On Learning Meaningful Code Changes via Neural Machine Translation

M. Tufano, J. Pantiuchina, C. Watson, G. Bavota, and D. Poshyvanyk

Source code evolves

Source code evolves, inevitably.

Software Evolution and Maintenance

```
public void addElement (Element elem) {
    myList.add(elem);
}
```

Software Evolution and Maintenance

```java
public void addElement (Element elem) {
     myList.add(elem);
}
```

```java
public void addElement (Element elem) {
          if(myList != null){
                    myList.add(elem);
          }
}
```

Corrective

Software Evolution and Maintenance

```
public void addElement (Element elem) {
        if(myList != null){
                myList.add(elem);
        }
}

public boolean addElement (Element elem) {
        if(myList != null){
                myList.add(elem);
                return true;
        }
        return false;
}
```

Corrective

Adaptive

Software Evolution and Maintenance

```java
public boolean addElement (Element elem) {
        if(myList != null){
                myList.add(elem);
                return true;
        }
        return false;
}
```

Corrective

Adaptive

```java
/**
* Add element in the list
* @param element to add
* @return true if element added, false otherwise
*/
public boolean addElement (Element elem) {
        if(myList != null){
                myList.add(elem);
                return true;
        }
        return false;
}
```

Perfective

Software Evolution and Maintenance

```java
/**
* Add element in the list
* @param element to add
* @return true if element added, false otherwise
*/
public boolean addElement (Element elem) {
        if(myList != null){
                myList.add(elem);
                return true;
        }
        return false;
}
/** * Add element in the list
* @param element to add
* @return true if element added, false otherwise
*/
public boolean addElement (Element elem) {
        if(myList == null){
                initList(myList);
        }
        myList.add(elem);
        return true;
}
```

Corrective

Adaptive

Perfective

Preventive

Software Evolution and Maintenance

Software Evolution and Maintenance

Automation

Automation

Heuristics + Handcrafted Rules + Human Experts

Learning from Changes

Automatically

Code Features

Learning from Changes

Automatically

Code Features

Change Patterns

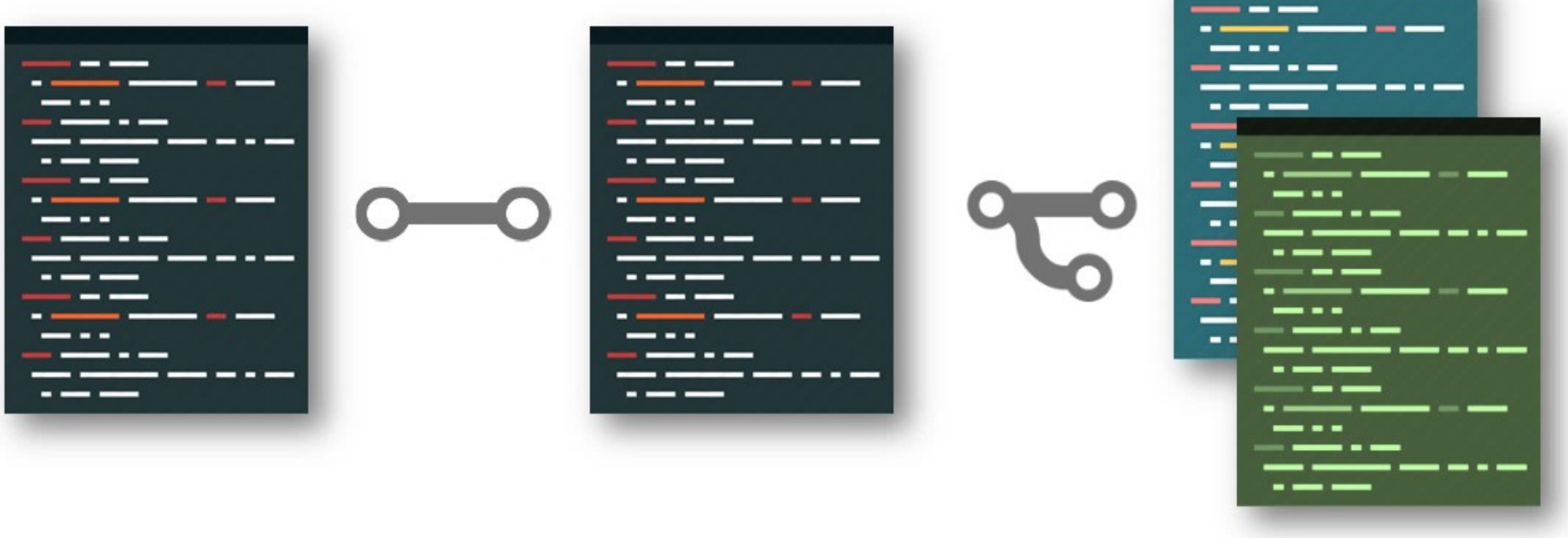
Learning from Changes

Automatically

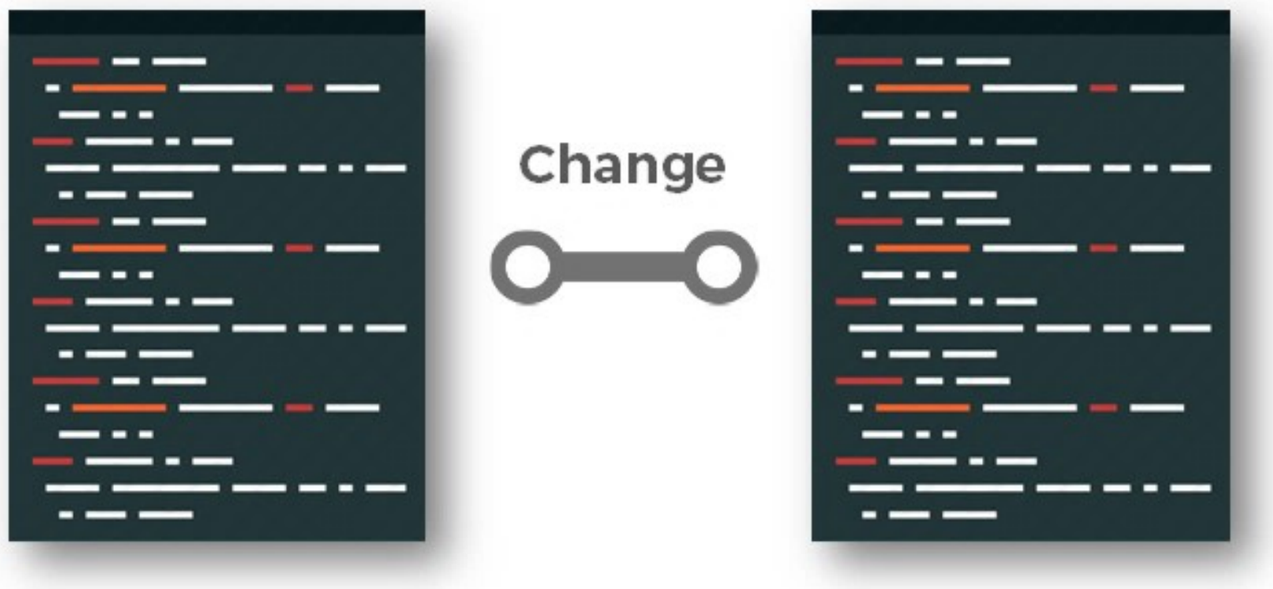
Code Features

Change Patterns

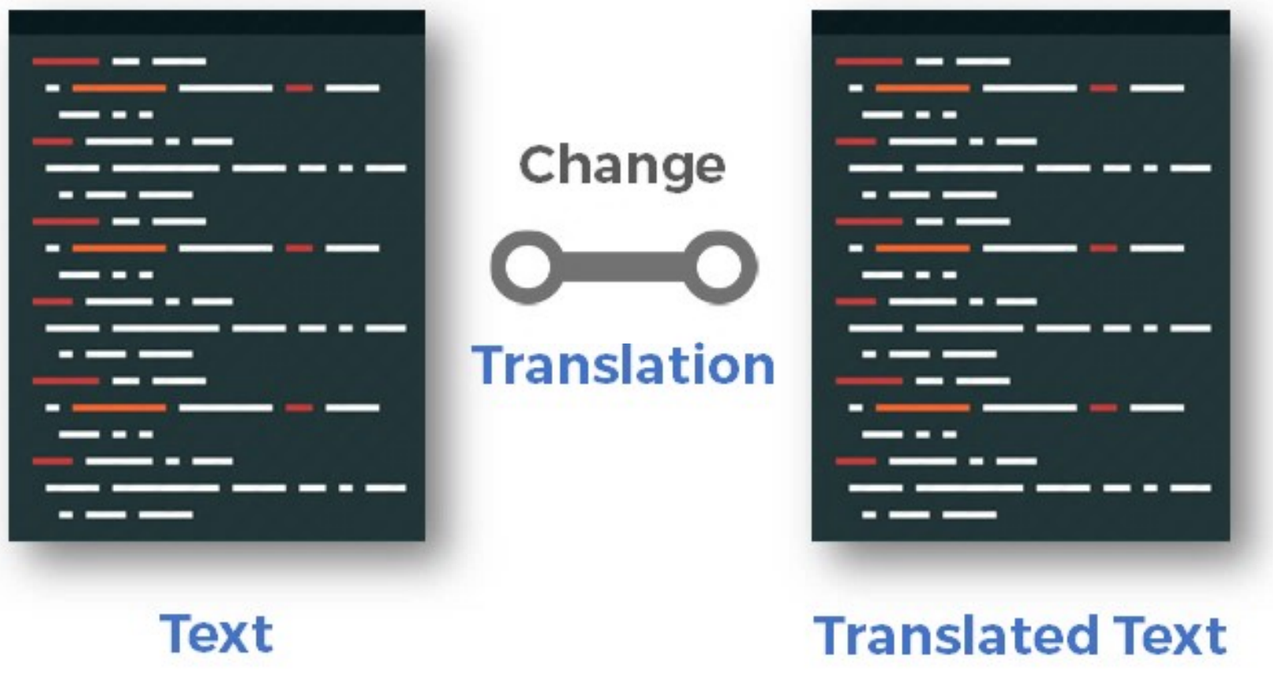
Apply Transformations



Neural Machine Translation



Neural Machine Translation



Neural Machine Translation

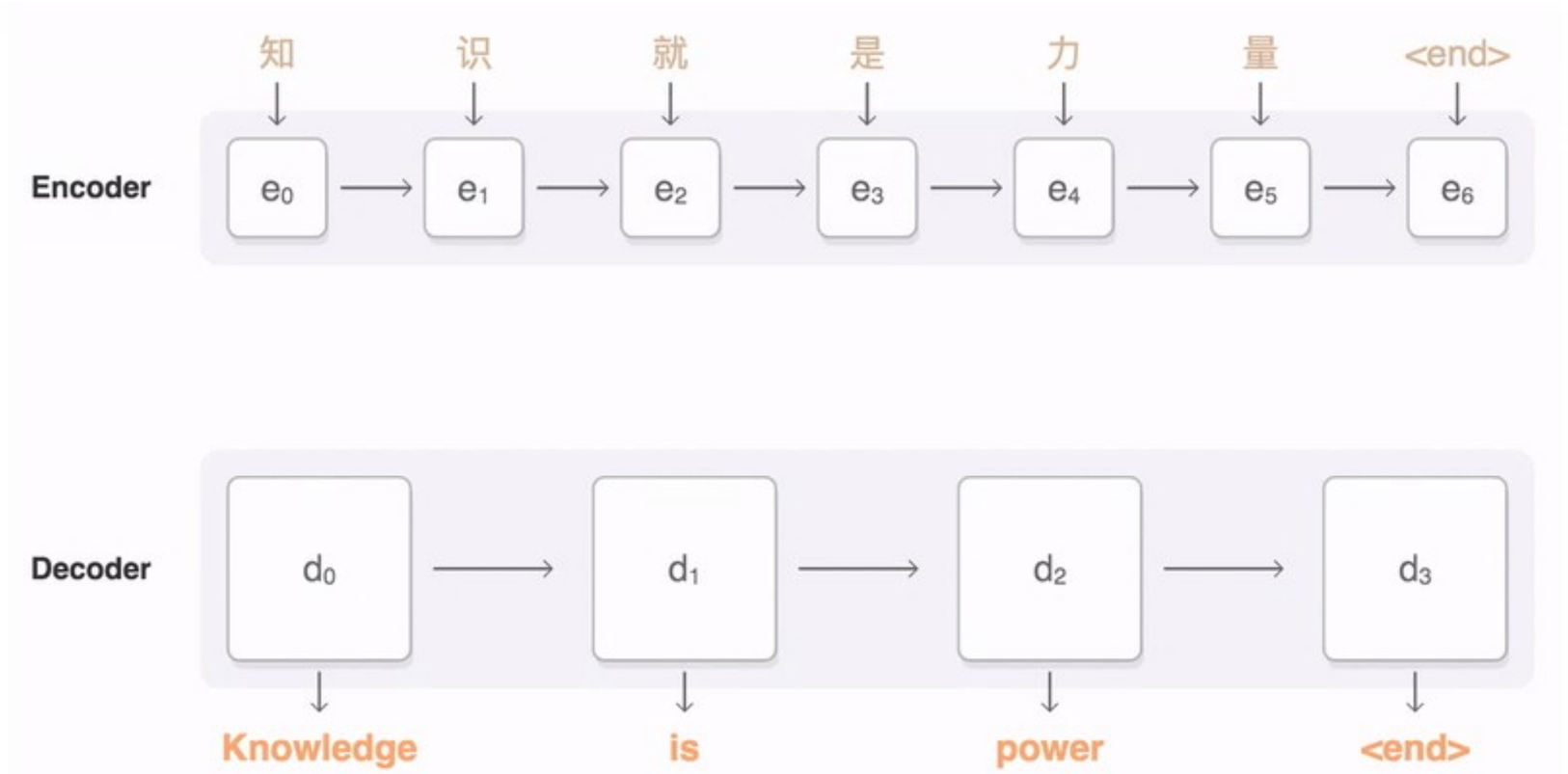
An end-to-end Deep Learning approach for automated translation

Outperforms phrase-based systems with no need of hand-engineered features

- Language Translation
- Text Summarization
- Question-Answering
- Conversational Models

Neural Machine Translation

An Encoder-Decoder architecture consisting of two Recurrent Neural Networks (RNNs) and an attention mechanism that aligns target with source tokens



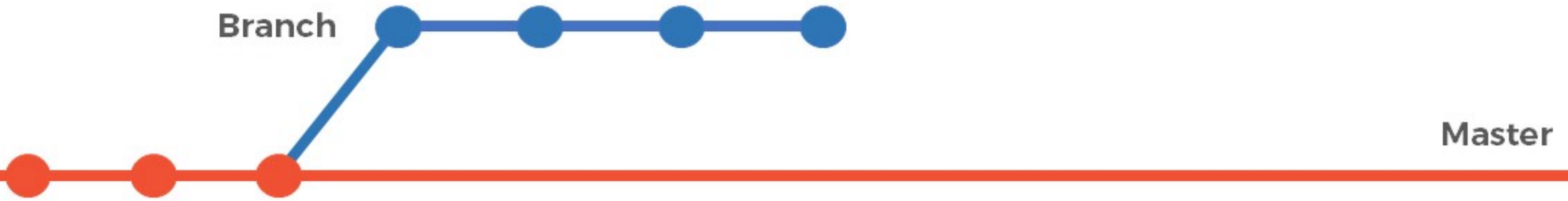
Learning Code Changes

Learning Code Changes

Master

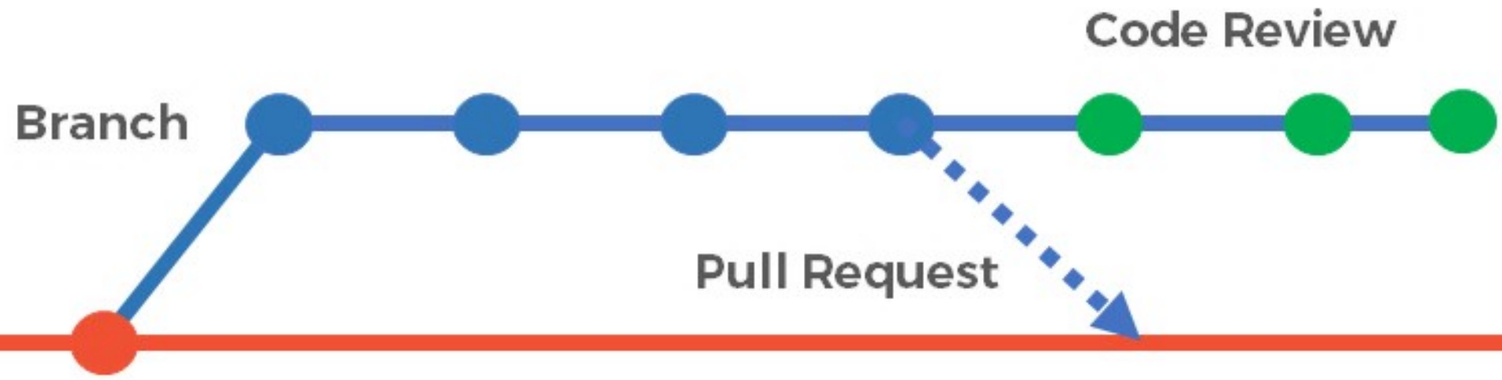
Learning Code Changes

Learning Code Changes

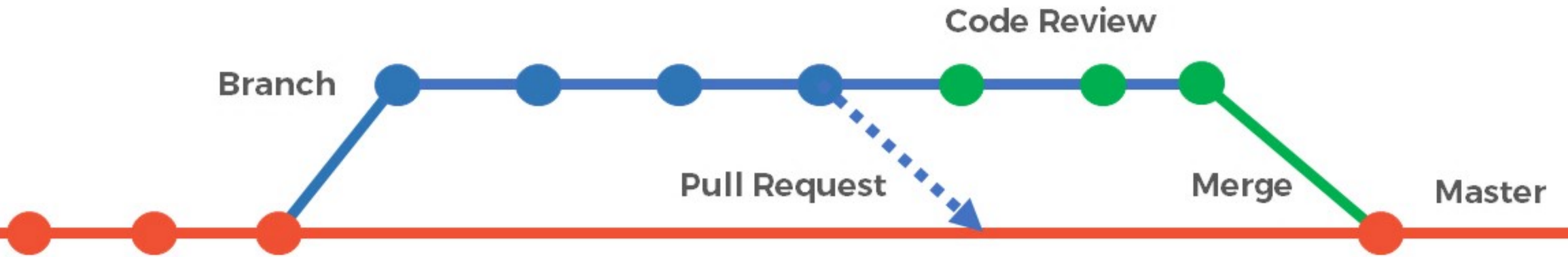


Learning Code Changes

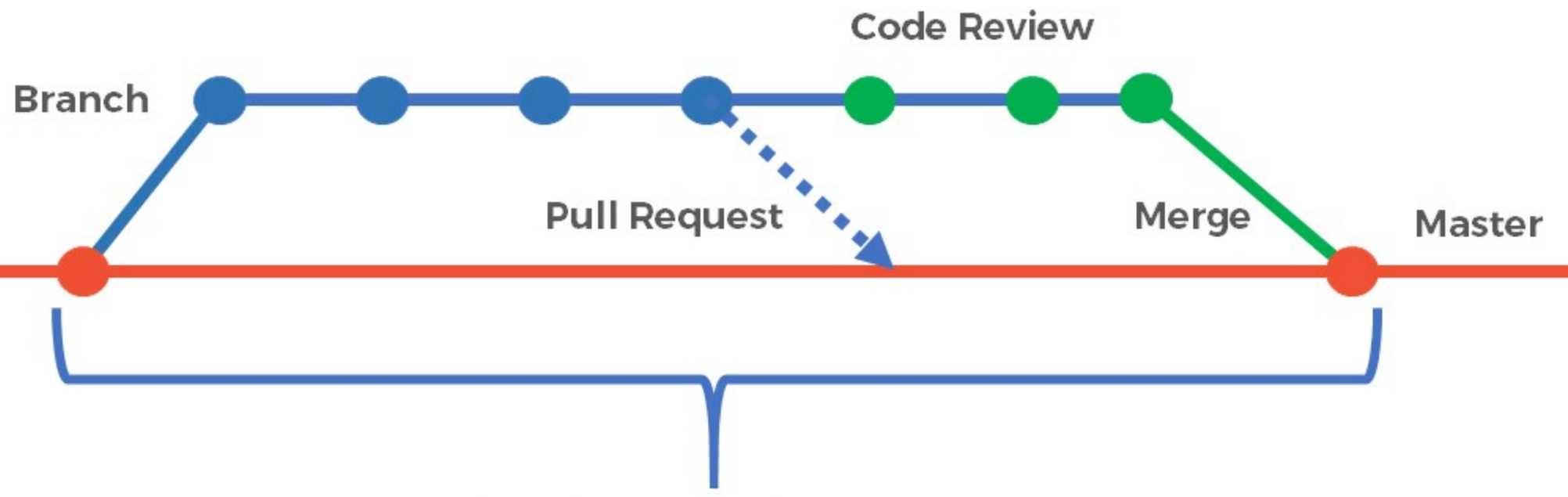
Learning Code Changes

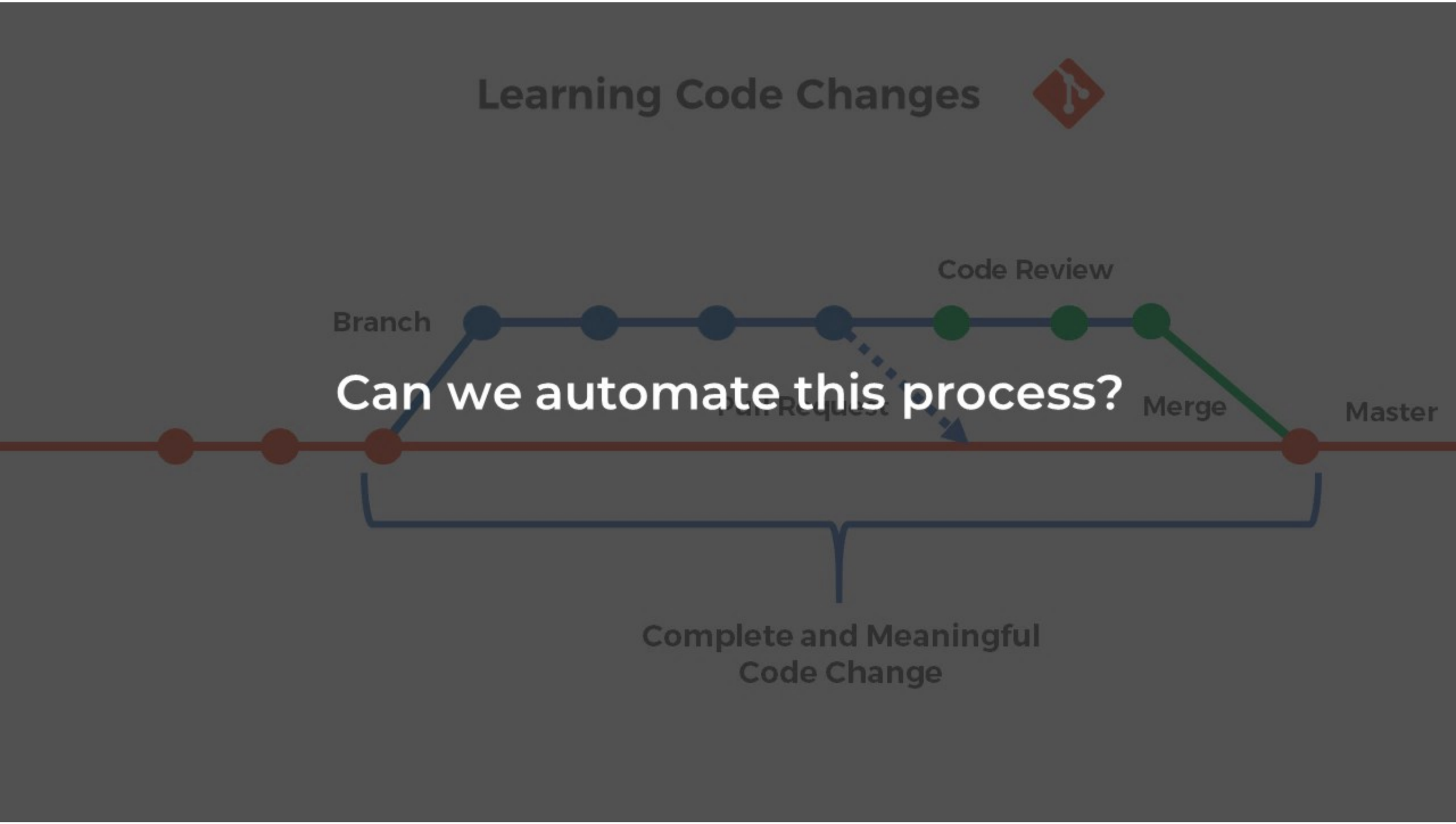


Learning Code Changes



Learning Code Changes




Learning Code Changes
Code Review
Branch
Merge
Master
Can we automate this process?
Complete and Meaningful
Code Change

Overview

1. Mine Code Changes

2. Extract Transformation Pairs

3. Code Abstraction

4. NMT Training

5. Evaluation & Taxonomy

1. Mine Code Changes

Meaningful Code Changes.
Not just commits.

1. Mine Code Changes

Meaningful Code Changes.
Not just commits.

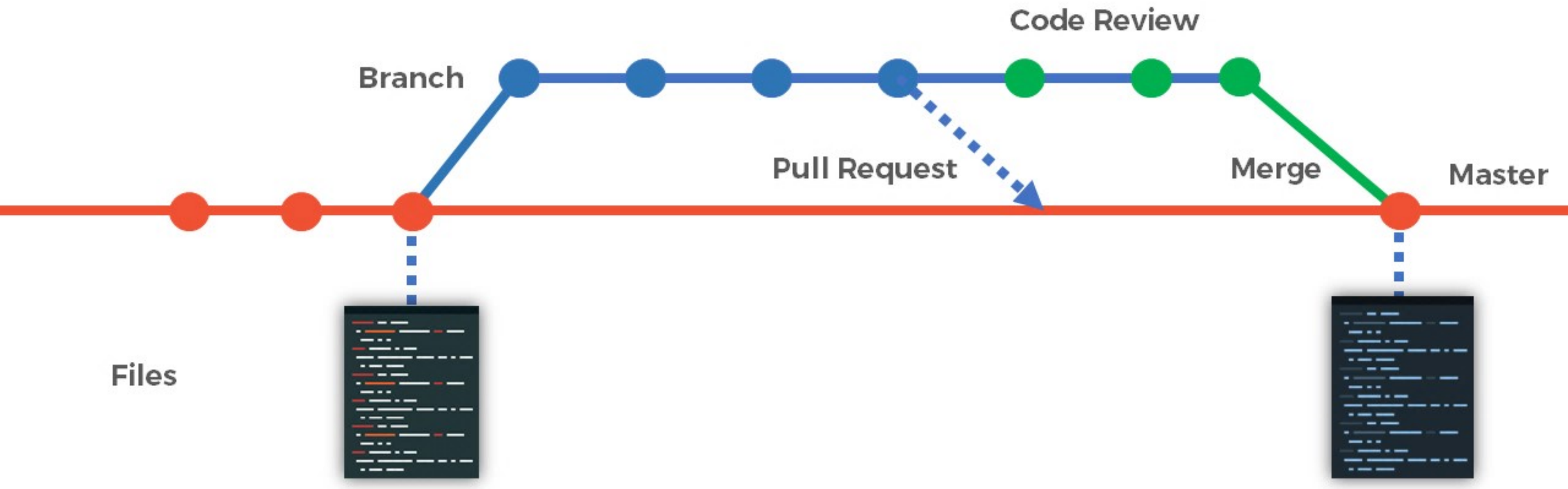
Crawler for Gerrit - Code Review System

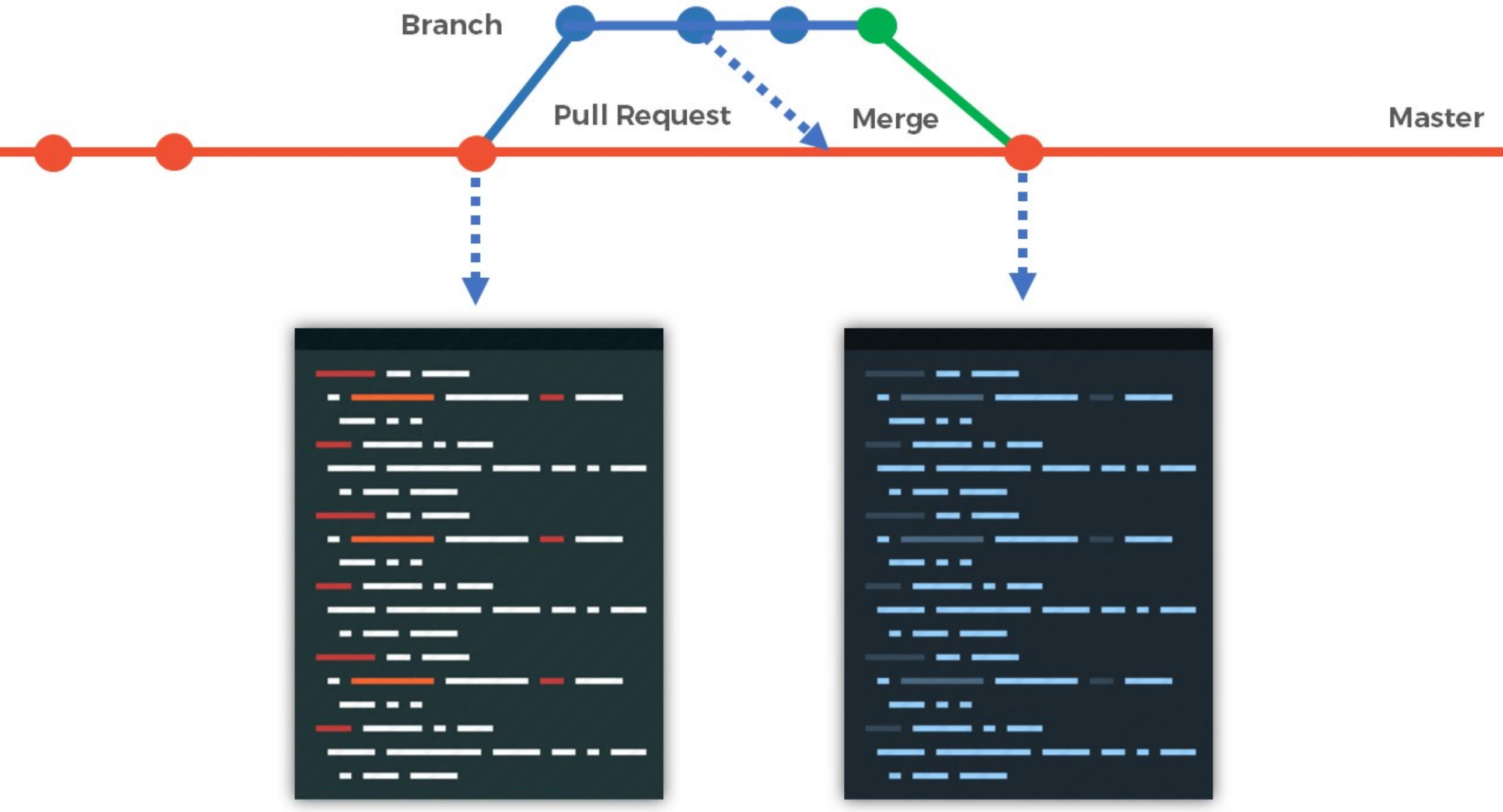
Extracts PRs : reviewed && accepted && merged

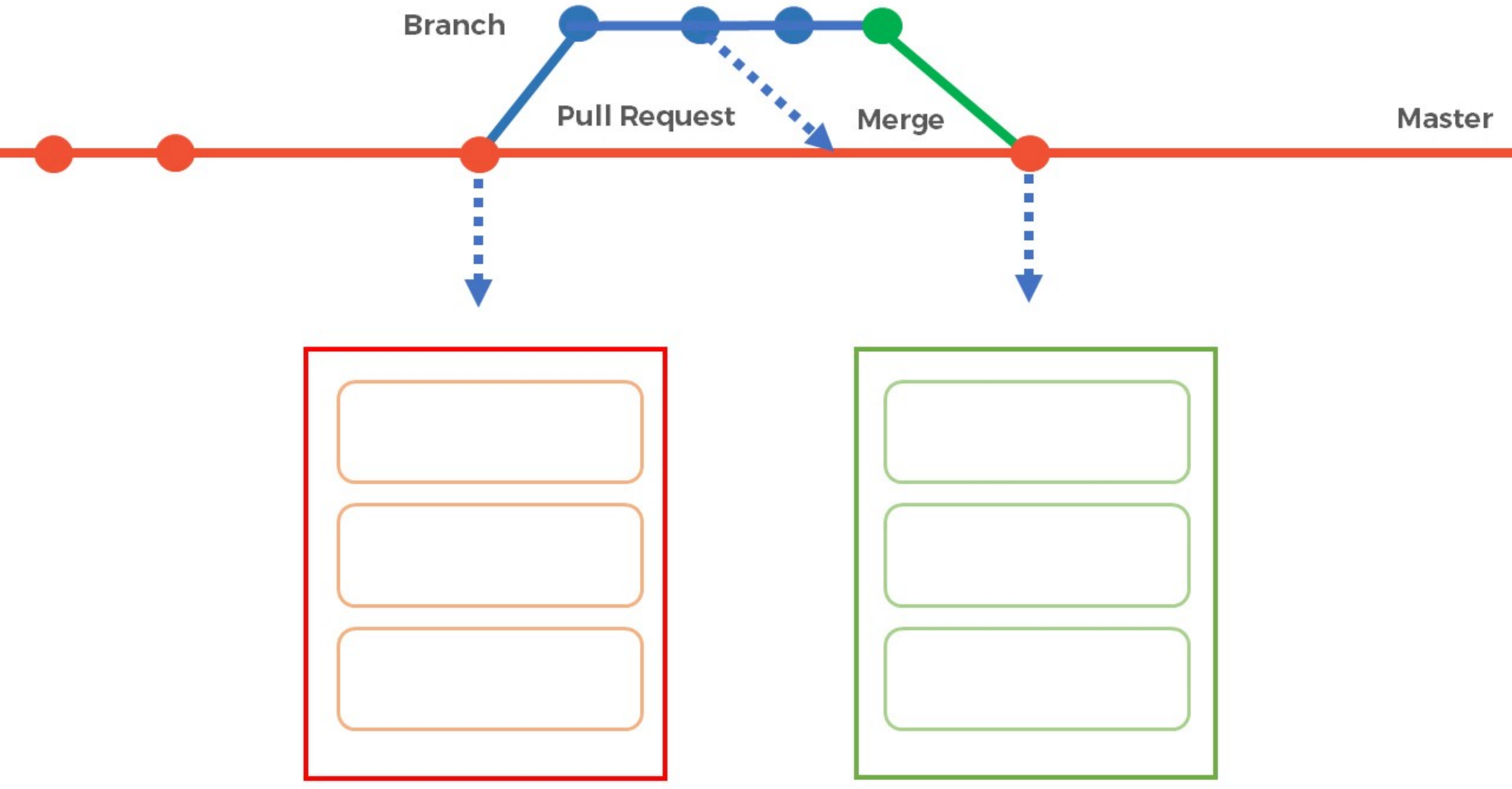
- Google
- Android
- Ovirt

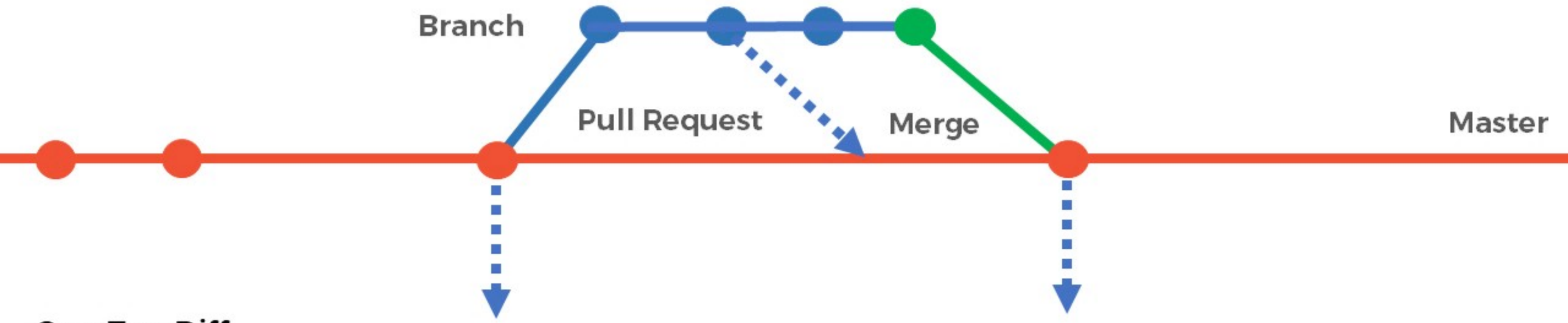
Data Extraction

Analyze every reviewed, accepted, merged PR



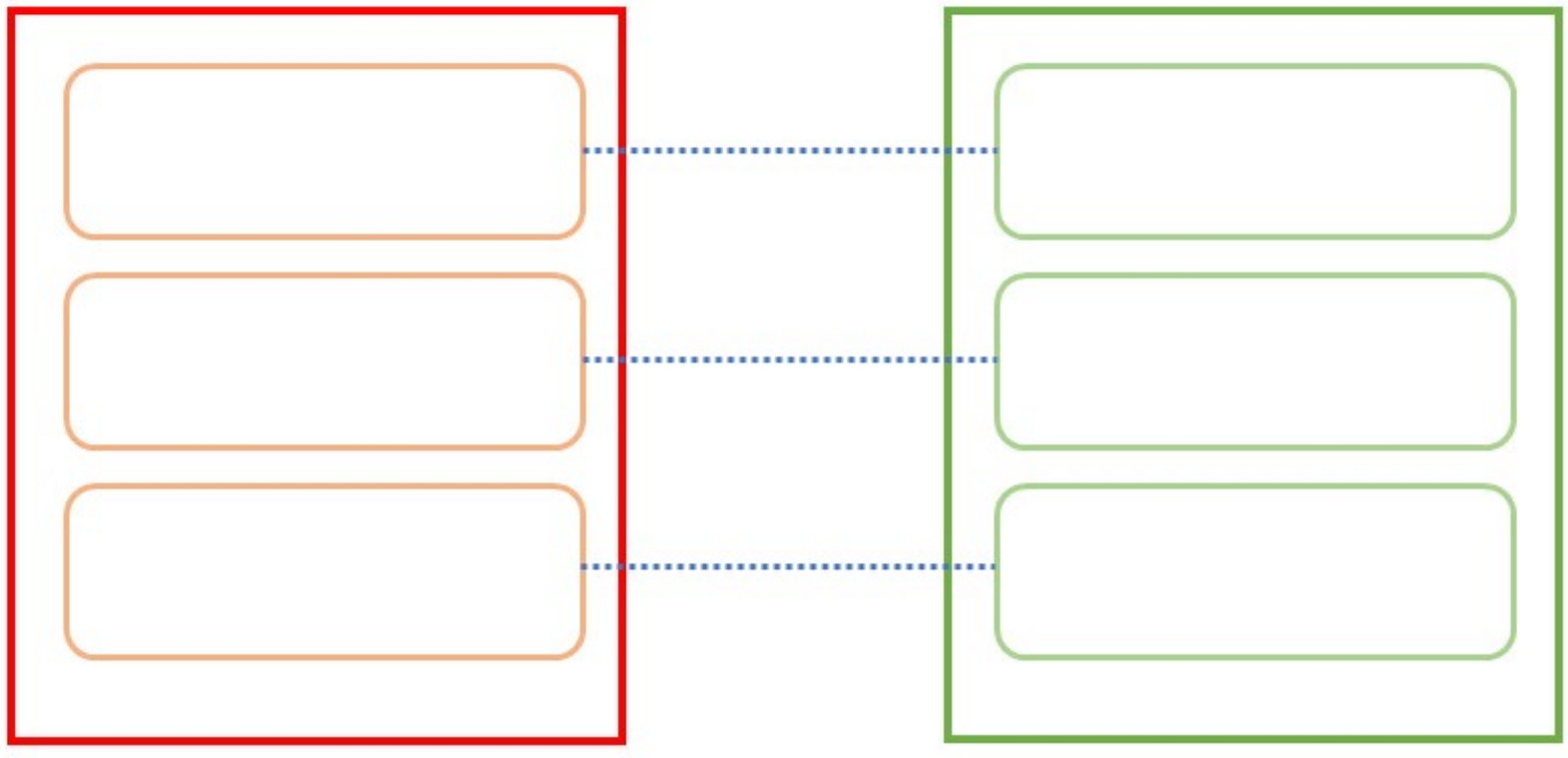

Branch
Pull Request
Merge
Master

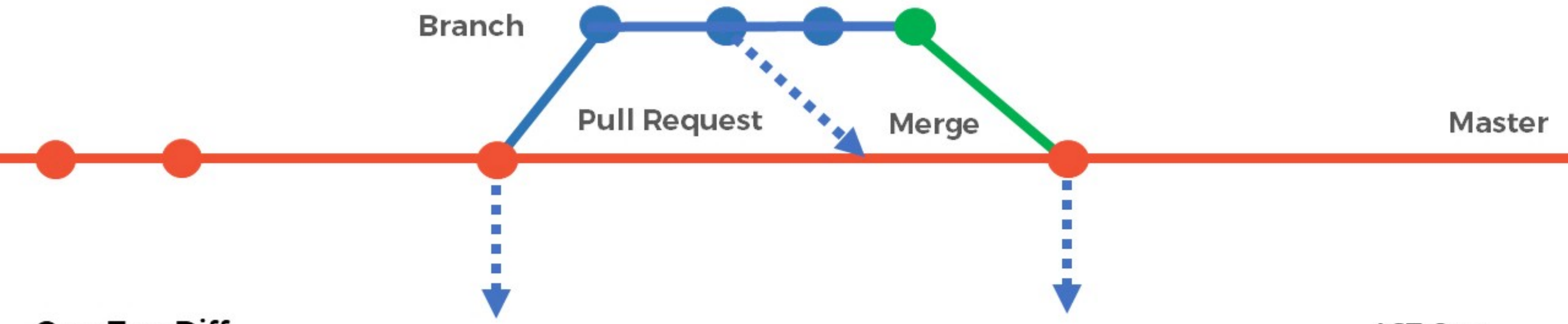

Branch
Pull Request
Merge
Master



GumTreeDiff

Method mapping
Semantic Anchors





GumTreeDiff

Method mapping
Semantic Anchors

Method-level
AST Diff

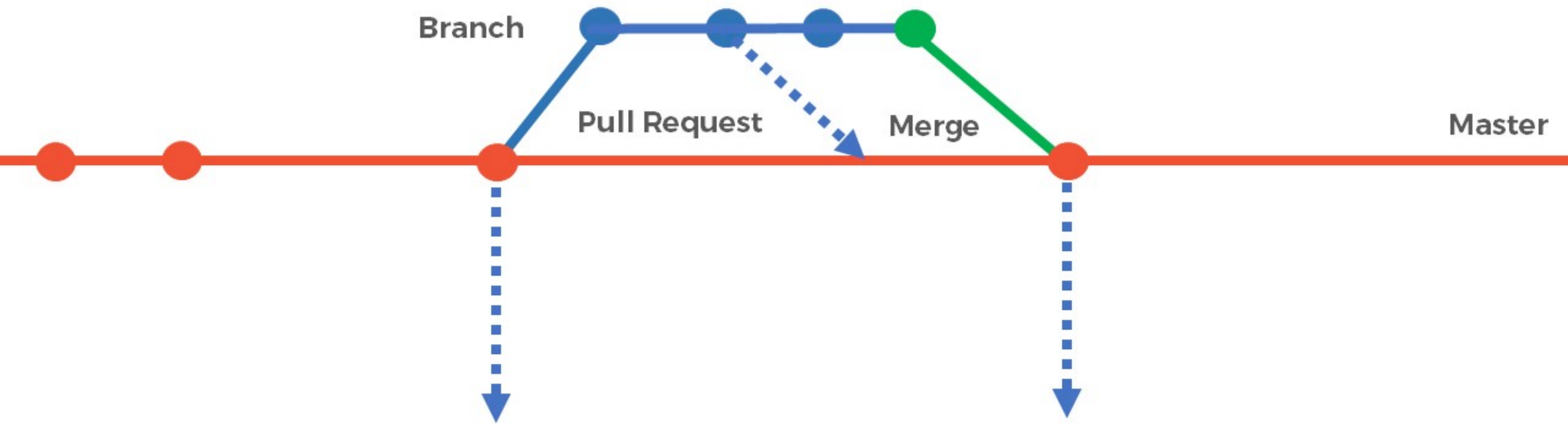
```
public void addElement ( Element<?> elem) {
        myList.add(elem);
}
```

```
public void addElement( Element<?> elem) {
if (!(myList.contains(elem)))
        myList.add(elem);
}
```

AST Ops

Insert IF-block at Method

...



Transformation Pair

$$tp = \{m_b, m_a\}$$

Method Before

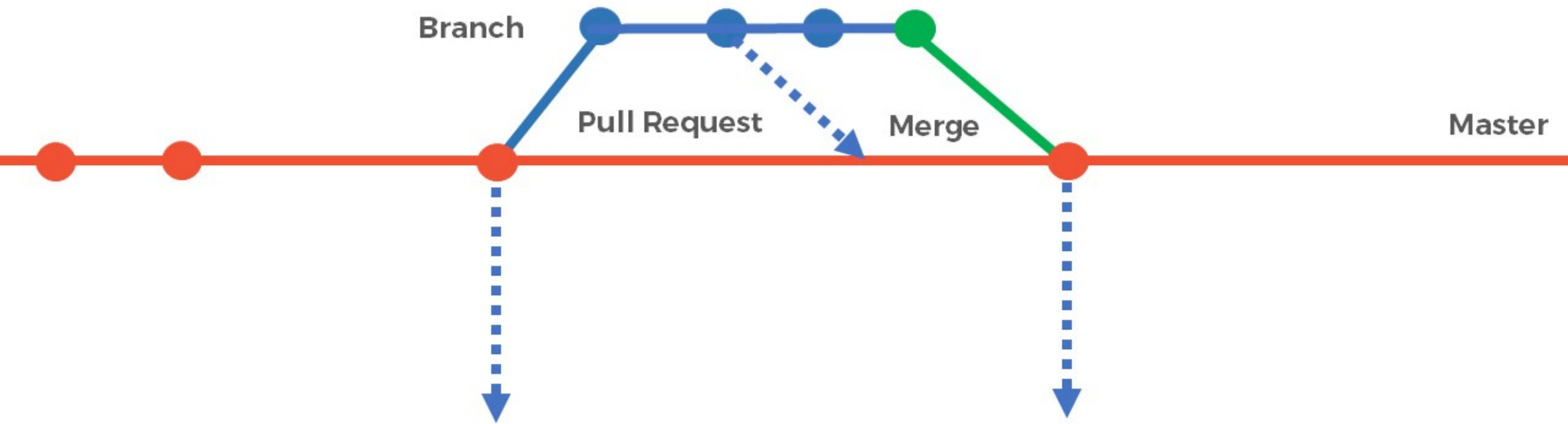
m_b

```
public void addElement ( Element<?> elem) {
        myList.add(elem);
}
```

Method After

m_a

```
public void addElement( Element<?> elem) {
if (!(myList.contains(elem)))
        myList.add(elem);
}
```



Transformation Pair

$$tp = \{m_b, m_a\}$$

Method Before

m_b

```
public void addElement ( Element<?> elem) {
        myList.add(elem);
}
```

Translation

Method After

m_a

```
public void addElement( Element<?> elem) {
if (!(myList.contains(elem)))
        myList.add(elem);
}
```

Method Granularity

$tp = \{m_b, m_a\}$

Implement a single task or functionality
Localized changes

Context for learning code changes
Variables, parameters, and method calls used in the method

Files or classes may be too large
Difficult to learn patterns of transformation

Method Granularity

$tp = \{m_b, m_a\}$

Implement a single task or functionality
Localized changes

Context for learning code changes
Variables, parameters, and method calls used in the method

Files or classes may be too large
Difficult to learn patterns of transformation

Limitations

No Changes outside Method body & signature
Modifications on class-level are not considered

Each Method transformed independently
Modifications on multiple methods at the same time

Vocabulary

Oxford English Dictionary contains entries for 171,476 words

Vocabulary

Oxford English Dictionary contains entries for 171,476 words

Developers are not constrained by any vocabulary

Code Abstraction

Goal: reduce Vocabulary

Source Code

```
public void addElement ( Element <?> elem) { if ( myList.size() > 0) { myList.add(elem); } }
```

Code Abstraction

Goal: reduce Vocabulary

Source Code

```
public void addElement ( Element <?> elem) { if ( myList.size() > 0) { myList.add(elem); } }
```

Abstracted code

```
public void        (       <?>      ) { if (           () >  ) {          (     ); } }
```

- Java Keywords and separators

Code Abstraction

Goal: reduce Vocabulary

Source Code

```
public void addElement ( Element <?> elem) { if ( myList.size() > 0) { myList.add(elem); } }
```

Abstracted code

```
public void          (        <?>      ) { if (      .size() > 0) {      .add(      ); } }
```

- Java Keywords and separators
- Idioms: frequent identifiers and literals (e.g. size, add, 0)

Code Abstraction

Goal: reduce Vocabulary

Source Code

```java
public void addElement ( Element <?> elem) { if ( myList.size() > 0) { myList.add(elem); } }
```

Abstracted code

```java
public void METHOD_1 ( TYPE_1 <?> VAR_1) { if ( VAR_2.size() > 0) { VAR_2.add(VAR_1); } }
```

- Java Keywords and separators
- Idioms: frequent identifiers and literals (e.g. size, add, 0)
- IDs : replace identifiers and literals with typified IDs (e.g., METHOD, TYPE, VAR, INT, STRING, etc.)

Code Abstraction

Goal: reduce Vocabulary

Idioms

Top-300 Most Frequent Identifiers and Literals

- **Variables** : `i, j, index, foo`
- **Type** : `java.lang.String, android.widget.Button`
- **APIs** : `toString(), add(), indexOf()`
- **Integer** : `0, 1, 16`
- **Float** : `0.1, 0.5, 1.0`
- **String** : “\n”, “[!?,]”
- **Char** : “a”, “;”, “b”

Typified IDs

Java Lexer + Parser to identify the type

- VAR_#
- TYPE_#
- METHOD_#
- INT_#
- FLOAT_#
- STRING_#
- CHAR_#

Example

Input Code

```
public void addElement ( Element <?> elem) { myList.add(elem); }
```

Input Code

```
public void addElement ( Element <?> elem) { myList.add(elem); }
```

Input Code

```
public void addElement ( Element <?> elem) { myList.add(elem); }
```

Abstracted Input Code

```
public void METHOD_1 ( TYPE_1 <?> VAR_1) { VAR_2.add(VAR_1); }
```

Mapping

ID	Value
METHOD_1	addElement
TYPE_1	Element
VAR_1	elem
VAR_2	myList

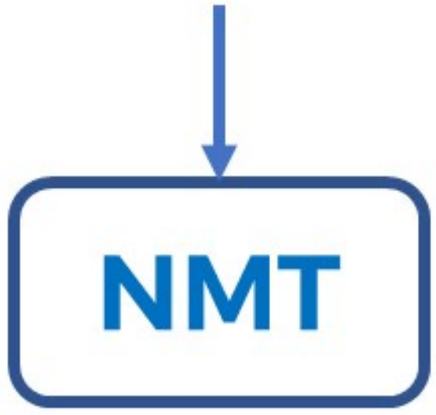
Input Code

```java
public void addElement ( Element <?> elem) { myList.add(elem); }
```

Abstracted Input Code

```java
public void METHOD_1 ( TYPE_1 <?> VAR_1) { VAR_2.add(VAR_1); }
```

Neural Machine Translation



Mapping

ID	Value
METHOD_1	addElement
TYPE_1	Element
VAR_1	elem
VAR_2	myList

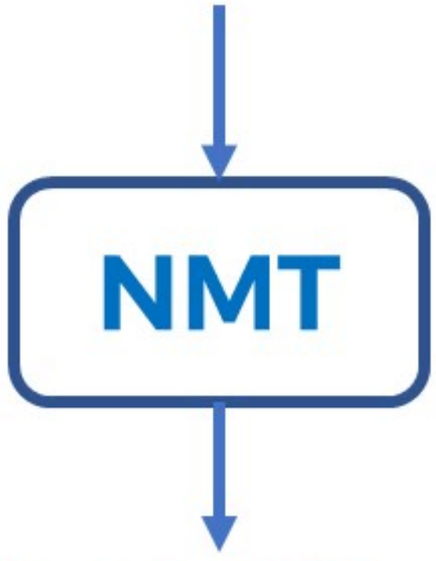
Input Code

```java
public void addElement ( Element <?> elem) { myList.add(elem); }
```

Abstracted Input Code

```java
public void METHOD_1 ( TYPE_1 <?> VAR_1) { VAR_2.add(VAR_1); }
```

Neural Machine Translation



Mapping

ID	Value
METHOD_1	addElement
TYPE_1	Element
VAR_1	elem
VAR_2	myList

Abstracted Changed Code

```java
public void METHOD_1 ( TYPE_1 <?> VAR_1) { if (! VAR_2.contains(VAR_1)) VAR_2.add(VAR_1); }
```

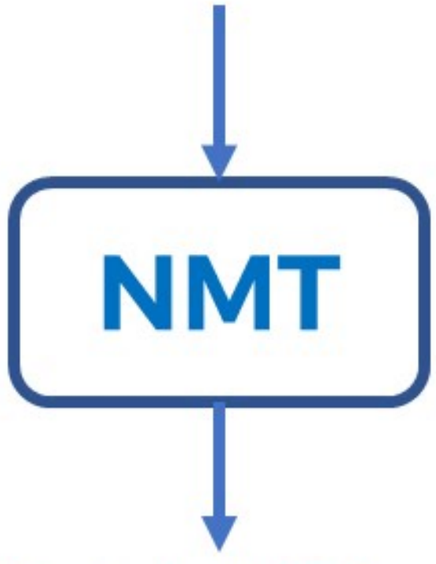
Input Code

```java
public void addElement ( Element <?> elem) { myList.add(elem); }
```

Abstracted Input Code

```java
public void METHOD_1 ( TYPE_1 <?> VAR_1) { VAR_2.add(VAR_1); }
```

Neural Machine Translation



Mapping

ID	Value
METHOD_1	addElement
TYPE_1	Element
VAR_1	elem
VAR_2	myList

Abstracted Changed Code

```java
public void METHOD_1 ( TYPE_1 <?> VAR_1) { if (! VAR_2.contains(VAR_1)) VAR_2.add(VAR_1); }
```

Concrete Code

```java
public void addElement ( Element <?> elem) { if (! myList.contains(elem)) myList.add(elem); }
```

Input Code

```
public void addElement ( Element <?> elem) { myList.add(elem); }
```

Abstracted Input Code

```
public void METHOD_1 ( TYPE_1 <?> VAR_1) { VAR_2.add(VAR_1); }
```

Neural Machine Translation

What if we can't map everything back?

Mapping

ID	Value
METHOD_1	addElement
TYPE_1	Element
VAR_1	elem
VAR_2	myList

Abstracted Changed Code

```
public void METHOD_1 ( TYPE_1 <?> VAR_1) { if (! VAR_2.contains(VAR_1)) VAR_2.add(VAR_1); }
```

Concrete Code

```
public void addElement ( Element <?> elem) { if (! myList.contains(elem)) myList.add(elem); }
```

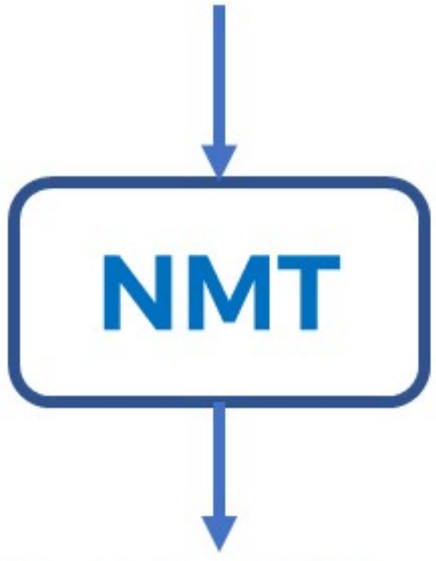
Input Code

```
public void addElement ( Element <?> elem) { myList.add(elem); }
```

Abstracted Input Code

```
public void METHOD_1 ( TYPE_1 <?> VAR_1) { VAR_2.add(VAR_1); }
```

Neural Machine Translation



Mapping

ID	Value
METHOD_1	addElement
TYPE_1	Element
VAR_1	elem
VAR_2	myList

Abstracted Changed Code

```
public void METHOD_1 ( TYPE_1 <?> VAR_1) { if (! VAR_2.contains(VAR_3)) VAR_2.add(VAR_1); }
```

Concrete Code

```
public void addElement ( Element <?> elem) { if (! myList.contains(???)) myList.add(elem); }
```

Input Code

```
public void addElement ( Element <?> elem) { myList.add(elem); }
```

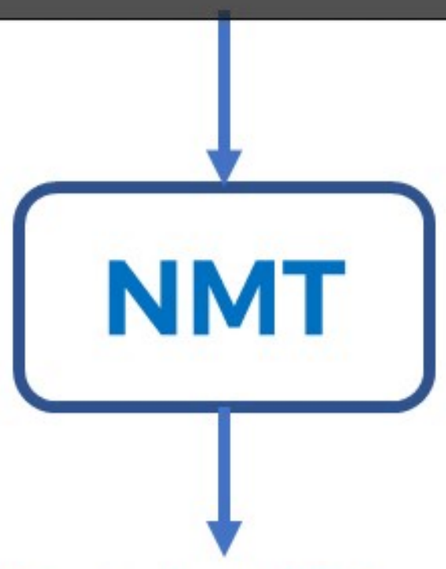
Abstracted Input Code

```
public void METHOD_1 ( TYPE_1 <?> VAR_1) { VAR_2.add(VAR_1); }
```

Before you say it...

Copy Mechanism can't help here!

Neural Machine Translation



Mapping

ID	Value
METHOD_1	addElement
TYPE_1	Element
VAR_1	elem
VAR_2	myList

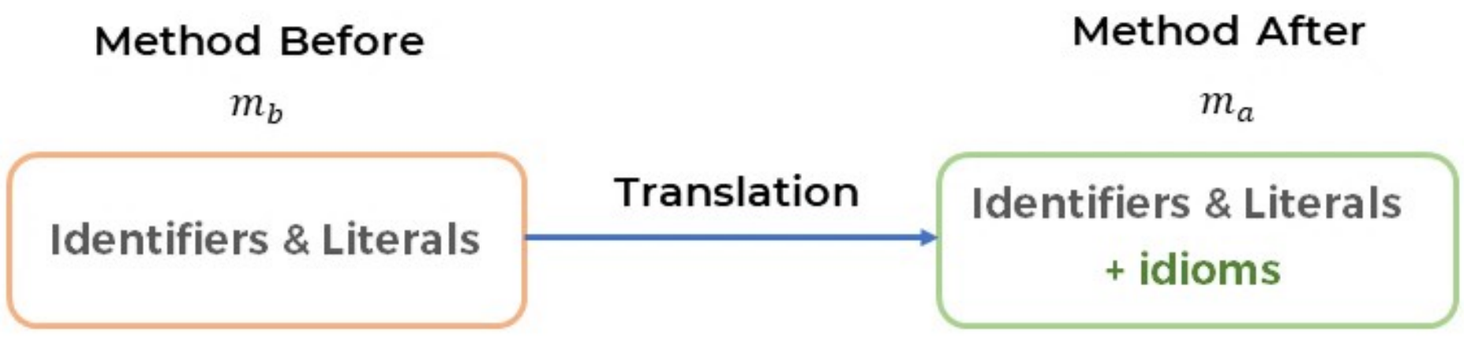
Abstracted Changed Code

```
public void METHOD_1 ( TYPE_1 <?> VAR_1) { if (! VAR_2.contains(VAR_3)) VAR_2.add(VAR_1); }
```

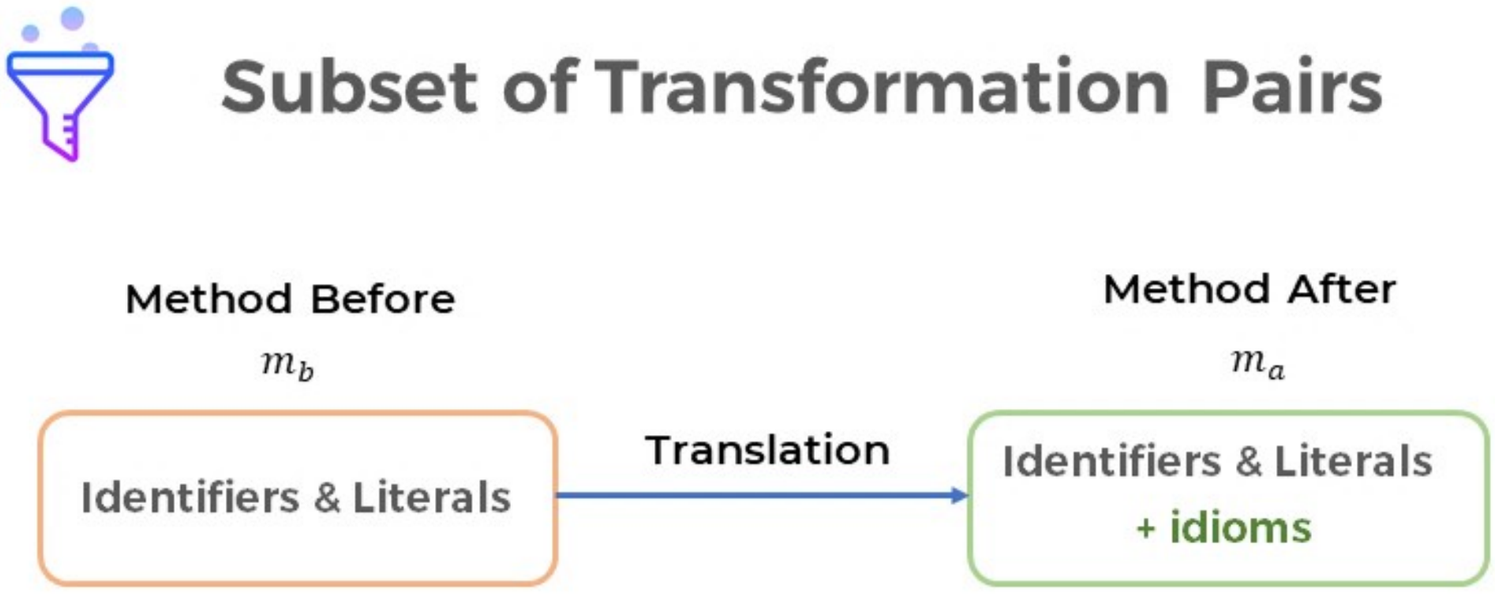
Concrete Code

```
public void addElement ( Element <?> elem) { if (! myList.contains(???)) myList.add(elem); }
```

Subset of Transformation Pairs



Subset of Transformation Pairs



Small Methods
- No longer than 50 tokens

Medium Methods
- Between 50 and 100 tokens

Filtering

- Equal Abstract Code before/after
- Duplicates (abstract code) in datasets

Dataset	M_{small}	M_{medium}
Google	2,165	2,286
Android	4,162	3,617
Ovirt	4,456	5,088
All	10,783	10,991

Neural Machine Translation

Model

Encoder + Decoder

Learns $P(y_1, \dots, y_m | x_1, \dots, x_n) = P(am_a | am_b)$

Encoder

Encodes a sequence of terms x in a vector representation h

- $x = am_b = (x_1, \dots, x_n)$
- $h = (h_1, \dots, h_n)$

Bi-directional RNN Encoder

- $h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$

Decoder

Decodes h into a sequence of terms y

- $y = am_a = (y_1, \dots, y_m)$

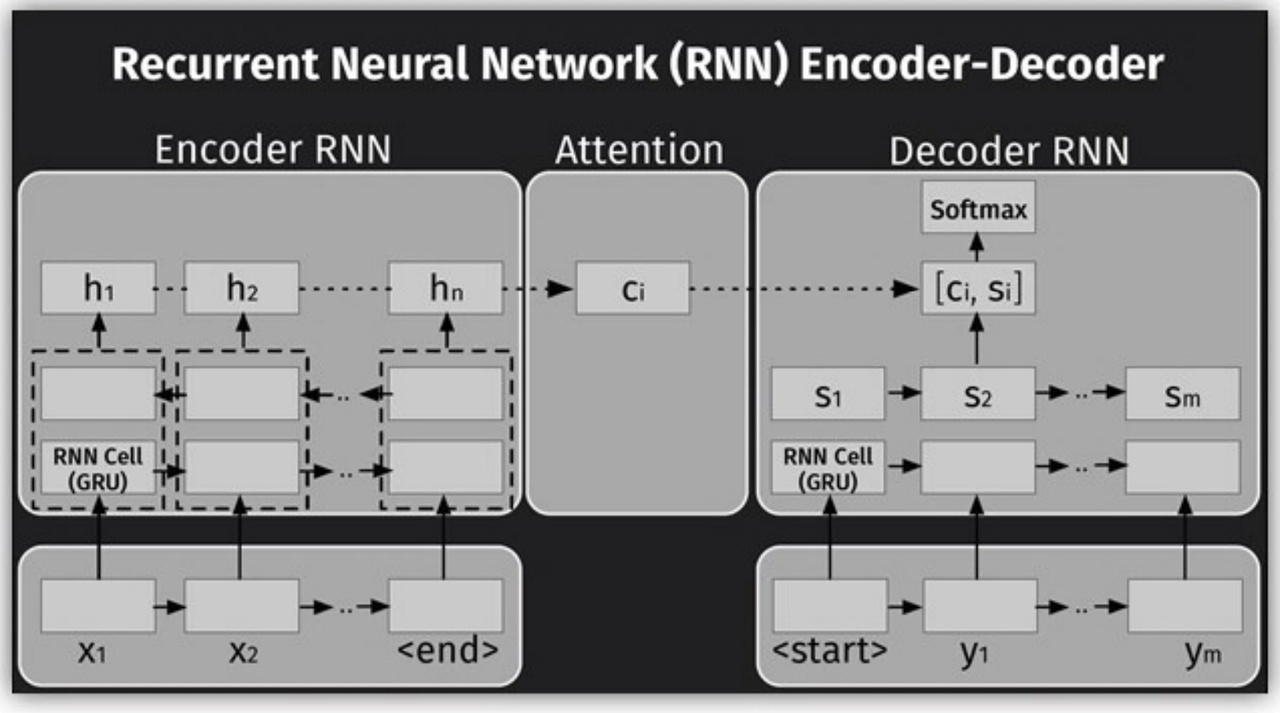
Decoding term y_i

- **Recurrent state s_i**
- **Previous terms $y_1, \dots, y_{i-1}$**
- **Context vector c_i**

Attention Mechanism

Attention to particular parts of the input sequence

- $c_i = \sum_{t=1}^{n} a_{it} h_t$



Neural Machine Translation

Hyperparameters

10 configurations

RNN Cells
- LSMT
- GRU

Layers
- 1
- 2
- 4

Units
- 256
- 512

Embedding Size
- 256
- 512

Overfitting

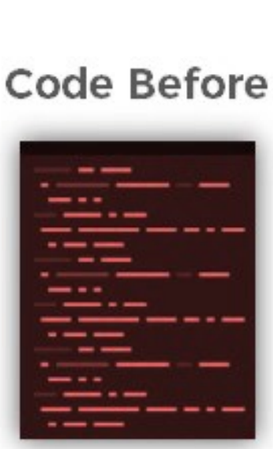
Early stopping is used by observing the loss function on the Validation set

ID	Embedding	Encoder		Decoder		Cell
		Layers	Units	Layers	Units	
1	256	1	256	2	256	GRU
2	256	1	256	2	256	LSTM
3	256	2	256	4	256	GRU
4	256	2	256	4	256	LSTM
5	256	2	512	4	512	GRU
6	256	2	512	4	512	LSTM
7	512	2	512	4	512	GRU
8	512	2	512	4	512	LSTM
9	512	1	256	2	256	GRU
10	**512**	**1**	**256**	**2**	**256**	**LSTM**

Neural Machine Translation

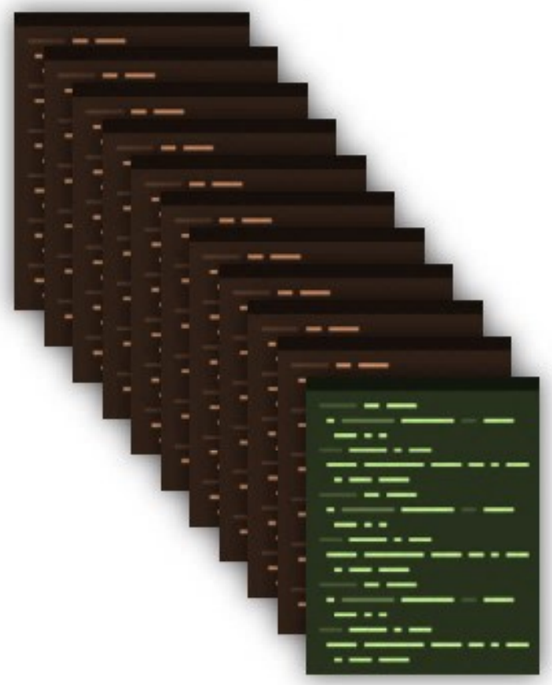
Generate many different Translations

Code Before



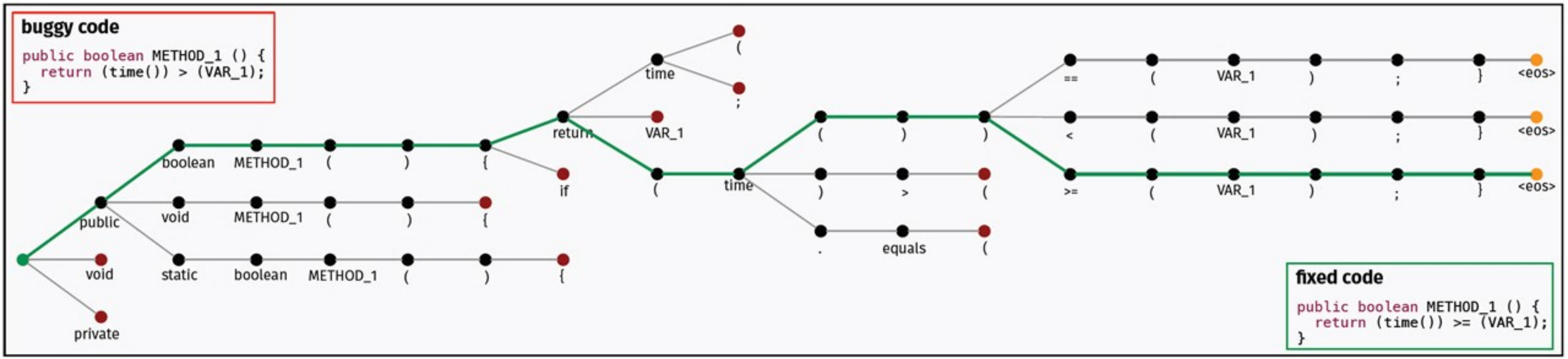
Translations

Potential Changes



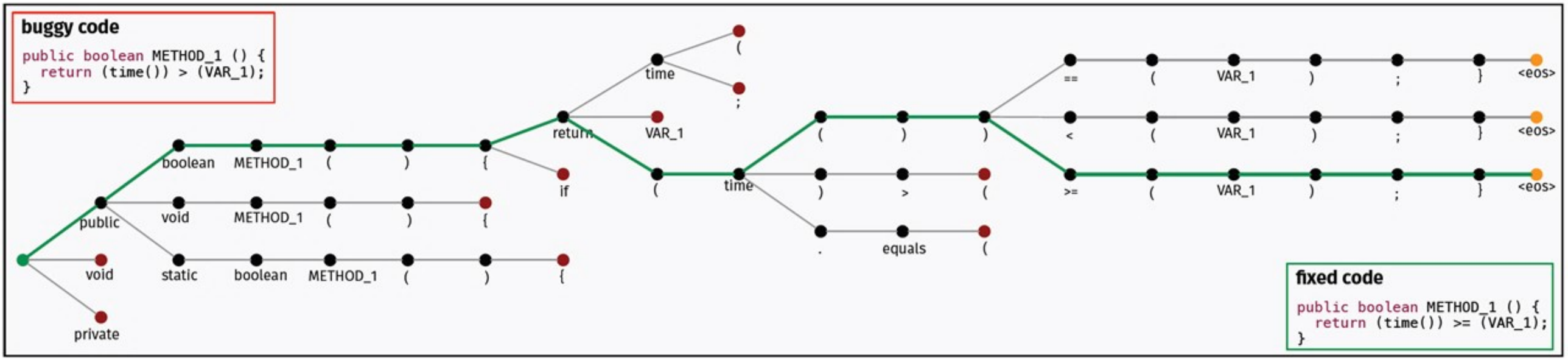
Beam Search

Keep track of k different hypotheses



Beam Search

Keep track of k different hypotheses

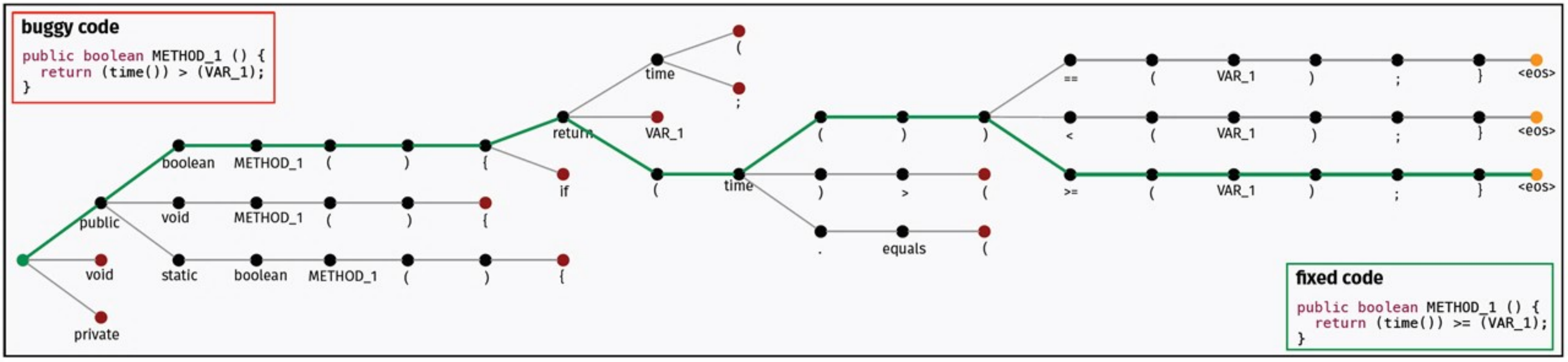


Time Step t

Hypotheses Set: $\mathcal{H}_t = \{(\tilde{y}_1^1, \ldots, \tilde{y}_t^1), (\tilde{y}_1^2, \ldots, \tilde{y}_t^2), \ldots, (\tilde{y}_1^k, \ldots, \tilde{y}_t^k)\}$

Beam Search

Keep track of k different hypotheses



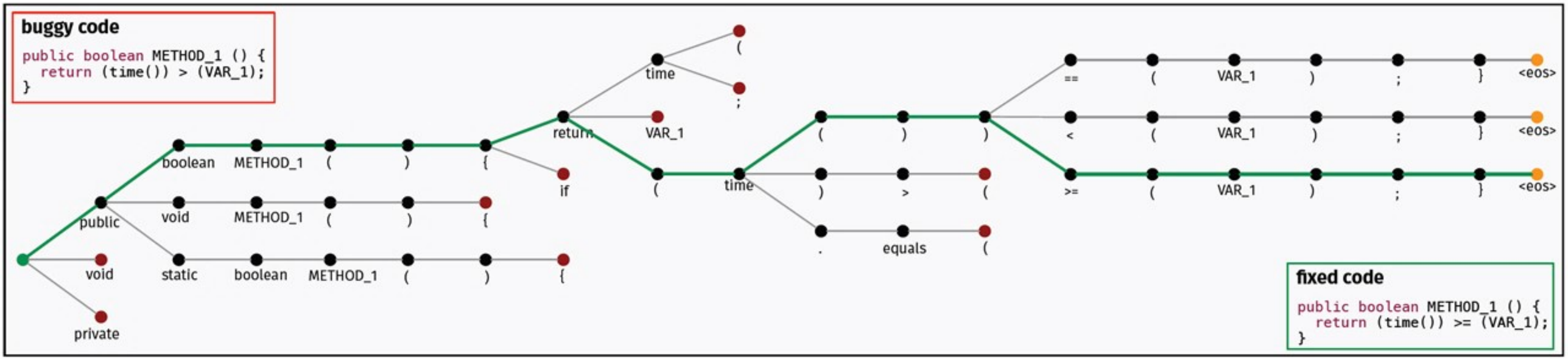
Time Step t

Hypotheses Set: $\mathcal{H}_t = \{(\tilde{y}_1^1, \ldots, \tilde{y}_t^1), (\tilde{y}_1^2, \ldots, \tilde{y}_t^2), \ldots, (\tilde{y}_1^k, \ldots, \tilde{y}_t^k)\}$

Single sentence hypothesis

Beam Search

Keep track of k different hypotheses



Time Step t

Hypotheses Set: $\mathcal{H}_t = \{(\tilde{y}_1^1, \ldots, \tilde{y}_t^1), (\tilde{y}_1^2, \ldots, \tilde{y}_t^2), \ldots, (\tilde{y}_1^k, \ldots, \tilde{y}_t^k)\}$

Single sentence hypothesis

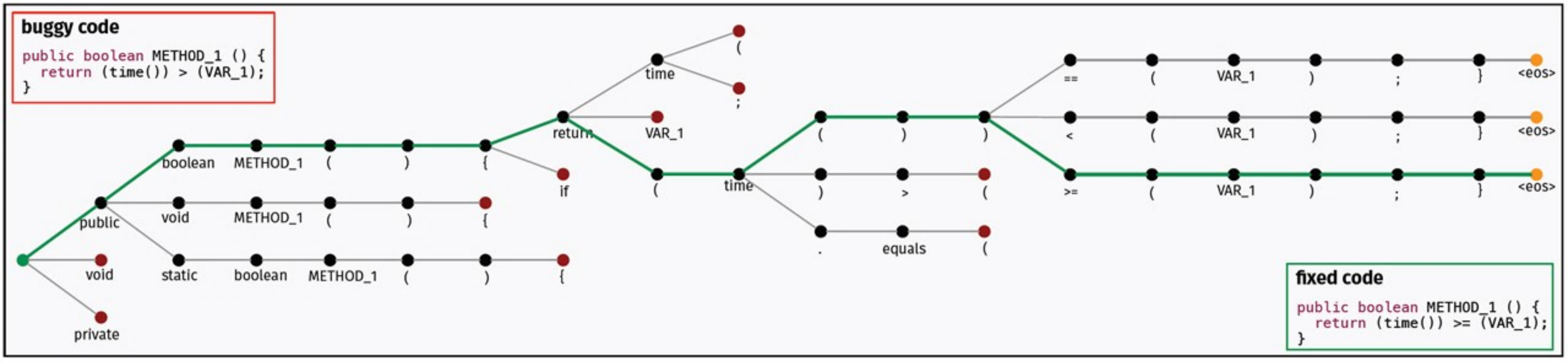
Time Step $t+1$

Candidate Set: $C_{t+1} = \bigcup_{i=1}^{k} \{(\tilde{y}_1^1, \ldots, \tilde{y}_t^1, v_1), \ldots, (\tilde{y}_1^k, \ldots, \tilde{y}_t^k, v_{|V|})\}$

Size: $k \cdot |V|$

Beam Search

Keep track of k different hypotheses



Beam Width : k

Suggested Translations

$$k = \{1, 5, 10\}$$

Experiments & Results

Experiments & Results

Successful Changes

- Perfect Predictions

Types of Changes

- Taxonomy of changes
- Qualitative Examples

Predictions

How many changes can the model perfectly predict?

Dataset	Beam	M_{small}	M_{medium}
Google	1	10 (4.62%)	7 (3.07%)
	5	17 (7.87%)	13 (5.70%)
	10	20 (9.25%)	17 (7.45%)
Android	1	40 (9.61%)	51 (14.12%)
	5	71 (17.06%)	73 (20.22%)
	10	79 (18.99%)	76 (21.05%)
Ovirt	1	55 (12.35%)	60 (11.78%)
	5	93 (20.89%)	90 (17.68%)
	10	113 (25.39%)	102 (20.03%)
All	1	228 (21.16%)	178 (16.21%)
	5	349 (32.40%)	306 (27.86%)
	10	388 (36.02%)	334 (30.41%)

Predictions

How many changes can the model perfectly predict?

Dataset	Beam	M_{small}	M_{medium}
Google	1	10 (4.62%)	7 (3.07%)
	5	17 (7.87%)	13 (5.70%)
	10	20 (9.25%)	17 (7.45%)
Android	1	40 (9.61%)	51 (14.12%)
	5	71 (17.06%)	73 (20.22%)
	10	79 (18.99%)	76 (21.05%)
Ovirt	1	55 (12.35%)	60 (11.78%)
	5	93 (20.89%)	90 (17.68%)
	10	113 (25.39%)	102 (20.03%)
All	1	228 (21.16%)	178 (16.21%)
	5	349 (32.40%)	306 (27.86%)
	10	388 (36.02%)	334 (30.41%)

30% of changes are predicted in the top-10 suggested translations

Taxonomy

What types of changes can the model perform?

Qualitatively assess the types of changes

- ALL the successful predictions by the NMT model

Procedure

- Step 1: Author X manually analyzed and described the code changes;
- Step 2: Author Y discussed and validated the described changes;
- Step 3: All authors iteratively refined a Taxonomy

Taxonomy

What types of changes can the model perform?

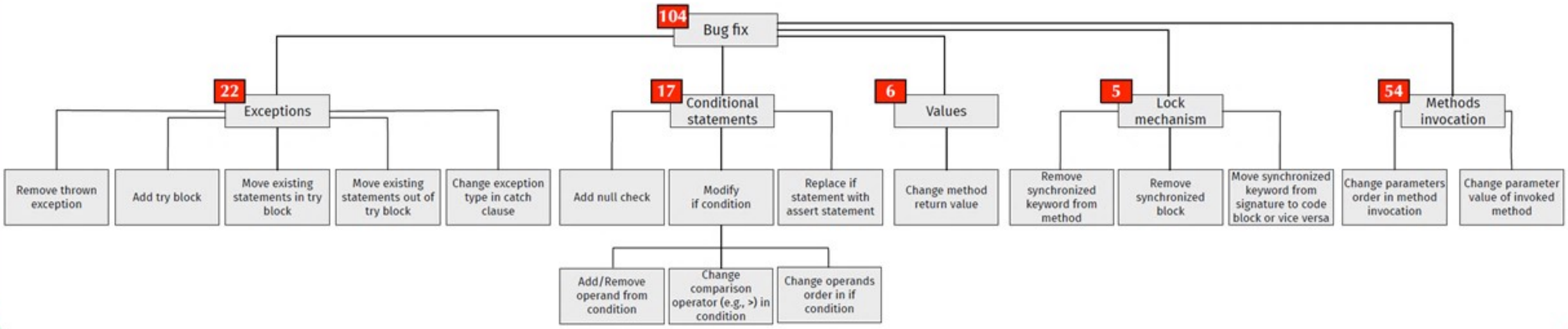
Bug-Fixes

Refactorings

Other Changes

Taxonomy

Bug-Fixes



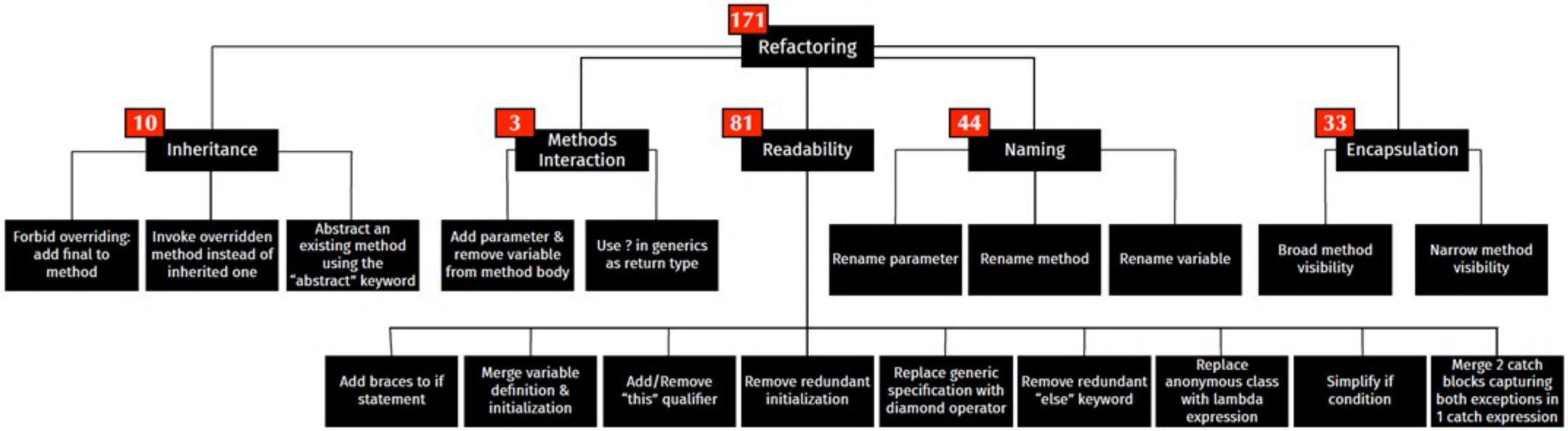
Refactorings

Other Changes

Taxonomy

Bug-Fixes

Refactorings



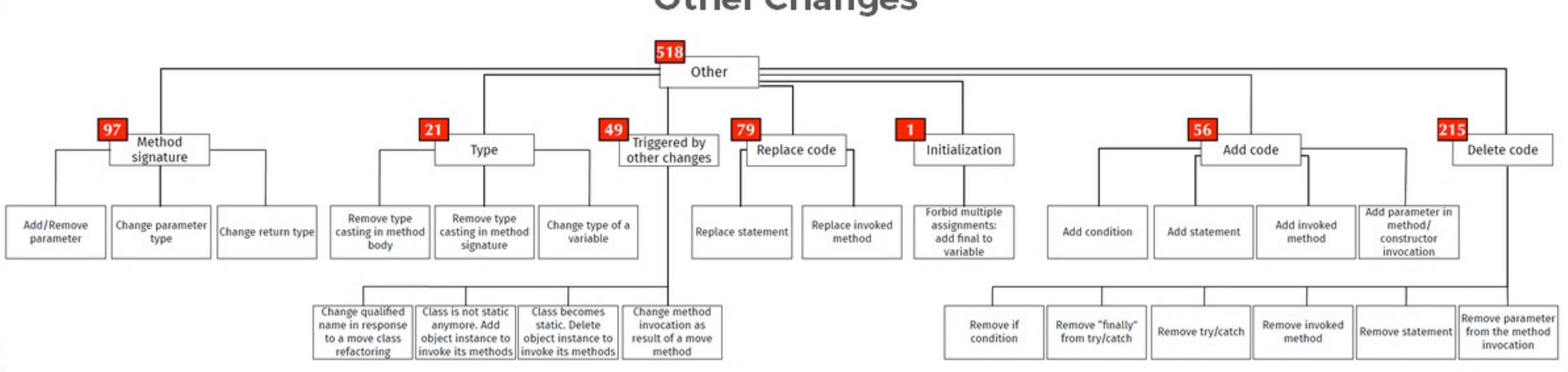
Other Changes

Taxonomy

Bug-Fixes

Refactorings

Other Changes



Examples

Refactorings

50859: use Callable functional interface

```
public boolean isDiskExist(final Guid id) {
    return execute(new Callable<Boolean>() {
        @Override
        public Boolean call() {
            try {
                Volume volume = proxy.getVolumeById(id.toString());
                return volume != null;
            } catch (OpenStackResponseException ex) {
                if (ex.getStatus() == HttpStatus.SC_NOT_FOUND) {
                    return false;
                }
                throw ex;
            }
        }
    });
}
```

```
208 public boolean isDiskExist(final Guid id) {
209     return execute(() -> {
210         try {
211             Volume volume = proxy.getVolumeById(id.toString());
212             return volume != null;
213         } catch (OpenStackResponseException ex) {
214             if (ex.getStatus() == HttpStatus.SC_NOT_FOUND) {
215                 return false;
216             }
217             throw ex;
218         }
219     });
220 }
```

https://gerrit.ovirt.org/#/c/50859/9/backend/manager/modules/bll/src/main/java/org/ovirt/engine/core/bll/storage/disk/cinder/CinderBroker.java

Examples

Bug-Fixes

283122: Detect and fix resource leakages in tests

```
public void test_getPort() throws IOException {
    DatagramSocket theSocket = new DatagramSocket();
    assertEquals("Expected -1 for remote port as not connected", -1,
            theSocket.getPort());

    // Now connect the socket and validate that we get the right port
    int portNumber = 49152; // any valid port, even if it is unreachable
    theSocket.connect(InetAddress.getLocalHost(), portNumber);
    assertEquals("getPort returned wrong value", portNumber, theSocket
            .getPort());

}
```

```
359     public void test_getPort() throws IOException {
360         try (DatagramSocket theSocket = new DatagramSocket()) {
361             assertEquals("Expected -1 for remote port as not connected", -1,
362                     theSocket.getPort());
363
364             // Now connect the socket and validate that we get the right port
365             int portNumber = 49152; // any valid port, even if it is unreachable
366             theSocket.connect(InetAddress.getLocalHost(), portNumber);
367             assertEquals("getPort returned wrong value", portNumber, theSocket
368                     .getPort());
369         }
370     }
```

https://android-review.googlesource.com/c/platform/libcore/+/283122/7/harmony-tests/src/test/java/org/apache/harmony/tests/java/net/DatagramSocketTest.java

Open Science

Online Appendix

Data, Results, Logs

- https://sites.google.com/view/learning-codechanges

Source Code

Source Code Abstraction

- https://github.com/micheletufano/src2abs

Neural Code Translator

- https://github.com/micheletufano/NeuralCodeTranslator

GitHub

On Learning Meaningful Code Changes via Neural Machine Translation

Conclusions

Learn many different types of Code Changes

Envision Recommendation Systems

- Refactoring recommenders
- Code Reviews