シーケント計算に基づく証明系と法的推論への適用

長田 博泰

シーケント計算に基づく証明系の実現とその法的推論への適用について述べる．証明系はシーケント計算を後向きに効率よく実行し，移植，拡張が容易なシステムである．推論を１ステップずつ対話的，あるいは自動的に実行することができる．このシステムを法的推論の論理解析に適用し，自然言語表現の論理式への変換の適切性および推論の整合性を確認する上で有効であることを示す．

１．はじめに

著者は，最近，社会情報過程における価値的情報と論理過程の相互連関への関心から，法律的論理，とくに最高裁大法廷判決において法的価値判断が鋭く対立する判決文の解析を行ってきた．その過程で論理分析および推論の道具として証明系の必要性を痛感し，試作することにした．

最近の証明系はビジュアル環境で動作し，豊富なユーザ支援機能を有する相当規模の大きなものも少なくない［例えば，Bonart and Sufrin, 1999］．しかし，マンパワー，開発期間，目的および可搬性を考慮して，柔軟性に富むコンパクトなシステムを開発した．

証明システムの実現方法の一つとして，論理型言語 Prolog で用いられている導出原理（Resolution Principle）がある．これは等号を含む古典論理を制限した形で実現している．すなわち，ホーン節のみを扱い，論理と制御が絡み合っているので，一階述語論理をそのまま表現することはできない．また，数学的帰納法を含む証明に適さず，論理と手順が一体になっているので基礎となる論理体系の変更も容易ではない．

これらを考慮し，自動的証明と利用者による証明への介入が可能で，しかも種々の論理を扱えるものにすることにした．古典一階述語論理の証明システムの中でシーケント計算の自動化が最も容易なので，このアプローチを採用した．

以下，２でシーケント計算とそれに基く証明法の要点を解説する．３で証明系実現の設計方針を述べ，４で証明系実現の上で考慮した点を技術的細部に立ち入らずに説明する．最後に，法的推論への証明系の適用可能性について事例を用いて考察する．１つは，法規を論理式化し，法規と事例の整合性を問う．もう１つは，判決文の論理化によりその論理的特徴を指摘する．

２．一階述語論理シーケント計算概要

シーケント計算の理解に必要な最小限の事項，証明方法および証明系を作成する上で考慮しなければならないことを簡潔に述べる［例えば，萩谷，1994］．

NAGATA Hiroyasu 札幌学院大学社会情報学部

2.1 シーケント計算

シーケントはつぎの形をとる：

$\phi 1, \cdots, \phi m \vdash \psi 1, \cdots, \psi n$

ここで，$\phi 1, \cdots, \phi m$ および $\psi 1, \cdots, \psi n$ は論理式のマルチセット（多重集合）である．マルチセットとは順序に無関係に要素の重複を許す集合である．通常，シーケントは順序のある論理式の並びで，論理式を交換するなどの，いわゆる構造規則を必要とするが，マルチセットを採用すると，この規則を必要としない．

シーケントは，つぎの論理式と同じ意味を有する：

$\phi 1 \wedge \cdots \wedge \phi m \rightarrow \psi 1 \vee \cdots \vee \psi n$

$\vdash \psi$ は ψ と同じ意味である．シーケントは論理式ではない，したがって，$\vdash$ 記号は論理記号ではない．

推論規則を簡便に表現するため，論理式のマルチセットを Γ および Δ で表す．コンマはマルチセットの和集合を意味する．Γ, Δ は Γ と Δ の和集合である．マルチセットが許されるところに書かれた論理式（例えば，$\Gamma \vdash \phi$ の ϕ）は一個の論理式からなる（シングルトン）マルチセットである．Γ, ϕ は ϕ が出現するマルチセットを表す．

妥当なシーケントとは全ての構造と付値のもとで真であるシーケントである．

シーケントの$\vdash$の右側，左側に同じ論理式が現れるなら，そのシーケントは基本であるという．これは公理とみなすことができる：

$\phi, \Gamma \vdash \Delta, \phi$

シーケント計算の規則は$\vdash$の左側，右側に論理演算子を導入する規則が対になっている．例えば，規則$\wedge$－左は$\vdash$記号の左側に論理積$\wedge$を，規則$\wedge$－右は$\vdash$記号の右側に論理積$\wedge$を導入する．各規則の横線の上のシーケントを前提，下のシーケントを結論という．表1に命題論理のシーケント規則を示す．

表1　命題論理のシーケント規則

左	右
$\dfrac{\phi, \psi, \Gamma \vdash \Delta}{\phi \wedge \psi, \Gamma \vdash \Delta}$	$\dfrac{\Gamma \vdash \Delta, \phi \quad \Gamma \vdash \Delta, \psi}{\Gamma \vdash \Delta, \phi \wedge \psi}$
$\dfrac{\phi, \Gamma \vdash \Delta \quad \psi, \Gamma \vdash \Delta}{\phi \vee \psi, \Gamma \vdash \Delta}$	$\dfrac{\Gamma \vdash \Delta, \phi, \psi}{\Gamma \vdash \Delta, \phi \vee \psi}$
$\dfrac{\Gamma \vdash \Delta, \phi \quad \psi, \Gamma \vdash \Delta}{\phi \rightarrow \psi, \Gamma \vdash \Delta}$	$\dfrac{\phi, \Gamma \vdash \Delta, \psi}{\Gamma \vdash \Delta, \phi \rightarrow \psi}$
$\dfrac{\phi, \psi, \Gamma \vdash \Delta \quad \Gamma \vdash \Delta, \phi, \psi}{\phi \Leftrightarrow \psi, \Gamma \vdash \Delta}$	$\dfrac{\phi, \Gamma \vdash \Delta, \psi \quad \psi, \Gamma \vdash \Delta, \phi}{\Gamma \vdash \Delta, \phi \Leftrightarrow \psi}$
$\dfrac{\Gamma \vdash \Delta, \phi}{\neg \phi, \Gamma \vdash \Delta}$	$\dfrac{\phi, \Gamma \vdash \Delta}{\Gamma \vdash}$

2.2 シーケント計算による証明

推論規則は一般に前向き，すなわち前提から結論に至る過程とみなされる．例えば，前提 $\Gamma \vdash \Delta, \phi$ と $\Gamma \vdash \Delta, \psi$ に$\wedge$－右を適用し，$\Gamma \vdash \Delta, \phi \wedge \psi$ を導く．このシーケントに他の規則を適用し別の結論を得るという過程を繰り返す．例えば，シーケント $\phi \wedge \psi \vdash \psi \wedge \phi$ の証明は以下のとおりである：

$$\dfrac{\dfrac{\phi, \psi \vdash \psi \qquad \phi, \psi \vdash \phi}{\phi, \psi \vdash \psi \wedge \phi}\ \wedge\text{－右}}{\phi \wedge \psi \vdash \psi \wedge \phi}\ \wedge\text{－左}$$

しかし，前向きの読み方は，与えられたシーケントの証明を遂行する助けになり難い．証明を見出すには，規則を後向き，すなわち，

ゴール（結論）からサブゴール（前提）をながめた方がよい．例えば，∧－右はゴール$\Gamma \vdash \Delta, \phi \wedge \psi$に対し，サブゴール$\Gamma \vdash \Delta, \phi$と$\Gamma \vdash \Delta, \psi$を返すとみなす．このサブゴールが証明されたなら，ゴールが証明できることになる．残りのサブゴールが基本シーケントに達するまでこの過程を繰り返す．後向きに見ると，上の証明は，証明されるべきシーケント$\phi \wedge \psi \vdash \psi \wedge \phi$から始まる．

このシーケントは∧－左によって$\phi, \psi \vdash \psi \vdash \phi$になる．このサブゴールは∧－右によって$\phi, \psi \vdash \psi$と$\phi, \psi \vdash \phi$になる．これらのサブゴールは基本シーケントであり，証明が完了する．命題論理では，この手順は必ず終了する．

シーケントは，論理式の分解に応じて証明が異なる．上の証明で最初に$\phi \wedge \psi \vdash \psi \wedge \phi$の左側の論理積を分解するとしよう．

$$\dfrac{\dfrac{\phi, \psi \vdash \psi}{\phi \wedge \psi \vdash \psi}\ \wedge\text{－左} \qquad \dfrac{\phi, \psi \vdash \phi}{\phi \wedge \psi \vdash \phi}\ \wedge\text{－左}}{\phi \wedge \psi \vdash \psi \wedge \phi}\ \wedge\text{－右}$$

この証明は∧－左が二度現われるという意味で上の証明より大きいといえる．各ステップでより少ない個数のサブゴールを選ぶと結果としてより短い証明を得る．

以上の考察からつぎの証明手順を得る：

- 初期ゴールとして証明されるべきシーケントをとる．
- 証明木の葉を選択し，これに規則を適用し，葉を置きかえる．
- すべての葉が基本シーケントである（成功）か，どの推論規則も葉に適用できない（失敗）とき停止する．

シーケントの証明に失敗したとき，失敗の情報からそのシーケントを偽にする情報を得ることができる．シーケントの証明を試みた結果，

$\Gamma \vdash \Delta$

となり，Γ，Δが同じ命題を含まないとき，Γの各命題を真に，Δの各命題を偽にすれば，元のシーケントを偽にすることができる．なぜなら，推論規則の前提のどれか一つが偽になるなら，その推論規則の結論も偽になるからである．

例えば，$\phi \vee \psi \rightarrow \phi$という論理式を考える．

$$\dfrac{\dfrac{\phi \vdash \phi \qquad \underline{\psi \vdash \phi}}{\phi \vee \psi \vdash \phi}\ \vee\text{－右}}{\vdash \phi \vee \psi \rightarrow \phi}\ \rightarrow\text{－右}$$

下線部のシーケントは基本ではない．ψを真，ϕを偽にすれば，$\psi \vdash \phi$は偽になる．すると，$\phi \vee \psi \vdash \phi$は偽になる．最終的に，$\phi \vee \psi \rightarrow \phi$が偽になる．

2.3 量化子のシーケント計算

命題論理は決定可能，すなわち，上の証明手順で任意の論理式が定理であるか否かを有限時間内に決定することができる．量化子を含むと，そのような手順は存在しない．

以下の説明で，$\phi[t/x]$はϕのxの自由な出現にtを代入して得られる論理式を表すものとする．

全称量化子には二つのシーケント規則がある：

$$\dfrac{\phi[t/x],\ \forall x.\ \phi,\ \Gamma \vdash \Delta}{\forall x.\ \phi \vdash \Delta}\ \forall\text{－左}$$

$$\dfrac{\Gamma \vdash \Delta,\ \phi}{\Gamma \vdash \Delta,\ \forall x.\ \phi}\ \forall\text{－右}$$

変数条件：xは結論に自由出現してはならない

∀－左規則は容易に納得できる．$\forall x.\ \phi$が真なら，xの自由な出現を任意の項tで置き換えた$\phi[t/x]$も真であるからである．

より複雑な∀－右の正当性を，前提が正し

いと仮定し，結論も正しいことを示す．与えられた構造と付値のもとでΓの論理式が真で，Δのどれも真でないとする．このとき，$\forall x.\ \phi$ が真であることを示さなければならない．x以外の変数をそのままとし，xのあらゆる付値に対し ϕ が真であることを示せばよい．∀－右の変数条件によって，xの値を変えてもΓあるいはΔの論理式の真偽に影響しない．前提が妥当であるから，ϕ は真でなければならない．

変数条件を無視すると結果は正しくない．つぎの推論において，xに0を割当てると結論は正しくない．

$x=0 \vdash x=0$

$\dfrac{}{x=0 \vdash \forall x.x=0}$ ∀－右 ？？？

存在量化子にも二つのシーケント規則がある：

$$\frac{\phi,\ \Gamma \vdash \Delta}{\exists x.\ \phi,\ \Gamma \vdash \Delta}\ \exists－左$$

$$\frac{\Gamma \vdash \Delta,\ \exists x.\ \phi,\ \phi[t/x]}{\Gamma \vdash \Delta,\ \exists x.\ \phi}\ \exists－右$$

変数条件：xは結論に自由出現してはならない

∀－左および∃－右規則は他の規則にない性質を有する，すなわち，後向きの証明において，この規則はゴールから論理式を取り去ることはなく，量化された論理式を項をそのボディに代入した論理式に展開する．論理式を繰返し展開することができる．証明に必要な量化された論理式を何回展開するかを予め決定することは不可能である．このため，証明手順が終了しないことがあり得る．つまり，1階論理は決定不可能である．

2.4 量化子をもつ定理の証明

量化子を有する論理式に対し，余計な探索を必要としないので後向きの証明手順が有効である．全称量化子を含んだ簡単な例で説明する：

$$\frac{\phi(x),\ \forall x.\ \phi(x) \vdash \phi(x),\ \psi(x)}{\forall x.\ \phi(x) \vdash \phi(x),\ \psi(x)}\ \forall－左$$

$$\frac{\forall x.\ \phi(x) \vdash \phi(x),\ \psi(x)}{\forall x.\ \phi(x) \vdash \phi(x) \vee \psi(x)}\ \vee－右$$

$$\frac{\forall x.\ \phi(x) \vdash \phi(x) \vee \psi(x)}{\forall x.\ \phi(x) \vdash \forall x.\ \phi(x) \vee \psi(x)}\ \forall－右$$

xは結論において自由ではないので，∀－右の変数条件を満足している．後向き証明手順では，この結論が開始のゴールである．最初に∀－左を適用し，論理式 $\phi(x)$ を挿入すると，その結果得られるサブゴールでxは自由変数である．その場合，量化された変数を名前替えせずに∀－右を適用することはできない：

$$\frac{\dfrac{\phi(x),\ \forall x.\ \phi(x) \vdash \phi(y),\ \psi(y)}{\phi(x),\ \forall x.\ \phi(x) \vdash \phi(y) \vee \psi(y)}\ \vee－右}{\phi(x),\ \forall x.\ \phi(x) \vdash \forall x.\ \phi(x) \vee \psi(x)}\ \forall－右$$

$$\frac{\forall x.\ \phi(x) \vdash \quad \forall x.\ \phi(x) \vee \psi(x)}{}\ \forall－左$$

先頭のシーケントは基本ではない．証明を完成するには再度∀－左を適用しなければならない．しかし，そうしても証明を完成することはできない．このことから，他に適用できる規則があるなら，∀－左あるいは∃－右規則を適用すべきでないことがわかる．

つぎの例は量化子にともなって発生する問題点を示している．

$$\frac{\phi(x),\ \phi(z) \vdash \exists z.\ \phi(z) \rightarrow \forall x.\ \phi(x),\ \phi(x),\ \forall x.\ \phi(x)}{\phi(z) \vdash \exists z.\ \phi(z) \rightarrow \forall x.\ \phi(x),\ \phi(x),\ \phi(x) \rightarrow \forall x.\ \phi(x)}\ \rightarrow－右\ \ \exists－右$$

$$\frac{\phi(z)\vdash\exists z.\ \phi(z)\rightarrow\forall x.\ \phi(x),\ \phi(x)}{\phi(z)\vdash\exists z.\ \phi(z)\rightarrow\forall x.\ \phi(x),\ \forall x.\ \phi(x)}\ \forall\text{－右}$$

$$\frac{\phi(z)\vdash\exists z.\ \phi(z)\rightarrow\forall x.\ \phi(x),\ \forall x.\ \phi(x)}{\vdash\exists z.\ \phi(z)\rightarrow\forall x.\ \phi(x),\ \phi(z)\rightarrow\forall x.\ \phi(x)}\ \rightarrow\text{－右}$$

$$\frac{\vdash\exists z.\ \phi(z)\rightarrow\forall x.\ \phi(x),\ \phi(z)\rightarrow\forall x.\ \phi(x)}{\vdash\exists z.\ \phi(z)\rightarrow\forall x.\ \phi(x)}\ \exists\text{－右}$$

ゴールから上向きにたどって，∃－右を適用し，自由変数zを導入する．存在記号をもつ論理式がサブゴールにあるが，それは他の規則が適用不可能になるまで休眠させる．つぎの推論，→－右が$\phi(z)$を左辺に移動する．サブゴールでxは自由でないから，∀－右を適用し，$\forall x.\ \phi(x)$を$\phi(x)$に置換える．そのサブゴールで再び∃－右を適用し，zにxを代入する．→－右適用後のサブゴールは，両辺に$\phi(x)$を有するので最終ゴールである．

量化子の規則，とくに∃－右および∀－左を適用する際に規則中の項tをどのように選ぶかが問題になる．これはどの項が最終的に基本サブゴールを生成し，証明を完成させるかを予想することである．上の証明で，最初の∃－右でzは任意に選択してよいが，2番目の∃－右ではxを選ばなければならない．

∃－右および∀－左の項の選択を引き延ばすことができる．そのために項の場所を示すメタ変数？a，？b，…を導入する．メタ変数に適当な項を代入してゴールを得ると，証明過程全体にこの代入をおこなう．例えば，サブゴールP(？a)，Γ⊢Δ，P(f(？b))は？aをf(？b)で置換えるなら基本シーケントになる．？aは外側のf(…)しか確定していないが，次第に確定する．適当な代入の決定過程－ユニフィケーション(単一化，Unification)－が量化子に関する自動的推論の鍵である．

メタ変数を用いると，規則∀－左はつぎのようになる．ここで，？aはメタ変数である．

$$\frac{\phi[?a/x],\ \forall x.\ \phi,\ \Gamma\vdash\Delta}{\forall x.\ \phi,\ \Gamma\vdash\Delta}\ \forall\text{－左}$$

メタ変数の導入によってつぎの問題が発生する．∃－右および∀－左には「xは結論で自由であってはならない」という変数条件がある．結論が任意の項で置換することができるメタ変数を含むとき，この条件をどのように扱うべきか？ここでは，自由変数に禁止されるメタ変数のリストを目印としてもたせることにする．自由変数$b_{?a1,\cdots,?ak}$はメタ変数？a1，…，？akに代入される項に含まれてはならないこと意味する．ユニフィケーションアルゴリズムではこれを守らなければならない．

以上を考慮して，目印の付けられた自由変数をパラメータ，メタ変数を単に変数と呼ぶことにする．

パラメータを用いると，∀－右規則は以下のようになる．

$$\frac{\Gamma\vdash\Delta,\ \phi[b_{?a1,\cdots,?ak}/x]}{\Gamma\vdash\Delta,\ \forall x.\ \phi}\ \forall\text{－右}$$

変数条件：bは結論に出現してはならない．？a1，…，？akは結論ですべて変数でなければならない．

変数条件の前半はパラメータbが既に使用されていないことを保証する．後半はbが後で代入によって紛れ込まないことを保障する．∃－左についても同様である．

パラメータは量化子の正しい推論を保証する．例えば，一般に，$\forall x.\ \phi(x,\ x)$は$\exists y.\ \forall x.\ \phi(x,\ y)$を含意しない．その証明の以下の試みを検討しよう．

$$\frac{\phi(?c,\ ?c),\ \forall x.\ \phi(x,\ x)\vdash\exists y.\ \forall x.\ \phi(x,\ y),\ \phi(?b,\ ?a)}{\forall x.\ \phi(x,\ x)\vdash\exists y.\ \forall x.\ \phi(x,\ y),\ \phi(?b,\ ?a)}\ \forall\text{－左}$$

$$\frac{\forall x.\ \phi(x,\ x)\vdash\exists y.\ \forall x.\ \phi(x,\ y),\ \phi(?b,\ ?a)}{}\ \forall\text{－右}$$

$\forall x.\ \phi(x, x) \vdash \exists y.\ \forall x.\ \phi(x, y)$, $\forall x.\ \phi(x, ?a)$

——————————————— ∃－右

$\forall x.\ \phi(x, x) \vdash \exists y.\ \forall x.\ \phi(x, y)$

先頭のシーケントを基本にすることはできない．$\phi(?c, ?c)$と$\phi(?b, ?a)$を一致させるために，?cと?aをb?aで置換しなければならない．しかし，パラメータb?aは?aに代入される項に出現することは禁じられている．∀－右および∃－左を適用して証明を続けても，基本シーケントは生成されないであろう．

上と比較するために，$\forall x.\ \phi(x, x)$が$\forall x.\ \exists y.\ \phi(x, y)$を含意することを証明しよう．

$\phi(?c, ?c)$, $\forall x.\ \phi(x, x) \vdash \exists y.\ \phi(a, y)$, $\phi(a, ?b)$

——————————————— ∀－左

$\forall x.\ \phi(x, x) \vdash \exists y.\ \phi(a, y)$, $\phi(a, ?b)$

——————————————— ∀－右

$\forall x.\ \phi(x, x) \vdash \exists y.\ \phi(a, y)$

——————————————— ∃－右

$\forall x.\ \phi(x, x) \vdash \forall x.\ \exists y.\ \phi(x, y)$

$\phi(?c, ?c)$と$\phi(a, ?b)$を$\phi(a, a)$に変換して証明を完成することができる．－右によってゴールにもち込まれるものはないから，パラメータaは変数による目印は付かない．

3．証明系設計方針

証明系の設計・開発に当たって，マンパワーと開発時間，目的，可搬性，拡張性等を考慮し，以下の方針で臨んだ：

1）著者一人で開発し，できる限り短期間で一応動作するものを実現したい．

2）研究に利用するだけで，教育に利用することは考えない．論理について理解していることを前提とし，教育的配慮からの支援・補助機能は考えない．

3）共同研究者間での利用を考慮し，移植が容易でなければならない．証明系開発に適した言語(例えば，Lisp，Prolog等)ではなく，手続き言語，実際にはC/C＋＋言語で開発する．

4）拡張等に対応できる柔軟なシステムが望ましい．C/C＋＋言語を採用したので，リスト構造等の処理から記述しなければならない反面，少しプログラミングを知っていれば，拡張等が比較的容易になる．

上述の設計方針に加えて，証明系の適用可能性・拡張性に考慮した．はじめから特殊目的，例えば，実務上の法的推論等への支援だけを目指すのであれば，法的推論のエキスパートシステムなどを設計することも考えられる．しかし，法律の実務家でも専門家でもない著者の法的論理への関心はそこにはなく，法的論理を解析し，法的論理の特徴を明らかにすることにある．その場合，通常の古典論理ばかりでなく，必要に応じて，特殊な論理を仮定することもあり得る．また，計算への線型論理の適用，あるいは自然言語意味論への型論理の適用にも有用な証明系の開発を目指した．

4．証明系の構成

シーケント計算による証明系はつぎの2つの部分に分けて考えるとことができる．1つは論理式を読み，後の処理に適した形に変換する，いわゆるパーザー部である．もう一つは内部形式に変換された論理式に証明手順，すなわち後向きのシーケント計算を対話的，あるいは自動的に適用する部分である．後者は多くの処理を含むが，つぎの3つが証明系作成の上で基本となる：

① ユニフィケーション

② 推論規則の適用方法

③ 証明の(半)自動化

以下，これらについてその概要をのべる．システムの実現には関数型言語MLで書かれた証明系HALを参考にした［Paulson, 1991］.

4.1 一階述語論理の表現

通常用いられる論理記号をそのまま入力するのは,煩わしいので以下の記法を採用する.全称量化子∀,存在量化子∃をそれぞれALL, EXで表す.また,論理演算子はつぎのとおりとする:

通常記号:¬ ∧ ∨ → ⇔
入力記号:~ & | −−> <−−>

例えば,論理式∃z.(φ(z) → ∀ x.φ(z))は以下のように表現する:

EX z.(P(z) −−> (ALL x.P(x)))

入力する論理式の構成規則を拡張BNFで示す:

```
論理式::=ALL 名前 . 論理式
        | EX 名前 . 論理式
        | 論理式 論理演算子 論理式
        | 原始論理式
原始論理式::=~ 原始論理式
            | (論理式)
            | 名前 引き数
引き数::= (項並び)
        | 空
項並び::= 項 {, 項}*
項::= 名前 引き数
    | ? 名前
```

ただし,{ }* は0回以上の繰返しを表す.

4.2 証明手順の実現

2で説明したシーケント計算は各ノードがシーケントである木として表すことができる.したがって,シーケント計算の証明手順はゴールから後向きに順に働き,証明木はルートから成長する.この手順を実現するには部分的に構成された証明木を表現するデータ構造−これを証明状態と呼ぶ−を必要とする.推論規則を証明状態に作用する関数として実現する.

4.2.1 ユニフィケーションアルゴリズム

証明の過程でゴールの原始論理式にユニフィケーションを行う.その基本アルゴリズムは項の対に対し,2つを一致させるような変数と項の対からなる置換の集合を見出すか,あるいはユニファイできないことを通知することである.その他の場合の処理は以下のとおりである:

関数適用:2つの関数適用は関数名が一致する場合のみユニファイ可能である.明らかにf(?a)とg(b, ?a)を一致させることはできない.g(t1, t2)をg(u1, u2)と一致させるにはt1とu1およびt2とu2を同時に一致させなければならない.したがって,g(?a, ?a)はg(b, c)を一致させることはできない.なぜなら,変数(?a)を同時に異なる2つの定項(bとc)に一致させることはできないからである.

パラメータ:2つのパラメータは同じ名前であるときだけユニファイ可能である.パラメータは関数適用とユニファイできない.

変数:問題となるのは変数?aと項tのユニファイである.?aがtに出現しないなら,ユニフィケーションは成功し,置換(?a, t)を返す.?aがtに出現するなら,ユニフィケーションは失敗である.つぎの2つの理由があり得る:

?aがtのパラメータに出現するなら,?aはそのパラメータおよび項に対する禁止された変数である.?aをtで置換するとある量化子規則の変数条件に違反することになる.

tが真に?aを含むなら,項はそれ自身を含むことはできないから,ユニファイできない.例えば,f(?a)と?aを同一の項に変換する置換はない.

これがよく知られた出現検査(occurs check)である.Prologインタプリターでは負荷が大きいのでこの検査を無視する.しかし,定理証明では健全性が効率に優先するので,出現検査を行わなければならない.

例1:g(?a, f(?c))とg(f(?b), ?a)のユニファイしよう.?aとf(?b)のユニファイからはじめる.結果は明らかである.他

の引数の?aをf(?b)で置換すると，f(?c)とf(?b)をユニファイしなければならない．これは?cを?bで置換すればよい．この結果，置換のリスト[(?a, f(?b)), (?c, ?b)]を得，ユニファイされた式はg(f(?b), f(?b))となる．

例2：g(?a, f(?a))とg(f(?b), ?b)のユニファイしよう．最初のステップは上と同様?aをf(?b)で置換する．つぎはf(f(?b)と?bをユニファイすることである．しかし，f(f(?b)は?bを含んでいるので不可能である．

パラメータでの代入：各パラメータは禁止される変数のリストをもっている．例えば，$b_{?a}$は?aに代入される項tの部分ではない．正しい置換が行われると，$b_{?a}$における?aの出現はt自身ではなく，tに含まれる変数によって置換される．例えば，?aをg(?c, f(?d))で置換すると，$b_{?a}$は$b_{?c,?d}$になる．?cあるいは?dへの代入が?aへの代入効果になり，したがって，?cと?dがパラメータに対して禁止される．

例えば，g(?a, f($b_{?a}$))とg(h(?c, ?d), ?c)とユニファイするために，まず?aをh(?c, ?d)で置換する．gの2番目の引数はf(b?c, ?d)と?cになる．これらの項は，?cがパラメータ$b_{?c,?d}$に対し禁じられているので，ユニファイできない．

∀－左と∃－右を表現するためにここで述べたパラメータ$b_{?a1,\cdots,?ak}$ではなく，通常，Skolem関数b(?a1, …, ?ak)を用いる．bは証明中に現われない関数記号であり，関数はパラメータのように働く．出現検査によってユニフィケーションが変数条件をチェックする．Skolem関数の引数が大きく成る可能性があるのに対し，パラメータによる扱いはコンパクトである．

4.2.2 推論規則の適用

分岐するノードはシーケントの前提をもつ推論規則をみなすことができる．後向き証明手順ではルートではなく，葉にアクセスする必要がある．証明を展開するには葉を分岐するノードに置換え，木の一部をコピーしなければならない．証明の探索過程では途中のノードは何の役割も持たない．途中のノードをすべて無視することができ，部分的証明木は2つの要素のみを保持すればよい．すなわち，ルート－主ゴール－は，証明したい論理式であり，葉－現在のサブゴール－は証明されるべく残っているシーケントである．

ゴールϕとサブゴールリスト[$\vdash \phi$]の対はϕの証明の初期状態である．ゴールϕと空のサブゴールリストの対は終了状態であり，証明が完了したことを表す．

2に掲げた14個の規則各々を証明状態から証明状態のリストへの関数として実現する．証明状態のサブゴールには1から順に番号をつける．各規則は，サブゴール番号と証明状態を受取って，証明状態を返す．たとえば，∧－左はつぎのように適用される：

conj_left_tac(3, st)

ただし，conj_left_tacは∧－左規則を適用する関数で，状態stのサブゴール3に∧－左を適用することを意味する．

サブゴールが$\phi \wedge \psi$, $\Gamma \vdash \Delta$なら，つぎの状態のサブゴール3はϕ, ψ, $\Gamma \vdash \Delta$になる．そうでないなら，∧－左をサブゴールに適用せず，conj_left_tacは空状態を返す．サブゴール5が$\Gamma \vdash \Delta$, $\phi \wedge \psi$なら，

conj_right_tac(5, st)

はサブゴール5の$\Gamma \vdash \Delta$, ϕとサブゴール6の$\Gamma \vdash \Delta$, ψをもつ状態をとる．

以上の各推論規則に対する関数に加えて，証明状態が基本シーケントになり得るかどうかを調べる関数unify_tacがある．⊢の左の論理式と右の論理式をユニファイできるなら，サブゴールを削除し，ユニファイに用いられる置換を証明状態のその他の部分に適用する．

4.2.3 証明手順の(半)自動化

上で述べた関数をコマンドとして入力し順次適用し対話的に証明を遂行することができる．簡単な論理式の対話的証明過程の例を図1に示す．

図1で，数字はサブゴール番号，L/Rは左右，Uはユニフィケーション，>，&，Aは，それぞれ論理演算子－－>，&，ALLを表す。

上のような実行方法は，推論過程を確認しながら，証明を進める場合には有効であるが，退屈な作業であることは否めない。証明を自動化するため，証明木の適用可能なユニフィケーション，推論規則(論理式の分解)，量化子の規則を深さ優先で探索し，見つかった規則を適用し，証明を実行する手順も用意している。量化子は無制限に繰返し展開されるので，自動的手順は停止しない可能性もある。その意味でこの手順は半自動的である。

5．法的推論への適用

法的論理は，最終的には三段論法と呼ばれる推論の適用である。すなわち，「すべての人間は死すべきものである」(大前提)「ソクラテスは人間である」(小前提)「故にソクラテスは死すべきものである」という例に示される推論方法を用いる。わが国のように成文法主義の場合，三段論法の大前提は，法律の条文の形をとる。問題は，結論を引出すべき小前提を作り上げること，すなわち，小前提を大前提の下におく方法，いいかえれば，ある事件の特殊の事実を一般的な規範のなかにもちこむこと－これを包摂（Subsumption）という－である［ヴィノグラドフ，1959］。

上述のように法的推論が深く論理と関わるから，法規範を論理的に表現し，これを自動的に処理する法的エキスパートシステム構築が試みられてきた［例えば，加賀山，1990］。ここでは，そのようなシステムの構築を目指していない。むしろ，法的推論における論理

図1 対話的証明例

入力論理式／コマンド	適用結果
P & Q －－> Q & P	1. empty ⊢ P & Q －－> Q & P
1 R>	1. P & Q ⊢ Q & P
1 L&	1. P, Q ⊢ Q & P
1 R&	1. P, Q ⊢ Q
	2. P, Q ⊢ P
1 U	No subgoals left.
2 U	No subgoals left.
(ALL x.(H(x) --> M(x)) &H(s))--> M(s)	1. empty ⊢ (ALL x.(H(x) －－> M(x)) & H(s)) －－> M(s)
1 R>	1. ALL x.(H(x) －－> M(x)) & H(s) ⊢ M(s)
1 L&	1. ALL x.(H(x) －－> M(x)), H(s) ⊢ M(s)
1 LA	1. H(?a_0) －－> M(?a_0), H(s), ALL x.(H(x) －－> M(x)) ⊢ M(s)
1 L>	1. M(?a_0), H(s), ALL x.(H(x) －－> M(x)) ⊢ M(s)
	2. H(s), ALL x.(H(x) －－> M(x)) ⊢ H(?a_0), M(s)
1 U	1. M(s), H(s), ALL x.(H(x) －－> M(x)) ⊢ M(s)
	2. H(s), ALL x.(H(x) －－> M(x)) ⊢ H(s), M(s)
	No subgoals left.

を分析し，論理的関係を闡明することを意図している。以下で法規と判決文を論理的に分析し，法的論理の特徴を考察するとともに，証明系による検証の有効性を例示する。

5.1 法規の解析

つぎの憲法条文を一階述語論理で表現し，その上でいくつかの特殊事例との整合性を分析しよう。

「何人も，法律の定める手続によらなければ，その生命若しくは自由を奪われ，又はその他の刑罰を科せられない。」（憲法31条[法定の手続の保障]）

この条文を［坂本・坂井，1971］にしたがって，以下のように論理化する。

L(y, x)	yはxの生命を奪う
Fr(y, x)	yはxの自由を奪う
P(y, x)	yはxに刑罰を科する
Fo(y, z)	yはzを適用する
Ll(z, y, x)	zは『yはxの生命を奪う』法である
Lf(z, y, x)	zは『yはxの自由を奪う』法である
Lp(z, y, x)	zは『yはxに刑罰を科す』法である
D(x, u)	xはuに住む
j	日本国

$$\forall x[D(x, j) \rightarrow [\forall z\{Ll(z, j, x) \wedge \neg Fo(j, z) \rightarrow \neg L(j, x)\} \wedge \forall z\{Lf(z, j, x) \wedge \neg Fo(j, z) \rightarrow \neg Fr(j, x)\} \wedge \forall z\{Lp(z, j, x) \wedge \neg Fo(j, z) \rightarrow \neg P(j, x)\}]] \quad \cdots(A)$$

つぎのようなケースがこの条文に違反するかどうかを調べる。

1）法の手続によってさえいれば，いかなる殺人行為もこの条文に違反するか否か？

これをつぎのように解釈する，すなわち，日本国が日本の法cに従い，日本在住のaに殺人行為をしたとし，つぎのように論理表現する：

$$D(a, j) \wedge Ll(c, j, a) \wedge Fo(j, c) \wedge L(j, a) \quad \cdots\cdots(1)$$

(1)が(A)に矛盾しない，つまり違反しないことを以下のように示すことができる。

(A)においてx＝a，z＝cとすれば，(A)から

$$(A) \equiv \neg D(a, j) \vee \neg Ll(c, j, a) \vee Fo(j, c) \vee \neg L(j, a) \quad \cdots\cdots(A)'$$

(1)が真だとすれば，各原始論理式が真であるから，(A)'においてFo(j, c)を除く論理式は偽である。しかし，Fo(j, c)は真であるから，(A)'は真である。すなわち，(A)と(1)は矛盾しない。

(A)'∧(1)を証明系を用いて実行すると，基本でないつぎの5個のシーケントを得る：

$$D(a, j), Ll(c, j, a), L(j, a) \vdash Fo(j, c)$$
$$\vdash D(a, j)$$
$$\vdash Ll(c, j, a)$$
$$\vdash Fo(j, a)$$
$$\vdash L(j, a)$$

これは，2で指摘したように，(A)'∧(1)を偽にする構造を与えている。すなわち，最初のシーケントはD(a, j)，Ll(c, j, a)，L(j, a)をすべて真，Fo(j, c)に偽にすれば，元のシーケントが偽になり，2番めのシーケントはD(a, j)を偽にすれば，全体が偽になることを意味する。以下同様である。したがって，これら以外の真理値を与えれば，全体を真にすることができる。

2）法の手続によらないが，生命も自由も奪わず，ただ，罰金刑を科すことはこの条文に違反するか否か？

これは「いかなる法の手続にもよらず，罰金刑を日本在住のaに科した」と解釈する。

$$\forall z\{Lp(z, j, a) \rightarrow \neg Fo(j, z) \wedge P(j, a)\} \quad \cdots\cdots(2)$$

(2)が(A)に矛盾することを以下のように示すことができる。

日本在住のaに罰金刑を科す日本の法律cが存在する，すなわちLp(c, j, a)が真である

とする。(A)から，つぎの論理式が成立する。

¬Fo(j, c)→¬P(j, a)≡

Fo(j, c)∨¬P(j, a) ……(B)

同様に，(2)から

¬Fo(j, c)∧P(j, a) ……(3)

明らかに，(B)と(3)は矛盾する，すなわち，一方が他方の否定になっている。したがって，(2)は(A)に違反する。

シーケント計算では，以下のように(B)と(3)の否定の論理和

¬(Fo(j, c)∨¬P(j, a))∨

¬(¬Fo(j, c)∧P(j, a))

あるいは，より直接的に

Lp(c, j, a)→

¬∀z{Lp(z, j, x)∧¬Fo(j, z)→

¬P(j, x)}∨

¬∀z{Lp(z, j, a)→

¬Fo(j, z)∧P(j, a)}

を証明することができる。とくに，後者は証明系の助けがなければ，長く退屈な作業を強いられる。このように比較的簡単な条文であっても，その整合性等のチェックには論理操作を支援するツールが有効である。

5.2 判決文の解析

現実には法規が多義的解釈を含み，その解釈をめぐって法的価値判断が鋭く対立したり，あるいは，新しい事態に対応すべく法規の解釈を当初より拡大し適用しなければならないことも少なくない。その具体的事例は判決文に見られる。

愛媛県が宗教法人靖国神社の例大祭などに公金を支出したことについて，憲法20条[信教の自由] 3項「国及びその機関は，宗教教育その他いかなる宗教活動もしてはならない」をめぐって争われた件に対する最高裁大法廷判決文の以下のパラグラフをとりあげる。これは，既に [大国他, 1999] で詳細に分析されているが，ここでは，各文の論理関係が明らかになるよう簡略化して提示する。

「①しかしながら，元来，政教分離規定は，いわゆる制度的保障の規定であって，信教の自由そのものを直接保障するものではなく，国家と宗教との分離を制度として保障することにより，間接的に信教の自由の保障を確保しようとするものである。②そして，国家が社会生活に規制を加え，あるいは教育，福祉，文化などに関する助成，援助等の諸施策を実施するに当たって，宗教とのかかわり合いを生ずることを免れることはできないから，現実の国家制度として，国家と宗教との完全な分離を実行することは，実際上不可能に近いものといわなければならない。

③さらにまた，政教分離原則を完全に貫こうとすれば，かえって社会生活の各方面に不合理な事態を生ずることを免れない。④これらの点にかんがみると政教分離規定の保障の対象となる国家と宗教との分離にもおのずから一定の限界があることを免れず，政教分離原則が現実の国家制度として具現される場合には，それぞれの国の社会的・文化的諸条件に照らし，国家は実際上宗教とある程度のかかわり合いを持たざるを得ないことを前提とした上で，そのかかわり合いが，信教の自由の保障の確保という制度の根本目的との関係で，いかなる場合にいかなる限度で許されないこととなるかが問題とならざるを得ないのである。⑤右のような見地から考えると，憲法の政教分離規定の基礎となり，その解釈の指導原理となる政教分離原則は，国家が宗教的に中立であることを要求するものではあるが，国家が宗教とのかかわり合いを持つことを全く許さないとするものではなく，宗教とのかかわり合いをもたらす行為の目的及び効果にかんがみ，そのかかわり合いが我が国の社会的・文化的諸条件に照らし相当とされる限度を超えるものと認められる場合にこれを許さないとするものであると解すべきである。」(『最高裁判所判例集』第

51巻第4号，1997：1680）

第①文は，政教分離規定が制度保障規定であることおよび制度保障規定は保障対象を直接保障せず，保障対象を制度として確立することにより，間接保障するものであることを述べる。したがって，以下のように論理化すれば充分その内容を反映しているであろう。

制度保障規定(政教分離規定)∧

∀x(制度保障規定(x)→(¬直接保障(x，f(x))∧

(制度として分離(x，g(x)，h(x))→間接保障(x，f(x)))))

ここで，f(x)，g(x)，h(x)はxによって間接保障される対象，xが分離する二つの対象を表す。xを政教分離規定とすれば，f(x)，g(x)，h(x)はそれぞれ政教分離規定が間接保障する信教の自由，政治，宗教と解すれば，第①文の帰結は以下の論理関係である。

¬直接保障（政教分離規定，信教の自由）∧

(制度として分離(政教分離規定，政治，宗教）→間接保障（政教分離規定，信教の自由)

第②文は，国が行政活動等をするなら，国は何らかの形で宗教と関わらざるをえないことを述べている。これを論理化する際の問題は，後に続く文との関係で国が宗教にかかわる場合とその程度をどのように表現するかである。ここでは，「xの場合に，国が宗教にかかわる」を述語（x，国，宗教）で表わすことにする。

規制する（国，社会生活)∨実施する（国，諸施策)→∃x関わる（x，国，宗教）

第③文は，国が宗教に全くかかわらないとすれば，社会生活に不合理を生ずることを指摘する。そして，不合理を生じないためには，国が宗教にかかわらざるを得ないという論理が隠れている。

∀x ¬関わる（x，国，宗教）

→不合理を生ずる（社会生活）

この対偶をとれば，

¬不合理を生ずる（社会生活）

→¬∀x ¬かかわる（x，国，宗教）

すなわち，

¬不合理を生ずる（社会生活）

→∃x かかわる（x，国，宗教）

となる。結局，第②文と③文の論理積からつぎの結論を得る：

(規制する（国，社会生活)∨実施する（国，諸施策))∧¬不合理を生ずる（社会生活）

→∃x かかわる（x，国，宗教）

この結論の後件と第①文の帰結の論理積

∃x かかわる（x，国，宗教）∧

(制度として分離(政教分離規定，政治，宗教）→間接保障（政教分離規定，信教の自由)

を成立させるxの存在条件を問題とすべきであることを第④文で指摘する。

最後に，このxが政治が宗教にかかわる行為の目的と効果によって判断されるべきであると結論する：

∃x かかわる（x，国，宗教)∧

(制度として分離(政教分離規定，政治，宗教)→間接保障(政教分離規定，信教の自由)

→∃u∃v(x∈目的(u，国，宗教，社会的・文化的条件)∧

程度(x)≦効果(v，国，宗教，社会的・文化的条件))

以上を要するに，第①文の政教分離規定が制度保障的観点から，政治と宗教を制度として分離し，間接的に信教の自由を保障するものであること，および第②，③文の国家は何程か宗教にかかわらざるを得ないことを前提とし，国家が宗教にかかわる場合と程度をその目的と効果から制限を加えるという結論に至る。

上述の論理式間に不整合のないことは容易に確認できるし，証明系を用いても示すことができる。しかし，各文の説得力，主張の正当性等はむしろ別のところにあると思われ

る。例えば，第①文の論理形式はつぎの文と基本的に同じである：

∀x(人間（x）→死ぬ（x））∧人間（ソクラテス）⊢死ぬ（ソクラテス）

すべての人間が死ぬという経験的事実を根拠として，歴史上の人物，ソクラテスが死ぬことを帰結する。これに対し，第①文は，経験科学的命題ではなく，憲法20条の制度保障的観点からの一つの主張であって，法的価値判断の表明である。同様に，第2，3文も現実の行政的判断からの主張である。憲法20条の立法目的あるいは信教の自由の保障の歴史的意義等を踏まえるなら，多少の不合理を認めても直接保障すべきものとする判断もあり得るであろう。さらに，ここで引用した文章にそって判断するにしても，国家が宗教にかかわる際の目的・効果を判断する基準があいまいであることにも問題は残る。うがっていえば，制度保障と国家が宗教にかかわることに固執し，信教の自由を間接保障するために，目的・効果に基づく論法を用いざるを得なかったと考えられなくもない。

このように判決文を論理的に解析することによって，どのような(法的)価値判断がどのように用いられているかを明らかにすることができる。価値判断の基準を闡明することによって，対立する価値判断を越える新たな価値判断の基準を展開する可能性も生まれてくる。

6．おわりに

ここで述べた試作システムはつぎの特徴をもつシーケント計算に基く証明系である：

1）対話的あるいは自動的にシーケント計算を支援する。

2）後向きに推論規則を適用するので，証明木の葉だけを記憶しておけばよいので，小さな作業領域で実行可能である。

3）シンプルな環境で動作し，基本機能しかサポートしていないので，可搬性に富む。

4）C/C＋＋で書かれているので，移植，拡張が容易である。

しかし，このようなものでも現実にはきわめて有効である。形式的手順による証明に避けられない煩雑で退屈極まりない作業をコンピュータが自動的に行ってくれるからである。現時点では，本試作システムは自動的証明システムというよりは，むしろ証明支援システムとして考えることができる。

さらに使いやすく有用な支援システムにするには，つぎのような機能を追加する必要がある。

1）推論規則をそのまま適用できる論理式しか扱えないので，入力した論理式を推論規則が適用可能な形式に自動的あるいはコマンドで変換する。

2）証明木を木で表現し，証明過程全体を把握できるようにする。

3）ビジュアル環境で推論規則の適用を指示できるようにする。

証明系の具体的問題への適用には，5で示したように自然言語表現を論理式に変換しなければならない。この作業は骨の折れる。とくに実用規模の問題を論理的に分析しようとするなら，証明系の実現よりも，むしろ論理式への変換が大きな問題である。テキスト本文を前処理し，その後の論理式への変換を支援するシステムが欠かせない。そのためには，自然言語の意味解析とその論理化を一体として捉えるアプローチが有効であろう[例えば，Carpenter，1997]。これらについては今後を期したい。

参考文献

Bonart, R. and Sufrin, B. (1999): Animating Formal Proof at the Surface: The Jape Proof Calculator, Computer Journal, Vol. 42, No. 3

Carpenter B. (1997): Type-Logical Semantics, The MIT Press

萩谷昌巳（1994）：ソフトウエア科学のための論

理学 岩波書店
加賀山 茂(1990):『法律家のためのコンピュータ利用法』有斐閣
大国充彦, 鳥居喜代和, 長田博泰, 田中 一(1999):社会情報解析－判決文における論理情報過程と価値情報過程との相互連関について－, 日本社会情報学会『社会情報学研究』, No. 3
Paulson, L. C. (1991): ML for the Working Programmer, Cambridge University Press
坂本百大, 坂井秀寿(1971):『新版現代論理学』東海大学出版会
ヴィノグラドフ, P. G.(1959):『法における常識』(末延三次・伊藤正巳訳, 岩波文庫)