ICOT Technical Report: TR-313

TR-313

論証支援システムにおける理論間の関係構造

南　俊朗、沢村　一

October, 1987

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

シンポジウム「情報とサイバネティックス」
1987年10月16日，九州大学・理学部

論証支援システムにおける理論間の関係構造

南　俊朗，沢村　一

富士通㈱・国際情報社会科学研究所

従来，様々な論理系を基礎とした証明チェッカー，コンストラクタについて研究がなされている．本稿では，それらとは異なり，ユーザが与えた論理系に基づいた論証を計算機で支援しようという意味での汎用な論証支援システムについて考察する．このような，システムにおいては種々の論理系を基礎とする理論群の管理を行う必要がある．

以下，まず，論証支援システムがどのようなものであるのかについて，概説を行い，その後，論証支援システムが理論間の関係構造に関してどのような情報を管理し利用するかについての考察を行う．

1．はじめに

現在，人間の知的活動に対する研究は，人工知能や認知科学などの分野で盛んに行われている．人間の内部では意識的，無意識的の両方の働きによって知的活動が行なわれているのであろうが，そのような知的活動の結果得られた知見を整理し，体系化することで初めてその知見を客観的に認識したり，厳密な形で他人へ伝えることができる．体系化を行い，知的活動を表現するためには，まず，何らかの形式的記号体系を設定し，その体系の中での記号操作法の枠組みを定式化することが必要となる．

問題設定を更に進めるために，人間がある問題の所在を認識し，それを体系化し，その下で問題解決を図るという状況について考察してみよう：

- まず，問題領域の持つ特徴，性格を分析し，どのような性質を持つものとして，問題領域を捉えたらよいかを考察し，そのイメージ・モデルを作る．
- 次に，頭の中のイメージ・モデルを元に，それを形式化し，対象モデルを作り上げ

る．そのモデルを用いて懸案の問題をどう表現し，解決したら良いかを考察する．すなわち，モデルによるシミュレーション実験を行い，その結果を参考にしながら問題解決の方策を探る．

・もし，対象モデルが問題を解決するのに十分精密であれば，肯定なり，否定なり，何らかの結論が得られるであろう．もしも，問題が解決できるだけの情報が足りない場合は，更に情報を補ったり，適当な仮定を設けることでモデルを修正し，結論を導くべく上記の実験を追加して行うことになる．

・結論が得られたなら，その結果を実際に適用してみる．適用の結果がモデルを用いた予想と異なる場合は，その原因を分析・検討し，モデルを修正する．

このような知的活動を論証と呼ぶことにする．論証を行うためには何らかの方法でモデルを構築することになるが，そのためには，モデルを記述するための表現体系が必要である．この表現体系を用いて，モデルを記述し，モデルの持っている性質を明らかにしたり，新しい概念を導入したりする．モデル記述のための表現体系がものの性質を記述し，それに関する推論を行うための記号体系であるということより，表現体系は一種の論理系であると考えられる．我々は，問題領域のそれぞれには，それに適した論理構造があるという認識に基づいて，ユーザの記述する様々な論理系を基礎とした（という意味での汎用の）論証過程の支援システムを実現することで，人間の知的活動の支援を行うことができ，そのような「論証支援システム」は知的活動をより強化するための道具となることができると考えている．

従来，多くの証明チェッカー，コンストラクタ（例えば，LCF/ML［1］，PL/CV2［2］，EKL［3］，CAP-LA［4］）が研究されてきたが，それらのシステムでは，ある特定の論理系の下での証明のチェック，構成を効率よく行うことを目指している．論理系を固定することで，その論理系特有の強力で効果的な機能を標準的に備えることができるという長所が生ずる．それに対して我々の論証支援システムは，1つのシステムの中で論理系自体を記述でき，その下での論証過程を支援しようというものであり，様々な論理系を統一的に取り扱うことができる．このシステムは汎用なシステムを目指す一方，論理系に依存した特有の効率的処理機能に関しては，その論理系毎に用意された戦略記述機能を用いることで，効率的な処理方法を記述し，それに基づいた証明を行うことで，高速化も同時に実現することを考えている．

汎用性を目指しているため，我々の論証支援システムは，論理系の違いも含めた様々なタイプの理論の集まりについての関係を把握し，管理する必要がある．特に，ある理論の証明が他の理論の証明の際にどのように参考にできるかは，大いに関心のある問題であり，本稿ではこの点に関する基礎的な考察を簡単に行う．

2．計算機による論証の支援

本節では，論証支援システムがどのように人間の行う論証過程を捉えているかについて述べ，また論証を行うためには，ユーザはどのような記述を行うことになるかについて説明する．

前節の考察の結果を受け，論証支援システムは，ユーザの行う論証過程は以下の３つの段階に分かれるものと認識する：

(1) 問題の把握，対象のイメージを作る

(2) 理論体系を構築し，モデルを記述する

(3) 記述されたモデルの性質を証明する

以下，それぞれの段階について詳細化する．ただし，前節でも述べたように，論証過程は(1)→(2)→(3)と単純に進むものではなく，(2)段階から(1)へ戻ったり，(3)から(1)，(2)へ戻ったりと，行きつ戻りつしながら進むものである．従って，ここでは，番号順に説明するが，それはあくまでも，大まかな進行状況に従うということである．

2．1　イメージ作り段階に対する支援

現在の論証支援システムでは，この段階に対する支援機能については，ほとんど考慮していない．デスク・アクセサリと呼ばれる，論証支援システムとは独立な，ミニ・ツール群を揃えたり，ＫＪ法［９］等の発想技法を提供することも，考えられるが，論証支援として適切なものに関しては，今後の研究課題である．

2．2　理論構築段階に対する支援

理論体系を構築する段階は，更に，次の３つに分けられる．

① 言語系の設定

何らかの表現系を設定するためには，そこで用いられる表現形態をまず定める必要がある．言語系の設定は，そこで用いられる構文の定義，および，記号の定義からなる．

② 導出系の設定

論理系における定理をどのように導くかを定めるのが導出系である．この表現には①で定められた言語表現を用い，標準的には，推論規則，書き換え規則等の導出規則を用いて表現される．

③ モデル記述

言語系，導出系を合せて論理系と呼ぶ．設定した論理系によって，想定している対象を記述する．記述は論理式の集合を公理系として与えることで，現在問題としている対象の論理的表現（モデル）である．

2．2．1 言語系の設定

例えば，様相論理を定義するためには，□，◇等の様相演算子が必要であるように，論理系はそれぞれ，その論理系特有の記号を持つ．その論理系で基本的なものの他，省略記法のために新たに導入される記号もある．構文の構造の定義は記号定義された記号を基本とし，それらに構文演算子を適用して得られる．すなわち構文定義は次の形式の構文規則の集合によって与えられる：

＜構文名＞ ← ＜記号名＞

＜構文名＞ ← ＜構文式＞

ここに，構文式は次の形式で与えられる：

＜構文式＞ ← ＜構文名＞

＜構文式＞ ← ＜構文演算子＞（＜構文式＞，．．．）

論証支援システムでは，前者の構文規則は記号定義として与え，後者の形式の規則のみを構文定義として与えるものとする． 記号に対する構文的構造は＜記号名，構文名＞なる対表現で与えられる．記号に対する情報としては，構文的構造以外に型等の意味論的属性も必要に応じて与えられる．そのような目的のために記号定義の際，記号の属性とその

属性に関する値を与えることができるが，本稿では，構文的構造のみを考察する．また，記号の定義を構文定義中で行う流儀も考えられるが，本システムでは構文構造定義と記号の構文属性定義は分離して扱う．それは，記号定義を別に与える方式の方が，記号に対する新たな属性を付与したりの変更が比較的容易であり，また一括して記号属性を管理できるという点で有利であるからである．しかし，言語系固有の記号と考えられ，その言語系で考える限り変化しない記号に関しては　＜構文名＞　←　＜記号名＞　という構文定義によって規定することも可能である．

表現の構文は，基本となる記号を元に構文演算子を施して得られる．構文演算子には，並置演算子，リスト構成子，木構成子等がある．これらの演算子／構成子は次のようなものである．

・並置演算子

並置演算子は引数を単に並べる働きをする．

例：並置（‘f’，‘(’，‘x’，‘)’）　によって　表現　f（x）が構成される．

通常の言語理論で用いられる文字列表現の構文は，このような並置演算子を用いて表現したものと考えられる．字下げ付きのテキストのような２次元的な文字列表現のためには，横方向の並置演算子の他に縦方向の演算子も必要となる．すなわち，２次元文字列表現の構文演算子は横並置演算子と縦並置演算子の２種類である．

・リスト演算子

リスト構成子は引数のリスト表現を作り出す．

例：リスト（‘a’，‘b’，‘c’）　によって　表現　［a，b，c］が構成される．

例えば，ＬＩＳＰ言語の場合，データの内部表現は全てリストであり，その表現形態はリスト構成子によって作り出されたものと考えられる．

・木演算子

木構成子は引数を要素とする木表現を作り出す．

例：木（‘ｔｒｅｅ’，‘x’，‘y’）　によって　表現　ｔｒｅｅ（x，y）

が構成される．

例えば，Ｐｒｏｌｏｇ言語の場合は，データは木構造で表現されている．この場合の内部表現は木構成子で作られたものと考えることができる．

以上で述べた構文規則集合，記号定義集合を与えることで言語系を設定できる．

［定義］

構文規則集合は，＜構文要素名＞←＜構文式＞　なる規則の集合である．

記号定義集合　とは　記号定義の集合である．

ここに，記号定義は　＜記号，構文名＞として与えられる．

言語系は　構文規則集合と記号定義集合を与えることで定められる．

なお，論証支援システムで用いられる言語系は「論理式」なる構文要素名を定義する必要がある．「論理式」表現を用いて次に述べる導出系の定義が行われる．

２．２．２　導出系の定義

導出系は「論理式」なる構文名を持つ言語系の下での「定理」なる概念を，特定の論理式として定義する．導出系を定めるための方法はいろいろ考えられるが，標準的な方法は次のものである：

（ｉ）基礎となる公理　を与える．

（ii）推論規則，書き換え規則等の論理式導出規則　を与える．

定理とは（ｉ）で与えられた公理に対して（ii）の変形規則を（０回以上）適用して得られる論理式のことである．

導出規則をどのように適用した結果定理が得られたかという導出規則の適用の表現をその定理の証明と呼ぶ．１つの証明に対する証明は複数個存在し得る．

［定義］

導出系は　基礎となる公理の集合，導出規則　の２つから構成される．

論理系は　記号系とその記号系に関するある導出系の２つから構成される．

導出規則の合成に名前を付けて１つの導出規則とすることができる．それを派生規則と呼ぶ．

２．２．３　モデルの記述

この段階では(1)で作成した論理系の下で対象の性格を明らかにする論理式の集合を対象公理系として提示する．対象公理系も含めて１つの理論が表現されたことになる．

［定義］

理論は　基底をなす　論理系　及び　公理系　からなる．

理論Ｔにおいて，公理より導出規則によって論理式φが導出されるとき，φをＴの定理であるといい，Ｔ⊢φと表わす．

問題対象を表現するための公理は，基礎公理と対象公理とに分けて与えられる．１つの理論を規定するための性質としては，両者に区別はない．それらの性格の違いは，ユーザが現在考慮している問題領域が本来的に持つべき性質であると考えるときは，それを基礎公理として表現し，ある対象に関する特定の性質であると考える時は，それを対象公理として表現することになる．論理系は基礎公理を含むが，それは論理系が問題領域自体を表現するものであり，その問題領域の中の特定の状況は対象公理によって規定されるものと考えているためである．

２．３　モデルに関連する定理の証明

論証の最後の段階では，それ以前の段階で定められた状況の下で，対象の性質に関する種々の定理を導出し，対象の持つ性質を明らかにすることである．問題解決の状況では，対象に関するどのような定理が成り立てば良いのかを表現し，その定理の証明を見付けることで問題の解決を図る．

証明された定理（とその証明）はその理論に関する定理のプールに入れられる．このプール内の定理はそれ以降の定理の証明の際に参照することができる．

以上で説明したような段階を経ながら論証支援システムのユーザは理論を構成し，その

理論における定理を作成していくことになる．論証支援システムは，起動されると，まずユーザに理論の一覧を示す．ユーザは自分の仕事をどの理論の上で行うかを決定する．既存の理論を選択することもできるし，新らしい理論を作成し，その下で仕事を進めることもできる．ある理論が選択されると，論証支援システムはその理論に属する定理を表示する．その定理を参照しながらユーザは新しい定理を考案していくことになる．新しく理論を作成する場合は，白紙の状態から全てを構築しても良いし，既存の理論の記号系，論理系，公理等を取り込んで利用することも自由にできる．これらの機能は，結局，エディタの機能であるということであり，理論を構成する個々のデータ毎に，そのデータを編集するためのエディタを提供することで論証支援システムはユーザを支援する［6，7］．

3．理論と理論の関係

本節では論証支援システムが理論間の関係についてどのような部分に注目するかについて考察する．特に，次の2つの観点に関して議論する．

1. ある理論での証明を行うために他の理論の証明を参照する場合の問題．
2. 新しい理論を追加したり，既存の理論を削除したりする場合に生じる，理論間の参照関係に関する問題．

3．1．他理論の証明の参照に伴う問題

前節でも述べたように，論証支援システムのユーザは，普段は既存の理論の下で論証を進めていくことになるが，必要に応じて，新しい理論を自由に作成することができる．しかし，新しい理論を全く初めから作り上げるのは大変な手間がかかるものであり，むしろ，以前に得られた理論を参考にして，部分的に修正を施したり，幾つかの理論を混ぜ合せたりして新しい理論を作る方がはるかに楽であり，効率的である．

どうかの方法で新しい理論を構成したとして，その理論の下で定理の証明を見付ける場合にも，別の理論を参考にできるならば参考にした方が証明の探索はより効率的であろう．そのような，目的のために，理論間にどのような関係があり，その関係を用いることで証明の際どのように参考できるかを管理する機能を論証支援システムは備えている必要がある．

1つの理論を規定するために必要なデータとしては，記号系，構文規則，導出規則，公理系（基礎公理および対象公理）がある．我々は，理論Tと理論T'の関係として，一方の理論で定理φが成り立つことで他方の理論で成り立つ定理に関してどういうことが言えるかに興味がある．

まず，理論Tの論理式がそのまま理論T'の論理式である場合を考える．すなわち，理論Tの記号は理論T'の記号であり，理論Tで論理式を構成する構文規則がそのまま理論T'の構文規則にもなっている場合である．さらに，導出規則，公理系に関しても理論Tのものがそのまま理論T'の導出規則，公理系であるとすると，理論Tの証明はそのまま理論T'の証明であるので，明らかに，Tの定理φはT'の定理となる．

［定義］

理論T，T'に関して，Tの論理式がすべてT'の論理式であるとき，任意のTの論理式φに対して，

$T \vdash \phi$ ならば $T' \vdash \phi$ であることを，$T \leq T'$ と書くことにする．

［命題1］

理論Tの記号系，構文規則，導出規則，公理系のそれぞれが理論T'のものの部分集合であるならば，$T \leq T'$ である．

既存の理論に導出規則や公理を加えたり，逆に，省くことで，新しい理論を作る場合はこの関係が成立する．既存の2つ，もしくはそれ以上の理論の公理系を集めて新理論を構成するような場合にもこの関係が成り立つ．論証支援システムはこのような関係の理論TとT'に対して，包含関係：$T \Rightarrow T'$ なる関係情報を記録する．

理論Tの導出規則，公理系がそのまま理論T'の導出規則，公理系に含まれていない場合にも$T \leq T'$が成立する場合もある．

［命題2］

理論Tの記号系，構文規則はそれぞれ理論T'の対応物の部分集合であるとする．理論Tの導出規則，公理のそれぞれが理論T'の定理，導出規則の合成として解釈できるなら

ば，T≦T'である．

この場合は理論Tの導出規則，公理のそれぞれに対して理論T'の導出規則の合成，定理への写像Fが与えられている．この写像Fは理論Tの証明をT'の証明へ，また，Tの定理をT'の定理への写像として拡張できる．したがって，Tにおける定理φと証明Pに対して，φは証明F（P）を持つT'の定理である．論証支援システムはこのような関係の理論に対して，解釈関係（<対応情報>）：T⇨T'なる関係情報を記録する．ここに，<対応情報>はTの導出規則や公理のそれぞれrに対する（r，F（r），F（r）がT'の導出規則の合成もしくは定理であることの証明）なる3つ組の集合である．

以上の2つの場合は理論Tの論理式φがそのまま理論T'の論理式である場合を取り扱ったが，そうでない場合にも一方の証明を他方の証明に用いることができる．

［命題3］

理論Tから理論T'への次のような変換Fが与えられているとする．

・FはTの導出規則のそれぞれに対して，T'における導出規則もしくは派生規則を与える．

・FはTの公理のそれぞれに対して，T'の定理を与える．

そのとき，FはTの定理からT'の定理への変換に拡張でき，FによってTの証明からT'の証明を得ることができる．

この場合は一般に論理式そのものを何らかの方法で変換することで，理論Tにおける定理およびその証明を理論T'の定理およびその証明へと変換することになる．論証支援システムはこのような理論T，T'に対して　対応関係（<変換情報>）：T⇨T'なる関係情報を記録する．ここに，<変換情報>の部分は，命題3で述べた変換方法そのものと，その正当性の証明の組の集合である．

以上述べたもの以外にも様々な理論間の関係情報を論証支援システムは管理する必要がある．また，T⇨T'かつT'⇨T"の両方の関係からT⇨T"の関係を推論することも行わなければならない．

3．2．理論の変更に伴う問題

論証支援システムは，ユーザが試行錯誤的に理論を作り，更新していく状況を想定しているため，1つの理論を作り上げる過程において色々なレベルでデータの更新が発生する．記号のレベルでの追加，削除があり，公理のレベルでの追加，削除がある．このような状況においてはデータの変更によって1つの理論内や，理論の集まり全体などでの整合性が成り立たなくなることが起る．例えば，ある理論の記号を削除すると，その理論内でその記号を用いている全ての論理式，定理とその証明は意味を持たなくなる．したがって，特定の記号，導出規則，公理，理論．．．のみを削除することでは全体としての整合性を保つことはできない．

整合性を保つためには，たとえば，ある記号を削除する場合は，その記号が使われている導出規則，論理式等を全て見付け出し，それらを削除するということを行うと良い．しかし，この方法では，削除が頻繁に起る場合，非常に非効率的である．そのような状況では，それに代って記号毎にその記号を使用している部分の一覧表を持ち，その表を参照して削除を行うという方法で保管するデータ量を増やす代りに削除の時間を減らすことができる．

一方，削除に制限を設けることで，この問題を解決する方法も考えられる．たとえば，ＬＣＦ［1］では理論に対して draft理論という概念を導入する． draft理論の場合は理論を構成する公理の変更が可能であり，少数の定理ならば作成できる．しかし， draft理論を親とする理論を作成することは許されない．ここに，理論の親子関係は，親の持つ公理を全て引き継ぐ理論をその理論の子理論とする関係である．1つの理論は複数個の親理論を持つことができる．一般的には新しい理論は draft理論として，まず作り，理論として完成した時点で通常の理論に昇格させる．その後は，他の理論から親理論として参照されることが可能となる．

我々の論証支援システムの場合は事情はもっと複雑である．参照関係にしても，記号系の間，構文系の間，公理系の間，．．．と理論間以外の参照関係が種々存在する．また，公理系はその基礎となっている構文系等の言語系を参照しているといったように，参照関係のタイプも様々である．このような状況の下での整合性の保持を次のように行うことにする．記号系，構文系等の系毎に参照関係を持つ．ある系が draftであるかどうかは陽には指定せず，他の系からの参照のない系は draftであるものとして，その変更は自由であ

る．たとえば，論理式の構文を定義した言語系はそれを参照する導出規則が生成される以前は自由に変更できる．一旦，それが参照されると，その変更は許されない．ただし，各系は参照カウンタを持ち，他の系から参照されるたびにその値を１増やし，参照が取り消されるたびに１減らす．参照カウンタの値が０の時はその系は draft状態と見なす．このメカニズムにより，ある系が参照していた系が勝手に書き換わるということは起こらない．

他の系から参照されている系をどうしても書き換えたい場合は，形式上は新しい系を生成することになる．すなわち，draft状態でない系を書き換えるときは，ユーザにその意図を確認した後，これまで，用いてきた系の情報は，それを参照している系のために保存し，その後ユーザは系を変更することができる．システム内では自動的に新しい系が生成され，その系は参照されていないので draft状態であり，変更が自由にできることになる．系の元の情報はその参照カウンタ値が０となった時点で削除される．このような管理方法によって，ユーザからは比較的自由に系を書き換えることができ，しかも，整合性の保持が実現できる．

４．おわりに

本稿では，論証支援システムにおける記号系，構文系，言語系，導出系，理論等の用語について定義し，様々な理論間の関係を管理する際の問題点について考察した．ある理論における定理およびその証明を参考に他の理論を構成する場合に関係情報が有効に用いられる．本稿では考察しなかったが，このようなシステムをユーザが実際に用いる状況を想定すると，ユーザとシステムとの対話の状況，すなわちユーザ・インタフェースに対する配慮が重要となる［５～８］．単に，論理式等の情報をいかに表示するべきかという問題に留まらず，論証過程において頻繁に現れるような，構成しては修正を繰り返す作業を効率的に行うためにはどのような支援が有効であるかに関しても多くの研究が必要である．また，問題を認識し表現する過程における発想を促すための支援法についても，今後の検討課題である．

謝辞

日頃，御指導，御鞭撻を頂く北川敏男会長，榎本肇所長に感謝いたします．また，論証支援システムの共同研究者である富士通研究所の裘東善，佐藤かおる，土屋恭子の各氏にも感謝します．なお，本研究の一部は第五世代コンピュータ・プロジェクトの一環としてICOTの委託で行ったものである．

参考文献

[1] Gordon, M. J., Milner, A. J. and Wadsworth, C. P. : Edinburgh LCF, LNCS 78, Springer-Verlag, 1979.

[2] Constable, R. L., Johnson, S. D. and Eichenlaub, C. D. : An Introduction to the PL/CV2 Programming Logics, LNCS 135, Springer-Verlag, 1982.

[3] Ketonen, J. & Weening, J. S. : EKL - An Interactive Proof Checker, User's Reference Manual, Dept. of Computer Science, Stanford University, 1984.

[4] ICOT CAP-WG : CAP プロジェクト(1)～(6)，情報処理学会第32回全国大会，１９８６．

[5] 南，沢村：論証支援システムの一構成，ソフトウェア科学会第３回大会，１９８６．

[6] 土屋，佐藤他：論証支援システムのための論理式エディタ，情報処理学会第35回全国大会，１９８７．

[7] 南，沢村：落書帳からの証明　－計算機による論証支援のための証明方法論の一考察－，情報処理学会第35回全国大会，１９８７．

[8] 沢村，南：汎用の論証支援システムの構想とその実現法，情報処理学会ソフトウェア基礎論研究会，１９８７年１０月．

[9] 川喜田二郎：発想法－創造性開発のために，中公新書１３６，中央公論社，１９６７．