複合圧縮度によるソースコード流用の検出

奈良先端科学技術大学院大学　情報科学研究科

田中智也†

門田暁人†　松本健一†

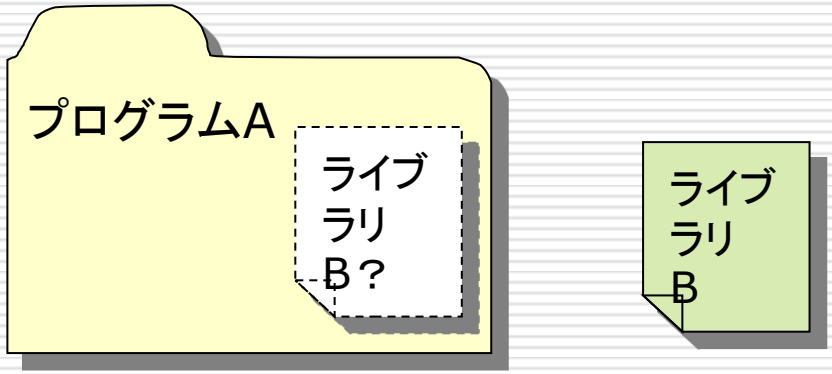
背景

- 近年，オープンソースソフトウェア(OSS)を流用したソフトウェア開発が増えている．
  - 開発コストの削減，高信頼性の確保
- 開発の外注などによりOSSのソースコードが混入し，ライセンス違反を犯してしまう事例がある．
  - SCEIが開発したPSゲーム「ICO」のライブラリ[1]
  - Microsoftが外注したWindows7へのUpgrade支援ツール[2]
- 法的リスクを負う可能性を下げるための対策が必要であると考えられる．
  - ソフトウェアを出荷する前に，ソースコード流用の有無を検出する．

[1]：“「USB版Windows 7」作成ツールにGPLコード　Microsoftが謝罪”, http://www.itmedia.co.jp/news/articles/0911/16/news026.html
[2]：“PlayStation 2 Game ICO Violates the GPL”, http://news.slashdot.org/article.pl?sid=07/11/28/0328215

プログラム圧縮による流用の検出

- プログラムAがライブラリBを流用しているか否かを調べたい.



- Hemelらの提案[3]
  - プログラムAがライブラリBを流用しているならば，Aの圧縮サイズと，A+Bの圧縮サイズはほぼ等しくなるはず.

[3]Armijn Hemel, Kark Trygve Kalleberg, Rob Vermaas, Eelco Dolstra, “Finding Software License Violations Through Binary Code Clone Detection," MSR 2011, May. 2011.

目的とアプローチ

- 目的
  - プログラム圧縮による流用検出の実用性を明らかにする.
- アプローチ
  - プログラム圧縮の度合いの尺度(複合圧縮度)を提案する.
  - 複合圧縮度による流用検出の精度を実験的に明らかにする.

複合圧縮度の提案

- 複合圧縮度：2つのプログラムを結合した場合の圧縮度合い
- 2つの尺度
  - Consize：複合圧縮度のサイズによる表現
  - ConRate：複合圧縮度の割合による表現

評価尺度の提案:複合圧縮度のサイズによる表現ConSize

$$ConSize = Comp(A) + Comp(B) - Comp(A|B)$$

ただしA, Bはプログラム
A|BはA, Bを連結したプログラム
Comp(X)はXの圧縮後のサイズ

- 流用がある場合,
  $(Comp(A) + Comp(B)) >> Comp(A|B)$ となるため, Consizeの増加が期待される.

評価尺度の提案:複合圧縮度の割合による表現ConRate

$$ConRate = \frac{Comp(A) + Comp(B) - Comp(A|B)}{Comp(A) + Comp(B)}$$

- ConSizeに比べて，個々のファイルサイズで除算しているため，ファイルサイズによる影響を受けにくいことが期待される.

評価実験：実験環境設定

- 実験目的
  - 各評価尺度の，流用の判別精度の評価を行う．
- 実験環境
  - 実験対象のソフトウェア
    - 105組の，オープンソースソフトウェアのペア
    - 開発言語：C/C++
  - クローン検出ツール（流用の正解集合を求めるため）
    - CCFinderX[4]
  - 圧縮ツール
    - Lzip
    - 辞書式圧縮法であるLZMAアルゴリズムを用いている．
      - 重複が多いほど高い圧縮率が得られる．

[4]：“CCFinderホームページ”, http://www.ccfinder.net/ccfinderx-j.html

評価実験：実験手順

- 以下の手順により，評価実験を行った．
  - 各OSSペアに対し，人手による分析を行い，正解集合を作成する．
    - 105組中，29組が流用あり，76組が流用なしであった．
  - 各OSSペアについて，個々に圧縮した場合のサイズと結合して圧縮した場合のサイズを記録する．
  - 記録したサイズに基づき，ConSize，ConRateを算出する．
  - 各評価尺度について，正解集合を用いてPrecision，Recallを算出する．

評価実験：手順1…正解集合の作成

- 以下の手順により，プログラムA,B間の流用の有無を調査した.
  - CCFinderXを用いて，A,B間のコードクローンを全て検出する.
  - 最長コードクローンに対し，目視で流用であるかどうかを確認する.
  - 流用でないと判断した場合，次に長いクローンに対し同様の作業を行う.
  - クローン長が30トークン以下になった場合，流用なしと判断する.

流用の例

```
578  /* Creating working area.  */
579  single_locale = (char *) alloca (strlen (categoryvalue) + 1);
580  ADD_BLOCK (block_list, single_locale);
581
582
583  /* Search for the given string.  This is a loop because we perhaps
584     got an ordered list of languages to consider for the translation.  */
585  while (1)
586    {
587      /* Make CATEGORYVALUE point to the next element of the list.  */
588      while (categoryvalue[0] != '¥0' && categoryvalue[0] == ':')
589    ++categoryvalue;
590      if (categoryvalue[0] == '¥0')
591    {
592      /* The whole contents of CATEGORYVALUE has been searched but
593         no valid entry has been found.  We solve this situation
594         by implicitly appending a "C" entry, i.e. no translation
595         will take place.  */
596      single_locale[0] = 'C';
597      single_locale[1] = '¥0';
598    }
599      else
600    {
601      char *cp = single_locale;
602      while (categoryvalue[0] != '¥0' && categoryvalue[0] != ':')
603        *cp++ = *categoryvalue++;
604      *cp = '¥0';
```

```
315  /* Creating working area.  */
316  single_locale = (char *) alloca (strlen (categoryvalue) + 1);
317  ADD_BLOCK (block_list, single_locale);
318
319
320  /* Search for the given string.  This is a loop because we perhaps
321     got an ordered list of languages to consider for th translation.  */
322  while (1)
323    {
324      /* Make CATEGORYVALUE point to the next element of the list.  */
325      while (categoryvalue[0] != '¥0' && categoryvalue[0] == ':')
326    ++categoryvalue;
327      if (categoryvalue[0] == '¥0')
328    {
329      /* The whole contents of CATEGORYVALUE has been searched but
330         no valid entry has been found.  We solve this situation
331         by implicitly appending a "C" entry, i.e. no translation
332         will take place.  */
333      single_locale[0] = 'C';
334      single_locale[1] = '¥0';
335    }
336      else
337    {
338      char *cp = single_locale;
339      while (categoryvalue[0] != '¥0' && categoryvalue[0] != ':')
340        *cp++ = *categoryvalue++;
341      *cp = '¥0';
342    }
```

コメントが一致しているため、流用と判断した。

クローンではあるが流用ではない例

```
415 static void usage(const char * str1, const char * str2)
416 {
417     if (str1)
418     debug(DEBUG_APPERROR, "¥nbad usage: %s %s¥n", str1, str2);
419
420     debug(DEBUG_APPERROR, "Usage: cvsps [-x] [-u] [-z <fuzz>] [-s
421     debug(DEBUG_APPERROR, "           [-f <file>] [-d <date1> [-d <
422     debug(DEBUG_APPERROR, "           [-v] [-h]");
423     debug(DEBUG_APPERROR, "");
424     debug(DEBUG_APPERROR, "Where:");
425     debug(DEBUG_APPERROR, "  -x ignore (and rebuild) CVS/cvsps.ca
426     debug(DEBUG_APPERROR, "  -u update CVS/cvsps.cache file");
427     debug(DEBUG_APPERROR, "  -z <fuzz> set the timestamp fuzz fa
428     debug(DEBUG_APPERROR, "  -s <patchset> generate a diff for a g
429     debug(DEBUG_APPERROR, "  -a <author> restrict output to patch
430     debug(DEBUG_APPERROR, "  -f <file> restrict output to patchsets
431     debug(DEBUG_APPERROR, "  -d <date1> -d <date2> if just one d
432     debug(DEBUG_APPERROR, "     revisions newer than date1.  If two
433     debug(DEBUG_APPERROR, "     show revisions between two dates
434     debug(DEBUG_APPERROR, "  -b <branch> restrict output to patch
435     debug(DEBUG_APPERROR, "  -v show verbose parsing messages")
436     debug(DEBUG_APPERROR, "  -t show some brief memory usage st
437     debug(DEBUG_APPERROR, "  --norc when invoking cvs, ignore the
438     debug(DEBUG_APPERROR, "  -h display this informative message"
439     debug(DEBUG_APPERROR, "¥ncvsps version %s¥n", VERSION);
440
441     exit(1);
```

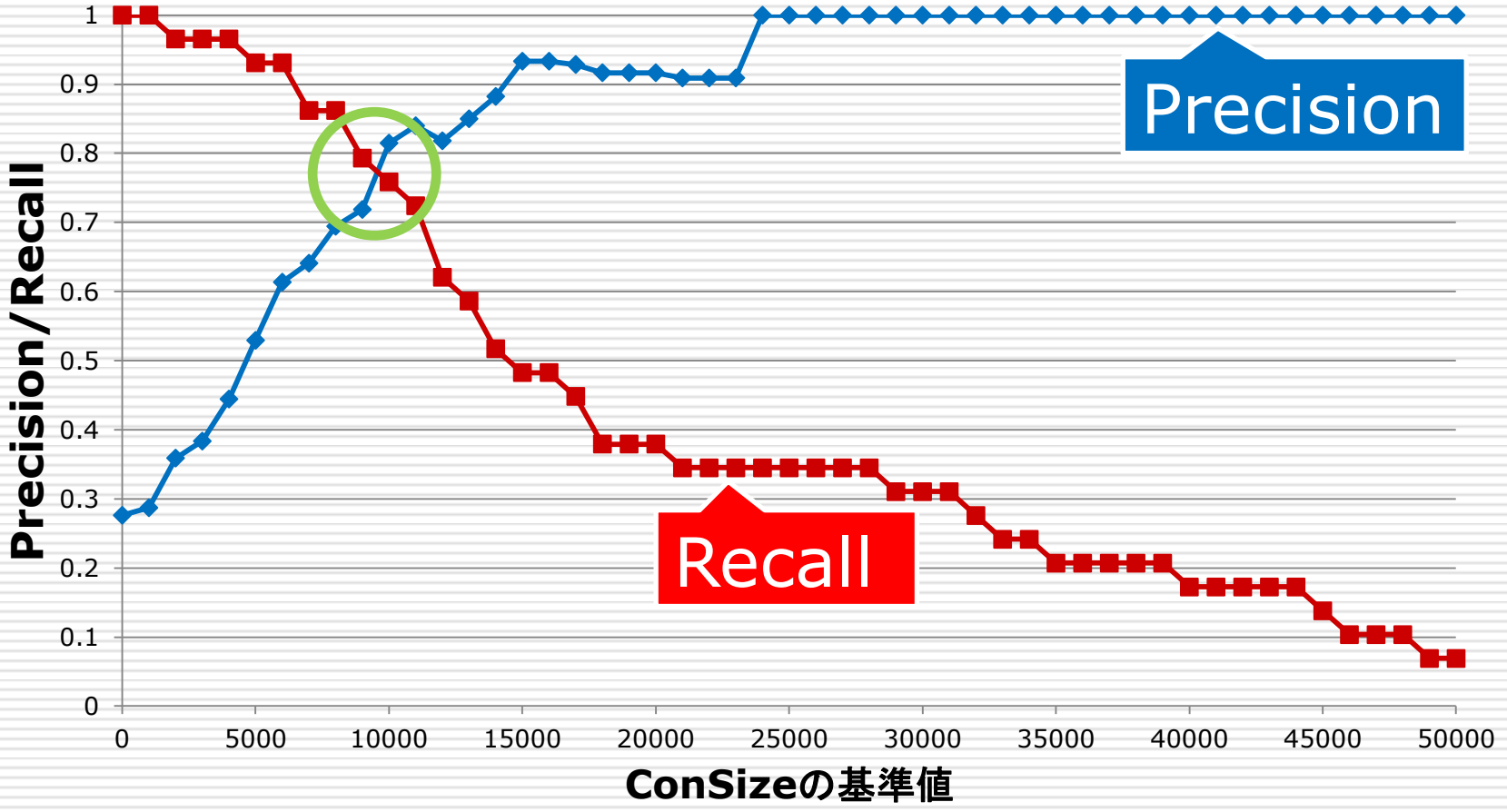
```
55 void print_usage(void)
56 {
57
58   fprintf(stderr, "Usage: nasdconv [options] inputfile [outputfile]¥n");
59   fprintf(stderr, " the following options are recognized:¥n");
60   fprintf(stderr, " -f arg specify geomagnetic field model, arg is one of
61   fprintf(stderr, "          igrf - Intl Geomagnetic Reference Field¥n");
62   fprintf(stderr, "          wmm - Dept of Defense World Magnetic Mode
63   fprintf(stderr, "          (default value is wmm)¥n");
64   fprintf(stderr, " -h     print this help message and then exit¥n");
65   fprintf(stderr, " -k     prefix 3 char alpha airport ids with ¥"K¥"¥n");
66   fprintf(stderr, " -m arg specify magnetic variation, arg is one of:¥n"
67   fprintf(stderr, "          db - use the database entry when available¥n
68   fprintf(stderr, "          fm - always use geomagnetic field model¥n");
69   fprintf(stderr, "          (default value is fm)¥n");
70   fprintf(stderr, " -t arg specify output format type, arg is one of:¥n")
71   fprintf(stderr, "          fplan - generate fplan (nav) database format¥
72   fprintf(stderr, "          generic - generate a simple report-like forma
73   fprintf(stderr, "          icao - generate ICAO Map world file format¥n
74   fprintf(stderr, "          nav - generate nav (fplan) database format¥r
75   fprintf(stderr, "          (default value is generic)¥n");
76
77 }
```

構文が一致しているためクローンとして検出されるが、流用とはいえない。

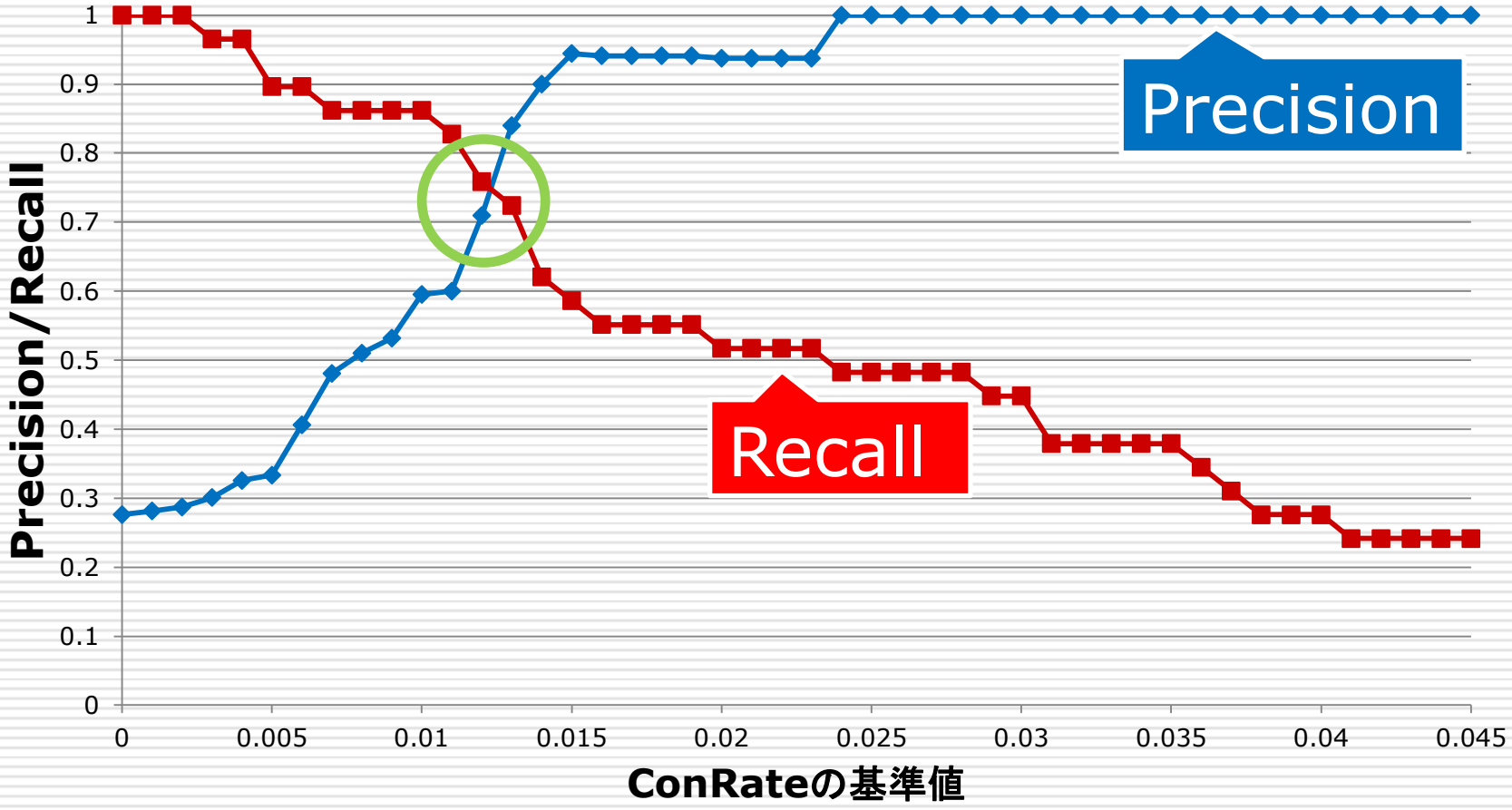
評価実験：
評価精度確認に用いる指標

- Precision，Recallを用いて評価精度を確認する．
  - Precision
    - 流用ありと判断したもののうち，実際に流用があったものの割合
  - Recall
    - 実際に流用があったもののうち，流用ありと判定できたものの割合
- ConSize，ConRateの各評価尺度に対して，これ以上の値であれば流用ありとみなすという基準値を設定し，その値を変動させることでPrecision，Recallがどのような値を取るかを確認する．

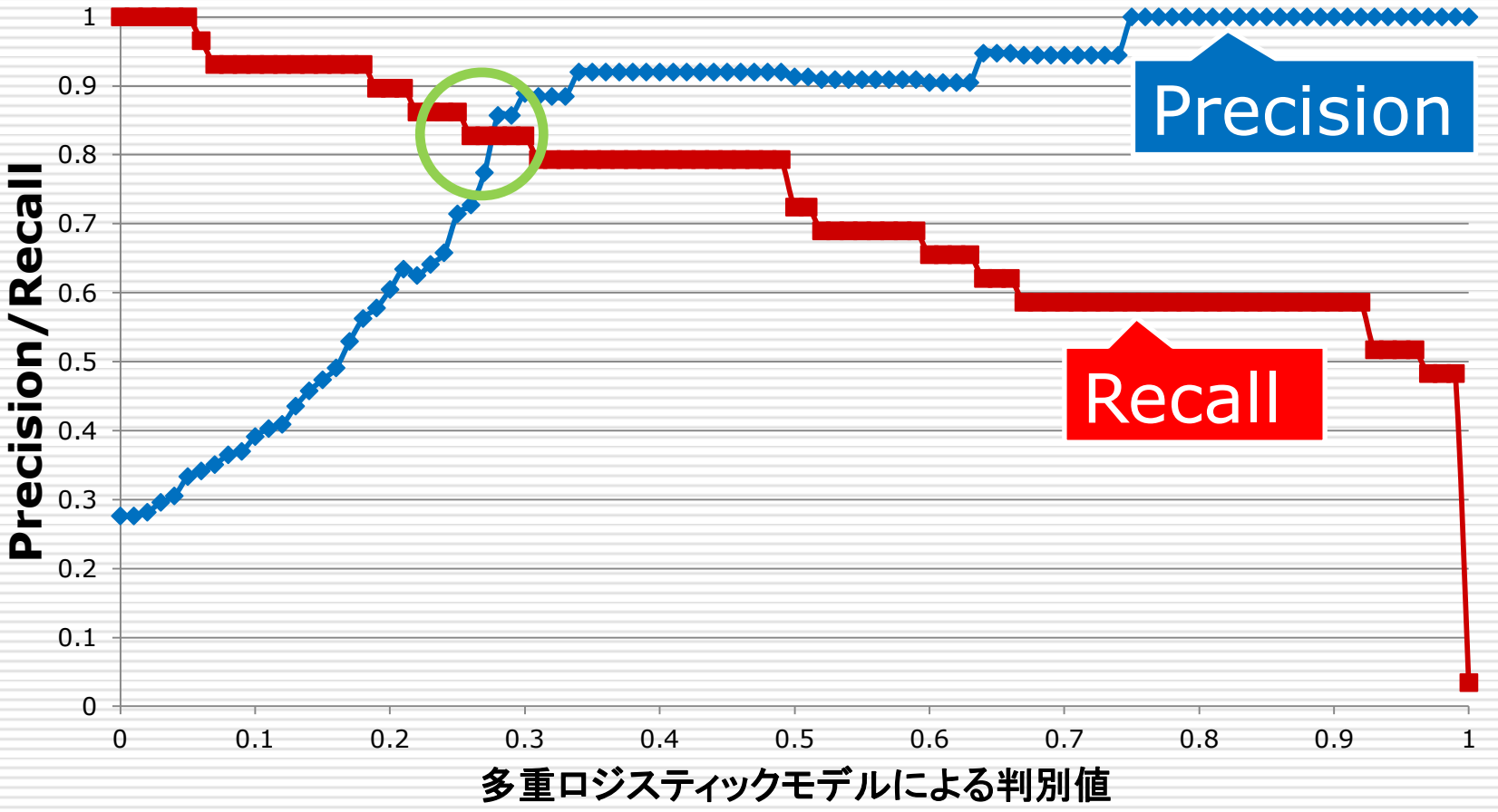
実験結果：流用あり(ConSize)



実験結果：流用あり（ConRate）



実験結果：
流用あり（多重ロジスティックモデル）



参考実験：バイナリファイルを対象とした複合圧縮度の比較実験

- バイナリファイルを用いた圧縮による流用検出実験
  - 5個のMP3タグエディタを用いた比較
    - STEPはSuperTagEditorを改変したもの
    - Tuyutag, Teatime, MP3tagはSuperTagEditorとは全く独立に開発されたもの

参考実験:実験結果

Project_A	Project_B	A_FileSize (Byte)	B_FileSize (Byte)	A_Comp Size(Byte)	B_Comp SIze(Byte)	A_Comp(%)	B_Comp(%)	(A+B) Comp(%)	AB_Comp Size(Byte)	AB Comp(%)	ConSize	ConRate
SuperTag Editor	STEP	724,992	819,200	285,983	302,991	0.394464	0.3698621	0.381412	468,959	0.303692	120,015	0.2038
SuperTag Editor	Tuyutag	724,992	957,952	285,983	363,109	0.394464	0.3790472	0.385688	645,765	0.383712	3,327	0.0051
SuperTag Editor	Teatime	724,992	1,006,080	285,983	353,460	0.394464	0.351324	0.369391	635,272	0.366982	4,171	0.0065
SuperTag Editor	MP3tag	724,992	4,731,112	285,983	1,650,581	0.394464	0.348878	0.354935	1,913,838	0.35077	22,726	0.0117
STEP	Tuyutag	819,200	957,952	302,991	363,109	0.369862	0.3790472	0.374813	662,995	0.373066	3,105	0.0047
STEP	Teatime	819,200	1,006,080	302,991	353,460	0.369862	0.351324	0.359644	652,360	0.357403	4,091	0.0062
STEP	MP3tag	819,200	4,731,112	302,991	1,650,581	0.369862	0.348878	0.351975	1,936,040	0.348816	17,532	0.0090
Tuyutag	Teatime	957,952	1,006,080	363,109	353,460	0.379047	0.351324	0.364846	619,515	0.31543	97,054	0.1354
Tuyutag	MP3tag	957,952	4,731,112	363,109	1,650,581	0.379047	0.348878	0.353958	2,010,288	0.35336	3,402	0.0017
Teatime	MP3tag	1,006,080	4,731,112	353,460	1,650,581	0.351324	0.348878	0.349307	1,999,835	0.348574	4,206	0.0021

まとめ

- まとめ
  - 圧縮によるプログラム流用流用検出の実用性について検討した.
    - 複合圧縮度の尺度としてConSize, ConRateを提案した.
    - 105組のC/C++プログラムを用いて実験した.
    - ConSize, ConRateを組み合わせて用いることで, Precision = 0.8, Recall = 0.8程度を達成できる.
    - 10組のバイナリプログラムを用いて同様の実験を行った.
      - 流用が明らかであるSuperTagEditorとSTEPの間には, 他の組み合わせと比べて高いConSize, ConRateが観測された.

今後の課題

- 今後の課題
  - 評価尺度の精度向上について検討する.
  - 実験対象プログラム数を増やす.
  - プログラムサイズが及ぼす影響について検討する.

ご清聴ありがとうございました。