Microsoft

Bug Detection and Repair with ML

Miltiadis Allamanis, Henry Jackson-Flux, Marc Brockschmidt

Microsoft Research, Cambridge, UK

https://arxiv.org/abs/2105.12787

@miltos1

miltos.allamanis.com

Hello 👋

Machine Learning for Code

- 🔍 Learned program analyses
- 🐞 Bug detection & repair
- 📦 Building blocks for learning-to-reason over code
- 🚀 Practically deployment of developer tools

Machine Learning for Structured & Dynamic Objects

- 🧠 Understanding, reasoning about, and predicting structured objects
- ⚡ Graph Neural Networks


@miltos1


miltos.allamanis.com

Towards AI Pair Programming

Specification Tuning

- A formal program analysis.
- Tune (discount some factors) to reduce false positives.

Specification Inference

- Assume most code complies with a latent spec.
- Predict a spec.
- Verify with standard methods.

Black-Box Analysis Learning

- Assume most code is “correct”.
- Model (latent) user intent and deviations from it.
- Raise warnings on detected deviations.

Graph Neural Networks on Program Analysis. Allamanis M, 2021

Learned Program Analyses

Why Self-Supervised Bug Detection and Repair?

```
def make_id(name):
    """
    Create a random id combined with the creditor name.
    @return string consisting of name (truncated at 22 chars), -,
    12 char rand hex string.
    """
    r = get_rand_string(12)
    if len(name 0.0%) <= 99.8% 22:
        name 0.0% =0.0% name 0.0%[:22]
    return name 0.0% +0.0% "-" +0.0% r 0.0%
```

https://github.com/raphaelm/python-sepaxml.git: /sepadd/utils.py

Why Self-Supervised Bug Detection and Repair?

```
def make_id(name):
    """
    Create a random id combined with the creditor name.
    @return string consisting of name (truncated at 22 chars), -,
    12 char rand hex string.
    """
    r = get_rand_string(12)
    if len(name 0.0%) <= 99.8% 22:
        name 0.0% =0.0% name 0.0%[:22]
    return name 0.0% +0.0% "-" +0.0% r 0.0%
```

https://github.com/raphaelm/python-sepaxml.git: /sepadd/utils.py

Bug Lab

Neurosymbolic Bug Fixing

Prior Work

DeepBugs:
A Learning Approach to Name-Based Bug Detection

MICHAEL PRADEL, TU Darmstadt, Germany
KOUSHIK SEN, University of California, Berkeley, USA

147

Learning to Fix Build Errors with Graph2Diff Neural Networks

Daniel Tarlow, Google
Subhodeep Moitra, Google
Andrew Rice, University of Cambridge & Google
Zimin Chen*, KTH Royal Institute of Technology
Pierre-Antoine Manzagol, Google
Charles Sutton, Google
Edward Aftandilian, Google

LEARNING TO REPRESENT PROGRAMS WITH GRAPHS

Miltiadis Allamanis
Microsoft Research
Cambridge, UK

Marc Brockschmidt
Microsoft Research
Cambridge, UK

GLOBAL RELATIONAL MODELS OF SOURCE CODE

Vincent J. Hellendoorn, Petros Maniatis, Rishabh Singh, Charles Sutton, David Bieber
Google Research
{vhellendoorn,maniatis,rising,charlessutton,dbieber}@google.com

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

Elizabeth Dinella*
University of Pennsylvania

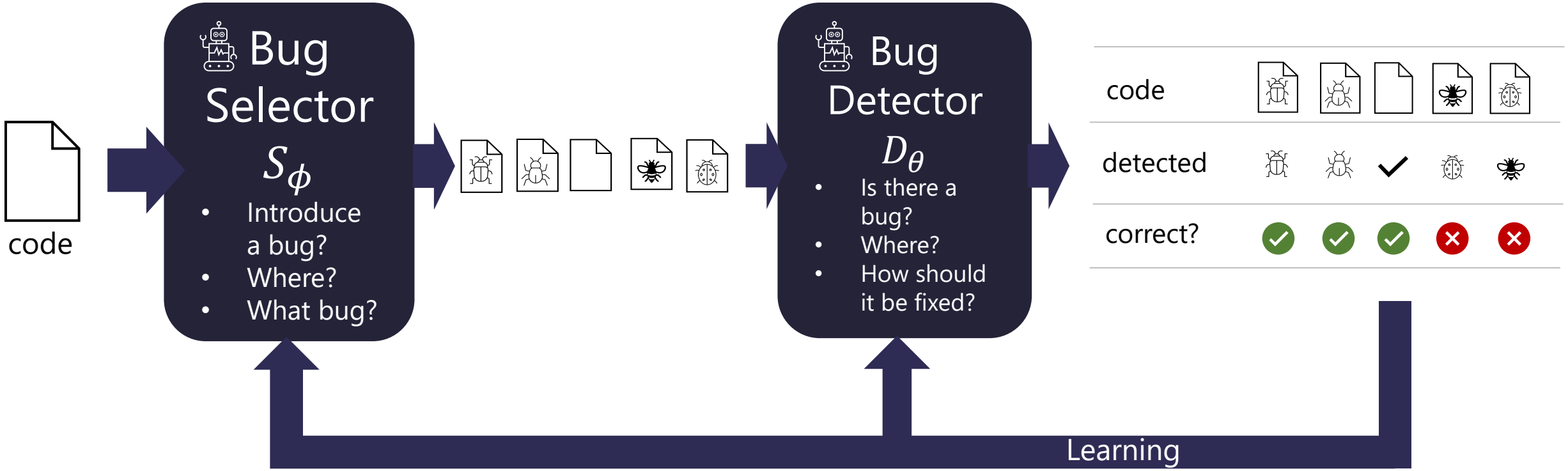
Hanjun Dai*
Google Brain

Ziyang Li
University of Pennsylvania

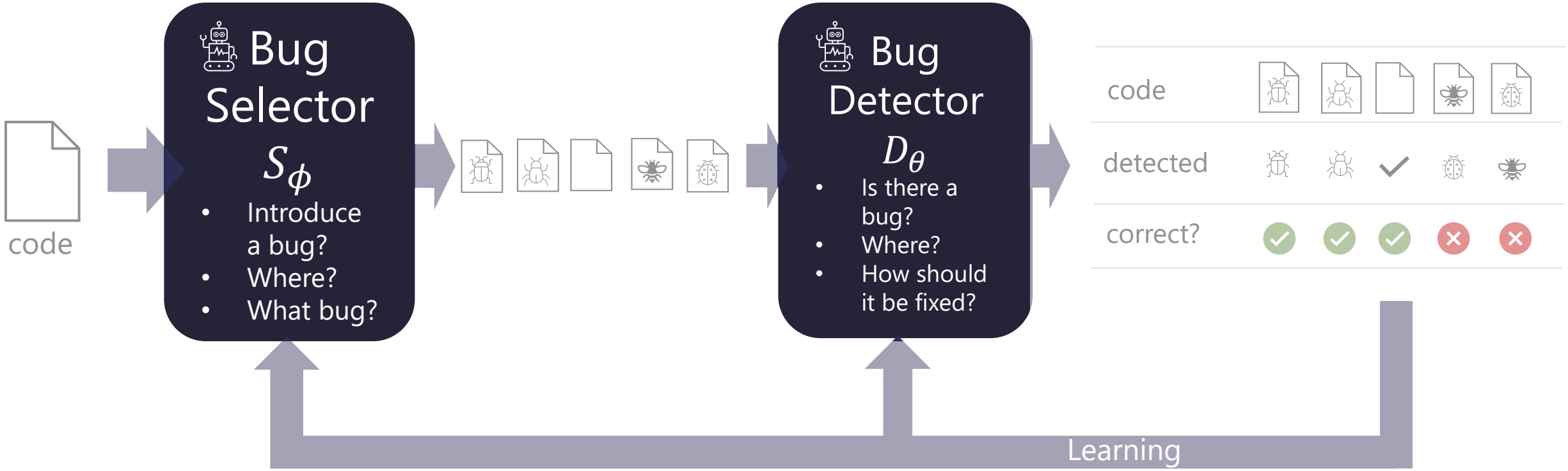
Mayur Naik
University of Pennsylvania

Le Song
Georgia Tech

Ke Wang
Visa Research

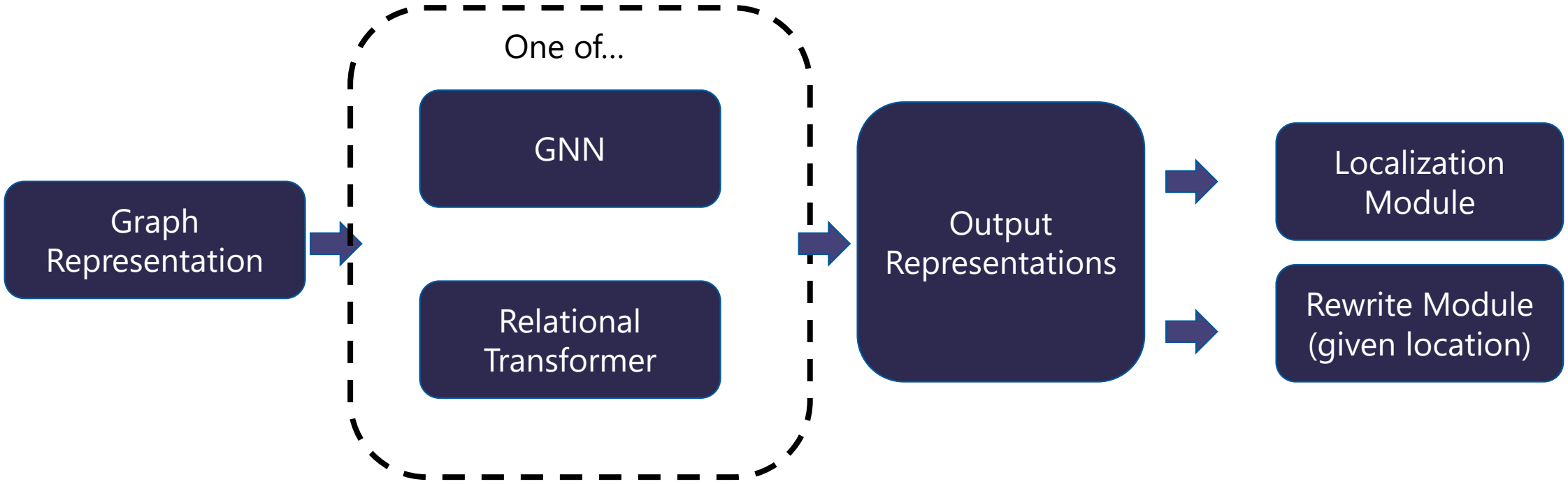

code
Bug Selector
S_ϕ
• Introduce a bug?
• Where?
• What bug?
Bug Detector
D_θ
• Is there a bug?
• Where?
• How should it be fixed?
code
detected
correct?
Learning

BugLab


code
Bug Selector
S_ϕ
• Introduce a bug?
• Where?
• What bug?
Bug Detector
D_θ
• Is there a bug?
• Where?
• How should it be fixed?
code
detected
correct?
Learning

BugLab

Neural Models



GNN: Allamanis, M., et al. “Learning to Represent Programs with Graphs." *ICLR* 2017

GREAT: Hellendoorn, V. J., et al. "Global Relational Models of Source Code." *ICLR* 2019

GNN Architecture

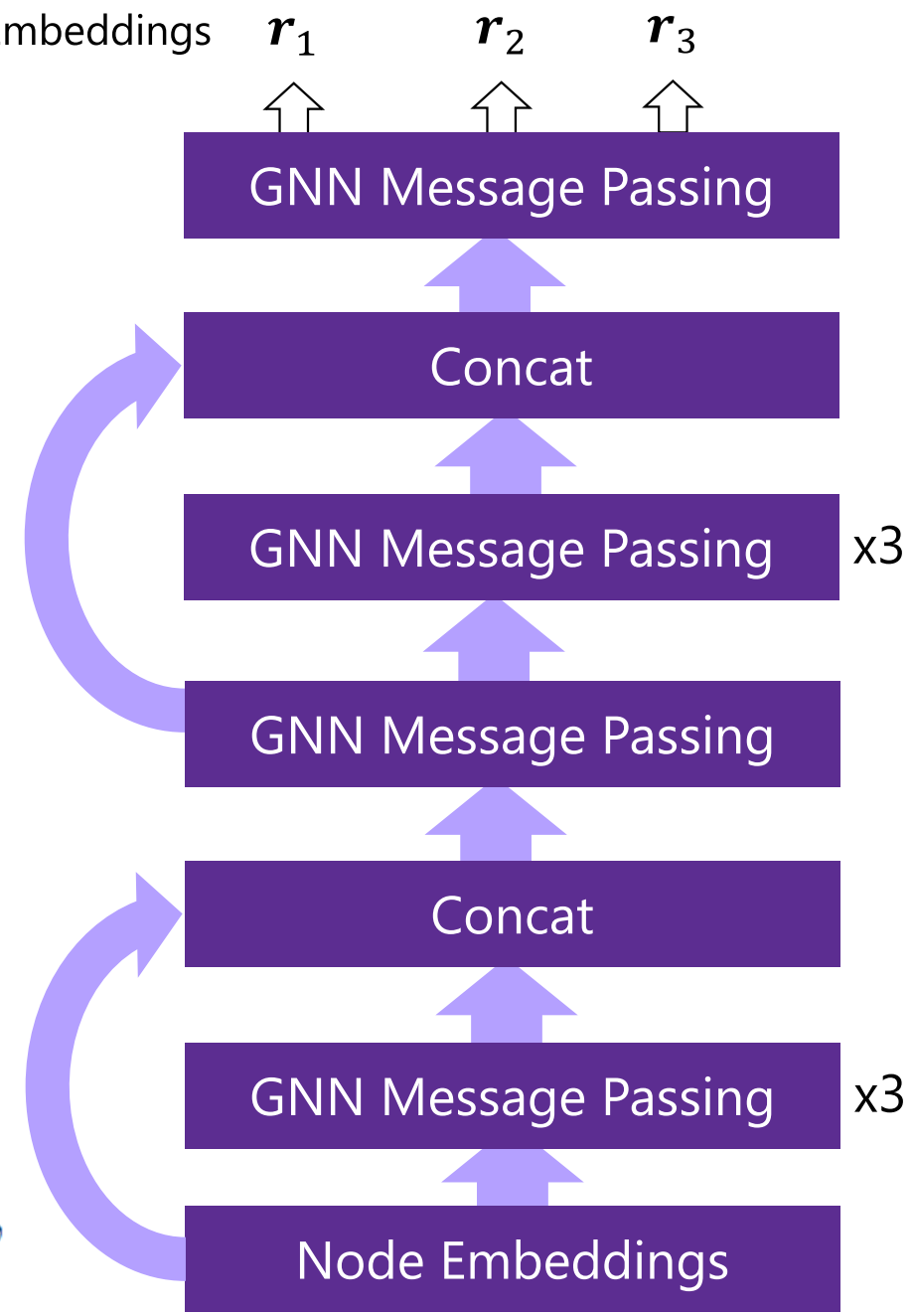
$$\boldsymbol{h}_{v_i}^{(t+1)} = f^{(t)}\left(\boldsymbol{h}_{v_i}^{(t)}, \bigoplus_{\forall v_j : v_i \xrightarrow{k} v_j} \left(m^{(t)}\left(\boldsymbol{h}_{v_i}^{(t)}, k, \boldsymbol{h}_{v_j}^{(t)}\right)\right)\right)$$

$$m^{t}\left(\boldsymbol{h}_{v_i}^{(t)}, k, \boldsymbol{h}_{v_j}^{(t)}\right) = W_k^{(t)}\left[\boldsymbol{h}_{v_i}^{(t)}, \boldsymbol{h}_{v_j}^{(t)}\right]$$

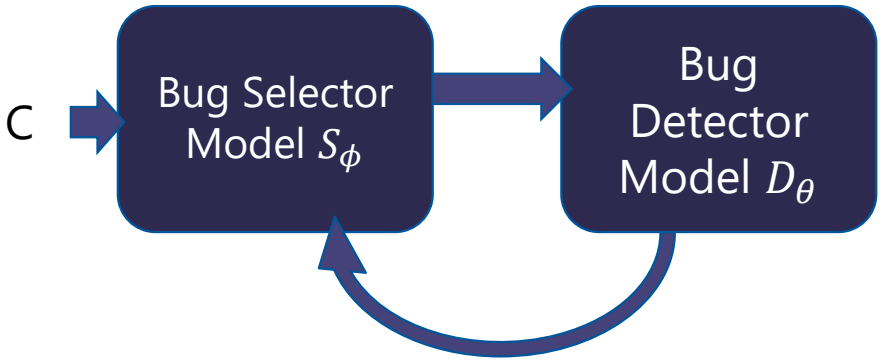
$$f^{(t)}(\cdot) = \tanh\left(W_f^{(t)} \cdot \textsc{LayerNorm}\left(\textsc{Gelu}(\boldsymbol{m})\right) + \boldsymbol{b}_f\right)$$

$$H^{(4)} = \textsc{GNN}_4\left(\left[H^{(0)}, \textsc{GNN}_3\left(\textsc{GNN}_2\left(\textsc{GNN}_1\left(H^{(0)}\right)\right)\right)\right]\right)$$

$$H^{(8)} = \textsc{GNN}_8\left(\left[H^{(4)}, \textsc{GNN}_7\left(\textsc{GNN}_6\left(\textsc{GNN}_5\left(H^{(4)}\right)\right)\right)\right]\right),$$



Objective



$$\min_\theta E_{c\sim C}\left[\max_{b\in R(c)} \mathcal{L}_{D_\theta}(b,c)\right]$$

All "available" rewrites.

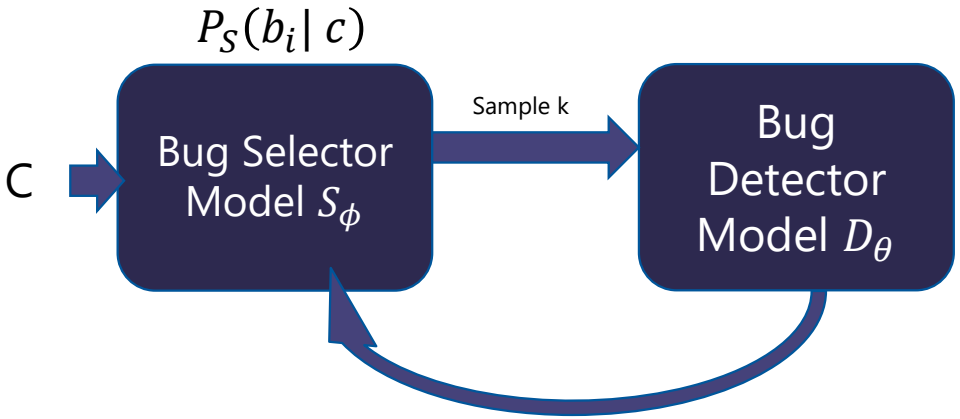
$$= \min_\theta E_{c\sim C}\left[\mathcal{L}_{D_\theta}\left(\left(\underbrace{\underset{b\in R(c)}{argmax}\, \mathcal{L}_{D_\theta}(b,c)}\right),c\right)\right]$$

Intractable:
Learn a model

Objective

Approximate/Model argmax with $S_\phi^{(T)}$ where T is some temperature

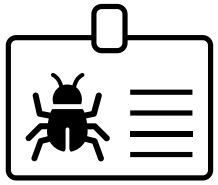
$$\min_\theta E_{c\sim C}\left[E_{b\sim S_\phi^{(T)}(c)}\left[\mathcal{L}_{D_\theta}(b,c)\right]\right]$$



- Given a snippet c from corpus C
- Bug selector model selects k bugs to insert $\propto P_S(b_i|c)$.
- Bug detector tries to detect bugs (fully supervised), if any.

$$P_D(c|b_i)$$

- Bug selector observes output

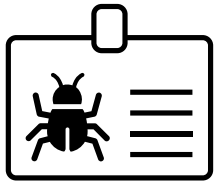


Types of Rewrites

	Example		
· Replace Variable Usage	`i`	→	`j`
· Replace Binary Operator	`+`	→	`-`
· Replace Assignment Op	`+=`	→	`-=`
· Replace Boolean Operator	`or`	→	`and`
· Replace Comparison Operator	`==`	→	`!=`
· Replace (some) Literals	`0`	→	`1`
· Argument Swap	`foo(a+1,b)`	→	`foo(b,a+1)`

Bug Lab

Neurosymbolic Bug Repair

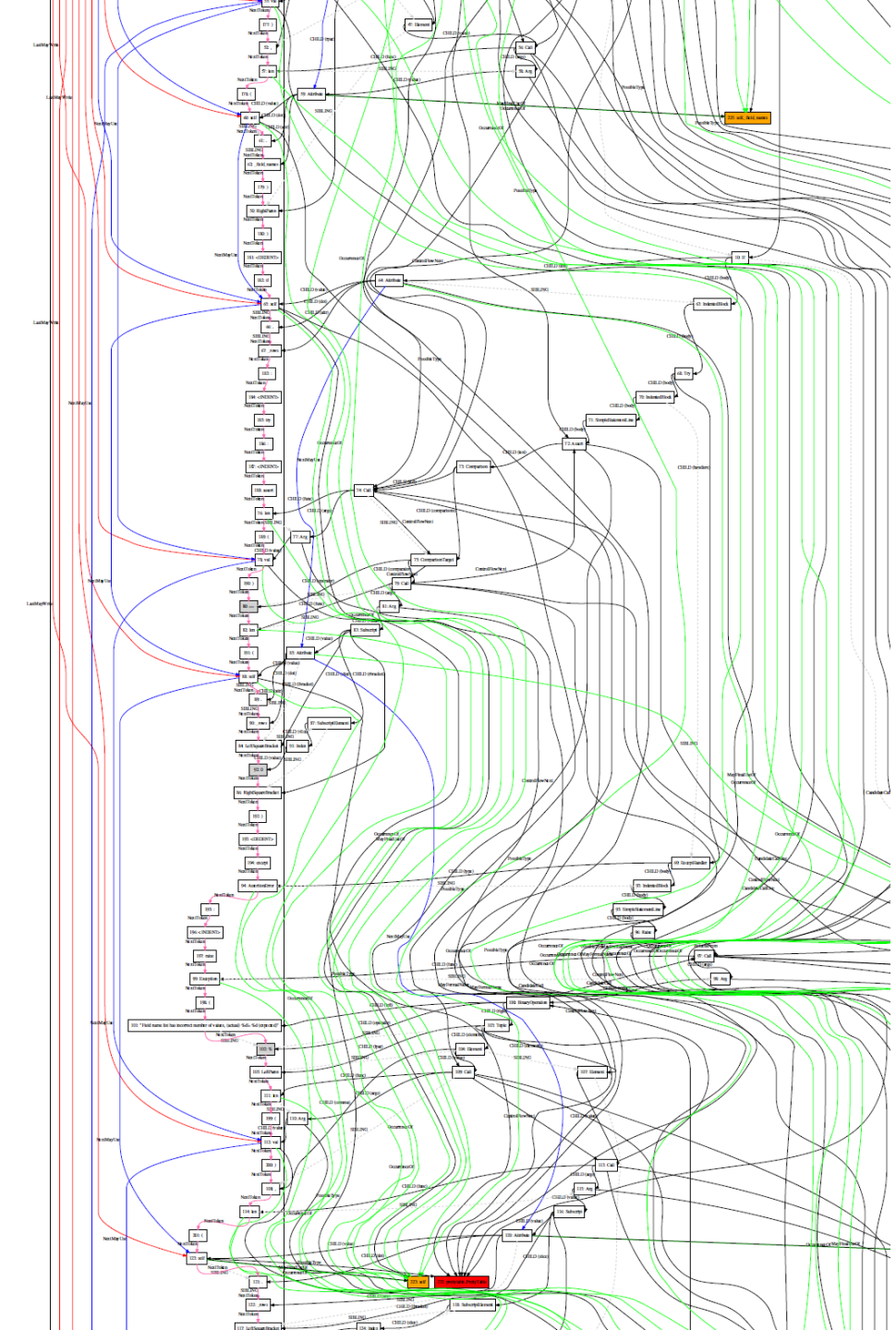


Types of Rewrites

```
def foo(a, b, c=0):
  if a[1] in[2] b[3]:
    c[4] +=[5] bar(b[7], c[8])[6]
  c_is_neg =[9] c[10] <[11] 0[12]
  if c_is_neg[13] or[14] a[15] is[16] int:
    return True[17], c[18]
  return c[19] >[20] 1[21], c[22]
```

ϵ: NoBug
l_1: b, c
l_2: not in
l_3: a, c
l_4: a, b
l_5: =, -=, *=, /=, //=, %=
l_6: bar(c, b)
l_7: a, c
l_8: a, b
l_9: +=, -=, *=, /=, //=, %=
l_{10}: a, b
l_{11}: <=, >, >=, ==, !=
l_{12}: -2, -1, 1, 2
l_{13}: a, b, c, not c_is_neg
l_{14}: and
l_{15}: b, c, c_is_neg
l_{16}: is not
l_{17}: False
l_{18}: a, b, c_is_neg
l_{19}: a, b, c_is_neg
l_{20}: >=, <, <=, ==, !=
l_{21}: -2, -1, 0, 2
l_{22}: a, b, c_is_neg

BugLab
Neurosymbolic Bug Repair



Code Representation

Entities (Nodes)

- Tokens
- Non-Terminal Nodes
- Symbols

Relationships (Edges)

Syntax

- AST Child
- AST Sibling
- Next Token

Data Flow

- MayFinalUseOf
- LastMayWrite
- NextMayUse

Symbols

- CandidateType
- OccurrenceOf
- CandidateMethodName

Function Calls

- CandidateFormalArg
- CandidateDocStringOf

Control Flow

- ControlFlowNext
- AssignedFrom
- ReturnsFrom
- YieldsFrom

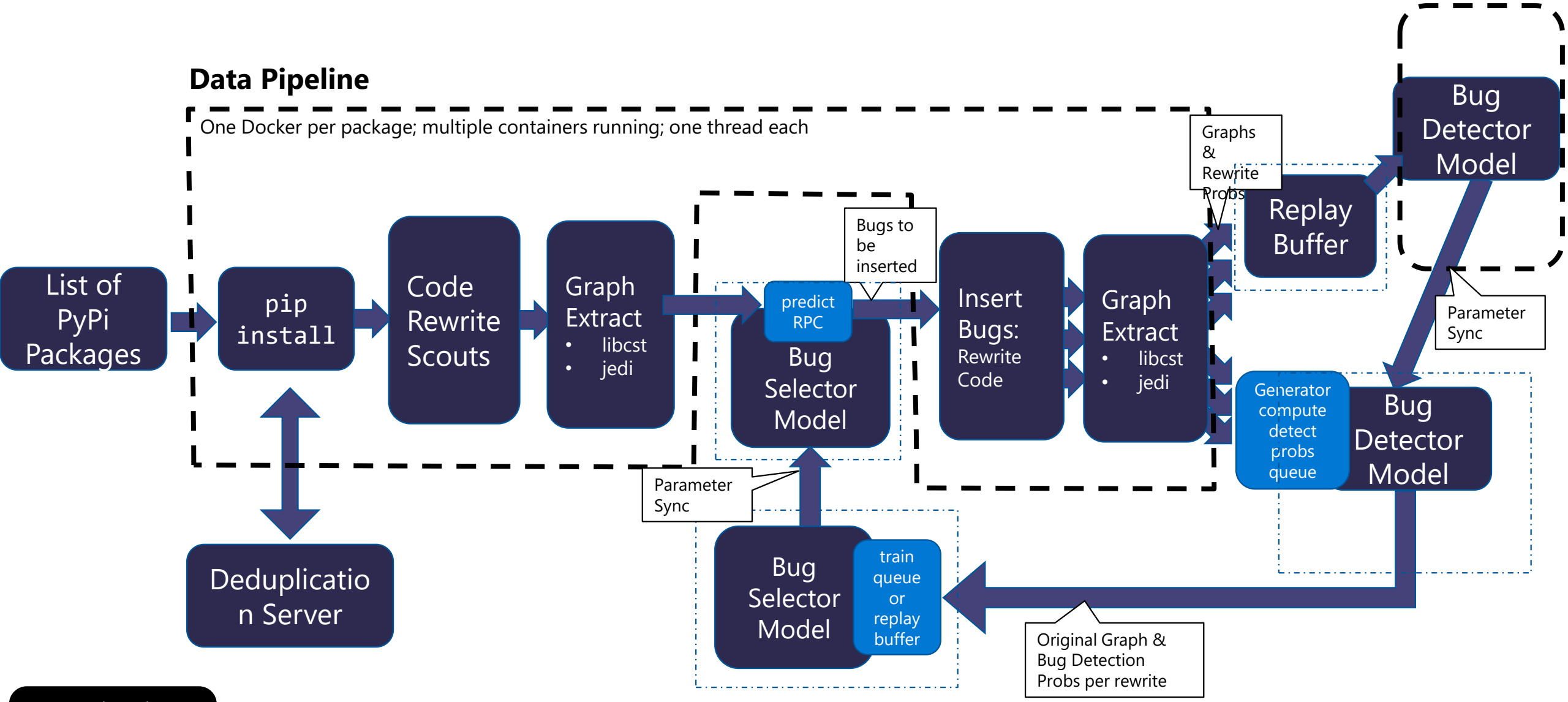
https://github.com/microsoft/neurips21-self-supervised-bug-detection-and-repair

(Almost) Semantic-Preserving Rewrites

- Variable Renaming
- Comment Deletion
- Comparison Expression Mirroring
- If-Else Branch Swapping

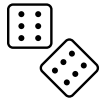
LibCST and https://github.com/microsoft/neurips21-self-supervised-bug-detection-and-repair

Infrastructure



https://github.com/microsoft/neurips21-self-supervised-bug-detection-and-repair

Evaluation Datasets



RandomBugs

- ~700k random bugs
- Relatively Large
- Potentially non-representative of real bugs

PyPIBugs

- 2k real bugs
- Manually curated/labeled
- Small. Used as testset only.

Localization & Repair Accuracy

RandomBugs

	GNN	GREAT
Supervised	62.4	51.0
BugLab	**70.3**	65.3

PyPIBugs

	GNN	GREAT
Supervised	20.1	16.5
BugLab	**26.2**	22.9

GNN: Allamanis, M., et al. “Learning to Represent Programs with Graphs." *ICLR* 2017

GREAT: Hellendoorn, V. J., et al. "Global relational models of source code." *ICLR* 2019

BugLab

Localization & Repair Accuracy

PyPIBugs

	PyPIBugs						PyPIBugs-PostFix			
	GNN			GREAT			GNN		GREAT	
	Joint	Loc	Repair	Joint	Loc	Repair	Loc	Joint AUC	Loc	Joint AUC
Supervised	20.0	28.4	61.8	16.8	25.8	58.6	17.8	0.087	20.7	0.044
Random Selector	21.2	27.0	69.2	20.6	26.8	67.2	47.5	0.108	**52.5**	0.117
PyBugLab	24.2	31.3	70.7	24.0	32.8	67.9	32.9	0.160	28.6	0.140
PyBugLab +Aug	**26.4**	**33.5**	**72.0**	23.2	29.7	68.8	32.6	**0.187**	48.2	0.129

GNN: Allamanis, M., et al. “Learning to Represent Programs with Graphs." *ICLR* 2017

GREAT: Hellendoorn, V. J., et al. "Global relational models of source code." *ICLR* 2019

Bug Lab

By Francis Barlow - http://mythfolklore.net/aesopica/barlow/59.htm, Public Domain, https://commons.wikimedia.org/w/index.php?curid=14476836

program analysis

The ~~boy~~ who

error

cried ~~wolf~~

Bug Lab

Neurosymbolic Bug Repair

Conversation 1 | Commits 1 | Checks 0 | Files changed 1

Changes from **all commits** | **File filter** | **Conversations** | **Jump to**

2 gremlin-python/src/main/python/gremlin_python/process/strategies.py

```
@@ -64,7 +64,7 @@ def __init__(self, partition_key=None, write_partition=None, read_partitions=Non
64  64              self.configuration["partitionKey"] = partition_key
65  65          if write_partition is not None:
66  66              self.configuration["writePartition"] = write_partition
67      -       if write_partition is not None:
    67  +       if read_partitions is not None:
68  68              self.configuration["readPartitions"] = read_partitions
69  69          if include_meta_properties is not None:
70  70              self.configuration["includeMetaProperties"] = include_meta_properties
```

2 webexteamssdk/api/rooms.py

@@ -213,27 +213,27 @@

```
213 213             )
214 214
215 215             # Return a room meeting info object created from the response JSON data
216 216             return self._object_factory("room_meeting_info", json_data)
217 217
218 218         def update(self, roomId, title, **request_parameters):
219 219             """Update details for a room, by ID.
220 220
221 221             Args:
222 222                 roomId(basestring): The room ID.
223 223                 title(basestring): A user-friendly name for the room.
224 224                 **request_parameters: Additional request parameters (provides
225 225                     support for parameters that may be added in the future).
226 226
227 227             Returns:
228 228                 Room: A Room object with the updated Webex Teams room details.
229 229
230 230             Raises:
231 231                 TypeError: If the parameter types are incorrect.
232 232                 ApiError: If the Webex Teams cloud returns an error.
233 233
234 234             """
235 235             check_type(roomId, basestring)
236     -           check_type(roomId, basestring)
    236 +           check_type(title, basestring)
237 237
238 238             put_data = dict_from_items_with_values(
239 239                 request_parameters,
```

Open Challenges

Explainability

Learning to Abstain - OoD

Uncertainty Estimation

Open Challenges

Incorporate more formal information

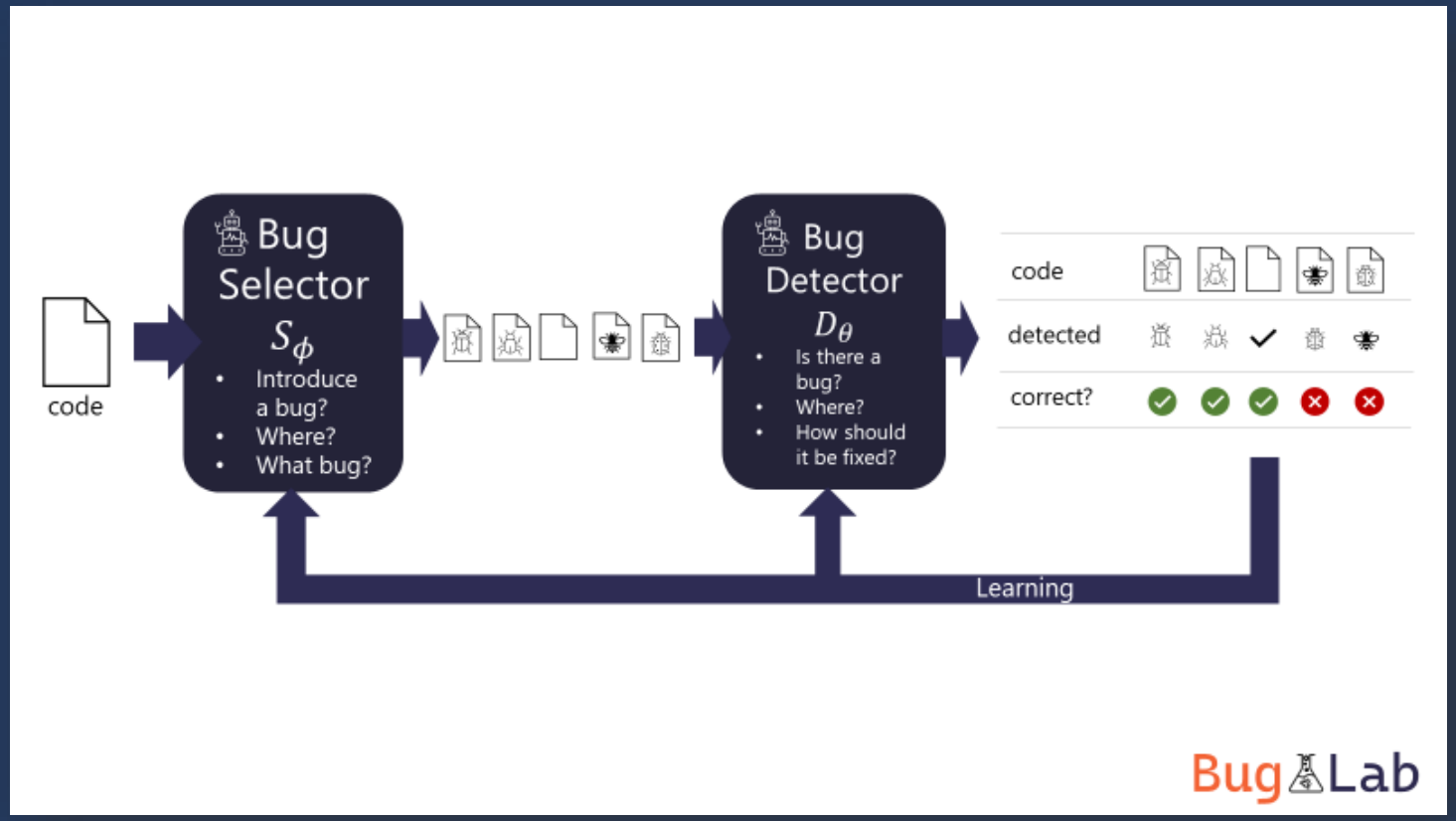
Longer Contexts

Performance / False Positives

Microsoft

@miltos1

miltos.allamanis.com



Self-Supervised Bug Detection and Repair

Miltiadis Allamanis, Henry Jackson-Flux, Marc Brockschmidt