Vol. 37 No. 4　　情　報　処　理　　Apr. 1996

解　説

並行計算の理論の最近の動向

3. 線形論理に基づく並行計算モデル†
—並行計算の論理的理解の試み—

岡　田　光　弘††

1. 線形論理は何が新しいか

1987年Jean-Yves Girardは線形論理という新しい理論をTheoretical Computer Science誌上に発表しました[1]．同誌の編集長であったMaurice Nivatは内容の重要性に鑑みて，Girardから提出されたこの論文をレフェリーイング（査読手続き）なし，また待ち期間なしという異例の取り扱いをして同誌の分冊1号のすべてのページを使って出版しました．それ以来線形論理の理論は論理学界および理論情報科学界で非常な注目を浴び，活発に研究が進められてきました．

以下本章では並行計算モデルの理論と関連する側面に限定して，線形論理がそれまでの論理とどのように違うかを最低限必要な範囲で説明することにします．

これまでの伝統的な論理学においては，∧（かつ）という論理記号についての次のような推論規則が無批判に認められてきました．

$$\frac{\Gamma \to A \quad \Gamma \to B}{\Gamma \to A \wedge B}$$

これは「ΓならばA」と「ΓならばB」が正しければ「ΓならばAかつB」も正しい，ということを主張しています．この∧に関する論理推論規則は多くの場合何の問題も生じないのですが，たとえば「100円あればチョコレートが一箱買える」と「100円あればキャンデーが一箱買える」という2つの文をこの論理推論の前提にあてはめようとすると，

$$\frac{\text{100円}\to\text{チョコレート} \quad \text{100円}\to\text{キャンデー}}{\text{100円}\to(\text{チョコレート}\wedge\text{キャンデー})}$$

となり，100円あればチョコレート一箱とキャンデー一箱両方買える，といった間違った結論が導き出されてしまいます．

このような欠陥はどこから出てきたのかというと，実はこれまでの伝統的な論理推論では，取り扱う命題は(1)時間性を持たない，また(2)含意（ならば）は消費関係を含まない，という条件を満たすものだけを扱ってきたことによります．このような時間性や資源の消費性を含まない命題の典型は，たとえば数学の命題で代表されるような，永遠不変の真理を主張する命題です．このような永遠不変の真理はギリシャ哲学以来エピステーメと呼ばれ人間の知識の理想と考えられてきたのです．そして，論理学の目的はそのような永遠不変の真理についての論理構造を解明することにあると考えられてきたわけです．その結果，伝統的な論理学は数学基礎論や科学基礎論の分野で今世紀はじめに大きな成功を収めたのです．ところが，上に出したような「100円あればチョコレートが一箱買える」という命題は直接資源の消費関係を表した命題ですし，また，上の推論の結論部分で∧（かつ）を「チョコレートもキャンデーも同時に両方買える」という意味で解釈した場合，論理記号∧（かつ）を「同時に成立する」という時間性の概念と結びつけて解釈していることに気づきます．実は，上の例で伝統的な論理推論を用いて不合理な結論が出てきた理由は，同時性（または並行性）概念や資源の消費概念を直接取り扱うことを前提としていない伝統的論理推論を用いて，同時性（並行性）や消費関係に関連した命題を推論しようとした，ということにあります．しかし，現在の情報科学の論理的理解に必要なのは，まさにこのような並行性や計算資源の消費性を実現し得るダイナミックな論理的枠組なのです．

このことから分かるのは，同時性（並行性）や消費関係などを表わす命題についての推論を考えるためには，伝統的論理学を超える新しい論理を

† A Concurrency Model Based on Linear Logic —Towards the Logical Understanding of Concurrency— by Mitsuhiro OKADA (Department of Philosophy, Keio University).

†† 慶應義塾大学文学部哲学科

導入しなければならない，ということです．そして，このことを実現させたのが線形論理という新しい論理理論なのです．たとえば，上の例についていえば，伝統的論理において考えられていた∧（かつ）に対して，線形論理では2つの異なった「かつ」を導入します．一方は$A \otimes B$で表される「AとBとが同時に成立する」ことを示す「かつ」で，他方は$A \& B$で表される「AとBもどちらでも成立し得る（しかし，現実にはその一方だけが成立する）」ということを意味する「かつ」です．そして，⊗についての推論規則と&についての推論規則として各々，

$$\frac{\Gamma \to A \quad \Delta \to B}{\Gamma, \Delta \to A \otimes B} \qquad \frac{\Gamma \to A \quad \Gamma \to B}{\Gamma \to A \& B}$$

を考えます．$\Gamma \to A$を「100円あればチョコレートが一箱買える」とし，$\Delta \to B$を「100円あればキャンデーが一箱買える」として左側の規則を適用すると，結論の左辺には仮定としてΓとΔの和集合が現れるので，結局「200円あればチョコレート一箱とキャンデー一箱とが同時に買える」という正当な結論が導き出せるのです．一方，右側の規則に上の2つの前提を適用すると，「100円あればチョコレート一箱を買うことも可能だし，キャンデー一箱を買うことも可能である」という，これまた正当な結論が導き出せるわけです．

この例は線形論理の新しさの氷山の一角を説明したにすぎませんが，このことからも同時性や並行性概念およびそれにともなう資源の消費関係などが新しい線形論理でうまく取り扱われるという様子を読者の皆さんに分かっていただけるのではないかと思います．（より詳しくは岡田[3]参照．）

もともと伝統的論理学と計算機科学の計算モデルの理論との間には密接な関係が確立されていました．ただし，伝統的論理学に基づいて構成される計算モデルの理論は伝統的計算モデル，すなわち継起的計算モデルの理論でした．線形論理を伝統論理の代わりに用いることにより，これまで確立されてきた継起的計算モデルの理論を，並行計算モデルの理論に変更しようとするのが，私達の基本的なアイディアなわけです．そこで，この事情が分かりやすいように，次の章では論理学，その中でも特に証明論といわれる分野で発達してきた証明の正規化理論と計算機科学における計算モデルの理論との一般的な関係を簡単に説明することにします．

論理学における重要な定理に，証明の正規化定理があります．この定理は，「与えられた論理体系の（一般的な形をした）証明木は，その論理体系の正規形といわれる特別な形の証明に常に書き換え可能である」ということを主張します．論理学の証明の正規化の概念を使って計算機科学や情報科学の計算モデルを考察する仕方には大きく分けて3通りあります．その1つは，証明検索（proof-search）パラダイムと呼ばれ，たとえばプログラミング言語の計算モデルでいえば，（プロローグなどの）論理プログラミング言語の計算モデルがその例です．第2は，簡約（reduction）パラダイムと呼ばれ，たとえば（LISPやMLなどの）関数型プログラミング言語の計算モデルがその例です．第3は，項書き換え（term-rewriting）パラダイムで，たとえば（OBJなどの）実行可能な代数仕様言語などの等式型プログラミング言語の計算モデルがその例です．ここで，基本となる論理体系のとり方によっていろいろ違った論理計算モデルが構成できます．特に，この基本になる論理体系として線形論理をとると，並行計算モデルの3つのパラダイムが構成できるのです．このような情報科学で使われる論理的な計算モデルの3つのパラダイムやそれらを線形論理に適用した場合のより詳しい説明は岡田[2),3)]を見ていただくことにして，誌面の制限のためここではこの3つのパラダイムのうちの証明検索パラダイムを例にとって線形論理による並行計算の論理モデルを紹介することにします．

ところで，線形論理の論理推論規則は，たとえば⊗（parallel-and）については，

$$\frac{\Gamma \vdash \Sigma, A \quad \Delta \vdash \Pi, B}{\Gamma, \Delta \vdash \Sigma, \Pi, A \otimes B} \qquad \frac{A, B, \Gamma \vdash \Delta}{A \otimes B, \Gamma \vdash \Delta}$$

という形になっています．左側のものは⊗の右入れ規則と呼ばれ，これは前章であげた⊗の導入規則のより一般的な形です．右側のものは⊗の左入れの規則と呼ばれます．線形論理の論理規則は，各論理記号について右入れ規則と左入れ規則からなっています．以下においてこの解説では，説明の簡略化のために，左入れの規則だけを使った並行計算の計算モデルについて説明をすることにします．右入れ規則も含めた完全な線形論

理体系およびその各々の通常の論理的な意味の説明については，岡田[2),3)]，竹内[4)]などを参照下さい．

2. Proofs-as-Processes 原理

この解説において，次のような論理的概念と並行プロセス概念との同一視を考えることにします．すなわち，論理記号をアクション名と，また（ある論理記号に対する）論理推論を（対応するアクションに基づく）状態の推移と同一視します．また，線形論理の各論理式はプロセスの仕様と同一視します．論理記号とアクション名の対応は次のとおりです．

- $A \otimes B$（並行アクション）プロセス A とプロセス B とを並行して実行する．
- $\alpha \otimes B$（送信アクション）トークン α を送り出し，プロセス B を実行する．
- $\alpha \multimap B$（受信アクション）トークン α を受け取り，その情報を用いてプロセス B を実行する．
- $!A$（bang-アクション）プロセス A のコピーをいくらでも必要な数だけ作って，各々のコピー A を実行する．
- $A \,\&\, B$（選択アクション）A か B のどちらかを選んで，それを実行する．
- $\exists x A$（guarding/renaming-アクション）および $\forall x A$（モーバイル・メッセージ受信）．これらについては，予定している本稿の第2部で説明します．

各アクション（action）は，対応する論理記号の論理推論規則を下から上に読むことによって次のように表されます．

- 並行プロセスの発動（parallel-invoke）

$$\frac{\Gamma, P, Q, \Delta \vdash}{\Gamma, P \otimes Q, \Delta \vdash}$$

（並行発動アクション $P \otimes Q$ は，並行プロセス P と Q を発動します．）ここで $P \otimes Q$ のまわりにある Γ や Δ は，各々論理式の有限列を表し，これらはプロセス $P \otimes Q$ の環境内にあってプロセス $P \otimes Q$ と並行して実行されている並行プロセスの集合と解釈します．

この並行発動アクションの特別な場合として，次の送信アクションが考えられます．

- メッセージ送信（sending）

$$\frac{\Gamma, \alpha, P, \Delta \vdash}{\Gamma, \alpha \otimes P, \Delta \vdash}$$

（送信アクション $\alpha \otimes P$ は，メッセージ α を送ると同時にプロセス P を発動させます．）ここで α, β, γ 等はメッセージ（実際には，メッセージを含むポート名またはチャンネル名）を表していると考えて下さい．そして，これらは原子論理式（論理記号を1つも含まない論理式）で表されていると仮定します．

- メッセージ受信（receiving）

$$\frac{(\alpha \vdash \alpha) \quad \Gamma, Q, \Delta \vdash}{\Gamma, \alpha \multimap Q, \alpha, \Delta \vdash}$$

（受信アクション $\alpha \multimap Q$ は，環境内にあるメッセージ α を受け取り，それを消費してプロセス Q を発動します．）

- 選択（choice）

$$\frac{P, \Delta \vdash}{P \,\&\, Q, \Delta \vdash} \qquad \frac{Q, \Delta \vdash}{P \,\&\, Q, \Delta \vdash}$$

（選択アクション $P \,\&\, Q$ は，（このシステムの環境の内部で）自由にプロセス P または Q を選び，つまり外から見ると P または Q が自由に選ばれ，その選ばれたプロセス P または Q が発動します．）

複製（bang）！アクションは次の線形論理推論規則に対応します．

（Milner の bang）

$$\frac{\Gamma, P, !P, \Delta \vdash}{\Gamma, !P, \Delta \vdash}$$

（$!P$ はプロセス P のコピーをつくり，そのプロセスを発動させる．）

上のような並行プロセス計算のアクションを表わす論理記号の集合が実はちょうど線形論理体系に一致するのです．より詳しくいうと，上の推論規則群はちょうど線形論理の正規証明（normal proofs）を与える規則に対応します．これに cut 規則と呼ばれる論理の推移律（または，しばしば三段論法とも呼ばれてきました．）を加えたものが，本来の線形論理なのですが，有名な Gentzen の基本定理（cut 消去定理）をこの線形論理に適用すると，線形論理の任意の証明は cut 規則を用いない正規形の証明に変形できることがいえるのです．このことから，プロセス計算のアクションとして有意味な論理推論規則の体系が，ちょうど線形論理体系の正規証明の規則に対応していることが分かるわけです．

論理学のことを少し知っている人のためにここで注意を加えると，実は上で使用している線形論理の論理式は，一般のそれに比べて非常に制限された形であることに気付きます．上の線形論理の含意は $\alpha \multimap P$ なる形のもの，つまり含意の前件がメッセージ α のような原子論理式に限られているわけです．これは，本質的には，ここで使われる各々の論理式は高々ホーン節である，ということを意味しています．この制約は実は，単に並行計算理論だけでなく他の情報科学の諸分野においても，論理プログラミング・パラダイムまたは証明検索パラダイムによる計算モデルを考える場合の本質的な特徴なのです．

ところで，本稿において私達はこれまでの論理学の伝統的な証明可能性の概念を超えた，より広い“証明”概念を考えていきます．そして，それが種々の並行計算モデルの“証明論的表現”を与える上で非常に重要になってくるのです．

実は，いままでの伝統的な証明概念は，ある論理式が与えられた公理群から“証明できる”ことを表すものでした．伝統的な証明は公理を葉(top node)として推論規則を枝分かれ(inner node)として構成される有限木と考えたわけです．

これに対して，私達は公理からはじまらない木も証明と考え，また有限でない木も証明と考えることにします．(これは，今まで二千数百年間論理学や数学で考えられてきた証明概念とはまったく違った新しい“証明”の見方です．)そして，証明の各道(path)，つまり私達の拡張された意味での証明木の中の根(root)から出発する道筋，の各々をプロセスの表現と考えます．こう解釈しますと，論理式 A の証明とは，A で表現されているプロセスの仕様に従って生成される可能なプロセスの集合を表していることになります．そして，公理に到達して完結しないような証明木の道にも，また有限の長さでは生成過程が停止しないような証明の無限の道にもプロセスとしての意味が自然に与えられることが分かります．これがリアクティヴプロセスと呼ばれている終りのないプロセスです．(オペレーティングシステムなどはその例です．)

たとえば，

$$\dfrac{\dfrac{\beta, \gamma^2 \vdash}{\beta^2, \beta \multimap \gamma^2 \vdash}}{\alpha, \alpha \multimap \beta^2, \beta \multimap \gamma^2 \vdash}$$

は，$\alpha \multimap \beta^2$ というプロセス(メッセージ α を受け取り，メッセージ β を2つ放出するプロセス)とプロセス $\beta \multimap \gamma^2$ (メッセージ β を受け取り γ^2 を放出するプロセス)とメッセージ α が初期設定された仕様だとし，そこからメッセージトークン γ を2つとメッセージトークン β を1つ生じてコミュニケーションプロセスが終了していることを示しています．

また，次の(無限木の)証明は，コミュニケーション・システムの仕様として，「α を受け取り β を放出するリアクティヴ(reactive)なプロセスと α をいくらでも生成するリアクティヴなプロセスが環境内に与えられている」ことを示し，output として「β をどんどん生成していく」という1つのリアクティヴなプロセスを表しているわけです．

$$\begin{array}{c} \vdots \\ !\alpha, !(\alpha \multimap \beta), \beta^n \vdash \\ \} \\ !\alpha, !(\alpha \multimap \beta) \vdash \end{array}$$

プロセス表現の種々の概念と証明論の種々の概念との関係を表す簡単な例として次のものを考えます．今次の2つの受信アクションに P, Q と名前を付けておくことにします．

$$\dfrac{(\alpha \vdash \alpha) \quad \Gamma, \beta^2 \vdash}{\Gamma, \alpha, \alpha \multimap \beta^2 \vdash} P$$

$$\dfrac{(\beta \vdash \beta) \quad \Gamma, \gamma^2 \vdash}{\Gamma, \beta, \beta \multimap \gamma^2 \vdash} Q$$

このとき，同じコミュニケーション環境の仕様 α^2, $(\alpha \multimap \beta^2)^2$, $(\beta \multimap \Gamma^2)^4$ から，たとえば次のような違った3つの証明図が与えられます．そして，その各々が別々のプロセス・スケジュールに(一対一)対応しているわけです．(見やすいように証明木中の論理式は省略して書きます．)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\gamma^8 \vdash}{\quad} Q}{\quad} Q}{\quad} Q}{\quad} Q}{\quad} P}{\alpha^2, (\alpha \multimap \beta^2)^2, (\beta \multimap \gamma^2)^4 \vdash} P$$

上の証明図はプロセススケジュール $PPQQQQ$ に対応している．

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\gamma^8\vdash}\,Q}{}\,Q}{}\,Q}{}\,P}{}\,Q}{\alpha^2,(\alpha\multimap\beta^2)^2,(\beta\multimap\gamma^2)^4\vdash}\,P$$

上はプロセススケジュール $PQPQQQ$ に対応.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\gamma^8\vdash}\,Q}{}\,Q}{}\,P}{}\,Q}{}\,Q}{\alpha^2,(\alpha\multimap\beta^2)^2,(\beta\multimap\gamma^2)^4\vdash}\,P$$

上はプロセススケジュール $PQQPQQ$ に対応.

また，2つ以上の論理推論を同時に（並行して）行うことを許す次のような証明図も考えることができます.

$$\dfrac{\alpha\vdash\alpha\qquad\dfrac{\dfrac{\dfrac{}{\gamma^8\vdash}\,Q}{}\,Q}{}\,Q}{\alpha\vdash\alpha\quad\beta\vdash\beta\quad\beta^3,(\beta\multimap\gamma^2)^3,\gamma^2\vdash}\,P,Q}{\alpha,(\alpha\multimap\beta^2),\beta^2,(\beta\multimap\gamma^2)^4\vdash}\,P$$

そしてこれは並行プロセススケジュール $P(P/\!/Q)QQQ$ に対応します．上の並行プロセススケジュールを線形化（継起化）したのが前の3つのプロセススケジュールのうちの最初の2つ $PPQQQQ$ と $PQPQQQ$ であると見ることもできます．このように並行プロセススケジュールの概念を導入して複数のプロセススケジュール（並行プロセスの継起的表現）を同一視できるのです．しかし，このような同一視によるプロセス表現（または証明）の同値類の中には前の3つのプロセススケジュールのうちの3つ目のものは入らないことが分かります．つまり，3つ目のプロセススケジュールは並行プロセススケジュールのレベルで考えても上の $P(P/\!/Q)QQQ$ とは違ったプロセスなのです．これらをも同一視するレベルのプロセス表示の概念はないでしょうか？ 特に，上の並行プロセススケジュールの中で継起的に表されている部分のうちのあるものは，たまたまこのスケジュールにおいて順序を逆にしたり，さらに並行にしたり変更できるところがあることが分かります．また，もっと並行性を拡張して，できる限り並行にスケジュールできるところを並行に表示する，ということにすることにより，前の $PQQPQQ$ とも同一視できるような並行プロセス表示のレベルも可能であることが分かります．このような，非本質的な継起性の区別を無視して得られるプロセス表現は並行計算の true concurrency model 理論でトレースと呼ばれていますが，実はそれが線形論理のプルーフネット（証明網）概念なのです．すなわち，上のプロセススケジュールや並行プロセススケジュールのレベルで現れた非本質的な継起的表現の区別を同一視したものが，**図-1** のプルーフネットで実現できます．このプルーフネットに対応するプロセス表現のレベルをプロセスのトレース（process trace）と呼ぶことにします．線形論理では，「証明」というと一般にはプルーフネットを指します．つまり，それだけプルーフネットのレベルに視点の中心を置いているわけです．ここではプルーフネットの詳しい説明は省きますが（例えば岡田[3]，Girard[1] を参照して下さい），このプルーフネットという概念こそが線形論理の誕生とともに論理学に導入された全く新しいレベルの「証明の同一性」の基準であり，またこのプルーフネットを用いて並行プロセスにとって非常に本質的な「プロセスの同一性」の基準の1つを与えることができる，ということを注意しておきたいと思います．特に，ここでプルーフネットから，その中に含まれている論理式の表示を消して得られるプルーフネットの型が，並行プロセスの同一性を論理的に考える場合に重要な役割を果たします．並行計算理論では，この同一性が「true concurrent models のトレース」概念に対応するのです.

今の線形論理の定式化のように左辺だけの式（sequent）を用いた式計算体系を考える場合には，公理へ到達して証明のある枝が完成すること

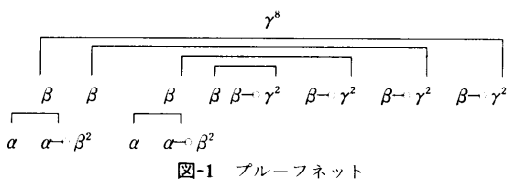

図-1 プルーフネット

(伝統的な意味で，証明検索が停止すること）は対応するプロセスの失敗と考えるのが自然です．つまり公理への到達をプロセスのdead-lockと解釈するわけです．(後編（第2部）でそのような具体例をいくつかあげます．)

証明のfairnessの概念は，証明論や定理自動証明理論（Theorem Proving）や論理プログラミング理論でおなじみのものを考えます．fairnessとは証明構成のstrategy（戦略）についての概念で，大ざっぱにいえば「極端なひいきなしに，環境内の個々の論理式を取り扱う」ということです．

実は証明とプロセス表現とを同一視するとき，証明概念に対する上のfairnessがプロセス（スケジュール）の通常のfairnessに対応することになるのです．

不完全でfairな証明網がdead-lockのないfairなプロセス・トレースに対応するわけです．以後，dead-lockなしでfairなプロセスのことをsafeなプロセスと呼ぶことにします．

線形論理の証明の概念と並行計算モデル（特にMilnerのπ-計算）とのより詳しい関係について，および仕様の論理的整合性と仕様を満たすsafeプロセスの存在との関係については次の完全性定理が成り立ちます．これは古典的なGödelの完全性定理の並行計算版です．

定理1（完全性（岡田）） 仕様Γが線形論理において整合的であることとΓの*safe*な並行プロセスが存在することとは同値である．

ここでΓが整合的とは，Γから矛盾が証明可能でないこと（$\Gamma \vdash \bot$または$\Gamma \vdash$が証明可能でないこと）とします．

以上のような線形論理の証明概念（プルーフネット）と並行プロセス概念とを対応させる枠組をProofs-As-Processesパラダイムと呼ぶことにします．そして，この枠組の中で並行計算理論を線形論理の証明論に還元にて，論理学的手法および概念を用いて並行プロセスの分析を行っていくわけです．特に，並行計算の形式的仕様，Concurrent Logic Programming言語の設計などの直接的な応用のほかに，並行計算理論において現れる種々の最適化問題や計算量の複雑さの問題が証明論の枠組で取り扱われ，現在いろいろな成果がでてきています．

本稿の続編の第2部では，今回の解説の具体的な応用例を与えるとともに，milnerのπ計算体系と線形論理との具体的な対応関係などを解説する予定です．特に，線形論理を並行プロセスの仕様記述言語と見なした場合のKahn's Data Flow Nets, Petri nets, Milnerのπ-計算などの表現による並行プロセスの記述がどのように線形論理を用いた論理的表現で表せるかを，いくつかの例を使って示します．

参考文献

1) Girard, Linear Logic,: Theoretical Computer Science (1987).
2) 岡田：Girardの線形論理とその応用，日本ソフトウェア科学会春期チュートリアル・レクチャー・ノート (1993).
3) 岡田：情報科学のための論理，産業図書 (1996)，およびその続編の「線形論理とその応用」(近刊).
4) 竹内：線形論理入門，日本評論社 (1995).
5) Kobayashi-Yonezawa: Proc. Workshop on Linear Logic and Logic Programming, ed. D. Miller, Univ. Penn. (Nov. 1992).

(平成7年5月19日受付)

岡田　光弘

1954年生．東京大学文学部哲学科卒業．慶応義塾大学文学研究科博士課程修了．文学博士．カナダ・ケベック州立コンコルディア大学情報科学科助教授，準教授，パリ大学（XI校）情報科学研究所（LRI）フランス国家招聘教授，フランス国立科学研究センター（CNRS）マルセイユ離散数学研究所（LMD）客員主任研究員などを歴任．論理学（証明論），哲学，理論情報科学，理論言語学などを専攻．現在慶応義塾大学文学部哲学科教授．著書「情報科学のための論理学」(1996，産業図書)，編著「Special Issue for Linear Logic」(J-Y. Girard, A. Scedrovとの共編)，「ENTCS」(欧州理論情報学会編，Elsevier-North Holland, 1996)，「Advances in Linear Logic Vol. 2」(近刊，Lafont, Regnierと共同編集）など．現在IEEE-Logic in Computer Science組織委員（93年度は大会委員長)，Rewriting Technique and Application組織委員（93年度は大会委員長)，日本科学哲学会編集委員，三田哲学会編雑誌「哲学」編集者など．