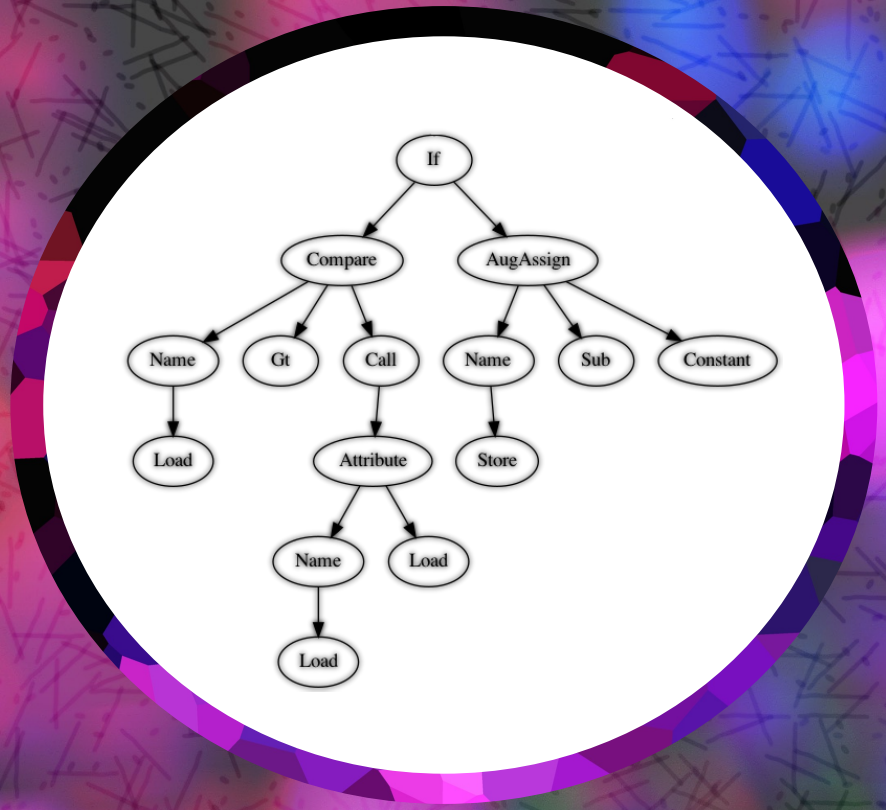
ソースコードの構文木表現による
構造類似性を用いた自動関数生成方式

北 椋太　岡田 龍太郎　峰松 彩子　中西 崇文
武蔵野大学データサイエンス学部
TransMedia Tech Lab



ソースコードの構文木表現による構造類似性を用いた自動関数生成方式

北 椋太　岡田 龍太郎　峰松 彩子　中西 崇文
武蔵野大学データサイエンス学部
TransMedia Tech Lab

1. 研究背景
2. 研究目的
3. 提案方式
4. 実験

研究背景

ソフトウェアの大規模化・利用分野の拡大

システムの不具合や故障が社会的な問題になることが増加

ソフトウェア開発ライフサイクル

保守のコストが占める割合が高い

保守作業の効率化が重要な課題となっている

研究背景

保守作業

プログラムの理解・修正に多くの時間を要する

コードクローン　…　ソースコード中に類似または一致したコード片のこと

ソースコードに不具合が発見された場合、

そのコードクローンすべてについて検査・修正が必要

研究目的

ソースコードの構文木表現による構造類似性を用いた自動関数生成方式

ソースコード中に含まれるコードクローンを検出し、

関数化することによって、一元的な管理を可能とする

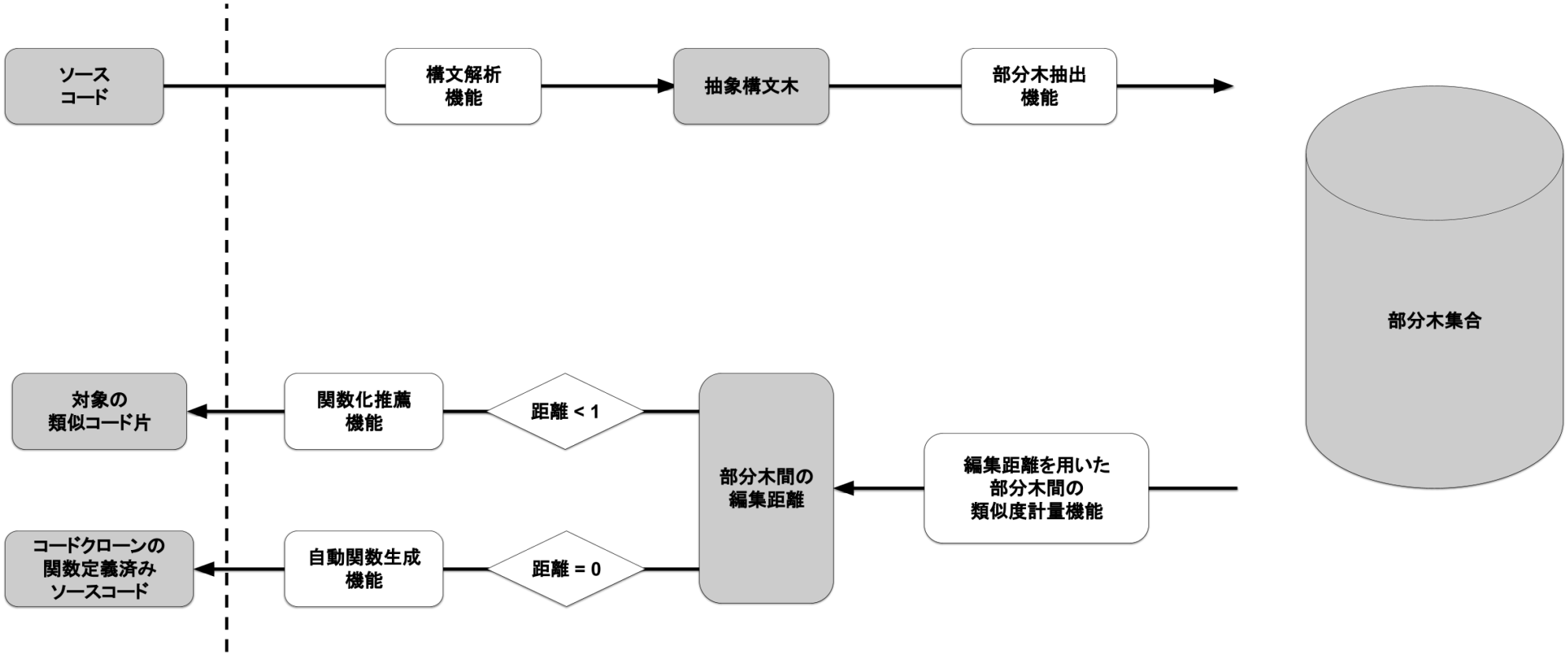
```
taro_h, taro_w = 170, 60
taro_std = round(22*(taro_h/100)**2, 2)
taro_bmi = round(taro_w/((taro_h/100)**2), 2)

jiro_h, jiro_w = 165, 55
jiro_std = round(22*(taro_h/100)**2, 2)
jiro_bmi = round(taro_w/((taro_h/100)**2), 2)
```

```
def cal_std_bmi(h, w):
    std_w = round(22*(h/100)**2, 2)
    bmi = round(w/((h/100)**2), 2)

taro_std, taro_bmi = cal_std_bmi(170, 60)
jiro_std, jiro_bmi = cal_std_bmi(165, 55)
```

提案方式　– システム構成図 -



実験1　関数化推薦方式の有効性検証

一致した部分木
(部分木の根のID1,部分木の根のID2)

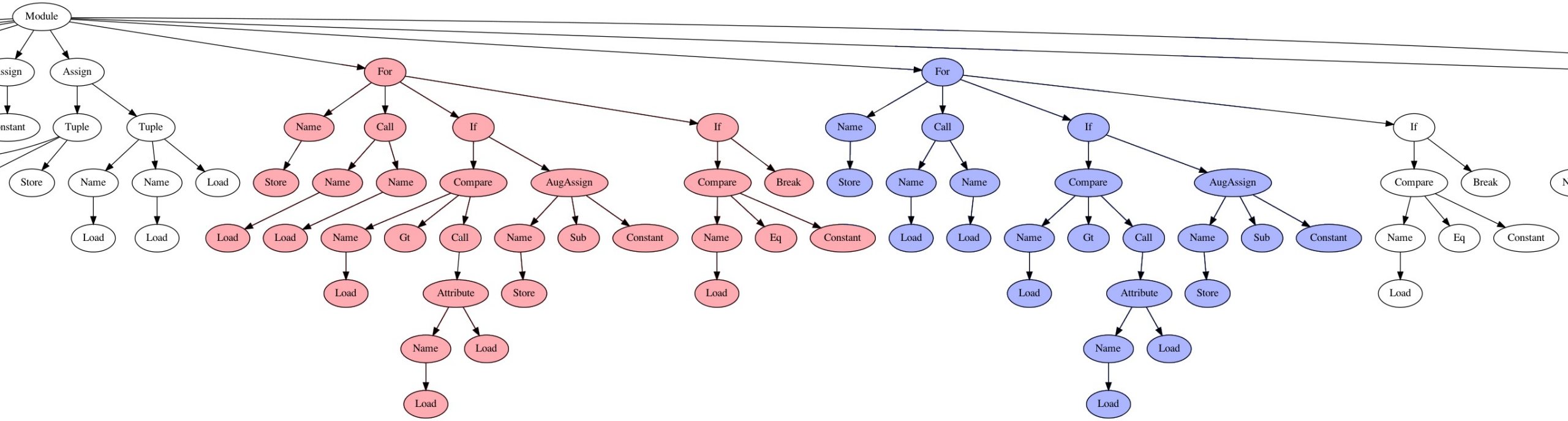
(19, 133),
(45, 75),
(55, 85),
(109, 146)
(41, 71),
(40, 70),
(32, 62)

類似した部分木
(部分木の根のID1,部分木の根のID2)：編集距離

(19, 45)：10.0,
(19, 55): 12.0,
(19, 75): 10.0,
(19, 85): 12.0,
(19, 109): 9.0,
(19, 126): 9.0,
(19, 146): 9.0,
(45, 55): 5.0,
(45, 85): 5.0,
(45, 109): 5.0,
(45, 126): 5.0,
…

実装途中

実験2　関数生成に用いる最適な深さの考察



実験3　自動関数生成方式の有効性検証

k=5の場合

入力

```
(前略)

for _ in range(cannon1):
    if rate1 > random.random():
        cannon2_ -= 1
    if cannon2_ == 0:
        break
for _ in range(cannon2):
    if rate2 > random.random():
        cannon1_ -= 1
    if cannon1_ == 0:
        break

cannon1, cannon2 = cannon1_, cannon2_
print(cannon1, cannon2)
```

出力

(関数部)

```
def function1(var1, var2):
    if var1 > random.random():
        var2 -= 1
    return var1, var2
```

(実行部)

```
(前略)

for _ in range(cannon1):
    rate1, cannon2_ = function1(rate1, cannon2_)
    if cannon2_ == 0:
        break
for _ in range(cannon2):
    rate2, cannon1_ = function1(rate2, cannon1_)
    if cannon1_ == 0:
        break

cannon1, cannon2 = cannon1_, cannon2_
print(cannon1, cannon2)
```

実験3　自動関数生成方式の有効性検証

k=6の場合

入力

(前略)

```
for _ in range(cannon1):
    if rate1 > random.random():
        cannon2_ -= 1
    if cannon2_ == 0:
        break
for _ in range(cannon2):
    if rate2 > random.random():
        cannon1_ -= 1
    if cannon1_ == 0:
        break

cannon1, cannon2 = cannon1_, cannon2_
 print(cannon1, cannon2)
```

出力

(関数部)

```
def function1(var1, var2, var3):
    for _ in range(var1):
        if var2 > random.random():
            var3 -= 1
        if var3 == 0:
            break
    return var1, var2, var3
```

(実行部)

(前略)

```
cannon1, rate1, cannon2_ =
          function1(cannon1, rate1, cannon2_)

cannon2, rate2, cannon1_ =
          function1(cannon2, rate2, cannon1_)

cannon1, cannon2 = cannon1_, cannon2_
print(cannon1, cannon2)
```