情報処理学会論文誌　Vol. 52　No. 8　2402–2411 (Aug. 2011)

階層的ライセンス知識を用いたライセンス特定ツールの開発

真鍋 雄貴[†1]　ダニエル モラレス ゲルマン[†2]
井上 克郎[†1]

開発者がオープンソースソフトウェア部品を利用する場合，それらのソフトウェアライセンス（ライセンス）を理解し，遵守する必要がある．しかし，開発者がライセンスを正確に読解することは容易ではない．そこで，部品のライセンスが既知のどのライセンスと合致しているか調べること（特定）で，ライセンス理解にかかるコストを削減したい．本稿では，オープンソースソフトウェアから抽出した知識に基づいたライセンス特定手法を提案する．使用する知識は，複数の大規模オープンソースソフトウェアから抽出した．評価として，既存のライセンス特定手法と回答率，正答率，所要時間で比較を行い，提案手法が既存の手法より優れていることを示した．

Development of License Identification Tool Using Hierarchical License Knowledge

Yuki Manabe,[†1] Daniel M. German[†2]
and Katsuro Inoue[†1]

Developers using OSS components have to understand and comply with the terms of their licenses. However, it is difficult to correctly understand them. To reduce this effort, we aim to identify the license names from known licenses. In this paper, we propose a license identification approach based on knowledge extracted from open source software. We compare the answer rate, the accuracy and the time required by our approach to existing license identification approaches. The result shows that our approach can identify the license of each source file more correctly than other existing approaches.

1. はじめに

開発コスト削減の一手段として，既存のソフトウェアの一部や全部を，部品として同一システム内や他のシステムで再利用することが広く行われている[1)]．再利用においては，ソフトウェア部品に修正が加えられることも多い．Selby によれば，NASA software development environment が持つ 25 のソフトウェアシステムのためのリポジトリを調査したところ，全体の 14.75%のモジュールが修正が加えられたうえで再利用されていた[2)]．

特に，オープンソースソフトウェア（以下，OSS）に含まれる関数やクラスなどのソフトウェア部品（以下，部品）が利用されることが多い．Li らによると，36%の企業がオープンソースソフトウェアを修正して利用している[3)]．

OSS の部品を利用するためには，多くの場合，ソースファイルのコメントとして記述されている，利用に関する許諾と許諾を受けるために負う義務を示したソフトウェアライセンス（以下，ライセンス）を読み，理解し，それを遵守しなければならない．

一般に，ライセンスは自然言語で記述された複数の文で構成されており，また，前提知識が必要だったり，他のファイルへの参照を含んだりするため，開発者がライセンスを正確に読解することは容易ではない．

一方，多くの OSS 部品のライセンスは，定型の文で構成された既知のライセンスが用いられている場合が多い．この場合，ライセンスの文章全体を詳細に読解せずに，既知のどのライセンスと合致しているかが分かれば（これをライセンスの特定と呼ぶ），どのような許諾条件や義務があるか比較的容易に認識でき，その許諾条件や義務をそのまま適用することで，効率的な部品の利用を促進することができる．

しかし，ライセンスの特定も容易ではない．ソースコード中のライセンスの記述には，綴り間違いや単語の同義語への変更などの表記揺れ，ソースファイルにより変化する著者名や組織名が含まれるため，簡単な照合だけで特定することは困難である．そのため，細かい差異を認識して，ライセンスの特定を効率良く行えるシステムが望まれる．

現在，いくつかのライセンス特定ツールが提案されているが，それらはライセンスの版（バージョン）を答えない，複数回答を報告する，ライセンスを特定する際に使用する知識

†1 大阪大学
Osaka University
†2 ヴィクトリア大学
University of Victoria

の管理が容易でない，などの問題がある．

本稿では，まず上記のライセンス特定の問題を詳しく理解するため，複数の OSS からソースファイルに含まれるコメントを収集し，調査を行い，ライセンス特定における課題を特定した[4]．次に，調査により明らかになったライセンス特定の課題と，OSS の現状に基づき，ライセンス特定ツールに必要な要件を定めた[5]．そして，それらの要件を満たせるよう手法の設計，実装を行った．

本研究では，階層的なライセンス知識を用いた特定手法を提案する．提案手法が従来手法と比較して精度，スケーラビリティ，ライセンス知識の管理の点でどのように優れているかを評価するため，評価実験を行った．結果として，提案手法は従来手法に比べ，実用的な精度，時間で特定することができた．また，大半のソフトウェアでソフトウェアに含まれるすべてのソースファイルに対して，ライセンスを特定することができることを示した．

以降，2 章で本稿で用いる用語を定義する．3 章でライセンス特定の課題について説明し，課題を解決するために必要な要件を述べ，そして，それらの要件と既存手法との対応関係を述べる．4 章では，関連研究として，既存のライセンス特定手法について説明する．5 章では提案するツールの設計について述べ，ツールの実行例を示す．6 章では提案するツールを精度と速度，スケーラビリティ，ライセンス知識の表現法の観点から他の既存のライセンス特定ツールとの比較を行った評価実験について述べる．7 章では，まとめと今後の課題について述べる．

2. 用　　語

ライセンスとは，著作者が定めたソフトウェアの利用者に対する指示の集合である．よく知られているライセンスの名前と本稿で用いるそれらの略記を**表 1** に示す．本稿では，ライセンスに含まれる指示の具体的な内容については議論しない．

ソースファイルには，プログラムコードとコメントが記述される．**コメント**はライセンスに関係のある文の系列（**ライセンス記述**）もしくはその他の文字列から構成されている．ライセンス記述中の各文のことを**ライセンス文**と呼ぶ．**図 1** にライセンス記述の例を，**図 2** に図 1 のライセンス記述に含まれるライセンス文の例を示す．

メタライセンス文とはライセンス文の集合を拡張正規表現で表記したものである．図 2 の文 (2) において，above が省かれた文も同様なライセンス文として取り扱う場合，これらを一般化したメタライセンス文は

```
Redistributions of source code must retain the (above )? copyright notice.
```

表 1　一般的なオープンソースライセンスの名前と本稿での略記
Table 1　Names of common open source licenses and their abbreviations used in this article.

Abbrev.	License Name
Apache	Apache Public License
BSD4	Original BSD, also known as BSD with 4 clauses
BSD3	BSD with 3 clauses (BSD4 minus advertisement clause)
BSD2	BSD with 2 clauses (BSD4 minus advertisement and endorsement clauses)
boostV1	The Boost Software license
CPL	Common Public License
CDDL	Common Development and Distribution License
EPL	Eclipse Public License
GPL	General Public License
GPLnoVersion	GPL with no version indicated
IBM	IBM Public License
LesserGPL	Lesser General Public License (successor of the Library GPL, also known as LGPL)
LibraryGPL	Library General Public License (also known as LGPL)
MIT/X11	Original license of X11 released by the MIT
MITold	License similar to the MIT/X11, but with different wording
MPL	Mozilla Public License
SameAsPerl	File is licensed in the same terms as Perl
SeeFile	File points to another where the its license is
SunSimpleLic	Simple license used by software developed by Sun Microsystems
ZLIBref	The zlib/libpng license
publicDomain	File is in the public domain

All right reserved. Redistributions of source code must retain the above copyright notice. Redistributions in binary form must reproduce the above copyright notice.

図 1　ライセンス記述の例
Fig. 1　An example of a license description.

(1)　All right reserved.
(2)　Redistributions of source code must retain the above copyright notice.
(3)　Redistributions in binary form must reproduce the above copyright notice.

図 2　ライセンス文の例
Fig. 2　Examples of license sentences.

となる，ただし，(xyz)?は xyz を省略可能であることを示す．

ライセンスルールとは，メタライセンス文の系列からライセンスへの対応を定義するためのメタライセンス文の系列とライセンスの組である．メタライセンス文の集合および，ライセンスルールの集合をまとめて**ライセンス知識**という．

ライセンスの特定とは，ソースファイルからそのソースファイルのライセンスを決定する作業をいう．

3. ライセンス特定の課題と要件

3.1 課　　題

ライセンス特定問題の課題を特定するため，我々は OSS に現れるライセンス宣言の特徴を調査した．調査では，各ソースファイルのライセンス記述について，ライセンス記述に含まれる文や，文の並びについて調査した．そして，調査により得られた知見をライセンス宣言の特徴としてまとめた．調査で，Linux，Eclipse JDT，OpenBSD，Mozilla などに含まれるソースファイルからなるおよそ 30,000 ファイルを調査対象とした．調査により，以下の課題を特定した．

課題 1：ライセンス記述の表記揺れ　ライセンス記述は表記揺れを含むことがある．1 つ目の理由は綴り間違いである．これに該当する記述として，本来分割されるべきでない単語が空白により分割されていたという記述がある．

もう 1 つは，あらかじめライセンス記述に著者名や団体名を記述する部分が決められている場合があり，そこに具体的な著者名や団体名が入ることにより表記揺れが発生する．例として，BSD ライセンスでは，条文として *“THIS SOFTWARE IS PROVIDED BY ⟨name⟩ AS IS AND ANY EXPRESS . . . ”* という条文を持つが，この条項のうち，*⟨name⟩* の部分は著作者名を入れることになっている．そのため，表記揺れが生じる．

次に，ライセンスの版の違いにより，表記揺れが発生する場合がある．例として，GPLv2 では *“either version 2 of the License, or (at your option) any later version.”* となる記述が，GPLv3 では，*“either version 3 of the License, or (at your option) any later version.”* となる．

また，著作者により，文法や綴りの変更が行われる場合がある．確認できたこの種の変更として，*“license”* と *“licence”*，*“it would be useful”* と *“it will be useful”*，*“developed by”* と *“written by”*，*“dealings in”* と *“dealings with”*，*“Redistribution”* と *“Redistributions”* の置換があった．また，句読点においてもコンマやセミコロンの追加，削除が行われていた．

課題 2：著作者による既存のライセンスの修正　著作者により，既存のライセンスにある条項を修正して使用する場合がある．また，開発者の目的にあったライセンスを定めるため，既存のライセンスに条項を追加したり，削除したりすることがある．

例として，BSD4 からの派生がある．BSD3 は BSD4 から 1 つの条項（宣伝条項）を取り除いて作成され，Apache v1.1 は BSD3 に 4 つの文を追加することで作成されている．また，Apache v1.1 から OpenSSL ライセンスが派生している．

より修正がとらえにくい例として，MIT/X11 ライセンスでは *“Premission to use, copy, modify, distribute, and sell this software . . . ”* から *“sell”* が欠落した場合が多く存在した．また，BSD の条項 *“Redistributions of source code must retain . . . ”* が *“Redistributions of source code or documentation must retain . . . ”* へ変更された場合があった．

別のテキストファイルにライセンス記述が記述される GPL や MPL などのライセンスでは，許諾者が補遺を用いてライセンスを修正することを認めている．これは，ライセンスの派生を避けるためである．これらの補遺は例外として知られている．

課題 3：ソースファイルとは異なるファイルへの参照がある　ライセンスの指定はコメントにより行われることが多い．しかし，他のファイルにライセンスが記述され，コメントにはそのファイルへの参照が記述されている場合がある．このようなファイルへの参照は *“For licensing information, see the file . . . ”*，*“See . . . for licensing information”*，*“See . . . for license detail”* などと記述されている．また，このようなファイルは *“main”*，*“root”*，*“top”* ディレクトリにある場合が多く，その名前も *“COPYING”*，*“LICENSE”*，*“README”*，*“copyright.txt”*，*“AUTHORS”*，*“COPYRIGHT_and_LICENSE”* などさまざまである．

課題 4：複数のライセンスが含まれる　1 つのファイルに複数のライセンスが含まれる場合がある．これは，ファイルのライセンスを複数のライセンスから選択できる場合，もしくはソースコード中に以前用いられていたライセンスのライセンス記述が残ったまま，新しいライセンス記述が記述された場合がある．これらのライセンスはそれぞれ別のライセンス記述で表現される場合と，1 つのライセンス記述で表現される場合がある．ライセンス特定問題においては，複数のライセンスが含まれている場合でも，すべてのライセンスが特定されていることが望ましいと我々は考えて，できるだけ多くのライセンスを特定するようにした．

3.2 ライセンス特定ツールに求められる要件

3.1 節で述べたライセンス特定の問題と，OSS におけるライセンスの特徴に基づき，今回のライセンス特定ツールに求められる要件を以下のように定めた．

要件 1：多くの表記揺れを考慮してライセンスを特定できる　課題 1 として，ライセンス記述に表記揺れが含まれるという課題をあげた．しかし，実際に，どのようにライセンス記述に表記揺れが含まれるか，類似したライセンス記述があるかどうかについては明らかになっていない．そのため，実際に開発されている OSS を多数調査し，その結果得られたライセンス記述の表記揺れを考慮してライセンスの特定ができる必要がある．また，調査により明らかにできた表記揺れや，類似したライセンス記述を認識してライセンスを特定するため，細かい文字列の差異により生じる対応するライセンスの違いを正しく認識できる機構を備えていることが望ましい．

要件 2：新しいライセンス記述への適合が容易　Open Source Initiative [⋆1] に承認されている OSS の定義を満たすライセンスは 76 種あるが，それ以外のライセンスも多数あり，必要に応じてそれらを特定できるようにしたい．また，課題 4 であげたとおり，1 つのライセンスが複数の記述を持つ場合がある．これらに対応するため，新しいライセンスに容易に適合できるようにするための機構を備えていることが望ましい．この機構が満たすべき要件としては，同一のライセンスに対して，複数のライセンス記述を定義できること，短いルールとしてライセンス記述を定義できること，ツールに手を加えることなく後からライセンス記述を追加できることがあげられる．

要件 3：高速に処理できる　OSS の部品を利用するためには，事前に各部品のライセンスを特定しておくことが便利である，しかし OSS の規模が大きいと，部品の数も増え，すべてのライセンスの特定に要する時間も大きくなる．したがって，個々のライセンスの特定には，スケーラビリティの高い，高速な処理が望まれる．

4. 既存のライセンス特定手法

ライセンス特定ツールの既存研究として，Fossology [6] と，ASLA [7] がある．

Fossology は bSAM アルゴリズムを用いて既知のライセンス記述とコメントを比較し，コメント中で類似したライセンス記述を持つライセンスを特定結果として出力する．しかし，このアルゴリズムの計算量が大きいため，スケーラビリティが低く，要件 3 を満たさない．

ASLA はメタライセンス文のみを用いて特定するライセンスの記述を行う．しかし，各記述の正規表現は複雑で容易に作成できないため，要件 2 を満たさない（詳しくは 6.3 節で述べる）．また，これらの研究論文にはメタライセンス文を作成するうえで，どのような調査に基づいているのか，また，それらの調査結果に基づいてどのようにメタライセンス文を作成したのかという点が示されていないため，要件 1 を満たしているかは不明である．

Fossology と同様にアルゴリズムを用いて既知のライセンス記述とコメントを比較するライセンス特定ツールとして Open Source License Checker [⋆2] がある．本手法はコメントとライセンス記述間で同一となる行を発見し，最長一致列となる部分を探索する．類似度はライセンス記述に対する最長一致列の割合である．

ALSA と同様に正規表現を用いたライセンス特定ツールとして，Ohcount [⋆3] がある．これらのツールではライセンス記述に対応する正規表現はハードコーディングされており，追加する仕組みは用意されていないため，要件 2 を満たすための条件であるツールに手を加えることなく後からライセンス記述を追加できることを満たさない．

また，商用のサービスとして，Palamida [⋆4]，Black Duck Protex [⋆5]，Protecode [⋆6] などがあるがどのような方法で特定を行い，どのような精度で特定できるか不明である．

ライセンスの不整合を扱った研究として，Agrawal らはライセンスの条項を $\langle actor, operation, action, object \rangle$ のタプルで表現し，ソフトウェアトレーサビリティツールの ArchStudio4 上で義務の伝搬，義務の衝突，システム全体で利用可能な権利を計算する手法を提案した [8]．また，German らはライセンスの不整合が生じたときの対処をライセンス統合パターンとして整理した [9]．これらの手法では，ライセンスの特定は扱っていない．

5. ツールの設計

図 3 に提案するツールの構成を示す．本ツールでは，ソースファイルを入力とし，特定できたライセンス名の列を出力する．本ツールは 5 段階の処理と 2 種類のライセンス知識からなる．これらのライセンス知識はツールの外部ファイルとして実装することにより，要件 2 の「ツールに手を加えることなくライセンス知識を追加できる」という条件を満たす．

本ツールでは，ライセンス知識により正確に特定できたライセンス名のみを出力とする．既存ツールである Fossology や OSLC では，ライセンス知識に入力のソースファイルのコメントと類似したライセンス記述により指定されているライセンスも出力として返す．その

⋆1 http://www.opensource.org/

⋆2 http://forge.ow2.org/projects/oslcv3/
⋆3 http://sourceforge.net/projects/ohcount/
⋆4 http://www.palamida.com/
⋆5 http://www.blackducksoftware.com/protex
⋆6 http://www.protecode.com/

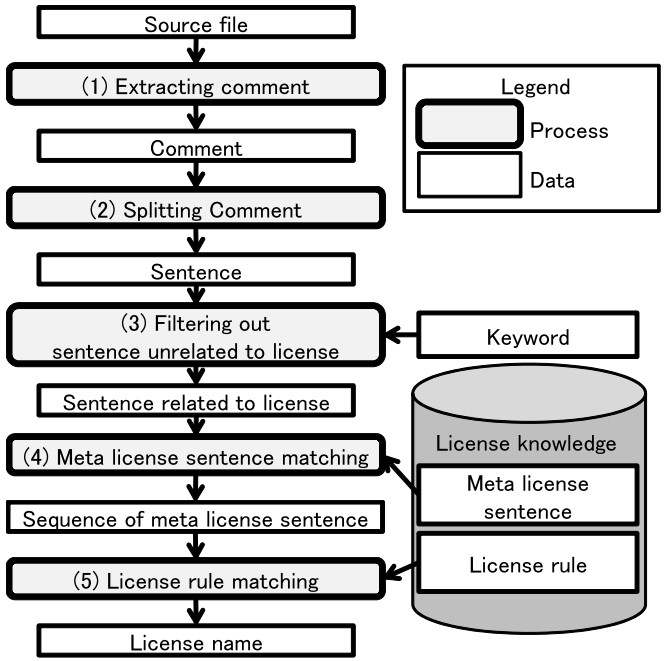

図 3　ライセンス特定ツールの構成
Fig. 3　Architecture of license identification tool.

ため，ライセンス知識が不足し，正確に特定できない場合でも，ライセンスを特定できる場合がある．一方で，出力されるライセンス名が多くなるため，実際には入力のソースファイルに関係のないライセンスが出力される可能性がある．これは，ライセンスを特定したソースファイルを利用する観点から考えたとき，誤ったライセンスに従って利用する可能性を高めることになる．そのため，提案手法では，正確に特定できたライセンスのみ出力することにした．

5.1　ライセンス知識

調査ではコメントを文に分割し，どのような文が存在するかを調べた．その結果，多数の表記揺れを検出した．これらを吸収するため，調査により得られた知識をライセンス知識（License Knowledge）として整理した．ライセンス知識は，427 のメタライセンス文（Meta license sentence），112 個のライセンスに対応する 126 個のライセンスルール（License rule）からなる．

提案手法では，**メタライセンス文**を “(メタライセンス文名)：(メタライセンス文に対応するライセンス文にマッチする拡張正規表現のパターン)” と記述する．本ツールでは，要件 2 を満たす条件の 1 つである「同一のライセンスに対して，複数のライセンス記述を定義できること」を満たすため，メタライセンス文を記述する際，同一のメタライセンス文名を持つ，複数のメタライセンス文を記述することを許す．例として，提案手法では，メタライセンス文 BSDcondSource に対するメタライセンス文を以下のように記述する．以下の記述において，*“s?”* は *“s”* が省略可能であることを意味し，また，*“(above)?”* は *“above ”* が省略可能であることを意味する．

```
BSDcondSource:Redistributions? of source code must retain the (above )?
copy right notice, this list of conditions(,)? and the following disclaimer
(, without modification)?
```

また，**ライセンスルール**を “(ライセンス名)：(メタライセンス文名の系列)” と記述する．ライセンスルールはメタライセンス文の系列とライセンス名（License name）からなる．例として，BSD2 に対応するライセンスルールを以下に示す．

```
BSD2:BSDpre,BSDcondSource,BSDcondBinary,BSDasIs,BSDWarr
```

このルールでは BSD2 のライセンス記述はメタライセンス文の例として示した BSDcondSource のほか，BSDpre，BSDcondBinary，BSDasIs，BSDWarr から構成されており，それらは BSDpre，BSDcondSource，BSDcondBinary，BSDasIs，BSDWarr の順に並ぶことを定義している．

以上のように，正規表現に対する記号名を導入することによって，あるライセンス記述に対応するライセンス知識をその記述に含まれる文（メタライセンス文）と文の並び（ライセンスルール）として表現することができる．このようにライセンス知識を構成することにより，要件 2 を満たす条件の 1 つである「ライセンス記述を短いルールとして記述できる」を満たすことができる．実際に，1 つの正規表現として定義した場合，上記の例において BSDpre，BSDcondSource，BSDcondBinary，BSDasIs，BSDWarr，BSDPre の各ライセンス文を 1 つの正規表現にまとめなければならず，長大な正規表現としてライセンス記述が表記される．一方，提案手法では，ライセンス記述を複数のルールにより定義できている．

また，ライセンスルールを分離したことにより，ライセンス文を複数のライセンスルールで共有でき，ライセンス知識全体を小さくすることができる．例として，ライセンスルール BSD4 では BSDpre，BSDcondSource，BSDcondBinary，BSDcondAdvPart1，BSDcondAdvPart2，BSDcondEndorseRULE，BSDasIs，BSDWarr となっており，8 つのライセンス文を使用する．一方，ライセンスルール BSD3 では，BSDpre，BSDcondSource，BSDcondBinary，BSDcondEndorseRULE，BSDasIs，BSDWarr と 6 つのライセンス文で構成されており，そのすべてが BSD4 のライセンス文になっている（順序は異なる）．このような場合，ライセンス文の共有によりライセンス知識全体をコンパクトに記述できている．

このようにライセンス知識をコンパクトに記述できるため，より多くのライセンス知識を保持することが可能である．そのため，多くのライセンスとそれらの派生に対応することができるようになり，ライセンス特定結果の精度を向上させることができる．

5.2 処　　理

提案手法ではソースファイルを入力とし，ライセンス名を出力する．提案手法では (1) コメント（Comment）の抽出，(2) コメントの文（Sentence）への分割，(3) 文のフィルタリング，(4) 文とメタライセンス文との照合，(5) メタライセンス文の系列（Sequence of meta license sentences）とライセンスルールとの比較を行う．以下，各手順について説明する．

(1) 初めに，ソースファイルからコメントを抽出する．提案手法ではソースコードからフォーマットやコメントを削除するユーティリティである Mangle [⋆1] を改変し，コメント抽出を行った．また，本ツールでは高速化のため，ソースファイル中で最初に現れたコメントの先頭から，次に現れるコードまでをコメントとして抽出する．

(2) 抽出したコメントを文へと分割する．ここでは，区切り文字 “. ”，“!”，“?”，“:” を検出し，区切り文字と区切り文字の間を文とする．また，文中の改行やタブは 1 つの空白にする．その結果，連続した空白が現れる場合は 1 つの空白に置き換える．

(3) 次に，文の集合から，ライセンスに関係のない文を取り除く．提案手法では，キーワード（Keyword）集合に含まれる単語をまったく含まない文はライセンスに関係のない文と見なす．**キーワード集合**とは，ライセンスに関係が深いと考えられる単語もしくは熟語の集合である．現在キーワード集合には，*“as is”*，*“wrote this file”*，*“acknowledgement”*，*“advertising”*，*“conditions”*，*“copies”* などの 82 語が含まれている．

(4) フィルタリング後，残った各文とメタライセンス文集合に含まれる各メタライセンス文の拡張正規表現パターンとマッチングを行い，マッチした場合は文をメタライセンス文名に置換し，メタライセンス文名の系列を得る．

(5) 最後に，メタライセンス文名の列とライセンスルール集合に含まれる各ライセンスルールのメタライセンス文名の列を比較し，一致したライセンスを出力する．

5.3 実　行　例

(1) 入力されたソースコードから以下のコメントを抽出したと仮定する．

```
Redistribution and use in source and binary forms, ...
 ...
1. Redistributions of source code must retain ...
 ... .
2. Redistributions in binary form must reproduce  ...
 ...
 ... .
```

(2) 初めに，コメントを文に分割する．上記のライセンス記述を分割するとき，*“Redistribution and use in source and binary forms, ...”*，*“1.”*，*“Redistributions of source code must retain ...”*，*“2.”*，*“Redistributions in binary form must reproduce ...”* の 5 文に分割される．

(3) 次のフィルタリングでは，*“1.”* と *“2.”* はキーワード集合に含まれる単語をまったく含まないため，これらの文を取り除く．次に，(4) 文とメタライセンス文との照合を行う．文 *“Redistribution and use in source and binary forms, ...”* は，メタライセンス文 *“BSDPre:Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met”* と同一なので，この文はメタライセンス文名 BSDPre に置換される．

(5) 得られたメタライセンス文の系列とライセンスルールとの比較を行う．メタライセンス文名の系列は *“AllRights, BSDpre, BSDcondSource, BSDcondBinary, BSDasIs, BSDWarr”* となっており，これは，ライセンスルール *“BSD2:BSDpre, BSDcondSource, BSDcondBinary, BSDasIs, BSDWarr”* とマッチするので，メタライセンス文名の系列は BSD2 に置換される．すなわち，このソースファイルは BSD2 ライセンスと特定される．

⋆1 http://gnu.triplemind.com/directory/devel/specific/mangle.html

6. 評価実験

6.1 ライセンス特定性能

本評価実験では，対象とするソースファイル集合に対し，本ツールと既存のライセンス特定ツールを適用し，ツールの出力とあらかじめ作成された正解集合を比較した．そして，得られた結果を回答率と正答率，所要時間で評価した．

ここでは，本ツールと FOSSology v1.0.0，ohcount v3.0.0rc，oslc v3 を用いた．これらのツールは異なる特徴が存在する．本ツールと Fossology では，ツールの結果として Unknown を出力する場合があるが，他の 2 つはつねに何らかのライセンスを出力する．OSLC はライセンスの一致をパーセントで示している．しかし，我々は他のツールと同条件にするため，OSLC が出力したライセンス名のみを特定した結果と見なした．

実験対象は以下の手順により作成した．初めに，Debian5.0.2 から 250 個のパッケージをランダムに選択した．次に，選択した各パッケージから 1 ファイルをランダムに選択した．これらの 250 ファイルの集合を N と呼ぶ．作成したサンプルは Debian5.0.2 の 80 万ファイルに依存することを考慮すると，信頼度 95%で誤差 ±6.2%である．

評価実験は以下の手順で行われた．初めに，N に含まれる各ソースファイルに対し，各ツールでライセンスを特定した．また，著者のうち，本ツールのライセンス知識の作成を主に行った著者とは異なる著者が手作業によりライセンスを特定した．本実験ではこの手作業によるライセンス特定結果を正解集合とした．

次に，各ツールから出力された結果と正解集合を比較した．そして，各ツールについて，各ファイルへのライセンス特定結果を以下の 4 つの集合に分類した．これらの集合は互いに重複しない．

C_v **ライセンス名と版が一致**　ツールの出力結果が，名前と版まで含めて正解と一致している．例として，正解が GPLv2 の場合，ツールの出力結果が GPLv2 である場合はこのクラスに分類される．また，ソースファイルにライセンス記述がなく，ツールの出力結果もライセンスなしに相当する結果だった場合もこのクラスに該当する．

C_n **ライセンス名のみが一致**　ツールの出力結果が正解と名前のみ一致している．例として，正解が GPLv2 の場合，ツールの出力結果が GPLv3 であればこのクラスに分類される．

I **不正解**　ツールの出力結果のライセンス名と正解のライセンス名が一致していない．例として，正解が GPLv2 の場合，ツールの出力結果が MPL v1.1 であればこのクラスに分類される．

表 2　各ツールの評価結果．同一の尺度で最も高かった値を太字で示した
Table 2　Result of accuracy evaluation. Bold means the highest number in each scale.

	本ツール	FOSSo.	ohcount	OSLC
$\|C_v\|$	200	137	83	57
$\|C_n\|$	1	84	137	8
$\|I\|$	6	28	30	185
$\|U\|$	43	1	0	0
回答率	82.8%	99.6%	**100.0**%	**100.0**%
正答率	**96.6**%	55.0%	33.2%	22.8%
正答率（名前のみ）	**97.1**%	88.7%	88.0%	26.0%
所要時間	**22 s**	923 s	27 s	372 s

U **不明**　ツールの出力結果として，不明が出力された場合，このクラスに分類される．

最後に，分類結果から回答率，正答率を求める．回答率，正答率は以下のとおりに定義する．

$$\text{回答率} = \frac{|C_v|+|C_n|+|I|}{|N|} \qquad \text{正答率} = \frac{|C_v|}{|N|-|U|}$$

$$\text{正答率（名前のみ）} = \frac{|C_v|+|C_n|}{|N|-|U|}$$

表 2 に実験結果を示す．本ツールは他のツールと比較して，回答率が低い．これは数多く不明を出力するからである．一方，名前のみを考慮した場合，名前と版の両方を考慮した場合において，本ツールは正答率では最も高かった．

実験結果から分かるように，本ツールはライセンス名と版の両方をかなり正確に特定することができる．ライセンス名のみであれば FOSSology や ohcount が優れており，その版に興味がないのであればこれらのツールの方が優れている．しかし，複数の版を持つライセンスは版によって条文が異なり，許諾条件も異なるため，ライセンス名だけでは目的を達成できない．

また，本ツールは他のツールに比べて回答率が低かった．これは，誤った結果を報告するよりはライセンスが不明であると報告する方がよいと我々は考えて設計したためである．なぜなら，ライセンスが未知であるソフトウェア部品をユーザは安易に調査せずに使用しようとは思わないが，明確にライセンスの特定結果が示されている場合，その結果を信用して安易に使用してしまうことが考えられるためである．

また，本ツールは 6 つのファイルのライセンスを特定できていない．これらの原因は，6 ファイル中 5 ファイルについてはライセンス文は存在していたが，メタライセンス文の不備によりライセンス文であると認識できなかったためである（メタライセンス文の変更，追加

表 3　スケーラビリティ評価の結果
Table 3　Result of scalability evaluation.

	本ツール	FOSSo.	ohcount	OSLC
所要時間	172 min	5,258 min	168 min	213 min

により容易に認識できるようになる）．また，他の 1 ファイルは通常のライセンスのようにライセンス文の系列としてライセンスを記述するのではなく，コメント文字列中にライセンス名が記述されていただけであったため認識できなかった．しかし，これらのファイルでは誤ったライセンスを特定していることはなかったため，本ツールの結果を信用しても使用者に不具合は生じない．

6.2　スケーラビリティ

提案手法のスケーラビリティを評価するため，大規模ソフトウェアに提案手法を適用し，所要時間と回答率を評価した．

評価では，Fedora [⋆1] 11 の SourceDVD [⋆2] に含まれるソースファイルに各ツールを適用し，所要時間と回答率を測定した．対象のソースファイルは 93,760 個であり，ファイルサイズの総計は 1.17 GB である．実験で使用した計算機の主要な性能は，CPU：Intel Core2Quad 2.4 GHz，メモリ：2 GB，OS：Ubuntu（Linux Kernel 2.6.28-17）である．

表 3 にその結果を示す．この結果から，本ツールは ohcount とほぼ同様に高速にライセンスを特定することができることが分かった．

また，この評価の際，Fedora11 Source DVD に含まれる各パッケージについて，回答率を調査した．調査により得られた度数分布を**図 4** に示す．グラフから 318 個のパッケージに対して，95%以上の回答率を示した．一方で 5%未満だったパッケージが 28 個あった．また，65%以上の回答率を示したパッケージが全体の 80.1%（457 個）を占めていた．

6.3　ライセンス知識の表現法

ライセンス知識の表現法の違いによる影響を評価するため，この評価では，各ツールがどのようにライセンス知識を表現しているか調査したうえで，いくつかのライセンスに対し，ライセンス知識のサイズがどのように異なるのか調査した．実験では，本手法を実装したツール，FOSSology，ohcount，OSLC，ASLA を対象とした．本調査におけるライセンス知識のサイズとは，あるライセンスに対するライセンス記述，その他のメタデータなども含む．ohcount はライセンス知識がメソッド呼び出しによりハードコーディングされている．このメソッドは引数としてライセンスの略称，ライセンスのページへの URL，ライセンス名，メタライセンス文を用いる．本実験では，これらのメソッド呼び出しで用いられている引数全体をライセンス知識とした．ASLA については実装が公開されていないため，不明である．

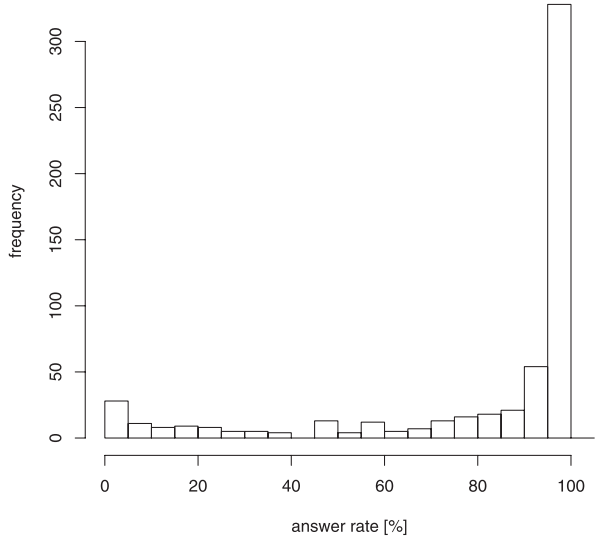

図 4　各パッケージの回答率に基づくヒストグラム
Fig. 4　Histogram based on answer rate of each package.

調査結果を**表 4** に示す．表中，本ツールのライセンス知識のサイズを “(ライセンスルールの合計サイズ) + (メタライセンス文の合計サイズ)” と表現した．ohcount は各ライセンスに対して 1 行程度の正規表現を用いており，最もライセンス知識のサイズが小さかった．ALSA も同様に正規表現を用いているが，そのサイズは不明である．参照文字列を用いる OSLC や FOSSology はライセンスへの参照を示す記述だけでなく，ライセンスの条項全文をライセンス記述として使用している場合があり，サイズが最も大きくなった．また，FOSSology は各ライセンスに対して複数のライセンス知識を持っているため，OSLC よりもサイズが大きかった．本ツールは 2 層のライセンス知識を用い，各ライセンスに対し複数のライセンス知識を持っているため，ohcount のライセンス知識よりもライセンス知識のサイズが大きかった．

⋆1 http://fedoraproject.org
⋆2 http://fedora.mirror.iweb.ca/releases/11/Fedora/source/iso/

表 4　既存手法と提案手法のライセンス知識のサイズ（文字数）
Table 4　The size of rules and simple regular expressions (# of characters).

	本ツール	FOSSo.	ohcount	OSLC	ASLA
ライセンス	2 層/正規表現	単層/参照文字列	単層/正規表現	単層/参照文字列	単層/正規表現
BSD3	267+4,032	3,223	147	1,930	不明
Apachev2	147+852	未実装	143	11,361	不明
X11	47+1,766	1,364	125	未実装	不明
MPLv1_1	15+191	24,293	151	23,650	不明

本ツールではライセンス知識の表現方法として 2 層のライセンス知識を用いている．そのため，既存のメタライセンス文を組み替えることで新しいライセンスルールを容易に定義できる．また，既知のメタライセンス文に類似した未知のライセンス文に対応するために正規表現を修正する場合でも，ライセンスルールに影響を与えない．

7. まとめと今後の課題

本稿では，階層的ライセンス知識を用いたライセンス特定ツールを提案した．大規模オープンソースソフトウェアを対象とし，ライセンス特定の課題をあげ，課題を解決するために必要なライセンス特定ツールの要件を述べた．そして，ライセンス特定ツールの要件に基づくライセンス特定ツールの設計について述べ，実装を行った．

評価では，大規模オープンソースソフトウェアを対象に特定の精度とスケーラビリティについて本ツールと従来手法との比較を行った．その結果，本ツールは従来手法と比べ高い精度で版も含めてライセンスを特定することができ，また，高速に処理することができた．さらに，大規模オープンソースソフトウェアに含まれていたパッケージのうち，大半のパッケージにおいてパッケージに含まれるソースファイルの 95%以上のソースファイルのライセンスを特定できていた．

今後の課題として，より多くの調査に基づくメタライセンス文やライセンスルール，キーワードの洗練がある．また，大規模オープンソースソフトウェアを対象としたライセンスに関する定量的な調査研究も重要な課題である．

謝辞　本研究は，日本学術振興会科学研究費補助金基盤研究（A）（課題番号：21240002）の助成を得た．また，本研究は，文部科学省グローバル COE プログラム（研究拠点形成費）の補助によるものである．

参 考 文 献

1) Braun, C.L.：再利用，ソフトウェア工学大辞典，Marciniak, J.J. (Ed.), Vol.1, pp.338–405, 朝倉書店 (1994).
2) Selby, R.W.: Enabling reuse-based software development of large-scale systems, *IEEE Trans. Softw. Eng.*, Vol.31, No.6, pp.495–510 (2005).
3) Li, J., Conradi, R., Bunse, C., Torchiano, M., Slyngstad, O. and Morisio, M.: Development with Off-the-Shelf Components: 10 Facts, *IEEE Software*, Vol.26, No.2, pp.80–87 (2009).
4) German, D.M., Manabe, Y. and Inoue, K.: A sentence-matching method for automatic license identification of source code files, *Proc. IEEE/ACM International Conf. on Automated Software Engineering* (*ASE 2010*), pp.437–446, ACM (2010).
5) 真鍋雄貴，German, D.M.，井上克郎：ライセンス知識に基づくライセンス特定ツールの設計，ウィンターワークショップ 2010・イン・倉敷論文集，Vol.2010, pp.19–20 (2010).
6) Gobeille, R.: The FOSSology project, *Proc. 2008 International Conf. on Mining Software Repositories* (*MSR 2008*), pp.47–50 (2008).
7) Tuunanen, T., Koskinen, J. and Kärkkäinen, T.: Automated software license analysis, *Automated Software Engineering*, Vol.16, No.3-4, pp.455–490 (2009).
8) Alspaugh, T.A., Asuncion, H.U. and Scacchi, W.: Analyzing software licenses in open architecture software systems, *Proc. 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development* (*FLOSS 2009*), pp.54–57 (2009).
9) German, D.M. and Hassan, A.E.: License integration patterns: Addressing license mismatches in component-based development, *Proc. 31st International Conf. on Software Engineering* (*ICSE 2009*), pp.188–198 (2009).

(平成 22 年 11 月 30 日受付)
(平成 23 年 5 月 14 日採録)

真鍋　雄貴（学生会員）
平成 18 年大阪大学基礎工学部情報科学科退学．現在，同大学大学院情報科学研究科博士後期課程 3 年．ソフトウェア再利用，ソフトウェアライセンスの研究に従事．ACM 会員．

ダニエル モラレス ゲルマン

2000 年ウォータールー大学博士課程修了．2001 年ヴィクトリア大学准教授．ソフトウェア進化，オープンソースソフトウェア，知的財産の研究に従事．

井上　克郎（フェロー）

昭和 54 年大阪大学基礎工学部情報工学科卒業．昭和 59 年同大学大学院博士課程修了．同年同大学基礎工学部情報工学科助手．昭和 59〜61 年ハワイ大学マノア校情報工学科助教授．平成元年大阪大学基礎工学部情報工学科講師．平成 3 年同学科助教授．平成 7 年同学科教授．平成 14 年大阪大学大学院情報科学研究科コンピュータサイエンス専攻教授．平成 20 年国立情報学研究所客員教授．同年情報処理学会フェロー．同年電子情報通信学会フェロー．工学博士．ソフトウェア工学の研究に従事．日本ソフトウェア科学会，電子情報通信学会，IEEE，ACM 各会員．