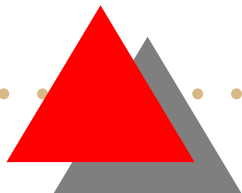


データマイニングを利用した プログラムの改善

田中 哲 <akr@m17n.org>

産業技術総合研究所 情報処理研究部門

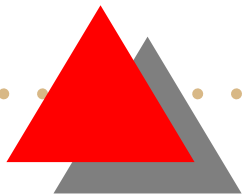




目的

プログラムを改善するネタを発見する

- どこをリファクタリングする？
- ライブラリに追加して嬉しいメソッドはどんなもの？
- どんなアスペクトをくくり出せる？
- どんな言語機構を追加すると嬉しい？




瀕出構造を抽象化

```
while (*p++ = *q++) ;
```

↓

```
strcpy(q, p);
```

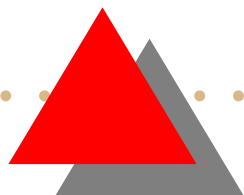
こういう抽象化候補を自動的に抽出したい



データマイニング

大量のデータから半自動的に規則性を見つけ出す方法
いろいろなアルゴリズムがある

- 半構造データ (XML) を対象とするアルゴリズム: FREQT
- リレーショナルデータベースを対象とするアルゴリズムなど



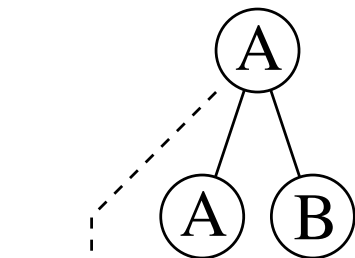
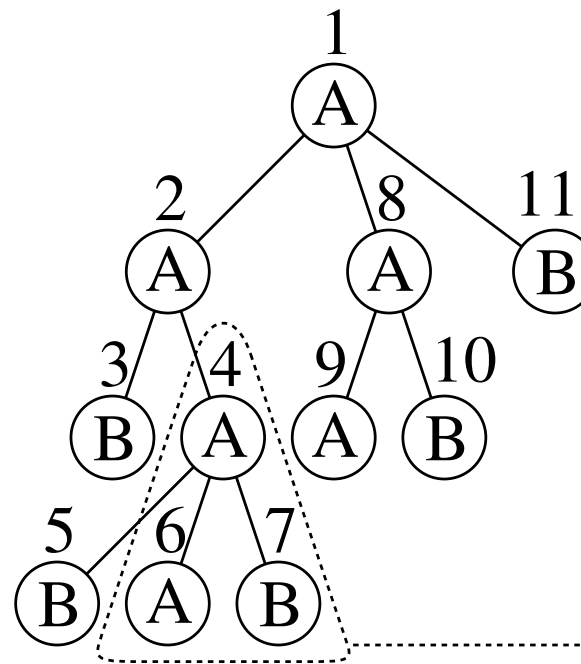
FREQT

半構造データ (XML) を対象とするデータ
マイニングアルゴリズム [浅井 et al, 2002]

Data Tree (size:11)

Pattern (size:3)

ラベルつき順序
木の中から瀕出
パターンを発見
する

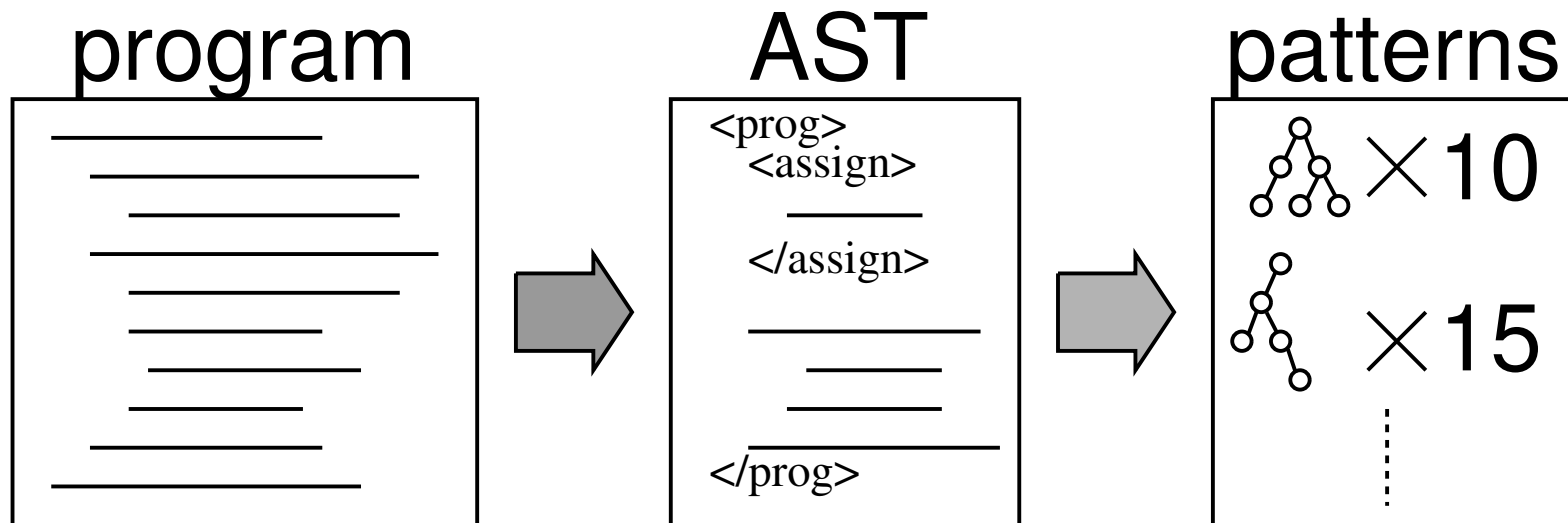



1	2	11
1	8	11
4	6	7
8	9	10

Root Occurrences
{1, 4, 8}

プログラムからパターンを抽出

1. プログラムを構文解析して構文木に変換
2. 構文木に FREQT を適用





データマイニング実験

- Apache Ant
- Apache Tomcat
- Xerces, JDOM, XOM のサンプル
- Ruby 標準添付ライブラリ

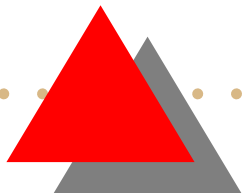


Apache Ant

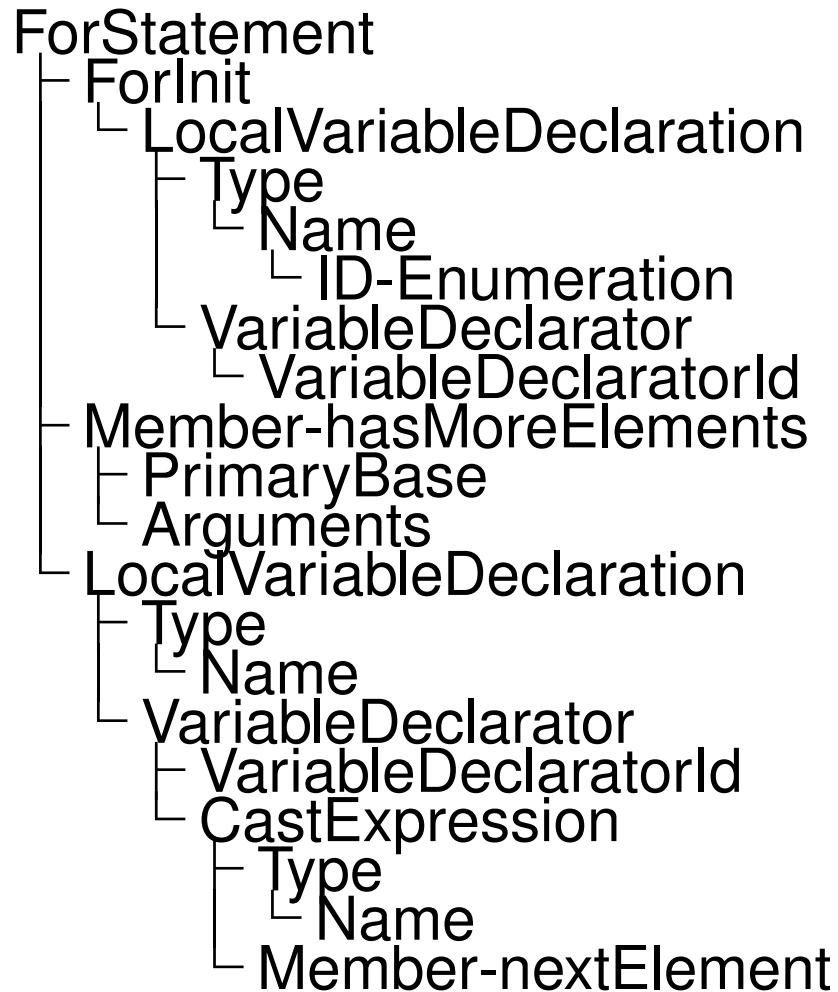
ビルドツール

- Java
- 8MBytes
- 93 万行
- 495621 ノード

37889 パターン発見



Ant から見つかった例 (構文木)



× 77

Ant から見つかった例 (Java)

```
for (Enumeration VAR = ...;
     VAR.hasMoreElements(); ) {
    TYPE VAR =
        (TYPE) VAR.nextElement();
    ...
}
```

× 77

Enumeration による繰り返し

J2SDK 1.5 (Tiger) での略記法

発見されたパターン:

```
for (Enumeration VAR = ...;
     VAR.hasMoreElements(); ) {
    TYPE VAR =
        (TYPE) VAR.nextElement();
    ...
}
```



J2SDK 1.5 (Tiger) による記法:

```
for (TYPE VAR : ...) {
    ...
}
```



言語の改善候補の発見

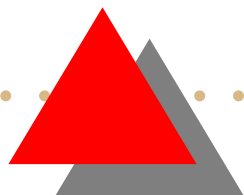
```
for (Enumeration VAR = ...;
     VAR.hasMoreElements(); ) {
    TYPE VAR =
        (TYPE) VAR.nextElement();
    ...
}
```

これは Java 言語の改善候補



パターンが多すぎる

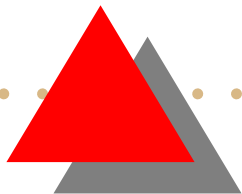
37889 パターンのなかで興味深いのはどれか？





パターンの絞り込み

- 興味のある要素を含んでいるものだけ表示
- 抽象化するとコードがたくさん減りそうなものから表示
- etc.

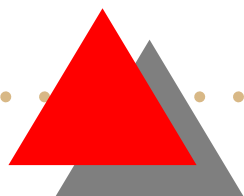




興味のある要素

目的によって異なる

- ライブラリの改善: API の呼出周辺だけ
- プログラムの改善: メソッド名などの識別子周辺だけ
- 言語の改善: なんでも

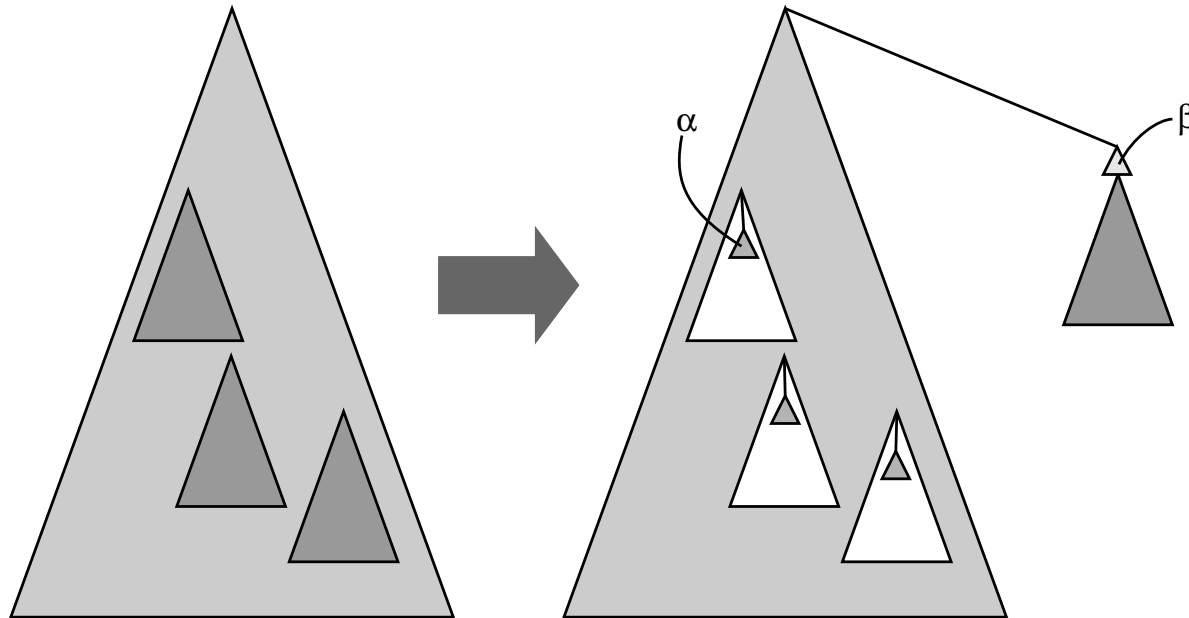


興味のある要素による絞り込み

Ant から発見されたパターン

全てのパターン	37889
識別子を含む	32558
nextElement	265
Enumeration	164
hasMoreElements	130

コードの減る量

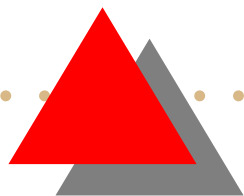


$$\begin{aligned} & \text{パターンサイズ} \times (\text{パターンの出現数} - 1) \\ & - \alpha \times \text{パターンの出現数} - \beta \end{aligned}$$



何に使う？

- 抽象化の候補となる瀬出構造の発見
 - どこをリファクタリングする？
 - ライブラリに追加して嬉しいメソッドはどんなもの？
 - どんなアスペクトをくくり出せる？
 - どんな言語機構を追加すると嬉しい？
- ライブラリの評価





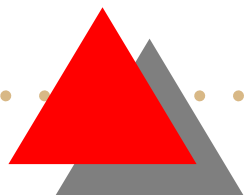
ライブラリの評価

Xerces, JDOM, XOM の比較

サンプルプログラムをデータマイニング

	全パターン数	識別子を含む最初
Xerces	13017	78 番目
JDOM	10625	930 番目
XOM	9945	347 番目

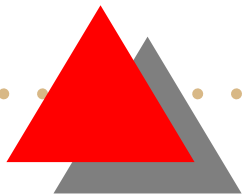
JDOM, XOM, Xerces の順に定型記述が少ない?





見つかったパターン

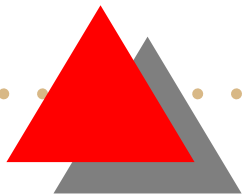
- 同じような import の並び
- Enumeration を使用した for ループ
- 整数のインクリメントによる for ループ
- servlet のメソッド定義
- Jakarta Commons の Logging での各クラスの定型的なコード
- Ruby での条件節における kind_of? の使用
- 区切りを挿入する繰り返し





見つからないパターン

- swap など名前的一致が必要なもの
- デザインパターン



swap

名前の違いは無視しているので、swap みたいなものは見つからない

t = a

a = b

b = t

tmp = i

i = j

j = tmp

変数 = 変数

変数 = 変数

変数 = 変数

変数 = 変数

変数 = 変数

変数 = 変数

デザインパターン

クラスの集合には順序が無い

```
class A {}  
class B extends A {}  
class C extends A {}
```

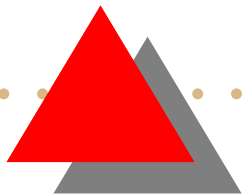
```
class B extends A {}  
class A {}  
class C extends A {}
```

順序木にエンコードするのは不自然



まとめ

- 瀕出構造を発見できた
- 瀕出構造を抽象化できた
- ライブラリの評価を試みた
- 名前的一致は扱えない
- 複数のクラスからなるパターンは見つからなかった





研究課題

- 木でなくグラフを対象にして名前の一致を扱う
- 順序がないノードを扱う
- データフローグラフに対するマイニング



ライブラリの改善

検出したパターンに対応する API を追加

`obj.m1()` + `obj.m2()` が頻出



`obj.m3()` と書けるように `m3` を定義

頻繁に行うことを簡単に書けるようになる

