修 士 論 文

号の列挙や参照表現をもつ法令文の論理式への変換

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

木村 祐介

2008 年 3 月

修 士 論 文

号の列挙や参照表現をもつ法令文の論理式への変換

指導教官　島津 明 教授

審査委員主査　島津 明 教授
審査委員　白井 清昭 准教授
審査委員　東条 敏 教授

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

0610030 木村 祐介

提出年月: 2008 年 2 月

概 要

21 世紀 COE プログラム「検証進化可能電子社会」において，法令工学という学問が提案された．法令工学とは，法令文書から，情報システムとして電子化された社会の形式仕様を獲得したり，法令の制定や改定においては，論理的矛盾や関連法令との不整合などが起きないための支援を，情報科学の手法により行う学問である．

法令工学の研究として，法令文書を論理式へ変換する手法の研究が進められている．これは，法令文の要件・効果構造を解析し，その論理表現を求めるものである．しかし，先行研究においては１文に１つの要件・効果構造を持つ法令文のみを扱っている一方，複雑な法令文では，号の列挙や参照表現により複数の要件・効果構造が表現されている．

そこで本研究では，号の列挙や参照表現により複数の要件・効果構造が表現されている場合に対して，文型の変換や文脈処理を行うことで，論理式に変換する方法を示す．

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　研究の背景と目的

現在，我々の社会の様々な分野で電子化が進んでいる．そのため，社会の構造や機能を明示的に記述し，規定している法令や法規は，このような電子社会を支える情報処理システムの仕様とみなすことができる．したがって，法令や法規の論理的な整合性を検証することで，情報処理システムの妥当性を検証することができる．

そのため，21 世紀 COE プログラム「検証進化可能電子社会」において，法令工学という学問が提案された [1, 2]．法令工学とは，法令文書から情報システムとして電子化された社会の形式仕様を獲得したり，法令文書がその制定目的にそって適切に作られ，論理的矛盾や文書的問題がなく，関連法令との整合性がとられていることを検査・検証し，法令の制定や改定においては，論理的矛盾や関連法令との不整合などが起きないための支援を，情報科学の手法により行う学問である．

近年では，法律文に関する推論を行うシステムとして，法律エキスパートシステムが開発されている [3]．しかし，このいったシステムは，扱う文を人手で論理式に変換する必要があった．

そこで，法令工学の研究として，法令文書を論理式へ変換する手法が研究され，システムが開発されている．

江尻 [4] は，田中らが提案した法令文の基本構造である要件・効果構造 [8] に基づき，法令文の構造を解析して全体的な論理構造を決定する手法を提案した．田中らは，法令文は「主題部，条件部，対象部，内容部，規定部」からなるが，実際の法令文においてはこれらの要素の一部が欠ける現象（φ化）が起きることがあるとしている．そこで，こうした場合も含めた変換のルールを提案し，実装したシステムによって変換された論理式と，人手によって変換された論理式と比較することで，その妥当性を検証した．

また，北田 [5] は，江尻のシステムによって解析された各部位の述語動詞に対して格解析を行い，原子式に変換する手法を提案した．述語動詞と名詞との意味的関係を表す深層

格を決定するための格フレーム辞書を構築した上で，格解析を行う際には，この格フレーム辞書をもとに，法令文の特徴も考慮しながら，解析する法令文中の述語動詞に対してとりうる深層格としてのスコア付けを行い，深層格を決定し，それをもとに原子文を生成した．

さらに信岡 [6, 7] は，所得税法や国民年金法の分析を行い，名詞句や各構造の扱える範囲を広げた．格解析については，全 13 都府県の条例から格フレーム辞書を作成し，格要素と格スロットの称号の度合いをスコア付けした上で各格スロットのスコアを足したものを最終的なスコアとして深層格を決定した．また，名詞句「A の B」については A と B の品詞にもとづいて処理法を選択し，加えて表現の言い換えや格構造の再帰的解析などを行うことによって，より多様な名詞句や格構造を扱えるようシステムを改良した．

しかし，これらの先行研究においては，１文に１つの要件・効果構造を持つ法令文のみを扱っている一方，複雑な法令文では，箇条書きや参照表現により，複数の要件・効果構造が表現されている．

そこで本研究では，以下のような表現に対して，正しく論理式に変換する処理法を明らかにすることを目的とする．

- 法律効果部に対して複数の法律要件部が箇条書きされている表現

  田中ら [8] によれば，法令文は，権利・義務関係を規定する法律効果部と，それが成立する条件を示す法律要件部に分けられる．従来のシステムは要件部と効果部が各々一つの文を対象としていたが，法令文書中には多様な表現が出現し，その中には複数の要件部と効果部が複雑な構造をなし，人間には読解が困難な文もある．そこで，これらの複数文を解析し，要件部と効果部を一対一に対応づけた文の集合に変換する．このように複雑で多様な文の構造を明確にすることは，コンピュータでの処理に限らず，人間が理解しやすい文の生成にも役立つ．

- ある条文の中で，他の条文を参照している表現

  法令のある条文中で，他の条文に書かれた規定などを参照している場合，その規定の具体的な内容を知らなければ適切な論理式に変換できないことがある．このため，参照先の条文中から，必要な規定等を抽出して利用する．

1.2　本論文の構成

本論文では，以降，第 2 章で関連研究を示し，第 3 章で法令文中の箇条書き・参照表現などの分析を行う．第 4 章で処理のモデルを示し，第 5 章でそれらに基づき提案する法令文の変換手法，第 6 章で提案手法による実験について示す．最後に第 7 章でまとめを行う．

第2章　関連研究

2.1　法令文の構造

田中ら [8] は，**要件・効果構造**を法令文の基本構造とみなし，法令文の分析手法を示した．

法令文は,「…は…するものとする。」といった，その条文が規定したい権利・義務関係の内容である**法律効果部**と，それを成立させるための「…の場合」「…とき」といった条件を示す**法律要件部**を含む．法律要件部はさらに要件部の主題を示す**主題部**とその条件を示す**条件部**に分けられ，法律効果部については条文の対象を示す**対象部**，条文の内容を示す**内容部**，その内容の規定を示す**規定部**に分けることができ，図 2.1 のような構造をとるとした．

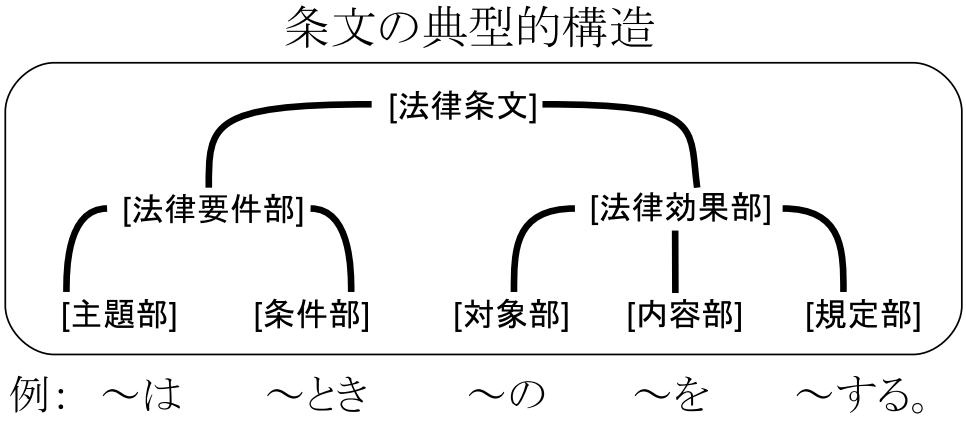


図 2.1: 要件・効果構造（[8] より）

2.2 論理式への変換

Mulkar ら [9] は，自然言語のテキストからの学習を目的として，推論エンジンが理解可能，かつ質問応答にも利用可能な表現の生成について論じ，化学の教科書や Wikipedia 等のウェブページから収集した科学に関するテキストを対象に，文法的に複雑な文から簡潔かつ意味的に正しい論理式を得るための手法を提案した．

複雑なテキストに対して解析を行った場合，パージングを誤り，必要な述語が不足することがある．そこで，アブダクション（問題点を仮定して推論を行う）によって情報を補い，結果として生じる曖昧性に対してはコストを計算して選択することで，結果を改善することができた．

また，Ferrández ら [10] は，論理式の処理により，文 (text) と含意仮説文 (hypothesis) の組に対して含意関係 (entailment) の有無を判定する手法を提案した．

辞書データベース WordNet を用い，WordNet の持つ単語間の関連情報によって，文と含意仮説文に出現する単語の意味的な類似度を求める．そして，文と含意仮説文の各々について得た論理式をもとに，含意関係の有無を判定する．

その際に，WordNet の情報のうち，含意に関係すると考えられる単語間の関連情報である，hyponym（下位語）・synonym（同義語）・entailment（含意語）の 3 種類のみを用いた解析と，その他に meronym（部分語），holonym（統合語）などの情報も加えた解析の結果を比較したところ，3 種類の関連情報のみを用いた解析のほうが良い結果を得られた．

法令文を論理式に変換する方法としては，江尻，北田，信岡ら [4, 5, 6, 7] の研究がある．

江尻 [4] は，田中らの**要件・効果構造** [8] に基づき，法令文は，

- “ 条件，付帯状況，結論 ”の形で記述されることが多い
- 語順が一般の自然言語と比較すると，かなり定まっている
- 固有の言い回しが多いことから形態素による解析が効果的である

といった特徴を持つとして，法令文の構造を解析して全体的な論理構造を決定する手法を提案した．法令文の論理構造は，要件・効果構造における分類をもとに，

$$\boxed{[\text{主題部}] \wedge [\text{条件部}] \Rightarrow [\text{対象部}] \wedge [\text{内容部}] \wedge [\text{規定部}]}$$

とすることができる．しかし，実際の法令文においては，条件部と対象部が欠けている等，全ての要素を含まない場合もある [8]．そこで，こうした場合も含めた変換のルールを提案し，実装したシステムによって変換された論理式と，人手によって変換された論理式と比較することで，その妥当性を検証した．

北田 [5] は，江尻のシステムによって解析された各部位の述語動詞に対して格解析を行い，原子式に変換する手法を提案した．原子式への変換の際に，述語動詞と名詞との意味的関係を表す深層格を決定する必要がある．そのため，千代田区生活環境条例・富山県条例第 54 号「情報通信技術の利用に関する条例」から取り出した述語動詞を分析，129 種類の述語動詞に対して，とりうる深層格と，深層格としてとる名詞とその頻度，それらに付随する表層格の情報を含む格フレーム辞書を構築した．

さらに，格解析を行う際には，構築した辞書をもとに，法令文の特徴も考慮しながら，解析する法令文中の述語動詞に対してとりうる深層格としてのスコア付けを行い，深層格を決定し，それをもとに原子文を生成している．

さらに，信岡 [6, 7] は，先行研究に加え，所得税法や国民年金法の分析を行い，名詞句や各構造の扱える範囲を広げた．

格解析については，全 13 都府県の条例から格フレーム辞書を作成し，格要素と格スロットの称号の度合いをスコア付けした上で各格スロットのスコアを足したものを最終的なスコアとして深層格を決定した．

また，名詞句「A の B」については A と B の品詞にもとづいて処理法を選択し，加えて表現の言い換えや格構造の再帰的解析などを行うことによって，より多様な名詞句や格構造を扱えるようシステムを改良した．

これらの研究にもとづいて実装された論理式変換システムの概要を図 2.2 に示す．

しかし，このシステムを用いて，図 2.3 のような，他の文を参照している条文を処理した場合，参照表現をそのまま論理式に変換するだけで，参照先の文の情報を用いることができない．このため，生成される論理式には，必要な情報が足りない上，意味のない述語が含まれてしまう．例では，四角で囲んだ「各号 (x4)」がそのような意味のない述語である．

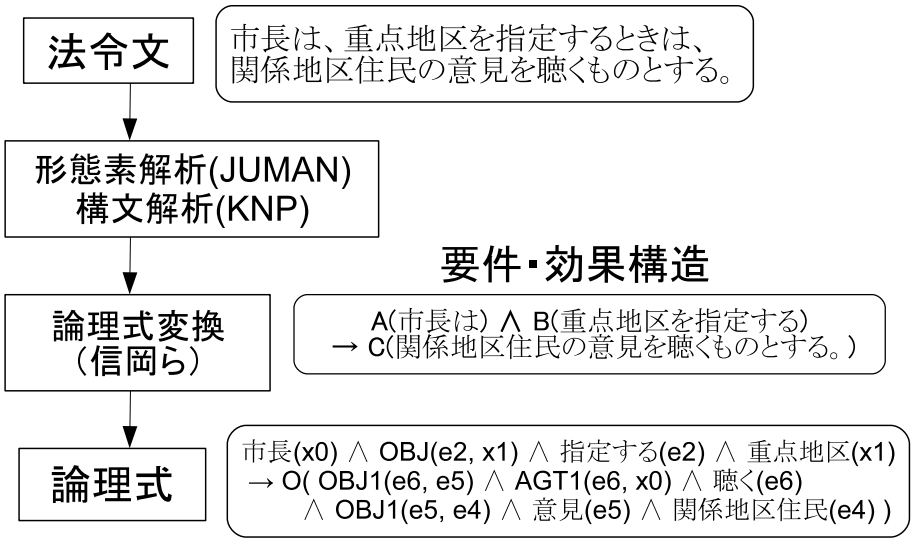


図 2.2: 論理式変換システムの概要

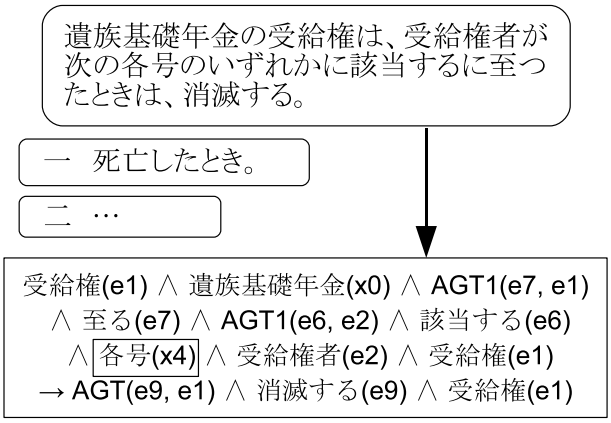


図 2.3: 既存システムで参照表現を含む文を処理

第3章　法令文の分析

3.1　法令の構造

法令は本則と附則から成り立っている．このうち附則には法令の施行期日，施行になった場合に現実の状態とうまく適合するようにするための経過的な事項などが規定されており，法令の実質的内容は本則に書かれている．

本則は，原則として異なる事項ごとに条に区分されており，同じ条の中でさらに細分する必要があるときは項に分け，更に項の中で号に分けることもある．号の中で更に細分化する必要がある場合には順次「イ，ロ，ハ…」さらに「(1),(2),(3)」を用いる．

表記については，各条の最初には「第○条」と記述し，項に分かれている場合も第一項には項番号を付けず，第二項以下は算用数字で「２」「３」等と付す．項の中で号を設ける場合には，一字下げて漢数字の「一」「二」等を用いて示し，さらに細分化する場合，もう一字下げ「イ」「ロ」等を表記する [11].

また，ある条とある条の間に，新たに条を追加したい場合，条番号をずらして付け替えるのではなく,「第○条の二，第○条の三」などの条番号をつける．これは第○条に付随するものではなく，単独の条として扱われる．つまり，各条は「第 (X) 条，第 (X) 条の二，第 (X) 条の三，第 (X+1) 条」といった順番で並べられ，ここで第 (X) 条における「次条」とは「第 (X) 条の二」を指し，第 (X+1) 条における「前条」とは「第 (X) 条の三」を指すことになる．このような形式をとっている部分にさらに条を追加する場合，同様にして「第○条の○の二」などとなる．

これらをまとめた法令の構造を図 3.1 に示す．右側に法令の本則部分の例を示し，それに対応する条文番号を左に示す．このような条文に対応する番号は，他の条文から参照される際などに用いられている．

対応する条文番号	法令（本則部分）の例
第一条	第一条　……。
第二条（第一項）	第二条　……。
第二条第二項	２　……。
第二条第二項第一号	一　……。
第二条第二項第二号	二　……。
第二条第二項第二号イ	イ　……。
第二条第二項第二号ロ	ロ　……。
第二条の二	第二条の二　……。
第二条の二の二	第二条の二の二　……。
第二条の二の三（第一項）	第二条の二の三　……。
第二条の二の三第二項	２　……。
第二条の三	第二条の三　……。
第三条	第三条　……。

図 3.1: 法令の構造（本則部分）

3.2　参照表現の分析

法令文は，一つの法令の中でも，他の法令との間でも，様々な形で参照しあっている．例えば3.2節に述べた箇条書き表現も，各号を参照しているといえるが，これは単一の項において、表記を簡略化するため，号などに分割した上で参照しているものといえる．本論文における**参照表現**とは，別に定義された他の項などを参照しているものとして，以降で分析を行う．

まず，実際の法令文における参照表現の出現をいくつか例示する．

> 付加年金の支給は、その受給権者が**第二十八条第一項に規定する**支給繰下げの申出を行ったときは、**第十八条第一項の規定**にかかわらず、**当該**申出のあった日の属する月の翌月から始めるものとする。
> （国民年金法第四十六条より）

という条文においては，太字にした「**第二十八条第一項に規定する**」や「**第十八条第一項の規定**」「**当該**」が参照表現にあたる．こうした条文の規定を正しく理解するには，第二十八条第一項などの条文を参照する必要がある．また,「**当該**」については，他の条文ではなく，同じ条文中の前に出てきた部分を参照すれば良いと考えられる．

そこで，第二十八条第一項を見ると，

> 第二十八条　老齢基礎年金の受給権を有する者であって六十六歳に達する前に当該老齢基礎年金を請求していなかったものは、社会保険庁長官に当該老齢基礎年金の支給繰下げの申出をすることができる。ただし、…（後略）

とあるので,「**第二十八条第一項に規定する**支給繰下げ」とは，ここに書かれた老齢基礎年金の支給繰下げであるとわかる．

参照表現について，他の出現例を示す.

> **前項**の年金の額は、二百円に当該解散した基金に係る加入員期間の月数を乗じて得た額とし、**同項**の一時金の額は、八千五百円とする。
> （国民年金法第百三十七条の十九第三項より）

上の例において,「前項」とは，一つ前の項，この場合は第百三十七条の十九第二項であり，その条文に記述された年金の額を指している.「同項」とは，直近に指示した条文の意味であり，この場合は同じく第百三十七条の十九第二項に記述された一時金の額を指している．

> **第百三十一条の二及び第百三十二条の規定**は、連合会の積立金の積立て及びその運用、業務上の余裕金の運用並びに事業年度その他その財務について準用する。この場合において、**同条第三項**中「前条及び前二項」とあるのは、「第百三十七条の二十一第三項において準用する前条及び前二項」と読み替えるものとする。
> （国民年金法第百三十七条の二十一第三項より）

この例では，準用という用語が使われている．これは，国民年金基金に対する国民年金基金連合会など，仕組みの良く似たものを複数規定する場合に，既存の条文を再利用して，一部の用語等を読み替えるものである．

つまり，第百三十一条の二と第百三十二条での国民年金基金に関する規定を国民年金基金連合会についても適用するが，同条（＝直近に指示した条＝第百三十二条）第三項の記述のうち,「前条及び前二項」を「第百三十七条の二十一第三項において準用する前条及び前二項」と読み替えた上で適用するという意味である．

そこで，第百三十二条第三項を見ると，

> 基金は、事業年度その他その財務に関しては、**前条及び前二項**の規定によるほか、政令の定めるところによらなければならない。

とあり，これが国民年金基金連合会の場合，

> 基金は、事業年度その他その財務に関しては、**第百三十七条の二十一第三項において準用する前条及び前二項**の規定によるほか、政令の定めるところによらなければならない。

として適用されるということになる．

3.2.1 参照表現の出現形式

このような参照表現について国民年金法を対象に分析を行った結果，表 3.1 のような出現形式があった．また，ここでは件数のみを示したが，各々の出現する条文番号は付録 A に示した．

ここで X は，参照先の条文を指すポインタといえる．例えば，

- 第 a 条第 b 項第 c 号
- 同項 （直近に指示した項）
- 前項 （現在の条文を基点とした相対的指示）
- 前条第 b 項 （相対指示と絶対指示）

表 3.1: 参照表現の出現形式（国民年金法）

参照表現の出現形式	出現件数
X に規定する Y(名詞)	86 件
X の規定による Y(名詞)	71 件
X の規定により Y(動詞)	109 件
X の規定によって Y(動詞)	14 件
X の規定にかかわらず Y(動詞)	26 件
X の規定に該当する Y(名詞)	1 件
X の規定に基づく Y(名詞)	1 件
X の規定に基づき Y(動詞)	2 件
X の規定の例によって Y(動詞)	1 件
X の規定に違反した Y(名詞)	3 件
X の規定に違反して、	16 件
X の規定 (を/は) 適用 (する/しない)	7 件
X の規定の適用が	4 件
X の規定の適用については	9 件
X の規定の適用を受ける	6 件
X の規定の適用上	3 件
X (のいずれか) に該当する	26 件
X (のいずれにも) 該当しない	3 件
X に掲げる	16 件
X に定める	17 件
X の場合において	13 件
X の場合に準用する	7 件
X において準用する	22 件
X を準用する	1 件
X において「…」という	3 件
X の規定は、…準用する。	39 件
X の規定及び…は、…準用する	1 件
(X 中「…」とあり、並びに) X 中「…」とあるのは	25 件
X において同じ	6 件
X と同様とする	1 件
X を除く	4 件
X の （その他名詞）	90 件
合計	633 件

- 前 n 項 （現在の条文の直前の n 個の項）
- 前項ただし書 （「ただし、」以下の部分）
- この法律 （この文を含む法律全体）

などが国民年金法の条文中に出現する．

さらに，これらの組合せで，

- X1 若しくは X2
- X1 並びに X2
- X1 及び X2
- X1 又は X2
- X1 から X2 まで
- X1 から X2 ， X3 及び X4
- 第 a 条第 b 項各号に掲げる
- 第 a 条第 b 項各号のいずれか

といった指示も出現する．

3.2.2 参照表現「X に規定する Y」

さらに，このような参照表現において，参照先の条文 X における語 Y の出現を調査した．ここでは，国民年金法中から「X に規定する Y」の形式で現れるものを対象とし，X が他の法令を指している 9 件を除いた 82 件を分類した結果を表 3.2 に示す．

表 3.2: 条文 X における語 Y の出現

Y という語が X の指す条文中に…	
一度だけ出現する	49 件
複数回出現する	24 件
出現しない	9 件

大半の例において，参照先の条文 X 中に Y という語が出現していることがわかる．そこで，参照元である「X に規定する Y」の語 Y と，参照先である条文 X 中の語 Y は同一のものを指しており，さらに，参照先の条文 X において，語 Y について必要な説明が補足されているといえる．この考えにもとづき，条文 X から語 Y についての情報を抽出する手法を 4.2 節で検討する．

3.3　箇条書きの分析

3.3.1　箇条書きの定義

法令の文中には，図 3.2 のように，条文中に「次の各号のいずれか…」といった表記が現れる．そのような場合，直後の各号において箇条書きの形式でそれぞれ条件が示されており，それらのうちのいずれかを満たしたとき，条文に示された法令効果部の記述が成立することになる．

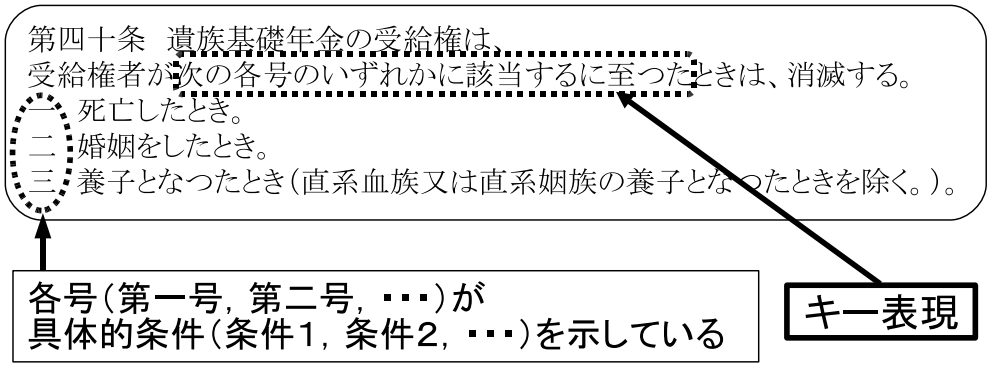


図 3.2: 箇条書き

その他にも，号の形式で箇条書きされているものとしては，

- 列挙型　…　複数のものをまとめ，定義などを行っているもの．例としては，

基金は、規約をもって次に掲げる事項を定めなければならない。 一　名称 二　事務所の所在地 三　地区 四　代議員及び代議員会に関する事項 五　役員に関する事項 六　加入員に関する事項 七　年金及び一時金に関する事項 八　掛金に関する事項 九　資産の管理その他財務に関する事項 十　解散及び清算に関する事項 十一　業務の委託に関する事項 十二　公告に関する事項 十三　その他組織及び業務に関する重要事項 （国民年金法第百二十条より）

- 計算型　…　複雑な計算を要する場合に，式を分割して表記しているもの．例としては，

> 改定率については、毎年度、第一号に掲げる率に第二号及び第三号に掲げる率を乗じて得た率を基準として改定し、当該年度の四月以降の年金たる給付について適用する。
> 一　当該年度の初日の属する年の前々年の物価指数に対する当該年度の初日の属する年の前年の物価指数の比率
> 二　イに掲げる率をロに掲げる率で除して得た率の三乗根となる率
> イ　…　に対する　…　の比率
> ロ　…　に対する　…　の比率
> 三　イに掲げる率をロに掲げる率で除して得た率
> イ　…　して得た率
> ロ　…　して得た率
> （国民年金法第二十七条の二第二項より抜粋）

といったものがある．

このうち条件文の箇条書き表現と列挙型について，次節でさらに分析する．

3.3.2　キー表現と条件文

国民年金法全 148 条中において，条件文の箇条書き表現は 34 件，列挙型の箇条書き表現は 10 件出現する．

また，これらの箇条書きが出現する条文において必ず出現する表現を調べた結果，図 3.3 のように表現できることがわかった．図の線に沿って左から右にたどったもの，例えば「次の各号に該当するに至つた」「次の各号に該当した」といった表現が出現する．この図の破線内を本論文では**キー表現**と呼ぶ．

ここで，**キー表現**の直後には「とき」「場合」などの単語が出現する．これらの単語と，条件文である各号の文末は，対応関係がかなり限定されることがわかった．その対応関係を，条件文の箇条書き表現について表 3.3 に，列挙型について表 3.4 に示す．

例えば，**キー表現**の直後の単語が「場合」ならば，その後に続く各条件文は「とき」で終わる．この，条件文から文末表現「とき」を除いた部分をこの箇条書き表現における**条件**とする．

また，列挙型についても便宜上，条件文の場合と同様に**条件**と呼ぶこととする．例えば，前述の列挙型の例では，**キー表現**は「次に掲げる」，表における名詞 A にあたるもの

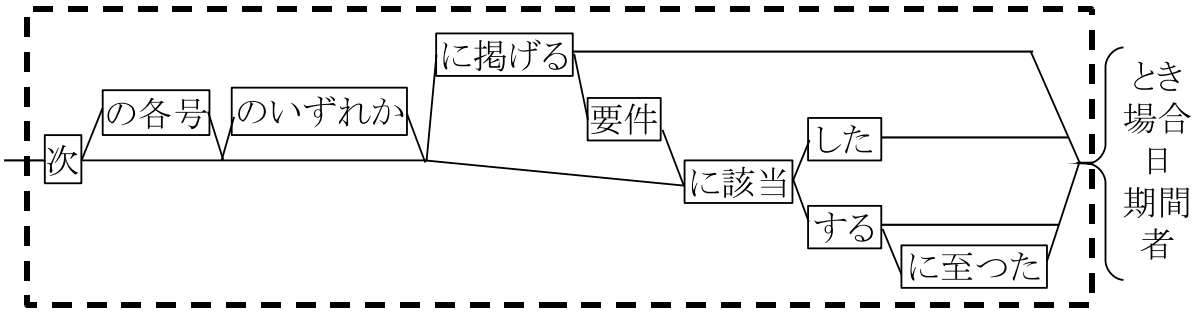


図 3.3: キー表現

表 3.3: キー表現と号の表現（条件文の箇条書き）

キー表現の出現形式	各号（条件文）の表現		国民年金法での出現
キー表現 + とき	**条件** + とき	9 件	第三十三条の二第三項, 第三十五条, 第三十六条の二, 第三十九条第三項, 第四十条, 第四十条第三項, 第五十二条の二第二項, 第八十九条, 第九十二条の六
キー表現 + 場合	**条件** + とき	9 件	第二十七条の三第二項, 第二十七条の四第二項, 第二十七条の語第二項, 第三十七条, 第七十二条, 第百三十七条第二項, 第百四十五条, 第百四十六条, 第百四十七条
キー表現 + 日	**条件** + とき	3 件	第八条, 第九条, 第百二十七条第三項
キー表現 + 期間	**条件** + もの	1 件	第四十五条
キー表現 + (人・集団を表す名詞)	**条件** + とき	5 件	第九十条, 第九十条の二, 第十条の二第二項, 第九十条の二第三項, 第九十条の三
キー表現 + 要件	**条件** + こと	1 件	第三十七条の二
キー表現 + 者	**条件** + こと	6 件	第七条, 第三十条, 第九十二条の三, 第百十二条, 第百十三条の二, 第百十四条
	条件 + もの		
	条件 + 者		
	条件 + (人・集団を表す名詞)		

表 3.4: キー表現と号の表現（列挙型の箇条書き）

キー表現の出現形式	列挙要素		国民年金法での出現
キー表現 + 名詞 **A** + を	**条件**	7 件	第五条, 第八十五条, 第百二十条, 第百二十三条, 第百三十七条の八, 第百三十七条の十一, 第百三十七条の十五第二条
キー表現 + 名詞 **A** + により	**条件**	3 件	第百三十五条, 第百三十七条の二十二

が「事項」，各**条件**が「名称」「事務所の所在地」などとなる．

これらの**キー表現**と**条件**（列挙型の場合は名詞 A も）を用い，箇条書きを取り除く処理法を 4.3 節で検討する．

3.4 括弧書きの分析

法令文において，括弧「()」は様々な用途に用いられている．前項までに分析した箇条書きや参照表現と同様，国民年金法における用途を分類した．

国民年金法全 148 条において，括弧書き表記は 391 件存在した．ただし，以下の分析では，二重括弧はまず内側の括弧についてカウントし，さらに内側の括弧を削除した状態で外側の括弧をカウントした．また，三重以上の括弧は国民年金法には存在しなかった．

以下，これら全 391 件の括弧の用途を分類しカウントした結果を示す．詳細な分類においては，正規表現を用いて表記している．

- 定義　…　47 件

  例：被用者年金各法の被保険者、組合員又は加入者（以下「第二号被保険者」という。）

  以下のいずれかの形式にあてはまる文を指す.

以下「.+」(又は「.+」)*という。	34 件
以下単に「.+」(又は「.+」)*という。	4 件
以下この(条|項|号)(及び<条文指示>)*において「.+」(又は「.+」)*という。	7 件
<条文指示>(及び<条文指示>)*において「.+」(又は「.+」)*という。	2 件

  ここで <条文指示> とは，3.3 節に示した,「第 a 条第 b 項第 c 号」等のような，条文を指示する表現にあたる．

- 範囲限定など　…　123 件

  例：被用者年金各法による年金たる給付（当該年金給付と同一の支給事由に基づいて支給されるものを除く。以下この条において同じ。）を受けることができるときは、…

  以下のいずれかの形式にあてはまる文を指す.

.+を除く。	55 件
.+を含む。	22 件
.+に限る。	19 件
.+をいう。	12 件
.+とする。	12 件
.+を含み、.+を除く。	1 件

「。」の後，ひとつ上で示した「定義」の形式にあてはまる文，あるいは以下のいずれかの文が追記される場合もある．

以下同じ。
以下この (章|条|項|号)(及び<条文指示>)*に (おいて|ついても) 同じ。
<条文指示>(及び<条文指示>)*に (おいて|ついても) 同じ。

- 条件＋指示　…　28 件

例：

…を乗じて得た額（そのうち二人までについては、それぞれ二十二万四千七百円に改定率を乗じて得た額とし、それらの額に五十円未満の端数が生じたときは、これを切り捨て、五十円以上百円未満の端数が生じたときは、これを百円に切り上げるものとする。）を加算した額とする。

以下に述べる各要素を，次のように接続した形式にあてはまる文を指す．

<前半部>(<接続部><前半部>)*<後半部><追記部>

– <前半部> は，次のいずれかにあたる．

.+においては、
.+にあっては、
.+とき (又は.+とき)*は、
.+場合 (にあって)?は、
.+については、

– <後半部> は，次のいずれかにあたる．

.+とし、以下「.+」という。
.+とする。
.+<範囲限定など>
その他（「二分の一」「その日」「障害の額」「〜とき」など）

ここで <範囲限定など> とは，ひとつ上で示した「範囲限定など」の形式にあてはまる文を指す．

– <接続部> は，次のいずれかにあたる．

.+とし、
.+、

– <追記部> は，次のいずれかにあたる．

以下この (章|条|項|号)(及び<条文指示>)*に (おいて|ついても) 同じ。
<条文指示>(及び<条文指示>)*に (おいて|ついても) 同じ。

- 法令指示　…　22 件

例：

恩給法（大正十二年法律第四十八号。他の法律において準用する場合を含む。）に基づく年金たる給付

これは，次のような形式の文を指す．

.+年法律第.+号 (。他の法律において準用する場合を含む。)?

- 無視できる見出し括弧　…　173 件

法令文では，各条の前に，その条についての簡単な見出しを置くことが多い．その場合，見出しは括弧で囲むことになっている．これらは，人間にとっての読みやすさの上では重要な要素ではあるが，本研究の目的である論理式への変換においては無視しても構わないと考えられる．

例：

（国民年金制度の目的） 第一条　国民年金制度は、日本国憲法第二十五条第二項に規定する理念に基き…

また，複雑な構造をもっている括弧書きとしては，第九十三条の三において，

…当該年度における当該被用者年金保険者に係る被保険者（厚生年金保険の管掌者たる政府にあっては、厚生年金保険の被保険者である第二号被保険者及びその被扶養配偶者である第三号被保険者とし、年金保険者たる共済組合等にあっては、当該年金保険者たる共済組合等に係る被保険者（国家公務員共済組合連合会及び地方公務員共済組合連合会にあっては、当該連合会を組織する共済組合の組合員である第二号被保険者及びその被扶養配偶者である第三号被保険者とし、日本私立学校振興・共済事業団にあっては、私学教職員共済制度の加入者である第二号被保険者及びその被扶養配偶者である第三号被保険者とする。以下同じ。）とする。）の総数…

というものがあり，ここでの括弧の構造は，

〜に係る被保険者（〜にあっては、〜とし、
〜にあっては、〜（〜にあっては、〜とし、
〜にあっては、〜とする。以下同じ。）
とする。）の総数〜

となっている．

ただし，これらの括弧書きについては，本論文においては処理法の検討および実装を行っていない．

第4章　処理のモデル

本研究における基本的な考え方は，先行研究による論理式変換システムでは処理できない箇条書きや参照表現を，あらかじめ取り除いておくことで，法令文を正しく処理できるようにすることである．図 4.1 にシステムのモデル図を示す.

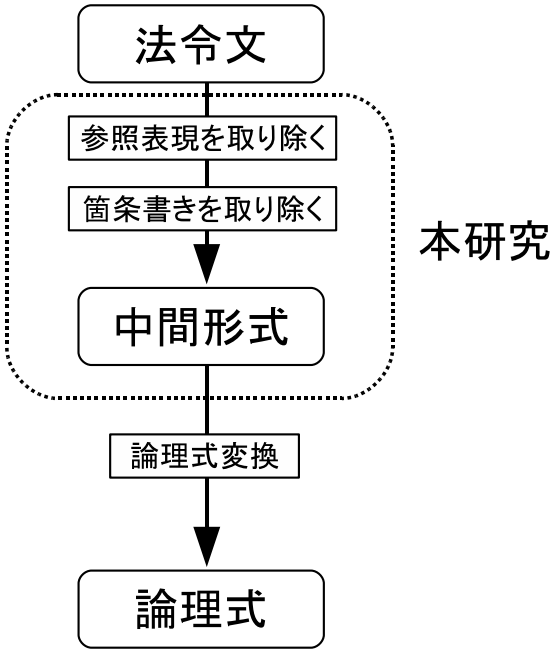


図 4.1: システムのモデル

前章において，法令文中の参照表現や箇条書きなどについての分析を行った．これをもとに，どのように取り除くかを次章で検討し，処理内容の詳細を示す.

既存システムと提案システムそれぞれの動作イメージを図 4.2 と図 4.3 に示した．出力である論理式のうち，変化する部分を破線で囲んでいる．既存システムでは処理できなかった号の記述を活かすことで，意味的に正しい論理式を得ることができるようになる.

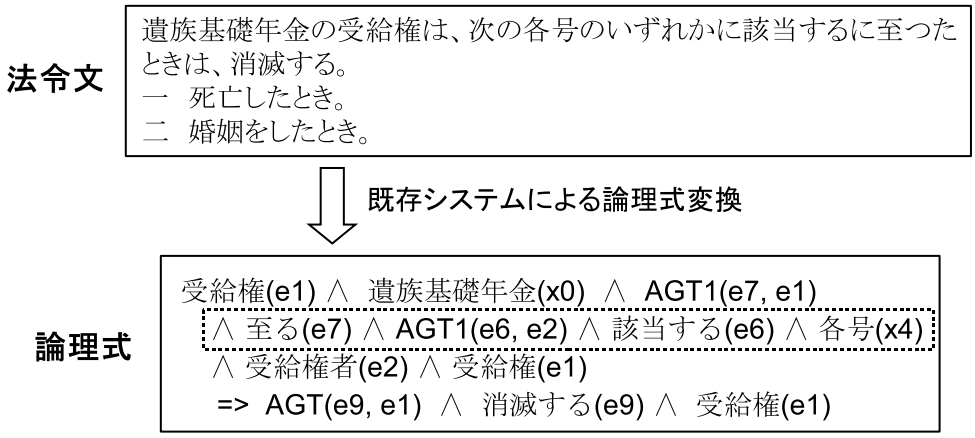


図 4.2: 既存システムの動作

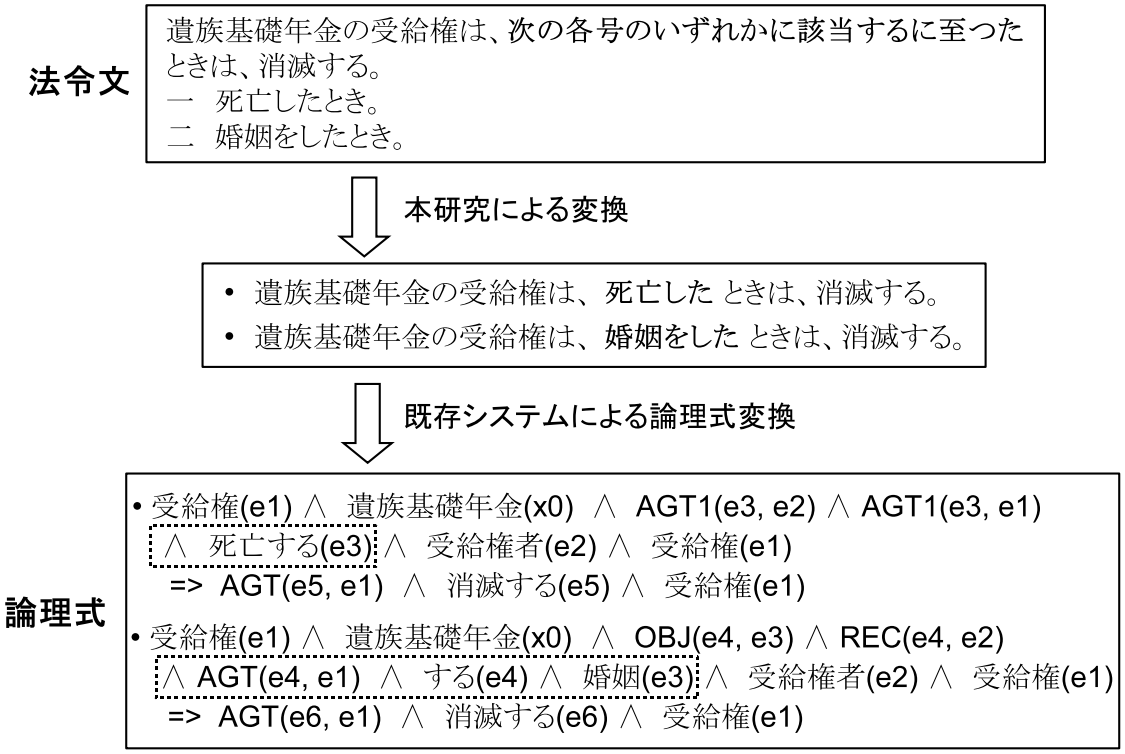


図 4.3: 提案システムの動作

第5章　論理式への変換処理

5.1　処理の流れ

本研究では，先行研究による論理式変換システム [4, 5, 6, 7] に組み込むという形で，箇条書きおよび参照表現に対する処理法を検討する．提案手法を組み込んだ後の全体の処理の流れを図 5.1 に示す.

まず，法令文の入力に対して，参照表現を探し，参照先から必要な情報を抽出して補う．それから，箇条書きがあれば複数の文に分解して箇条書き表現を取り除く.

参照表現の処理においては,「前条」「次条」など条文番号の相対的な指示表現を扱う．しかし，箇条書き表現の処理の際に，条文の分割などが行われ，条文番号などの情報が一部失われてしまうため，参照表現の処理を先に行う必要がある.

以上の処理によって変換された法令文に対して，一文ずつ，先行研究による論理式変換処理を行い，論理式を出力する.

5.2　参照表現に対する処理

以降，この節では参照表現「X に規定する Y」に対する処理について述べる.「X の規定により Y する」など，他の参照表現については，本論文では処理法の検討を行っていない.

5.2.1　参照表現の発見

まず，3.3.1 節で述べたように，条文を指示している表現は,「第 a 条第 b 項第 c 号」など，形式が決まっているため，これを正規表現を用いた文字列マッチングによって発見する．その後「に規定する」という語が続くので，これは「X に規定する Y」という形式の参照表現であると判断する.

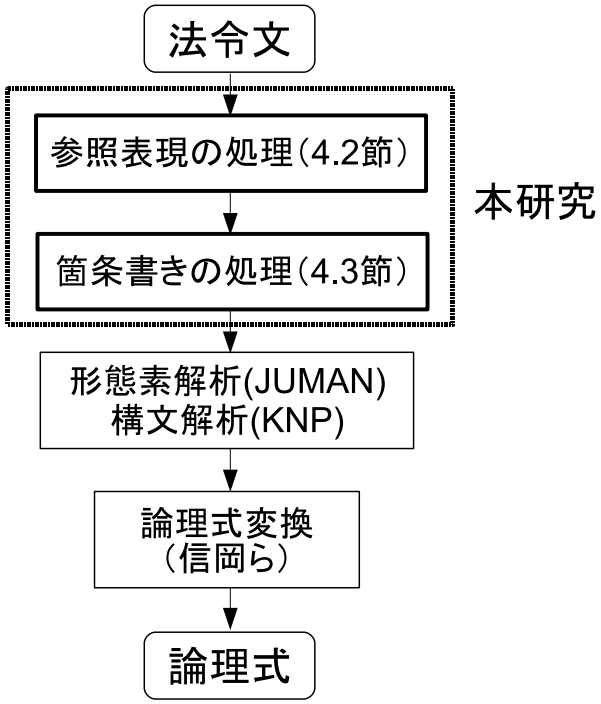


図 5.1: 処理の流れ

5.2.2 条文を指示する表現に対する処理

条文 X が「第 a 条第 b 項第 c 号」等と明示している場合は問題ないが,「前条第二項」等といった指示の場合，それが具体的に第何条の第何号であるかを知る必要がある．

そこで，参照表現に対する処理の際には，現在処理している条文の番号「第 x 条第 y 項第 z 号」をあらかじめ取得しておき，それらの変数を増減させて具体的な条文番号を得る方法が考えられる．例えば「前条」ならば「第（x-1）条第 1 項」である．しかし，「第 x 条」の次が「第 x 条の二」であったり，前が「第（x-1）条の八」であったりした場合，このような単純な方法では取得できない．

そこで，あらかじめ法令文書全体を走査して，通し番号を振っておく．例えば国民年金法の第百条は，通し番号では 142 番目の条となる．この通し番号を用いて,「前条」「次条」等の処理を行う．

また，このようにして取得した条文番号は保持しておき，もし次に「同条」等が現れた際に用いる．

5.2.3 参照先条文の取得

次に，条文 X からその内容を取得する必要がある．

ここで，法令文書に XML タグを自動付加するプログラムを作成し，これによって生成された XML 形式の法令文書を用いた．XML 形式の法令文書は，以下のような形式をとる．

```
<?xml version="1.0" encoding="UTF-8"?>

<document>
<article num="1" serialnum="1"><title>第一条</title>
<paragraph num="1" serialnum="1"><title>第一項</title>
<text>国民年金制度は、日本国憲法第二十五条第二項に規定する理念に基き、老
齢、障害又は死亡によって国民生活の安定がそこなわれることを国民の共同連帯に
よって防止し、もって健全な国民生活の維持及び向上に寄与することを目的とする。
</text>
</paragraph>
</article>
<article num="2" serialnum="2"><title>第二条</title>
<paragraph num="1" serialnum="2"><title>第一項</title>
<text>国民年金は、前条の目的を達成するため、国民の老齢、障害又は死亡に関し
て必要な給付を行うものとする。</text>
</paragraph>
</article>
```

法令文書全体を囲むタグが <document> で，同様に，条に対応するタグが <article>，項は <paragraph>，<item> となっている．

<article> 以下のタグが持つ属性値 num は，条などの番号を示す，serialnum は，第一条などからの通し番号を示す．項や号においては，例えば上記の第二条第一項は，第一条第一項に続く 2 つ目の項であるため，seriulnum="2" である．また，条においても，条の挿入によって「第○条の二」などといった条番号がある場合，num と serialnum の値にずれが生じてくる．また，本来の法令文には「第一項」は表記しないこととなっているが，便宜上，XML タグを付加する際に書き加えている．

また，条等の番号表記を示すタグが <title> で，文を示すのが <text> タグである．

条文 X からその内容を取得する処理においては，XML 形式の法令文書をハッシュテーブルに保持し，X の内容をキーとして必要な条文全体を取得する．

5.2.4 参照先条文からの情報抽出

例として，図 5.2 の文 A 中の参照表現に対する処理を示す．

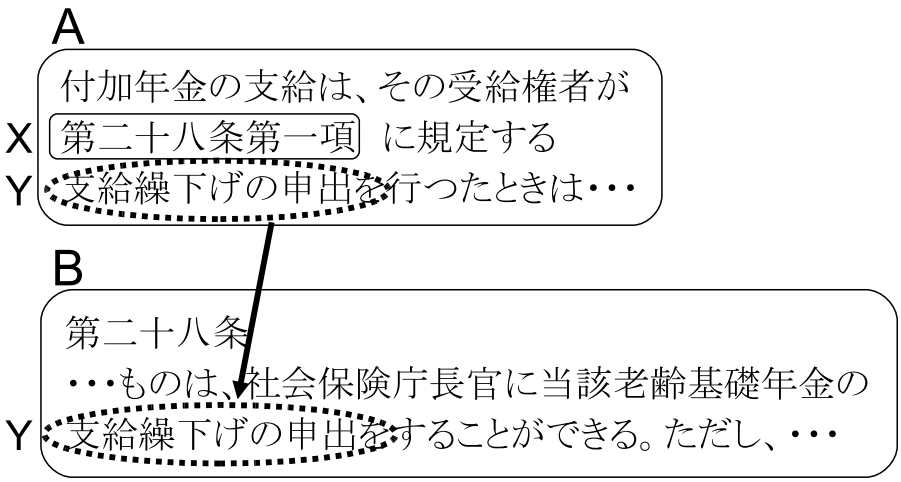


図 5.2: 参照表現の処理 1〜2

X の指す条文，つまり参照している先の文である B から，A 文中の「規定する」直後の表現と一致する語で，なるべく長いものを探す．図 5.2 の例では「支給繰下げ」ではなく「支給繰下げの申出を」となる．これらを同定する．

参照先の文 B を構文解析し，得た構文木を図 5.3 に示す．先ほど同定した語「支給繰下げの申出を」にあたる要素を修飾している要素（例では「当該老齢基礎年金の」）を，参照元の文に必要な情報として抽出する．

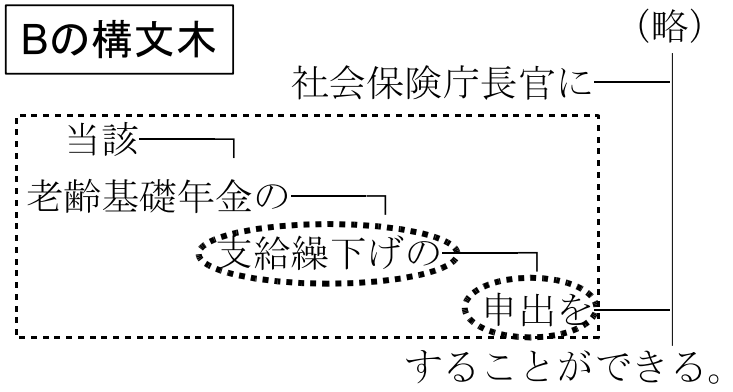


図 5.3: 参照表現の処理 3

5.2.5　置換による参照表現の削除

参照元のに対して，参照表現「X に規定する」を，前項で得た構文木の要素に置き換える．図の例では，文 A が，

> 付加年金の支給は、その受給権者が**第二十八条第一項に規定する**支給繰下げの申出を行ったときは…

から，

> 付加年金の支給は、その受給権者が**当該老齢基礎年金の**支給繰下げの申出を行ったときは…

と書き換えられ，参照表現を取り除くことができた．

5.3　箇条書きに対する処理

3.1 節において，条件文の箇条書きの際に必ず出現する**キー表現**と，各条件文のうち、「〜とき」の「とき」などの文末表現を取り除いた部分を**条件**と定義した．以降で述べる提案手法では，このような箇条書きを取り除くため，**キー表現**の出現する本文に対して，**条件**を加えることで，箇条書き表現を取り除き，単独で意味をなす文にする．

5.3.1　キー表現と条件の抽出

最初に，入力である法令文から，3.2 節に述べた**キー表現**を探す．図 3.3 に示したキー表現の形式を正規表現で表し，入力文との文字列マッチングを行う．キー表現がマッチした場合，

- 表 3.3 に示したキー表現の出現形式ならば，条件文の箇条書きとみなし，その後に条件文が号の形式で箇条書きされていると考えられる．そこで，各号を読み込み，表の対応に従って，キー表現に対応する文末表現を探す．発見した文末表現を除いた，それより前の部分を**条件**とする．
- 表 3.4 に示したキー表現の出現形式ならば，列挙型の箇条書きとみなし，その後に要素が列挙されていると考えられる．そこで，各号を読み込み，文全体を**条件**とする．

いずれの場合も，条件文は複数存在するため，複数の**条件**が抽出されることになる．

5.3.2　置換による箇条書き文の変形

本文中に発見した**キー表現**を，同じく発見した**条件**に置き換えることで，箇条書き表現を取り除く．

ただし，列挙型の場合，**キー表現**に加え，その後に続く単語（表 3.4 における名詞 A）までを，**条件**に置き換える．

ここで，号は複数存在するため，各々の条件を置き換えた文ができ，結果としては号の数と同数の文が出力されることになる．図 5.4 にここでの処理法を示す.

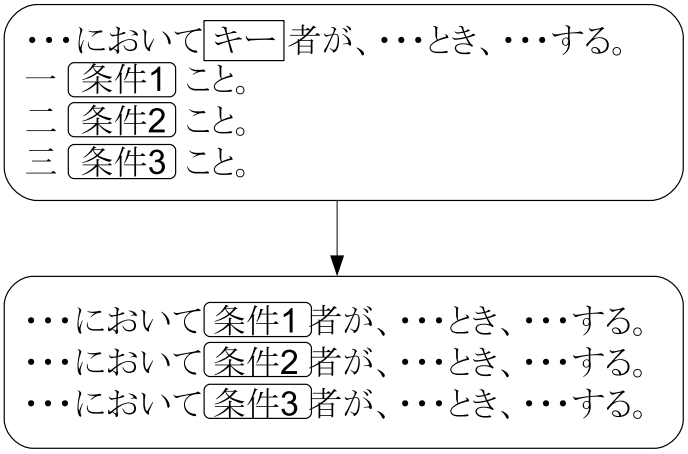


図 5.4: 箇条書きの処理法

実際の法令文で例を示すと，

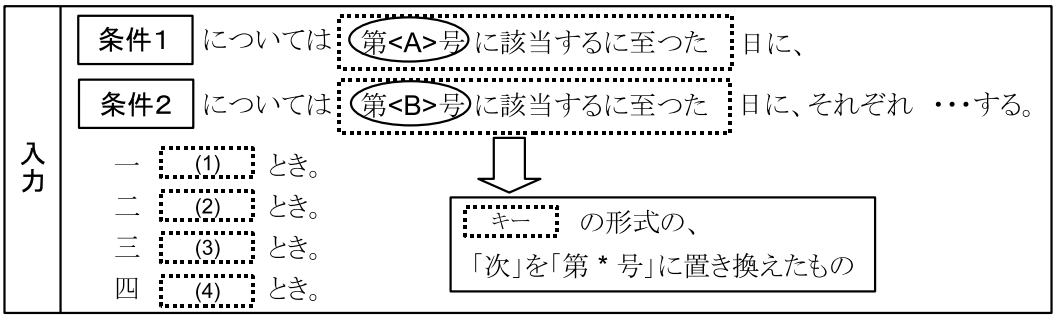
入力	遺族基礎年金の受給権は、次の各号のいずれかに該当するに至つたときは、消滅する。 一　死亡したとき。 二　婚姻をしたとき。　（国民年金法第四十条より）

ここで,「次のいずれかに該当するに至つた」が**キー表現**,「死亡した」「婚姻をした」が**条件**である．この**キー表現**を，各号から抽出された**条件**で置き換え，

出力	• 遺族基礎年金の受給権は、死亡したときは、消滅する。 • 遺族基礎年金の受給権は、婚姻をしたときは、消滅する。

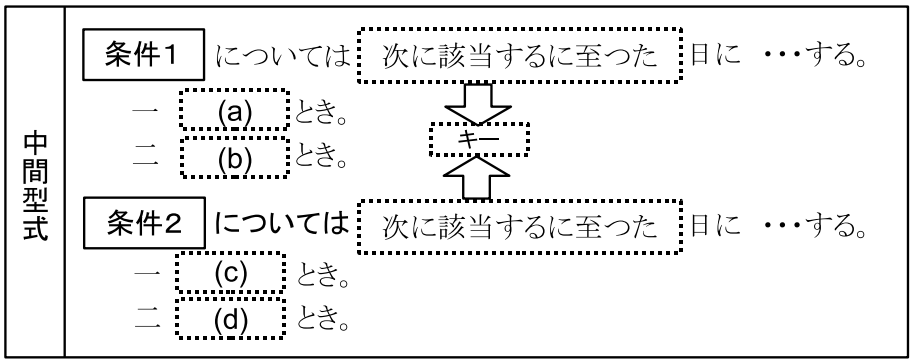
このように箇条書きがなくなり，論理式への変換が容易になる．

より複雑な箇条書きとしては，別の条件によって，用いる条件文を振り分ける文型がある．



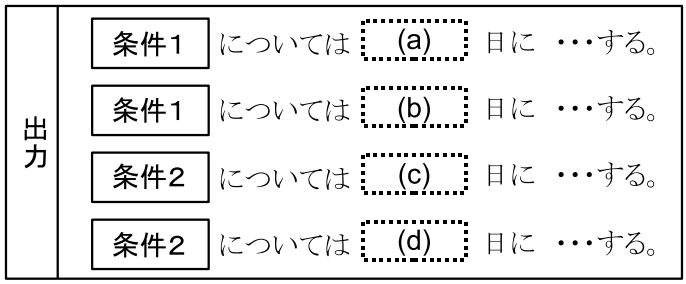
さらに,「第〇号」は複数指定されていることがあり，2つの場合「第 a 号又は第 b 号のいずれか」，3つ以上の場合「第 a 号から第 c 号までのいずれか」という表現になる．

このような文に対しては,「条件1」「条件2」等の各々に対して，指定されている号を割り当て，次のように，いちど中間形式に変換する，つまり複数の箇条書き表現に分解する．



ここで,「(a)」「(b)」は「第<A>号」に対応する号の条件文,「(c)」「(d)」は「第<B>号」に対応する号の条件文である．

こうして分解された複数の箇条書きに対して，各々，前述の箇条書き処理を行い，結果は次のようになる．



国民年金法第八条に対する処理例を示すと，

原文

第七条の規定による被保険者は、第七条第一項第二号及び第三号のいずれにも該当しない者については第一号から第三号までのいずれかに該当するに至つた日に、二十歳未満の者又は六十歳以上の者については第四号に該当するに至つた日に、その他の者については同号又は第五号のいずれかに該当するに至つた日に、それぞれ被保険者の資格を取得する。

一　二十歳に達したとき。

二　日本国内に住所を有するに至つたとき。

三　被用者年金各法に基づく老齢給付等を受けることができる者でなくなつたとき。

四　被用者年金各法の被保険者、組合員又は加入者の資格を取得したとき。

五　被扶養配偶者となつたとき。

これを複数の箇条書きに分解するが，ここで「その他の者については」という表現がある．これは,「他の条件にあてはまらない者については」と解釈し,「第七条第一項第二号及び第三号のいずれにも該当しない者」と「二十歳未満の者又は六十歳以上の者」のいずれでもない者，と変換した上で分解を行う．

中間形式

- 第七条の規定による被保険者は、第七条第一項第二号及び第三号のいずれにも該当しない者については次に該当するに至つた日に、被保険者の資格を取得する。

  一　二十歳に達したとき。

  二　日本国内に住所を有するに至つたとき。

  三　被用者年金各法に基づく老齢給付等を受けることができる者でなくなつたとき。

- 第七条の規定による被保険者は、二十歳未満の者又は六十歳以上の者については次に該当するに至つた日に、被保険者の資格を取得する。

  四　被用者年金各法の被保険者、組合員又は加入者の資格を取得したとき。

- 第七条の規定による被保険者は、**第七条第一項第二号及び第三号のいずれにも該当しない者**でも**二十歳未満の者又は六十歳以上の者**でもない者については次に該当するに至つた日に、それぞれ被保険者の資格を取得する。

  四　被用者年金各法の被保険者、組合員又は加入者の資格を取得したとき。

  五　被扶養配偶者となつたとき。

これら3つの箇条書きに対して，通常の箇条書き処理を行い，最終的な結果は次のようになる．

出力

- 第七条の規定による被保険者は、第七条第一項第二号及び第三号のいずれにも該当しない者については**二十歳に達した**日に、被保険者の資格を取得する。
- 第七条の規定による被保険者は、第七条第一項第二号及び第三号のいずれにも該当しない者については**日本国内に住所を有するに至つた**日に、被保険者の資格を取得する。
- 第七条の規定による被保険者は、第七条第一項第二号及び第三号のいずれにも該当しない者については**被用者年金各法に基づく老齢給付等を受けることができる者でなくなつた**日に、被保険者の資格を取得する。
- 第七条の規定による被保険者は、二十歳未満の者又は六十歳以上の者については**被用者年金各法の被保険者、組合員又は加入者の資格を取得した**日に、被保険者の資格を取得する。
- 第七条の規定による被保険者は、第七条第一項第二号及び第三号のいずれにも該当しない者でも二十歳未満の者又は六十歳以上の者でもない者については**被用者年金各法の被保険者、組合員又は加入者の資格を取得した**日に、それぞれ被保険者の資格を取得する。
- 第七条の規定による被保険者は、第七条第一項第二号及び第三号のいずれにも該当しない者でも二十歳未満の者又は六十歳以上の者でもない者については**被扶養配偶者となつた**日に、それぞれ被保険者の資格を取得する。

また，実際には参照表現の処理が先に行われるため，これらの文中の「第七条」や「第七条第一項第二号」の部分には具体的な情報が入っているべきである．しかし，本論文では参照表現のうち「X に規定する Y」を処理対象として検討したのみであるため，ここでの参照表現に対する処理は行えない．

第6章　実験

6.1　国民年金法に対する実験

以上の処理法にもとづいて参照表現と箇条書き表現に対する処理を実装し，先行研究のシステムに組み込んで国民年金法の全文に対してクローズドテストを行った.

6.1.1　参照表現に対する処理実験

参照表現に対して，前章の処理法にもとづいて提案システムを実装した．3.2.2 節に挙げたように，国民年金法全 148 条中の参照表現「X に規定する Y」は 91 件あり，そのうち 9 件は他の法令を参照しているもの，9 件は語 Y が条文 X 中に存在しないものである．他の 73 件のうち，語 Y についての補足が必要でないもの[1]が 20 件あったため，残りの 53 件について，提案システムによる情報の補足を行った.

結果を表 6.1 に示す.

表 6.1: 参照表現「X に規定する Y」に対する処理結果

	処理文数	
情報を抽出できた	22	41.5%
一部の情報のみ抽出できた	11	20.8%
適切な情報を抽出できなかった	20	37.7%
合計	53	100%

情報を抽出できたものの中にも，余分な情報を抽出したなどの原因で，埋め込んだ後の文が文法的に不自然であるものや，抽出した中に「その」等の指示語が含まれてしまったため，さらなる処理が必要であるものが一部存在した.

適切な情報が抽出されなかった例を以降で示す.

[1]例えば，第九十条第一項に「当該保険料に係る期間を第五条第四項に規定する保険料全額免除期間に算入することができる。」とあるが，第五条第四項ではすでに「保険料全額免除期間」という用語についての定義がなされており，論理式上では，その情報を改めて補足する必要がない.

部分的に取り出せたが，必要な情報が不足していた

例として，第三十六条の二第三項に対する処理を挙げる．

同項第一号に規定する給付の額

とあり，ここで「同項第一号」とは「第三十六条の二第一項第一号」である．その条文を取り出すと，

恩給法に基づく年金たる給付、労働者災害補償保険法の規定による年金たる給付その他の年金たる給付であって政令で定めるものを受けることができるとき。

となっている．ここで，3つの「給付」がまとめて規定されているが，最初の「給付」だけを取り出してしまい，結果は，

恩給法に基づく年金たる給付の額

となってしまい，必要な情報が不足している．

その他，取り出した情報が適切でなかった

例として，第五条第二項に対する処理を挙げる．

同項第三号に規定する被保険者としての被保険者期間を…

とあり，ここで「同項第三号」とは「第七条第一項第三号」である．その条文を取り出すと，

第二号被保険者の配偶者であって主として第二号被保険者の収入により生計を維持するもののうち二十歳以上六十歳未満のもの（以下「第三号被保険者」という。）

となっている．この条文では「第三号被保険者」という用語が定義されており，抽出すべき情報も「第三号被保険者」という用語である．しかし，最長の一致である「被保険者」で，最も前に出現するものを抽出した場合,「第二号被保険者」を取り出してしまい，結果は，

第二号被保険者としての被保険者期間を…

となってしまう．これは意味が変化してしまっており，誤りである．

6.1.2 箇条書き表現の検出実験

続いて，箇条書きの検出を行った結果を表 6.2 に示す．前述の通り，国民年金法には複数の条件文をもつ箇条書き表現が 34 件出現するが，このうち 1 つを検出できなかった．一方，箇条書き処理の対象としないものを誤って検出したものが 1 件あった．よって，適合率・再現率ともに 97.1%であった．

また，これらの箇条書き表現ひとつにつき，それぞれ複数の条件文をもつため，表では条件文の数も併記した．

表 6.2: 箇条書き表現の検出結果

	箇条書き数	条件文数
箇条書き検出成功	33	119
箇条書き検出失敗	1	5
全箇条書き数	34	124
誤検出	1	2

6.1.3 箇条書き表現に対する処理実験

次に，検出に成功した 33 件の箇条書き表現に対して，箇条書きを取り除く処理を行った．33 件の箇条書き表現に対して 119 の条件文があるため，全ての処理が成功した場合に出力される文は 119 文となる．実際の実験結果は表 6.3 の通り，不正確な出力がなされたものや，エラーが出力されて処理できなかったものがあった．

表 6.3: 検出した箇条書き表現に対する処理結果

	条件文数	%
箇条書き処理成功	87	73.1%
不正確な出力	21	17.7%
エラー	11	9.2%
合計	119	100%

不正確な出力とは，処理はできたものの出力文が意味を維持していないものを指す．様々な理由で，不自然な日本語になっていたり，意味が変わってしまったもので，例えば国民年金法第三十七条において，

被保険者又は被保険者であった者が**次の各号のいずれかに該当した**場合…

という本文に対して，次のような条件文が記述されている．

被保険者が、死亡したとき。

この場合，**キー表現**の「次の各号のいずれかに該当した」を**条件**である「被保険者が、死亡した」で置き換えることになるので，出力は

被保険者又は被保険者であった者が**被保険者が、死亡した**場合…

となり，冗長で理解しにくいだけではなく，論理式変換を失敗させてしまう．

エラーが出力されるものは，この実験での 11 件は，すべて同じ文型で，各号が単純な条件文ではなく，それぞれの条件を満たしたときに用いる値などが併記されているものである．例えば国民年金法第二十七条の三第二項では，

次の各号に掲げる場合における基準年度以後改定率の改定については、前項の規定にかかわらず、**当該各号に定める率**を基準とする。

という本文に対して，

一　物価変動率が名目手取り賃金変動率を上回り、かつ、名目手取り賃金変動率が一以上となるとき　**名目手取り賃金変動率**
二　物価変動率が一を上回り、かつ、名目手取り賃金変動率が一を下回るとき　**ー**

と，あてはまる条件によって「当該各号に定める率」が変化することを示している．

6.2　所得税法に対する実験

続いては，第 3 章・第 4 章において国民年金法に対する分析にもとづいて検討した処理法が，他の法令に対してどの程度有効であるかを調査するため，オープンテストとして，所得税法を用いて同様の処理を行い，出力を評価した．その結果を表 6.4 に示す．

不正確な出力，エラーともに国民年金法より大幅に増える結果となった．

これは，処理法の検討時に想定していなかった文型が増えたためであるが，特に，前節の実験で最後に述べた「当該各号に定める○○」という文型が，箇条書きの数では 26 件，

表 6.4: 検出した箇条書き表現に対する処理結果

	条件文数	%
箇条書き処理成功	219	51.4%
不正確な出力	123	28.9%
エラー	84	19.7%
合計	426	100%

条件文の数では 139 文と，国民年金法と比べて頻出したため，これらが全てエラー出力，あるいは出力はされたものの誤った文となった影響が大きかった．

例えば，所得税法第百五十条では，

> 第百四十三条（青色申告）の承認を受けた居住者につき**次の各号のいずれかに該当する**事実がある場合には、納税地の所轄税務署長は、**当該各号に掲げる**年までさかのぼつて、その承認を取り消すことができる。…

という本文に対して，次のような条件文が記述されている．

> 一　その年における　…　行なわれていないこと。　その年

この場合，**条件**である「その年における　…　行なわれていないこと。　その年」をそのまま埋め込んでしまい，出力は

> 第百四十三条（青色申告）の承認を受けた居住者につき**その年における　…　行なわれていないこと。　その年**は、納税地の所轄税務署長は、当該各号に掲げる年までさかのぼつて、その承認を取り消すことができる。…

となる．途中に全角スペースが入るなど，誤った文になってしまい，正しく論理式に変換することもできない．

第7章　おわりに

7.1　まとめ

本論文では，法令文を論理式に変換する先行研究のシステムでは扱えない表現である，法令文中の箇条書き表現と参照表現を取り除く手法について述べた．

各々，国民年金法における出現を分析し，それにもとづき，法令文を変換して箇条書きや参照表現を取り除く処理法を検討した．

実験の結果，結果，参照表現の処理については，4割近くの参照表現について，何らかの必要な情報を取得することができ，また何も抽出されなかった例についても，一部はそもそも情報を補完する必要がないものであった．その他，情報の抽出を誤った例に対しては，分類して問題点を分析した．構文木から情報を得るという方針そのものは適当であると考えているが，本論文で提案した単純な手法だけでは処理が難しい問題であることがわかったため，正しい情報をより多く取得できるように，構文木の扱い方を工夫することと，その他の手法も併用することが考えられる．

一方，箇条書き表現に対する処理については，国民年金法の全文に対して70%強の箇条書き表現に対して正しく処理を行うことができたが，全ての文型に対応しきれていなかった．また，オープンテストとして，所得税法に対して同様の処理を行った結果，処理の成功率が50%強にまで下がったことから，国民年金法と所得税法の間でも文型の傾向に違いがあると思われる．誤りの中でも出現頻度の高いものをより詳細に分析し，処理を加えることで処理を改善できると考えられる．

7.2　今後の課題

「法令文の論理式への変換」というテーマは大きなものであり，本研究ではやりつくせなかった点も多数残されている．その中でも特に本研究の内容と近いものを今後の課題として以下に挙げる．

- 箇条書き表現については，検出はほぼ成功したものの，想定していなかった文形式

に対してうまく処理できなかったため，これらに対応する処理を加える必要がある．それにより，所得税法においても同様の文形式が出現していたため，所得税法における処理の成功率も向上すると考えられる．

- 参照表現については，本論文で処理法を検討した「X に規定する Y」の他にも「X の規定により Y する」など，様々な表現があるため，それらに対する処理法を検討することが今後の課題といえる．

  その中でも特に,「第一項の申出」といった表現は,「第一項に規定する申出」と同様であると仮定して，同様の処理ができるのではないかと考えている．

  また，所得税法においては,「第百四十三条（青色申告）の承認を受けた…」など，条文指示の後に括弧書きで条文の内容を細くしていると考えられる表現が出現するため，この例では，第百四十三条の文を取得せずとも，括弧書きの中を抽出して「青色申告の承認を受けた…」としても良いのではないかと考えられる．

  加えて，現在「第一項又は第二項」「第三項及び第四項」や「前二項（＝前項および前々項）」といった，複数の条文をまとめて指示している表現に対しても，現状では対応できていない．

- 3.4 節で分析のみを行った**括弧書き**や,「ただし、」に続いて表記される**ただし書き**部分などからも情報を補足する必要がある．

- ただし書き部分を除けば１つの条文（一項）は一文から成り立っていることが多いが，**二文以上からなる条文**もあり，これらは単純に分割して処理することが可能かどうかの検討．また，そのような条文を指示する場合,「第〇条第〇号前段（後段）」といった表現が用いられる．

- 国民年金法第八十七条など，条文の中に**表組み**を含むものがあるので，これらも分解して処理する必要がある．その際，表中の特定の列に対する指示は,「第一欄，第二欄」等，または「上欄，中欄（３段になっている場合），下欄[1]」といった用語が用いられる [11]．

- 国民年金基金に対する国民年金基金連合会など，仕組みの良く似たものを複数規定する場合に，既存の条文を再利用して，一部の用語等を読み替える場合がある．その際,

[1]法令文は本来縦書きで表記されるため．

第○条の規定は，△△について**準用する**．この場合において，第○条中「……」とあるのは「…」と読み替えるものとする．

といった表現が用いられる．これに対する処理も必要となるであろう．

- 法令文の中には，国民年金法や所得税法も含め，促音の「っ」が「つ」になっているなど，**旧仮名遣い**を用いているものがあるが，現在のシステムで形態素解析に用いている JUMAN，構文解析に用いている KNP は，旧仮名遣いの文を処理できないため，法令文を予め現代仮名遣いに変換しておく必要がある．

  本研究で対象とした国民年金法と所得税法に関しては,「つた」を「った」に,「つて」を「って」に文字列変換するのみで，仮名遣いを改め，かつ無関係な文を誤って変換することもないと確認しているが，その他一般の法令に対しては処理を誤る可能性があるため，仮名遣いに対する処理も検討する必要がある．

謝辞

本研究を進めるにあたり，多大なご指導，ご支援を頂いた島津明教授に篤くお礼申し上げます．さらに，貴重なご助言を頂いた白井清昭准教授，中村誠助教に深く感謝致します。また，同期の皆様にもお世話になりましたことをお礼申し上げます．

参考文献

[1] 片山卓也. 検証進化可能電子社会 -情報科学による安心な電子社会の実現-. 情報処理, vol.46, No.5, pp.515-521, 2005.

[2] 片山卓也 (編). 法令工学の提案. JAIST Press, 2007.

[3] 吉野一. 法律エキスパートシステムの基礎. ぎょうせい, pp.12-24, 1986.

[4] 江尻暁, 北田安希雄, 島津明. 法令文の論理式への変換 -論理構造について-, 言語処理学会第 12 回年次大会, pp.624-627, 2006.

[5] 北田安希雄, 江尻暁, 島津明. 法令文の論理式への変換 -原子文について-, 言語処理学会第 12 回年次大会, pp.628-631, 2006.

[6] 信岡俊祐, 中村誠, 島津明. 法令文の論理式への変換, 言語処理学会第 13 回年次大会発表論文集, pp.254-257, 2007.

[7] Makoto Nakamura, Shunsuke Nobuoka, and Akira Shimazu. Towards Translation of Legal Sentences into Logical Forms. Proceedings of the First International Workshop on Juris-informatics (JURISIN 2007), pp. 6-17, Miyazaki, 2007.

[8] 田中規久雄, 川添一郎, 成田一. 法律条文の標準構造, 情報処理学会研究報告 (自然言語処理), 1993.

[9] Rutu Mulkar, Jerry R. Hobbs and Eduard Hovy. Learning from Reading Syntactically Complex Biology Texts, In proceedings of the 8th International Symposium on Logical Formalizations of Commonsense Reasoning, part of the AAAI Spring Symposium Series 2007, Vancouver, Canada.

[10] Ó. Ferrández, R. M. Terol, R. Muñoz, P. Martínez-Barco and M. Palomar. Deep Vs. Shallow Semantic Analysis Applied to Textual Entailment Recognition, Proceedings of

5th International Conference on Natural Language Processing(FinTAL 2006), Turku, Finland, 2006.

[11] 上田章, 笠井真一. 条例規則の読み方・つくり方, 学陽書房, 2000.

付 録A　国民年金法中の参照表現

X に規定する	86 件	第五条第二項，第五条第二項，第五条第二項，第五条第四項，第五条第五項，第五条第六項，第五条第七項，第五条の三第一項，第十二条第七項，第十二条第八項，第十七条第二項，第十九条第二項，第十九条第三項，第十九条第四項，第二十条第二項，第二十七条第一項第三号，第二十七条第一項第五号，第二十七条第一項第七号，第三十条第一項，第三十条の二第一項，第三十五条第一項第二号，第三十五条第一項第二号，第三十五条第一項第三号，第三十六条の二第二項，第三十六条の二第二項，第三十六条の二第三項，第三十六条の二第三項，第三十六条の二第四項，第三十六条の二第四項，第三十六条の二第四項，第三十六条の二第五項，第三十六条の二第六項，第三十六条の三第二項，第三十六条の四第二項，第三十六条の四第二項，第三十六条の四第三項，第三十六条の四第三項，第三十九条第一項，第三十九条第二項，第三十九条第三項，第四十条第二項，第四十条第二項，第四十六条第一項，第五十二条第一項，第五十二条の二第三項，第五十二条の三第二項，第五十二条の三第二項，第五十二条の六第一項，第七十二条第一項第二号，第八十五条第一項，第八十五条第一項第一号，第八十五条第一項第一号，第九十条第一項，第九十条第一項，第九十条第一項第四号，第九十条第四項，第九十条の二第一項，第九十条の二第二項，第九十条の二第三項，第九十条の二第五項，第九十条の三第一項，第九十条の三第三項，第九十二条の五第五項，第九十二条の六第一項第一号，第九十四条の五第三項，第九十四条の五第四項，第九十四条の五第五項，第百一条の二第一項，第百五条第一項，第百八条第一項，第百八条第一項，第百八条の二第一項，第百八条の三第二項，第百九条第二項，第百九条第二項，第百十六条第三項，第百二十五条の二第一項，第百二十五条の二第二項，第百二十八条第三項，第百二十八条第三項，第百二十八条第四項，第百三十七条の二第一項，第百三十七条の十三の二第一項，第百三十七条の十三の二第二項，第百三十七条の十九第一項，第百三十七条の十九第六項
X の規定による	71 件	第九条第一項，第十二条第三項，第十二条第三項，第十二条第四項，第二十七条の二第四項，第二十七条の三第三項，第二十七条の四第三項，第二十七条の五第三項，第三十条の二第四項，第三十条の二第四項，第三十四条第三項，第三十六条第三項，第三十六条の二第一項，第三十六条の三第一項，第三十六条の四第一項，第三十六条の四第一項，第三十六条の四第二項，第三十六条の四第二項，第四十一条の二第二項，第四十三条第一項，第四十四条第一項，第四十五条第一項，第四十五条第一項第一号，第四十五条第一項第二号，第五十二条の四第二項，第七十二条第一項第一号，第七十二条第一項第一号，第七十二条第一項第二号，第七十二条第一項第二号，第八十五条第一項第三号，第八十七条の二第二項，第八十七条の二第四項，第九十条第二項，第九十条第三項，第九十二条の三第三項，第九十二条の三第四項，第九十二条の三第五項，第九十二条の四第六項，第九十二条の六第一項，第九十二条の六第一項第二号，第九十二条の六第一項第四号，第九十二条の六第一項第四号，第九十六条第四項，第九十六条第五項，第九十六条第六項，第百一条第七項，第百二条第四項，第百七条第三項，第百十一条の二第一項，第百十二条第一項第三号，第百十三条の二第一項第一号，第百二十三条第三項，第百三十条第二項，第百三十条第二項，第百三十二条第三項，第百三十四の二条第一項，第百三十五条第一項第一号，第百三十七条の十一第三項，第百三十七条の十八第五項，第百三十七条の十九第五項，第百三十七条の十九第八項，第百三十七条の十九第八項，第百三十七条の二十一第二項，第百三十七条の二十二第一項第二号，第百三十八条第一項，第百三十八条第一項，第百四十一条第三項，第百四十三条第一項，第百四十三条第一項，第百四十三条第一項，第百四十五条第一項第四号

X の規定により	109 件	第四条の三第二項，第五条第二項，第五条第二項，第五条第四項，第五条第四項，第五条第五項，第五条第五項，第五条第六項，第五条第六項，第五条第七項，第五条第七項，第五条の二第二項，第五条の三第一項，第五条の三第一項，第十二条第九項，第十六条の二第一項，第二十条第二項，第二十条第二項，第二十条第四項，第二十六条第一項，第二十七条第一項，第二十七条第一項，第二十七条第一項第八号，第三十一条第二項，第三十二条第一項，第三十二条第二項，第三十二条の二第三項，第三十四条第六項，第三十六条の三第一項，第三十六条の三第一項，第三十六条の四第二項，第四十一条第二項，第四十二条第二項，第四十二条第二項，第四十九条第一項，第五十二条の六第一項，第八十五条第一項第二号，第八十七条の二第一項，第八十七条の二第一項，第八十七条の二第二項，第八十七条の二第三項，第八十七条の二第三項，第八十七条の二第三項，第八十七条の二第四項，第八十九条第一項，第九十条第一項，第九十条第一項，第九十条の二第一項，第九十条の二第一項，第九十条の二第二項，第九十条の二第二項，第九十条の二第三項，第九十条の二第三項，第九十条の二第六項，第九十条の三第一項，第九十条の三第一項，第九十二条の四第二項，第九十二条の四第三項，第九十二条の四第四項，第九十二条の四第四項，第九十二条の四第六項，第九十二条の五第四項，第九十二条の六第二項，第九十三条第三項，第九十四条第一項，第九十四条第一項，第九十四条第一項，第九十四条第二項，第九十四条第二項，第九十四条第二項，第九十四条第二項，第九十四条第二項，第九十四条第二項，第九十四条第四項，第九十五条の二第一項，第百八条の三第三項，第百十二条第一項第二号，第百十二条第一項第三号，第百十三条の二第一項第一号，第百十三条の二第一項第二号，第百十四条第一項第一号，第百十四条第一項第三号，第百十六条第一項，第百十六条第一項，第百二十五条の四第一項，第百二十七条第三項第三号，第百二十七条第三項第三号，第百三十四条の二第二項，第百三十七条第一項，第百三十七条第二項第一号，第百三十七条第二項第二号，第百三十七条の十三の四第一項，第百三十七条の十五第一項，第百三十七条の十七第二項，第百三十七条の十七第七項，第百三十七条の十八第一項，第百三十七条の十九第二項，第百三十七条の十九第四項，第百三十七条の十九第四項，第百三十七条の十九第七項，第百三十七条の二十第一項，第百三十七条の二十一第一項，第百三十七条の二十三第一項，第百三十七条の二十四第一項，第百三十七条の二十四第二項，第百四十二条第一項，第百四十二条の二第二項，第百四十三条第二項，第百四十七条第一項第三号
X の規定によって	14 件	第三十五条第一項，第四十条第二項，第四十条第三項，第四十二条第二項，第四十六条第二項，第五十二条の二第三項，第八十五条第一項第二号，第九十六条第二項，第九十七条第一項，第九十七条第四項，第百六条第二項，第百十三条第一項，第百四十一条第二項，第百四十七条第一項第一号
X の規定にかかわらず	26 件	第十条第一項，第二十条第二項，第二十七条の二第三項，第二十七条の三第一項，第二十七条の三第二項，第二十七条の四第一項，第二十七条の四第二項，第二十七条の五第一項，第二十七条の五第二項，第二十八条第三項，第二十八条第四項，第三十条の二第三項，第三十条の三第三項，第三十二条第二項，第三十三条第二項，第三十二条の二第一項，第三十六条の二第四項，第三十九条第一項，第三十九条の二第一項，第四十六条第一項，第四十九条第三項，第五十二条の二第二項，第五十二条の四第二項，第七十六条第二項，第九十二条の四第四項，第九十四条の六第一項
X の規定に該当する	1 件	第四十五条第三項，
X の規定に基づく	1 件	第七十六条第二項
X の規定に基づき	2 件	第百三十九条の二第一項，第百三十九条の二第一項
X の規定の例によって	1 件	第五十条第一項

X の規定に違反した	3 件	第七十九条第一項，第百二十五条の三第二項，第百三十七条の十三の三第二項
X の規定に違反して、	16 件	第九十二条の六第一項第三号，第百十二条第一項第一号，第百十三条第一項，第百十四条第一項第一号，第百十四条第一項第二号，第百十四条第一項第四号，第百四十五条第一項第一号，第百四十五条第一項第二号，第百四十五条第一項第三号，第百四十六条第一項第一号，第百四十六条第一項第二号，第百四十六条第一項第三号，第百四十七条第一項第一号，第百四十七条第一項第二号，第百四十七条第一項第四号，第百四十八条第一項
X の規定を適用 (する/しない) X の規定は適用しない	7 件	第二十八条第二項，第三十六条の二第二項，第三十六条の二第三項，第三十六条の二第五項，第四十五条第一項，第五十二条の二第三項，第百二条第五項
X の規定の適用が	4 件	第三十二条の二第一項，第三十九条第一項，第三十九条の二第一項，第百十三条の三第二項
X の規定の適用については	9 件	第三十二条の二第二項，第三十七条の二第二項，第三十九条第二項，第四十五条第三項，第九十二条の四第三項，第九十二条の四第四項，第九十二条の四第四項，第九十二条の四第四項，第百三十七条の十九第六項
X の規定の適用を受ける	6 件	第八十九条第一項，第九十条第一項，第九十条の二第一項，第九十条の二第二項，第九十条の二第三項，第九十条の二第三項
X の規定の適用上	3 件	第七条第二項，第三十二条の二第四項，第三十七条の二第二項
X (のいずれか) に該当する	26 件	第八条第一項，第八条第一項，第八条第一項，第九条第一項，第九条第一項，第九条第一項，第九条第一項第二号，第九条第一項第三号，第九条第一項第四号，第九条第一項第五号，第九条第一項第六号，第三十条の二第一項，第三十条の三第一項，第三十四条第四項，第三十六条第二項，第三十六条の二第一項，第三十七条第一項，第四十条第二項，第五十一条第一項，第五十二条の三第一項，第九十条の二第一項第二号，第九十条の二第二項第二号，第九十条の二第三項第二号，第九十条の三第一項第二号，第百二十七条第三項，第百二十七条第三項
X (のいずれにも) 該当しない	3 件	第七条第一項第一号，第八条第一項，第三十七条第一項第四号
X に掲げる	16 件	第二十七条の二第二項，第二十七条の二第二項，第二十七条の四第一項，第二十七条の四第二項第一号，第二十七条の四第二項第一号，第二十七条の四第二項第三号，第二十七条の五第二項第二号，第三十七条の二第一項第一号，第八十五条第一項第一号，第八十五条第一項第二号，第九十二条の三第一項，第九十二条の四第一項，第百二十条第二項，第百三十五条第二項，第百三十七条の十五第二項，第百三十七条の二十二第二項
X に定める	17 件	第二十八条第三項，第三十三条第二項，第三十二条の二第一項，第三十九条第一項，第三十九条の二第一項，第五十二条の四第二項，第八十七条の二第一項，第八十七条の二第二項，第八十九条第一項第三号，第九十二条第二項，第九十三条第四項，第九十四条第五項，第百十九条の二第六項，第百二十二条第八項，第百三十七条第六項，第百三十七条の六第六項，第百三十七条の十第八項

X の場合において	13 件	第十条第二項，第十九条第二項，第十九条第三項，第二十二条第二項，第三十九条の二第二項，第四十五条第二項，第四十五条第三項，第九十三条第二項，第九十四条第二項，第九十四条第三項，第九十四条の三第二項，第九十七条第二項，第百三十七条第三項
X の場合に準用する	7 件	第三十条の二第二項，第三十条の三第二項，第三十条の四第三項，第三十四条第五項，第三十六条第三項，第四十九条第二項，第九十条の三第二項
X において準用する	22 件	第五条の三第一項，第百十四条第一項第一号，第百十四条第一項第三号，第百三十七条の二十一第二項，第百三十七条の二十一第二項，第百三十七条の二十一第三項，第百三十八条第一項，第百三十八条第一項，第百四十五条第一項第一号，第百四十六条第一項第一号，第百四十六条第一項第三号，第百四十七条第一項第一号，第百四十七条第一項第一号，第百四十七条第一項第一号，第百四十七条第一項第一号，第百四十七条第一項第二号，第百四十七条第一項第二号，第百四十七条第一項第三号，第百四十七条第一項第三号，第百四十七条第一項第三号，第百四十七条第一項第四号
X を準用する	1 件	第百三十八条第一項
X において「…」という	3 件	第四条の三第二項，第三十四条第四項，第九十条第一項
X の規定は、…準用する。	39 件	第三十条の二第二項，第三十条の三第二項，第三十条の四第三項，第三十四条第五項，第三十六条第三項，第四十二条第三項，第四十六条第二項，第四十九条第二項，第五十二条の五第一項，第九十条の二第四項，第百五条第二項，第百五条第二項，第百五条第五項，第百七条第三項，第百二十七条の二第一項，第百三十三条第一項，第百三十三条第一項，第百三十三条第一項，第百三十四条の二第一項，第百三十四条の二第一項，第百三十七条第五項，第百三十七条第五項，第百三十七条の七第四項，第百三十七条の八第二項，第百三十七条の九第一項，第百三十七条の十三第六項，第百三十七条の十五第五項，第百三十七条の十七第五項，第百三十七条の十八第五項，第百三十七条の十九第八項，第百三十七条の十九第八項，第百三十七条の二十一第一項，第百三十七条の二十一第一項，第百三十七条の二十一第一項，第百三十七条の二十一第一項，第百三十七条の二十一第二項，第百三十七条の二十一第三項，第百三十七条の二十四第三項
X の規定及び…は、…準用する	1 件	第百三十七条の二十一第二項
(X 中「…」とあり、並びに) X 中「…」とあるのは	25 件	第十八条の三第一項，第三十条の三第二項，第四十二条第二項，第四十五条第三項，第四十六条第二項，第四十九条第二項，第五十二条の五第一項，第九十二条の三第二項，第百二十七条の二第一項，第百二十七条の二第一項，第百三十三条第一項，第百三十三条第一項，第百三十三条第一項，第百三十四条の二第一項，第百三十四条の二第一項，第百三十四条の二第一項，第百三十四条の二第一項，第百三十七条の七第四項，第百三十七条の二十一第一項，第百三十七条の二十一第一項，第百三十七条の二十一第一項，第百三十七条の二十一第一項，第百三十七条の二十一第二項，第百三十七条の二十一第二項，第百三十七条の二十一第三項

X において同じ	6 件	第十二条第一項，第三十四条第四項，第三十六条の二第三項，第五十二条の三第二項，第百一条第一項，第百十六条第一項
X と同様とする	1 件	第百四十三条第二項
X を除く	4 件	第二十七条第一項，第百一条第五項，第百三十七条の二十四第三項，第百三十八条第一項
X の （その他名詞）	90 件	第二条第一項（目的），第四条の三第二項（財政均衡期間），第七条第三項（認定），第十条第二項（承認），第十条第二項（承認の申請），第十条第二項（承認の申請），第十二条第一項（届出），第十二条第六項（届出），第十二条第八項（経由），第十二条第九項（届出），第十六条の二第一項（調整），第二十条第二項（申請），第二十条第四項（申請），第二十条第四項（申請），第二十八条第二項（申出），第二十八条第二項（申出），第二十八条第三項（申出），第二十八条第四項（申出），第三十条の二第一項（障害基礎年金），第三十条の二第三項（請求），第三十条の二第三項（障害基礎年金），第三十条の二第四項（障害基礎年金），第三十条の二第四項（請求），第三十条の三第三項（障害基礎年金），第三十条の四第二項（障害基礎年金），第三十四条第三項（請求），第七十六条第一項（目的），第八十七条第四項（保険料改定率），第八十七条第五項（保険料改定率），第八十七条の二第四項（申出），第九十二条の三第二項（委託），第九十二条の四第一項（委託），第九十四条の五第二項（報告），第九十六条第三項（督促状），第百一条第三項（審査請求），第百一条第三項（再審査請求），第百一条第五項（審査請求），第百一条第五項（再審査請求），第百二条第二項（時効），第百五条第二項（届出），第百八条の三第一項（目的），第百九条第一項（届出），第百九条第三項（認可），第百九条第三項（認可），第百十一条の二第二項（国民年金基金又は国民年金基金連合会），第百十一条の二第二項（違反行為），第百十一条の二第二項（罰金刑），第百十三条の三第一項（違反行為），第百十三条の三第一項（刑），第百十五条第一項（目的），第百十九条第二項（設立委員），第百十九条の二第二項（公告），第百十九条の二第四項（規約），第百十九条の四第二項（設立），第百二十条第三項（規約），第百二十条第四項（政令），第百二十二条第四項（設立），第百二十四条第三項（設立），第百二十四条第六項（設立），第百二十七条第二項（申出），第百二十八条第三項（免許），第百二十八条第三項（事業），第百二十八条第六項（業務），第百二十八条第六項（申出），第百三十七条の六第二項（公告），第百三十七条の六第四項（規約），第百三十七条の七第三項（設立），第百三十七条の十第四項（設立），第百三十七条の十二第三項（設立），第百三十七条の十二第六項（設立），第百三十七条の十五第五項（信託），第百三十七条の十七第三項（交付），第百三十七条の十七第四項（交付），第百三十七条の十七第五項（年金），第百三十七条の十七第六項（交付），第百三十七条の十七第八項（通知），第百三十七条の十七第八項（通知），第百三十七条の十八第二項（交付），第百三十七条の十八第三項（交付），第百三十七条の十八第四項（交付），第百三十七条の十九第三項（年金），第百三十七条の十九第三項（一時金），第百四十二条第三項（命令），第百四十二条第三項（命令），第百四十二条第四項（命令），第百四十二条第四項（命令），第百四十二条第四項（命令），第百四十二条第五項（命令），第百四十四条第一項（違反行為），第百四十四条第一項（罰金刑）
合計	633 件	

付 録B　プログラム

B.1 conditions.pl - 条件文の箇条書き表現に対する処理

```
#!/usr/local/bin/perl
# 条件の箇条書きを処理する (s0610030) 2007/12/20

use utf8;
use strict;

my $charset = 'euc-jp';
#my $charset = 'shiftjis';
binmode(STDIN, ":encoding($charset)"); # 標準入力のコード指定
binmode(STDOUT,":encoding($charset)"); # 標準出力のコード指定
binmode(STDERR,":encoding($charset)"); # 標準エラー出力のコード指定

my $infile = $ARGV[0]; # 引数から入力 XML ファイル名
if(!$infile) { $infile = 'nenkinhou.txt'}; # デフォルトの入力 XML ファイル名

open(INFILE, "<:utf8", $infile) || die "開けません！";

my $article = '';
my $nextline = '';
my ($article,$paragraph,$item,$subitem);

$_ = <INFILE>; # 1 行目は読み飛ばす

$_ = <INFILE>;

while ($_)
{
chomp($_);

$nextline = '';

if (s/^[　|\t]?(第 [一二三四五六七八九十百千]+条 (の [一二三四五六七八九十百千]+)*)[　|\t]+//)
{ $article = $1; $paragraph = ' １'; $item = ''; $subitem = ''; }

if (s/^[　|\t]?([１２３４５６７８９０]+)[　|\t]+//)
{ $paragraph = $1;  $item = ''; $subitem = ''; }

print STDERR "$article : $paragraph\n";

if (s/(次 (の各号)?(のいずれか)?(に掲げる|(に掲げる要件)?に該当 (した|する (に至った|至つた)?)))/<<KEY>>/)
{

my $key = $1;
my $next = '';
my $outf = 0;

if (s/(.*?)<<KEY>>(とき|場合|日|期間|者|要件|.+?から)(.+?)/$1<<KEY>>$2<<SUP>>$3/)
{ $next = $2; }
elsif (s/(.*?)<<KEY>>([^、。]+?)(を|は|により)(.+?)/$1<<KEY>>$3<<SUP>>$4/)
{ $next = $3; } # キー直後を取り除く場合（次に掲げる「事項」を、次に掲げる「理由」により等）
```

```
# print STDERR "$article : $paragraph : $_\n";

my $org = $_;
if (!$next) { next; }

print "\n\n$article - $paragraph\n$org\n* $key - $next\n\n";

my $condition = <INFILE>;
chomp($condition);

while ($condition =~ s/^[ 　|\t]*([一二三四五六七八九十百千]+(の[一二三四五六七八九十百千]+)*)[ 　|\t]+//) {
$item = $1;
my $outtext = '';
my $supplement = '';
$outf = 0;

$condition =~ s/^([^(（)）]+?|(.+?（[^(（)）]）)+?|(.+?（(.+?（[^(（)）]）)+?）)+?)。(.+)$/$1（$+）/;
# 二文以上に分かれている場合、二文目以降を括弧に入れる
if ($condition =~ s/^(.+)((（[^(（)（）)]*?）|（[^(（)）]*?(（[^(（)（）)]*?）)+?[^(（)]*?）)(。)?$/$1/)
{ $supplement = $2; } # 文末にある括弧（二重括弧の場合外側）を補足説明とみなす

if (($next eq 'とき')
|| ($next eq '場合')
|| ($next eq '日')) {
if ($condition =~ s/^(.+)とき(。)?$/$1/) { $outf = 1; } }

if ($next eq '期間') {
if ($condition =~ s/^(.+)もの(。)?$/$1/) { $outf = 1; } }

if ($next eq '者') {
if ($condition =~ s/^(.+)(こと|もの|者)(。)?$/$1/) { $outf = 1; }
else {   # そのまま入れる
if (!($condition =~ m/　/)) {
$condition =~ s/^(.+)(。)?$/$1/;
$outtext = $org;
$outtext =~ s/<<KEY>>.*?<<SUP>>/$condition$supplement/;
}
} }

if ($next eq '要件') {
if ($condition =~ s/^(.+)こと(。)?$/$1/) { $outf = 1; } }

if ($next =~ m/^.+から$/) {
if ($condition =~ s/^(.+)とき(。)?$/$1/) { $outf = 1; } }

if ($next eq 'を' || $next eq 'は' || $next eq 'により') {
$condition =~ s/^(.+)(。)?$/$1$supplement/;
$supplement = '';
$outf = 1; }

if ($outf) {
$outtext = $org;
$outtext =~ s/<<KEY>>/$condition/;
$outtext =~ s/<<SUP>>/$supplement/;
# $outtext =~ s/<<KEY>>/【$condition】/;
# $outtext =~ s/<<SUP>>/≪$supplement ≫/;
}

if ($outtext) {
print "[$article:$paragraph:$item]\t$outtext\n";
} else { print ";ERROR:[$article:$paragraph:$item]\t$condition\n";
print STDERR "; $article - $paragraph 条件文が形式に合っていません\n"; }

$condition = <INFILE>;
chomp($condition);
}
```

```
$nextline = $condition;

# } else { # 条件の箇条書きじゃない場合はそのまま出力
# if (s/^([一二三四五六七八九十百千]+)　//) { $item = $1; }
# if (s/^([イロハニホヘトチリヌルヲ]+)　//) { $subitem = $1; }
# if (!(m/^($|(　)|(（）/)) { # 空行，行頭が全角スペースか括弧の場合は出力しない
# if ($_ eq "削除") { print ";"; } # 「削除」のみの行はコメントアウト
# print "[$article:$paragraph";
# if ($item) { print ":$item"; }
# if ($subitem) { print ":$subitem"; }
# print "]\t$_\n";
# }
# ここまでをコメントアウトすれば箇条書きのみ出力

}
if ($nextline) { $_ = $nextline; } else { $_ = <INFILE>; }

}
close(INFILE);
```

B.2 reference.pl - 他条文への参照表現に対する処理

```
#!/usr/local/bin/perl
# 条件の箇条書きを処理する (s0610030) 2007/12/20

use utf8;
use strict;
require 'xml_search.pl';
#require 'ref_search.pl'; # このファイル内に組み込んだ

our $xmldata = &xml_read();

my $charset = 'euc-jp';
#my $charset = 'shiftjis';
binmode(STDIN, ":encoding($charset)");     # 標準入力のコード指定
binmode(STDOUT,":encoding($charset)");     # 標準出力のコード指定
binmode(STDERR,":encoding($charset)");     # 標準エラー出力のコード指定

my $infile = $ARGV[0];     # 引数から入力ファイル名
if(!$infile) { $infile = 'nenkinhou.txt'};     # デフォルトの入力ファイル名

open(INFILE, "<:utf8", $infile) || die "$infile が開けません";

$_ = <INFILE>; # 1 行目は読み飛ばす

$_ = <INFILE>;

my($article,$pre_article,$same_article);
my($paragraph,$pre_paragraph,$same_paragraph);
my($item,$pre_item,$same_item);
my($tempnum,$onenum);

my $op;
my $btext;
my $xtext;
my($w,$x,$y);
my $xflag;

while ($_)
{
chop($_);

my $nextline = '';

if (s/^(第 [一二三四五六七八九十百千]+条 (の [一二三四五六七八九十百千]+)*)  //)
    {
    $pre_article = $article;
    $tempnum = $1;
    $tempnum =~ s/^第 ([一二三四五六七八九十百千]+) 条//;
    $article = &kan_num($1);
     while ($tempnum =~ s/^の ([一二三四五六七八九十百千]+)//) {
        $onenum = &kan_num($1);
        $article = $article. '_'. $onenum;
    }
    $pre_paragraph = '0'; $paragraph = '1';
    $pre_item = '0'; $item = '';
    }

if (s/^([１２３４５６７８９０]+)  //)
    {
    $pre_paragraph = $paragraph;
    $paragraph = &zen_num($1);
    $pre_item = '0'; $item = '';
    }
```

```
#    print "$article - $paragraph\n";

if (s/^([一二三四五六七八九十百千]+)　//)
    {
    $pre_item = $item;
    $item = &kan_num($1);
    }

# ここから条文指示

$op = $_;
$btext = $_; # 変更前テキスト
$xtext = ''; # 変更後テキスト格納
$xflag = 0; # 変更フラグ

while ($op =~ s/^(.*?)((第 [一二三四五六七八九十百千]+(条 (の [一二三四五六七八九十百千]+)*|項|号)|(前|次|同)(条|項
|号))+)(.+)/$8/) {
    $w = $1;    # 指示の前
    $x = $2;    # 指示先
    $y = $8;    # 指示の後
#    $extext = ''; # 抽出テキスト格納

    if ($w =~ m/.+法（（.+?）)?$/ || $w =~ m/.+法律$/ )
       { $op = $w. $y; }  # 直前が「〜法（律)」のときは別法令の参照なのでいまは除外
    else {
        my($xarticle,$xparagraph,$xitem)
         = &refid($x,$article,$pre_article,$same_article
        ,$paragraph,$pre_paragraph,$same_paragraph,$item,$pre_item,$same_item);
       $same_article = $xarticle;
       $same_paragraph = $xparagraph;
       $same_item = $xitem;

#    if ($y !~ m/^[）)]/) { # 直後が「)」のときは法令の説明なのでここでは除外

        if ($op =~ s/^に規定する (.+)/$1/) { # 直後が「に規定する」の場合のみ処理する
            my $reftext = &xml_cont_search($xmldata,$xarticle,$xparagraph,$xitem);
            my($corstr, $corlen) = &ref_search($op,$reftext);

           $xflag = 1;

            if ($corlen) { $x = substr($corstr,0,(length($corstr)-$corlen)); }
            else { $x .= "(抽出失敗)"; }

            print STDERR localtime(time). " / $article:$paragraph:$item / $x / $xarticle:$xparagraph:$xitem
 / [$corstr,$corlen] /". substr($y,0,$corlen). "\n";
           }
       }

       $xtext .= $w. "[". $x. "]";

   }
    if($xflag) {
       $xtext .= $op;
        print "$article:$paragraph:$item\t$btext\t$xtext\n";
   }

if ($nextline) { $_ = $nextline; } else { $_ = <INFILE>; }
}
close(INFILE);

sub refid {    # 参照表現中の指示部分を具体的な条文番号に変換
    my($x,$article,$pre_article,$same_article
    ,$paragraph,$pre_paragraph,$same_paragraph,$item,$pre_item,$same_item) = @_;
    my $xbak = $x;
    my $xarticle = $article;
    my $xparagraph = $paragraph;
```

```
    my $xitem = $item;
    my($tempnum,$onenum);

    if ($x =~ s/^(第[一二三四五六七八九十百千]+条(の[一二三四五六七八九十百千]+)*)//) {
       $tempnum = $1;
       $tempnum =~ s/^第([一二三四五六七八九十百千]+)条//;
       $xarticle = &kan_num($1);
       while ($tempnum =~ s/^の([一二三四五六七八九十百千]+)//) {
          $onenum = &kan_num($1);
          $xarticle = $xarticle. '_'. $onenum;
       }
       $xparagraph = '1'; $xitem = '';
    }
    if ($x =~ s/^前条//) { $xarticle = $pre_article; $xparagraph = '1'; $xitem = ''; }
    if ($x =~ s/^同条//) { $xarticle = $same_article; $xparagraph = '1'; $xitem = ''; }
    if ($x =~ s/^次条//) {
       $xarticle =~ m/(([0-9]+_)*)([0-9]+)/;
       $onenum = $3 + 1; $xarticle = $1. $onenum; $xparagraph = '1'; $xitem = ''; }

    if ($x =~ s/^第([一二三四五六七八九十百千]+)項//) { $xparagraph = &kan_num($1); $xitem = ''; }
    if ($x =~ s/^前項//) { $xparagraph = $pre_paragraph; $xitem = ''; }
    if ($x =~ s/^同項//) { $xarticle = $same_article; $xparagraph = $same_paragraph; $xitem = ''; }
    if ($x =~ s/^次項//) { $xparagraph++; $xitem = ''; }

    if ($x =~ s/^第([一二三四五六七八九十百千]+)号//) { $xitem = &kan_num($1); }
    if ($x =~ s/^前号//) { $xitem = $pre_item; }
    if ($x =~ s/^同号//) { $xarticle = $same_article; $xparagraph = $same_paragraph; $xitem = $same_item;}
    if ($x =~ s/^次号//) { $xitem++; }

    if ($x) { print STDERR "; $article:$paragraph:$item - 規定外の条文指示です:$xbak\n"; }

    return($xarticle,$xparagraph,$xitem);
}

sub zen_kan {    # 全角数字→漢数字
    my $out = '';
    my $temp = '';
    my $in = $_[0];
    my $length = length($in);
    my @kan = ('〇', '一', '二', '三', '四', '五', '六', '七', '八', '九');
    my @zen = ('０', '１', '２', '３', '４', '５', '６', '７', '８', '９');

    while ($length--) {
       $in =~ s/^([０１２３４５６７８９])//;
       if ($1 ne $zen[0]) {    # "０"はとばす
          foreach (1 .. 9) {
             if ($1 eq $zen[$_]) { $temp = $kan[$_]; }
          }
          if (($length) && ($1 eq $zen[1])) { $temp = ''; }    # "一百"とかを防ぐ
          if ($length == 1) { $temp = $temp. "十"; }
          if ($length == 2) { $temp = $temp. "百"; }
          if ($length == 3) { $temp = $temp. "千"; }
       }
       $out = "$out$temp";
       $temp = '';
    }
    return $out;
}


sub zen_num {    # 全角数字→数値
    my $out = 0;
    my $temp = 0;
    my $in = $_[0];
    my $length = length($in);
    my @zen = ('０', '１', '２', '３', '４', '５', '６', '７', '８', '９');
```

```
    while ($length--) {
       $in =~ s/^([０１２３４５６７８９])//;
        foreach (1 .. 9) {
            if ($1 eq $zen[$_]) { $out = $out + $_; }
        }
        if ($length) { $out = $out * 10; }
    }
    return $out;
}

sub kan_num {
    # 漢数字→数値に変換（1〜9999 まで）
    # 条文番号に用いられる形式のみ対応（「二千三十」は良い、「二、〇三〇」はダメ）
    my $out = 0;
    my $temp = 0;
    my $in = $_[0];
    my $length = length($in);
    my @kan = (’〇’, ’一’, ’二’, ’三’, ’四’, ’五’, ’六’, ’七’, ’八’, ’九’);

    while ($length--) {
       $temp = 0;
        if ($in =~ s/^([〇一二三四五六七八九])//) {
            foreach (1 .. 9) {
                if ($1 eq $kan[$_]) { $temp = $_; }
            }
        }
        if ($in =~ s/^十//) { $temp = $temp * 10; if (!$temp) { $temp = 10; } }
        if ($in =~ s/^百//) { $temp = $temp * 100; if (!$temp) { $temp = 100; } }
        if ($in =~ s/^千//) { $temp = $temp * 1000; if (!$temp) { $temp = 1000; } }

       $out = $out + $temp;
    }
return $out;
}

sub ref_search { #
    my($searchstr,$reftext) = @_;
#    print STDERR "+ $searchstr : $reftext\n";

    my $refstr; # 抽出候補の文字列
    my $corstr; # 一致部分が最長の文字列
    my $corlen = 0; # 一致部分が最長の文字列の，一致部分の長さ
    my $kakarinum; # 処理中の要素の係り先番号
    my $a;
    my $reftexttemp;

#    print STDERR "@@@ $reftext\n";
   $reftext =~ s/( ([^( ()() )]+) | ([^() )]+? (.+?) [^( ()]+?) )//g; # 括弧書きを削除
#    print STDERR "@@ $reftext\n";


    while ($reftext =~ m/^(.+?。)(.*)$/ || $reftext =~ m/^([^。]+?)()$/) {
        my @tempstr; # 処理中の文字列（各要素ごとに，そこに係っているすべての文を入れる）
        my $curnum = -1; # 処理中の要素番号

       $reftexttemp = $1;
       $reftext = $2;
print STDERR "@ $reftexttemp\n";

        open(OUTTEMP, ">:encoding(euc-jp)", ’reference.temp1’)  || die "reference.temp1 に書き込めません";
        print OUTTEMP "$reftexttemp\n";
        close(OUTTEMP);

        system "/opt/nlp/bin/juman -e < reference.temp1 |/opt/nlp/bin/knp -tab > reference.temp2";
```

```
        open(INSTR, "<:encoding(euc-jp)", 'reference.temp2') || die "reference.temp2 が開けません";

       $_ = <INSTR>; # 1 行目は読み飛ばす

        while (<INSTR>)
       {
           chop($_);

#print STDERR ":: $_\n";

           if (m/^EOS/) {
              $tempstr[$kakarinum] .= $tempstr[$curnum];
          } elsif (m/^\* ([\-0-9]+)/) {
               if ($curnum > -1) { $tempstr[$kakarinum] .= $tempstr[$curnum]; }
              $curnum++; $kakarinum = $1;
#print STDERR "* $curnum, $kakarinum\n";
          } else {
               if ($curnum < 0) { $corlen = -1; }
               else {
                   m/^(.+?) /; $tempstr[$curnum] .= $1;

                  $a = $corlen + 1; # 現時点で最長の一致より長いものを探す
#print STDERR "** $corlen : $corstr : $tempstr[$curnum] : ". substr($searchstr,0,$corlen+1). "\n";
                  while(index($tempstr[$curnum], substr($searchstr,0,$a)) > -1 && $a <= length($searchstr))
                      { $corstr = $tempstr[$curnum]; $corlen = $a; $a++;
#print STDERR "*** $corlen : $corstr : $tempstr[$curnum]\n";
                      }
              }
          }
       }

        close(INSTR);

   }

#        print STDERR "- $corlen : $corstr : $tempstr[$curnum]\n";
    return $corstr,$corlen; # 抽出した情報と一致した文字数を返す
}
```

B.3 xml_search.pl - XML 形式の法令文から条文検索（reference.pl で使用）

```
#!/usr/local/bin/perl

# XML 形式の法令文ファイル（文字コード UTF-8）から指定した条文を引っぱり出す
#
# 第一引数：条文を指す文字列（「第百三十七条の十五第二項第二号」など）
# 第二引数：XML ファイル名（省略するとデフォルトのファイル名）
# （例）perl contsearch.pl 第百三十七条の十五第二項第二号 nenkinhou.xml
#
# perl v5.8.8 で動作確認
# 木村祐介（s0610030）2007.10.9

use utf8;
use strict;
use XML::Simple;
use Data::Dumper;
use Encoding;

sub xml_read {    # xml 化した法令文をハッシュ表に読み込み

    my $infile = $_[0];     # 引数から入力 XML ファイル名
    if(!$infile) { $infile = 'nenkinhou.xml'};     # デフォルトの入力 XML ファイル名

   # XML ファイルの内容をハッシュテーブルに登録
   # 出現するエレメントが 1 つでも配列に格納するのは、article, paragraph, item
   # num をハッシュのキーとみなす
    my $simple = new XML::Simple(forcearray => ['article', 'paragraph', 'item'], keyattr => ['num']);
    my $data   = $simple->XMLin($infile);

    return $data;
}

sub xml_cont_search { # 位置情報から、ハッシュ表をたどり、条文を取得する
    my($data,$article,$paragraph,$item) = @_;
    my $text = '';


    if ($item) {
       $text = $data->{article}->{$article}->{paragraph}->{$paragraph}->{item}->{$item}->{text};
   } else {
       $text = $data->{article}->{$article}->{paragraph}->{$paragraph}->{text};
   }

#    print STDERR "xml_cont_search: $data,$article,$paragraph,$item,$text\n";

    return $text;
}


sub xml_locator {
   # 「第＊条（の＊）第＊項第＊号」形式の文字列から、
   # 位置変数 $article（条番号）, $paragraph（項番号）, $item（号番号）に変形

    my $article = '';
    my $paragraph = '';
    my $item = '';

    my $temp = '';     # 一時変数
    my $one = '';     # 一時変数

    my $in = $_[0];

   # 「第＊条（の＊）」部分を処理し、変数$article に入れる
    if ($in =~ s/^(第[一二三四五六七八九十百千]+条(の[一二三四五六七八九十百千]+)*)//)
```

```
    {
        $temp = $1;
        $temp =~ s/^第 ([一二三四五六七八九十百千]+) 条//;
        $article = &kan_num($1);
        while ($temp =~ s/^の ([一二三四五六七八九十百千]+)//) {
            $one = &kan_num($1);
            $article = "$article\_$one";
        }
    }

    # 「第＊項」部分を処理し、変数$paragraph に入れる
    if ($in =~ s/^第 ([一二三四五六七八九十百千]+) 項//)
        { $paragraph = &kan_num($1); }

    # 「第＊号」部分を処理し、変数$item に入れる
    if ($in =~ s/^第 ([一二三四五六七八九十百千]+) 号//)
        { $item = &kan_num($1); }

    my @result = ($article, $paragraph, $item);

    return \@result;
}


sub kan_num {
    # 漢数字→数値に変換（1〜9999 まで）
    # 条文番号に用いられる形式のみ対応（「二千三十」は良い、「二、〇三〇」はダメ）
    my $out = 0;
    my $temp = 0;
    my $in = $_[0];
    my $length = length($in);
    my @kan = (’〇’, ’一’, ’二’, ’三’, ’四’, ’五’, ’六’, ’七’, ’八’, ’九’);

    while ($length--) {
        $temp = 0;
        if ($in =~ s/^([〇一二三四五六七八九])//) {
            foreach (1 .. 9) {
                if ($1 eq $kan[$_]) { $temp = $_; }
            }
        }
        if ($in =~ s/^十//) { $temp = $temp * 10; if (!$temp) { $temp = 10; } }
        if ($in =~ s/^百//) { $temp = $temp * 100; if (!$temp) { $temp = 100; } }
        if ($in =~ s/^千//) { $temp = $temp * 1000; if (!$temp) { $temp = 1000; } }

        $out = $out + $temp;
    }
return $out;
}

1;
```

B.4 xml_tagging.pl - プレーンテキストの法令文に自動で XML タグ付け

```
#!/usr/local/bin/perl

# 法令文テキストを XML 形式に変換する（入力・出力とも文字コード UTF-8）
#
# 入力するプレーンテキストファイル名（文字コードを UTF-8 にしておくこと）と
# 出力先となる XML ファイル名は、ソースで指定する
#
# perl v5.8.8 で動作確認
# 木村祐介 (s0610030) 2007.10.9

use utf8;
use Encoding;

$infile = 'nenkinhou.txt';     # 入力元（プレーンテキストファイル名）
$outfile = 'nenkinhou.xml';     # 出力先（XML ファイル名）

open(INFILE, "<:utf8", $infile) || die "入力ファイル '$infile' を開けません！";
open(OUTFILE, ">$outfile") || die "出力ファイル '$outfile' を開けません！";

$article_num = 0;     # 条
$article_flag = 0;
$paragraph_num = 0;     # 項
$paragraph_flag = 0;
$item_num = 0;         # 号
$item_flag = 0;

print OUTFILE '<?xml version="1.0" encoding="UTF-8"?>';

print OUTFILE "\n\n<document>\n";

while (<INFILE>)
{
chop($_);

if (s/^(第 [一二三四五六七八九十百千]+条 (の [一二三四五六七八九十百千]+)*)  /<article num="<<num>>" serialnum="<<
serialnum>>"><title><<title>><\/title>\n<paragraph num="1" serialnum="<<serialnum_p>>"><title>第一項<\/titl
e>\n<text>/)
    {
    $title = $1;
    $tempnum = $title;
    $tempnum =~ s/^第 ([一二三四五六七八九十百千]+) 条//;
    $num = &kan_num($1);
     while ($tempnum =~ s/^の ([一二三四五六七八九十百千]+)//) {
        $onenum = &kan_num($1);
        $num = "$num\_$onenum";
    }
    $article_num++;
    $paragraph_num++;
     s/<<title>>/$title/;
     s/<<num>>/$num/;
     s/<<serialnum>>/$article_num/;
     s/<<serialnum_p>>/$paragraph_num/;

     if ($article_flag) { s/^</<\/article>\n</; }
     if ($paragraph_flag) { s/^</<\/paragraph>\n</; }
     if ($item_flag) { s/^</<\/item>\n</; }
    $article_flag = 1; $paragraph_flag = 1; $item_flag = 0;

    $_ = Encode::encode('utf-8',$_) if utf8::is_utf8($_);     # Wide character in print at... エラーが出ないよ
うに
     print OUTFILE "$_</text>\n";
    }
```

```
if (s/^([１２３４５６７８９０]+)  /<paragraph num="<<num>>" serialnum="<<serialnum>>"><title>第<<title>>項<\/t
itle>\n<text>/)
    {
    $title = &zen_kan($1);
    $num = &zen_num($1);
    $paragraph_num++;
     s/<<title>>/$title/;
     s/<<num>>/$num/;
     s/<<serialnum>>/$paragraph_num/;

     if ($paragraph_flag) { s/^</<\/paragraph>\n</; }
     if ($item_flag) { s/^</<\/item>\n</; }
    $paragraph_flag = 1; $item_flag = 0;

    $_ = Encode::encode('utf-8',$_) if utf8::is_utf8($_);     # Wide character in print at... エラーが出ないよ
うに
     print OUTFILE "$_</text>\n";
    }

if (s/^([一二三四五六七八九十百千]+)  /<item num="<<num>>" serialnum="<<serialnum>>"><title>第<<title>>号<\/ti
tle>\n<text>/)
    {
    $title = $1;
    $num = &kan_num($title);
    $item_num++;
     s/<<title>>/$title/;
     s/<<num>>/$num/;
     s/<<serialnum>>/$item_num/;

     if ($item_flag) { s/^</<\/item>\n</; }
    $item_flag = 1;

    $_ = Encode::encode('utf-8',$_) if utf8::is_utf8($_);     # Wide character in print at... エラーが出ないよ
うに
     print OUTFILE "$_</text>\n";
    }

}

if ($item_flag) { print OUTFILE "</item>\n"; }
if ($paragraph_flag) { print OUTFILE "</paragraph>\n"; }
if ($article_flag) { print OUTFILE "</article>\n"; }
print OUTFILE "</document>\n";

close(INFILE);
close(OUTFILE);



sub zen_kan {    # 全角数字→漢数字
my $out = '';
my $temp = '';
my $in = $_[0];
my $length = length($in);
my @kan = ('〇', '一', '二', '三', '四', '五', '六', '七', '八', '九');
my @zen = ('０', '１', '２', '３', '４', '５', '６', '７', '８', '９');

while ($length--) {
    $in =~ s/^([０１２３４５６７８９])//;
     if ($1 ne $zen[0]) {    # "０"はとばす
         foreach (1 .. 9) {
             if ($1 eq $zen[$_]) { $temp = $kan[$_]; }
         }
         if (($length) && ($1 eq $zen[1])) { $temp = ''; }    # "一百"とかを防ぐ
         if ($length == 1) { $temp = "$temp十"; }
         if ($length == 2) { $temp = "$temp百"; }
```

```
        if ($length == 3) { $temp = "$temp 千"; }
    }
    $out = "$out$temp";
    $temp = '';
}

return $out;
}

sub zen_num {    # 全角数字→数値
my $out = 0;
my $temp = 0;
my $in = $_[0];
my $length = length($in);
my @zen = ('０', '１', '２', '３', '４', '５', '６', '７', '８', '９');

while ($length--) {
    $in =~ s/^([０１２３４５６７８９])//;
    foreach (1 .. 9) {
        if ($1 eq $zen[$_]) { $out = $out + $_; }
    }
    if ($length) { $out = $out * 10; }
}

return $out;
}

sub kan_num {    # 漢数字→数値
my $out = 0;
my $temp = 0;
my $in = $_[0];
my $length = length($in);
my @kan = ('〇', '一', '二', '三', '四', '五', '六', '七', '八', '九');

while ($length--) {
    $temp = 0;
    if ($in =~ s/^([〇一二三四五六七八九])//) {
        foreach (1 .. 9) {
            if ($1 eq $kan[$_]) { $temp = $_; }
        }
    }
    if ($in =~ s/^十//) { $temp = $temp * 10; if (!$temp) { $temp = 10; } }
    if ($in =~ s/^百//) { $temp = $temp * 100; if (!$temp) { $temp = 100; } }
    if ($in =~ s/^千//) { $temp = $temp * 1000; if (!$temp) { $temp = 1000; } }

    $out = $out + $temp;
}

return $out;
}
```

B.5 kimura.lisp - 法令文全体を入力として，一文ずつ論理式変換処理

```
;;; 木村（s0610030）による追加・修正部分　2007/12/20

;; (kimuratest "nenkinhou.txt" "result.txt") でテスト開始
;;
;; "nenkinhou.txt"は法令文の原文
;; （文字コードは UTF-8,「至つた」等は「至った」等に直しておくこと）
;; "result.txt"は結果の出力ファイル（同名のファイルがあれば上書き）
;;
;; １．まず"nenkinhou.txt"を Perl プログラム"conditions.pl"に通す
;;     すると箇条書きの条件文の処理を行い"conditions.temp"に出力
;; ２．"conditions.temp"は１行に１条文の形式で条文が列挙されている
;;     これを (inputall) で読んで，括弧書き・ただし書き等を削除した上で
;;     １文ずつ論理式変換を行い，まとめて"result.txt"に出力する
;;
;; "result.txt"は,
;;  "入力された文"
;;  "コメントを削除し，括弧書きとただし書きも削除した文（実際に論理式に変換する文）"
;;  "出力された論理式"
;;  ""（空の文字列）
;; の繰り返しになっている

(require :regexp2) ;; 正規表現で括弧書き・ただし書き等の削除に用いる (ACL じゃないと無理)

;; テスト開始
(defun kimuratest (ifname ofname)
  (progn
(run-shell-command (concatenate 'string "/opt/nlp/bin/perl conditions.pl " ifname " > conditions.temp"))
(inputall "conditions.temp" ofname)))

;; 平文ファイル"ifile"から１行ずつ論理式に変換し、"ofile"に結果出力
(defun inputall (ifname ofname)
  (let ((sentence) (sentence2) (lf))
    (with-open-file (ostream ofname :direction :output
    :if-exists :new-version :if-does-not-exist :create)
      (with-open-file (istream ifname :direction :input)
(while (setq sentence (read-line istream nil)) ;; ファイルから１文ずつ読み出す

  (print sentence) (print sentence ostream) ;; 入力した平文１つを出力

  (if (match-re "^;.*$" sentence) ;; ";"で始まる行は無視（エラーメッセージ等）
    (progn (print "") (print "" ostream))
    (progn

;; [] 内はコメントとみなして削除、半角スペースとタブも削除
(setf sentence2 (replace-re sentence "[[].*?]" ""))
(setf sentence (replace-re sentence2 "[  ]" ""))

;; 括弧書きを削除し，さらに最初の一文以降を削除する
;; 括弧書きやただし書き等はいずれは処理する必要があるが，今のところは削除して論理式に変換
(setf sentence2 (replace-re sentence "( ([^( ()() )]+) | ([^() )]+? (.+?) [^( ()]+?) )" ""))
(setf sentence (replace-re sentence2 "。.*$" "。"))

(print sentence) (print sentence ostream);; 入力した平文（削除処理後）を出力

(setf lf (pat-matching (input1-text sentence)))
(print lf) (print lf ostream)

(print "") (print "" ostream)

  ))
)
      )
    )
```

```
  )
)

(defun input1-text (text) ;; 平文のテキストを入力とする
  (let ((sexp2))
    (knp-parse-text text "temp")
    ;(setf sexp2 (del-punctuation (extraction (load-sexp "temp"))))
    (setf sexp2 (extraction (load-sexp "temp")))))

; Error: Supplied vector not large enough. エラー対策 08.01.07 mnakamur
; echo で長文を吐き出すように run-shell-command を実行すると上記エラーが出るため,
; 条文を一旦ファイル (kimura.out) に出してから読込むように修正.
(defun plain-text-write (fname sent)
  (with-open-file (stream fname :direction :output :if-exists :rename :if-does-not-exist :create)
  (format stream "~A" sent)))

(defun knp-parse-text (text outfile) ;; 平文テキストを入力として KNP 出力を得る
  (plain-text-write "kimura.out" text)
  (run-shell-command
   (concatenate 'string "cat kimura.out |/opt/nlp/bin/juman -e |/opt/nlp/bin/knp -sexp > " outfile)))




;(defun kt2 () (inputall "conditions.temp" "conditions.result.txt"))

;(defun printall (fname) ;; ファイルから 1 行ずつそのまま出力するだけ（テスト用）
;  (let ((sentence))
;    (with-open-file (stream fname :direction :input)
;     (while (setq sentence (read-line stream nil)) (print sentence))
;    )
;  )
;)
```