Machine Learning for Program Analysis

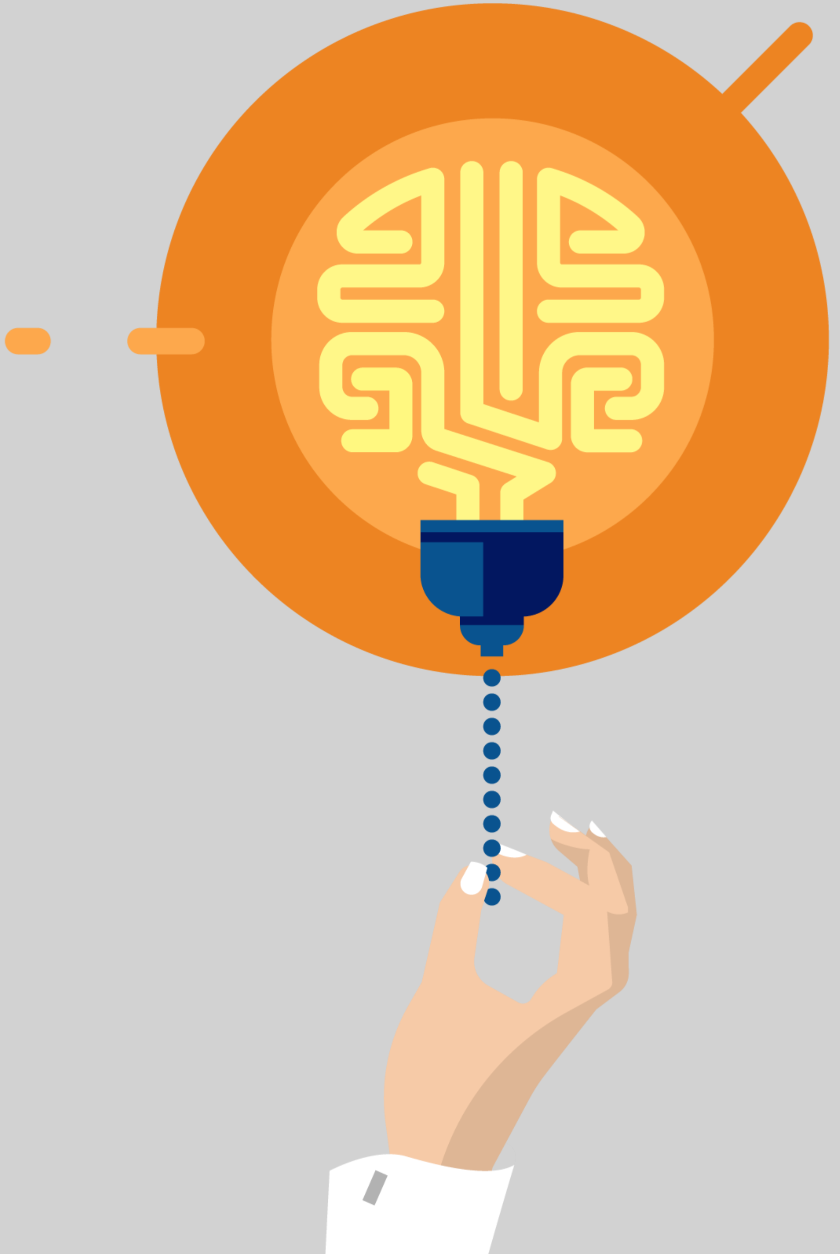
Miltos Allamanis

Microsoft Research, Cambridge

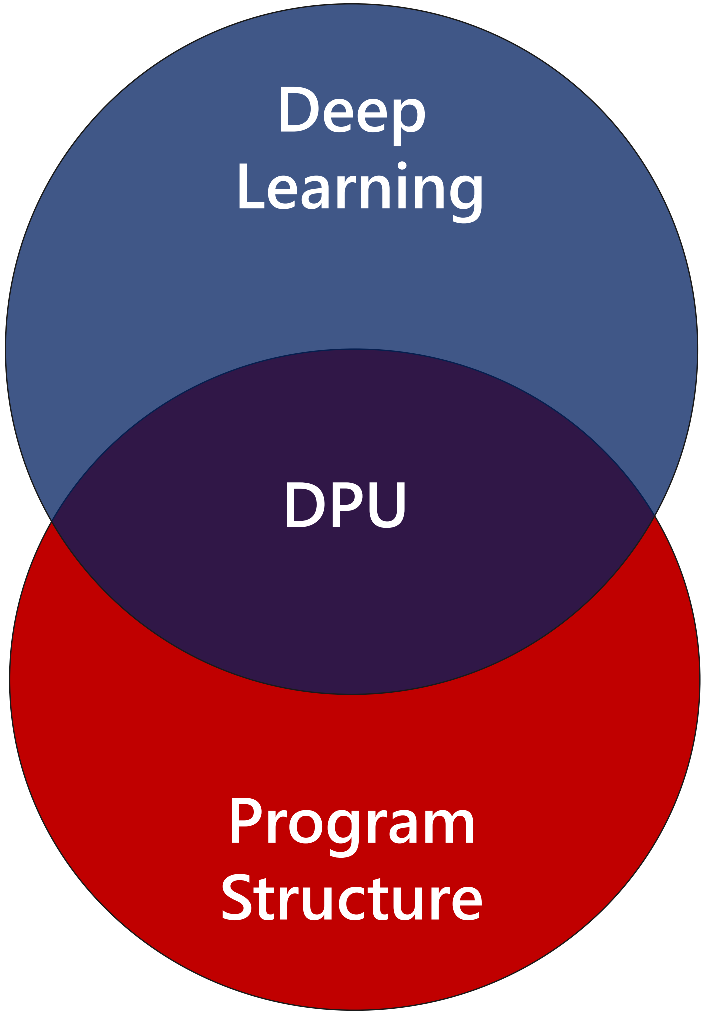
@miltos1

miltos.allamanis.com

Joint work with Marc Brockschmidt, Mahmoud Khademi, Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Santanu Dash, Earl T. Barr



All Data AI — Deep Program Understanding



- ✓ Understands images/language/speech
- ✓ Finds patterns in noisy data

- Requires many samples
- Handling structured data is hard

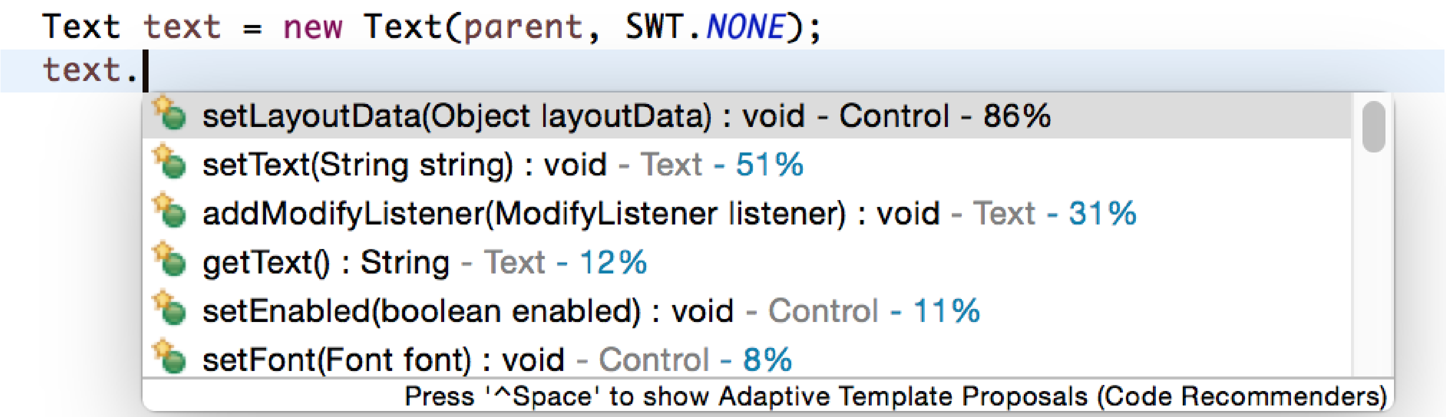
- ✓ Interpretable
- ✓ Generalisation verifiable

- Manual effort
- Limited to specialists

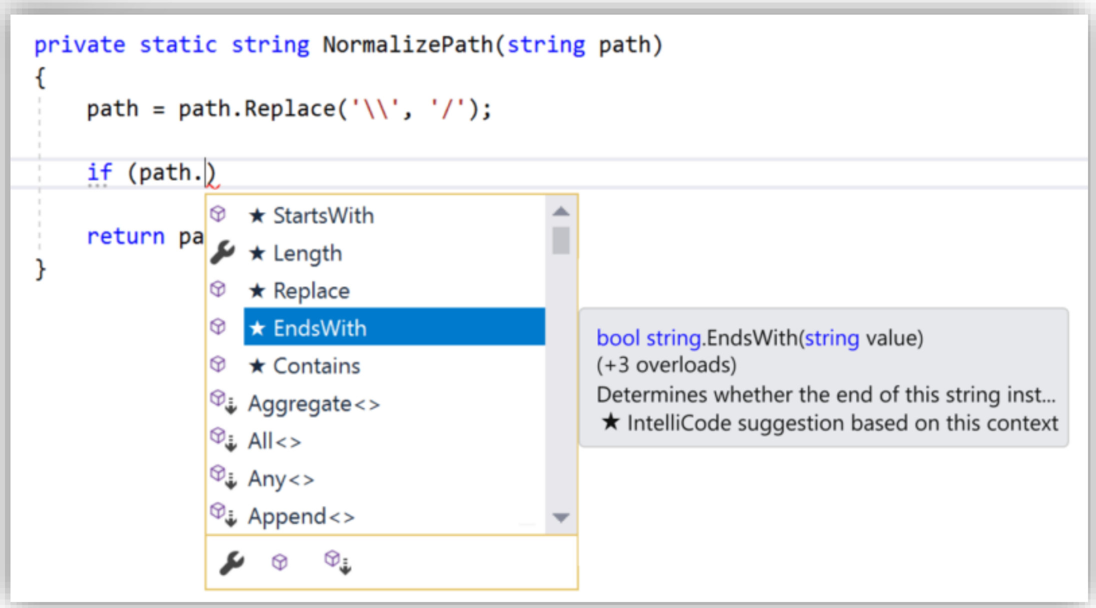
Microsoft

Source code is meant to be read by humans too.

Code Autocompletion



http://www.eclipse.org/recommenders/



https://visualstudio.microsoft.com/services/intellicode/

Argument Swapping

Declaration: `void foo(Duration responseTTLDuration, Duration frequencyCapDuration, List<A> slotResponse)`

Invocation: `foo(frequencyCapDuration, responseTTLDuration, slotResponse)`

Type	Parameter	Original argument	Correct argument
Duration	responseTTLDuration	frequencyCapDuration	responseTTLDuration
Duration	frequencyCapDuration	responseTTLDuration	frequencyCapDuration
List<A>	slotResponse	slotResponse	slotResponse

Rice *et al.* 2017 “Detecting Argument Selection Defects”

Research in ML+Code

- Infer latent intent
- Ambiguous information

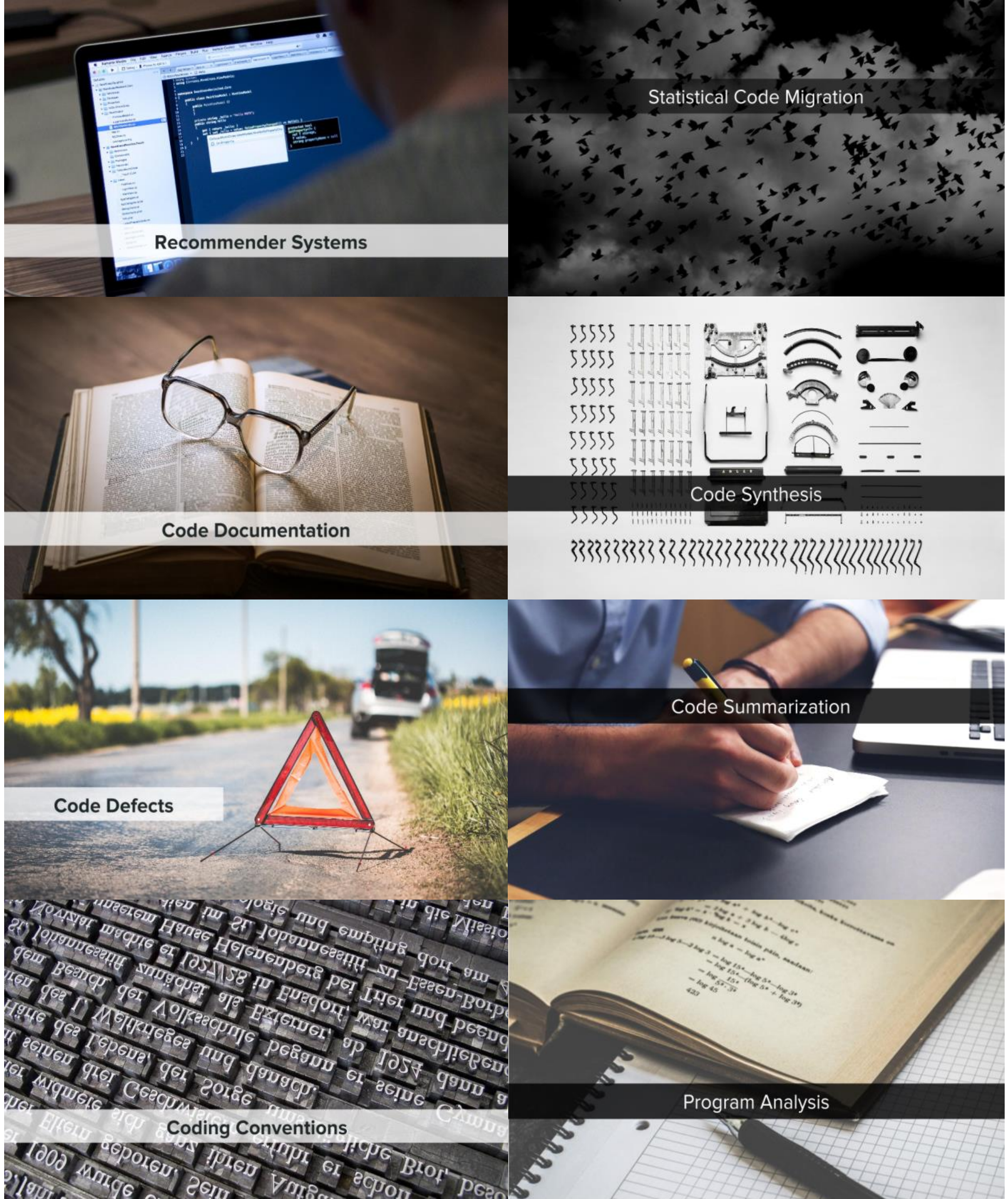
https://ml4code.github.io

1

A Survey of Machine Learning for Big Code and Naturalness

MILTIADIS ALLAMANIS, Microsoft Research
EARL T. BARR, University College London
PREMKUMAR DEVANBU, University of California, Davis
CHARLES SUTTON, University of Edinburgh and The Alan Turing Institute

Research at the intersection of machine learning, programming languages, and software engineering has recently taken important steps in proposing learnable probabilistic models of source code that exploit code's abundance of patterns. In this article, we survey this work. We contrast programming languages against natural languages and discuss how these similarities and differences drive the design of probabilistic models. We present a taxonomy based on the underlying design principles of each model and use it to navigate the literature. Then, we review how researchers have adapted these models to application areas and discuss cross-cutting and application-specific challenges and opportunities.

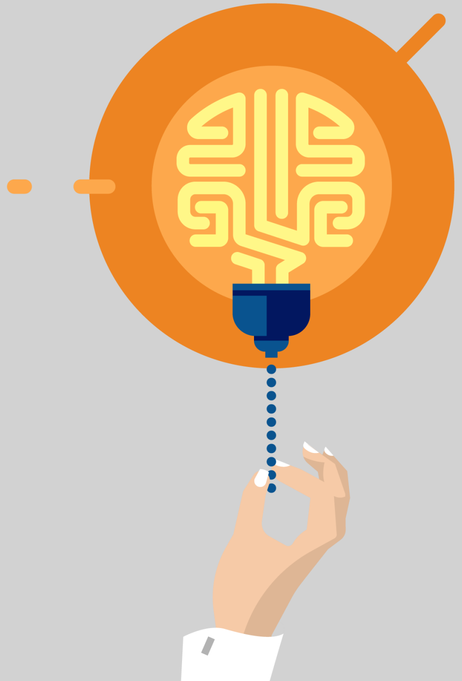


Microsoft

- Understanding Code
- Types & Machine Learning
- What's next?

“Understanding” Source Code

...with graph neural networks.



Variable Misuse Task

```
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);
```

Possible type-correct options: `clazz`, `first`

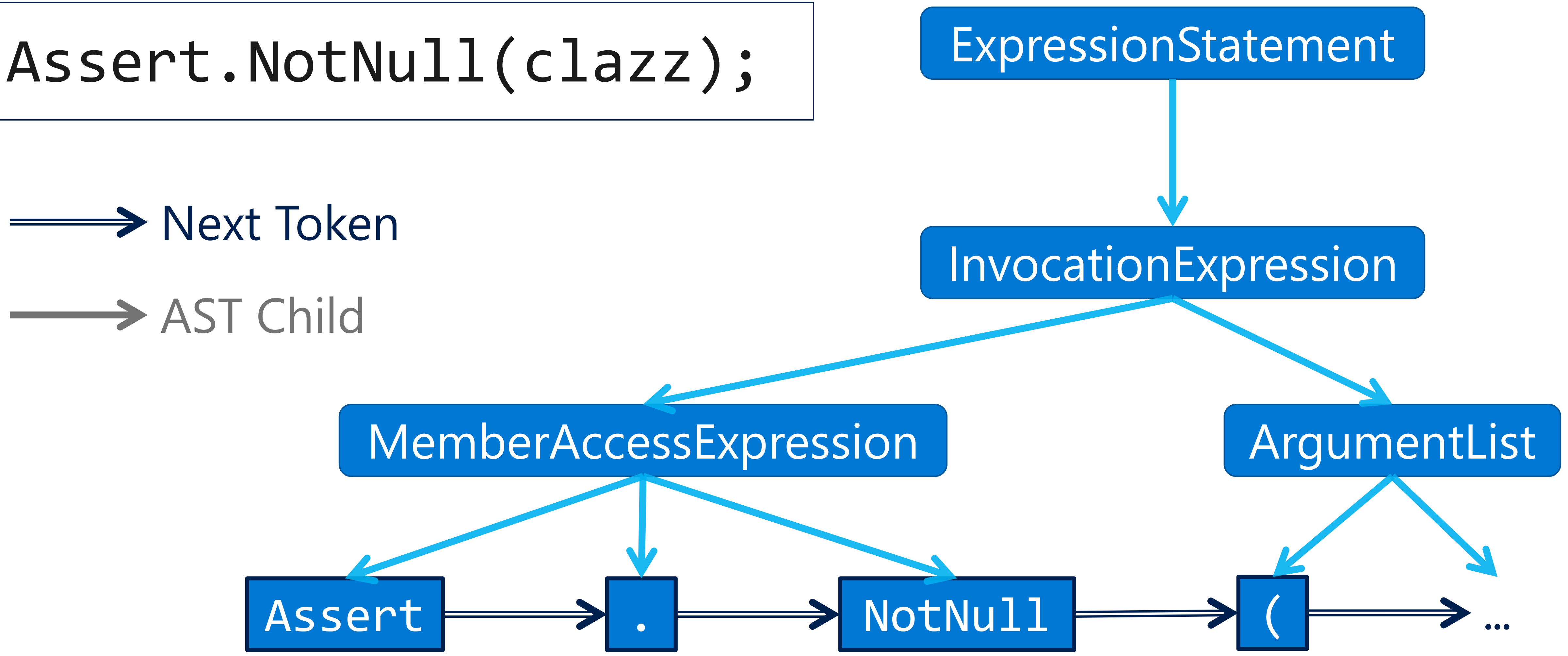
⚠ Not easy to catch with static analysis tools.

A Program Graph Representation

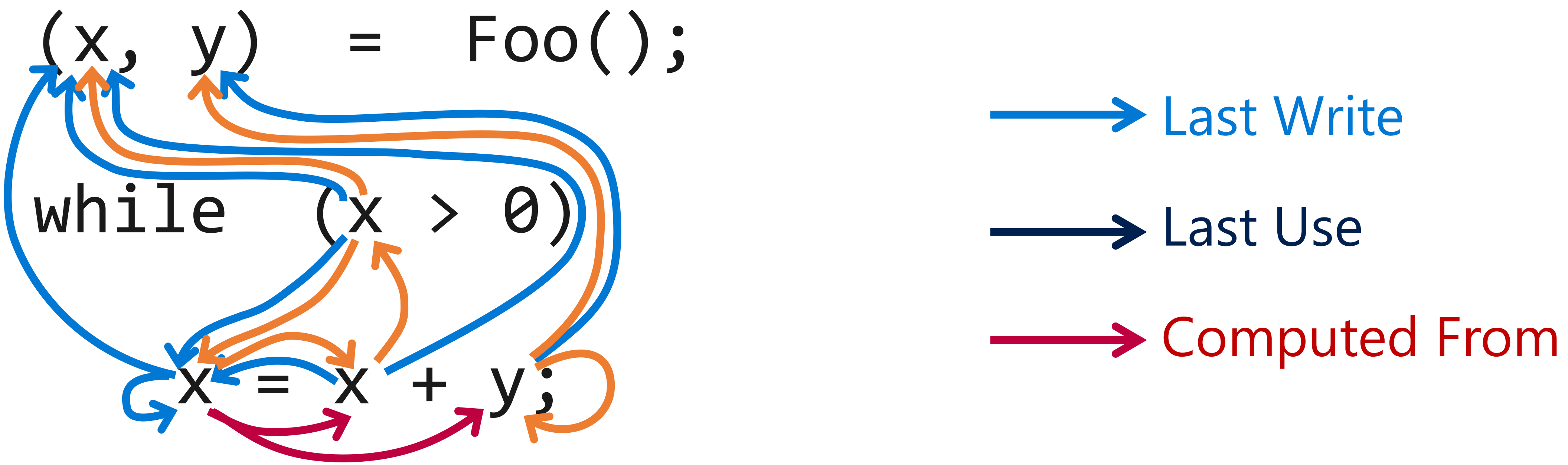
```
int SumPositive(int[] arr, int lim) {
  int sum = 0;
  for (int i = 0; i < lim; i++)
    if (arr[i] > 0)
        sum += arr[i];

  return sum;
}
```

A Program Graph Representation: Syntax

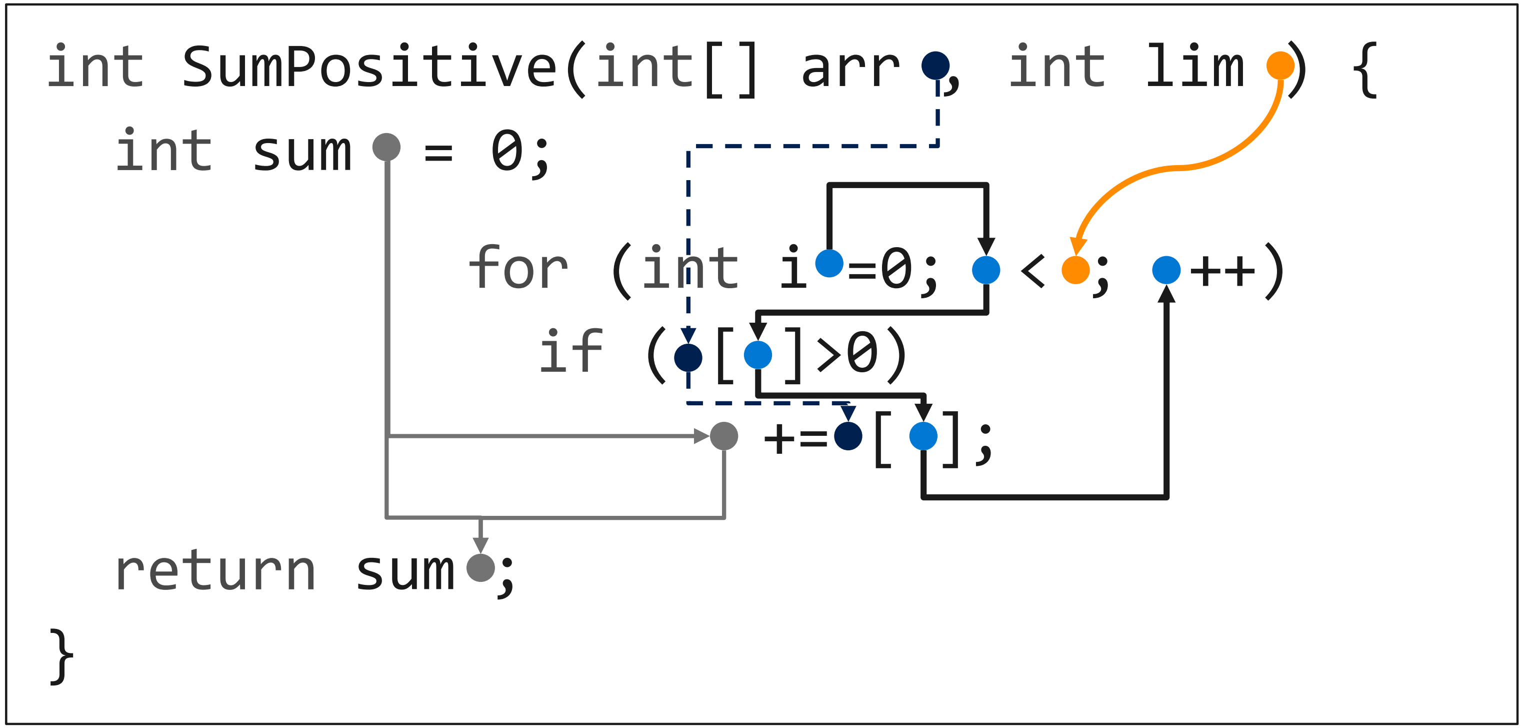
```
Assert.NotNull(clazz);
```

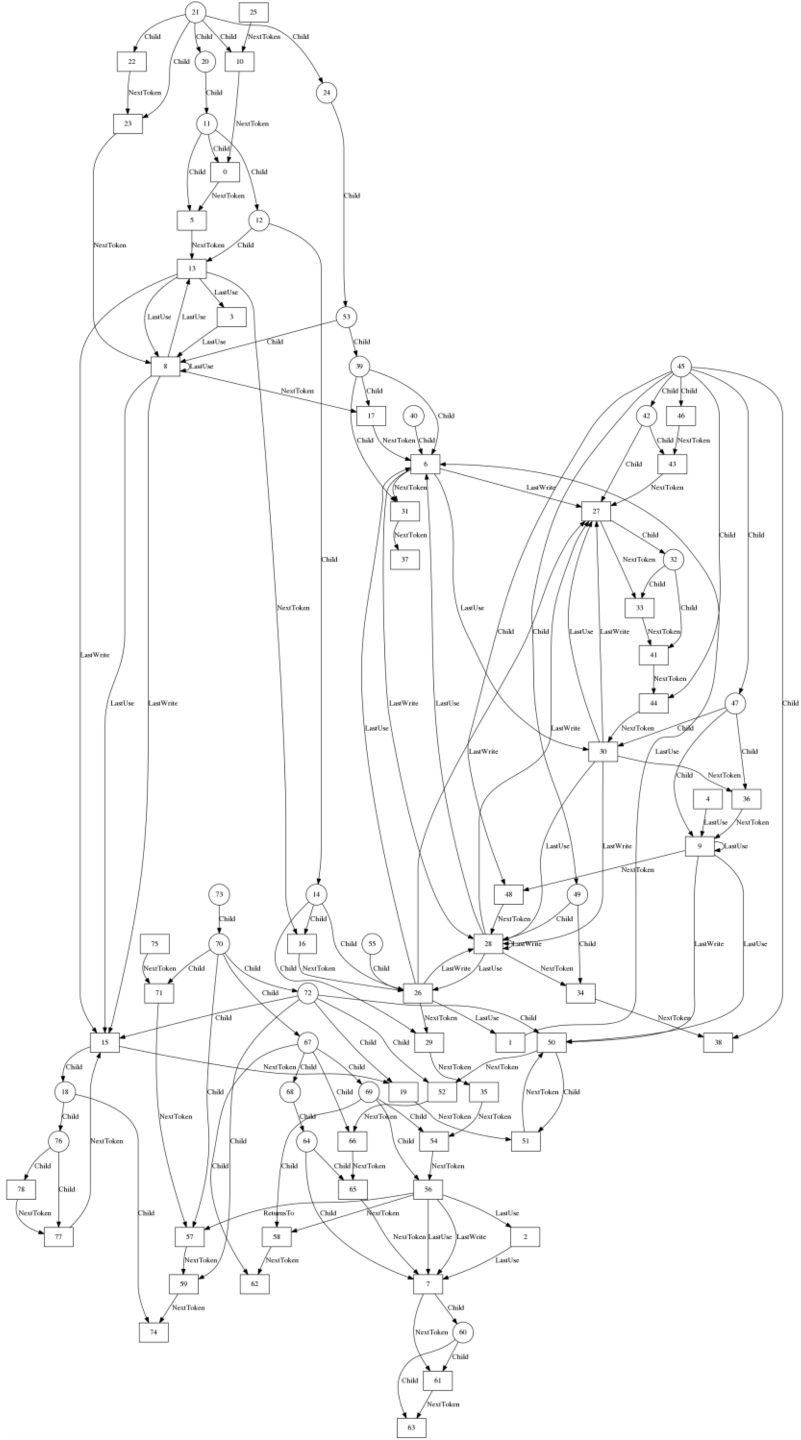


A Program Graph Representation: Data Flow



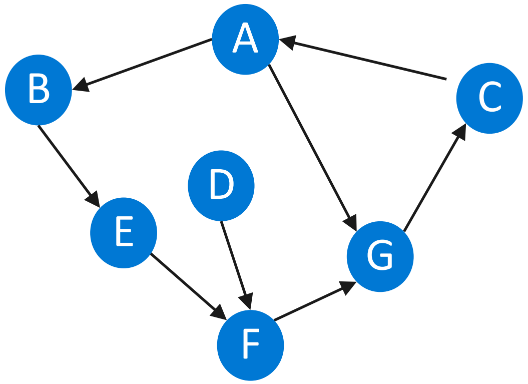
Programs as Graphs



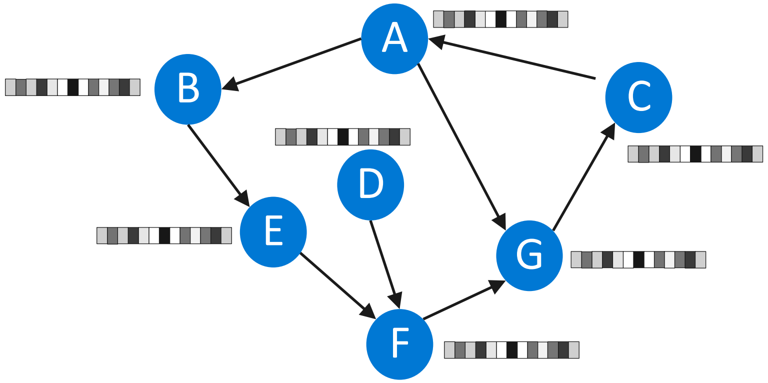


~900 nodes/graph ~8k edges/graph

Graph Neural Networks



Graph Representation
of Problem

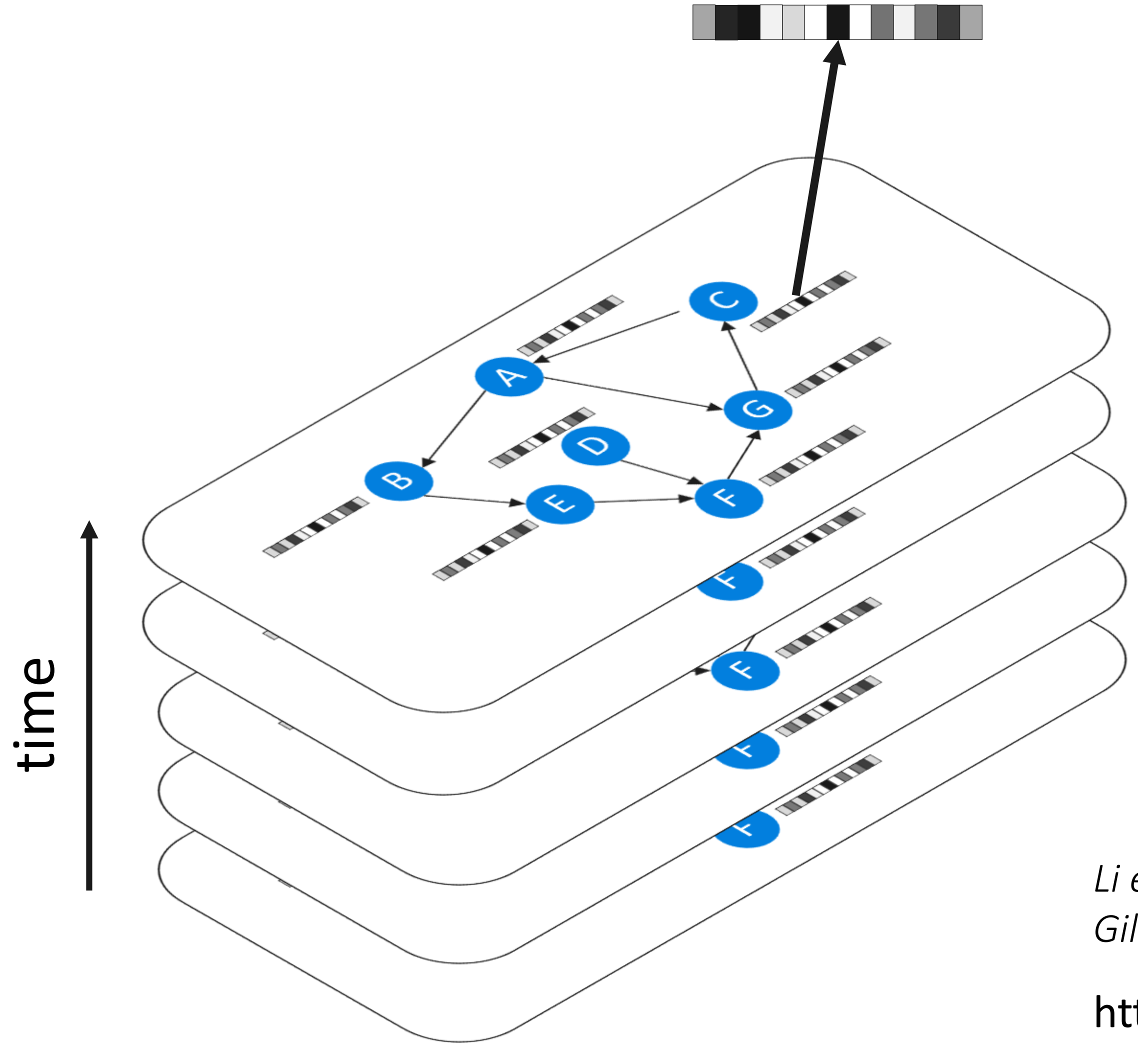


Initial Representation
of each node

Li et al (2015). Gated Graph Sequence Neural Networks.

Gilmer et al (2017). Neural Message Passing for Quantum Chemistry.

Graph Neural Networks

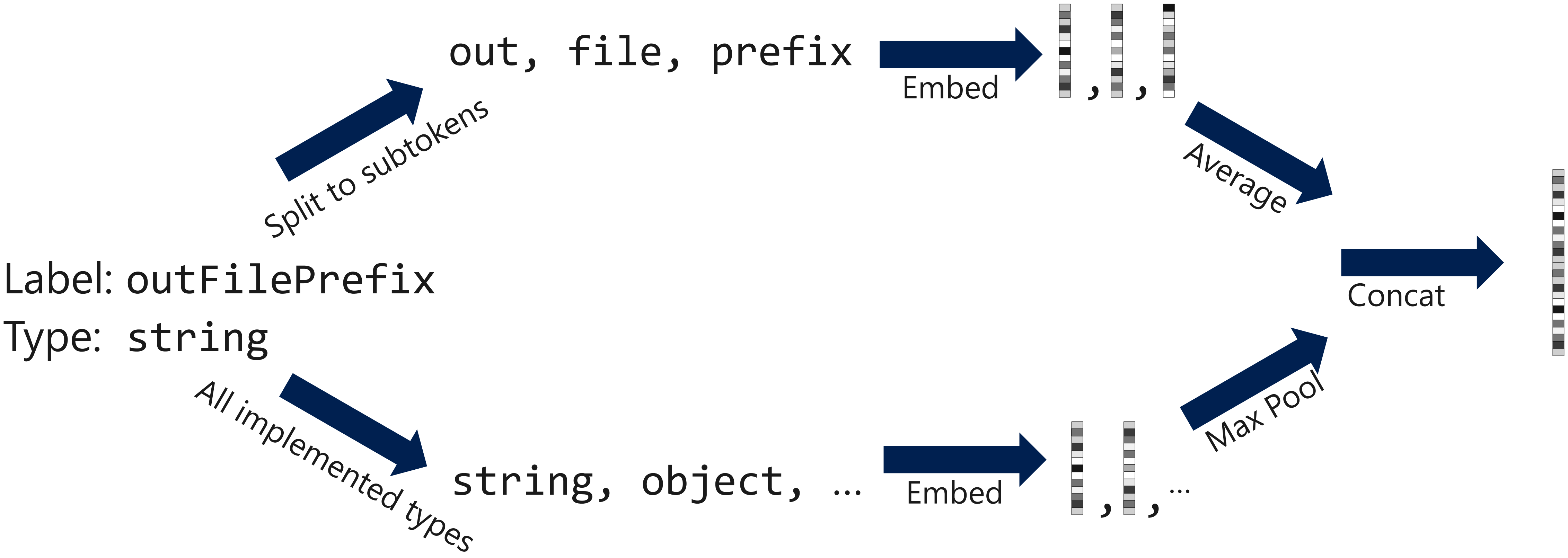


- node selection
- node classification
- graph classification

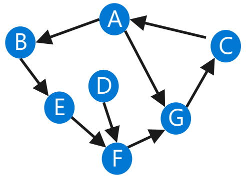
Li et al (2015). Gated Graph Sequence Neural Networks.
Gilmer et al (2017). Neural Message Passing for Quantum Chemistry.

https://github.com/microsoft/tf-gnn-samples/

Initial Node Representations



Graph Representation for Variable Misuse




```
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);
```

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

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

Quantitative Results – Variable Misuse

Accuracy (%)	BiGRU	BiGRU+Dataflow	GGNN
Seen Projects	50.0	73.7	**85.5**

Seen Projects: 24 F/OSS C# projects (2060 kLOC): Used for train and test

3.8 type-correct alternative variables per slot (median 3, $\sigma = 2.6$)

Quantitative Results – Variable Misuse

Accuracy (%)	BiGRU	BiGRU+Dataflow	GGNN
Seen Projects	50.0	73.7	**85.5**
Unseen Projects	28.9	60.2	**78.2**

Seen Projects: 24 F/OSS C# projects (2060 kLOC): Used for train and test
Unseen Projects: 3 F/OSS C# projects (228 kLOC): Used only for test
3.8 type-correct alternative variables per slot (median 3, σ= 2.6)

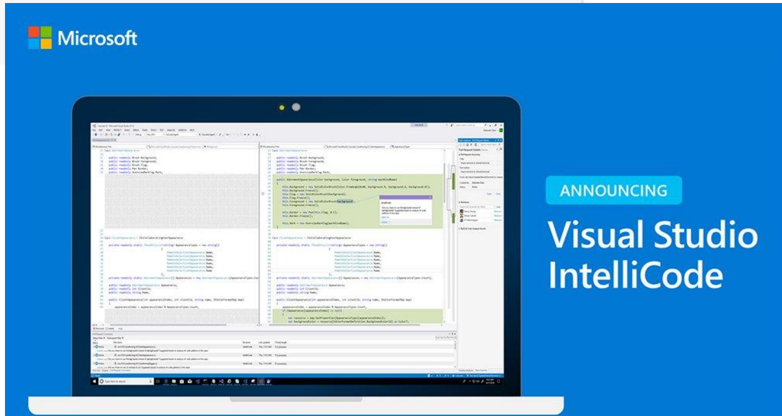
```
// Create or update the document.
var newDocument = await cosmosClient.UpsertDocumentAsync(cosmosDbCollectionUri, document);

if (updateRecord)
{
    logger.WriteLog($"Updated {existingDocument} to {newDocument}");
}
else
{
    logger.WriteLog($"Added {existingDocument}");
```

[redacted] Update 1 1 Resolved

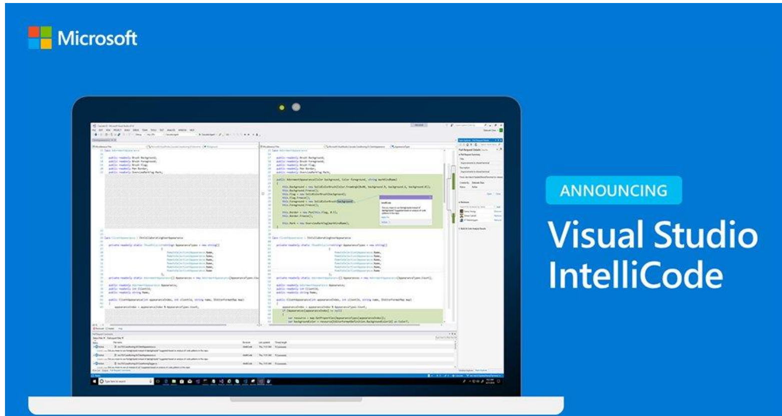
Based on this repo's code patterns, did you intend to use 'newDocument' (confidence 92%) rather than 'existingDocument` (confidence 7%) here? Review is recommended by Research bot's Variable Misuse analysis.

JK [redacted]

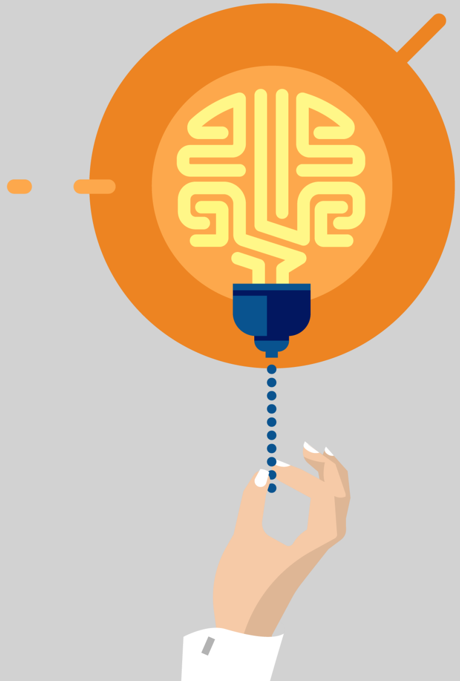
+1



```
var x = ComputeX()
if (some-condition-without-x) {
    UseX(x)
} else {
    UseOtherVars(y)
}
// x not used after this point
```

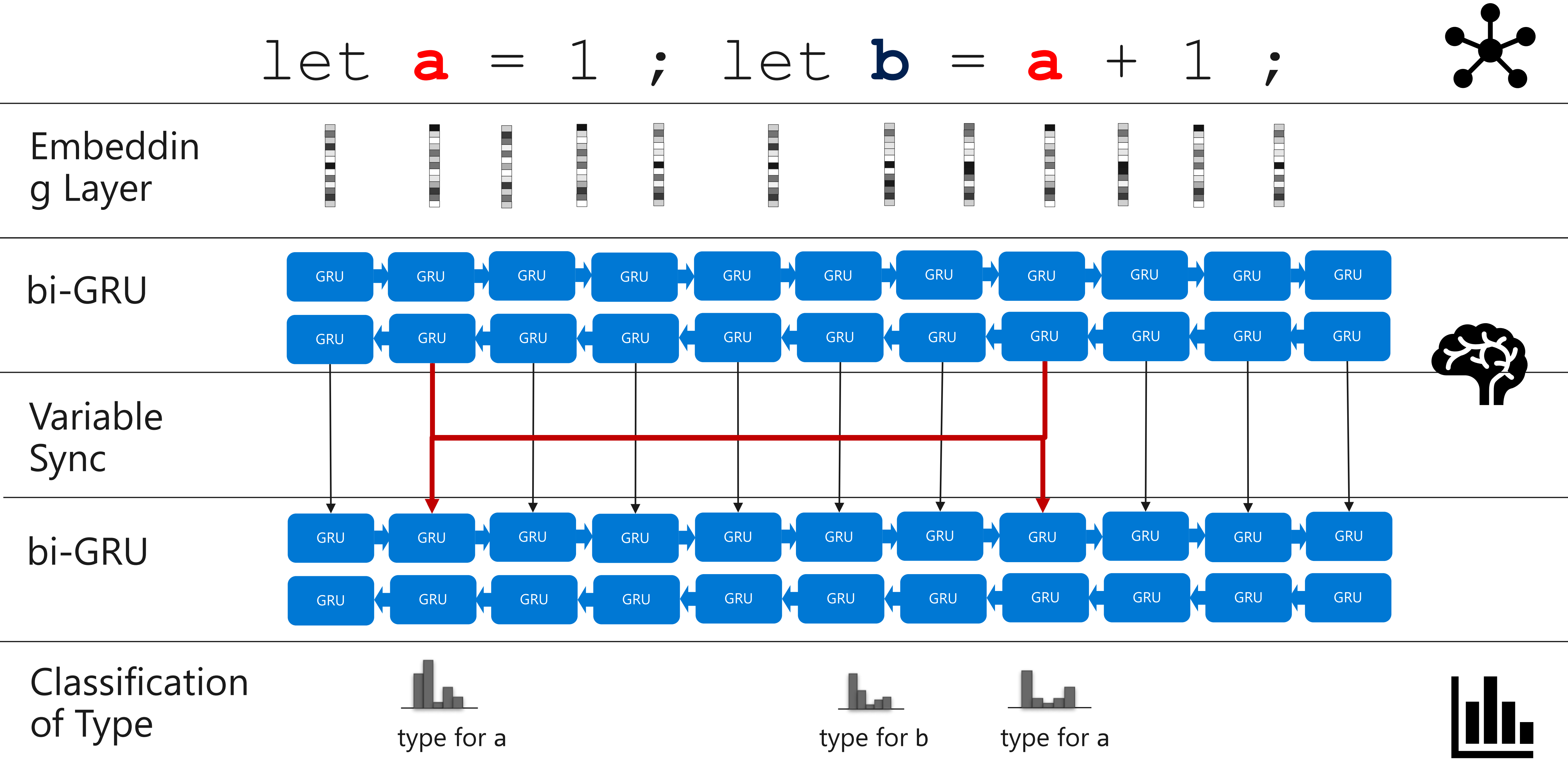


Types & Machine Learning

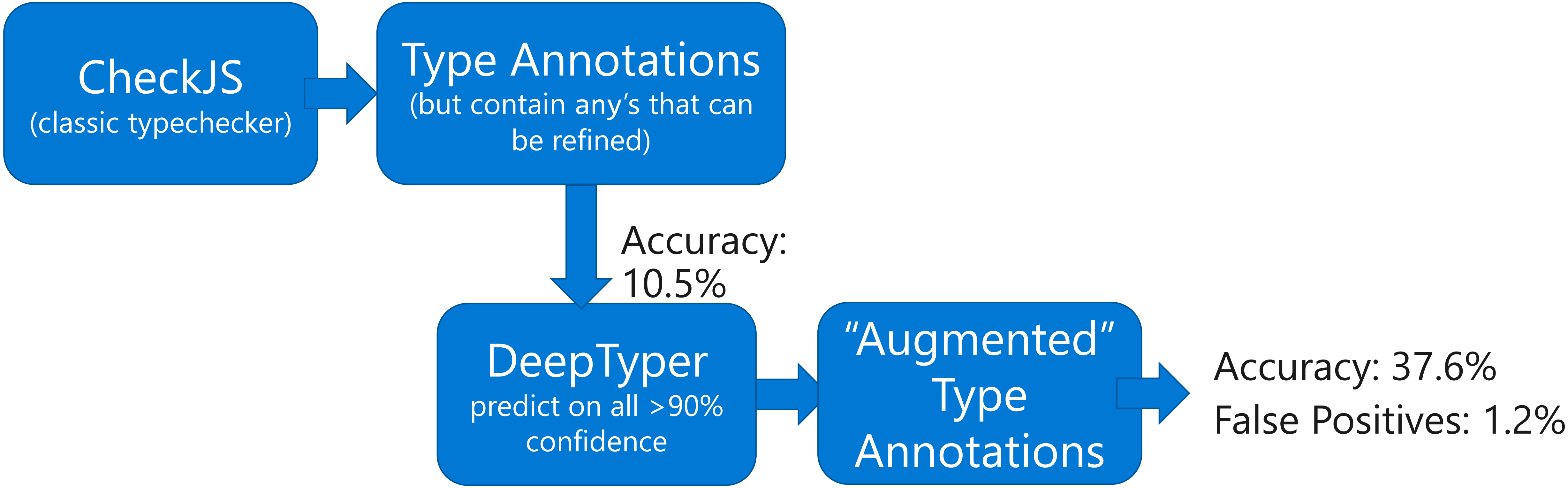


DeepTyper

Hellendoorn, Barr, Bird, Allamanis, 2018



Combining DeepTyper with CheckJS



Hellendoorn, Barr, Bird, Allamanis, FSE 2018

Conceptual Types

Defined Types

"a password" → `string password;`

"a JSON string" → `string data = Json.Load();`

Latent; we don't observe in the *conceptual* types.

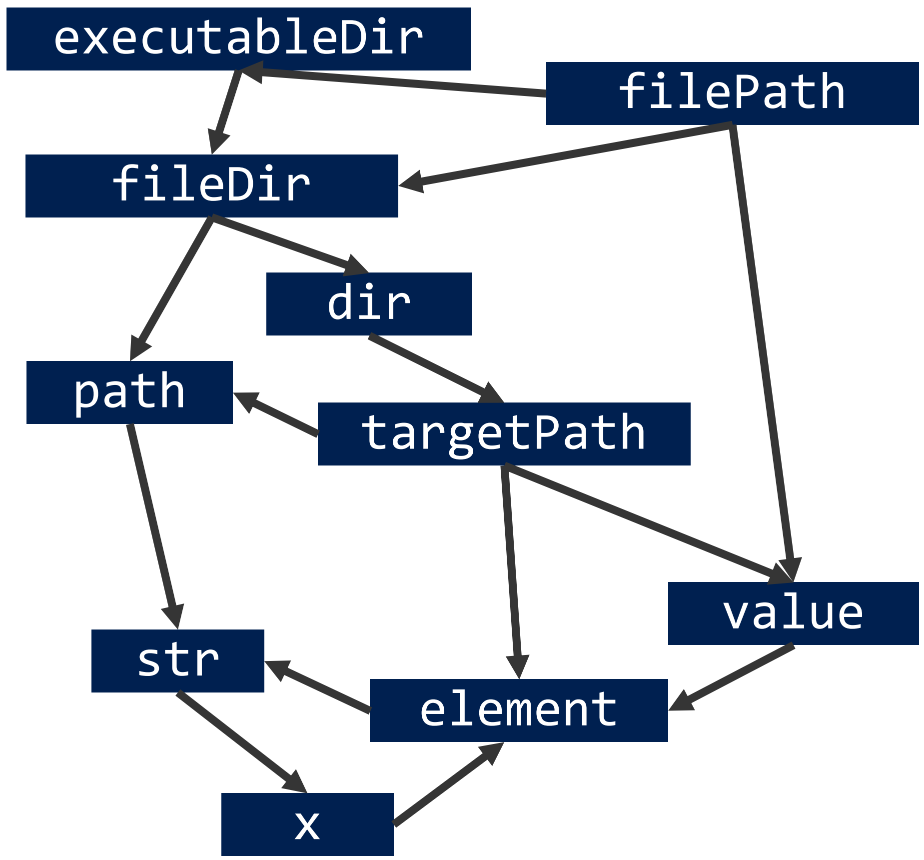
Defined explicitly by the programmer.

RefiNym: Nominal Refinements

```
EncryptedString                    UnencrytedString
string EncryptAndSignCookie(string cookieValue, FormsAuthenticationConfiguration config) {
      EncryptedString
      string encryptedCookie =
             config.CryptographyConfiguration.EncryptionProvider.Encrypt(cookieValue);

      var hmacBytes = GenerateHmac(encryptedCookie, config);
      string hmacString = Convert.ToBase64String(hmacBytes);
     EncryptedString

      return hmacString + encryptedCookie;
}
```

Dash, Allamanis, Barr. 2018

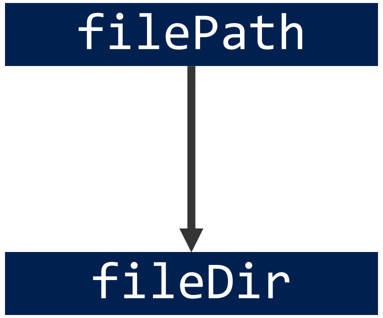
Name Flow Graphs



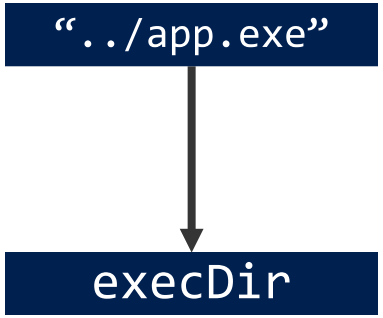
Representation:

- [Static] Data Flow
- Identifier Names

Constructing Name Flows – Assignment

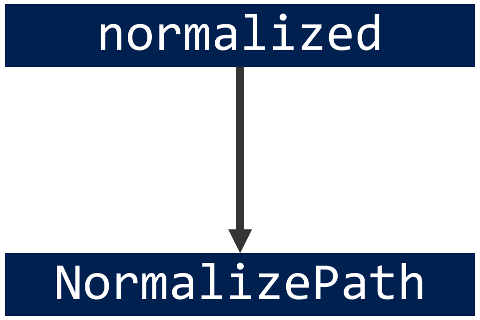



```
fileDir = filePath
```



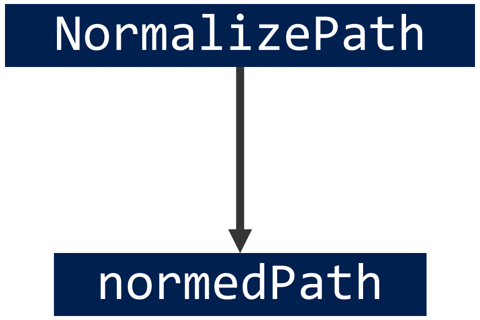

```
execDir = “../app.exe”
```

Constructing Name Flows – Returns



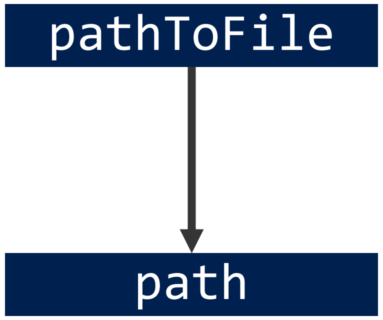

```
def NormalizePath(…){
    …

    return normalized;
}
```

Constructing Name Flows – Function Calls



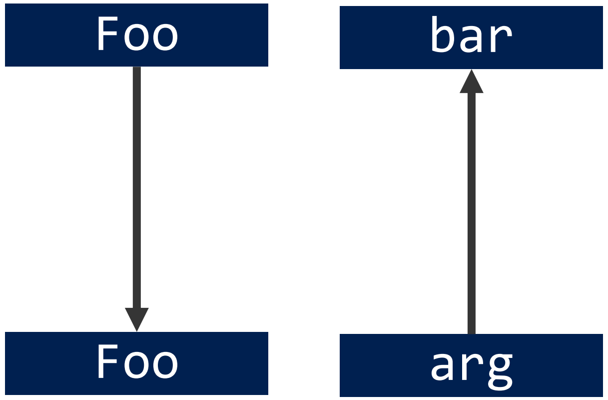

```
normedPath = NormalizePath(…)
```

Constructing Name Flows – Actuals to Formals




```
Exists(pathToFile)


def Exists(string path) { … }
```

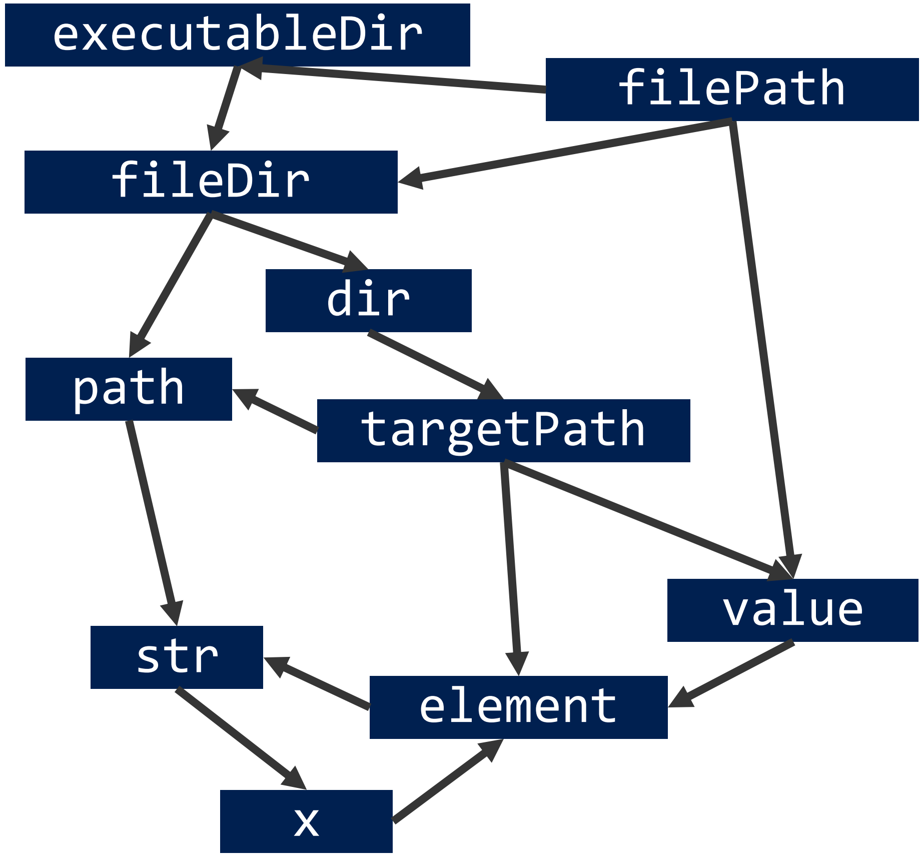
Constructing Name Flows – Override




```
def override string Foo(string bar) { … }
```

```
def string Foo(string arg) { … }
```

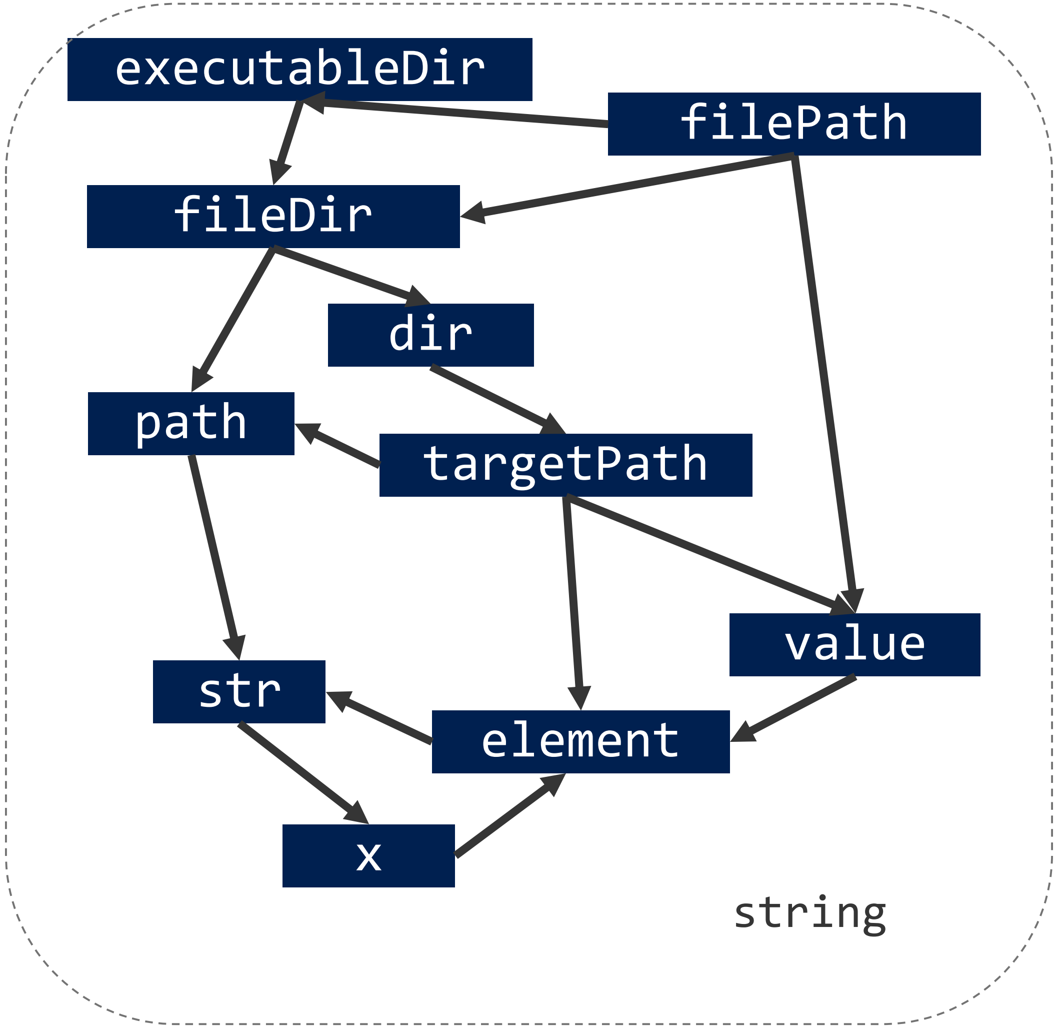
Constructing Name Flows – Summary



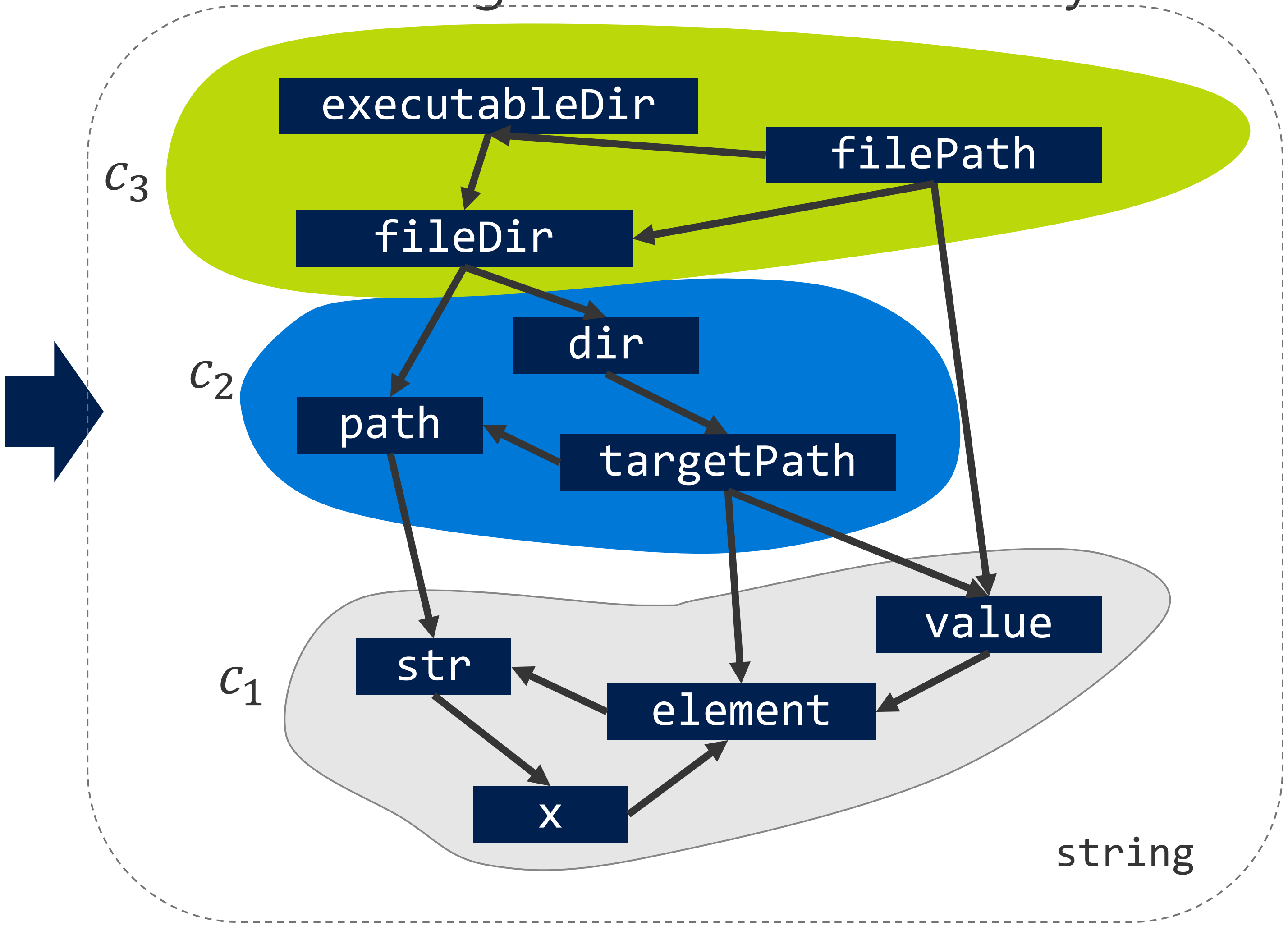
- Capture type-correct flows
- Capture names of variables/methods

From Dataflow to Nominal Type Refinements

through information theory…



Dataflow of `string` variables/method and their names.



Nominal `string` refinements.

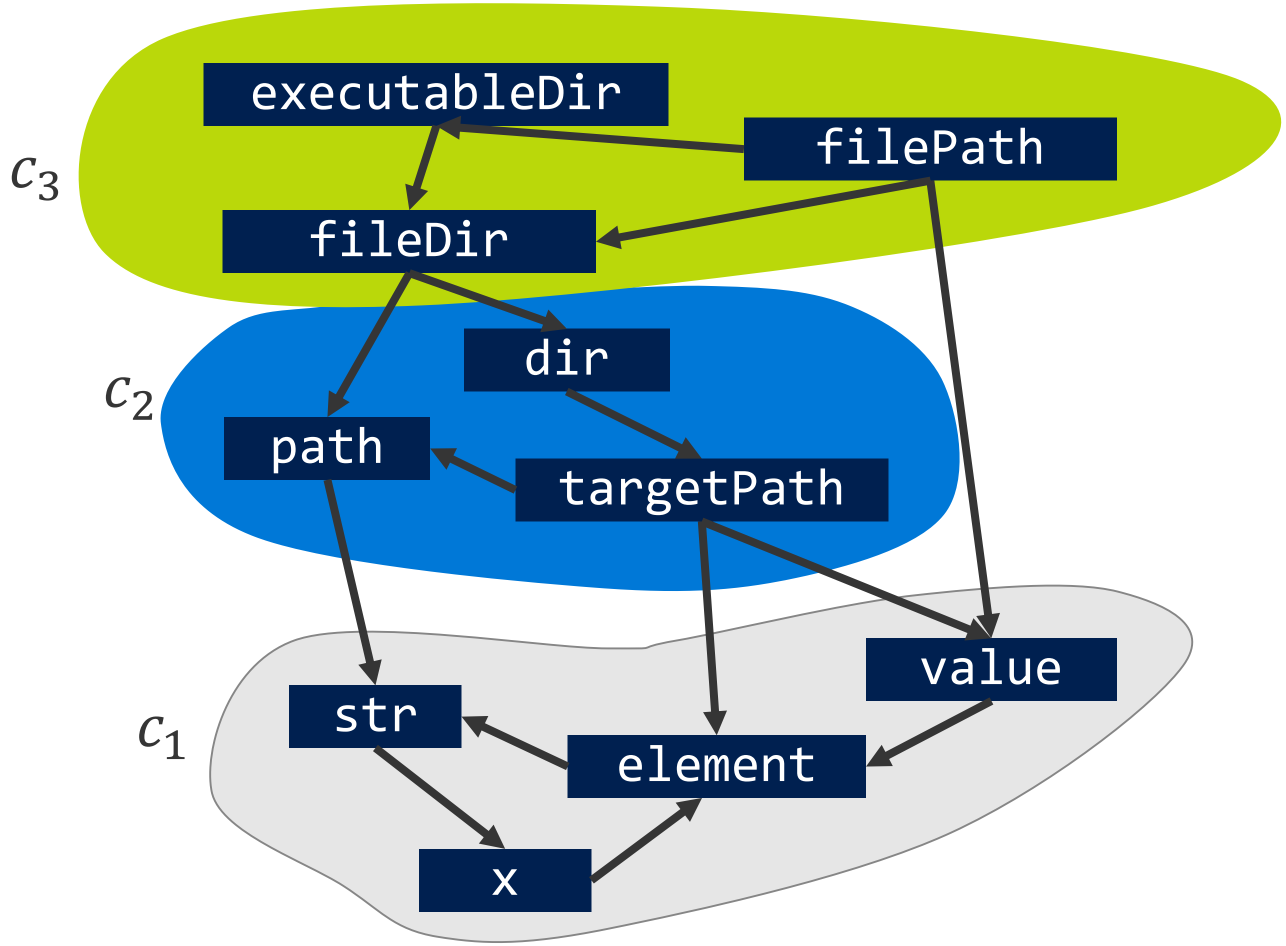
Information-Theoretic Clustering

We want both

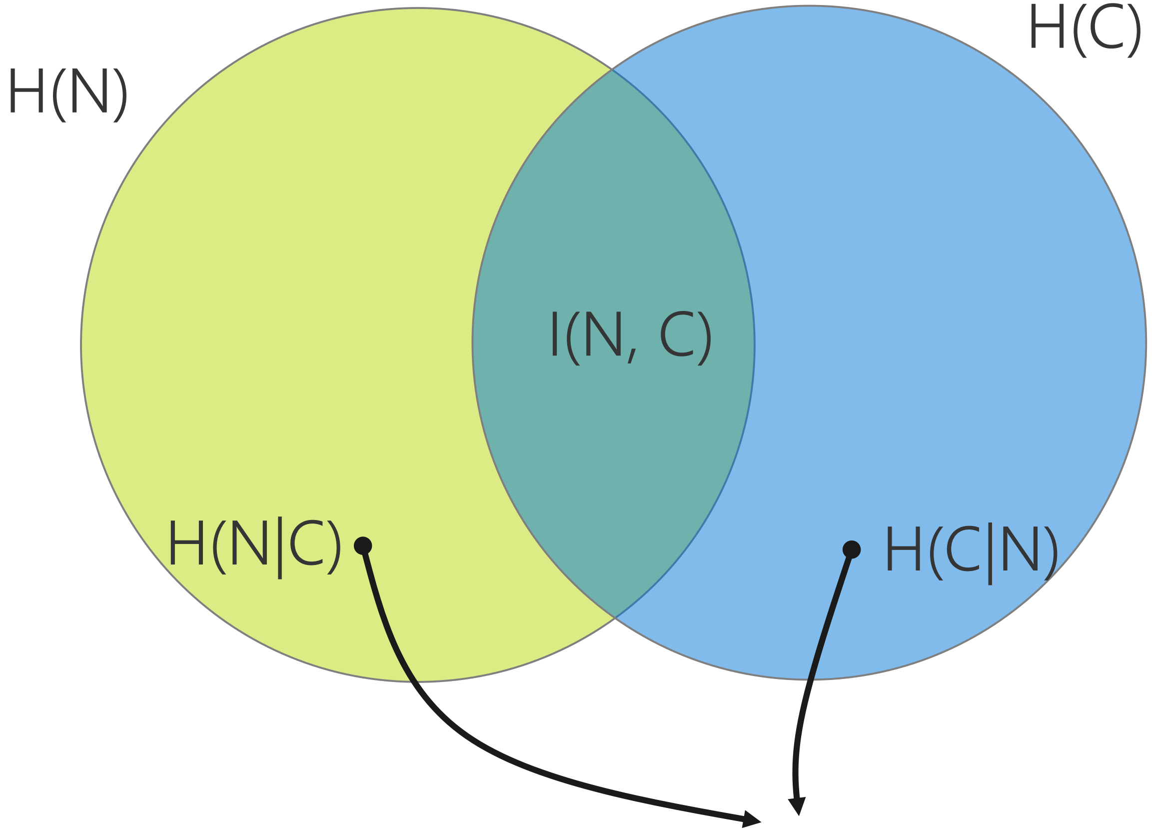
$P(\text{cluster}|\text{name})$

$P(\text{name}|\text{cluster})$

to have low entropy

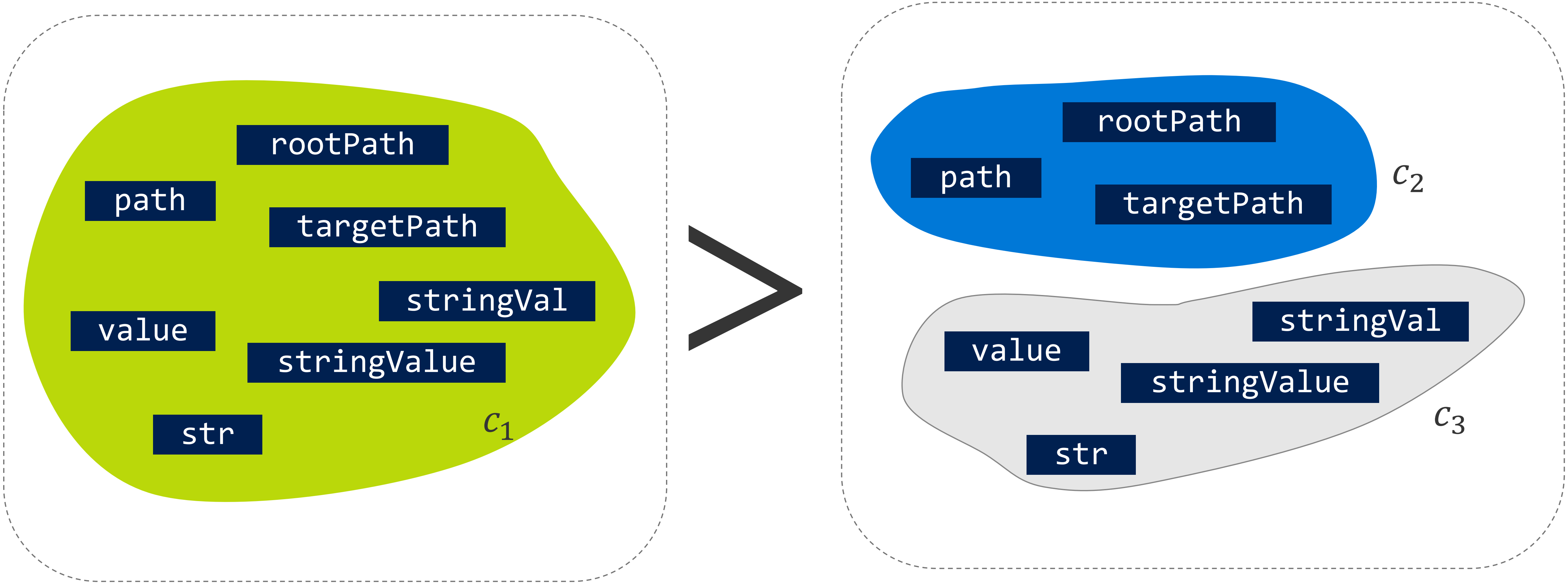


Variation of Information: Objective



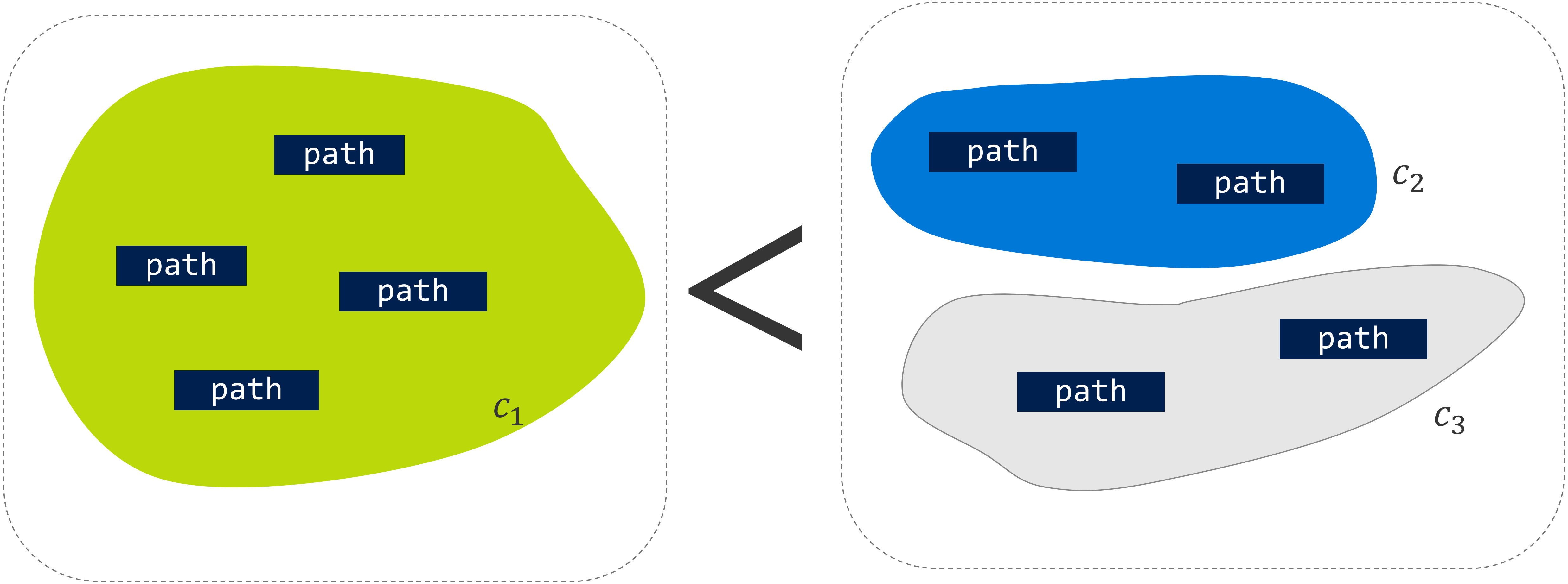
$$C^* = argmin_C VI(N, C) \; s.t. \exists R = (C, \subseteq)$$

Variation of Information: Intuition



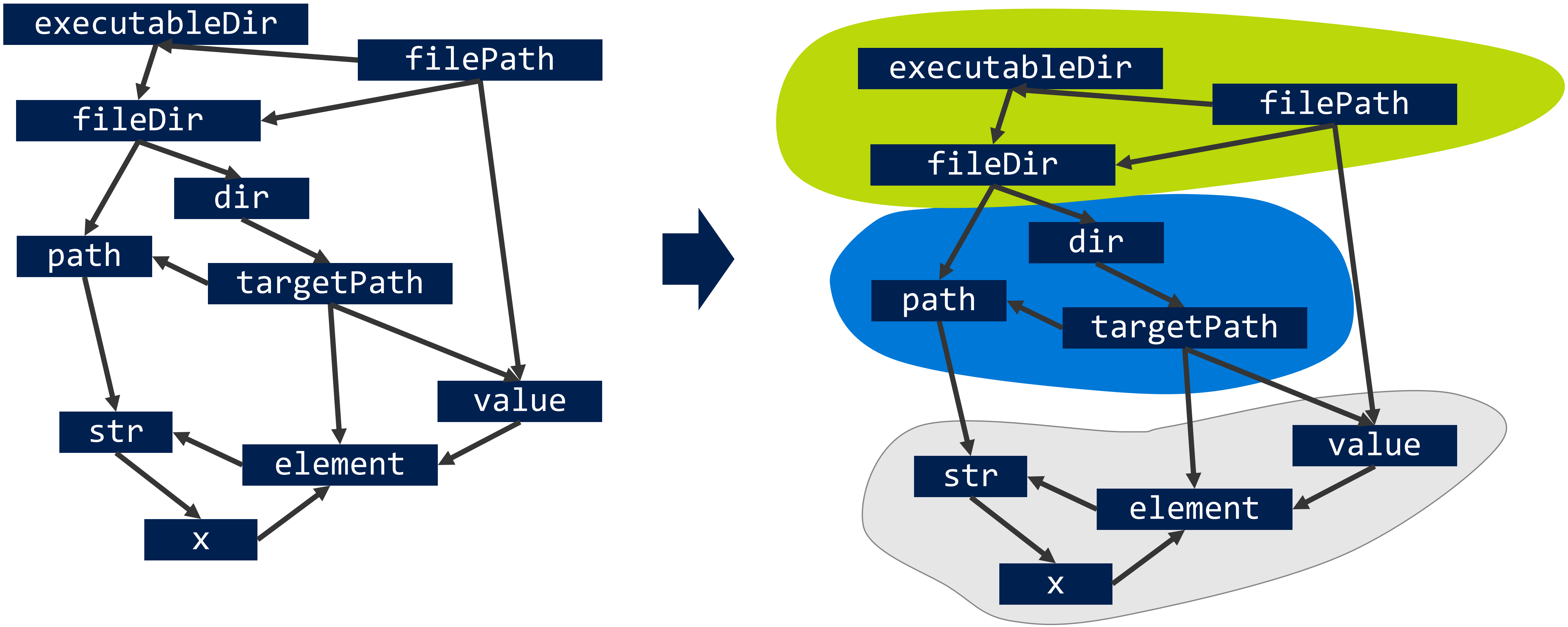
$$VI(N,\{c_1\}) > VI(N,\{c_2,c_3\})$$

Variation of Information: Intuition



$$VI(N,\{c_1\}) < VI(N,\{c_2,c_3\})$$

From Dataflow to Nominal Type Refinements

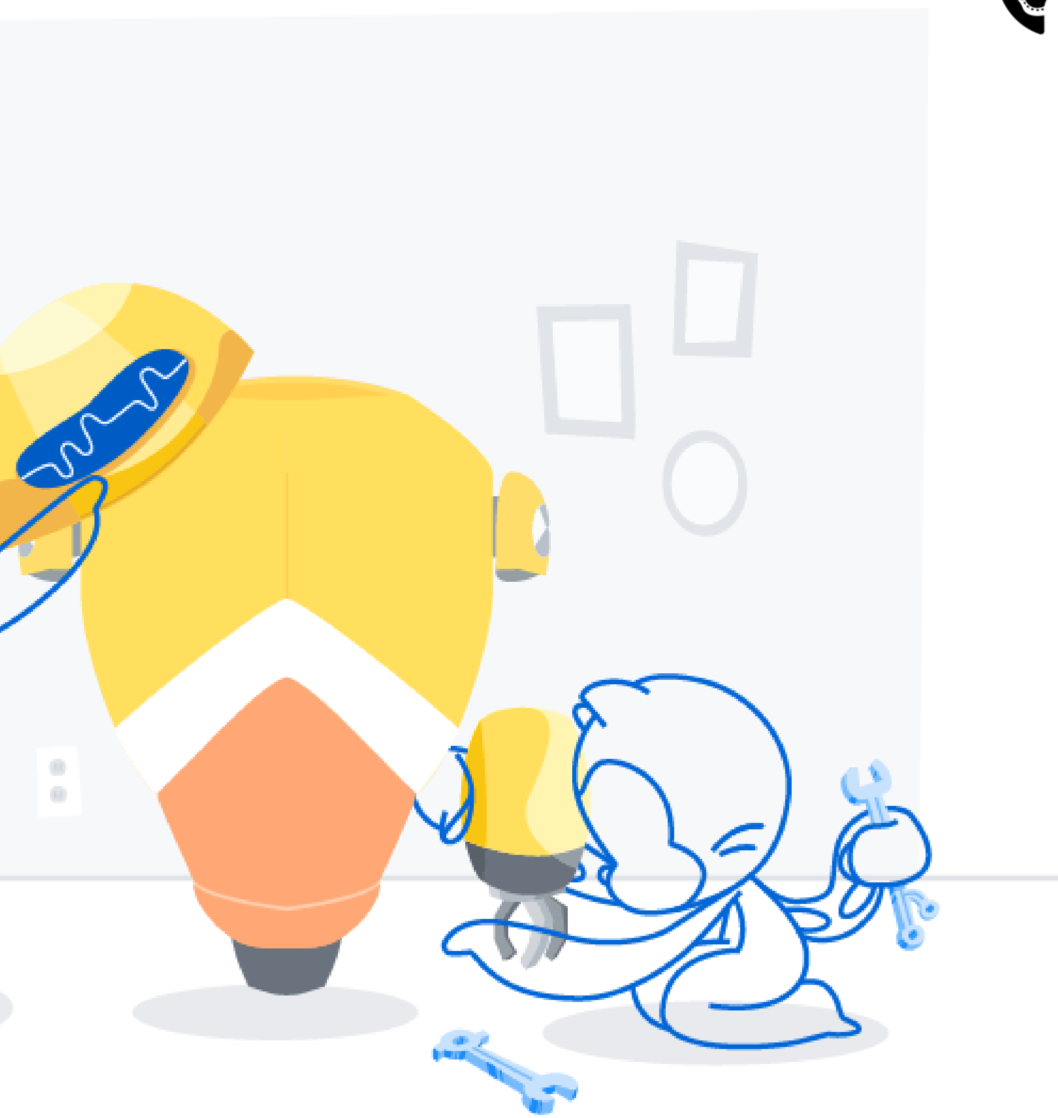


$$C^* = argmin_C VI(N, C) \ \ s.t. \exists R = (C, \subseteq)$$

	Full name of nodes or constant values
1	path, Path, originalRequestPath, modifiedRequestPath, owinRequestPath, "/ emailConstraint/", contentPath, basePath, IViewEngineHost::ExpandPath, AspNetRootPathProvider::GetRootPath, "/", DiagnosticsConfiguration::GetNormalizedPath, NancyContext::ToFullPath, ModulePath
2	DefaultCulture, defaultCulture, cookieCulture, cultureLetters, name
3	earlyExitReason, "Requires Any Claim", "Requires Claims", "Requires Authentication", "Accept"
4	IObjectSerializer::Serialize, DefaultObjectSerializer::Serialize, JsonObject::ToString, SimpleJson::SerializeObject, HttpQsCollection::toString
5	method, Method, "PUT", "POST", "PATCH", "OPTIONS", "HEAD", "GET", "DELETE"
6	value, cookieValue, sourceString, "SomeValue", cookieValueEncrypted, attemptedValue, decryptedValue, defaultValue
7	password, "password", realPassword, plainText, Password
8	HttpUtility::UrlDecode, HttpUtility::UrlPathEncode, HttpUtility::UrlEncodeUnicode, redirectUrl, fallbackRedirectUrl, url, path
9	header, "Accept-Language", "Accept-Encoding", "Accept-Charset", "X-Custom", "Content-Disposition", "Vary", "Etag"

Full name of node or constant value in bepuphysics
damping, SuspensionDamping, starchDamping, dampingConstant, angularDamping, LinearDamping
currentDistance, distance3, candidateDistance, pointDistance, distanceFromMaximum, grabDistance, VariableLinearSpeedCurve::GetDistance, tempDistance
goalVelocity, driveSpeed, GoalSpeed
minRadius, MinimumRadius, Radius, minimumRadiusA, WrappedShape::ComputeMinimumRadius, topRadius, MaximumRadius, graphicalRadius, TransformableShape::ComputeMaximumRadius
blendedCoefficient, KineticFriction, dynamicCoefficient,KineticBreakingFrictionCoefficient
angle, myMaximumAngle, MinimumAngle, currentAngle, MaximumAngle, steeringAngle, MathHelper::WrapAngle
targetHeight, Height, ProneHeight, crouchingHeight, standingHeight
Mass, effectiveMass, newMassA, newMass
M22, m11, M44, resultM44, M43, intermediate, m31, X, Y, Z


Microsoft
GitHub

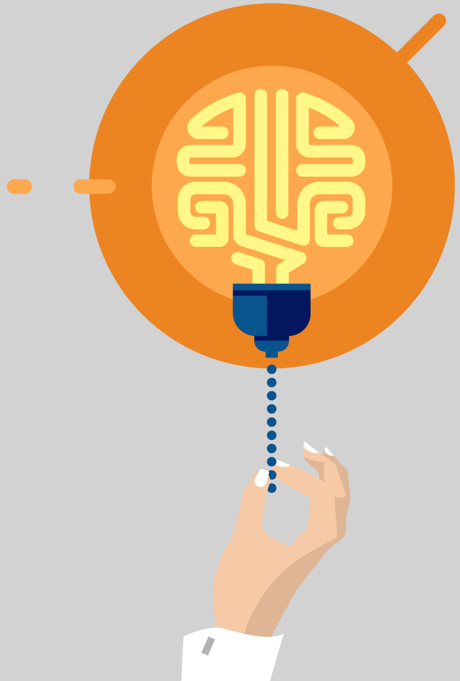


CodeSearchNet

- A corpus of 6 millions functions with metadata.
- A small human-annotated set of relevance annotations.
- A semantic code search challenge.

https://github.com/github/CodeSearchNet

Closing Thoughts



Machine Learning
Learn Patterns from Rich Structure

- Capture human elements within code
- Make probabilistic predictions

Formal Methods
Reasoning

- Capture formal constraints
- Exact reasoning

Photo by Luke Stackpoole

Variable Misuse Task

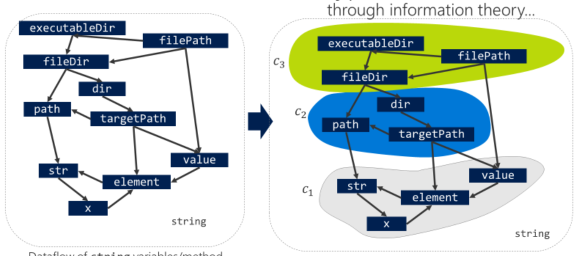
```
var clazz=classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

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

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

Possible type-correct options: `clazz`, `first`

⚠ Not easy to catch with static analysis tools.

From Dataflow to Nominal Type Refinements

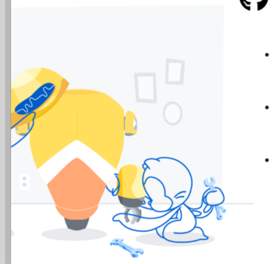
through information theory...



Dataflow of `string` variables/method and their names.

Nominal `string` refinements.

CodeSearchNet



- A corpus of 6 millions functions with metadata.
- A small human-annotated set of relevance annotations.
- A semantic code search challenge.

https://github.com/github/CodeSearchNet

Machine Learning
Learn Patterns from Rich Structure

- Capture human elements within code
- Make probabilistic predictions

Formal Methods
Reasoning

- Capture formal constraints
- Exact reasoning

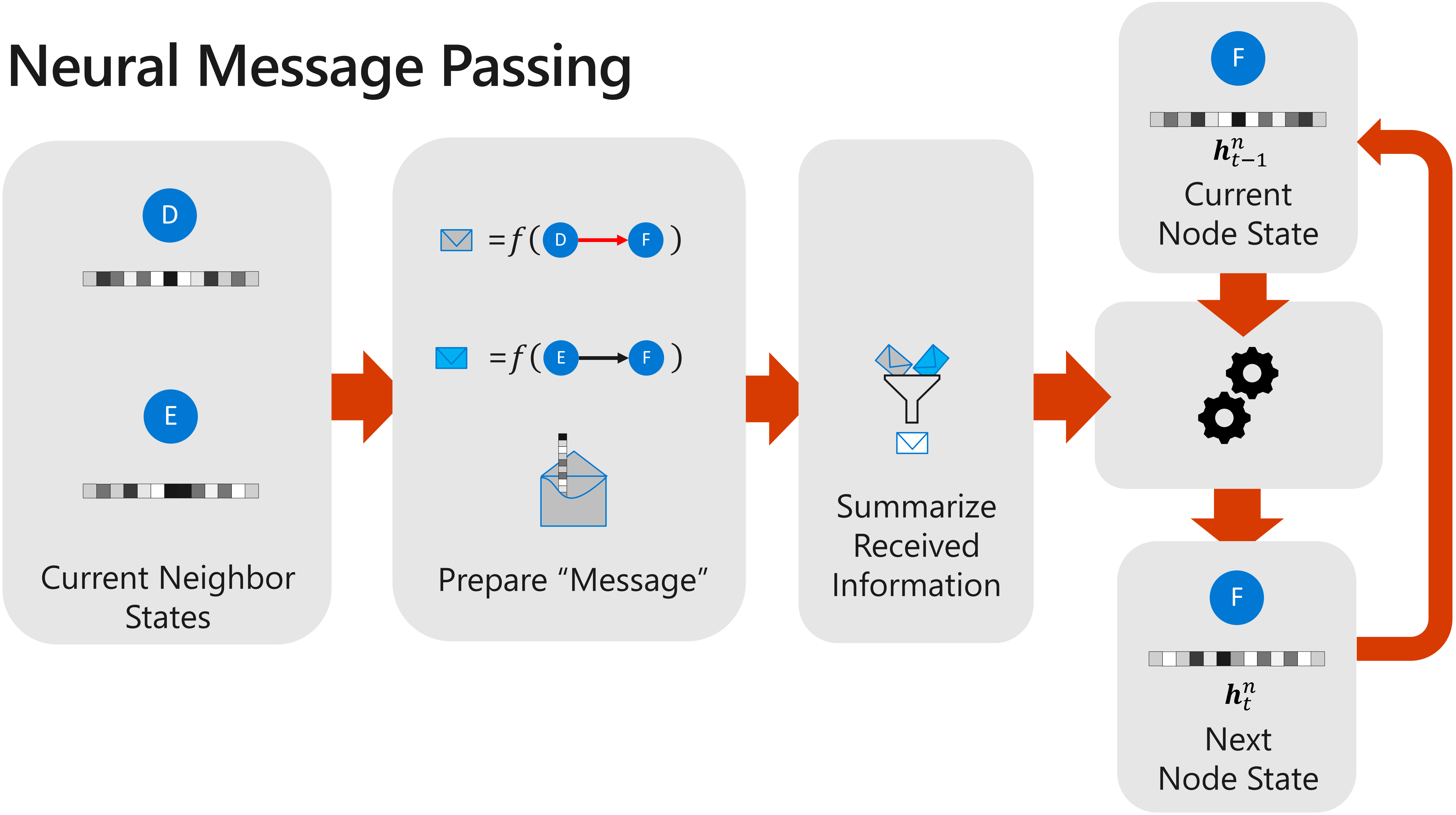
@miltos1

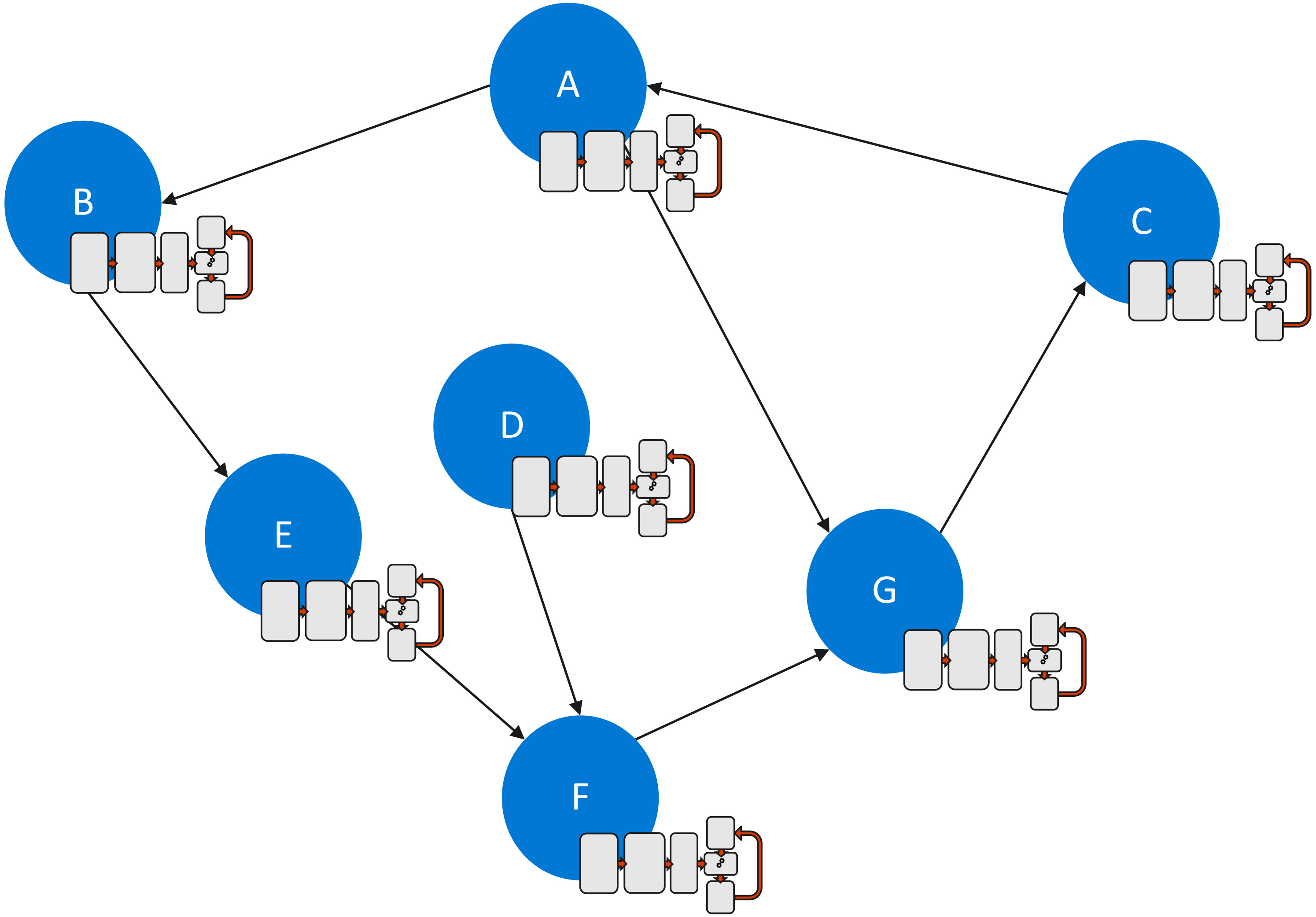
miltos.allamanis.com

Microsoft

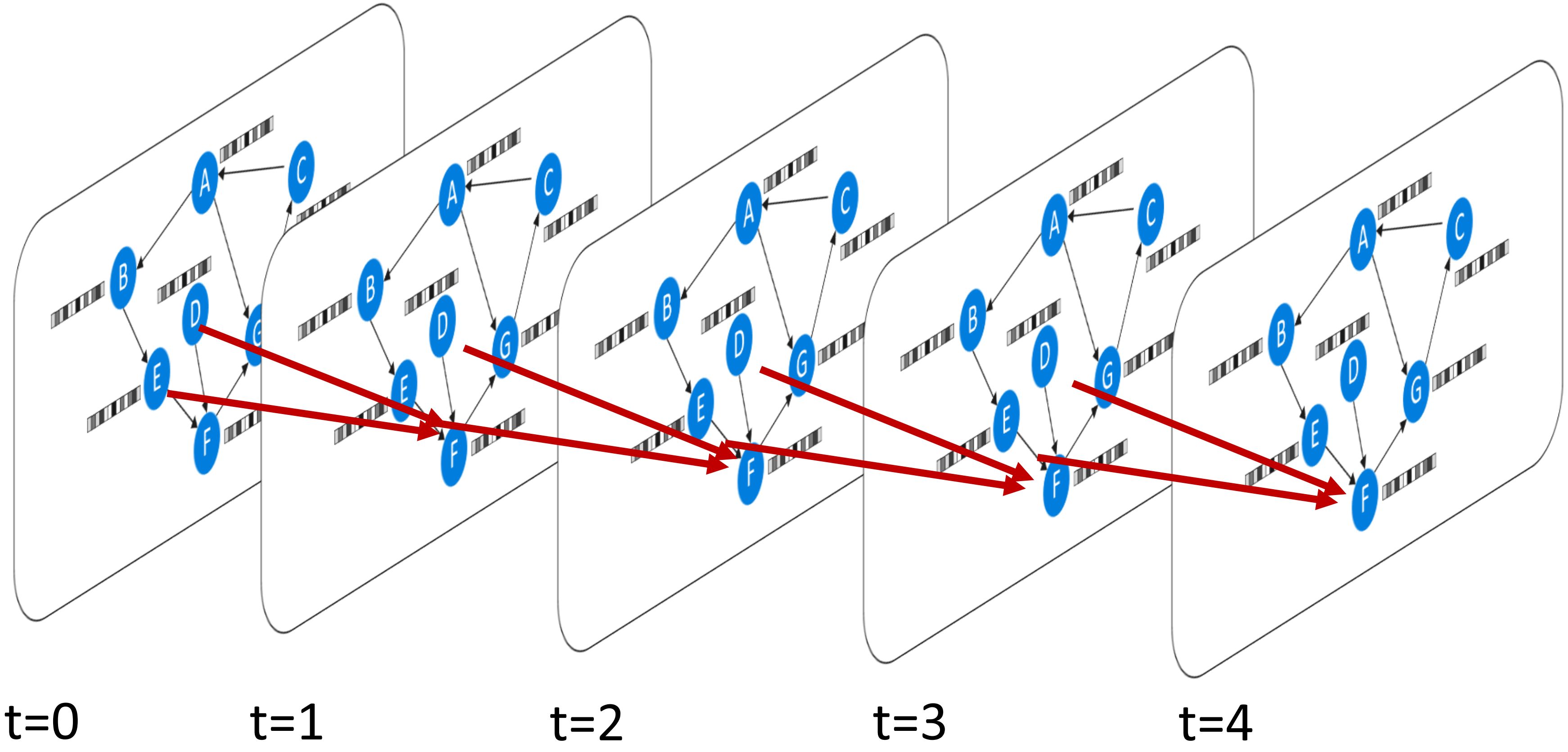
Microsoft

Neural Message Passing




A
B
C
D
E
F
G

Graph Neural Networks: Message Passing



GNNs: Synchronous Message Passing (All-to-All)