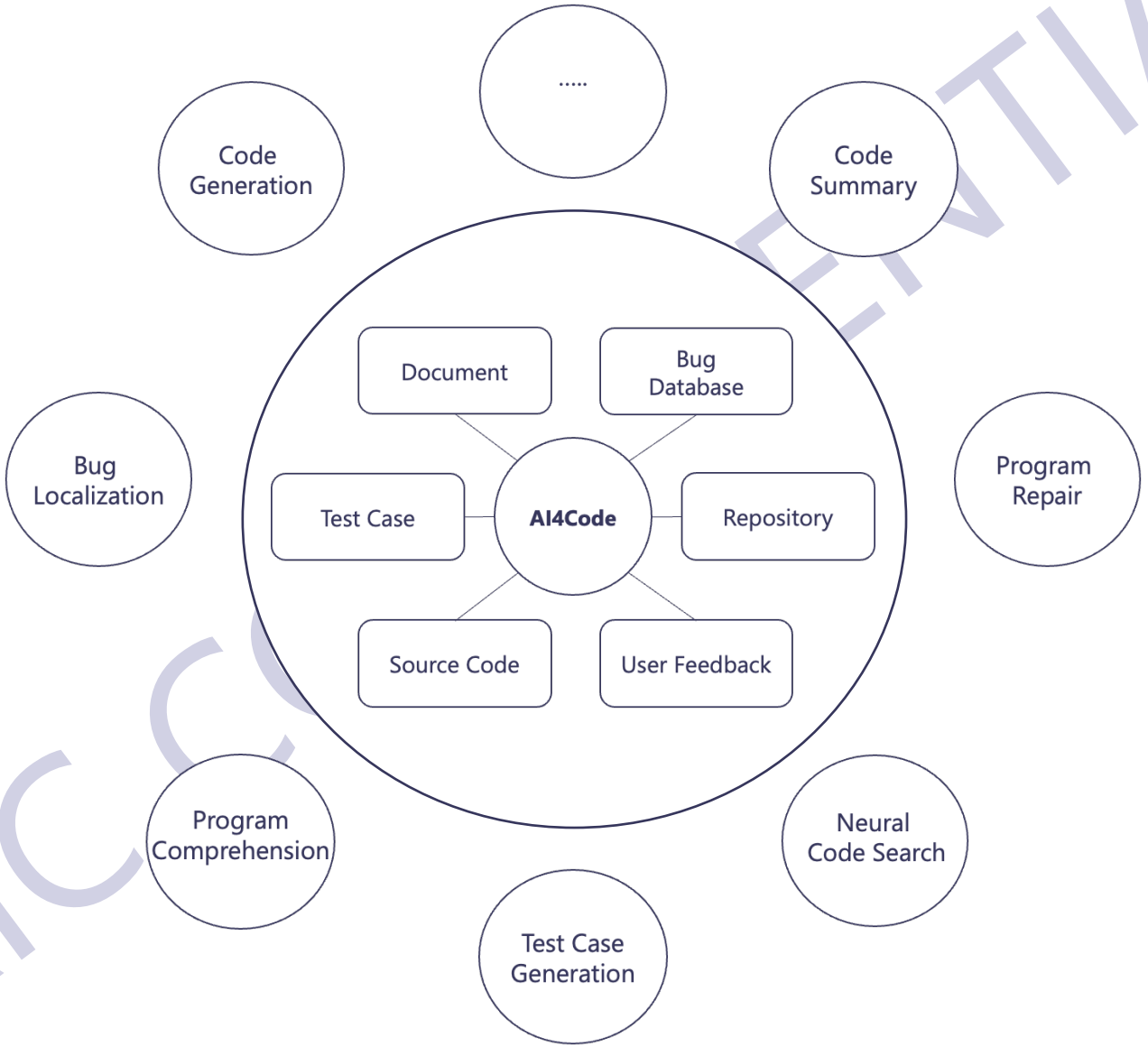
Areas in Software Development



Data Generated from Software Development Process

User/Business	Developer/Project	Code
Requirement Usage Scenario User Interface Feedback Bug Report Feature Request Private data Performance Integration Budget	Specification Documentation Repository Test case Bug database Email Discussion Priority Schedule	Lexes/Syntax Control/Data flow Call relation Data Dependencies Path Condition Thread Runtime ....

Building AI Models for Big Code

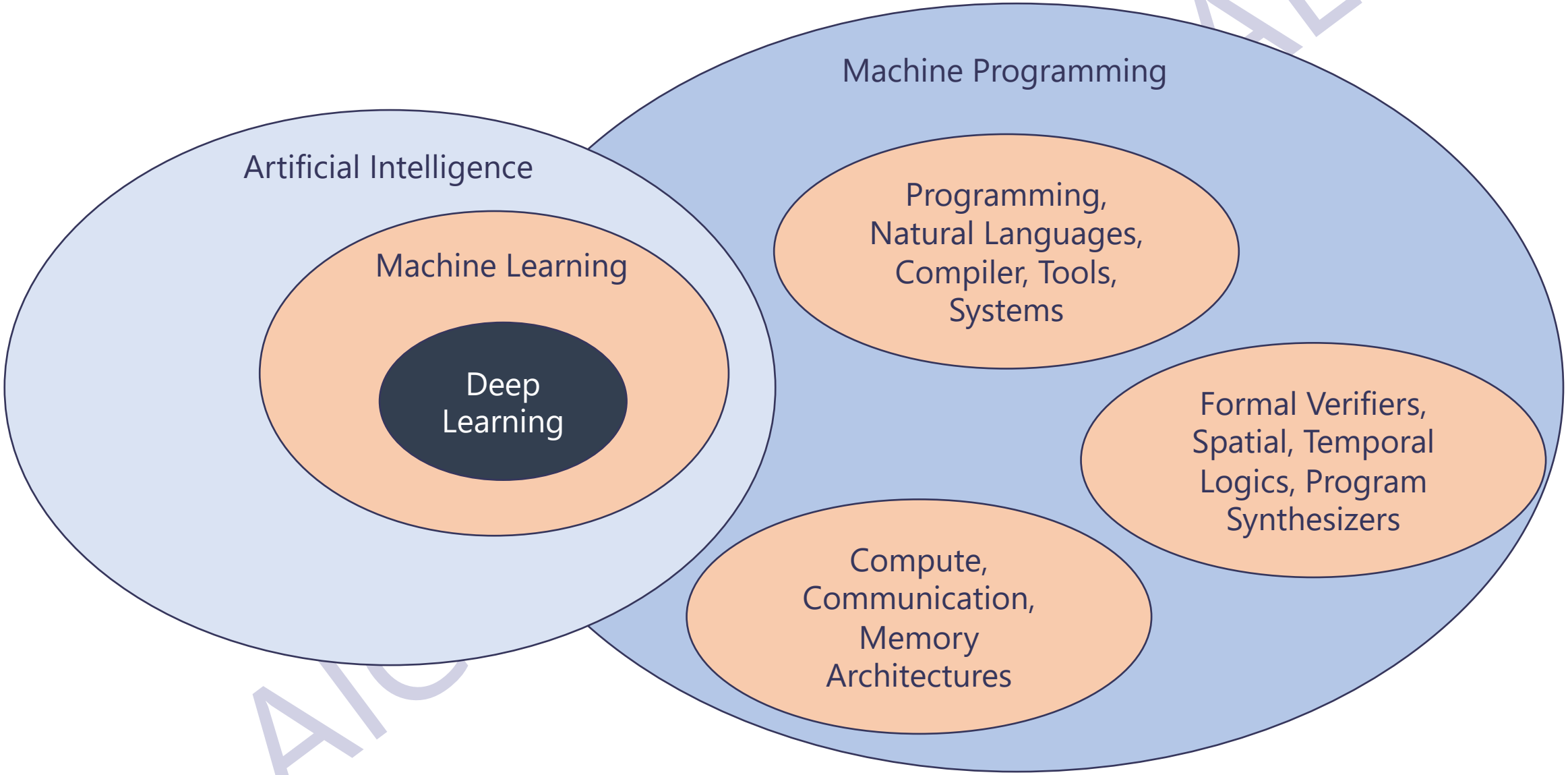


Machine Programming

✓ Humans communicate intention to machine.
✓ Machine handle the programming.
Thus, “Machine Programming”

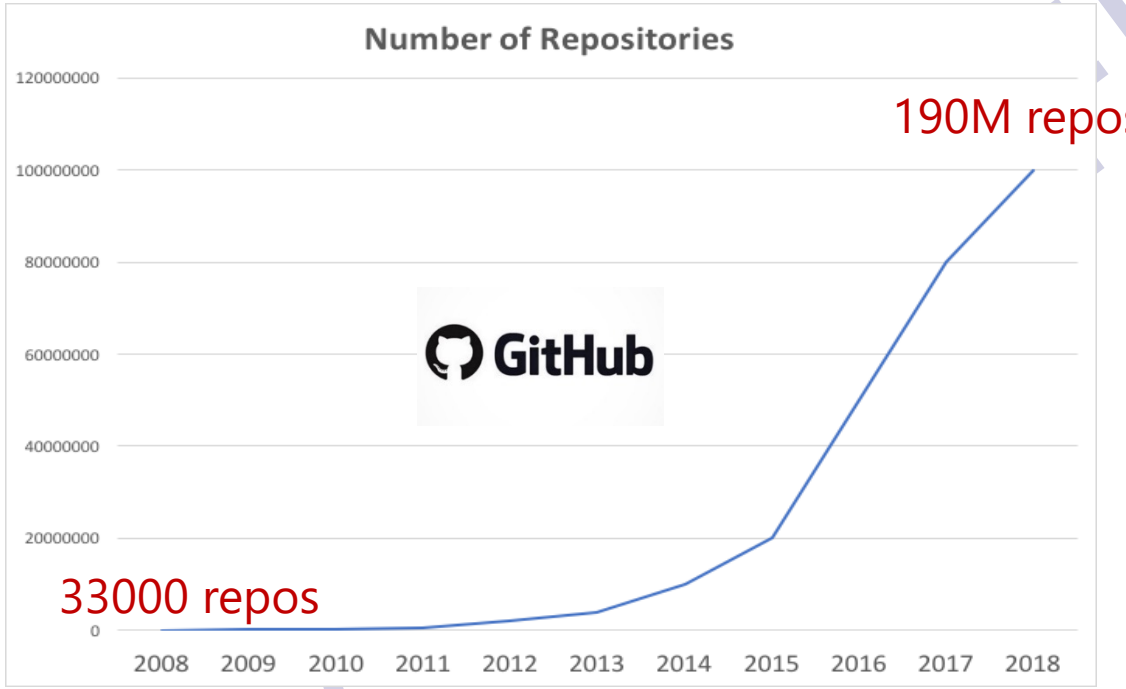
- Increasing the software’s **productivity.**
- Closing the **performance** gap between amateur developer and expert developer.
- Enhancing the **reliability** of software system.

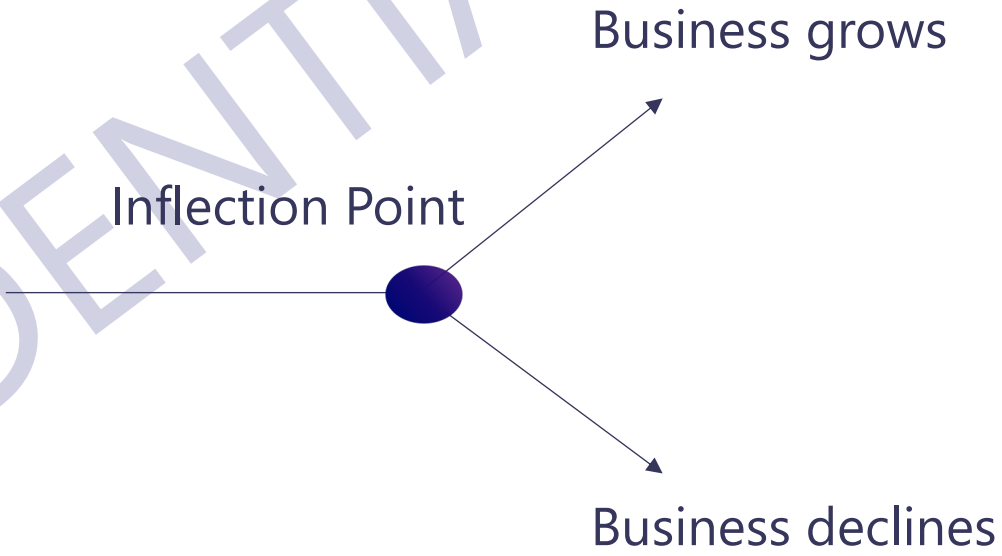
A View of Machine Programming (AI4Code)



The Inflection Point of Machine Programming

- ML and formal methods
  - Transformer, DeepCoder, BERT, NetSyn, …
- New & improved hardware
  - CPU, GPU, NNP, TPU, FPGAs, ….
- Large codebase dataset (BIG CODE !!)

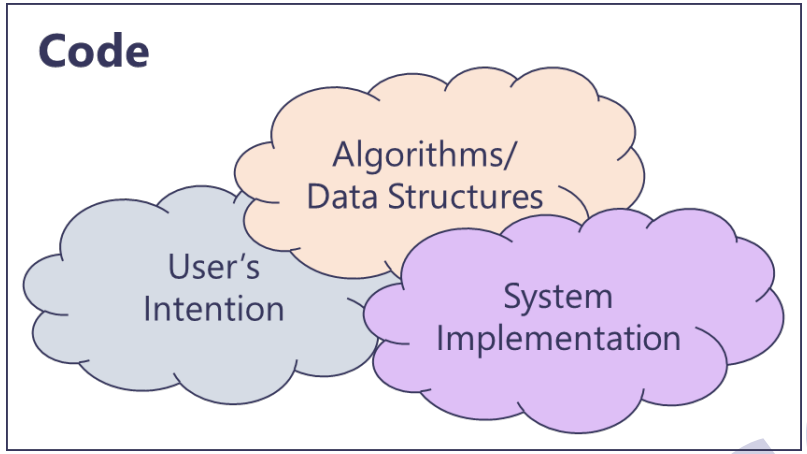




Machine Programming needs all of this to work

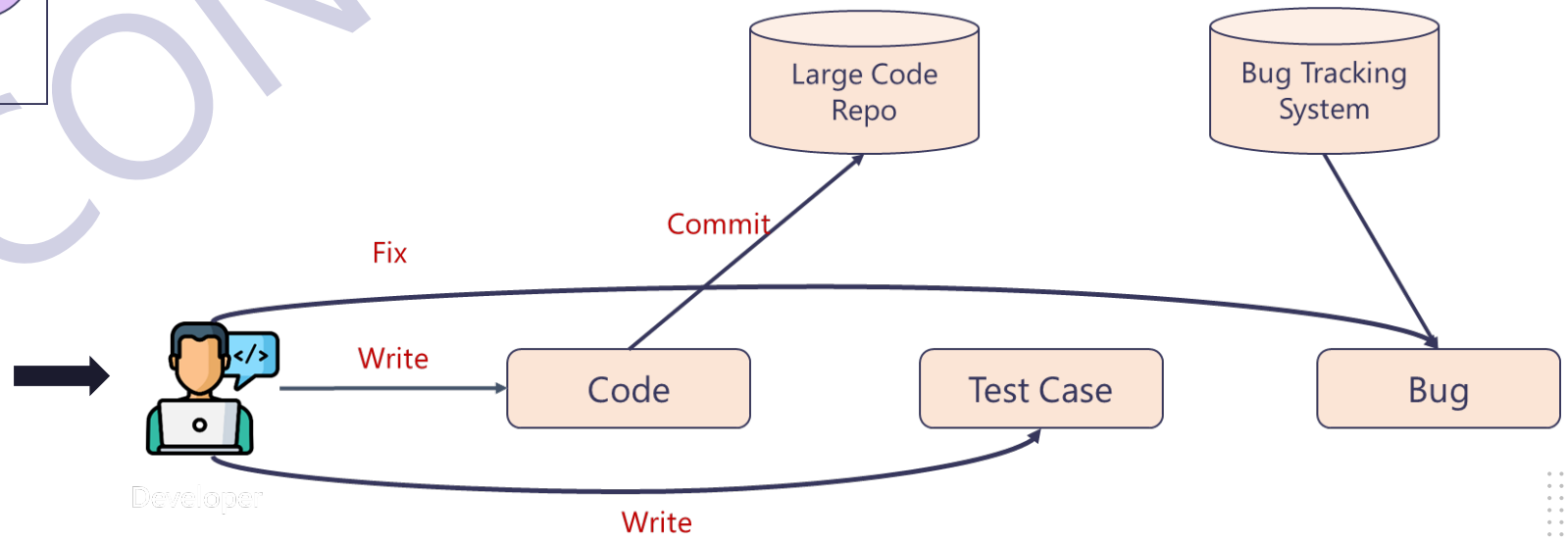
Today's Programming

- Programming is slow and error-prone.
- Programmers struggle to ensure correctness, security, and performance.
- Code constantly evolves to keep pace with an external environment.

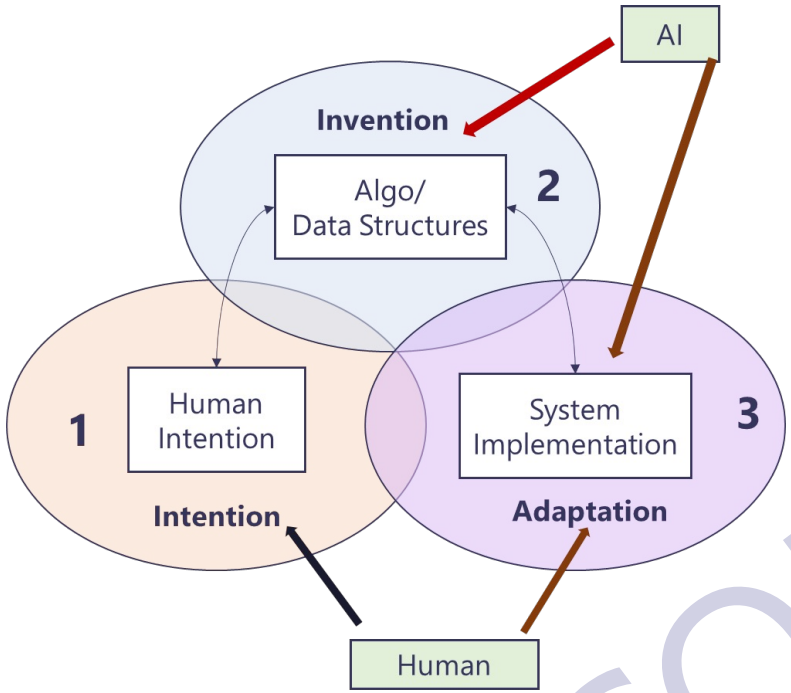


- A program usually contains:
  - The developer's intent
  - Code to express that intent.
  - Specifications for software/hardware adaptation.
- Human developers must handle all of these steps.

Basic work-flow of Today's Programming



Tomorrow's Programming

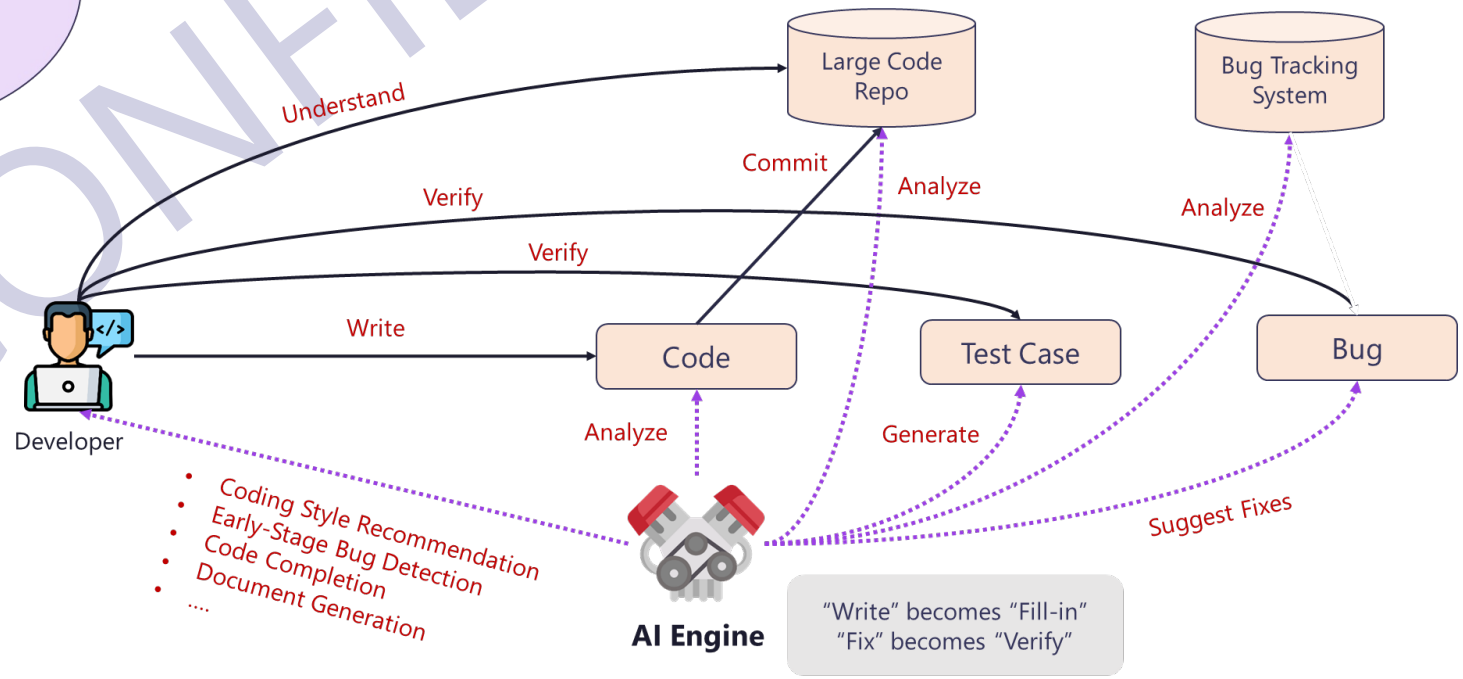


- Human only specifies intention:

"**Computer, gives me a program X that does Y**".

- Next, the system determines what it needs to construct that program.
- May have dialogue interface between human and machine.

Basic work-flow of Tomorrow's Programming



How is Machine Programming being implemented in the industry?

AI4Code Trending

Auto Completion	Boilerplate	Security scans	Code Reference
Optimize	Test Cases	Context Aware	Recommend APIs
Translation	Find Alternatives	Explain Code	Bias Detection
Architect/Design	Detect Anti-patterns	Refactoring	UI Development

Salesforce, AWS, Github (owned by Microsoft), and OpenAI regularly demonstrate the latest advancements in LLMs that are applied to software engineering tasks.

copilot.github.com

GitHub Copilot

Sign up >

Technical Preview

Your AI pair programmer

With GitHub Copilot, get suggestions for whole lines or entire functions right inside your editor.

Sign up >

sentiment.ts | write_sql.go | parse_expenses.py | addresses.rb

```
#!/usr/bin/env ts-node

import { fetch } from "fetch-h2";
```

Amazon CodeGuru

Automate code reviews and optimize application performance with ML-powered recommendations

Get started with Amazon CodeGuru

FEATURED

AWS Certification

Explore the resources available to help you prepare for your AWS Certification.

Find your most expensive lines of code and improve code quality

Jun 4, 2021, 10:25am EDT | 26,279 views

IBM CodeNet: Artificial Intelligence That Can Program Computers And Solve A $100 Billion Legacy Code Problem

Paul Smith-Goodson Contributor

Moor Insights and Strategy Contributor Group

Cloud

Analyst-in-residence, Quantum Computing

Intel Machine Programming Tool Detects Bugs in Code

Today, Intel unveiled ControlFlag – a machine programming research system that can autonomously detect errors in code.

News

- December 3, 2020
- Contact Intel PR

More New Technologies News →

» Watch video: "Intel Labs Day 2020: Justin Gottschlich Second Session from Intel PR on Vimeo"

What's New: Today, Intel unveiled ControlFlag – a machine programming research system that can autonomously detect errors in code. Even in its infancy, this novel, self-supervised system shows promise as a powerful productivity tool to assist software developers with the labor-intensive task of debugging. In preliminary tests, ControlFlag trained and learned novel defects on over 1 billion unlabeled lines of production-quality code.

intel newsroom | Top News Sections | News By Category | All News | Search News

News Byte

July 29, 2020

Contact Intel PR

Intel, MIT and Georgia Tech Deliver Improved Machine-Programming Code Similarity System

What's New: Today, Intel unveiled a new machine programming (MP) system – in conjunction with Massachusetts Institute of Technology (MIT) and Georgia Institute of Technology (Georgia Tech). The system, machine inferred code similarity (MISIM), is an automated engine designed to learn what a piece of software intends to do by studying the structure of the code and analyzing syntactic differences of other code with similar behavior.

ai.facebook.com/blog/aroma-ml-for-code-recommendation/

FACEBOOK AI | Research | Publications

ML APPLICATIONS | DEVELOPER TOOLS

Aroma: Using machine learning for code recommendation

April 04, 2019

AI gives software development tools a boost

GitHub Copilot, DeepDev, IntelliCode, and other code-focused applications of machine learning can help us deliver better code, faster.

POSTED ON NOVEMBER 6, 2018 TO DEVINFRA, ML APPLICATIONS

Getafix: How Facebook tools learn to fix bugs automatically

Microsoft | Visual Studio | Products | Downloads | Buy | Support

Visual Studio IntelliCode

AI-assisted development

Sign up for news and updates >

trusted-programming.github.io

Trustworthy Programming lab, Huawei Research Ireland

We are leading the smooth transition towards more trustworthy software engineering through innovative R&D in **programming theory**, e.g., lambda calculus-based functional programming, deep code learning-based intelligent software engineering; **programming technology and methods**, e.g., model-driven software development (MDSD), domain-specific languages (DSL), and **programming tools**, e.g., code transpiler, deep code learner, bug localizer, etc.

Blog posts

- Rustconf Reflections
- Explainer of Deep Code Learning
- Our Rust Mission at Huawei

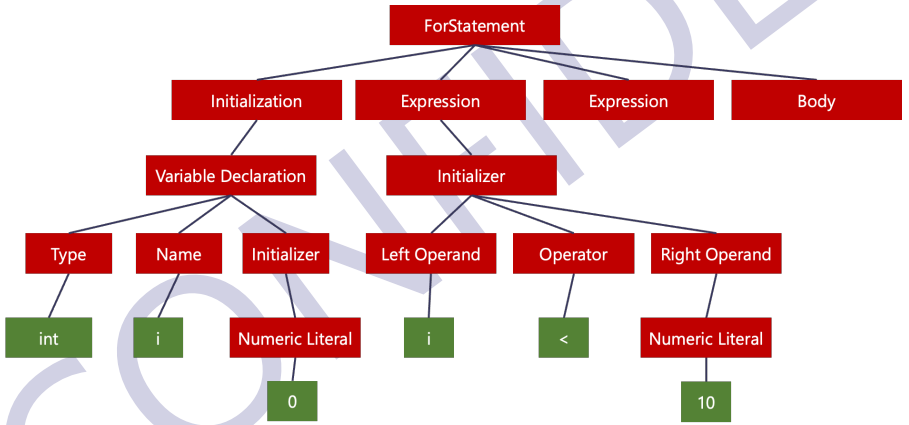
AI4Code Principles and Applications

Code Representations

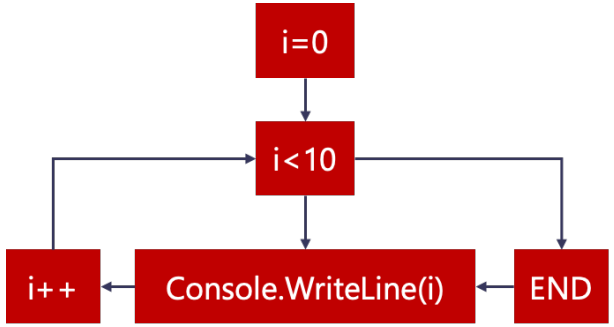
Token-based

```
for (int i = 0; i < 10; i++){
        Console.WriteLine(i);
}
```

Syntax-based (Tree)



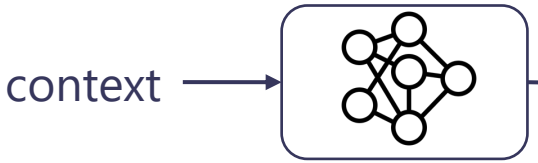
Graph-based



Probabilistic Models of Code

Code Generating Models

- Probability distribution over code, e.g. tokens or AST nodes.
- Model how the code is written

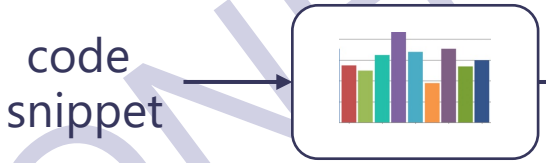


$$P_{\mathcal{D}}(\mathbb{c}|C(\mathbb{c}))$$

- Context C $(\mathbb{C})$
- Code representation $\mathbb{C}$

Representational Models of Code

- Yield conditional probability distribution over code element property given the code input
- Be able to predict the property

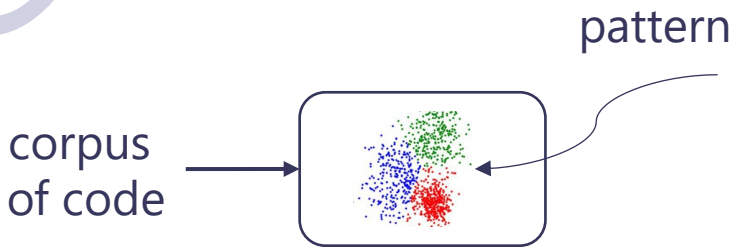


$$P_{\mathcal{D}}(\pi|f(\mathbb{c}))$$

- Code property π
- f to convert code snippet $\mathbb{C}$ into code representation

Pattern Mining of Code

- Infer, without supervision, a likely latent structure within code corpus.
- Find reusable and human-interpretable patterns.



$$P_{\mathcal{D}}(f(\mathbb{c})) = \sum_{\mathbf{l}} P_{\mathcal{D}}(g(\mathbb{c})|\mathbf{l})P(\mathbf{l})$$

- View of code $g(\mathbb{C})$
- l: set of latent variables to infer

Distributed Representation of Code

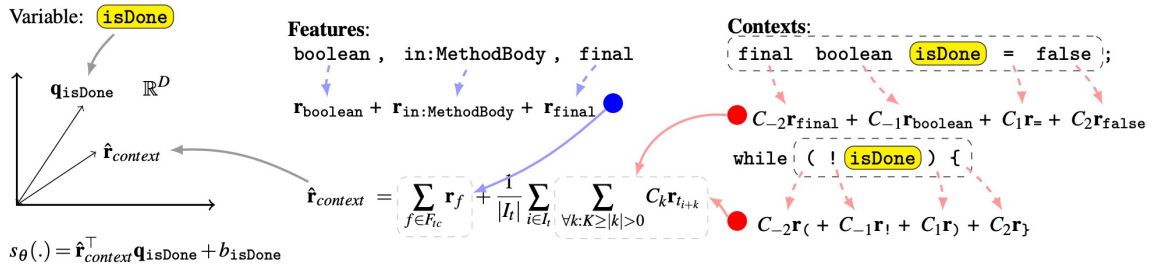
Suggesting Accurate Method and Class Names

Miltiadis Allamanis† Earl T. Barr‡ Christian Bird⋆ Charles Sutton†

†School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
{m.allamanis, csutton}@ed.ac.uk

‡Dept. of Computer Science
University College London
London, UK
e.barr@ucl.ac.uk

⋆Microsoft Research
Microsoft
Redmond, WA, USA
cbird@microsoft.com



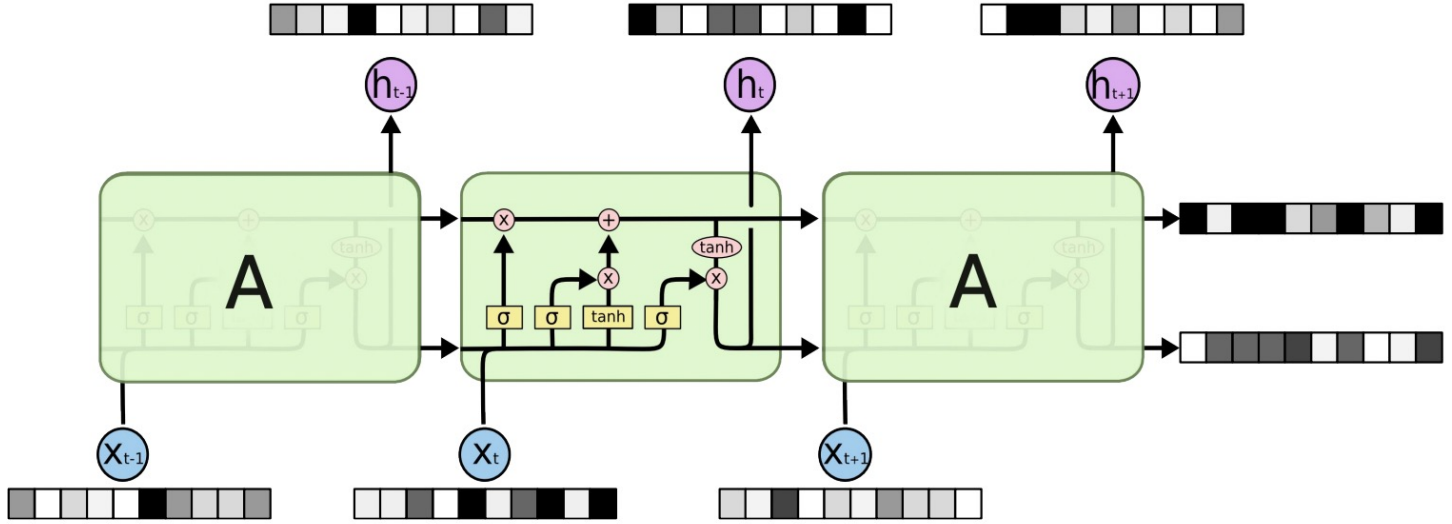
Log-Bilinear model to learn distributed representation of code

N-gram Language Models of Code

```
public void execute(                    task) {
        if (task == null)
            throw new NullPointerException();
        ForkJoinTask<?> job;
        if (task instanceof ForkJoinTask<?>) // avoid re-wrap
            job = (ForkJoinTask<?>) task;
        else
            job = new ForkJoinTask.AdaptedRunnableAction(task);
        externalPush(job);
    }
```

$$P(t_0 \dots t_M) = \prod_{m=0}^{M} P(t_m | t_{m-1} \dots t_{m-n+1})$$

Neural Language Models of Code

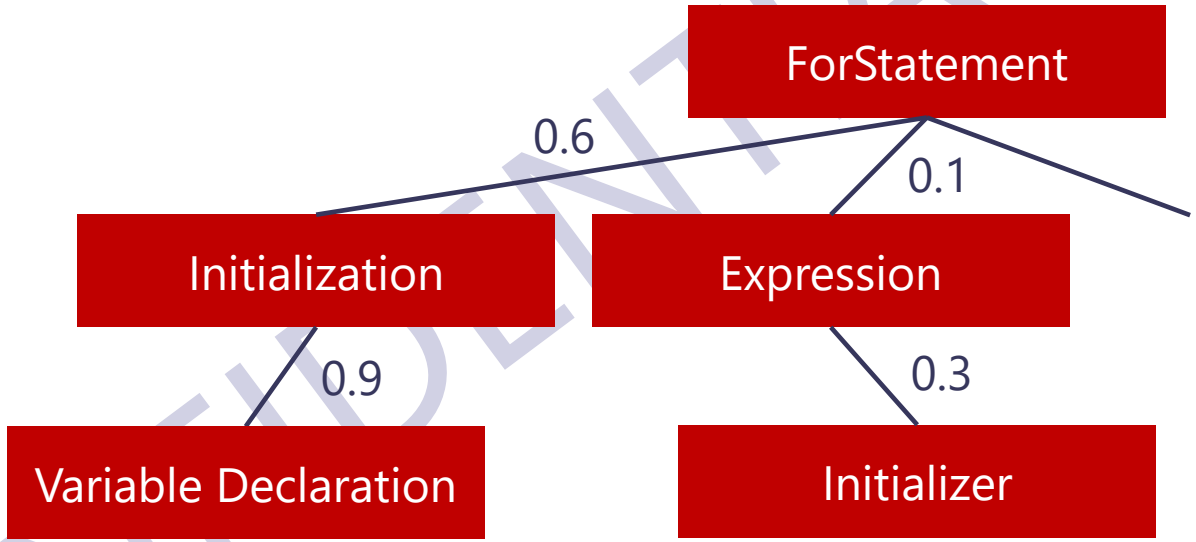



```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
  int i;
  if (classes[class]) {
    for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
      if (mask[i] & classes[class][i])
        return 0;
  }
  return 1;
}
```

Probabilistic Context Free Grammar for Code Generation

Assign Probability to Rules in the Context Free Grammar

- E -> E + E (prob=0.7)
- E ->T (prob=0.3)
- F -> (E) (prob=0.1)
- T -> F * F (prob=0.6)
- F -> F (prob=0.1)
- F -> id (prob=0.9)
- .....



Structured Generative Models of Natural Source Code

Chris J. Maddison[†] CMADDIS@CS.TORONTO.EDU
University of Toronto
Daniel Tarlow DTARLOW@MICROSOFT.COM
Microsoft Research

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

Structural Language Models of Code

Uri Alon [1] **Roy Sadaka** [1] **Omer Levy** [2 3] **Eran Yahav** [1]

[1]Technion, Israel [2]Tel Aviv University [3]Facebook AI Research. Correspondence to: Uri Alon <urialon@cs.technion.ac.il>, Roy Sadaka <roysadaka@gmail.com>, Omer Levy <omerlevy@gmail.com>, Eran Yahav <yahave@cs.technion.ac.il>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020.

Automated Bug Detection & Program Repair

Detect-Localize-Repair: A Unified Framework for Learning to Debug with CodeT5

Nghi D. Q. Bui, Yue Wang, Steven C.H. Hoi
Salesforce Research Asia
{nghi.bui,wang.y,shoi}@salesforce.com

Abstract

Automated software debugging is a crucial task for improving the productivity of software developers. Many neural-based techniques have been proven effective for debugging-related tasks such as bug localization and program repair (or bug fixing). However, these techniques often focus only on either one of them or approach them in a stage-wise

Zhu et al., 2021; Mashhadi and H
Ding et al., 2020; Wang et al., 202
naturalness hypothesis of softwa
et al., 2016). They adopt a generic
proach to train neural networks t
acquire bug-fix patterns through
massive corpora of previous bug-fi

However, these techniques suf

Deep Learning for Bug-Localization in Student Programs

Rahul Gupta[1] **Aditya Kanade**[1,2] **Shirish Shevade**[1]
[1]Department of Computer Science and Automation,
Indian Institute of Science, Bangalore, KA 560012, India
[2]Google Brain, CA, USA
{rahulg, kanade, shirish}@iisc.ac.in

Abstract

Providing feedback is an integral part of teaching. Most open online courses on programming make use of automated grading systems to support programming assignments and give real-time feedback. These systems usually rely on test

Practical Program Repair in the Era of Large Pre-trained Language Models

Chunqiu Steven Xia
University of Illinois at Urbana-Champaign
chunqiu2@illinois.edu

Yuxiang Wei
University of Illinois at Urbana-Champaign
ywei40@illinois.edu

Lingming Zhang
University of Illinois at Urbana-Champaign
lingming@illinois.edu

***Abstract*—Automated Program Repair (APR) aims to help velopers automatically patch software bugs. However, current te-of-the-art traditional and learning-based APR techniques e the problem of limited patch variety, failing to fix com-icated bugs. This is mainly due to the reliance on bug-fixing tasets to craft fix templates (traditional) or directly predict tential patches (learning-based). Large Pre-Trained Language**

Repair (APR) tools have been built to automatically generate potential patches given the original buggy program [6].

Among traditional APR techniques [7]–[16], template-based APR has been widely recognized as the state of the art [17], [18]. These techniques leverage fix templates, often designed by human experts, to fix specific types of bugs in the source

Less Training, More Repairing Please: Revisiting Automated Program Repair via Zero-shot Learning

Chunqiu Steven Xia
University of Illinois Urbana-Champaign
chunqiu2@illinois.edu

Lingming Zhang
University of Illinois Urbana-Champaign
lingming@illinois.edu

In *Proceedings of The 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2022)*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

ABSTRACT

Due to the promising future of Automated Program Repair (APR), researchers have proposed various APR techniques, including heuristic-based, template-based, and constraint-based techniques. Among such classic APR techniques, template-based techniques have been widely recognized as state of the art. However, such template-based techniques require predefined templates to perform repair, and their effectiveness is thus limited. To this end, researchers have leveraged the recent advances in Deep Learning to further improve APR. Such learning-based techniques typically view APR as a Neural

1 INTRODUCTION

Software systems are all-pervasive in everyday life from monitoring financial transactions [69], controlling transportation systems [73] to aiding healthcare tools [8]. Software bugs in these systems can af-

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 25, 2021, accepted January 9, 2022, date of publication January 18, 2022, date of current version January 28, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3144079

A Survey of Challenges in Spectrum-Based Software Fault Localization

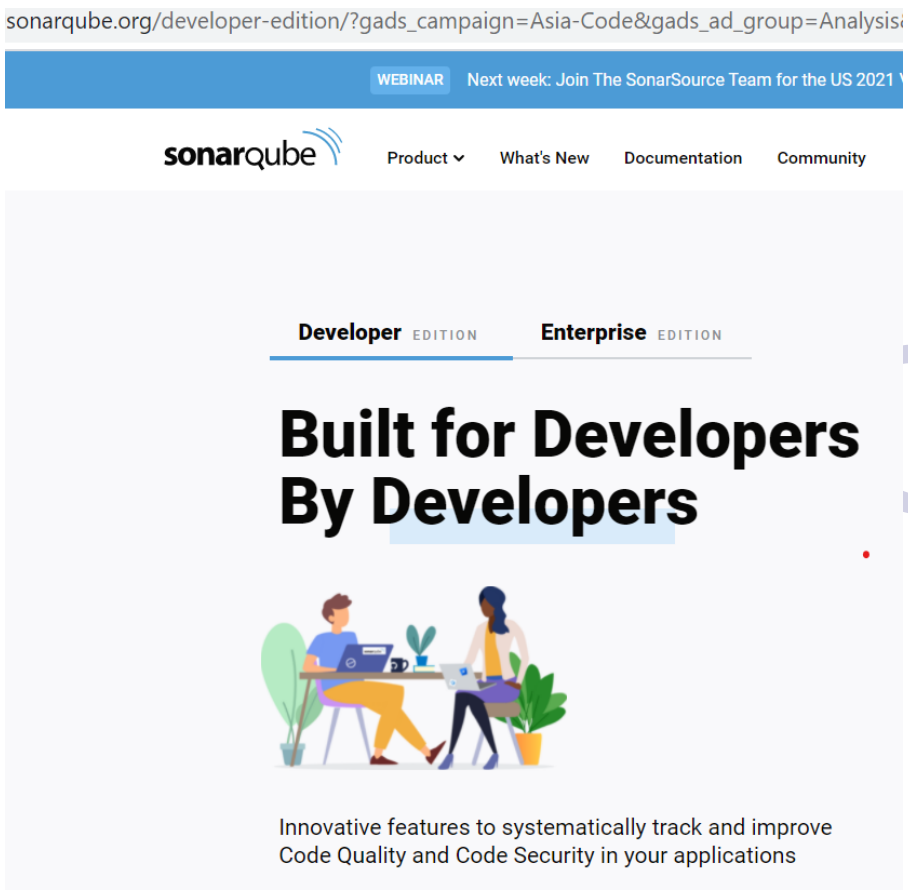
QUSAY IDREES SARHAN[1,2] **AND ÁRPÁD BESZÉDES**[1]
[1]Department of Software Engineering, University of Szeged, 6720 Szeged, Hungary
[2]Department of Computer Science, University of Duhok, Duhok, Kurdistan Region 42001, Iraq

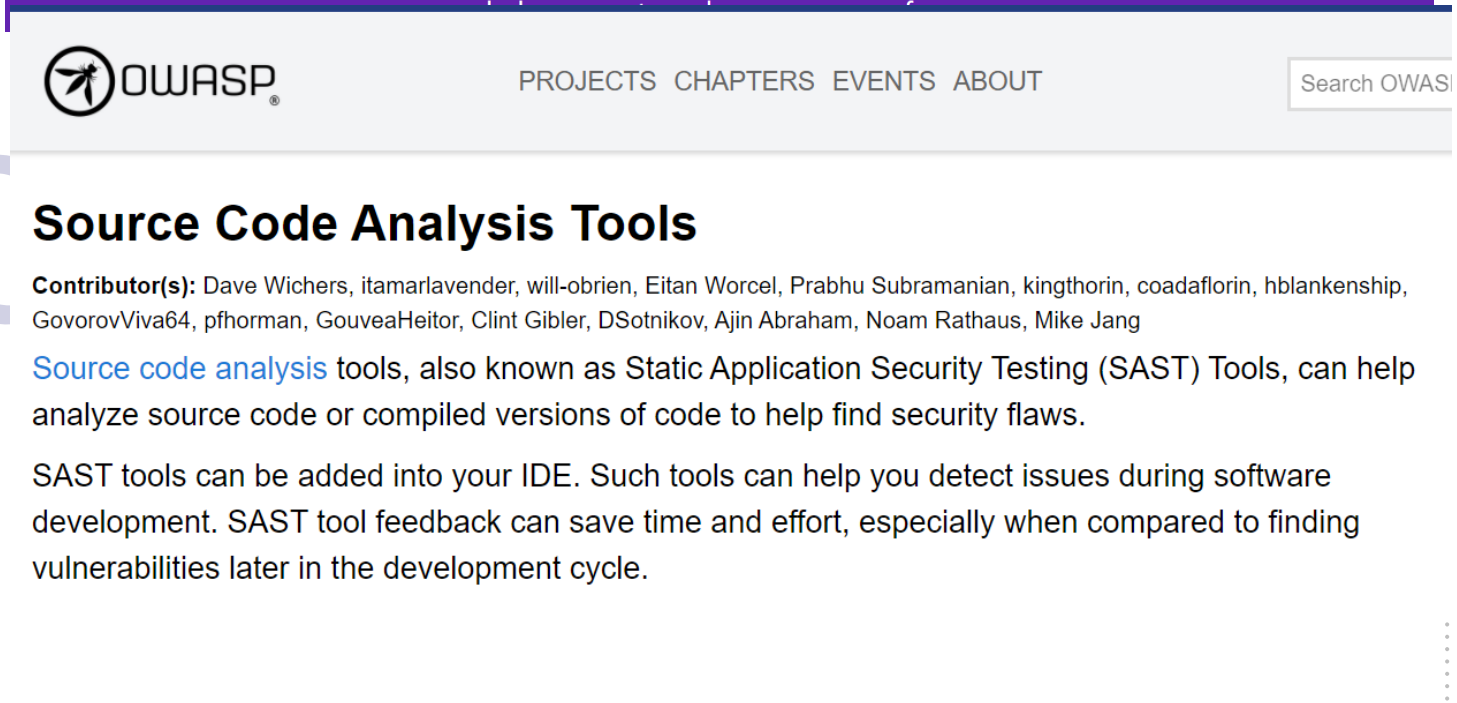
Corresponding author: Qusay Idrees Sarhan (sarhan@inf.u-szeged.hu)

This work was supported in part by the University of Szeged Open Access Fund under Grant 5459; in part by the Ministry of Innovation and Technology, NRDI Office, Hungary within the framework of the Artificial Intelligence National Laboratory Program (MILAB); and in part by the Hungarian National Research, Development and Innovation Office, through the "Security Enhancing Technologies for the IoT," under Grant 2018-1.2.1-NKP-2018-00004. The work of Qusay Idrees Sarhan was supported by the Stipendium Hungaricum Scholarship Program.

Learning to Fix Bug

Bug detection tools are dominated by Program Analysis





Learning to Fix Bug (cont.)

Some semantic bugs cannot be fixed by program analysis

Missing "copy"

```
module.exports = function (grunt) {
  grunt.initConfig({
    execute: {...}, copy: {...}, checktextdomain: {...}
    wp_readme_to_markdown: {...}, makepot: {...}})
  ...
  grunt.registerTask('default',['wp_readme_to_markdown'
    ,'makepot','execute','checktextdomain'])
};
```

innerHtml

```
function clearEmployeeListOnLinkClick(){
  document.querySelector("a").addEventListener("click",
    function(event){
      document.querySelector("ul").InnerHTML = "";
    }
  );
}
```

Semantic Bugs in Javascript

Method 1.

```
1     public List<String> getSkewedColumnNames(String alias) {
2         ...
3         else {
4             //...
5             TypeInfo typeInfo = TypeInfoUtils.getTypeInfoFromObjectInspector(this.metaData.
6                 getTableForAlias(tabAlias).getDeserializer().getObjectInspector());
7             desc = new exprNodeConstantDesc(typeInfo.getStructFieldTypeInfo(colName), null);
8         }
9         ...
10    }
```

Method 2.

```
1     public TypeInfo getStructFieldTypeInfo(String field) {
2         String fieldLowerCase = field.toLowerCase();
3         for(int i=0; i<allStructFieldNames.size(); i++) {
4             if (field.equals(allStructFieldNames.get(i))) {
5                 return allStructFieldTypeInfos.get(i);
6             }
7         }
8         throw new RuntimeException("cannot find field " + field + "(lowercase form: " +
9         fieldLowerCase + ") in " + allStructFieldNames);
10    }
```

Method 3.

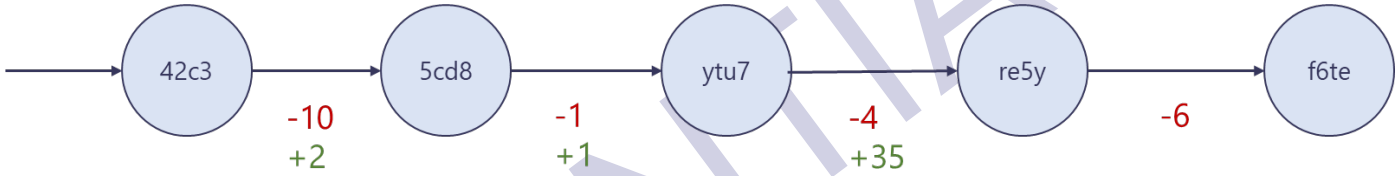
```
1     public Table getTableForAlias(String alias) {
2 -       return this.aliasToTable.get(alias);
3 +       return this.aliasToTable.get(alias.toLowerCase());
4     }
```

Convert "alias" to lower case

Semantic Bugs in Java

Learning to Fix Bug from Commit History

AI can help to fix semantic bugs by learning from commit history



guava/src/com/google/common/primitives/Unsig

```
@@ -195,12 +195,12 @@ public long longValue() {
195 195     */
196 196    @Override
197 197    public float floatValue() {
198       -    @SuppressWarnings("cast")
199       -    float fValue = (float) (value & UNSIGNED_M
200       -    if (value < 0) {
201       -      fValue += 0x1.0p63f;
      198 +    if (value >= 0) {
      199 +      return (float) value;
202 200      }
203       -    return fValue;
      201 +    // The top bit is set, which means that th
      202 +    // So we can ignore the bottom 8, except f
      203 +    return (float) ((value >>> 1) | (value & 1
204 204    }
```

mysql / mysql-server Public

Watch 549

Code | Pull requests 2 | Actions | Projects | Security | Insights

5.5

Commits on Aug 20, 2018

Bug#27788907 SOME FILE OPERATIONS IN MF_IOCACHE2.C ARE NOT INSTRUMENTED — marcalff committed on Aug 20, 2018 — bac287c

Commits on Aug 19, 2018

Bug #26791931: INCORRECT BEHAVIOR IN ALTER TABLE REORGANIZE — Sreeharsha Ramanavarapu committed on Aug 19, 2018 — a653fca

Commits on Aug 3, 2018

BUG#28144933 - MYSQL-SERVER RPM DOES NOT INSTALL PERL-DATA-DUMPER AS... — bkandasa committed on Aug 3, 2018 — fe5df42

Commits on Jul 27, 2018

Merge branch 'mysql-5.5.61-release' into mysql-5.5 — hramilison committed on Jul 27, 2018 — 56b5d42

The programm was first pilote on October 23 in Orava and Bardejov two regions with the highest number of Covid-19 cases in the country. Nearly 141.000 people, 91% of those who were eligible, got tested in the two regions over the three days of the pilot. In the rest of the countitry the testing took place simultaneously on Saturday and S encouraged everyone older than 10 to take part in th ple older than 65 years who spend most of their ti disabilitiies cancer patients, immuno compromised ble groups were exempt. Europe suffers record case numbers as France& Macron warns

country

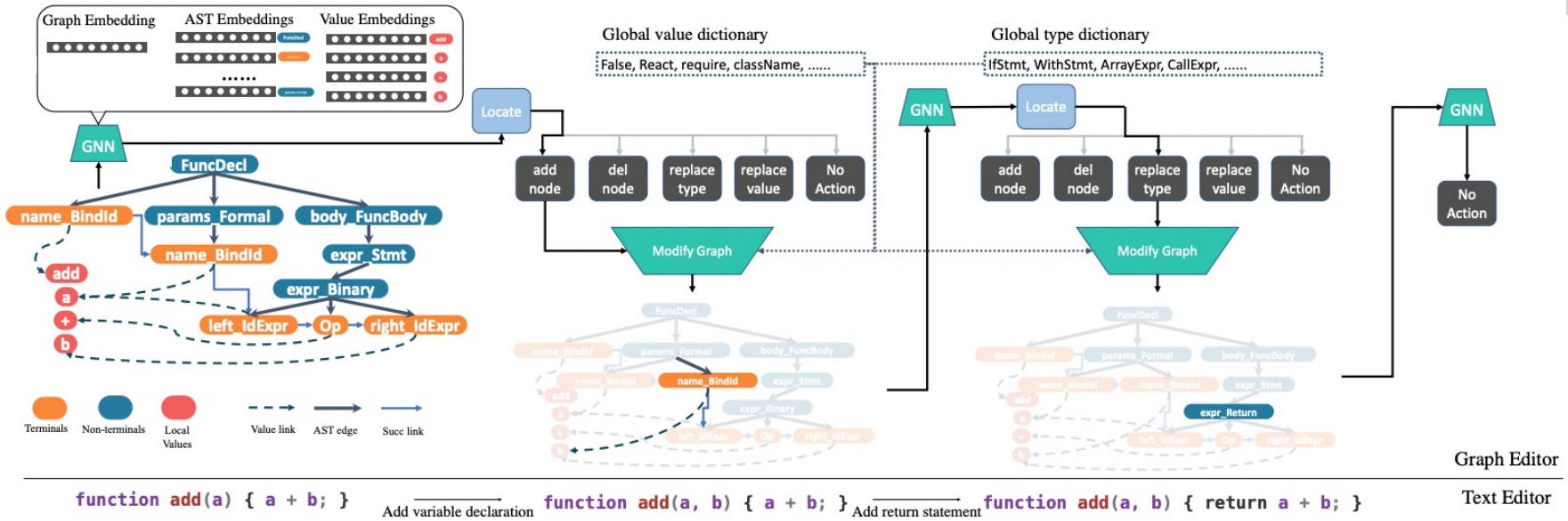
Ignore In this text

Imagine this is similar to using AI for grammar error correction in Natural Language Processing

```
@@ -105,8 +105,8 @@
105 105
106 106      private transient BiEntry<K, V>[] hashTableKToV;
107 107      private transient BiEntry<K, V>[] hashTableVToK;
108       -    private transient @Nullable BiEntry<K, V> firstInKeyInsertionOrder;
109       -    private transient @Nullable BiEntry<K, V> lastInKeyInsertionOrder;
      108 +    @Weak private transient @Nullable BiEntry<K, V> firstInKeyInsertionOrde
      109 +    @Weak private transient @Nullable BiEntry<K, V> lastInKeyInsertionOrder
110 110      private transient int size;
111 111      private transient int mask;
112 112      private transient int modCount;
```

```
1006 1006      }
1007 1007
1008 1008      // Guarded By Segment.this
1009       -    ReferenceEntry<K, V> previousAccess = nullEntry();
     1009 +    @Weak ReferenceEntry<K, V> previousAccess = nullEntry();
1010 1010
1011 1011      @Override
1012 1012      public ReferenceEntry<K, V> getPreviousInAccessQueue() {
```

Learning to Fix Bug from Commit History (cont.)



Perform steps of program transformation to convert buggy program to non-buggy program

$$p(g_{fix}|g_{bug};\theta) = p(g_1|g_{bug};\theta)p(g_2|g_1;\theta)\ldots p(g_{fix}|g_{T-1};\theta)$$

Published as a conference paper at ICLR 2020

HOPPITY: LEARNING GRAPH TRANSFORMATIONS TO DETECT AND FIX BUGS IN PROGRAMS

Elizabeth Dinella* University of Pennsylvania
Hanjun Dai* Google Brain
Ziyang Li University of Pennsylvania
Mayur Naik University of Pennsylvania
Le Song Georgia Tech
Ke Wang Visa Research

ABSTRACT

We present a learning-based approach to detect and fix a broad range of bugs in Javascript programs. We frame the problem in terms of learning a sequence of graph transformations: given a buggy program modeled by a graph structure, our model makes a sequence of predictions including the position of bug nodes and

Getafix: Learning to Fix Bugs Automatically

Johannes Bader
Facebook
jobader@fb.com

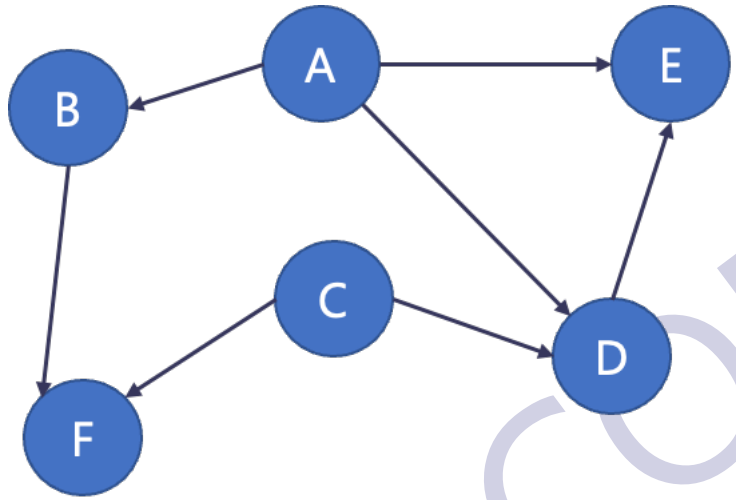
Andrew Scott
Facebook
andrewscott@fb.com

Michael Pradel
Facebook
michael@binaervarianz.de

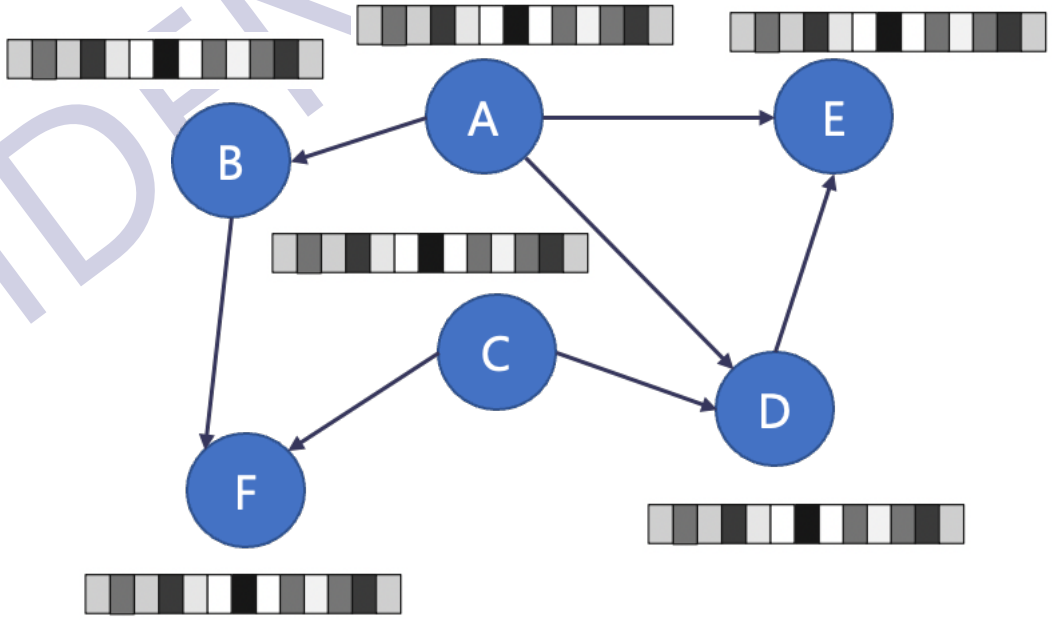
Satish Chandra
Facebook
satch@fb.com

Graph Neural Network for Code Understanding

Graph Notation: **G = (V, E)**
- Nodes/Vertices **V**
- Edges/Links **E**

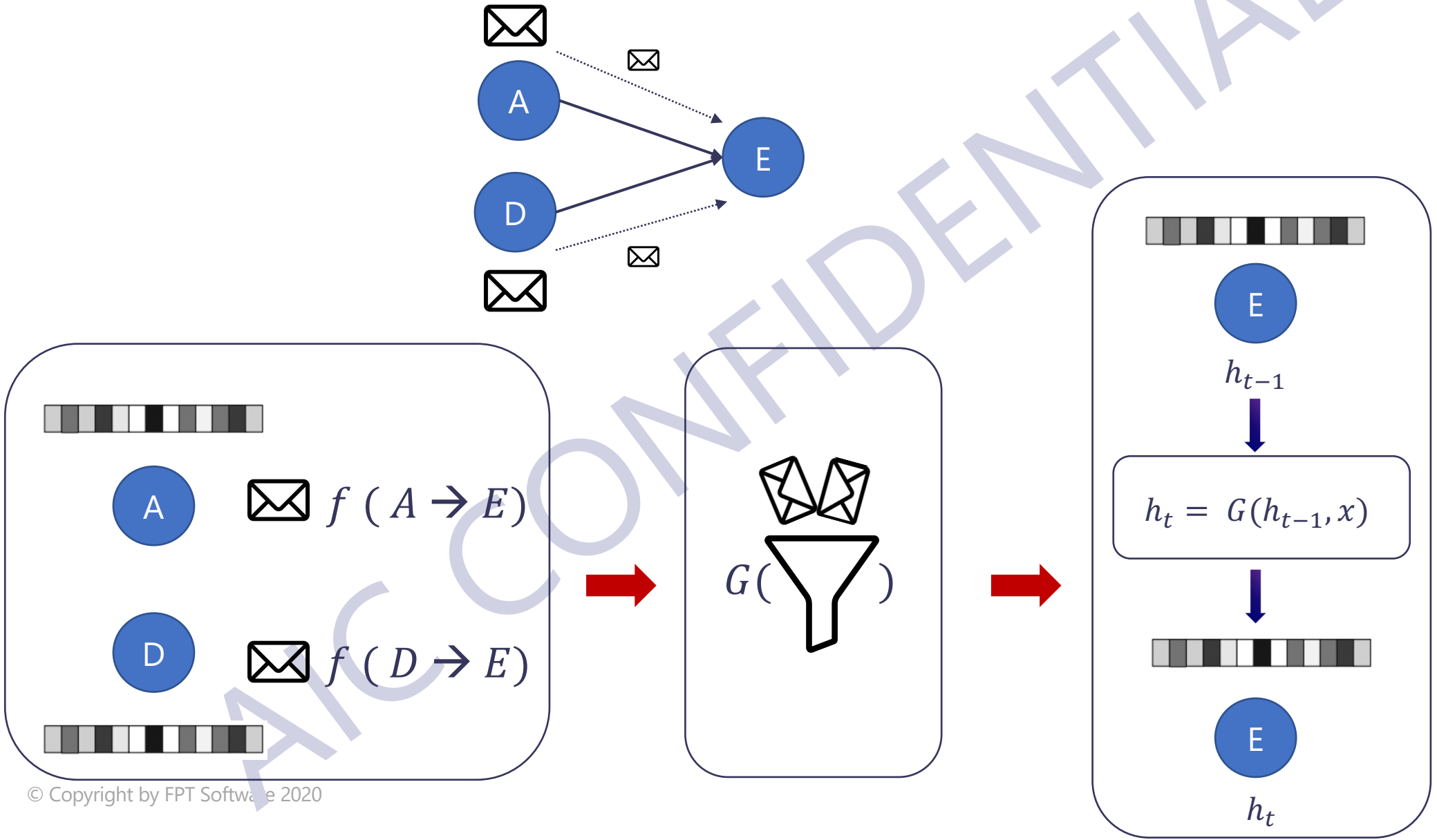


Graph Representation

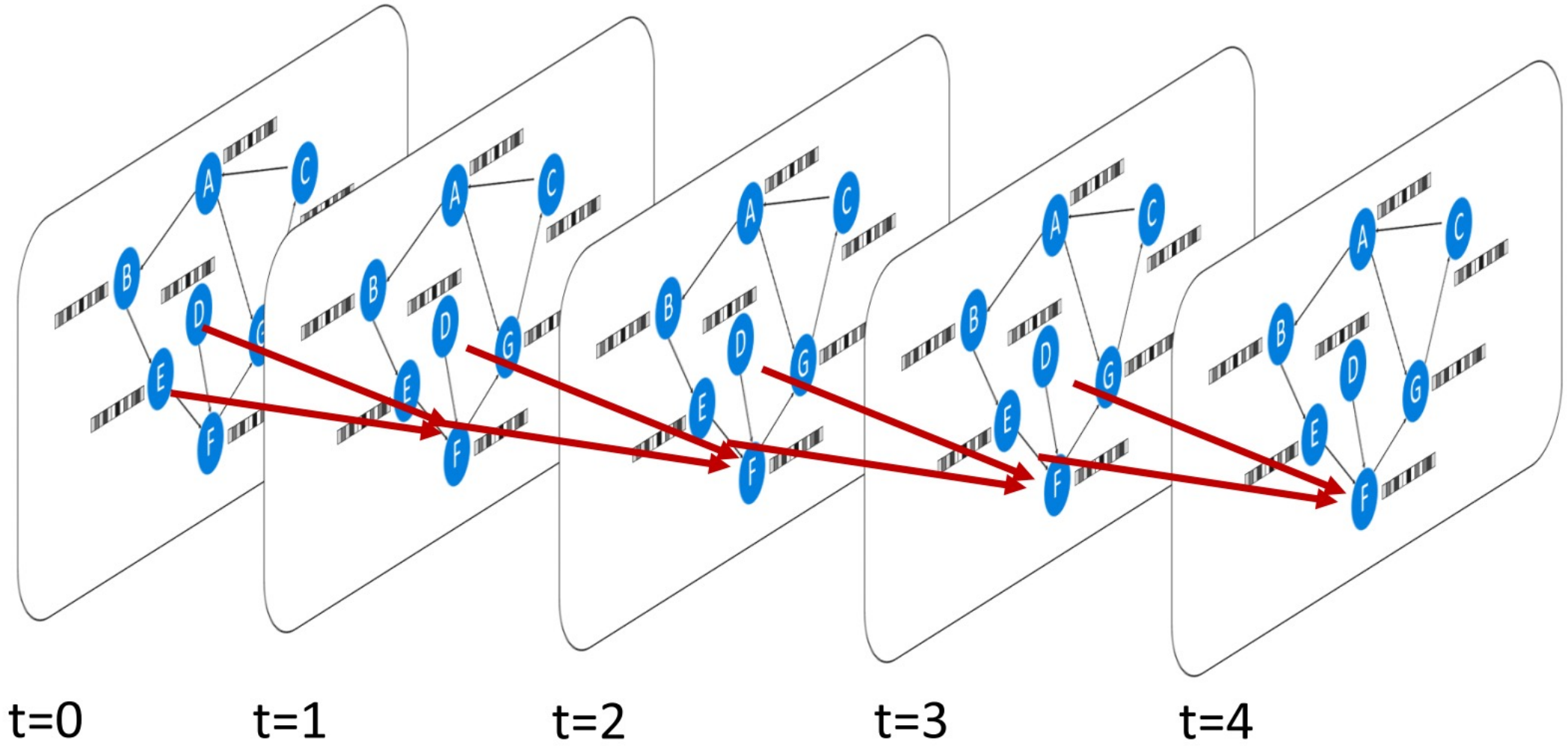


Initial Representation of each ode

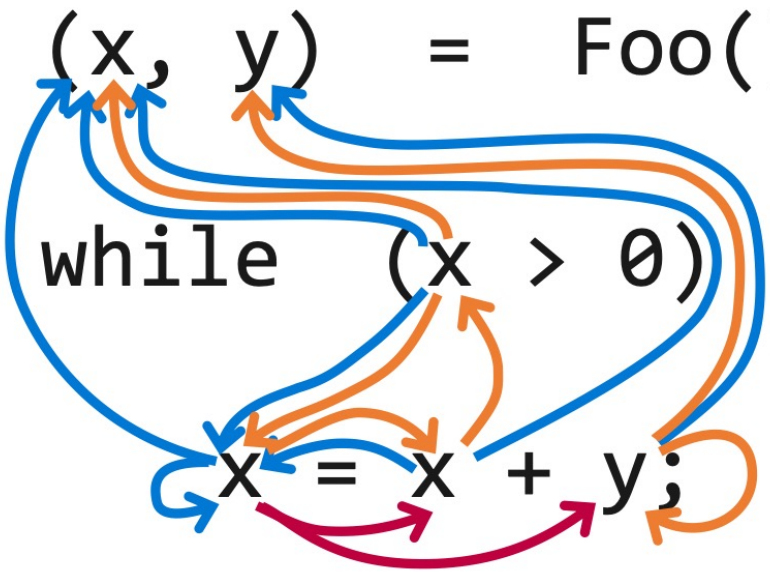
Graph Neural Network (cont.): Message Passing



Graph Neural Network (cont.): Message Passing



Represent a Program as Graph with Data Flow



Last Write

Last Use

Computed From

Graph Neural Network to Fix Variable Mis-used Bugs

```
var clazz=classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first=classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull( clazz );

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Real variable mis-used bug found in RavenDB

How to know the variable “clazz” is mis-used?

Possible type-correct options: clazz, first

Graph Neural Network to Fix Variable Mis-used Bugs

```
var clazz=classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first=classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(      );

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

```
var clazz=classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first=classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull( SLOT );

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

first

clazz

Fill-in-the-blank task on program graph

Goal: make the representation of SLOT as close as possible to the representation of the correct candidate node

$$f\left(\boldsymbol{h}_T^{SLOT}, \boldsymbol{h}_T^{first}\right) \gg f\left(\boldsymbol{h}_T^{SLOT}, \boldsymbol{h}_T^{clazz}\right)$$

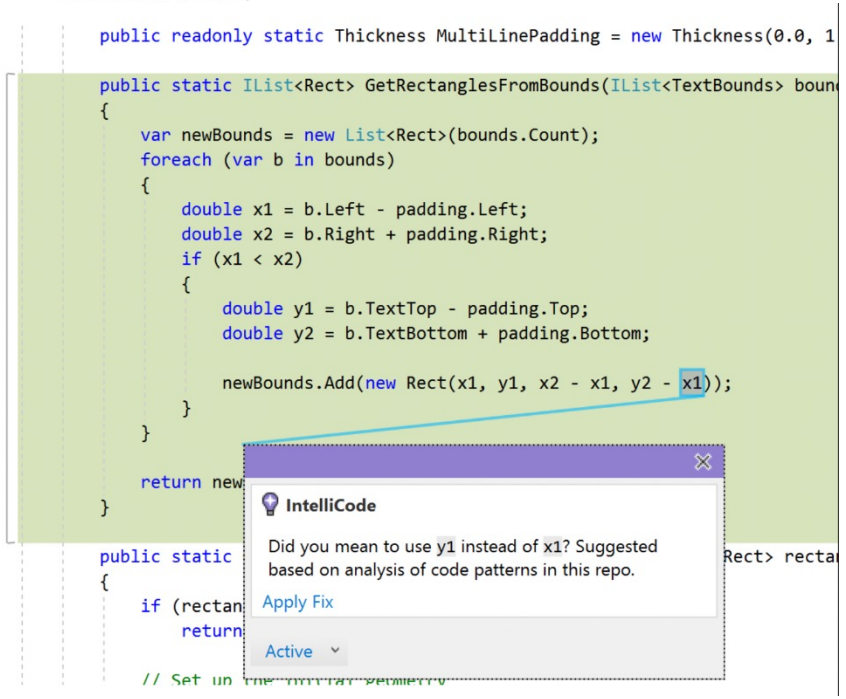
Published as a conference paper at ICLR 2018

LEARNING TO REPRESENT PROGRAMS WITH GRAPHS

Miltiadis Allamanis
Microsoft Research
Cambridge, UK
miallama@microsoft.com

Marc Brockschmidt
Microsoft Research
Cambridge, UK
mabrocks@microsoft.com

Mahmoud Khademi*
Simon Fraser University



IntelliCode (Microsoft) can recommend mis-used variable using AI

Refine Variable's Name with Probabilistic Graphical Model

Probabilistic Model for Code with Decision Trees

Veselin Raychev
Department of Computer Science
ETH Zürich, Switzerland
veselin.raychev@inf.ethz.ch

Pavol Bielik
Department of Computer Science
ETH Zürich, Switzerland
pavol.bielik@inf.ethz.ch

Martin Vechev
Department of Computer Science
ETH Zürich, Switzerland
martin.vechev@inf.ethz.ch

Predicting Program Properties from "Big Code"

Veselin Raychev
Department of Computer Science
ETH Zürich
veselin.raychev@inf.ethz.ch

Martin Vechev
Department of Computer Science
ETH Zürich
martin.vechev@inf.ethz.ch

Andreas Krause
Department of Computer Science
ETH Zürich
andreas.krause@inf.ethz.ch

Rename meaningless variable name to meaningful variable name using statistical modeling

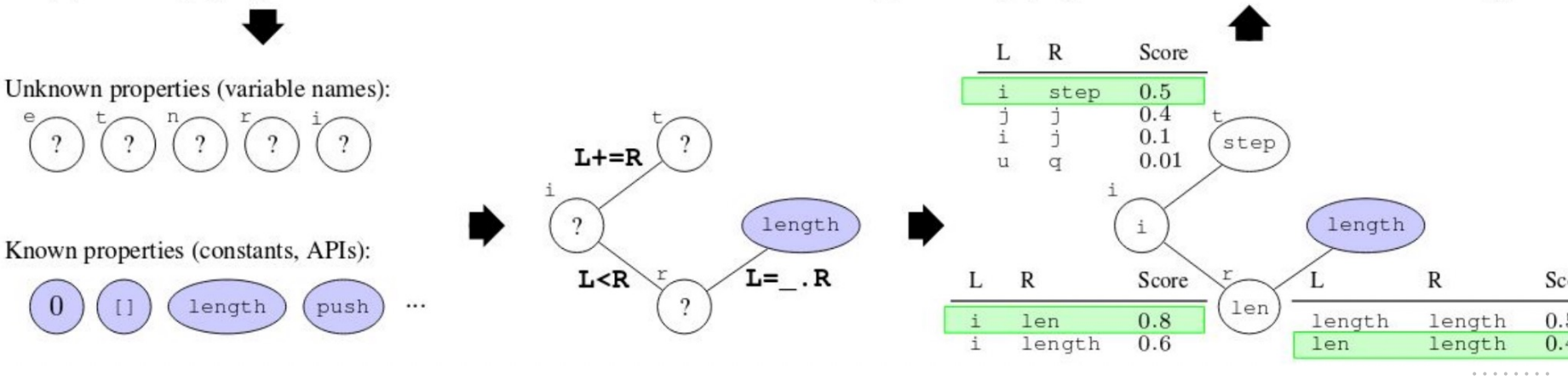
```
function chunkData(e, t) {
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t) {
    if (i + t < r) {
      n.push(e.substring(i, i + t));
    } else {
      n.push(e.substring(i, r));
    }
  }
  return n;
}
```

(a) JavaScript program with minified identifier names

e => str
t => step
n => colNames
r => len

```
/* str: string, step: number, return: Array */
function chunkData(str, step) {
  var colNames = []; /* colNames: Array */
  var len = str.length;
  var i = 0; /* i: number */
  for (; i < len; i += step) {
    if (i + step < len) {
      colNames.push(str.substring(i, i + step));
    } else {
      colNames.push(str.substring(i, len));
    }
  }
  return colNames;
}
```

(e) JavaScript program with new identifier names and types



Code Summarization

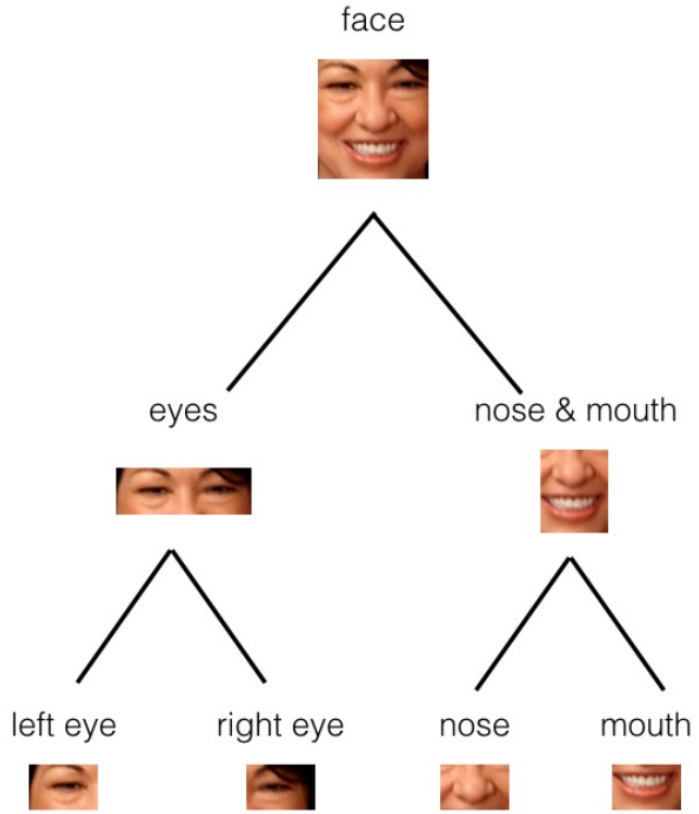
```
/*
A function to [redacted]
*/
void [redacted]()
  TreeView myTreeView = new TreeView()
  myTreeView.Nodes.Clear();
  foreach (string parentText in xml.parent)
  {
    TreeNode parent = new TreeNode();
    parent.Text = parentText;
    myTreeView.Nodes.Add(treeNodeDivisions);
    foreach (string childText in xml.child)
    {
      TreeNode child = new TreeNode();
      child.Text = childText;
      parent.Nodes.Add(child);
    }
  }
}
```

Generate Docstring: **add a child node to a TreeView**

Predict Method's Name: **addChildNode**

Code Summarization includes:
- **Method name prediction**
- **Comment generation**
- **Docstring generation**

Learning to Summarize Code with Capsule Theory



Dynamic Routing Between Capsules

Sara Sabour **Nicholas Frosst**

Geoffrey E. Hinton
Google Brain
Toronto
{sasabour, frosst, geoffhinton}@google.com

- Combine observations from smaller parts to build up a more complex part of a picture.
- Different parts of the pictures have dependency on each other

Learning to Summarize Code with Capsule Theory (cont.)

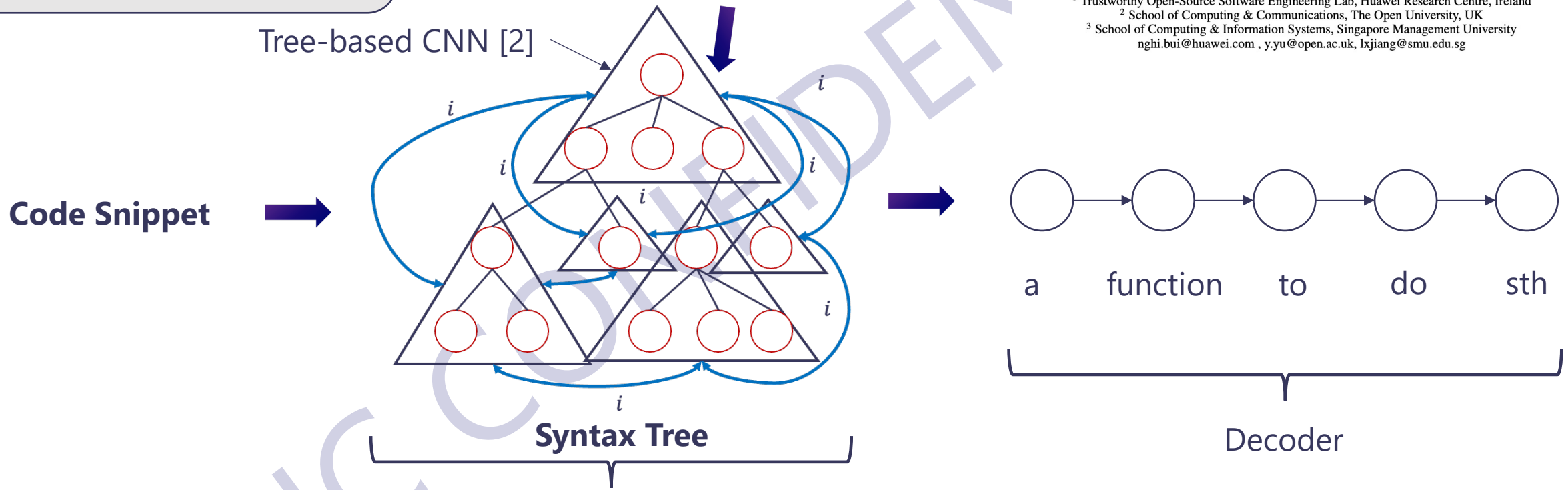
Modeling Syntax Tree Using Capsule Theory [1]

Capture the Dependency Across Different Parts of the Syntax Tree

The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)

TreeCaps: Tree-Based Capsule Networks for Source Code Processing

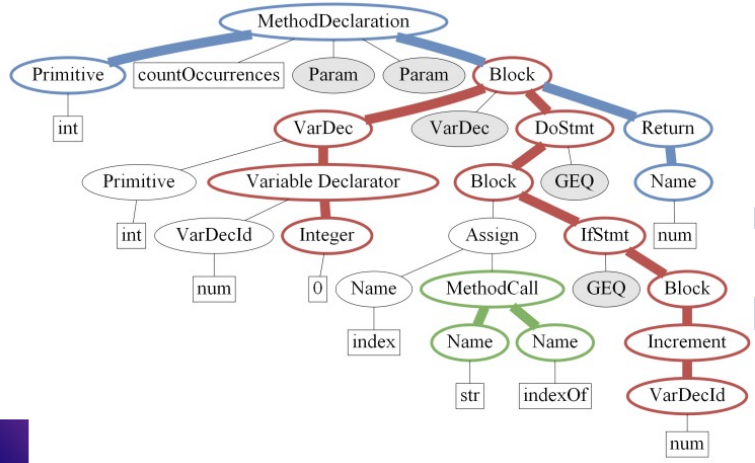
Nghi D. Q. Bui [1 3] **Yijun Yu** [1 2] **Lingxiao Jiang** [3]
[1] Trustworthy Open-Source Software Engineering Lab, Huawei Research Centre, Ireland
[2] School of Computing & Communications, The Open University, UK
[3] School of Computing & Information Systems, Singapore Management University
nghi.bui@huawei.com , y.yu@open.ac.uk, lxjiang@smu.edu.sg



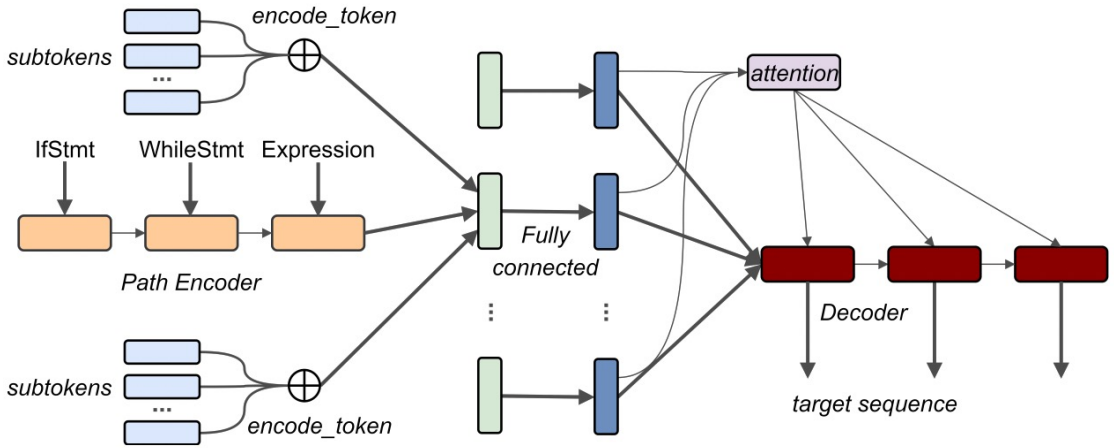
[1] Dynamic Routing Between Capsules, NIPS 2017
[2] Convolutional Neural Networks over Tree Structures for Programming Language Processing, AAAI 2016

Learning to Summarize Code by Path Abstraction

```java
int countOccurrences(String str, char ch) {
    int num = 0;
    int index = -1;
    do {
        index = str.indexOf(ch, index + 1);
        if (index >= 0) {
            num++;
        }
    } while (index >= 0);
    return num;
}
```



Modeling Paths of the Syntax Tree



code2vec: Learning Distributed Representations of Code

URI ALON, Technion
MEITAL ZILBERSTEIN, Technion
OMER LEVY, Facebook AI Research
ERAN YAHAV, Technion

We present a neural model for representing snippets of code as continuous distributed vectors ("code embed-

Published as a conference paper at ICLR 2019

CODE2SEQ: GENERATING SEQUENCES FROM STRUCTURED REPRESENTATIONS OF CODE

Uri Alon
Technion
urialon@cs.technion.ac.il

Shaked Brody
Technion
shakedbr@cs.technion.ac.il

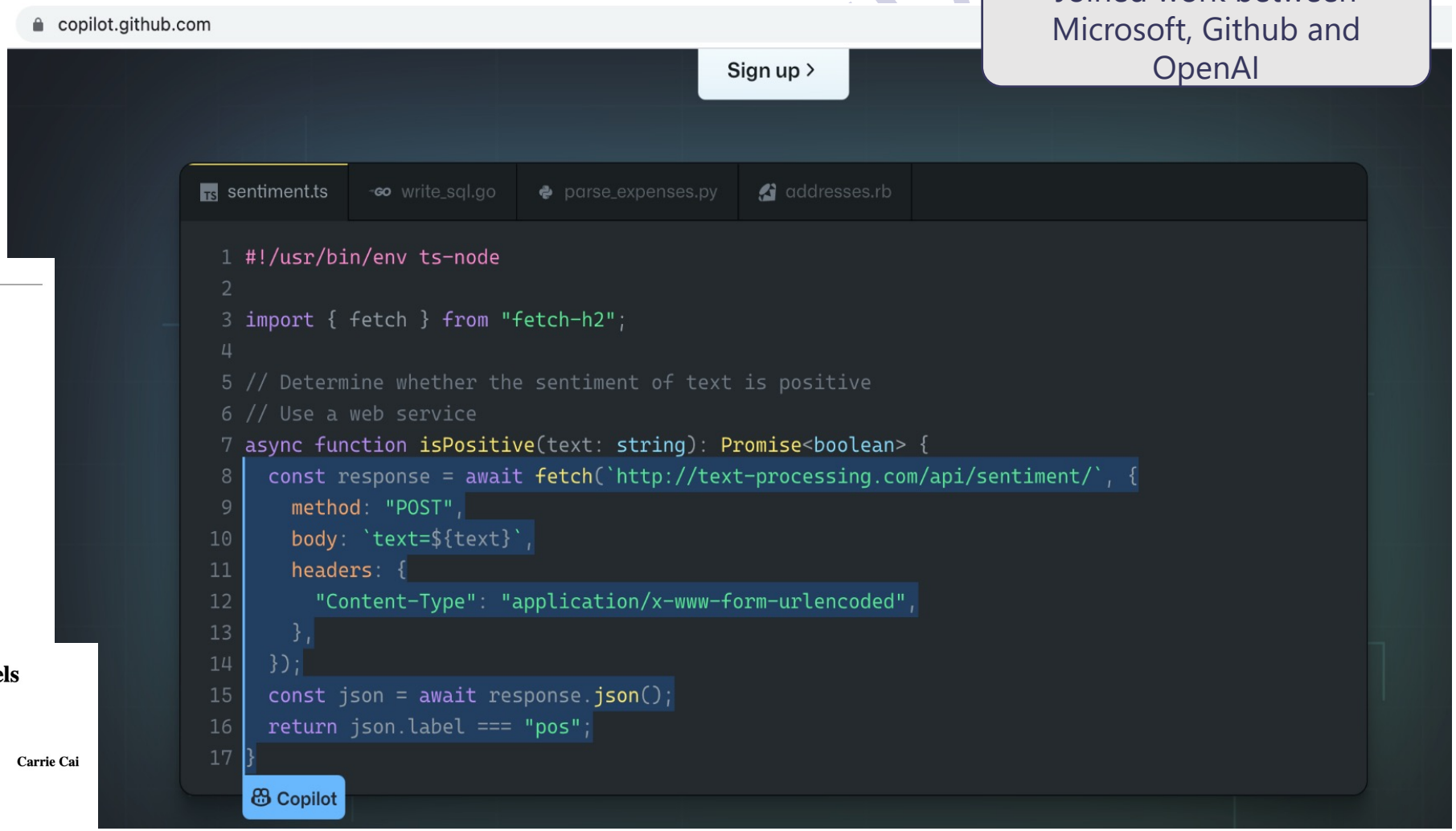
Omer Levy
Facebook AI Research
omerlevy@gmail.com

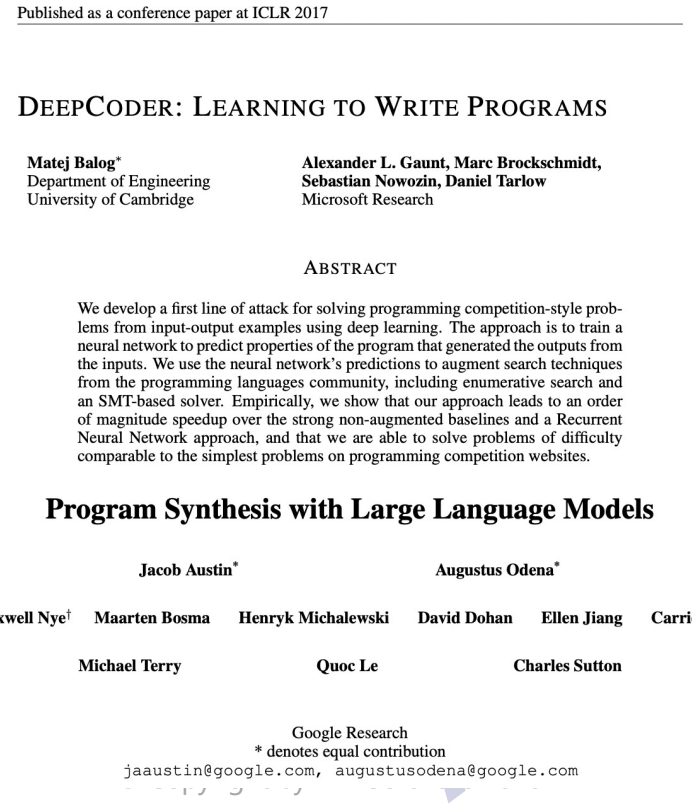
Eran Yahav
Technion
yahave@cs.technion.ac.il

Program Synthesis: Generate Program from Description

Program Synthesis is one of the Holy-Grail problem in Computer Science

Joined work between Microsoft, Github and OpenAI





Published as a conference paper at ICLR 2017

DEEPCODER: LEARNING TO WRITE PROGRAMS

Matej Balog*
Department of Engineering
University of Cambridge

Alexander L. Gaunt, Marc Brockschmidt,
Sebastian Nowozin, Daniel Tarlow
Microsoft Research

ABSTRACT

We develop a first line of attack for solving programming competition-style problems from input-output examples using deep learning. The approach is to train a neural network to predict properties of the program that generated the outputs from the inputs. We use the neural network's predictions to augment search techniques from the programming languages community, including enumerative search and an SMT-based solver. Empirically, we show that our approach leads to an order of magnitude speedup over the strong non-augmented baselines and a Recurrent Neural Network approach, and that we are able to solve problems of difficulty comparable to the simplest problems on programming competition websites.

Program Synthesis with Large Language Models

Jacob Austin* **Augustus Odena***

Maxwell Nye† **Maarten Bosma** **Henryk Michalewski** **David Dohan** **Ellen Jiang** **Carrie Cai**

Michael Terry **Quoc Le** **Charles Sutton**

Google Research
* denotes equal contribution
jaaustin@google.com, augustusodena@google.com

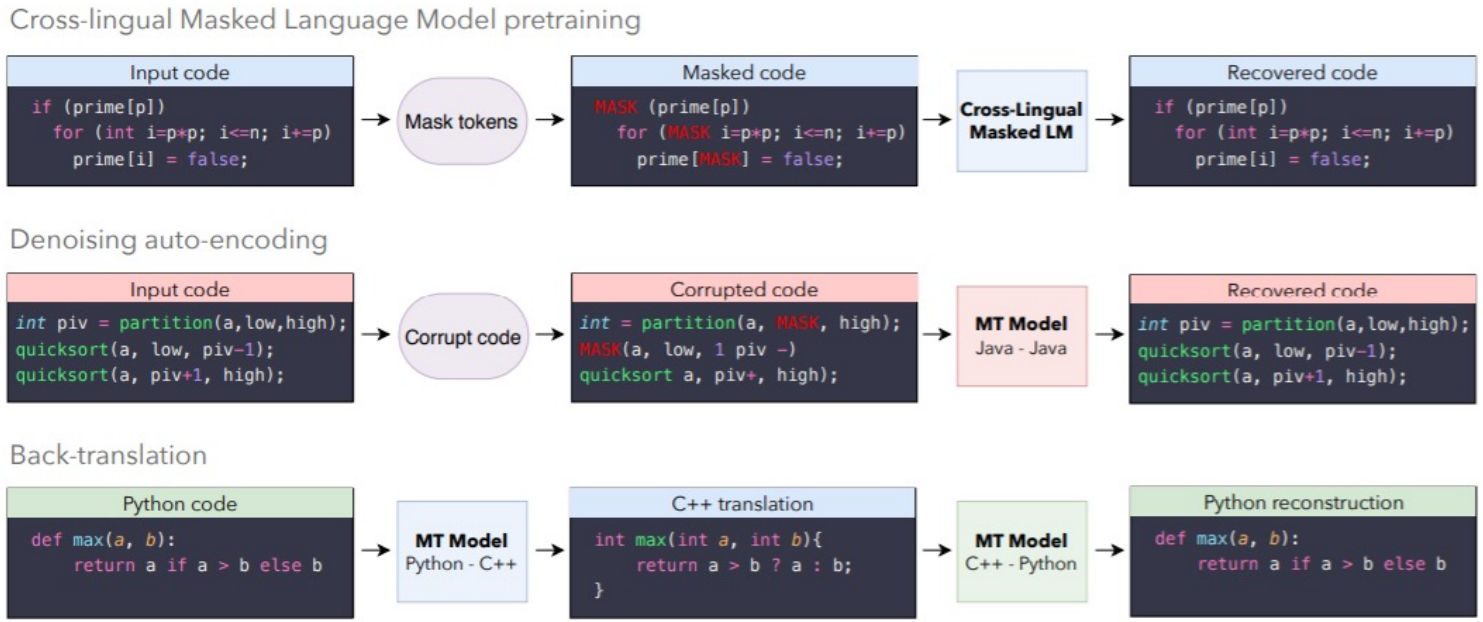
39

Code Migration

There is a huge need for language migration

- Language Migration
  - Cobol ➔ Python, Java, etc.
  - C ➔ Rust
  - …
- Library Version Migration
  - Python2 ➔ Python3
  - Tensorflow 1.x ➔ Tensorflow 2.x
  - ….
- Platform Migration
  - X86 ➔ ARM
  - Android ➔ OSX
  - Intel 8080 ➔ Intel 8086
  - ….

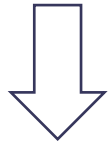
- **Build neural-based models to translate code (inspired from Natural language processing).**
- **Use compiler-based techniques to validate the translated programs.**



"**Unsupervised Programming Languages Translation**", Facebook AI Research

Code Migration in the Era of LLMs

Pretrained Foundation Large Language Models (GPT, T5, BERT)



Fine-Tune

- Large scale parallel datasets are available.

Zero-shots Learning

- LLMs can directly translate language X to language Y with certain level of accuracy.
- Work for cases that does not require very accurate translation, just need the sketch of the output.

Few-shots Learning

- Human can give instructions (in a form of a few samples – few shows) on how to translate (with additional contexts or configurations).

AI-Assisted Automated Testing

- Test Case Generation: To generate unit test given a function.
- Test Case Prioritization: Prioritize which test cases to run first to enhance the fault detection rate as early as possible.
- Image Recognition: automatically detect changes on a website and update element to DOM (code generation) without manual effort.
- Predictive Self-Healing: update test cases when the code changed.
- Bug Prediction: running test cases to predict bugs.
- .....

AI4Code Research at Fsoft: Wins & Innovations

AI4Code Research at FSoft

Intelligent Software Engineering

Technical Directions

Program Comprehension
- Code Summarization
- Code Refactoring
- Specs Understanding
- Code Search

Program Synthesis
- Test Case Generation
- Code Generation
- Text-to-SQL
- Code Migration

Code Quality
- Bug Localization
- Program Repair
- Defect Prediction

MLOps
- Requirement Engineering
- Continuous Delivery
- Performance Monitoring

Expected Outcomes

Combine program analysis techniques with machine learning to under
- Published top-tier conference papers
- Develop tools that can help developers to understand code betters.

Create software tools for code summarization and code search (APIs, stand-alone software)

Build large language models of code.
- Can be fine-tuned for different tasks
- Build an interface for interactive program synthesis.

Create software interfaces (e.g. plugins) that can assist developer in coding.
Create software packages that can assist non-developers for software-related tasks (e.g., retrieve data from SQL)

Analyze and understand the quality of code in Fsoft:
- Propose AI-models that can and enhance the code quality.
- Package the models into the tool that can be easily used by software developers

Create software tools that can scan the code, localize bugs and suggest fixes for such bugs.

Introduce industry and open source community best practices:
- Analyzing Software Design Best Practices to deploy AI-based software
- Requirements Engineering for AIOps.
- Academic research in MLOps.

Developer's Pain Points

Low-Code Developer

Wants to create apps for a specific platform but doesn't want to spend hours learning to code.

Junior Developer

Just joined a company, can program but doesn't fully understand code format of company.

Experienced Developer

Wants to avoid duplicate, repetitive, or basic-level work that could be automated.

AI4Code Solutions

Low-Code Developer

- Interactive Code Generation (Natural language to Code)
- Code Summarization
- Low-code/No-code Automation
- Text-to-SQL

Junior Developer

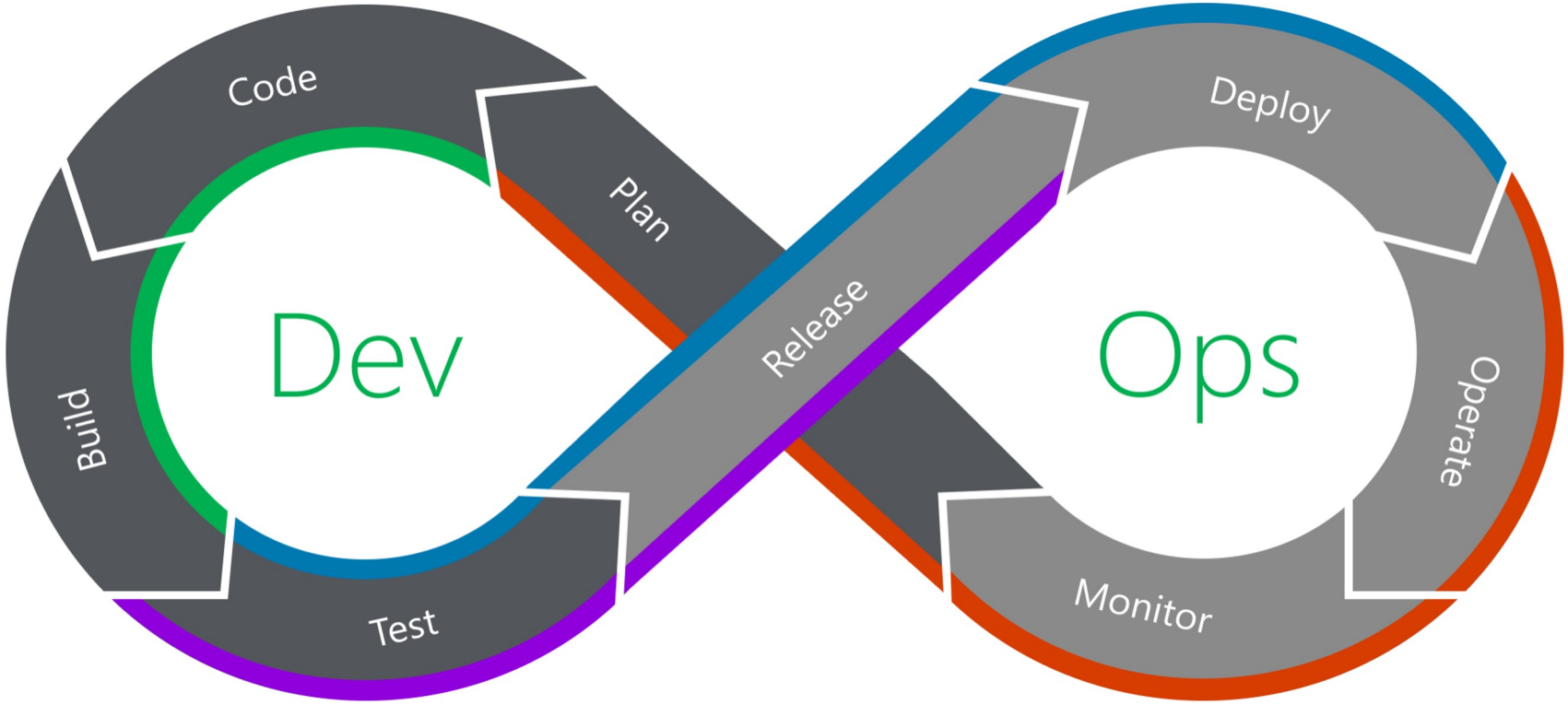
- Code Search/ Completion
- Anti-Pattern Detection
- Code Migration

Experienced Developer

- Bug Localization / Vulnerability Detection
- AI Testing Automation
- Root cause analysis for Bugs/Failure
- AI Assisted Refactoring

The AI-Powered Software Life Cycle

The AI-powered development for easier, faster, and more reliable software development. Enable broader access with better low/no-code support.



Wins & Innovation from In-houss LLMs

	Problem(s) Solved	Relatedness to FSoft	Status
Docify	• Generate comments for code automatically. • Refactor code with decent variable names and identifiers (**Self-documenting**). • Integrated with **DoxyGen** as an end-to-end solution to generate code documentation.	• Some projects in **GAM** utilize **DoxyGen**, a 20-year-old tool, to document code that adheres to ISO standards. Although DoxyGen assists in generating the format, the actual code comments must still be **written manually**. • Code documentation is also an important issues for other projects within Fsoft.	• MVP ready, • Rolling out • ~600 installtions • ~1100 registered users.
TestGen	• Generate unit tests for a class in Java and Python. • The test cases are more diverse and have higher code coverage than DiffBlue.	• Teams at Fsoft often report a **shortage** of **developers skilled in writing tests**, which is also a **tedious** and time-consuming task. • TestGen aims to **replace DiffBlue** as the primary solution for generating unit tests within the company	Model Ready
CodeQuality	Identify issues in source code, related to best practice, anti-pattern and security bug	• Can assist teams in Fsoft to debug customer's code faster and more reliable.	On-going

Docify: Understand & Generate Code Document using LLMS

Accept translation | Reject translation

```
(Source language: English)
    /**
    * Returns the larger of two ints. This is equivalent to the LINQ's max () function except that it doesn't take into account the sign of the arguments.
    *
    * @param a - the first int to compare. Can be negative.
    * @param b - the second int to compare. Can be negative.
    *
    * @return the larger of a and b or a if a is greater than b and b is less than a
    */
(Target language: Japanese (日本語))
    /**
    * LINQ の max () 関数に相当する.ただし,この関数は議論の符号を考慮しない.
    *
    * @param a 負のインテクトができます.
    * @param b 負のインテクトができます.
    *
    * @return a と b の大きいか a が b と b が a より大きいか
    */

    static int max(int a, int b) {
        return (a > b) ? a : b;
    }

    /**
    * Knapsack's algorithm. Given a set of weights and a set of values this method returns the maximum value of the Knapsack's algorithm.
    *
    * @param W - the number of neighbors in the graph. Must be at least W + 1.
    * @param wt - the weight vector of the neighbors. It is assumed that wt [ i ] is greater than or equal to w.
    * @param val - the value vector of the neighbors. It is assumed that val [ i ] is greater than or equal to wt [ i ].
    * @param n - the number of neighbors. It is assumed that n is greater than or equal to W.
    *
    * @return the maximum value of the Knapsack's algorithm. This is an implementation of the algorithm
    */
    static int knapSack(int W, int wt[], int val[], int n) {
        int i, w;
        int K[][] = new int[n + 1][W + 1];
```

Docstring generated by Docify

Support translation to Japanese

Docify: Self-Documenting Concept

```
function chunkData(e, t) {
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t) {
    if (i + t < r) {
      n.push(e.substring(i, i + t));
    } else {
      n.push(e.substring(i, r));
    }
  }
  return n;
}
```

Code with meaningless identifiers

```
function divideDataIntoChunks(data, chunkSize) {
    var chunks = [];
    var dataLength = data.length;
    var currentIndex = 0;
    for (; currentIndex < dataLength; currentIndex += chunkSize) {
        if (currentIndex + chunkSize < dataLength) {
            chunks.push(data.substring(currentIndex, currentIndex + chunkSize));
        } else {
            chunks.push(data.substring(currentIndex, dataLength));
        }
    }
    return chunks;
}
```

Code with meaningful identifiers, easier to read and maintain

TestGen: Unit Test Generation

```java
package com.javatpoint.logic;
public class Calculation {

    public static int findMax(int arr[]){
        int max=0;
        for(int i=1;i<arr.length;i++){
            if(max<arr[i])
                max=arr[i];
        }
        return max;
    }
}
```

Function

```java
import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class CalculationTest {

    @Test
    public void testFindMax() {
        assertEquals(4, Calculation.findMax(new int[] {1, 3, 4, 2}));
        assertEquals(-1, Calculation.findMax(new int[] {-12, -3, -4, -1}));
    }
}
```

Corresponding unit test

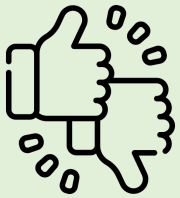
Practical Consideration

Complement, Not Replace

Low-Resource Environment

Explanability

Privacy-Preserving



Feedback Loop

Metrics

THANK YOU

Contact: nghibdq@fsoft.com.vn
Website: https://bdqnghi.github.io/