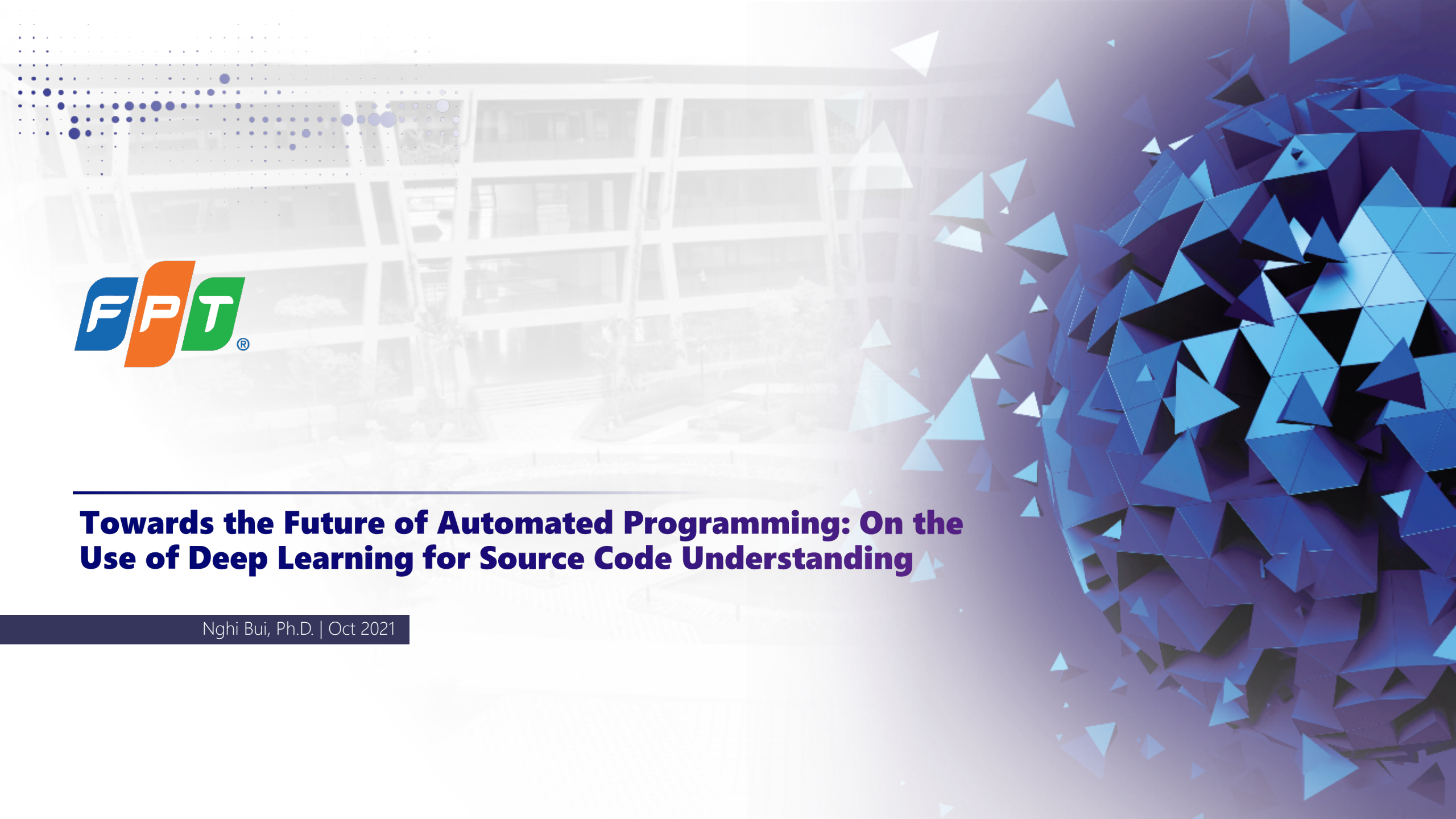


Towards the Future of Automated Programming: On the Use of Deep Learning for Source Code Understanding

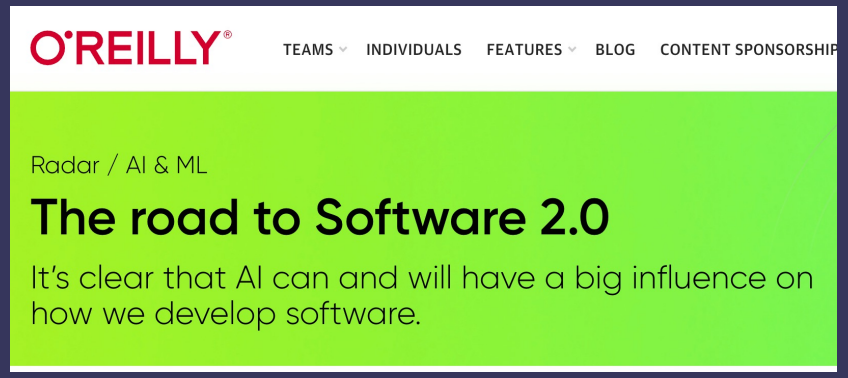
Nghi Bui, Ph.D. | Oct 2021

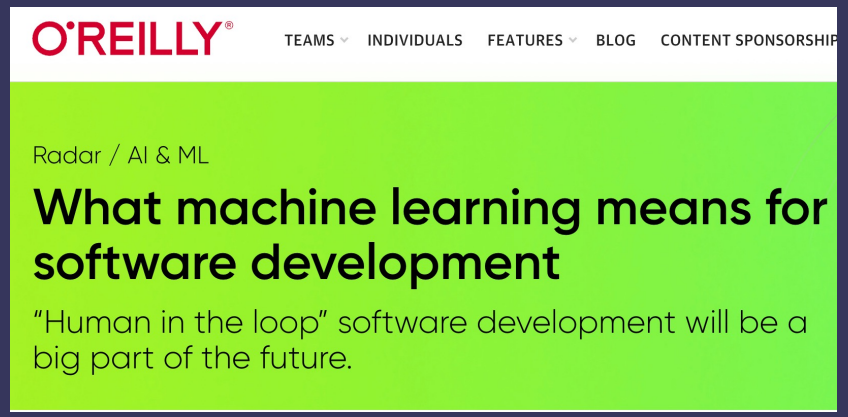
About the Speaker

- Ph.D. in Computer Science from Singapore Management University.
  Research Interests:
  - Software Analytics, Programming Languages, Deep Learning.
- Previous Affiliations:
  - Research Scientist: Software Analytics Research Lab, Singapore Management University.
  - Research Scientist: Trustworthy Open Source and Software Engineering Lab, Huawei Research Center in Ireland.
- Current Affiliation:
  - AI Scientist/Mentor: FSoft AI Lab.
- Website: https://bdqnghi.github.io/
- Github: https://github.com/bdqnghi

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

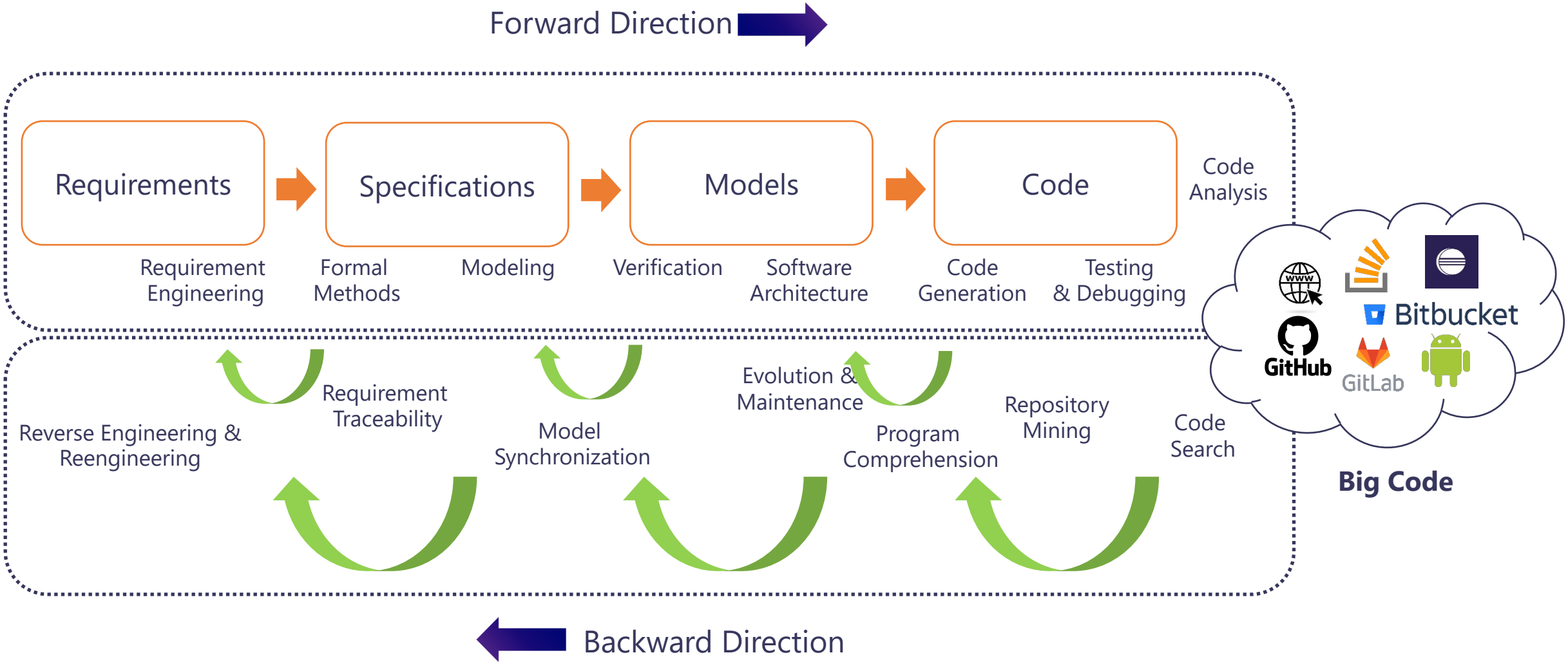
Machine Programming

AI4Code

Automated Programming

The Automation of Software Development

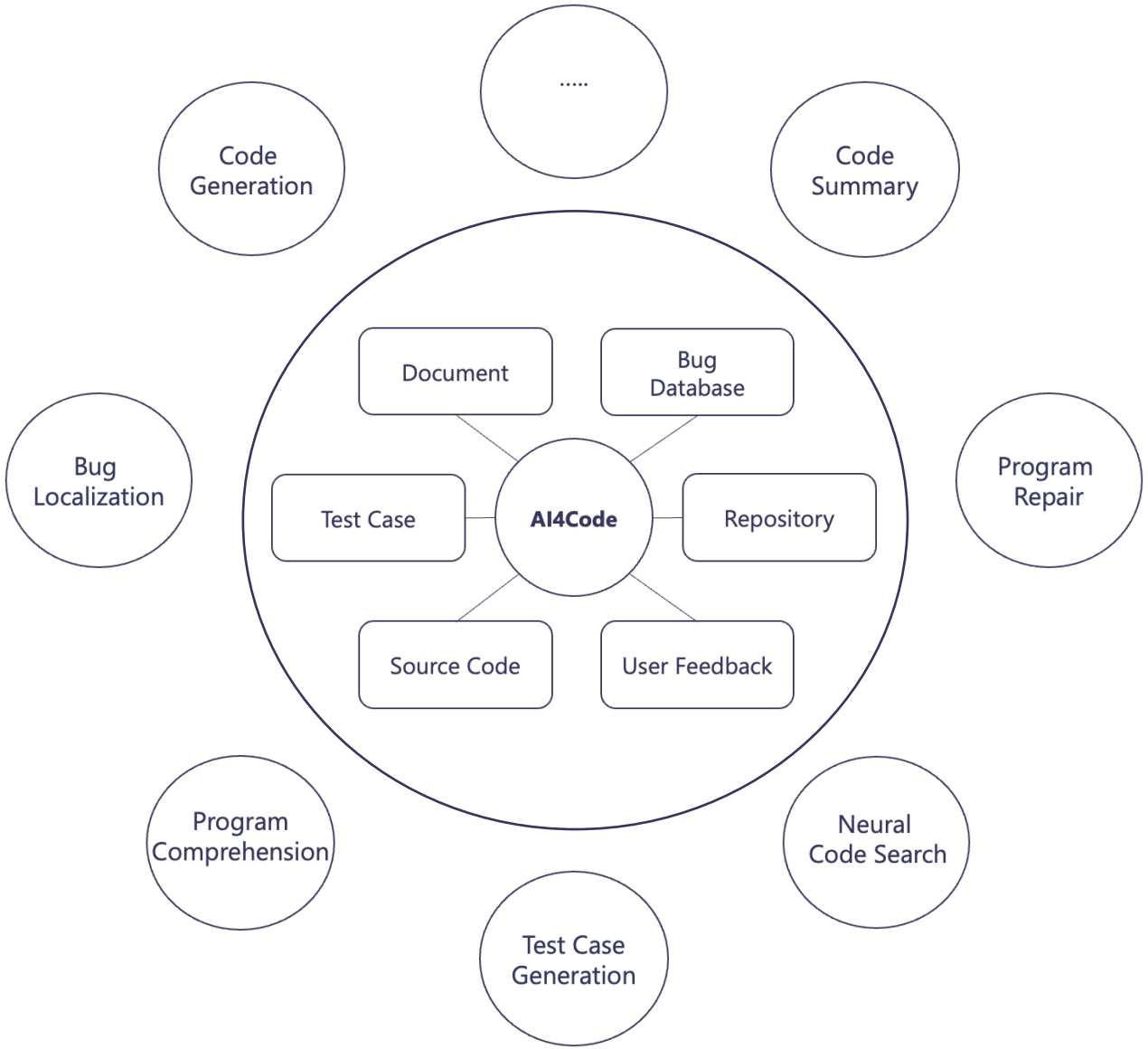
Areas in Software Development



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



HOW WOULD THAT IMPACT SOFTWARE DEVELOPMENT?

PRODUCTIVITY

PERFORMANCE

RELIABILITY

Software's Productivity

Jun 20, 2019, 10:22pm EDT | 28,193 views

The Future With Level 5 Autonomous Cars

SOFTWARE IS LIMITING AUTONOMOUS SYSTEMS

Well-publicized accidents and missteps show that even at this limited autonomy level, AI systems struggle in application.

Software

Programmer's Performance

YOU CODE IS SUFFERING FROM THE NINA GAP

Only 10% of programmers are ninjas

Ninja Gap: Initiated by Intel, which is the performance difference between an amateur programmer and an expert programmer.

System's Reliability

What Boeing's 737 MAX Has to Do With Cars: Software

Investigators believe faulty software contributed to two fatal crashes. A newly discovered fault will likely will keep the 737 MAX grounded until the fall.

Who's responsible when an autonomous car crashes?

US & WORLD | TECH | POLITICS

Facebook apologizes after wrong translation sees Palestinian man arrested for posting 'good morning'

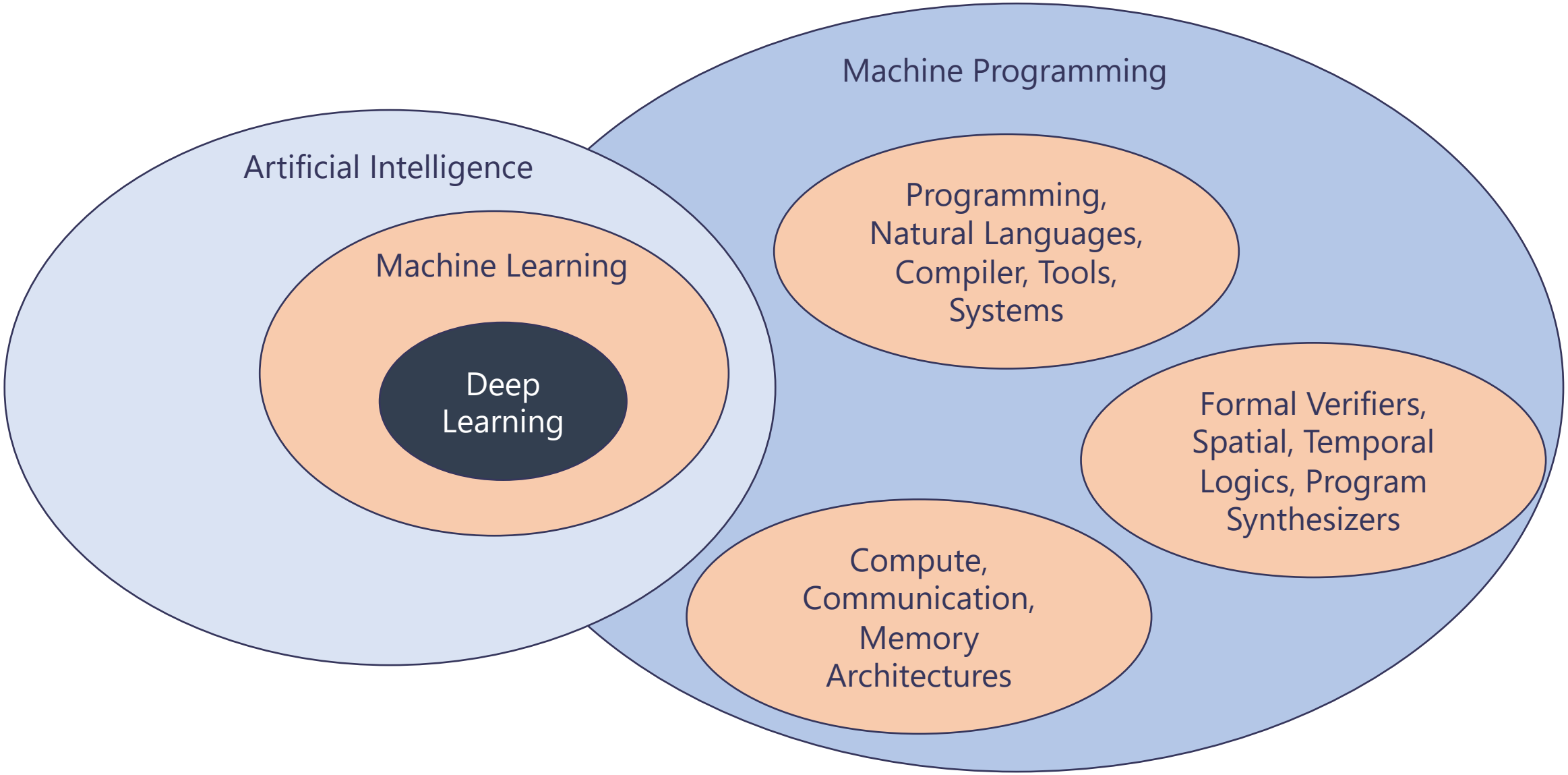
AUTONOMOUS SYSTEMS IS SUFFERRING FROM LIFE-THREATENING BUGS

Machine Programming

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

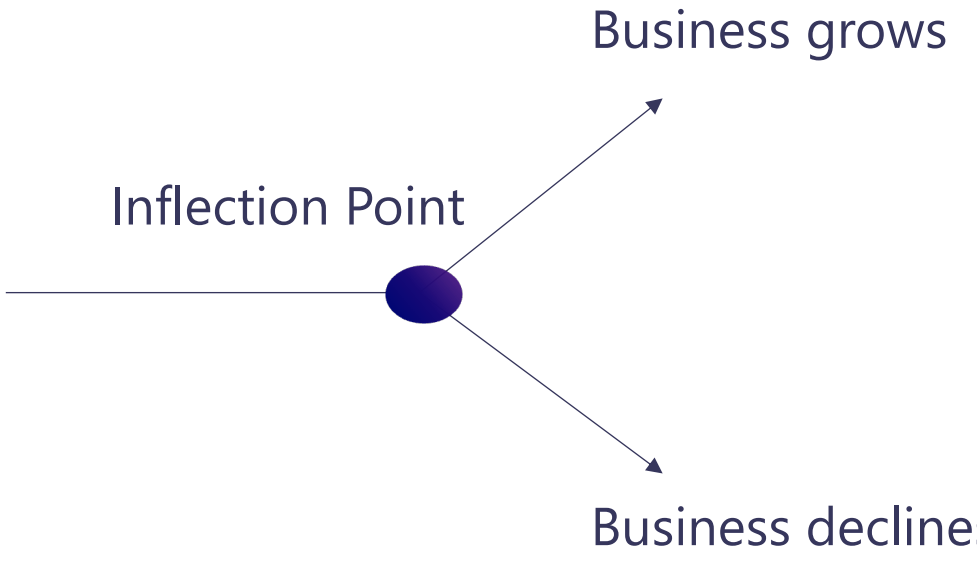
- Increasing the software’s **productivity.**
- Closing the **performance** gap between amateur developer and expert developer.
- Enhancing the **reliability** of software system.

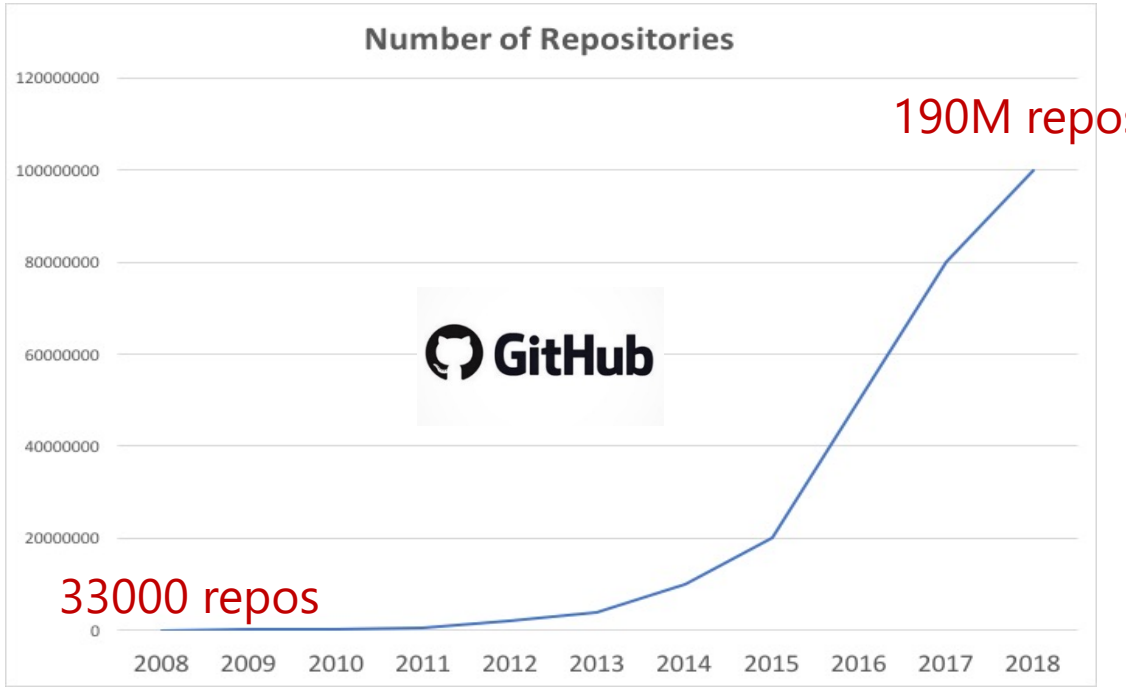
A View of Machine Programming



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 !!)

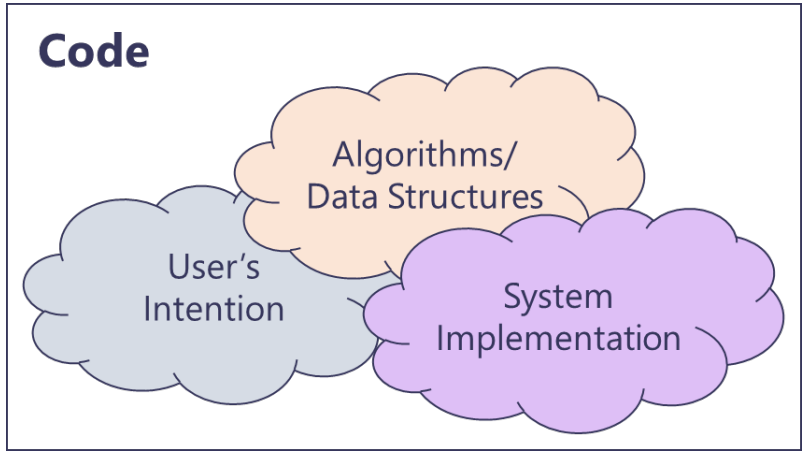




Machine Programming needs all of this to work

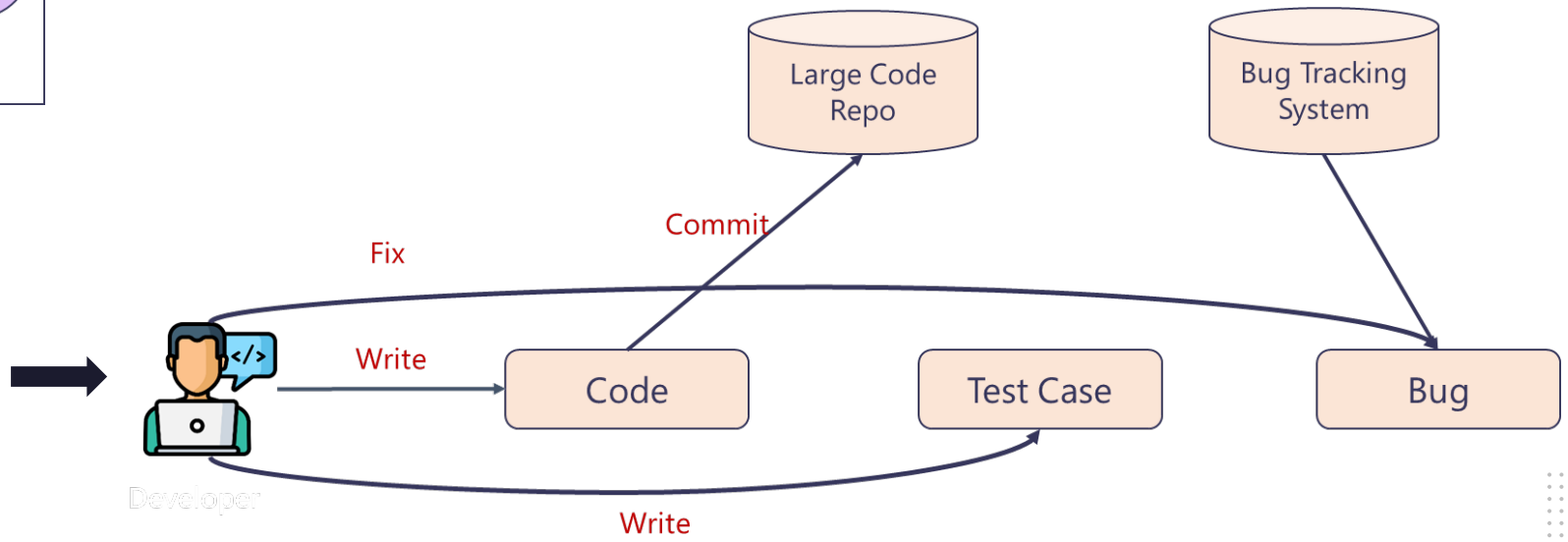
Today's Programming

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

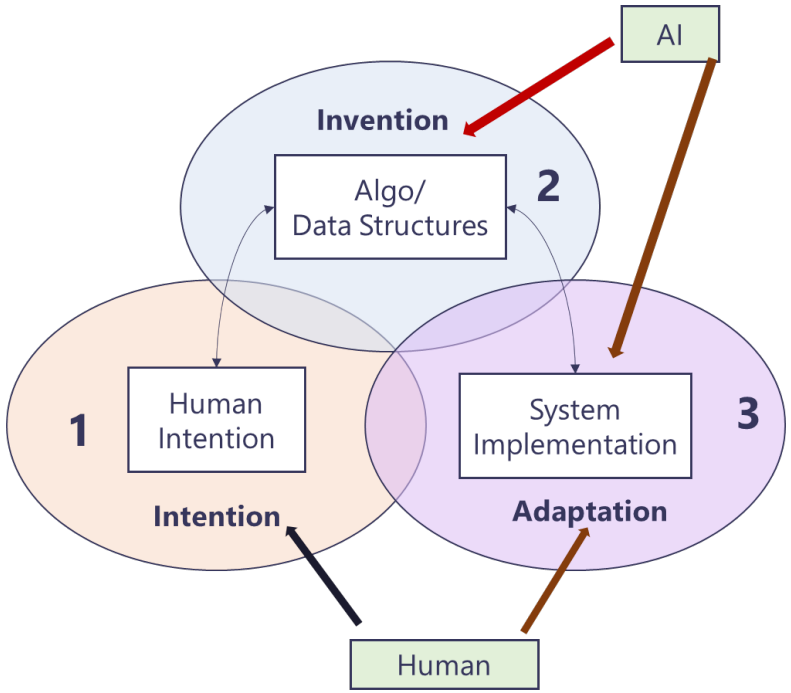


- A program usually contains:
  - The developer's intent
  - Code to express that intent.
  - Specifications for software/hardware adaptation.
- Human developers must handle all of these steps.

Basic work-flow of Today's Programming



Tomorrow's Programming

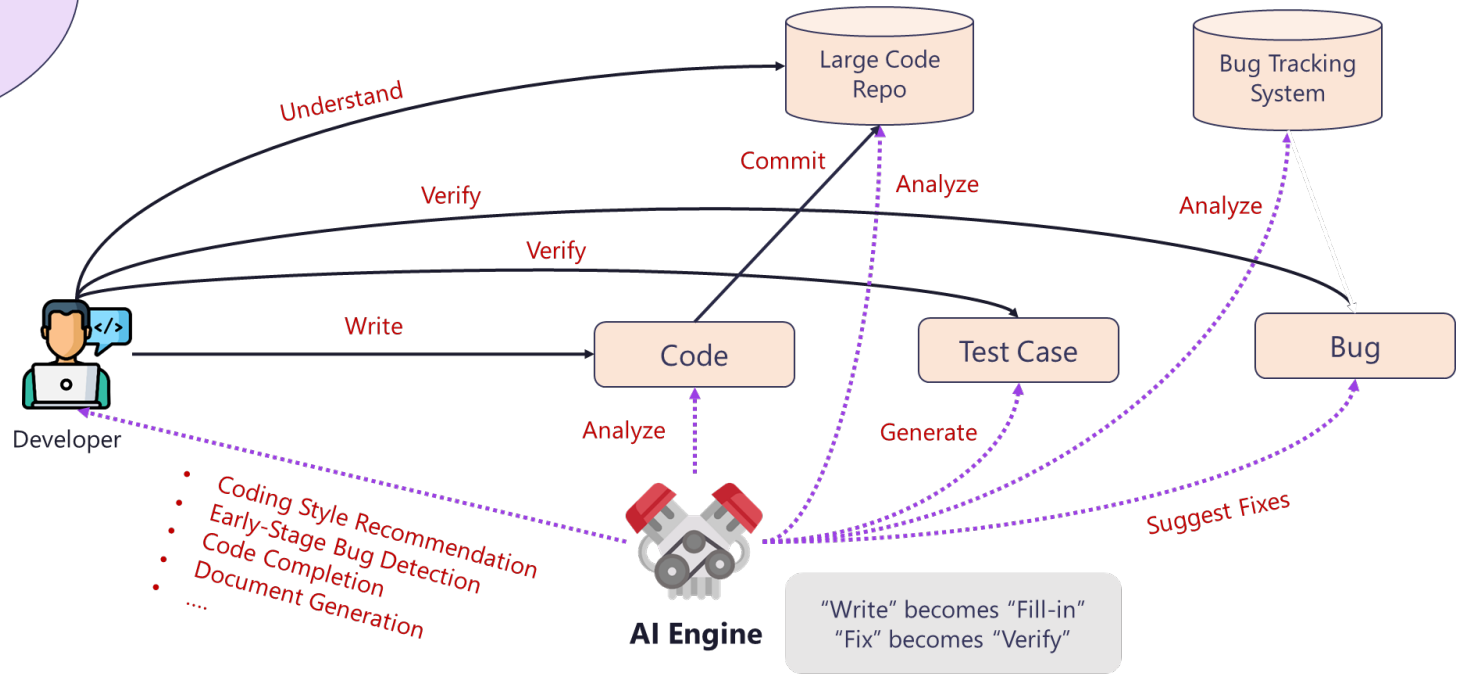


- Human only specifies intention:

"**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?

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

Machine Programming Principles and Applications

Programming Language vs Natural Language

Executability: Programming language is executable, natural language is not

Formality: programming languages are designed top-down by a few designers for many users. Natural languages, in contrast, emerge, bottom up, “through social dynamics”.

Cross-Channel Interaction: Code’s two channels, the algorithmic and the explanatory channels, interact through their semantic units.

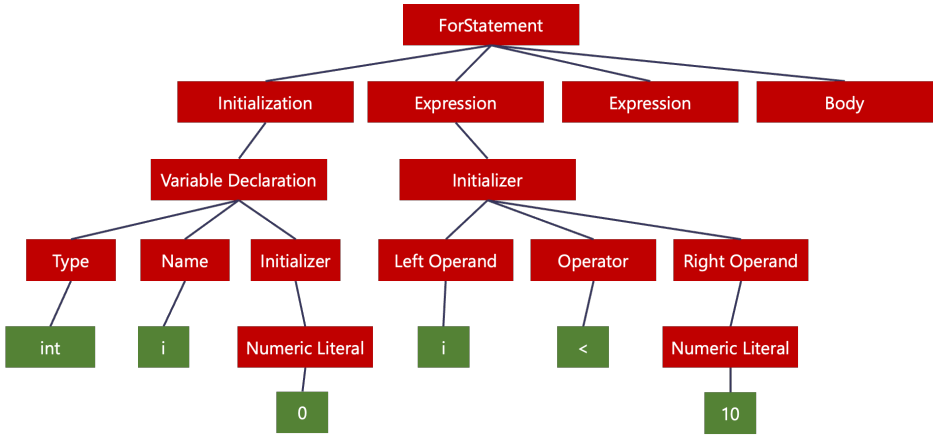
→ **NLP techniques are applicable for programming language, but needs additional contexts from Compiler/Formal Language theory.**

Code Representations

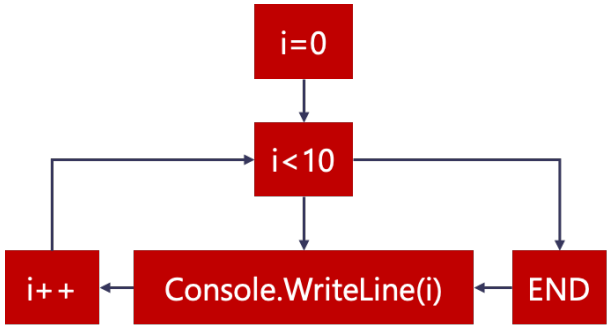
Token-based

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

Syntax-based (Tree)



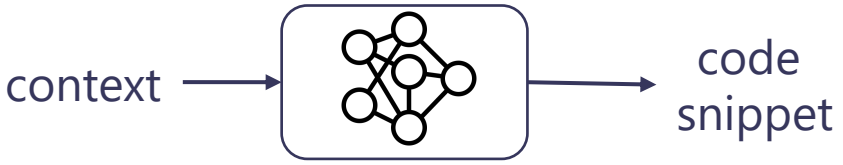
Graph-based



Probabilistic Models of Code

Code Generating Models

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

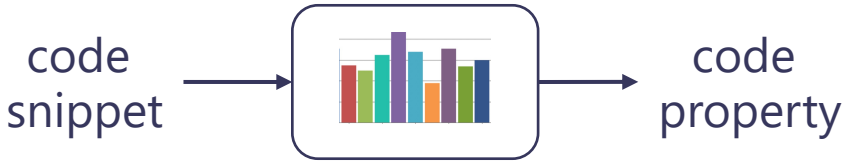


$$P_{\mathcal{D}}(\mathbb{c}|C(\mathbb{c}))$$

- 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

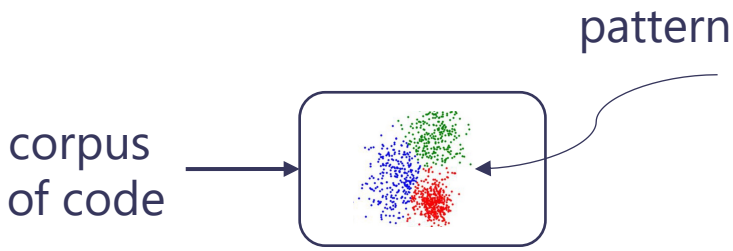


$$P_{\mathcal{D}}(\pi|f(\mathbb{c}))$$

- 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

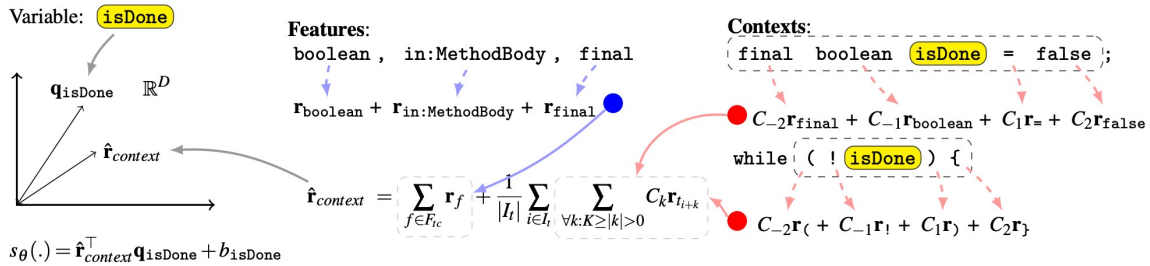
Suggesting Accurate Method and Class Names

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



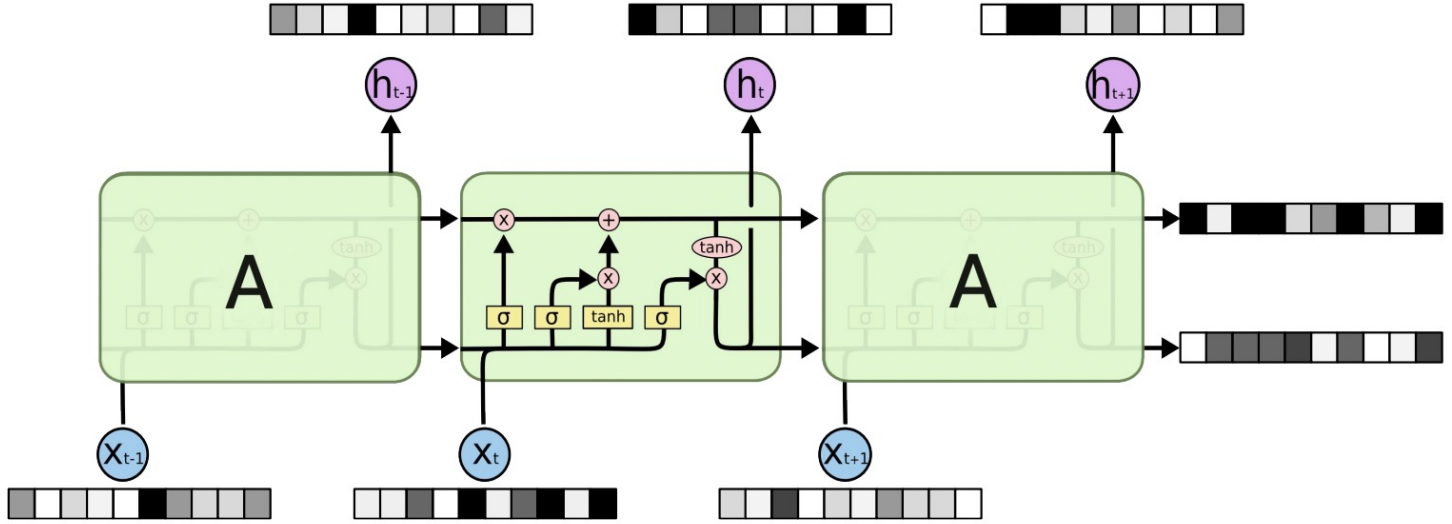
Log-Bilinear model to learn distributed representation of code

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

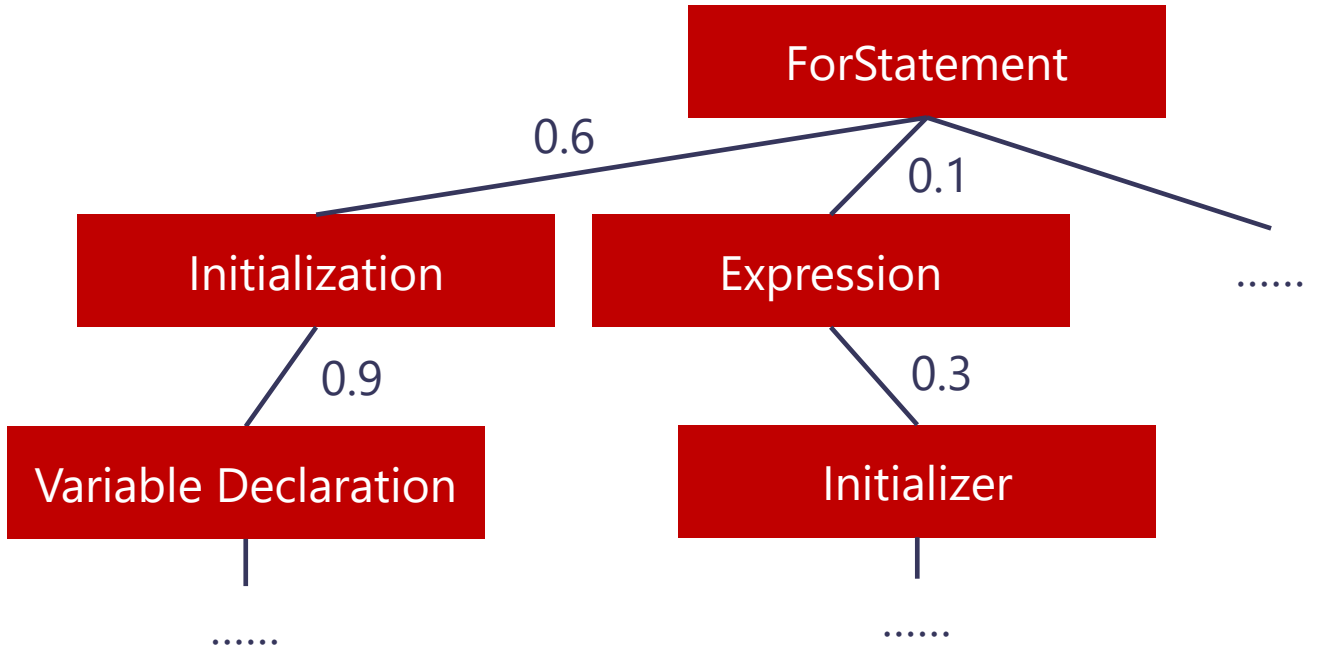



```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
  int i;
  if (classes[class]) {
    for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
      if (mask[i] & classes[class][i])
        return 0;
  }
  return 1;
}
```

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. Copyright 2014 by the author(s).

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.

Learning to Fix Bug

Bug detection tools are dominated by Program Analysis

sonarqube.org/developer-edition/?gads_campaign=Asia-Code&gads_ad_group=Analysis

WEBINAR Next week: Join The SonarSource Team for the US 2021

sonarqube Product What's New Documentation Community

Developer EDITION Enterprise EDITION

Built for Developers By Developers

Innovative features to systematically track and improve Code Quality and Code Security in your applications

fbinfer.com

Infer Docs Support Blog Twitter Facebook GitHub Search

A tool to detect bugs in Java and C/C++/Objective-C code before it ships

Infer is a static analysis tool - if you give Infer some Java or C/C++/Objective-C code it produces a list of potential bugs. Anyone can use Infer to intercept critical bugs before they have shipped to users, and

OWASP PROJECTS CHAPTERS EVENTS ABOUT Search OWASP

Source Code Analysis Tools

Contributor(s): Dave Wichers, itamarlavender, will-obrien, Eitan Worcel, Prabhu Subramanian, kingthorin, coadaflorin, hblankenship, GovorovViva64, pfhorman, GouveaHeitor, Clint Gibler, DSotnikov, Ajin Abraham, Noam Rathaus, Mike Jang

Source code analysis tools, also known as Static Application Security Testing (SAST) Tools, can help analyze source code or compiled versions of code to help find security flaws.

SAST tools can be added into your IDE. Such tools can help you detect issues during software development. SAST tool feedback can save time and effort, especially when compared to finding vulnerabilities later in the development cycle.

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.

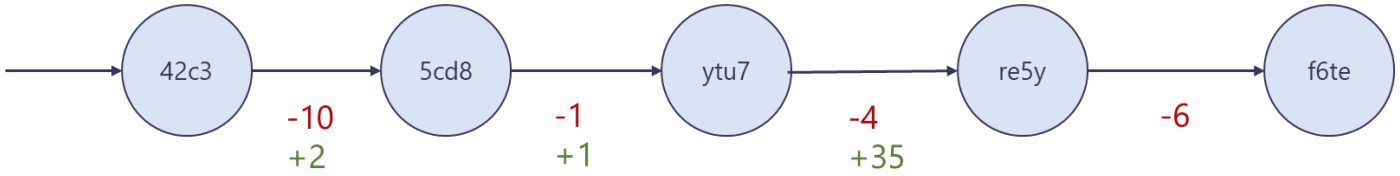
```
1     public Table getTableForAlias(String alias) {
2 -       return this.aliasToTable.get(alias);
3 +       return this.aliasToTable.get(alias.toLowerCase());
4     }
```

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

```
@@ -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() {
```

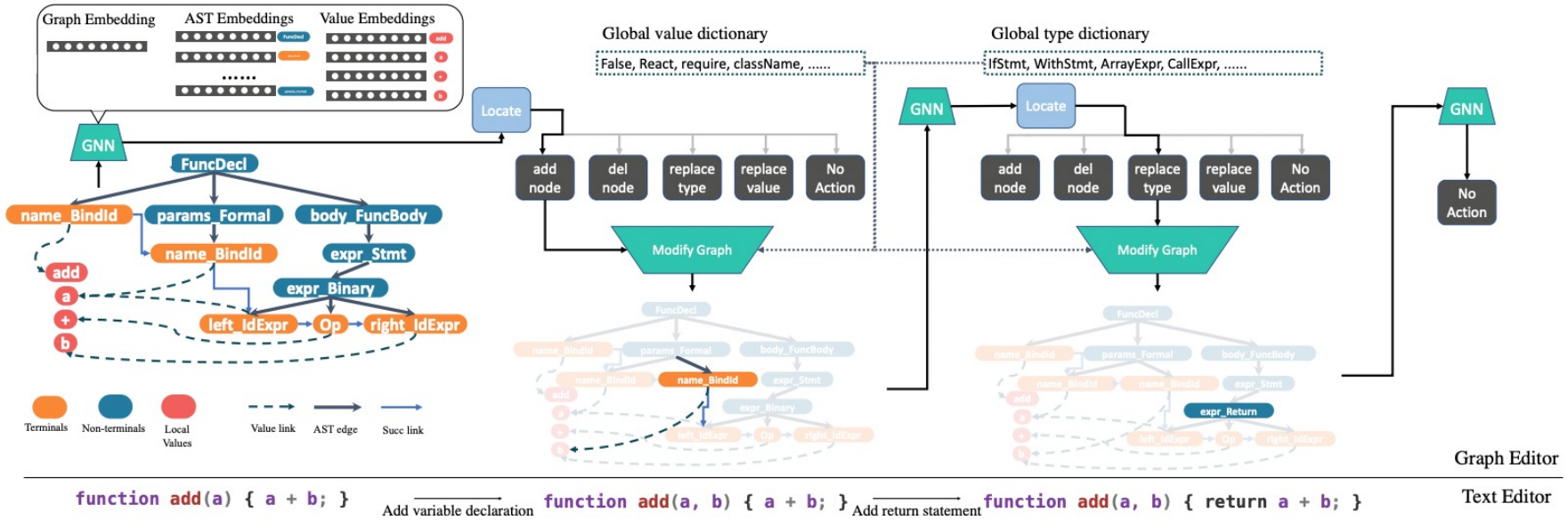
The programm was first pilote on October 23 in Orava and Bardejov two regions with the highest number of Covid-19 cases in the country. Nearly 141.000 people, 91% of those who were eligible, got tested in the two regions over the three days of the pilot. In the rest of the countitry the testing took 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

Learning to Fix Bug from Commit History (cont.)



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

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

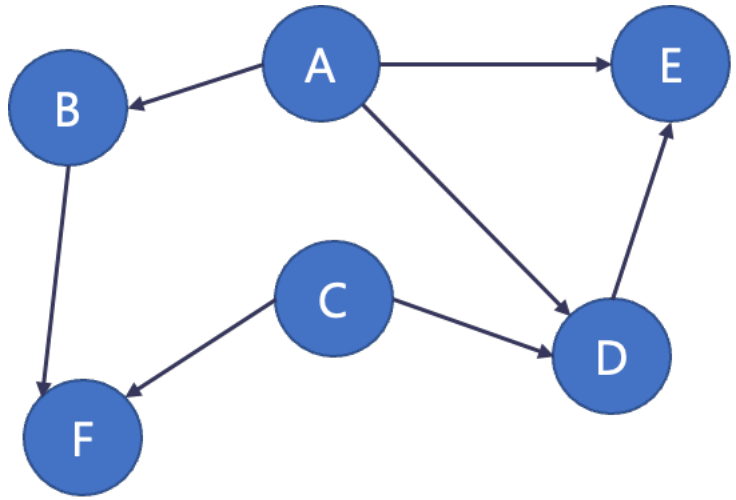
Michael Pradel
Facebook
michael@binaervarianz.de

Satish Chandra
Facebook
satch@fb.com

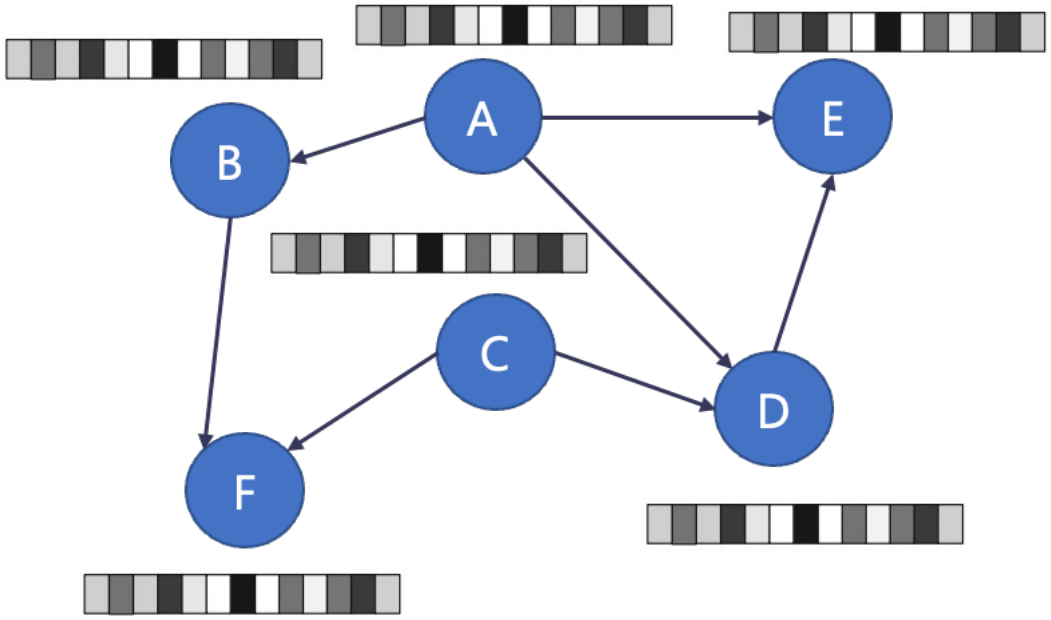
$$p(g_{fix}|g_{bug};\theta) = p(g_1|g_{bug};\theta)p(g_2|g_1;\theta)\ldots p(g_{fix}|g_{T-1};\theta)$$

Graph Neural Network for Code Understanding

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

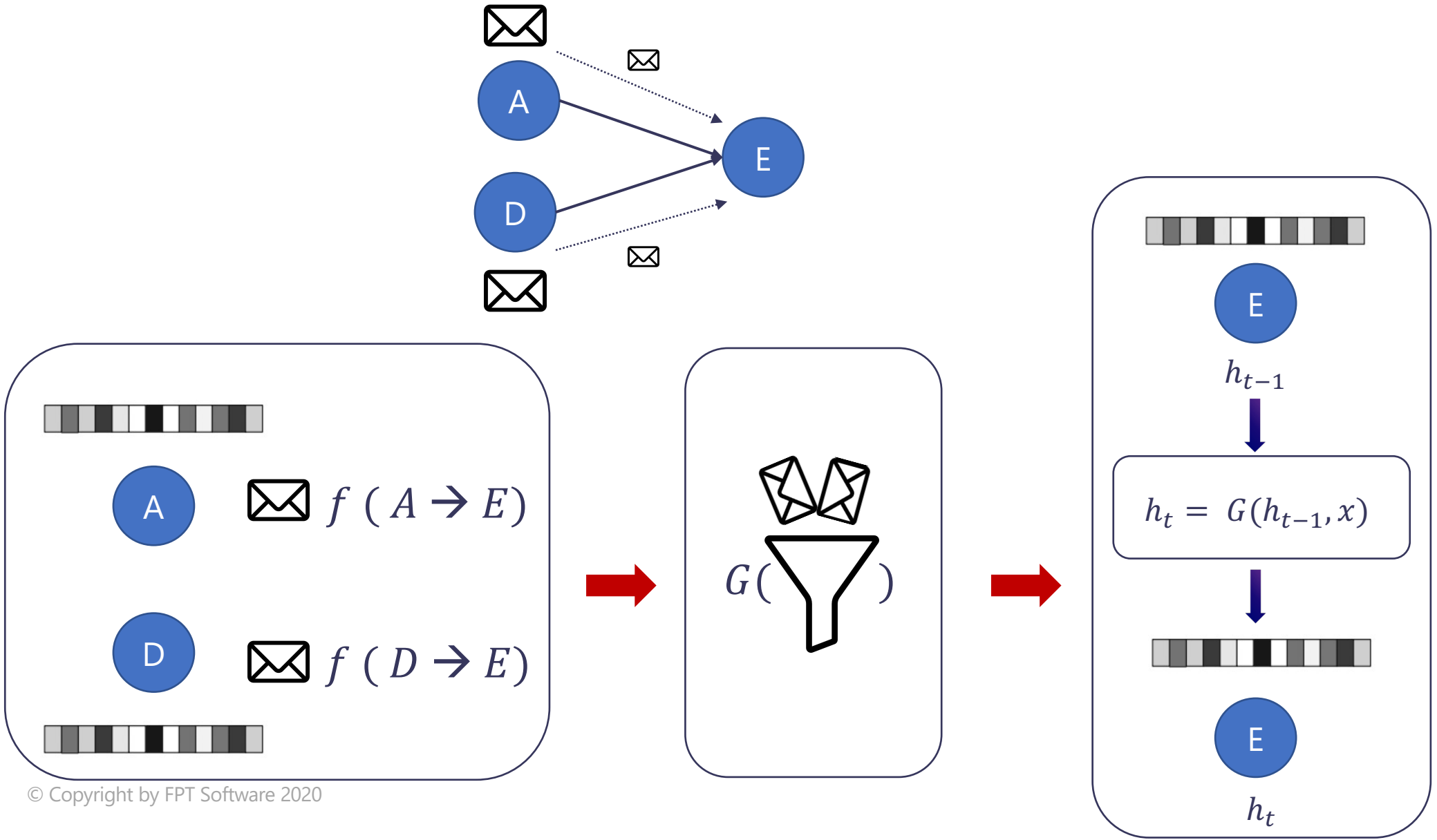


Graph-based Problem

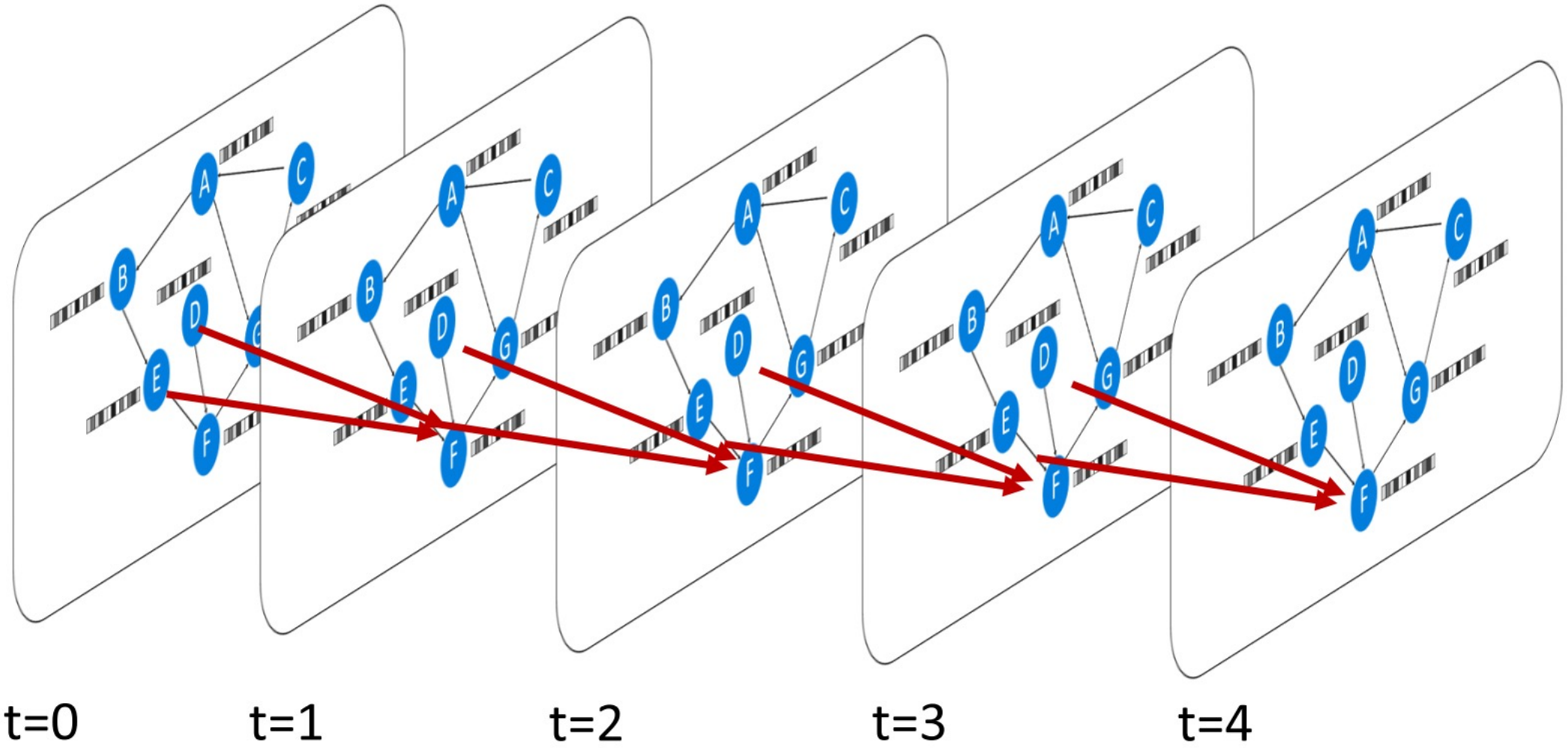


Initial Representation of each ode

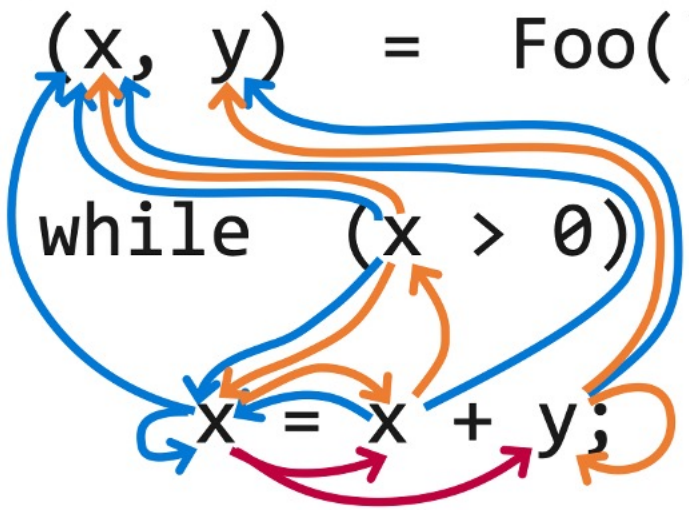
Graph Neural Network: Message Passing



Graph Neural Network: Message Passing (cont.)



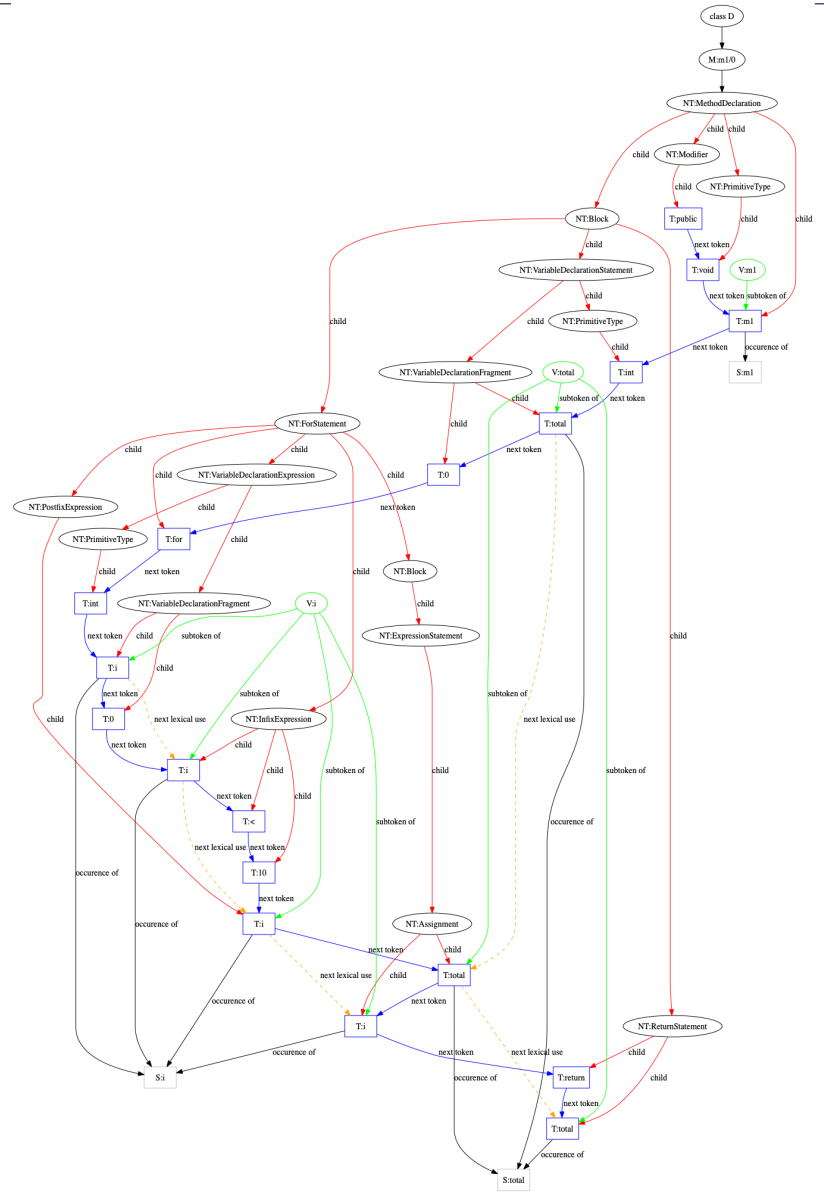
Represent a Program as Graph with Data Flow



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 );      first     clazz

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

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

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

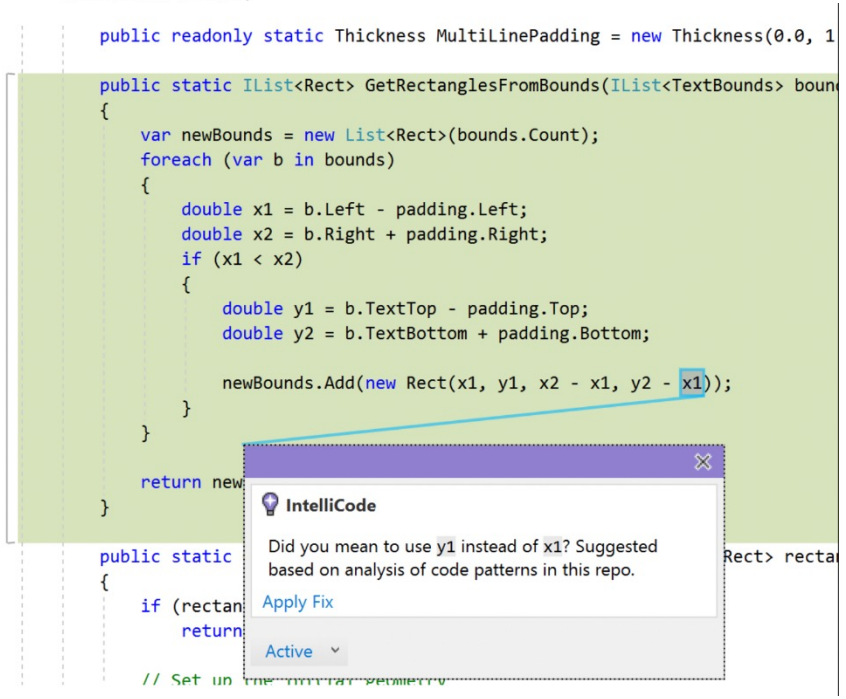
Published as a conference paper at ICLR 2018

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 conditional random field

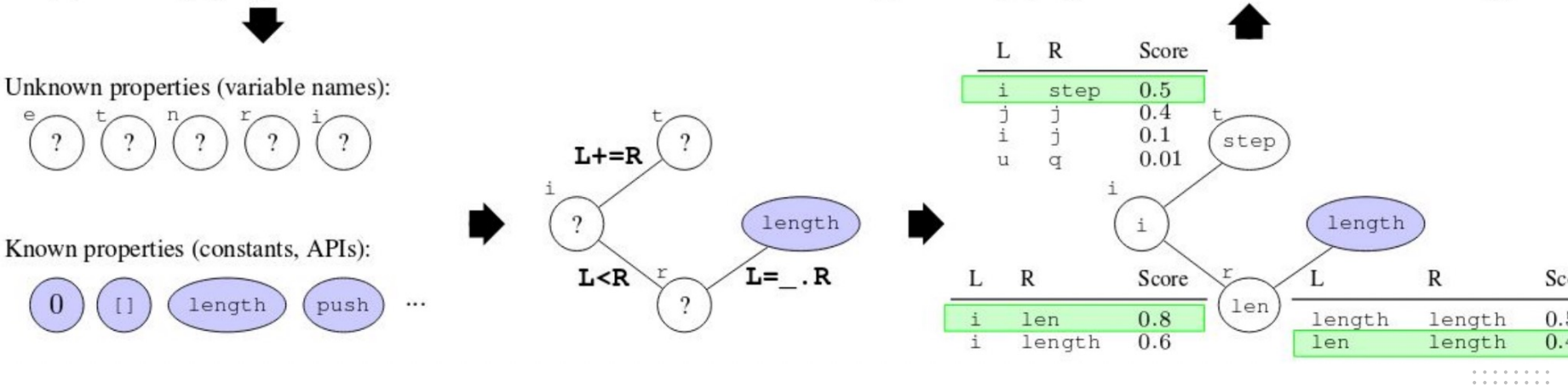
```
function chunkData(e, t) {
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t) {
    if (i + t < r) {
      n.push(e.substring(i, i + t));
    } else {
      n.push(e.substring(i, r));
    }
  }
  return n;
}
```

(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

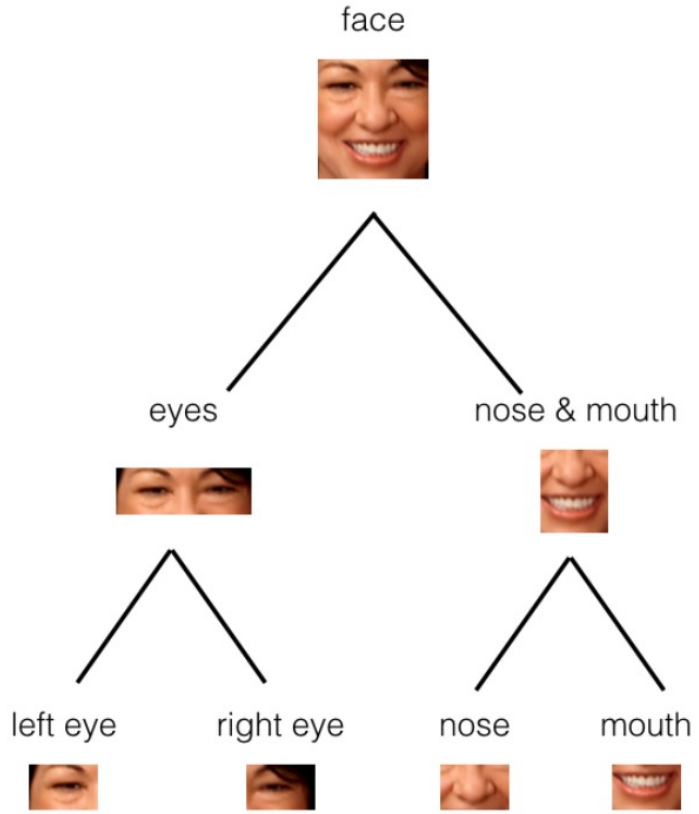
```
/*
A function to [redacted]
*/
void [redacted]()
  TreeView myTreeView = new TreeView()
  myTreeView.Nodes.Clear();
  foreach (string parentText in xml.parent)
  {
    TreeNode parent = new TreeNode();
    parent.Text = parentText;
    myTreeView.Nodes.Add(treeNodeDivisions);
    foreach (string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

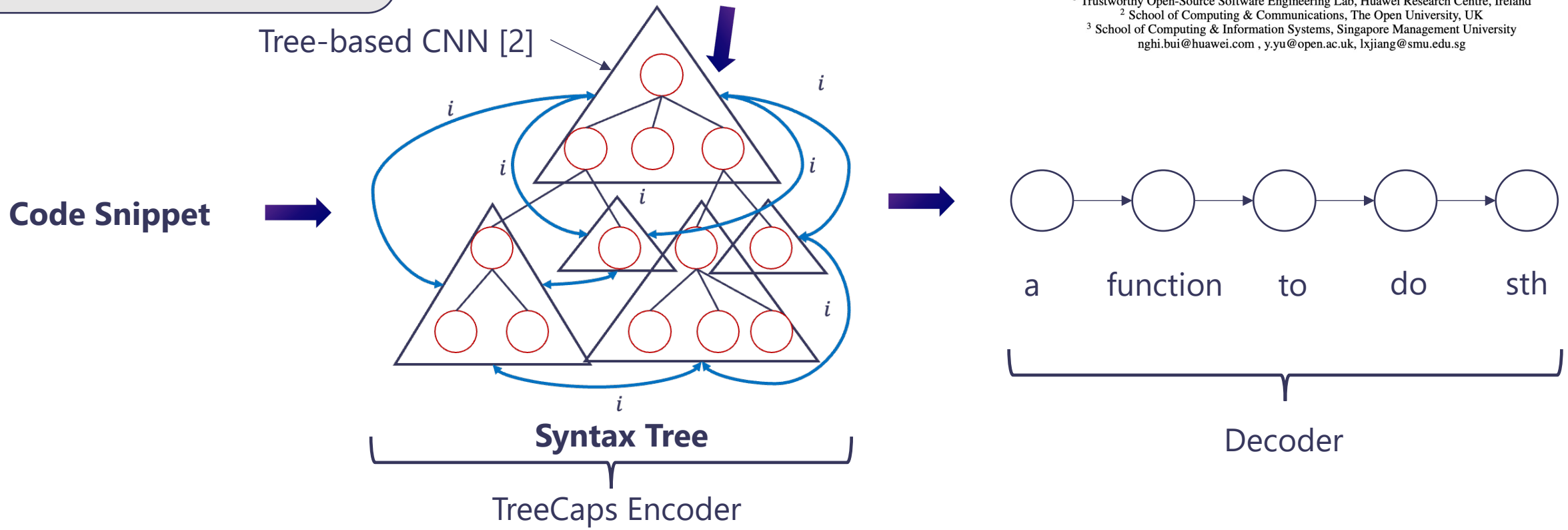
Geoffrey E. Hinton
Google Brain
Toronto
{sasabour, frosst, geoffhinton}@google.com

- 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

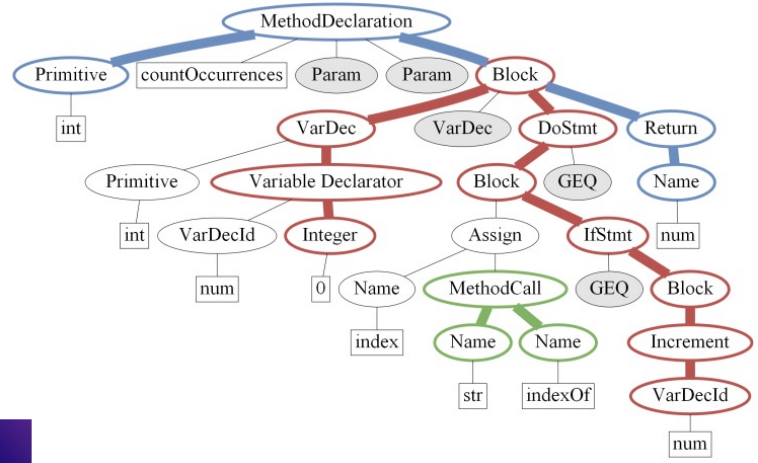
Nghi D. Q. Bui [1 3] **Yijun Yu** [1 2] **Lingxiao Jiang** [3]

[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

```
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;
}
```



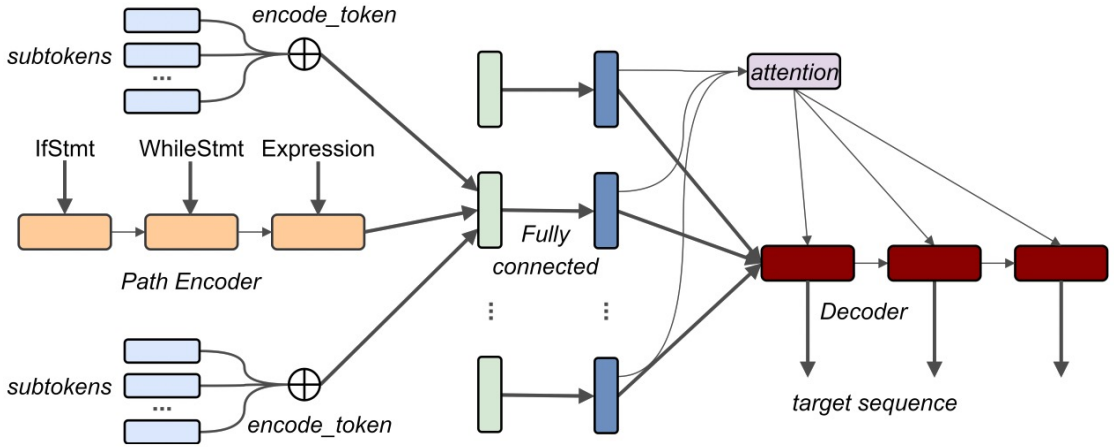
Modeling Paths of the Syntax Tree

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: Large Language Models for Code

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

Evaluating Large Language Models Trained on Code

Mark Chen [*1] Jerry Tworek [*1] Heewoo Jun [*1] Qiming Yuan [*1] Henrique Ponde de Oliveira Pinto [*1] Jared Kaplan [*2] Harri Edwards [1] Yuri Burda [1] Nicholas Joseph [2] Greg Brockman [1] Alex Ray [1] Raul Puri [1] Gretchen Krueger [1] Michael Petrov [1] Heidy Khlaaf [3] Girish Sastry [1] Pamela Mishkin [1] Brooke Chan [1] Scott Gray [1] Nick Ryder [1] Mikhail Pavlov [1] Alethea Power [1] Lukasz Kaiser [1] Mohammad Bavarian [1] Clemens Winter [1] Philippe Tillet [1] Felipe Petroski Such [1] Dave Cummings [1] Matthias Plappert [1] Fotios Chantzis [1] Elizabeth Barnes [1] Ariel Herbert-Voss [1] William Hebgen Guss [1] Alex Nichol [1] Alex Paino [1] Nikolas Tezak [1] Jie Tang [1] Igor Babuschkin [1] Suchir Balaji [1] Shantanu Jain [1] William Saunders [1] Christopher Hesse [1] Andrew N. Carr [1] Jan Leike [1] Josh Achiam [1] Vedant Misra [1] Evan Morikawa [1] Alec Radford [1] Matthew Knight [1] Miles Brundage [1] Mira Murati [1] Katie Mayer [1] Peter Welinder [1] Bob McGrew [1] Dario Amodei [2] Sam McCandlish [2] Ilya Sutskever [1] Wojciech Zaremba [1]

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.

prompt

Write a python function to check if a given number is one less than twice its reverse. Your code should satisfy these tests:

```
assert check(70) == False
assert check(23) == False
assert check(73) == True
```

model

```
def check(n) :
    if n == 2*int(str(n)[::-1])-1 :
        return True
    else :
        return False
```

prompt

Write a function to find the smallest missing element in a sorted array. Your code should satisfy these tests:

```
assert smallest_missing([0, 1, 2, 3, 4, 5, 6], 0, 6) == 7
assert smallest_missing([0, 1, 2, 6, 9, 11, 15], 0, 6) == 3
assert smallest_missing([1, 2, 3, 4, 6, 9, 11, 15], 0, 7) == 0
```

model

```
def smallest_missing(arr, n, m):
    smallest = min(n, m)
    for i in range(n, m + 1):
        if arr[i] <= smallest:
            smallest += 1
    return smallest
```

prompt

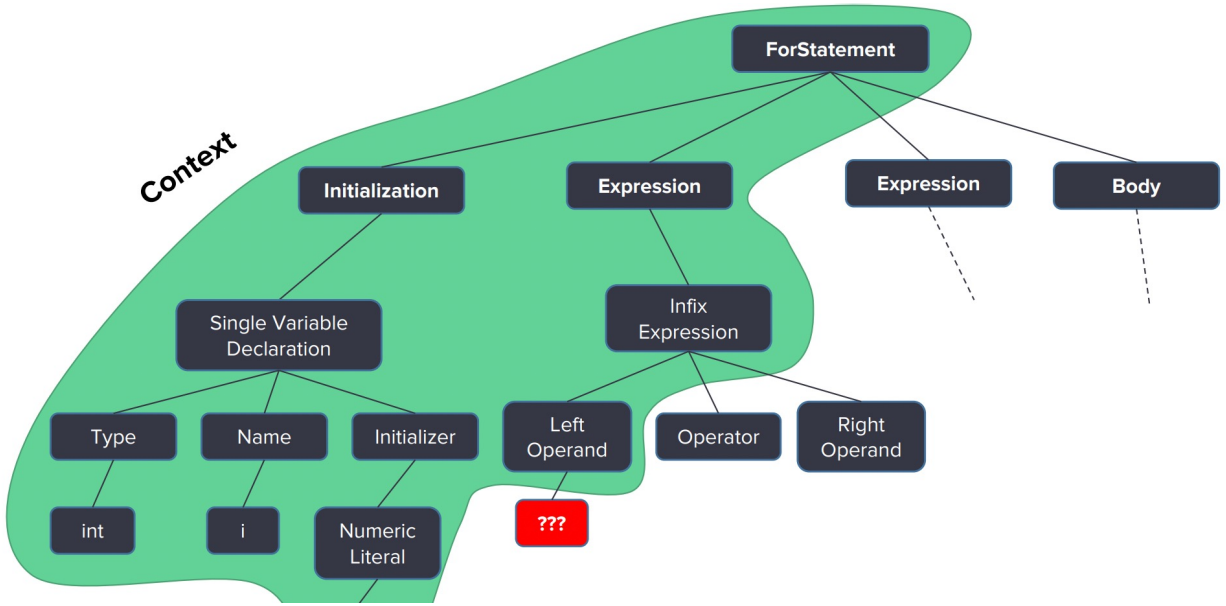
Please, solve the mathematical problem: a and b start walking towards each other at 4pm at a speed of 2 kmph and 3 kmph. They were initially 15 km apart. At what time do they meet? n0 = 4.0, n1 = 2.0, n3 = 15.0.

model

```
n0 = 4.0
n1 = 2.0
n2 = 3.0
n3 = 15.0
t0 = n1 + n2
t1 = n3 / t0
answer = n0 + t1
```

How Machine Programming Can Bring Benefits to Fsoft?

Code Generation




```
private static void log(String value) {
  if (value != null
      && ???
    value = value.substring(0, 55)+"...";
  LOG.info(value);
}
```

True ref:		`value.length() > 55`	
SLM top-5:	(9.6%)	`value.length() > 0`	
	(7.3%)	**`value.length() > 55`**	✓
	(1.8%)	`value.startsWith("...")`	
	(1.5%)	`!value.startsWith("...")`	
	(0.9%)	`value.charAt(0) == '.'`	

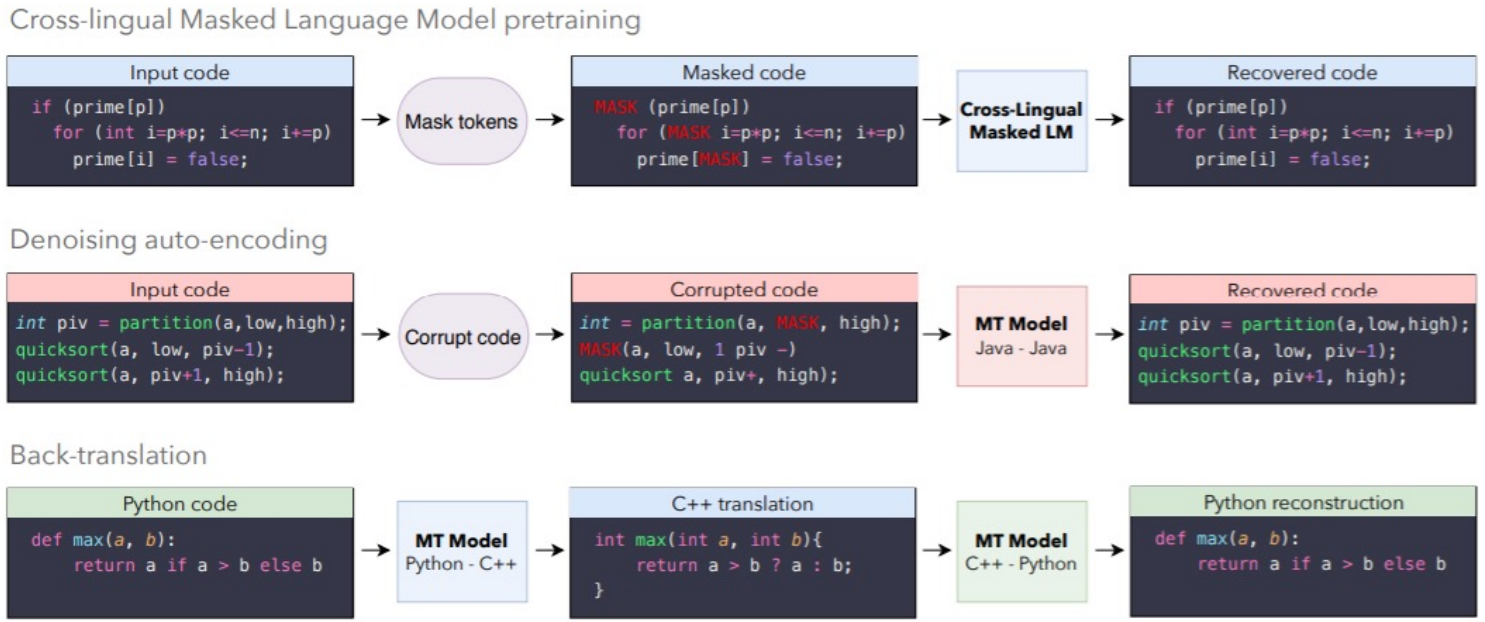
Fill in the missing code given the surrounding context

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.**



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

AI-Assisted Automation 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.
- .....

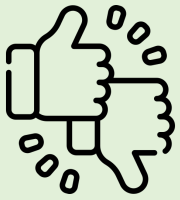
Practical Consideration

Complement, Not Replace

Low-Resource Environment

Explainability

Privacy-Preserving



Feedback Loop

Metrics

THANK YOU

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