論説

証明責任とその周辺概念の論理プログラミングによる定式化

2006 年 4 月入学
佐藤　健

I. はじめに

本論文では，我々が以前行った論理プログラミングによる証明責任の定式化[1) 2)]を用いて，暫定真実，推定規定，間接反証などの証明責任周辺概念ならびに主張責任などの主張立証にかかわる他の責任概念を定式化できるかを検討する。

裁判における法的推論の主な特徴の一つとして不完全情報下での推論がある。裁判においては，証明しようとしている事実が過去の出来事に起因することが多く，その出来事を完全に再現することは不可能であり，また，被告，原告の情報収集能力は有限であるため完全な情報を得ることは不可能である。

したがって，被告，原告が弁論を尽くしたとしても，判決を得るために必要な事実の存否を確定できない場合があるが，裁判所は当事者の請求に対して応答義務があるために，その場合でも判決をしなければならない。そのような状況を解決するために証明責任のような制度上の工夫がされている。

証明責任とは，当事者の側から捉えて，ある事実が真偽不明のときにその事実の存在ま

1)　Ken Satoh et al., *Formalizing a Switch of Burden of Proof by Logic Programming*, PROCEEDINGS OF THE 1ST JSAI WORKSHOP ON JURIS-INFORMATICS 76 (2007).
（http://research.nii.ac.jp/~ksatoh/papers/jurisin2007.pdf にて入手可能，2009 年 8 月 14 日最終検索。）

2)　Ken Satoh et al., *Abductive Reasoning for Burden of Proof*, PROCEEDINGS OF THE 2ND JSAI WORKSHOP ON JURIS-INFORMATICS 93 (2008).
（http://research.nii.ac.jp/~ksatoh/papers/jurisin2008.pdf にて入手可能，2009 年 8 月 14 日最終検索。）

たは不存在が仮定されて裁判がなされることにより当事者の一方が被る危険ないし不利益のことである[3]。

本論文では，この証明責任及び周辺概念を用いた推論について論理的な解析を行う。証明責任による推論は人間の行う推論であり，論理的な解析を行う利点として以下の三点があげられる。

第一に，論理的な問題と法的な問題を分離することにより法学者は法的な問題に集中できる。論理的な解析をすることにより，形式論理的に解決できる部分かどうかを明らかにして，論理的な問題と法的な問題の切り分けができるのではないかと考える。もし論理的に解決可能であることがわかれば，論理学でのさまざまな知見が利用できる可能性がある。

第二に，暗黙の仮定が明らかになる。たとえば，論理的な解析を行うことにより，証明責任を用いた推論でどのような推論図式を与えたら機械的な推論が可能かが判明する。その場合に今まで考えられてきた推論方法のみでその結果が機械的に導かれない場合には，隠れた推論図式が存在することが明らかになり，証明責任の理解が深まる。また 2 つの図式が法律学的に異なるものであるとされたときに，論理的にみて同じであれば，論理的ではない何らかの前提がそこにあることが判明する。III.3 節や IV.2 節の間接反証の議論を参照されたい。

第三に，論理的な表現により，その論理システムの性質検証が容易になる。たとえば，ある法律の表現が，別の法律の表現と同じであるかどうかについて，論理的な変換を施すことで明らかになる。III.1 節の暫定真実の議論ならびに III.2 節の法律上の推定の議論を参照されたい。

ただし，証明責任を通常の演繹論理で定式化するのは不可能であることに注意されたい。通常の演繹論理では，情報が付け加わっても今まで得た結論が変わることがない。この性質を「単調性」と呼ぶ。通常の演繹論理では，真偽のわからない命題はわからないままであり，情報が付け加わって確実になったときにのみ真偽を確定していくので，そのような単調性が保たれる。しかし，証明責任を用いた裁判上の推論はこれと異なり，ある情報状態で法的命題の真偽が不明な場合にその真偽を無理やり決定してしまうのであるから，その後に，その決定された真偽値と矛盾する事実が判明した場合には，その法的命題の真偽が異なりうる。つまり，証明責任は単調性を持たないのである。したがって，このような推論を演繹論理によって定式化することは困難である。

II. 論理プログラミングによる定式化

そこで非単調性をもつ論理システムが必要となる。人工知能の分野ではそのような非単調性を持つ論理が研究されているが，本論文では論理プログラミング[4]と呼ばれる論理システムによる定式化を選ぶ。なぜならば，それは以下の利点があるからである。第一に，論理プログラミングはルールの形をしていてルールを用いた法的推論に親和性があること，第二に，ある命題が証明できない場合にその命題を偽と仮定する「失敗による否定」[5]というメカニズムを持っており，このメカニズムが証明責任を表現するのに有用であること，である。

さらに，論理プログラミングは名が示すとおりコンピュータプログラムであって，実行が可能であり，原告，被告の証明活動過程をプログラムの実行過程として見ることも可能になる。

論理プログラムは，以下の形式のルールの集合である。「:-」は定義を示すものである。

3) 高橋宏志『重点講義民事訴訟法 上』（有斐閣，2005）457 頁。

4) Robert Kowalski, Logic for Problem Solving (1979).
（http://www.doc.ic.ac.uk/~rak/ にて入手可能，2009 年 8 月 14 日最終検索。）

5) Keith Clark, *Negation as Failure*, in Logic and Databases 293 (Herve Gallaire & Jack Minker eds., 1978).

R :- F1,...,Fn.

ここで R は述語名(引数1,...,引数m) という形式の式（アトムと呼ばれる）であり F1,...,Fn は，各々，述語名(引数1,...,引数m) または not 述語名(引数1,...,引数m) の形式の式である。論理的な意味としては条件部 F1,...,Fn が真であれば結論部 R が真であることを表す。Fi(i=1,...,n) が真であるとは，以下のように定義される。

① Fi が「述語名(引数1,...,引数m)」のとき：
結論部が述語名(引数1,...,引数m) であるルールであって，そのルールの条件部（Fi :- J1,...,Jk というルールの J1,...,Jk のこと）が真であるようなものが少なくとも 1 つ存在することである。

② Fi が「not 述語名(引数1,...,引数m)」のとき：
結論部が述語名(引数1,...,引数m) であるルールのうち条件部が真となるものがないことである。

上の定義でわかるように，論理プログラミングのルールの形式は，法律要件が満たされれば法律効果が発生する，という法律の形に非常によく似ている。さらに証明責任で使われる真偽不明の状態が，論理プログラミングでは，その事実を証明できないことに対応している。このため論理プログラミングは，法律要件→法律効果という推論図式および証明責任の定式化に非常に適していると考えられる。

この考え方に基づき，我々は証明責任の定式化を行ってきた[6]。本論文では，第 2 の論文の定義にしたがう[7]。

まず，ある法律要件 f に対して，現在提出されている証拠により f 成立の心証度が証明度を超えていることを q(f) と表わす。すると，f の存在が確定されることを prove(f) と表したときに，それが成り立つのは f 成立の心証度が証明度を超えているときであるから，これを論理プログラミングのルールで表すと

prove(f) :- q(f).

となる。

これに対して，f に対する証明責任に基づいて f の存在を仮定する場合を考える。この場合というのは，q(f) であるか f が真偽不明のときである。この 2 つのどちらかの状態であることを default(f) と表す。そして，真偽不明の状態を表すために，法律要件の不存在を−f と表し，現在提出されている証拠により f 不存在の心証度が証明度を超えていることを q(-f) と表すことにすると，f が真偽不明であることは「not q(f) かつ not q(-f)」と表すことができる。すると，default(f) は，

default(f) :- q(f).
default(f) :- not q(f), not q(-f).

と定義される[8]。なお，この 2 つのルールは

default(f) :- not prove(-f).

と等価である[9]。そしてこのことは，default(f) を証明する代わりに not prove(-f) を証明すればよいことを表している。したがって，簡単のため以下では default(f) の代わりに not prove(-f) を用いることにする。

例として，消費貸借契約に基づく貸金返還請求訴訟の場合のルールを示す。貸金返還請求権の要件事実は「貸金の返還合意」，「貸金交付」，「弁済期合意」，「弁済期到来」であり，その抗弁の一つとしては「弁済」がある[10) 11)]。すると，抗弁事実は，それが真偽不明である

6) Satoh et al., *supra* note 1, Satoh et al., *supra* note 2.

7) 我々は，2 つの論文において以下で述べる q- 述語に対応する p- 述語を導入したが，2 つの論文の間での p- 述語の定義は異なっている。ここでは混乱を避けるため，Satoh et al., *supra* note 2 の論文の p- 述語を q- 述語と呼ぶことにする。

8) 結論部が同じルールが複数ある場合には，どちらかのルールの条件部が真であれば結論部は真となる（本文①参照）。つまりそれらの条件部は「または」の関係にある。

9) 形式的証明は末尾の補遺 A を参照されたい。

10) 司法研修所編『改訂問題研究要件事実』（法曹会，2007）38-54 頁。

11) 以下では，簡単化のため，抗弁として弁済のみ考えることにする。もし他の抗弁も考慮する場合には，not prove(抗弁事実1), not prove(抗弁事実2),... というようにルールの条件部に付け加える必要がある。

ときは偽であると仮定されるので，この状況を not prove(抗弁事実) と表わすことができる。したがって貸金返還請求権の成立条件は以下のルールで表すことができる [12]。

prove(貸金返還請求権) :-
　prove(貸金返還合意),
　prove(貸金交付),
　prove(弁済期合意),
　prove(弁済期到来),
　not prove(弁済).

本論文では，証明責任の定式化として上記のルールを用いたときに，日本の民事訴訟法上の証明責任にかかわる概念が正しく定式化できるかどうかについて検証する。

III. 証明責任にかかわる概念の定式化

1　暫定真実

暫定真実とは，無条件の推定規定のことである。これは，ある効果の法律要件の不存在の証明責任を，その効果を争う相手方に負わせるための立法技術で，但書で規定するのと同義である [13]。f が暫定真実により推定されるというのは，文言上は無条件であるが，裁判上では，f の不存在が証明されないときに f の存在が認定されるということである。つまり論理プログラミングのルールで表すと，

prove(f) :- not prove(-f).

となる。これが但書を用いたプログラムと等価になることを論理的に示す。

任意の結論 c の法律要件が c1,...,cn 及び f だったとし，f は暫定真実だとする。このことを論理プログラムで表すと

prove(c) :- prove(c1),..., prove(cn), prove(f).
prove(f) :- not prove(-f).
prove(X) :- q(X).

となる。簡単のために prove に関する定義はこれらのみとする。このとき，この論理プログラムを演繹論理式に変換することができる。その変換は完備化（completion）と呼ばれている [14]。その考え方は，prove の :- の右辺の条件部をすべて論理和で結合したものを prove の定義と考え，その定義から外れていることを表す not P を論理否定 $\neg P$ と考えるということである。

したがって，完備化によって上のルール群は

$$prove(X) \leftrightarrow \big((X = c \wedge prove(c1) \wedge \ldots \wedge prove(cn) \wedge prove(f)) \vee (X = f \wedge \neg prove(-f)) \vee q(X)\big)$$

と変換される [15]。そして，これは，以下の式と同値である。

$$prove(X) \leftrightarrow \big((X = c \wedge prove(c1) \wedge \ldots \wedge prove(cn) \wedge \neg prove(-f)) \vee (X = f \wedge \neg prove(-f)) \vee q(X)\big)$$

形式的証明は末尾の補遺 B を参照されたい。これは，

prove(c) :-
　prove(c1),..., prove(cn), not prove(-f).
prove(f) :- not prove(-f).
prove(X) :- q(X).

を完備化したものと一致する。この新しいプログラムの意味は

12) 下のルールの結論部の prove の引数は請求権であって，法律要件ではなく引数の意味が異なるが，便宜上そのような引数も許すとする。

13) 新堂幸司『新民事訴訟法（第 4 版）』（弘文堂，2008）541 頁。

14) Clark, *supra* note 5.

15) 本文の完備化の結果を日本語で記述しようとすれば，X が証明されたというためには（左辺），$X = c$ かつ，$c1$ が証明され，かつ，$c2$ が証明され，…かつ，f が証明されたとき（右辺第 1 項），または，$X = f$ かつ，f の不存在が証明されないとき（右辺第 2 項），または，X の心証度が証明度を超えたときである（右辺第 3 項），ということができる。

c1 かつ…かつ cn ならば c である。ただし f でないときはその限りでない。

であり，暫定真実で分離して書くことと但書に書くこととは等価であることがわかる。

2　法律上の推定

法律上の推定とは，事実 f の代わりに事実 a の証明すれば事実 f を構成要件とする法律効果を得ることである。ただし，事実 f を直接証明することも妨げない。相手方としては，事実 a の証明を妨げる立証をしてもよいが事実 f の不存在の証明をしてもよい[16]。この考え方を論理プログラミングのルールで表すと

```
prove(f) :- prove(a), not prove(-f).
```

となる。

任意の結論 c の法律要件を c1,...,cn 及び f とし，f は前提事実 a によって法律上推定されるとする。

```
prove(c) :- prove(c1),..., prove(cn), prove(f).
prove(f) :- prove(a), not prove(-f).
prove(X) :- q(X).
```

となる。簡単のために prove に関する定義はこれらのみとする。（この論理プログラムをプログラム I と呼ぶことにする）

これは，以下のプログラム（プログラム II と呼ぶ）と等価である。形式的証明は末尾の補遺 C を参照されたい。

```
prove(c) :-
    prove(c1),..., prove(cn),
    prove(f), not prove(-f).
prove(c) :-
    prove(c1),..., prove(cn),
    prove(a), not prove(-f).
prove(f) :- prove(a), not prove(-f).
prove(X) :- q(X).
```

上の論理プログラムの c についての 2 つのルールの意味を一文で述べると以下のようになる。

c1 かつ…cn かつ，a または f のときは c である。ただし f でないときはその限りではない。

つまり，法律上の推定は（技巧的ではあるが）上のような但書に変更することが可能である。

ところで，ローゼンベルクは法律上の推定は真の推定であり[17]，暫定真実は推定ではなく普通の証明責任規定である[18]としている。その理由かどうかははっきり明言していないが，暫定真実に関してはすべて但書に変換できることを示して，証明責任が転換されているだけのことだとしている。これが理由だとすれば，但書に書ける場合には真の推定ではない，ということになるのではないかと考える。とすると，法律上の推定も論理的には上のような但書で表現できるので真の推定ではなく，証明責任の転換技法となるのではないかと考えられる[19]。

3　間接反証

証明責任を負う当事者が，主要事実 A の存在を強く推認させるいくつかの間接事実 (a,b,c) の証明に成功した場合，A につき証明責任を負わない相手方が，a,b,c とは両立する別個の間接事実（d）の存在を証明することによって，a,b,c から A を推認することが誤りであること（推定事実の不存在）を証明し，または少なくとも A の存在につき真偽不明状態に引き戻す証明活動を間接反証とい

16) 新堂・前掲注 13)540 頁。
17) ローゼンベルク（倉田卓次訳）『証明責任論 全訂版（復刻版）』（判例タイムズ社，2001）234 頁。
18) ローゼンベルク・前掲注 17)238 頁。
19) もっとも，新堂・前掲注 13)541 頁は，法律上の推定の場合には証明主題の選択の機能があると言っており，これを，法律上の推定と暫定真実の違いとも見ることもできるだろう。ただ，このことは論理プログラミング上では暫定真実の但書のルールは 1 つであるのに対し法律上の推定の但書のルールは 2 つであることに対応しているに過ぎず，但書に書けるか否かの問題とは次元を異にする。

う。この場合 d の存在について相手方は積極的に確定する必要があり，その限りで証明責任を負う[20]。この定義からすると，間接反証も抗弁も証明図式は非常に類似するものになると考えられる。本節では，この類似性について論理プログラミングを使って検証してみる。

ここでは，倉田卓次判事が間接反証の視点から把握しなおせると主張している[21]例として無断転貸による賃貸借契約解除について取り上げる[22]。最判昭和 41 年 1 月 27 日民集 20 巻 1 号 136 頁は，民法 612 条の無断転貸による解除権の発生について「背信行為と認めるに足りない特段の事情」があるときは解除権は発生しないとし，その「背信行為と認めるに足りない特段の事情」を賃借人側（被告側）で主張・証明責任を負う抗弁事実とした。これに対して倉田判事は，この場合を間接反証として以下のように説明している[23]。すなわち，無断転貸による解除のような事例においては，無断転貸を昇華させた「背信行為」（賃貸人・賃借人間の信頼関係を破壊するような賃借人の行為）を解除権の発生要件とする条文が不文の法規として存在しているとみる。そして，この場合，解除の効果を訴求する賃貸人すなわち原告側に「背信行為」の証明責任があることになるが，無断転貸の事実を，この背信行為という法律要件を証明する重要な間接事実としてみて，この間接事実を原告が立証すれば一応証明をつくしたことにするのである。そして被告がこの無断転貸の事実そのものを攻撃せず，それと両立しうる間接事実で「背信行為とは認めるに足りない特段の事情」を立証した場合には，その間接事実がプラスされて「無断転貸ではあるが背信行為とは認めるに足りない」という判断を導くことにより要件事実としての背信行為が真偽不明の状態となる。倉田判事はこのような被告の立証活動を間接反証ととらえている。

この説明にしたがって，間接反証による本件の推論を論理プログラミングで表現することを考える。通常の要件事実の証明とは別に間接事実 f が証明されることを indirectprv(f) と表し，証拠から f が成立する心証が得られることを indirectq(f) とする。すると，上の説明に対応する論理プログラム（プログラム III と呼ぶ）は，以下のようになると考えられる[24]。

```
prove(契約解除) :- prove(背信行為).
prove(背信行為) :-
    indirectprv(無断転貸),
    not indirectprv(無背信性評価根拠事実).
indirectprv(無断転貸) :-
    indirectq(無断転貸).
indirectprv(無背信性評価根拠事実) :-
    indirectq(無背信性評価根拠事実).
```

これに対して，通説では「無断転貸」や「背信行為とは認めるに足りない特段の事情」を法律要件としてみて，直接，契約解除の効果を導いている。これに対応する論理プログラム（プログラム IV と呼ぶ）は，以下のようになる。

```
prove(契約解除) :-
    prove(無断転貸),
    not prove(無背信性評価根拠事実).
prove(無断転貸) :-
    q(無断転貸).
prove(無背信性評価根拠事実) :-
    q(無背信性評価根拠事実).
```

これらのプログラムの形式的相違は，プログラム III では「背信行為」という命題が導入されていること以外には，述語名の相違のみである。もしプログラム III で「背信行為」を使わず第 2 のルールの prove(背信行為) を直接第 1 のルールの条件部と置き換え，indirectprv → prove, indirectq → q と名前を付け替えれば全く同じプログラムになってし

20) 新堂・前掲注 13)543 頁。
21) 倉田卓次『民事実務と証明論』（日本評論社，1987）247 頁。
22) 高橋・前掲注 3)491 頁以下。
23) 倉田・前掲注 21)223 頁。
24) 評価障害事実については記載を省略する。また，簡単のために無断転貸の中に転貸借契約成立，基づく引き渡し，転貸者の使用収益などの法律要件が含まれているとする。

まう。すなわち形式的な意味としては同値のプログラムである。したがって，これらに違いを見出すとすれば論理的な意味以外の何らかの法的な前提があることになる。実際，倉田判事も（通説か倉田説の）「両者いずれにしても証明責任の分配は変わらないのであるから，ひっきょう理論構成の相違にすぎない，ともいえよう」としている。倉田判事の考え方としては，不文の法規を仮定する本件の場合，「法律要件分類説からみれば法条からはみ出した要件は簡単なスッキリしたものほど扱い易い[25]」と考えてプログラムIIIの構成をとるとしているようである。

私見では，法律要件が明示されていれば，どちらにしろ当事者にとって証明すべきことは同一であるから，どちらの考え方をとっても実質的には違いがないのではないかと考える。また，間接事実か要件事実かの違いを無視するとすれば，倉田判事の「背信行為」という命題の導入は，少なくとも論理学上の知識の整理を図る意味では役に立つといえるのでプログラムIIIの方に若干分があるようにも感じられる。ただしそれは間接反証を導入した効果ではなく，「背信行為」という抽象概念を導入した効果であることに注意されたい。

IV. 主張責任と間接反証との関係

本章では，証明責任の周辺概念として主張責任を導入して，間接反証との関係について述べる。このための準備として，allege述語を新しく導入する。allege(f)は事実fを当事者が狭義の弁論において争点として主張することを表す。このような概念を表す述語を導入することで，民事訴訟における主張責任について議論できることを以下に示す。

1　主張責任と証明責任の分離

日本では，主張責任は証明責任と一致するといわれており，両者の分化が必要ではないといわれている[26]。ただし，ローゼンベルクは主張責任と証明責任の食い違う個別的な場合があるということを述べている[27]。ローゼンベルクは，その例として，「相手方の不履行による解除…の場合の理由づけのためには，債権者は不履行の事実を主張しないことには主張自体が整わない」が，ドイツ民法の明文では，「不作為債務の場合を除き，履行の有無についての証明責任は債務者に属する」ので主張責任と証明責任が異なっていると述べている。これを論理プログラムで表すと以下のようになる。

```
prove(債務不履行解除) :-
    prove(債務の発生原因事実),
    allege(-履行),
    not prove(履行).
prove(X) :- q(X).
```

この場合には，債権者は，履行がされないことを主張すると，債務者が履行を証明しない限り（not prove(履行)に対応），債務不履行解除が成立することになる（prove(債務不履行解除)に対応）[28]。逆にいえば，債権者が，履行がないことを主張しなければ，prove(債務不履行解除)が成立しない。つまり，ここでは，主張責任が果たされないことによって履行があったことを擬制しているとも見ることができる。このことを太田勝造教授は，「主張責任とは，当事者双方が当該規範について事実の主張をなさないときに，要件の存否が擬制されて，当事者の一方の被る不利益ないし危険のことである」と述べている[29]。

また，高橋宏志教授は，合理的理由さえあれば，証明責任と主張責任が一致しない例外

25) 倉田・前掲注21)230頁。
26) 高橋・前掲注3)470頁。
27) ローゼンベルク・前掲注17)61頁，ただし以下の例はドイツ民法における例である。
28) ただし，日本では，履行の主張は抗弁事実となっているため（司法研修所編・前掲注10)51頁），債権者が不履行の主張をする必要はない。したがって，この例は日本の民事上では主張責任と証明責任を分離する例とはならないようである。
29) 太田勝造『裁判における証明論の基礎』（弘文堂，1982）137頁。

を認めてもよいことになろう，とする[30]。たとえば，主張責任と証明責任の分離を許せば，証拠偏在の場合の立証活動の負担の軽減をすることができると考えられる。証拠が一方当事者Aに偏在し，他方当事者Bが要証事実fを証明することが困難な状況下において，Bにfの主張責任を負わせ，Aに¬fの証明責任を負わせることで，立証活動の負担を公平に分割することもできる，と考えられる。このような新たな制度の発想ができるようになることは，allege述語の導入の効果の一つと考えられる。

2　主張責任と間接反証

Ⅲ.3節において，間接反証と通常の抗弁の証明について，証明責任に着目すると違いが生じないことを述べたが，主張責任を導入すると違いが生じる。倉田判事は，「間接反証事実は間接事実であるから，主張責任はない」[31]としていることから，allege述語を導入した論理プログラムでは，間接反証と通常の抗弁の証明の違いを表現することができる。つまり，倉田説からは，背信行為だけが主要事実であるから背信行為があったことを主張すればよいので，以下のプログラムになる。

```
prove(契約解除) :- prove(背信行為).
prove(背信行為) :-
    allege(背信行為),
    indirectprv(無断転貸),
    not indirectprv(無背信性評価根拠事実).
indirectprv(無断転貸) :-
    indirectq(無断転貸).
indirectprv(無背信性評価根拠事実) :-
    indirectq(無背信性評価根拠事実).
```

これに対し，通説によると，請求原因では無断転貸（転貸借契約成立，基づく引き渡し，使用収益）の事実を主張立証すればよく，抗弁として「背信行為に当たらない特段の事情」の主張立証が許される。このような見解からすれば，それらのレベルまで主張が必要なので以下のようなプログラムになる。

```
prove(契約解除) :-
    prove(無断転貸),
    not prove(無背信性評価根拠事実).
prove(無断転貸) :-
    allege(無断転貸),
    q(無断転貸).
prove(無背信性評価根拠事実) :-
    allege(無背信性評価根拠事実),
    q(無背信性評価根拠事実).
```

後者の定式化を用いると，prove(無断転貸)が成立していた場合は，「無背信性評価根拠事実」の証拠が，他の事実の証明の中で提示されていたとしても（つまりq(無背信性評価根拠事実)が成り立っていても），「無背信性評価根拠事実」があることを被告側が主張していなければprove(契約解除)が成立してしまう。これに対して，前者の定式化を用いると，「無背信性評価根拠事実」の証拠が弁論のどこかで提出されていれば，prove(契約解除)が成立しなくなる。したがって，主張責任に関して考慮することによって初めて推論上で間接反証を使う場合と間接反証を使わない場合の違いが生じるということになる。

以上見てきたように，allege述語の導入により，民事訴訟法上でのより詳細な議論ができるようになった。つまり，このように，新たな民事訴訟法上の概念に対応する述語を導入していくことで，論理プログラミングを用いてより精緻な理論を作り上げることができるということである。

V. 関連研究

論理プログラミングで法律を定式化する試みは80年代から始まっている。Sergotら[32]

30) 高橋・前掲注3)471頁。
31) 倉田・前掲注21)249頁。
32) Marek Sergot et al., *The British Nationality Act as a Logic Program*, 29 COMMUNICATIONS OF THE ASSOCIATION FOR COMPUTING MACHINERY 370 (1986).

は，英国国籍法に関して論理プログラミングを用いた定式化を試みており，「失敗による否定」が法的命題の否定を表現するのに適していると主張している。日本では吉野一教授をリーダーとした論理プログラミングをベースにした法律エキスパートシステムプロジェクトが行われた[33]。

本論文は我々の以前の証明責任の定式化[34]を用いて，日本の民事訴訟法上の証明責任にかかわるさまざまな概念の定式化を試みたものである。我々の第一の研究[35]は，人工知能と法律（AI and Law）研究分野でPrakkenが提起した「論理プログラミングによって法律の証明責任を定式化できるか」という問題[36]に対して，肯定的な答えを与えたものである。その本質は，証明責任の考え方を論理プログラミングに変換しなおしたものである。我々の第二の研究[37]は，証明責任を用いた推論において，さらに仮説推論を導入して，足りない要件事実を自動的に推論するものである。この二つの研究は論理プログラミングによる証明責任を用いた推論形式の枠組みを与えたものといえる。

これに対して本論文では，この枠組みにより，日本の民事訴訟法の分野での証明責任に関係する諸概念がどのように定式化できるかを検証している。したがって，本論文は，我々の枠組みが民事訴訟法での証明責任の概念を正しく捉えているかについての検証と見ることもできる。

VI. 結論

本論文の成果は，論理プログラミングによる証明責任の定式化を用いて暫定真実，法律上の推定，間接反証，主張責任などの証明責任の周辺概念を定式化し解析したことである。

今後の研究としては，以下があると考える。第一に，証明責任の分配が論理プログラミングの表現から形式的に導かれるかを検討することがある。現在の感触では，これは大変困難ではないかと思われる。まず，上の無断転貸の例でもみられるように，法文上にないような「背信性不存在の評価根拠事実」のような条件まで証明責任の分配を考えなければならないとなると，そのような条件がどういう解釈に基づき導かれるのかということに答えなければならない。しかし，条件を探索する手掛かりは，現実世界の理解があって初めて得られるものだと思われる。そうなると現実世界がどうあるべきか，または，どうなっているかについて論理的に記述しなければならず，この作業は膨大でほぼ不可能である。さらに，そのような新たな条件がないと仮定したとしても，証明責任の分配には価値判断が入ると考えられる。たとえば，証明責任の議論においては「当事者双方の証明の難易度を考慮し」という表現や「当事者双方の利益を考えて」という表現が見受けられる。このような考慮事項は，結果に関する価値判断に関わっていると考えられ，最初の問題と同じく現実世界の理解が必要となり，同じように困難と思われる。もし，形式的にできるとすれば，一般的条項のうち権利障害，権利消滅または権利阻止の効果を持つものが，個別条項の条件などに書かれていなくても適用される，ということぐらいではないかと推察する[38]。

第二に，証明責任の諸説を論理プログラミングで統一的に表現することで，それらの比

（http://www.doc.ic.ac.uk/~rak/papers/British Nationality Act.pdf にて入手可能，2009 年 8 月 14 日最終検索。）

33) 吉野一編者代表『法律人工知能：法的知識の解明と法的推論の実現』（創成社，2000）。

34) Satoh et al., *supra* note 1, Satoh et al., *supra* note 2.

35) Satoh et al., *supra* note 1.

36) Henry Prakken, *Modelling Defeasibility in Law: Logic or Procedure?*, 48 FUNDAMENTA INFORMATICAE 253 (2001).（http://www.cs.uu.nl/groups/IS/archive/henry/burden.pdf にて入手可能，2009 年 8 月 14 日最終検索。）

37) Satoh et al., *supra* note 2.

38) たとえば，信義則と信頼関係破壊の法理の関係など。

較ができるようにすることが考えられる。本論文では，規範的事実に関して間接反証を用いる説と主要事実としてとらえる説について比較した。同じように他の問題に対しても説がさまざまあったときに，それを定式化して比較できないかを検討したいと考える。ただし，この定式化はあくまで，ある状況下において何が形式的に推論でき，何が推論できないか，ということを明らかにするのみで，その後，その結論が妥当であるのかどうかについては，再び価値判断が入ってしまうので，それ以上の比較は別に行わなければならないことに注意されたい。

第三に，法学教育への応用が考えられる。たとえば，本論文のシステムにおいて原告，被告の主張を一つでも欠いたときにどのような帰結が生じるかの過程を学習者に示すことで，証明責任の分配についての学習者の理解の手助けになるのではないかと考える。さらに，証明責任の分配について争いがあるときに，複数の分配理論を別々に実行させ，結論の違いを示して，どの理論が望ましいのかを学習者に議論させるきっかけになる可能性もある。

謝辞：本論文に関して貴重なご助言をいただきご指導いただきました太田勝造教授，ならびに本論文に関してさまざまな有用なコメントをいただきました岡崎克彦判事に感謝いたします。

補遺 A. default(F) が not prove(-F) であることの証明

default(F) :- q(F).
default(F) :- not q(F), not q(-F).
に対して完備化[39]を行うと，

$$default(F) \leftrightarrow \big(q(F) \vee \big(\neg q(F) \wedge \neg q(-F)\big)\big)$$

となる。これは以下と同値である。

$$default(F) \leftrightarrow \big(\big(q(F) \wedge q(-F)\big) \vee \big(q(F) \wedge \neg q(-F)\big) \vee \big(\neg q(F) \wedge \neg q(-F)\big)\big)$$

$q(F) \wedge q(-F)$ はいつでも不成立であるから除くと

$$default(F) \leftrightarrow \big(\big(q(F) \wedge \neg q(-F)\big) \vee \big(\neg q(F) \wedge \neg q(-F)\big)\big)$$

となり，結局

$$default(F) \leftrightarrow \neg q(-F)$$

となる。これは，すなわち

$$default(F) \leftrightarrow \neg prove(-F)$$

である。この論理式は，
default(F) :- not prove(-F).
という 1 つのルールの完備化と一致するため，最初の 2 つのルールと上記 1 つのルールは同値となる[40]。

補遺 B. 暫定真実における 2 つの論理式の同値性の証明

$$prove(X) \leftrightarrow \big(\big(X = c \wedge prove(c1) \wedge \ldots \wedge prove(cn) \wedge prove(f)\big) \vee \big(X = f \wedge \neg prove(-f)\big) \vee q(X)\big)$$

が，

$$prove(X) \leftrightarrow \big(\big(X = c \wedge prove(c1) \wedge \ldots \wedge prove(cn) \wedge \neg prove(-f)\big) \vee \big(X = f \wedge \neg prove(-f)\big) \vee q(X)\big)$$

と同値であることを証明する。
第一式の X に f を代入すると

$$prove(f) \leftrightarrow \big(\neg prove(-f) \vee q(f)\big)$$

となり，X に $-f$ を代入すると

$$prove(-f) \leftrightarrow q(-f)$$

となるので，これを $prove(f)$ の $\leftrightarrow$ の右辺で

39) 完備化の説明は Ⅲ.1 節を参照されたい。
40) このルールの意味は，F が証明責任によって真と仮定されるのは，-F が証明できないときであるということである。このことは，証明責任を用いた法命題の充足を表現する場合には，ノンリケットの表現は不要であることを示しているといえる。この指摘は太田勝造教授による。

置き換えると，

$$prove(f) \leftrightarrow \big(\neg q(-f) \vee q(f)\big) \quad (1)$$

となる。

述語qの意味から$q(f) \wedge q(-f)$がいつでも偽となるため，ド・モルガンの法則により$\neg q(f) \vee \neg q(-f)$が恒真となる。この恒真式を (1) 式の右辺に$\wedge$で結びつければ，右辺は$\neg q(-f)$と同値となる。したがって，(1) 式は，

$$prove(f) \leftrightarrow \neg q(-f)$$

となる。一方$q(-f)$は$prove(-f)$と同値であるから，結局，

$$prove(f) \leftrightarrow \neg prove(-f)$$

となる。したがって，第一式の中の$prove(f)$を$\neg prove(-f)$に置き換えて，第二式を得る。

補遺 C. プログラム I とプログラム II の等価性の証明

プログラム I を再掲する。

```
prove(c) :- prove(c1),..., prove(cn), prove(f).
prove(f) :- prove(a), not prove(-f).
prove(X) :- q(X).
```

完備化によって上記プログラムを以下の論理式に変換することができる。

$$prove(X) \leftrightarrow \big(\big(X = c \wedge prove(c1) \wedge \ldots \wedge prove(cn) \wedge prove(f)\big) \vee \big(X = f \wedge prove(a) \wedge \neg prove(-f)\big) \vee q(X)\big)$$

と変換される。

Xにfを代入すると

$$prove(f) \leftrightarrow \big(prove(a) \wedge \neg prove(-f)\big) \vee q(f)$$

となり，Xに$-f$を代入すると

$$prove(-f) \leftrightarrow q(-f)$$

となるので，これを$prove(f)$の$\leftrightarrow$の右辺で置き換えると，

$$prove(f) \leftrightarrow \big(prove(a) \wedge \neg q(-f)\big) \vee q(f)$$

となる。この式は，

$$prove(f) \leftrightarrow \big(prove(a) \vee q(f)\big) \wedge \big(\neg q(-f) \vee q(f)\big)$$

と変形でき，このときqの意味から$q(-f) \wedge q(f)$は恒偽であるから，$\neg q(-f) \vee q(f)$は$\neg q(-f)$と同値となるため（補遺 B 参照），

$$prove(f) \leftrightarrow \big(prove(a) \vee q(f)\big) \wedge \neg q(-f)$$

となる。すると，$prove(X)$は，

$$prove(X) \leftrightarrow \big(\big(X = c \wedge prove(c1) \wedge \ldots \wedge prove(cn) \wedge \big(prove(a) \vee q(f)\big) \wedge \neg q(-f)\big) \vee \big(X = f \wedge prove(a) \wedge \neg prove(-f)\big) \vee q(X)\big) \quad (1)$$

となる。次にプログラム II について考える。

```
prove(c) :-
    prove(c1),..., prove(cn),
    prove(f), not prove(-f).
prove(c) :-
    prove(c1),..., prove(cn),
    prove(a), not prove(-f).
prove(f) :- prove(a), not prove(-f).
prove(X) :- q(X).
```

この論理プログラムの完備化を考えると以下になる。

$$prove(X) \leftrightarrow \big(\big(X = c \wedge prove(c1) \wedge \ldots \wedge prove(cn) \wedge prove(f) \wedge \neg prove(-f)\big) \vee \big(X = c \wedge prove(c1) \wedge \ldots \wedge prove(cn) \wedge prove(a) \wedge \neg prove(-f)\big) \vee \big(X = f \wedge prove(a) \wedge \neg prove(-f)\big) \vee q(X)\big)$$

これは以下のように簡単化される。

（$prove(c)$の 2 つのルールの共通項でくくる）

$$prove(X) \leftrightarrow \big(\big(X = c \wedge prove(c1) \wedge \ldots \wedge prove(cn) \wedge \big(prove(f) \vee prove(a)\big) \wedge \neg prove(-f)\big) \vee \big(X = f \wedge prove(a) \wedge \neg prove(-f)\big) \vee q(X)\big)$$

Xにfを代入すると

$$prove(f) \leftrightarrow \big(\big(prove(a) \wedge \neg prove(-f)\big) \vee q(f)\big)$$

であるから，$prove(f)$を右辺で置き換えると，$prove(X)$の選言部は以下のようになる。

$$
\begin{aligned}
&prove(X) \leftrightarrow \\
&\quad \big(\big(X = c \wedge prove(c1) \wedge \ldots \wedge prove(cn) \\
&\qquad \wedge \big(\big(prove(a) \wedge \neg prove(-f)\big) \vee q(f) \\
&\qquad\quad \vee prove(a)\big)\big) \\
&\qquad \wedge \neg prove(-f)\big) \\
&\quad \vee \big(X = f \wedge prove(a) \wedge \neg prove(-f)\big) \\
&\quad \vee q(X)\big)
\end{aligned}
$$

これを変形して（$prove(a)$についての吸収律），$prove(-f) \leftrightarrow q(-f)$ を用いると以下の式になる。

$$
\begin{aligned}
&prove(X) \leftrightarrow \\
&\quad \big(\big(X = c \wedge prove(c1) \wedge \ldots \wedge prove(cn) \\
&\qquad \wedge \big(q(f) \vee prove(a)\big) \wedge \neg q(-f)\big) \\
&\quad \vee \big(X = f \wedge prove(a) \wedge \neg prove(-f)\big) \\
&\quad \vee q(X)\big)
\end{aligned} \tag{2}
$$

これは (1) と一致するので，プログラムⅠとプログラムⅡは論理的に等価となる。

（さとう・けん）