情報処理学会論文誌 Vol. 49 No. 7 2692–2702 (July 2008)

特許請求項における多重多数項引用の検出と書き換え

新森昭宏[†1] 大屋由香里[†1] 谷川英和[†2]

特許書類では「特許請求の範囲」に，発明を特定するために必要な事項を「請求項」（クレーム）ごとに区分して記載する．請求項には，独立して記述する形式（独立形式請求項）と，他の請求項を引用して記述する形式（引用形式請求項）とがある．引用形式請求項において，複数の請求項を引用する場合に「多数項引用」と呼び，さらに引用元が多数項引用となっている場合に「多重多数項引用」と呼ぶ．多重多数項引用は，日本では認められているが，アメリカでは認められていないため，多重多数項引用の形で記述された特許を，アメリカに出願する場合は「特許請求の範囲」を書き換える必要がある．実務上，3 つの書き換え法が使われているが，人手での書き換え作業は煩雑なものである．本稿では，多重多数項引用の検出と書き換えを自動的に行うアルゴリズムを提案する．提案アルゴリズムを，2004 年に公開された約 42 万件の公開特許公報からランダムに抽出した 100 件のデータと，データパターン分類を行って設定したサンプルデータを用いて評価し，その妥当性を確認した．

Detecting and Rewriting Multiple-multiple Dependent Form Claims

Akihiro Shinmori,[†1] Yukari Ooya[†1]
and Hidekazu Tanigawa[†2]

In patent specification documents, things to specify inventions are described as claims. There are two types of claims: “independent claims” that describe inventions independently and “dependent claims” that describe them by citing other claims. Dependent claims which cite multiple claims are called “multiple dependent claims”. If the cited claims are also multiple dependent claims, the citing claims are called “multiple-multiple dependent claims”. While multiple-multiple claims are legally accepted in Japan, they are not in the USA. Therefore if patents that include multiple-multiple claims are applied in the USA, the claims need to be rewritten. There are three kinds of rewriting methods used in practice, but it is cumbersome to rewrite claims manually. In this paper, we propose an algorithm to detect multiple-multiple dependent claims and rewrite them automatically. The proposed algorithm was evaluated by using 100 patent specification documents which are randomly selected from about 420 thousand documents disclosed in 2004 and by using sample data selected from data pattern analysis. The validity of the algorithm was confirmed by the evaluation.

1. はじめに

特許書類では「特許請求の範囲」に，発明を特定するために必要な事項を「請求項」（クレーム）ごとに区分して記載する．請求項には，独立して記述する形式（独立形式請求項）と，他の請求項を引用して記述する形式（引用形式請求項）とがある．引用形式請求項において，複数の請求項を引用するものは「多数項引用形式請求項」（multiple dependent form claim）と呼ばれている[1]．

図 1 に，文献 1) 記載の例に加筆して作成した多数項引用形式請求項の例を示す．図 1 において，請求項 4 と請求項 5 が多数項引用形式請求項である．

多数項引用形式請求項は日本でもアメリカでも認められているが，これを引用してさらに多数項引用形式請求項とすることはアメリカでは拒絶理由を有することとなる[2),3)]．実際，文献 2) には，“A multiple dependent claim shall not serve as a basis for any other multiple dependent claim.” と規定されている．また，文献 4) によれば，韓国においても同様の状況である．

本稿では，多数項引用形式請求項の引用元が多数項引用形式請求項となっている場合を「多重多数項引用」と呼ぶこととする．図 1 の例の場合，多数項引用形式請求項である請求項 5 は，多数項形式引用形式である請求項 4 を引用しているため，多重多数項引用となっている．

2004 年に公開された約 42 万件の公開特許公報からランダムに抽出した 100 件を調査したところ，44 件で多重多数項引用が使われていた．つまり，2004 年に公開された特許の 4 割以上で，多重多数項引用が使われている可能性がある．

特許は国ごとの制度であるため，日本で行った発明を使って外国でビジネスを行う場合

†1 株式会社インテックシステム研究所
INTEC Systems Institute, Inc.
†2 IRD 国際特許事務所
IRD Patent Office

【請求項 1】流体と石鹸を含んでなる混合物。
【請求項 2】流体が水である請求項 1 の混合物。
【請求項 3】染料をさらに含む請求項 2 の混合物。
【請求項 4】石鹸が苛性アルカリ石鹸である請求項 1 または請求項 2 記載の混合物。
【請求項 5】石鹸が特別加工した苛性アルカリ石鹸である請求項 1 または請求項 4 記載の混合物。

図 1　多数項引用と多重多数項引用の例
Fig. 1　An example of multiple dependent claim and multiple-multiple dependent claim.

は，日本での特許出願・取得だけでは不十分であり，それぞれの国で特許を出願・取得する必要がある．その場合，日本語で作成した特許明細書を翻訳する必要が生じるが，多重多数項引用となっている特許明細書を，アメリカや韓国など多重多数項引用が許容されない国に出願する場合は，翻訳に先立って，「特許請求の範囲」を書き換える必要がある．通常は，こうした書き換え作業は人手で実施されているが，その作業は煩雑なものであり，ミスを犯す可能性も高い[5]．実際，文献 5) は，「非常に手間がかかり，また，従属関係の修正でミスを犯す可能性が非常に高くなるので細心の注意が必要」と述べている．

また，文献 6) は，多重多数項引用は日本で許容されているとはいえ，修正の際に混乱が生じる可能性や，審査時に審査官に正しく判断されない可能性があるため，その使用を避けるようにアドバイスしている．このため，特許出願にあたっては，たとえアメリカに出願する特許でないとしても，請求項の引用関係を整理し，多重多数項引用の存在の有無を確認することは，意義が大きいと思われる．

本稿では，多重多数項引用を自動検出するアルゴリズムと，自動書き換えを行うためのアルゴリズムを提案する．このアルゴリズムにより，多重多数項引用となっている特許明細書を多重多数項引用が許容されない国に出願する場合におけるチェック作業と書き換え作業を効率化することが期待できる．また，海外出願をしない場合であっても，請求項の引用関係を把握し整理するためのチェック作業を効率化することが期待できる．

2 章では，特許請求項の引用関係を視覚的に表現する「クレームツリー」の作成と表示について述べる．3 章では，多重多数項引用について，実務で用いられている 3 つの書換法を説明する．4 章では，多重多数項引用の自動検出と自動書き換えのアルゴリズムを提案する．5 章では，評価と考察について述べる．6 章では，今後の課題について述べる．7 章では，関連研究について述べる．

2. クレームツリーの作成と表示

特許書類に記載された複数の特許請求項の引用関係を視覚的に表現したものを「クレームツリー」と呼ぶ．クレームツリーを作成するためには，引用形式請求項とその引用元を検出することが必要である．引用形式請求項とその引用元を検出するためには，**図 2** に示す記述パターンの有無を調べることになる．なお，図 2 では，Perl の正規表現を用いてパターンを表現している．たとえば，\d は，全角または半角の数字 1 文字にマッチする．また，「乃至」または「ないし」は，法律用語としては範囲指定を意味するということに留意されたい．

- 1 つの特許請求項を引用，または複数の特許請求項を「範囲指定形式」で引用している場合：
  請求項 \d+(から |〜| ー | 乃至 | ないし)(請求項)?\d+
- 複数の特許請求項を「選択形式」で引用している場合：
  請求項 \d+ (または | 又は |, |、)*(請求項)?\d+
- 上記 2 つが混合した場合：
  請求項 \d+ (から |〜| ー | 乃至 | ないし)(請求項)? \d+ (または | 又は |, |、)*(請求項)?\d+

図 2　引用形式請求項を検出するための記述パターン
Fig. 2　Description patterns for detecting dependent claims.

図 2 の 3 つの類型の例を以下に示す．

- 請求項 1 から請求項 3
- 請求項 1 または請求項 3 または請求項 4
- 請求項 1 から請求項 3，請求項 4

ここで，特許明細書は弁理士や知的財産権担当者などの専門家が作成したものだけでなく，個人発明家が作成したものや海外特許を翻訳して作成されたものがあることに留意する必要がある．すなわち，通常の記述スタイルを逸脱したものや記述エラーを含むものが確実に存在する．こうした特許明細書の場合，図 2 に示す記述パターンだけでは検出できない可能性がある．しかし，そのような事例は，きわめて少数かつ特異的であるため，図 2 に示す記述パターンを微修正することで対応可能である．たとえば，「請求項 2 乃至請求項 3」と書くべきところを，「乃至」の文字を間違えて「請求項 2 及至請求項 3」としている特許明細書についても，「範囲指定形式」として解釈したい場合は，1 番目のパターンに「及至」を追加することになる．

図 1 をクレームツリーで表現したものを**図 3** に示す．図 3 では，引用される請求項を左上に，引用する請求項を右下に配置し，引用関係が存在する請求項間を線で結ぶことで，請求項間の引用関係を表現している．この表現法は，本稿の著者らが発明し特許出願しているものであり[7]，クレームツリーを簡潔に分かりやすく表現することができる．この図において，独立形式請求項は最も左の列に配置され，引用形式請求項は左から 2 列目以降に配置

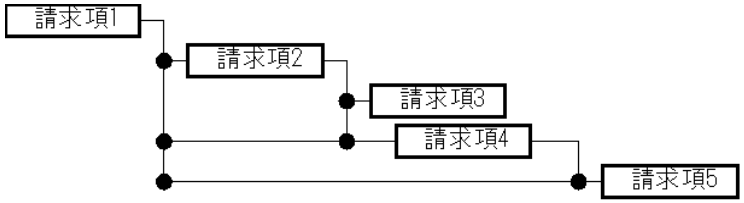


図 3　図 1 に対応するクレームツリー
Fig. 3　Claim tree corresponding to the claims in Fig. 1.

されることになる．任意の引用形式請求項から左横に延びる線上で，黒丸が書かれた箇所を上および左にたどることで，その引用形式請求項の引用元が分かるようになっている．つまり，図中で線上に書かれた黒丸は，電気回路図における結線状況と類似の意味合いを持っている．この表現法により，たとえば，請求項 3 は請求項 2 だけを引用していること，請求項 4 は請求項 1 と請求項 2 とを引用しているということが簡潔に示されている．

なお，図 3 には閉路（cycle）が存在するため，グラフ理論的な意味ではツリー（tree）ではない．しかし，弁理士などの知的財産権専門家の業界では「クレームツリー」という用語が習慣的に使われ，定着しているため，本稿でもそれにならうことにする．

3. 多重多数項引用の書き換え

多重多数項引用の形で記述された「特許請求の範囲」を書き換える場合，弁理士の実務では，**図 4** に示す 3 つの書き換え方法が使われている．

書換法 1 は，元々の意味内容を少し変更する形での書き換えとなるが，処理しやすく，かつ権利範囲が最も広い請求項の引用は残す形での書換法であるため多用されている．書換法 2 と書換法 3 は，元々の意味内容を変更させない形での書き換えであり，アメリカや韓国など多重多数項引用が許容されない国に出願する場合における拒絶理由を排除することができる．書換法 2 は，書換法 3 に比べて最小限の書き換えを行うところが異なる．書換法 3 は，書き換え後の請求項数が最も多くなるが，多数項引用形式請求項を完全になくす形で書き換えが行われる．なお，書換法 1 においても，多数項引用形式請求項を完全になくす形で書き換えが行われる．

このような書き換えを自動的に行うためのアルゴリズムを開発し，ツールを実装した．このアルゴリズムについては次の章で述べる．ツールの実装においては，書き換えた請求項には，「(書換済)」という文字列を先頭に挿入することとした．

書換法 1　すべての多数項引用形式請求項について，引用元の請求項のうち，最上位のものだけを引用するように書き換える
書換法 2　多重多数項引用となっている多数項引用形式請求項を書き換える
書換法 3　すべての多数項引用形式請求項を書き換える

図 4　実務で使われている 3 つの書換法
Fig. 4　Three rewriting methods used in practice.

【請求項 1】流体と石鹸を含んでなる混合物。
【請求項 2】流体が水である請求項 1 の混合物。
【請求項 3】染料をさらに含む請求項 2 の混合物。
【請求項 4】（書換済）石鹸が苛性アルカリ石鹸である請求項 1 記載の混合物。
【請求項 5】（書換済）石鹸が特別加工した苛性アルカリ石鹸である請求項 1 記載の混合物。

図 5　図 1 に対して書換法 1 を適用した結果
Fig. 5　Result of application of the method 1 for Fig. 1.

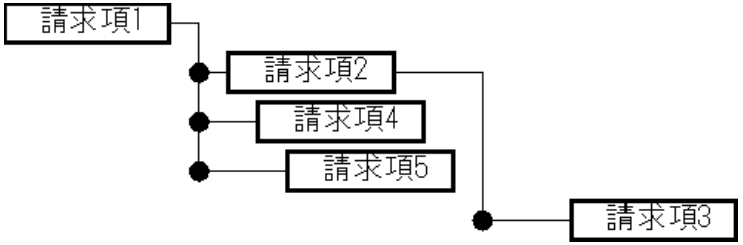


図 6　図 5 に対応するクレームツリー
Fig. 6　Claim tree corresponding to the claims in Fig. 5.

【請求項 1】流体と石鹸を含んでなる混合物。
【請求項 2】流体が水である請求項 1 の混合物。
【請求項 3】染料をさらに含む請求項 2 の混合物。
【請求項 4】石鹸が苛性アルカリ石鹸である請求項 1 または請求項 2 記載の混合物。
【請求項 5】（書換済）石鹸が特別加工した苛性アルカリ石鹸である請求項 1 記載の混合物。
【請求項 6】（書換済）石鹸が特別加工した苛性アルカリ石鹸である請求項 4 記載の混合物。

図 7　図 1 に対して書換法 2 を適用した結果
Fig. 7　Result of application of the method 2 for Fig. 1.

図 1 に対して上記の 3 つの書換法を適用した結果とそれぞれのクレームツリーを**図 5**，**図 6**，**図 7**，**図 8**，**図 9**，**図 10** に示す．

図 7 では，旧請求項 5 が新請求項 5 と新請求項 6 とに展開され，全体の請求項数が 1 つ増加している．また，図 9 では，旧請求項 4 が新請求項 4 と新請求項 5 とに展開され，さ

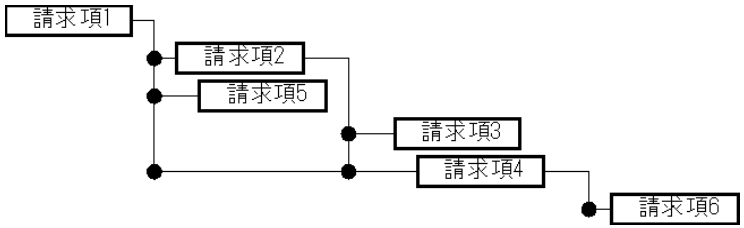


図 8　図 7 に対応するクレームツリー
Fig. 8　Claim tree corresponding to the claims in Fig. 7.

【請求項 1】流体と石鹸を含んでなる混合物。
【請求項 2】流体が水である請求項 1 の混合物。
【請求項 3】染料をさらに含む請求項 2 の混合物。
【請求項 4】（書換済）石鹸が苛性アルカリ石鹸である請求項 1 記載の混合物。
【請求項 5】（書換済）石鹸が苛性アルカリ石鹸である請求項 2 記載の混合物。
【請求項 6】（書換済）石鹸が特別加工した苛性アルカリ石鹸である請求項 1 記載の混合物。
【請求項 7】（書換済）石鹸が特別加工した苛性アルカリ石鹸である請求項 4 記載の混合物。
【請求項 8】（書換済）石鹸が特別加工した苛性アルカリ石鹸である請求項 5 記載の混合物。

図 9　図 1 に対して書換法 3 を適用した結果
Fig. 9　Result of application of the method 3 for Fig. 9.

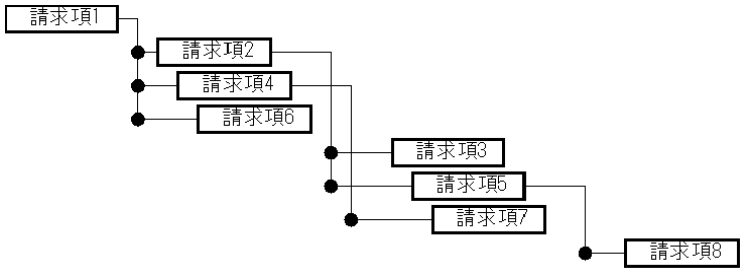


図 10　図 9 に対応するクレームツリー
Fig. 10　Claim tree corresponding to the claims in Fig. 9.

らに旧請求項 5 が，新請求項 6，新請求項 7，新請求項 8 へと展開されている．その結果，全体の請求項数が 3 つ増加している．ここで，旧請求項 5 の新請求項 7 と新請求項 8 への展開は，旧請求項 4 の展開にともなって生じているものであることに留意されたい．

4. 自動検出と自動書き換えのアルゴリズム

4.1 自動検出アルゴリズム

自動検出を行うにあたり，2 章に述べたように，引用関係を解析したうえで，多数項引用の検出を行う．その後，多重多数項引用の検出を行う．**図 11** に自動検出のアルゴリズムを示す．

本アルゴリズムは，引用テーブル，多数項引用テーブル，多重多数項引用チェックテーブルの 3 つのテーブルを使用する．引用テーブルは，各請求項が引用している請求項のリストを保持する．多数項引用チェックテーブルと多重多数項引用チェックテーブルはそれぞれ，多数項引用と多重多数項引用の有無を記録するために用いる．

なお，本アルゴリズムでは，多重多数項引用だけでなく多数項引用も検出するようにしている．検出された多数項引用の請求項番号の集合は，次に述べる自動書き換えアルゴリズムの書換法 3 の入力として使用される．

4.2 自動書き換えアルゴリズム

図 3 に示したクレームツリーでは，多数項引用は 1 つの閉路を形成し，多重多数項引用は隣接する 2 つの閉路を形成している．したがって，図 3 の多重多数項引用の書き換えは，こうした閉路を解消し，ツリーに変換する処理である．しかしながら，**図 12** に示すように，複数の独立形式請求項を引用する形で多数項引用が形成される場合もあることを考慮する必要がある．

書換法 1 は，図 11 で作成された引用テーブルを用いることで容易に実現可能である．書換法 2 と書換法 3 は，書き換え対象として，多重多数項引用の請求項の集合を入力するか，多数項引用の請求項の集合を入力するかが違うだけで，書き換えのアルゴリズム自体は同じである．また，図 5，図 7，図 9 に例示されているように，書換法 1 においては請求項の数が変化しないのに対し，書換法 2 と書換法 3 においては，書き換え対象の請求項が展開されるため請求項の数が増加する．

書き換え対象の請求項が展開される場合，それを引用している請求項も展開される必要がある．また，請求項数が増加することにともなって，請求項の番号が 1 から順に正しく振り直される必要があり，かつ引用形式請求項における引用番号も正しく振り直される必要がある．つまり，自動書き換え処理では，以下の 4 点を達成する必要がある．

展開性　書き換え対象の請求項を正しく展開する．

展開多重性　書き換え対象の請求項の展開にともなって，それを引用している請求項も展開

入力：
- 特許請求項テキストの配列 $claim[k], (k = 1, \cdots, n)$ 但し，n は請求項の数

出力：
- 多重多数項引用である請求項番号の集合
- 多数項引用である請求項番号の集合

使用するデータ構造：
- 引用テーブル $citation[k], (k = 1, \ldots, n)$（初期値をすべて，空リストにセットする）
- 多数項引用チェックテーブル $multi\_citation[k], (k = 1, \ldots, n)$（初期値をすべて，$-1$ にセットする）
- 多重多数項引用チェックテーブル $multi\_multi\_citation[k], (k = 1, \ldots, n)$（初期値をすべて，$-1$ にセットする）

処理：

```
(1)  for i = 1 to n do
(2)    claim[i] のテキストに対して，図 2 のパターンを用いて
       マッチングを行い，引用元の請求項のリストを citation[i]
       の値としてセットする．
(3)  end for
(4)  for i = 1 to n do
(5)    if |citation[i]| ≥ 2 の then
(6)      multi_citation[i] = 1
(7)    end if
(8)  end for
(9)  for i = 1 to n do
(10)   if multi_citation[i] = 1 then
(11)     foreach j in citation[i] do
(12)       if multi_citation[j] = 1 then
(13)         multi_multi_citation[i] = 1
(14)       end if
(15)     end foreach
(16)   end if
(17) end for
(18) 多重多数項引用チェックテーブルと多数項引用チェックテーブルで値が 1 となっている添字番号の集合
     をそれぞれ，多重多数項引用となっている請求項番号の集合，多数項引用となっている請求項番号の集
     合として返す
```

図 11　多重項引用と多重多数項引用の自動検出アルゴリズム

Fig. 11　Algorithm for detecting multiple dependent and multiple-multiple dependent forms.

する必要が生じる．たとえば，請求項 i が書き換え対象であった場合，請求項 $j(j > i)$ が請求項 i を引用しているのであれば，請求項 j は書き換え対象でなくても展開される必要がある．その処理を正しく行う．

連番正当性　書き換えによる請求項数の増加にともない，請求項の連番を，1 から最大請求項番号までの間で，抜けなく昇順で正しく設定する．

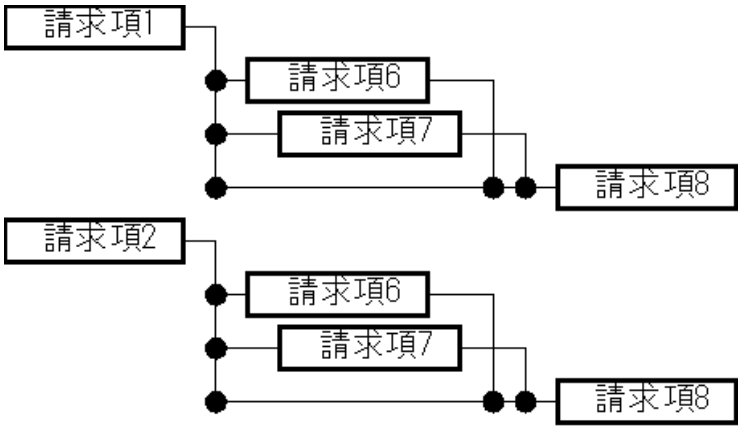


図 12　複数の独立形式請求項を引用して多数項引用と多重多数項引用が形成されている例（特開 2004-174526 のクレームツリーの一部）

Fig. 12　An example in which multiple dependent and multiple-multiple dependent form claims are created citing two independent claims (A portion of claim tree of publication number 2004-174526).

引用番号正当性　書き換えによる請求項番号の変化にともない，引用形式請求項の引用番号を正しく設定する．

以上をふまえ，書換法 2 と書換法 3 について開発した自動書き換えアルゴリズムを**図 13** と**図 14** に示す．本アルゴリズムでは，引用状況や変換状況，変換後の請求項番号の情報を管理するために，引用テーブル，展開チェックテーブル，請求項番号変換テーブル，引用変換テーブルの 4 つのテーブルを使用する．また，請求項文字列の書き換えを行うために，「引用箇所」より前の文字列と後の文字列をそれぞれ格納した 2 つのテーブルを使用する．ここで，「引用箇所」とは，図 2 に示す記述パターンにマッチした箇所のことである．

引用テーブルは，自動検出アルゴリズムの場合と同じである．図 11 と同様の処理によりセットされる．

展開チェックテーブルは，書き換え処理によって請求項が展開されるかどうかを記録するためのものである．その際，展開多重性を考慮した処理を行う（7 行目から 11 行目）．

請求項番号変換テーブルは，各請求項が自動書き換え後にどういう番号の請求項に変換されるかについてのリストを保持する．請求項番号変換テーブルは，図 13 の 14 行目から 29 行目までセットされる．請求項番号変換テーブルをセットする処理においては，まず展開

入力：

- 特許請求項テキストの配列 $claim[k]$, $(k = 1, \cdots, n)$ 但し，n は特許請求項の数
- 書換対象の請求項番号の集合 T

出力：

- 書換後の特許請求項テキストの配列 $claim\_mod[k]$, $(k = 1, \cdots, m)$ 但し，m は書換後の請求項の数

使用するデータ構造：

- 引用テーブル（図 11 と同じ）
- 展開チェックテーブル $claim\_expand[k]$ $(k = 1, \cdots, n)$（初期値をすべて 0 にセットする）
- 請求項番号変換テーブル $claim\_trans[k]$ $(k = 1, \cdots, n)$
- 引用変換テーブル $citation\_trans[k]$ $(k = 1, \cdots, n)$
- 「引用箇所」より前の文字列を格納したテーブル $before\_str[k]$ $(k = 1, \cdots, n)$（初期値をすべて空文字列にセットする）
- 「引用箇所」より後の文字列を格納したテーブル $after\_str[k]$ $(k = 1, \cdots, n)$（初期値をすべて空文字列にセットする）

処理：

```
<      引用テーブルのセット    >
(1)    図 11 の (1), (2), (3) の処理で，citation を初期化する．かつ，その際に，図 2 のパターンにマッチした文字列の
       前の文字列を before_str に，後の文字列を after_str に格納する．
<      展開チェックテーブルのセット    >
(2)    for i =(T の最小要素) to n do
(3)      if |citation[i]| = 0 then 次の for ループへ移る
(4)      if i ∈ T then
(5)        claim_expand[i] = 1
(6)      else
(7)        foreach j in citation[i] do
(8)          if claim_expand[j] = 1 then
(9)            claim_expand[i] = 1
(10)         end if
(11)       end foreach
(12)     end if
(13)   end for
<      請求項番号変換テーブルのセット    >
(14)   seqno = 1
(15)   for i = 1 to n do
(16)     if claim_expand[i] = 0 then
(17)       claim_trans[i] = (seqno + +)
(18)     else
(19)       foreach j in citation[i] do
(20)         if claim_expand[j] = 0 then
(21)           claim_trans[i] の要素として seqno を追加し，
               seqno + +
(22)         else
(23)           for k = 1 to |claim_trans[j]| do
(24)             claim_trans[i] の要素として seqno を追加し，
                 seqno + +
(25)           end for
(26)         end if
(27)       end foreach
(28)     end if
(29)   end for
```

図 13　書換法 2 と書換法 3 に対する自動書き換えアルゴリズム（前半）
Fig. 13　Rewriting algorithm for the rewriting methods 2 and 3 (first half).

処理（続く）：

```
<      引用変換テーブルのセット    >
(30)   for i = 1 to n do
(31)     if |citation[i]| = 0 then 次の for ループへ移る
(32)     foreach j in citation[i] do
(33)       if |citation[j]| = 0 then
(34)         citation_trans[i] = claim_trans[j]
(35)       else
(36)         for k = 1 to |claim_trans[i]| do
(37)           citation_trans[i] に claim_trans[k] を追加
(38)         end for
(39)       end if
(40)     end foreach
(41)   end for
<      書換処理    >
(42)   for i = 1 to n do
(43)     if i < (T の最小要素) then
(44)       claim_mod[i] = claim[i]
(45)     else
(46)       if |claim_trans[i]| = 1 then
(47)         if claim[i] は独立形式請求項 then
(48)           claim_mod[claim_trans[i] の先頭の要素 ] =
               claim[i] の先頭に（書換済）をつけたもの
(49)         else
(50)           claim_mod[claim_trans[i] の先頭の要素 ] =
               claim[i] の引用している請求項の番号を
               citation_trans[i] の値に置き換えた書換文字列
               を生成し，その先頭に（書換済）をつけたもの．
               （書換文字列は，before_str[i],「引用箇所を
               変換した文字列」，after_str[i] を連結して
               生成する．「引用箇所を変換した文字列」は，
               citation_trans[i] の要素が 1 のときはそのまま，
               2 以上のときは各番号の先頭に「請求項」という文
               字をつけ，「または」で連結して生成する）
(51)         end if
(52)       else
(53)         for j = 1 in |claim_trans[i]| do
(54)           claim_mod[claim_trans[i] の j 番目の要素 ] =
               claim[i] が citation_trans[i] の j 番目の要素
               だけを引用するようにした書換文字列を生成し，そ
               の先頭に（書換済）をつけたもの
               （書換文字列は，before_str[i],「引用箇所を
               変換した文字列」，after_str[i] を連結して
               生成する．「引用箇所を変換した文字列」は，
               citation_trans[i][j] の先頭に「請求項」という
               文字をつけて生成する）
(55)         end foreach
(56)       end if
(57)     end if
(58)   end for
```

図 14　書換法 2 と書換法 3 に対する自動書き換えアルゴリズム（後半）
Fig. 14　Rewriting algorithm for the rewriting methods 2 and 3 (last half).

チェックテーブルを参照し（16 行目），その値が 0 のとき，つまり書き換え処理によって展開されない請求項については単純に連番（*seqno*）を割り当てる（16 行目と 17 行目）．書き換え処理により展開される請求項については，それが引用している請求項をチェックし（19 行目），書き換え処理によって展開されないものの場合は連番を追加する（21 行目）．書き換え処理によって展開されるものの場合は，それらが展開される分の連番を追加する（24 行目）．ここで，23 行目と 24 行目の処理がポイントであり，そこではこれまでにセットした請求項番号変換テーブルのエントリを参照している．これは，引用形式請求項は，それ以前に記述した請求項を引用するという特性[8] を利用している．

引用変換テーブルは，引用番号正当性を達成するために，自動書き換え後の引用形式請求項が引用する請求項番号のリストを保持する．このテーブルの作成は，引用テーブルと請求項番号変換テーブルを参照しながら行う．まず，引用形式請求項において，それが引用している請求項が引用形式でない場合は請求項番号変換テーブルの該当するエントリをコピーする（34 行目）．引用形式の場合は，引用している請求項ごとに請求項番号変換テーブルの該当するエントリをコピーする（36 行目から 38 行目）．

引用テーブル，展開チェックテーブル，請求項番号変換テーブル，引用変換テーブルがすべてセットされた後で，書き換え処理を行う．書き換え処理においては，書き換え対象の請求項番号集合の最小要素に達するまでは，書き換え処理を行わない（44 行目）．最小要素に達した以降は，請求項番号変換テーブルのエントリの数が 1 であるかどうかによって，展開する必要があるかどうかを判定する（46 行目）．展開する必要がない場合で，かつ独立形式請求項の場合は，請求項番号を修正するだけでよいため，元々の請求項をそのままコピーして，先頭に（書換済）をつける（48 行目）．展開する必要がない場合で，かつ引用形式請求項の場合は，元々の請求項の引用する請求項番号を引用変換テーブルの該当するエントリの値に置き換え，先頭に（書換済）をつける（50 行目）．展開する必要がある場合は，請求項番号変換テーブルのエントリの数だけ展開処理を行い，展開後の請求項が引用する請求項番号を引用変換テーブルを参照しながら設定する（54 行目）．

図 1 に対して，書換法 3 で書き換えを行う際に作成される引用テーブル，展開チェックテーブル，請求項番号変換テーブル，引用変換テーブルを**表 1**，**表 2**，**表 3**，**表 4** にそれぞれ示す．書換法 3 であるので，書き換え対象となるのは請求項 4 と請求項 5 であり，$T = (4, 5)$ である．表 3 により，請求項 4 は請求項 4 と請求項 5 とに，請求項 5 は請求項 6 と請求項 7 と請求項 8 とに書き換えられることが分かる．また，表 4 により，請求項 4 は書き換え後にそれぞれ請求項 1 と請求項 2 を，請求項 5 は書き換え後にそれぞれ請求項 1，請求項 2，請求項 5 を引用することになるということが分かる．

表 1　図 1 に対応する引用テーブル
Table 1　Citation table for Fig. 1.

請求項番号	引用している請求項リスト
1	()
2	(1)
3	(2)
4	(1,2)
5	(1,4)

表 2　図 1 に対応する展開チェックテーブル
Table 2　Claim expand table for Fig. 1.

請求項番号	請求項が展開されるかどうか
1	(0)
2	(0)
3	(0)
4	(1)
5	(1)

表 3　図 1 に対応する請求項番号変換テーブル
Table 3　Claim translation table for Fig. 1.

請求項番号	自動書き換え後の請求項番号のリスト
1	(1)
2	(2)
3	(3)
4	(4,5)
5	(6,7,8)

表 4　図 1 に対応する引用変換テーブル
Table 4　Citation translation table for Fig. 1.

請求項番号	自動書き換え後の引用請求項番号のリスト
1	()
2	(1)
3	(2)
4	(1,2)
5	(1,4,5)

5. 評価と考察

5.1 評　　価

本節では，自動検出と自動書き換えのアルゴリズムの妥当性に関する評価方法とその結果に関して述べる．

自動検出のアルゴリズムの評価にあたっては，多重多数項引用を正しく検出できるかどうかが評価のポイントとなる．自動書き換えのアルゴリズムの評価にあたっては，前章で説明した展開性，展開多重性，連番正当性，引用番号正当性がすべて達成されているかどうかが評価のポイントとなる．

自動検出については，ランダムに抽出したデータによる評価を行った．自動書き換えについては，ランダムに抽出したデータによる評価と，データパターン分類から設定したサンプルデータによる評価を行った．以下にその結果を述べる．

5.1.1 ランダムデータによる評価

2004 年に公開された約 42 万件の公開特許公報からランダムに抽出した 100 件のデータを用いて評価を行った．なおこれは，1 章で言及したデータである．

偏ったデータが抽出されていないことを確認するために，国際特許分類（IPC）コードの先頭の記号（A から H まで）を抽出してその分布を調べた．2004 年に公開された特許は，2002 年と 2003 年に出願されたものであり，特許庁が公開している 2002 年と 2003 年の年間特許出願データの分布との相関を調べることで，このデータが全体傾向を反映したものかどうかが分かる．結果は，相関係数 0.84 という値であり，全体傾向を反映したものであるといえる．

1 章で述べたとおり，100 件のデータ中，44 件の特許明細書で多重多数項引用が存在していた．自動検出アルゴリズムはそのすべてを正しく検出した．自動書き換えアルゴリズムは，後述するように多くの請求項を生成する場合があり，その結果をすべて確認することは困難であるが，書き換え対象となる請求項からランダムに抽出した 10 件については正しく書き換えられていることを確認した．

5.1.2 データパターン分類から設定したサンプルデータによる評価

対象となるデータのパターン分類を行い，それぞれのサンプルデータによる評価を行った．パターン分類を**図 15** に，そのイメージを**図 16** に示す．なお，図 16 では，書き換え対象の請求項を斜線を入れた箱で，書き換え対象でない請求項を白の箱で表現している．

評価は，図 15 の 8 つのパターンについて，それぞれサンプルデータを設定して実施した．

Pattern-1　書換対象の請求項が 1 つの場合
- **Pattern-1-1**　書換対象の請求項を引用している請求項が存在しない場合
- **Pattern-1-2**　書換対象の請求項を引用している請求項が 1 つ存在する場合
- **Pattern-1-3**　書換対象の請求項を引用している請求項が 2 つ以上存在する場合

Pattern-2　書換対象の請求項が 2 つ以上の場合
- **Pattern-2-1**　書換対象の複数の請求項に引用関係が無い場合
- **Pattern-2-2**　書換対象の複数の請求項に引用関係が有る場合
  - **Pattern-2-2-1**　書換対象の請求項の 1 つを，別の書換対象の請求項が直接引用している場合
  - **Pattern-2-2-2**　書換対象の請求項の 1 つを，別の書換対象の請求項が，更に別の請求項を経由して間接的に引用している場合
  - **Pattern-2-2-3**　書換対象の請求項の 2 つ以上を，別の書換対象の請求項が直接引用している場合
  - **Pattern-2-2-4**　書換対象の請求項の 2 つ以上を，別の書換対象の請求項が，更に別の請求項を経由して間接的に引用している場合

図 15　書き換えデータのパターン分類
Fig. 15　Data pattern classification for rewriting data.

サンプルデータはパターンごとに，人為的に用意した 1 件と，2004 年に公開された 42 万件の公開特許公報の中から前項で用いたものとは違うデータとしてランダムに抽出した 2 件の，合計 3 件を使用した．その際，図 15 で，「2 つ以上」とある部分については 2 または 3 としてサンプルデータを設定または抽出した．その結果，すべてのサンプルデータの書き換え処理において，展開性，展開多重性，連番正当性，引用番号正当性の 4 つが達成されていることを確認した．

5.2 考　　察

5.2.1 多重多数項引用の書き換えについて

本稿で提案した，書換法 2 と書換法 3 に対する自動書き換えアルゴリズムは，請求項の数を大幅に増加させてしまう場合がある．特に，特許請求項の引用が多段階に行われており，かつ，多重多数項引用の構造も多段階になっている場合，多くの請求項が生成されることになる．前節での評価データにおける請求項数に関する統計情報を**表 5** に示す．

実際，前節の評価データの中には，書換法 2 による自動書き換えで 1,296 個，書換法 3 による自動書き換えで 2,448 個の請求項が生成されたものがあった．この明細書は，請求項数が 16，うち独立請求項数が 2，クレームツリーの引用の深さが 14，多数項引用の請求項数が 9，多重多数項引用の請求項数が 8 という複雑なものであった．

特許出願料は一般に，請求項の数が増加するほど高くなる．実際，アメリカの特許出願費用は，請求項数が 20 を超えた時点で急に高くなるため[2),3)]，実務上は，各請求項の重要性と出願費用を考慮しながら，請求数が 20 を超えないようにすることが望ましい．

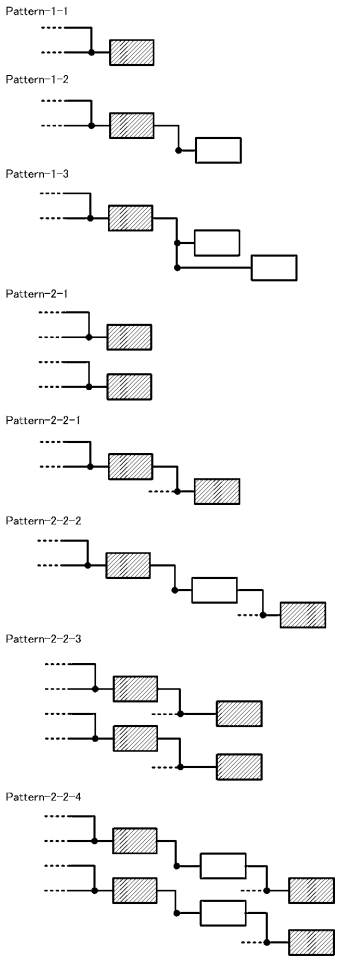


図 16　書き換えデータのパターン分類のイメージ
Fig. 16　Image of data pattern classification for rewriting data.

本稿で提案した書換法 2 と書換法 3 に対する自動書き換えアルゴリズムは，このような観点は考慮せず，機械的な書き換えを行う．したがって，実務での利用の際には，書き換え後の請求項数が 20 を超えた場合に，人手でのチューニング処理を行う場合が多い．しかし，

表 5　評価データにおける請求項数に関する統計情報
Table 5　Statistics on the number of claims for the evaluation data.

		書換法 2	書換法 3
処理対象明細書数		44	65
書き換え前	請求項数の平均値	9.3	8.0
	請求項数の最大値	29	29
	請求項数 20 以下の明細書数	43 (97.7%)	61 (98.4%)
書き換え後	請求項数の平均値	78.4	90.2
	請求項数の最大値	1,296	2,448
	請求項数 20 以下の明細書数	19 (43.2%)	35 (53.8%)

その場合であっても，このような自動書き換えアルゴリズムを使用しないでまったく人手で書き換えを行うことに比べると，作業効率は大幅に改善される．また，表 5 に示すように，書き換え後の請求項数が 20 を超えないものが，書換法 2 の場合は 43.2%，書換法 3 の場合は 53.8%存在しており，それらの明細書については，自動書き換え結果をそのまま使用することができる．

6. 今後の課題

本稿で提案した書換法 2 に対する自動書き換えアルゴリズムは，引用元の請求項を区別することなく機械的な書き換えを行う．引用元の請求項のうち，多数項引用のものとそうでないものを区別し，多重多数項を解消するために必要最小限の展開だけを行うようにアルゴリズムとデータ構造を改造することで，書き換え後に生成される請求項数を抑制することができる可能性がある．あるいは，本稿で提案したアルゴリズムで自動書き換えした後の処理として，複数の請求項を 1 つにまとめる処理を行うということも考えられる．このような処理を行うことで，出願費用が請求項の数に比例しているような特許制度の場合は，請求項数を抑制することでコスト削減につながる．

たとえば，図 1 を少し変形して，請求項 5 が請求項 2 も引用している形である**図 17** を考える．この場合，書換法 2 を適用した結果は**図 18** となる．この中で請求項 5 と請求項 6 とは共に多数項引用ではなく，かつ請求項 1 と請求項 2 も多数項引用ではないので，これらをまとめて**図 19** のようにしてもよく，かつ，これにより，請求項数を減らすことができる．

また，何らかの方法により個々の請求項の重要度，またはそれが規定する権利範囲の広さを自動判定することができれば，書き換え後に生成される請求項のうち，重要度の低いもの

･･･（注： 請求項 4 までは，図 1 と同じ）
【請求項 5】流水が水であり，石鹸が特別加工した苛性アルカリ石鹸である請求項 1 または請求項 2 または請求項 4 記載の混合物。

図 17　図 1 を変形した例
Fig. 17　An example modified from Fig. 1.

･･･（注： 請求項 4 までは，図 7 と同じ）
【請求項 5】（書換済）流水が水であり，石鹸が特別加工した苛性アルカリ石鹸である請求項 1 記載の混合物。
【請求項 6】（書換済）流水が水であり，石鹸が特別加工した苛性アルカリ石鹸である請求項 2 記載の混合物。
【請求項 7】（書換済）流水が水であり，石鹸が特別加工した苛性アルカリ石鹸である請求項 4 記載の混合物。

図 18　図 17 に対して書換法 2 を適用した結果
Fig. 18　Result of application of the method 2 for Fig. 17.

･･･（注： 請求項 4 までは，図 18 と同じ）
【請求項 5】（書換済）流水が水であり，石鹸が特別加工した苛性アルカリ石鹸である請求項 1 または請求項 2 記載の混合物。
【請求項 6】（書換済）流水が水であり，石鹸が特別加工した苛性アルカリ石鹸である請求項 4 記載の混合物。

図 19　図 18 の請求項 5 と請求項 6 をまとめた結果
Fig. 19　Result of merging of the claim 5 and the claim 6 in Fig. 18.

や規定する権利範囲の狭いものを削除する処理が実現できる．しかし，請求項については，表現上の特徴を利用した構造解析が可能となった段階であり[9]，その意味に踏み込んで重要度を判定することは今後の課題である．

7. 関 連 研 究

特許請求項における多重多数項引用の存在とその書き換えに関する課題については，弁理士などの実務家の間ではよく知られていたが，情報処理技術の観点からその検出と書き換えを行う手法に関する研究，特許出願，ツール，商用製品などは，我々の知る限り存在しない．

なお，特許請求項間の引用関係を解析してクレームツリーを表示するプログラムについては，特許出願が行われているほか[10]，「クレームチャート」として表示を行う商用製品が販売されている[11]．しかし，電気回路図における結線状況を示すような形でクレームツリーを表示することで複雑な引用関係を簡潔に表示する手法は，本稿で述べた研究に関連して我々が発明した手法である．

そのほか，特許書類を対象とした言語処理の研究としては，特に特許請求項に着目し，その可読性向上を目的とした構造解析に関する研究[9),12)]や，特許検索への応用のための構造解析に関する研究がある[13]．

8. お わ り に

多重多数項引用の検出と書き換えを自動的に行うアルゴリズムを提案し，評価によってその妥当性を確認した．今後の課題も残っているが，現段階では十分に実用に耐えるアルゴリズムである．

本研究は，特許に関する工学的アプローチである「特許工学」[14]の確立に向けた活動の一環である．特許ライフサイクルを支援するツール群の整備を更に進める必要があると考えている．

謝辞　本研究は，独立行政法人情報処理推進機構（IPA）2006 年度次世代ソフトウェア開発事業の成果を応用発展させたものです．各種の助言をいただいた，東京工業大学の奥村学准教授，広島市立大学の難波英嗣講師に感謝いたします．

参 考 文 献

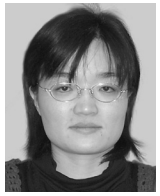
1) 竹田和彦：特許の知識（第 6 版），ダイヤモンド社 (2000).
2) Patent, U.S. and Office, T.: CONSOLIDATED PATENT LAWS, United States Code Title 35 – Patents (2007). http://www.uspto.gov/web/patents/legis.htm
3) 木梨貞男：米国特許入門，工業調査会 (2001).
4) 金　成鎬：日本企業が注意すべき韓国独特の制度，パテント，Vol.56, No.3, pp.7–12 (2003).
5) 倉増　一：特許翻訳の基礎と応用，pp.102–103, 講談社 (2006).
6) 葛西泰二：特許明細書のクレーム作成マニュアル，工業調査会 (1999).
7) 特願 2007-032913：クレーム従属関係図生成装置，クレーム従属関係図生成方法，請求項検出装置，請求項検出方法，クレーム従属関係変更装置，クレーム従属関係変更方法，及びプログラム (2007).
8) 特許庁：特許・実用新案審査基準 (2007). http://www.jpo.go.jp/shiryou/kijun/kijun2/tukujitu_kijun.htm
9) 新森昭宏，奥村　学，丸川雄三，岩山　真：手がかり句を用いた特許請求項の構造解析，情報処理学会論文誌，Vol.45, No.3, pp.891–905 (2004).
10) 特開 2005-4794，情報処理装置およびプログラム (2005).
11) インパテック株式会社：特許請求項分析支援ソフト「クレームチャート」. http://www.inpatec.co.jp/service/ccl.htm
12) 小西一也，新海正吾，高木　通，三部康夫：特許調査を効率化する請求項理解支援機能，情報処理学会研究報告–電子化知的財産・社会基盤，No.32–5, pp.19–26 (2006).
13) Konishi, K.: Query Terms Extraction from Patent Document for Invalidity Search,

Proc. NTCIR-6 Workshop Meeting, National Institute of Informatics (2005).
14) 谷川英和，河本欣司：特許工学入門，中央経済社 (2003).

(平成 19 年 11 月 17 日受付)
(平成 20 年 4 月 8 日採録)

新森　昭宏（正会員）

1983 年京都大学理学部卒業．1990 年コロラド大学大学院コンピュータサイエンス科修士課程修了，2005 年東京工業大学総合理工学研究科博士課程修了．1983 年（株）インテック入社．現在，（株）インテックシステム研究所 ICT 研究部長．博士（工学）．技術士（情報工学部門）．



大屋由香里

2001 年富山大学工学部知能情報工学科卒業，同年（株）インテック入社．現在，（株）インテックシステム研究所 ICT 研究部員．

谷川　英和（正会員）

1986 年神戸大学工学部システム工学科卒業．同年松下電器産業（株）入社．情報関連技術の研究開発に従事．1999 年弁理士登録．2002 年 IRD 国際特許事務所を開設（所長・弁理士），現在に至る．2003〜2007 年京都大学 COE 研究員．2007 年より京都大学非常勤講師．博士（情報学）．