PHOG: Probabilistic Model for Code

Pavol Bielik, Veselin Raychev, Martin Vechev

Software Reliability Lab
Department of Computer Science
ETH Zurich

ICML@NYC
International Conference on Machine Learning

JUNE 19-24 2016 NEW YORK

Vision

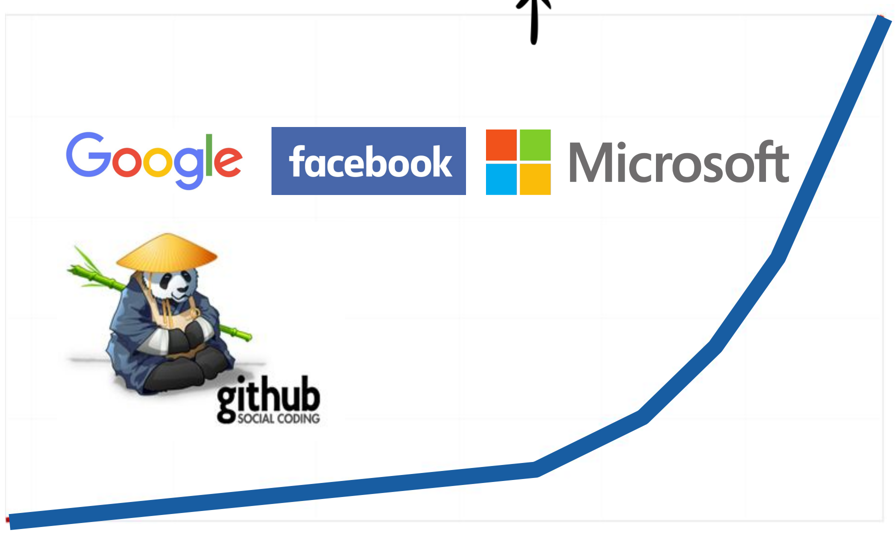
Statistical Programming Tool

↑

Probabilistic Model

↑

number of repositories



last 8 years

15 million repositories

Billions of lines of code

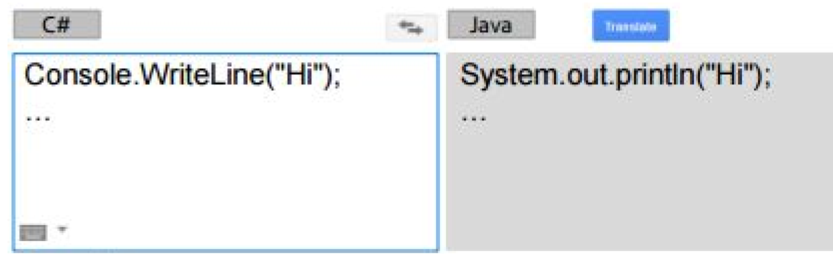
High quality, tested, maintained programs

Statistical Programming Tools

Write new code [PLDI'14]:
Code Completion

```
Camera camera = Camera.open();
camera.SetDisplayOrientation(90);
                ?
```

Port code [ONWARD'14]:
Programming Language Translation



Understand code/security [POPL'15]:
JavaScript Deobfuscation
Type Prediction

JS NICE

www.jsnice.org

Debug code:
Statistical Bug Detection

likely error

```
...
for x in range(a):
    print a[x]
```

All of these benefit from the probabilistic model for code.

Statistical Programming Tools

Write new code [PLDI'14]:
Code Completion

```
Cam
cam
```

Port code [ONWARD'14]:
Programming Language Translation

C# Java Translate

```
");
```

Und
Java
Typ

Programming Languages

+

Machine Learning

JS nice

www.jsnice.org

```
for x in range(a):
    print a[x]
```

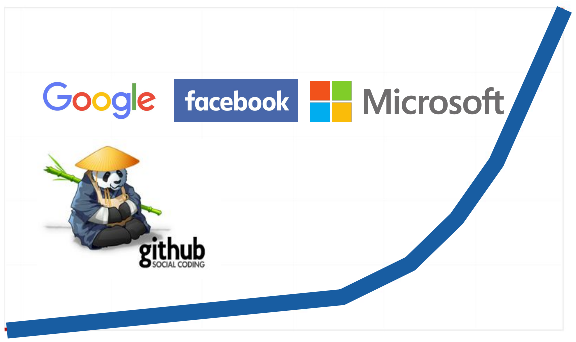
All of these benefit from the probabilistic model for code.

Model Requirements

Existing Programs

Learning

Model



Probabilistic Model

Widely Applicable

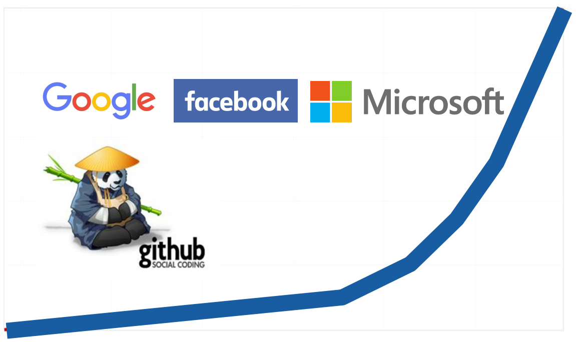
Efficient Learning

High Precision

Explainable Predictions

Model Requirements

Existing Programs | Learning | Model



→

Probabilistic Model

PHOG: Probabilistic Higher Order Grammar

Widely Applicable | **Efficient Learning** | **High Precision** | **Explainable Predictions**

Example Query

```
awaitReset = function(){
  ...
  return defer.promise;
}

awaitRemoved = function(){
  fail(function(error){
    if (error.status === 401){
      ...
    }
    defer.reject(error);
  });
  ...
  return defer.?
}
```

→ **PHOG** →

Correct prediction →

	P
promise	**0.67**
notify	0.12
resolve	0.11
reject	0.03

Challenges

```
awaitReset = function(){
  ...
  return defer.promise;
}

awaitRemoved = function(){
  fail(function(error){
    if (error.status === 401){
      ...
    }
    defer.reject(error);
  });
  ...
  return defer.?
}
```

Long distance dependencies

Correct prediction

	P
promise	**0.67**
notify	0.12
resolve	0.11
reject	0.03

Challenges

```
awaitReset = function(){
  ...
  return defer.promise;
}

awaitRemoved = function(){
  fail(function(error){
    if (error.status === 401){
      ...
    }
    defer.reject(error);
  });
  ...
  return defer.?
}
```

Long distance dependencies

Program semantics

Correct prediction

	P
promise	**0.67**
notify	0.12
resolve	0.11
reject	0.03

Challenges

```
awaitReset = function(){
  ...
  return defer.promise;
}

awaitRemoved = function(){
  fail(function(error){
    if (error.status === 401){
      ...
    }
    defer.reject(error);
  });
  ...
  return defer.?
}
```

Long distance dependencies

Program semantics

Explainable predictions

→ PHOG →

Correct prediction →

	P
promise	**0.67**
notify	0.12
resolve	0.11
reject	0.03

Existing Approaches for Code

Syntactic

[Hindle et al., 2012]
[Allamanis et al., 2015]

(features)
label conditioning context

$\underset{x}{\arg\max}\ P(\ x \mid$  $)$

Bad fit for programs

Existing Approaches for Code

Syntactic

[Hindle et al., 2012]
[Allamanis et al., 2015]

(features)

label conditioning context

$\arg\max_x P($ x | )

Semantic

[Nguyen et al., 2013]
[Allamanis et al., 2014]
[Raychev et al., 2014]

$\arg\max_x P($ x | defer reject promise)

Bad fit for programs

Hard-coded heuristics
Task & Language specific

PHOG: Concepts

Program synthesis learns a function that explains the data.
The function returns a conditioning context for a given query.

Use function to build a probabilistic model.
Generalizes PCFGs to allow conditioning on richer context.

Generalizing PCFG

Context Free Grammar

$$\alpha \rightarrow \beta_1 \ldots \beta_n$$

	P
Property $\rightarrow$ x	0.05
Property $\rightarrow$ y	0.03
Property $\rightarrow$ promise	0.001

PHOG: Generalizes PCFG

Context Free Grammar

$$\alpha \rightarrow \beta_1 \ldots \beta_n$$

	P
Property → x	0.05
Property → y	0.03
Property → promise	0.001

Higher Order Grammar

$$\alpha[\gamma] \rightarrow \beta_1 \ldots \beta_n$$

	P
Property[reject, promise] → promise	**0.67**
Property[reject, promise] → notify	0.12
Property[reject, promise] → resolve	0.11

Conditioning on Richer Context

$$\boldsymbol{\alpha}[\gamma] \rightarrow \boldsymbol{\beta}_1 \dots \boldsymbol{\beta}_n$$

What is the best conditioning context?

Conditioning on Richer Context

$$\alpha[\gamma] \rightarrow \beta_1 \ldots \beta_n$$

What is the best conditioning context?

- APIs
- Fields
- Identifiers
- Constants
- Control Structures
- ...

Conditioning on Richer Context

$$\alpha[\gamma] \rightarrow \beta_1 \ldots \beta_n$$

What is the best conditioning context?

- APIs
- Fields
- Identifiers
- Constants
- Control Structures
- ...

Source Code → ? → γ

Conditioning Context

Higher Order Grammar

Production Rules R:

$$\alpha[\gamma] \rightarrow \beta_1 \ldots \beta_n$$

Function:

f:  $\rightarrow \gamma$

Parametrize the grammar by a function used to dynamically obtain the context

Higher Order Grammar

Production Rules R:

$$\alpha[\gamma] \rightarrow \beta_1 \ldots \beta_n$$

Function:

$$f: \quad \text{AST} \quad \rightarrow \gamma$$

Parametrize the grammar by a function used to dynamically obtain the context

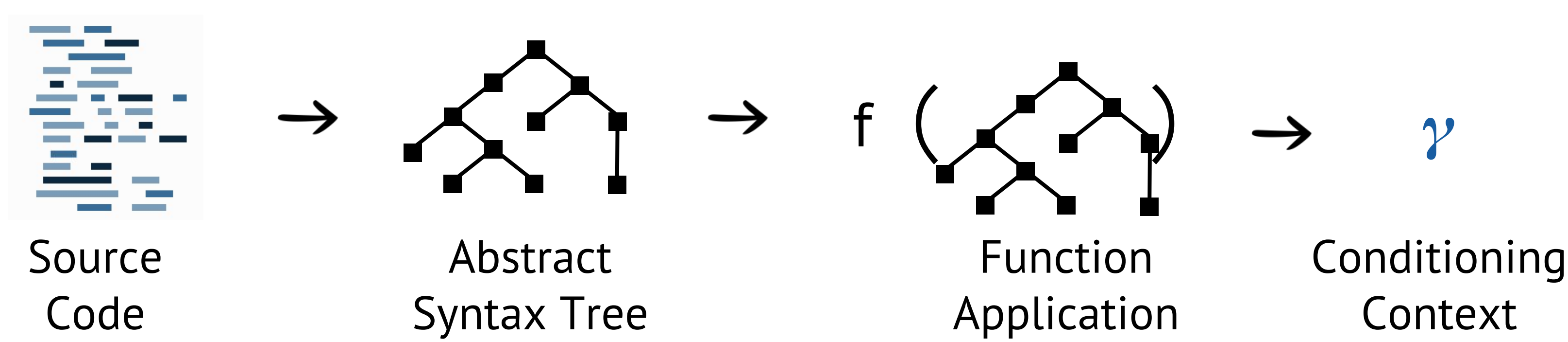
Higher Order Grammar

Production Rules R:

$$\alpha[\gamma] \rightarrow \beta_1 \ldots \beta_n$$

Function:

$$f: \quad \text{AST} \quad \rightarrow \gamma$$



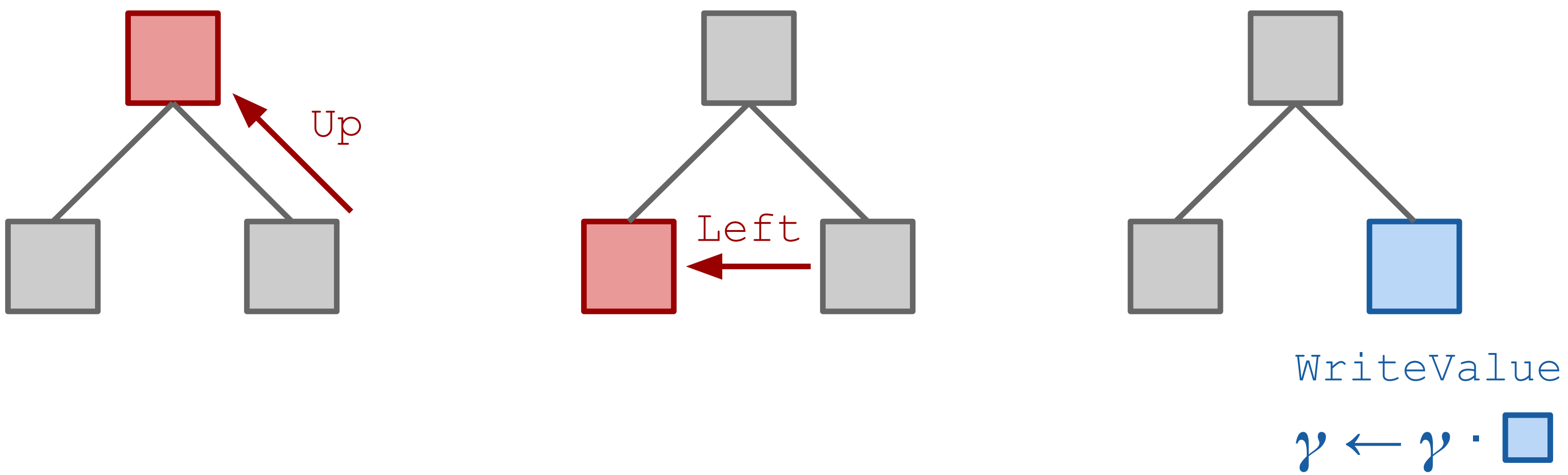
Function Representation

In general:
Unrestricted programs (Turing complete)

Our Work:
TCond Language for navigating over trees
and accumulating context

```
TCond    ::=  ε | WriteOp TCond | MoveOp TCond
MoveOp   ::=  Up, Left, Right, DownFirst, DownLast,
              NextDFS, PrevDFS, NextLeaf, PrevLeaf,
              PrevNodeType, PrevNodeValue, PrevNodeContext
WriteOp ::=  WriteValue, WriteType, WritePos
```

Expressing functions: TCond Language




```
TCond    ::=  ε | WriteOp TCond | MoveOp TCond
MoveOp   ::=  Up, Left, Right, DownFirst, DownLast,
              NextDFS, PrevDFS, NextLeaf, PrevLeaf,
              PrevNodeType, PrevNodeValue, PrevNodeContext
WriteOp  ::=  WriteValue, WriteType, WritePos
```

Example

Query TCond Program γ

```
elem.notify(
   ... ,
   ... ,
   {
     position: 'top',
     hide: false,
     ?
   }
);
```

Example

Query	TCond	γ
	Left	{}
	WriteValue	{hide}

```
elem.notify(
   ... ,
   ... ,
   {
     position: 'top',
     hide: false,
     ?
   }
);
```

Example

Query	TCond	γ
	Left	{}
	WriteValue	{hide}
	Up	{hide}
	WritePos	{hide, 3}

```
elem.notify(
  ... ,
  ... ,
  {
    position: 'top',
    hide: false,
    ?
  }
);
```

Example

Query	TCond	γ
	Left	{}
	WriteValue	{hide}
	Up	{hide}
	WritePos	{hide, 3}
	Up	{hide, 3}
	DownFirst	{hide, 3}
	DownLast	{hide, 3}
	WriteValue	{hide, 3, notify}

```
elem.notify(
   ... ,
   ... ,
  {
     position: 'top',
     hide: false,
     ?
  }
);
```

Example

```
elem.notify(
   ... ,
   ... ,
  {
    position: 'top',
    hide: false,
    ?
  }
);
```

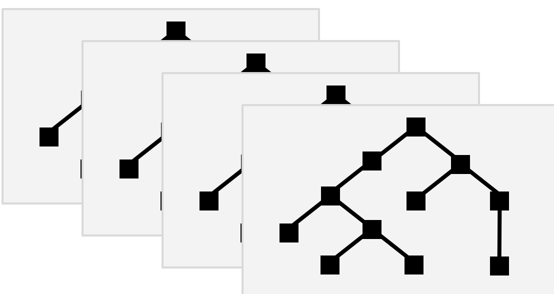
Query	TCond	γ
	Left	{}
	WriteValue	{hide}
	Up	{hide}
	WritePos	{hide, 3}
	Up	{hide, 3}
	DownFirst	{hide, 3}
	DownLast	{hide, 3}
	WriteValue	{hide, 3, notify}

↓

{ Previous Property, Parameter Position, API name }

Learning PHOG

Existing Dataset



```
TCond   ::=  ε | WriteOp TCond | MoveOp TCond
MoveOp  ::=  Up, Left, Right, ...
WriteOp ::=  WriteValue, WriteType, ...
```

TCond Language

Program Synthesis

Enumerative search

Genetic programming

$$f_{best} = \arg\min_{f \in \text{TCond}} \mathit{cost}(D, f)$$

$|d| \ll |D|$

$|\text{cost}(d, f) - \text{cost}(D, f)| < \varepsilon$

Representative sampling

Learning Programs from Noisy Data. POPL '16, ACM.

Evaluation

Probabilistic Model of JavaScript Language

20k TCond learning 100k PHOG training 50k Blind Set

GitHub

Evaluation

	Code Completion Error Rate
PCFG	49.9%
n-gram	28.7%
Naive Bayes	45.8%
SVM	29.5%
PHOG	**18.5%**

Evaluation

Code Completion

	Error Rate	Example
Identifier	38%	`contains = jQuery …`
Property	35%	`start = list.length;`
String	48%	`'[' + attrs + ']'`
Number	36%	`canvas(xy[0], xy[1], …)`
RegExp	34%	`line.replace(/( | )+/, …)`
UnaryExpr	3%	`if (!events || !…)`
BinaryExpr	26%	`while (++index < …)`
LogicalExpr	8%	`frame = frame || …`

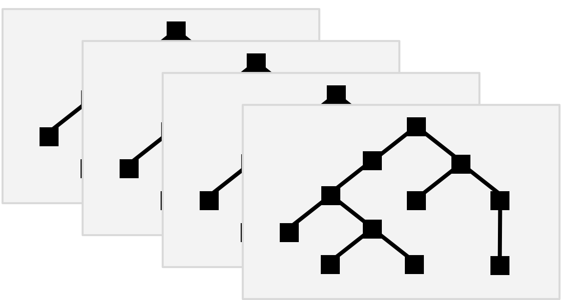
Evaluation

	Training Time	Queries per Second
PCFG	1 min	71 000
n-gram	4 min	15 000
Naive Bayes	3 min	10 000
SVM	36 hours	12 500
PHOG	**162 + 3 min**	**50 000**

PHOG: Probabilistic Higher Order Grammar

Widely Applicable	Efficient Learning	High Precision	Explainable Predictions

dataset



```
TCond    ::=  ε | WriteOp TCond | MoveOp TCond
MoveOp   ::=  Up, Left, Right, ...
WriteOp  ::=  WriteValue, WriteType, ...
```

TCond Language

Key Ideas:

- **Learn a function that explains the data. The function dynamically obtains the best conditioning context for a given query.**

$$f_{best} = \underset{f \in \text{TCond}}{\arg\min}\ cost(D, f)$$

- **Define a new generative model that is parametrized by such learned function.**

$$\text{PHOG}(f_{best})$$