並列プログラムの検証体系に関する論理学的研究

村　上　昌　己

並列プログラムの検証体系に関する論理学的研究

村上　昌己

目　次

第１章　まえがき

プログラムの意味を形式的に記述し，取り扱うことは，プログラムを見通しよく設計するために重要である．プログラムの形式的意味記述のためには，適当な計算の数学的モデルが必要である．プログラムあるいは計算を数学的議論の対象として扱う際，数理論理学的な手法を用いる立場はTuring，Churchの計算可能性の理論以来，プログラム図式の理論，Scottのプログラム束の不動点理論等様々な方面で何度か用いられて来ている．特に近年，論理型プログラミング，関係データベースなどに見られるように，実際にユーザによって行われるプログラミングにおいて，論理学的手法の有用性が改めて注目をあつめている．そこで，プログラムの形式的モデルを論理学的対象として与える公理的意味論，即ちプログラム正当性の検証の理論が，意味記述法として有用であると考えられる．

また近年，計算機システムの並列化に伴い，並列プログラムの生産性，信頼性を向上させるための工学的手法として，並列計算の数学的モデルに関する議論の重要性がたかまっている．

しかし並列計算と呼ばれるものは，オペレーティングシステムのように複数のユーザのための資源管理，プロセスの制御等を主に問題とするもの，またスーパーコンピュータ，推論マシンのように計算の高速化のために並列化を行うものなど，その形態，レベルも様々であり，それぞれの場合によって問題の関心は異なったものとなる．したがってその数学的モデルにも様々なものが提案されており，またプログラムの意味記述の方法にもいくつかの種類がある．

本研究の目的は，並列性を含むプログラムの形式的意味記述のための論理学的手法を確立することである．

本論文では，公理的意味論が特に有効であろうと思われる局面として，ユーザレベルでの並列プログラミングを想定し，そこで使用される言語のモデルとして，共有資源型と通信型の二種類のモデルについて，それぞれ形式的セマンティクスを与え，各モデルについて議論の対象となるプログラムの性質の検証のための公理系を与え

る．

さらに，並列プログラムの公理的検証体系について，Aptの完備性(Apt Bergstra Meerteins 79) の概念を拡張し，幾つかの結果を示す．

１．１　並列プログラム

従来から提案されている並列計算の形態およびその数学的モデルについて，次の４つの点から分類することができる．

（１）各プロセスの動作が時間的に同時に起こるという事実を陽にモデル化するか，あるいはひとつ下のレベルでは各動作は逐次的に実行されるものとしてモデル化するか．

即ち，ハードウェアアルゴリズムなどでは各セルの動作は実際に同時におこなわれており，各セル間の入出力の同期などを問題とするときは同時性を陽に表すモデルが必要となる．あるいは脳などの生体における情報処理等も同時性を陽に表すモデル化が適していると考えられる．一方，タイムシェアリングシステム，あるいはファイル管理のプログラムで，実際にヘッドが一つしかないディスク装置などにおさめられたファイルを複数のユーザが共有する場合などは逐次的な動作におとして扱うモデルが適している場合がある．

同時性を陽に表すモデルとしてはＣＳＰ(Hoare 78)のfailure model(Milner 80)，ペトリネット（ベクトル加算系）(Peterson 81)，またセルオートマトン等がある．一方，逐次的動作におとして議論するモデルとしてはフロー表現，イベント表現(Shaw 78) などがある．このように逐次的動作におとして議論するモデルについては，各プロセスをいかなる順序で実行しても，その動作がプログラマの意図したものとなっていることを示す問題，即ち実行の速度独立性が議論の対象となる．

（２）システム全体が無限に続く実行を意図したものか，あるいは有限時間の後に実行を終了してその結果を残すものか．

即ちオペレーティングシステムやコンピュータネットワークなど

は外部からそれなりの干渉がないかぎり，無限に動き続けることを意図している．あるいは生物の遺伝子のプログラムも，次々と世代を交代しながら無限に自己の複製を産み出す並列システムといえる．

このような無限な実行を扱うモデルとしてはω言語（泉　稲垣　本多　昭58）等がある．無限な実行を扱うモデルに関しては次のような点が議論の対象となる．即ち，実行が続く間，そのシステムにある条件が成立ち続けることを示す問題（不変性），また各プロセスについてあるプロセスが不当に実行を長く待たされることがないか(fairness)等である．一方有限時間で終了するプロセスとしては，concurrent prolog (Shapiro 83)等で書かれた問題解決のプログラム，またスーパーコンピュータ等で計算の高速化のために並列化を行ったプログラムなどがある．またoccam(INMOS 84),Concurrent Pascal(Brinch Hansen 77)等のシステム記述言語でも有限時間で終了する並列プログラムを書くことが可能である．このような有限な実行のモデルとしては，相互通信逐次型プロセス系（ＣＳＰ）(Hoare 78)，cobegin-coend 文(Owicki 76) 等がある．これらのモデルについては，実行が終了した後の結果の正しさ，あるいはいかなる入力に対しても実行が有限時間で終了するか否か（停止性）などが議論の対象となる．実行が終了した後の結果の正しさを議論する問題の例としてはBisantine-General Agreement(Lamport Shostak Peasa to appear)の問題がある．

（３）各プロセスの情報の交換は共有資源によるか，通信によるか．

即ちオペレーティングシステムの資源管理，プロセッサの割当て等の仕事に関する問題は本質的に共有資源のモデルで扱われるべきである．Concurrent Pascal(Hansen 77) はモニタ(Andrews Schneider 83)によって共有資源の管理を行う並列プログラム記述言語である．このように共有資源を扱うシステムの問題として，複数のプロセスが同時に同じ資源を不当にアクセスすることを避ける問題，即ち相互排除性，また共有資源をお互いに奪いあうなどして実行が先に進まなくなるようなことはないことを保証する問題（デ

ッドロックフリー性）等が議論の対象となる．また共有資源の割当てに関する問題の例として「哲学者の食事」(Dynning Phirosophor Problem)(Brinch Hansen 73)，「喫煙問題」(Smoking Problem) (Peterson 81)等がある．一方通信型の情報交換を行うシステムについては，ＣＳＰ,Concurrent Prolog(Shapiro 83)のように通信によってプロセス間の同期をも実現する場合があり，通信機能による同期がデッドロックフリー性などを保証できるか否か等が議論の対象となる．通信型の並列システムに関する例題としては，先にのべたBisantine General Agreement(Lamport Shostak Pease, to appear)の問題，またはDrinking Phirosophor Problem(Chandy Misra 84) 等がある．またオブジェクト指向プログラミング (Shapiro Takeuchi 83)などは，すべての動作をプロセス間の通信によって実現するシステムと考えられる．

（４）システム全体のプロセス数は固定されているか，動的に変化するか．

ＴＳＳのようなシステムにおいて，複数のユーザから送られる各タスクをプロセスとみなし，それらを制御するシステムは，各ユーザのlogin,logoutをプロセスの生成，消滅と考えればプロセス数の動的な変化を含むシステムと考えられる．またConcurrent Prologのように並列性と再帰呼び出しを兼ね備えた言語は，必然的にプロセス数の動的な変化を含みうる．一方コンピュータネットワークのようなシステムはハードウェア的に独立した一台の計算機を一つのプロセスと考えることによって，プロセス数の固定された並列システムとして扱うことができる．あるいは同じＴＳＳシステムであっても，各端末機をプロセスと考えた場合は，プロセス数の固定された並列システムの例となる．

フロー表現，イベント表現 (Shaw 78)等はプロセス数の動的変化を含む場合のモデルである．またＣＳＰ等はプロセス数が固定された場合のモデルである．

本論文では，次のような並列プログラムを対象とする．

（1）原則として，プログラムの実行は一つ下位のレベルでは逐次的になっているものと考える．このような考えかたは，複数のプロセスが一つのプロセッサで処理される場合，或いはファイル等を共有する場合などは自然な仮定である．また共有資源を持たず，通信機能によってプロセスが結合されている場合は，通信以外の動作については同時に行われたか，逐次的に行われたかは本質的な問題ではない．

このような並列プログラムはある種の非決定性プログラムとして扱うことができる．

例として，P_1，P_2 の二つのプロセスがそれぞれ，

P_1 ： $x_1 := 0$ ； $y := x_1 + 1$

P_2 ： $x_2 := 1$ ； $y := x_2 + 1$

のようなプログラムであったとするとき，P_1，P_2 を並行に走らせた場合に，たどる可能性のある実行系列は次のようなものとなる．

$x_1 := 0$ ； $y := x_1 + 1$ ； $x_2 := 1$ ； $y := x_2 + 1$

$x_1 := 0$ ； $x_2 := 1$ ； $y := x_1 + 1$ ； $y := x_2 + 1$

⋮

$x_2 := 1$ ； $y := x_2 + 1$ ； $x_1 := 0$ ； $y := x_1 + 1$

すなわちP_1，P_2 を並行に走らせるプログラムは，P_1 とP_2 の実行系列をシャッフルした系列を非決定的に選ぶプログラムであると考える．このように並列プログラムをある種の非決定性プログラムとみなす考えかたは従来からよく見られる方法であり，形式的意味を与える際には有用な方法である．

非決定性プログラムとしての並列プログラムの実行を図式的に見てみると，次のような木として考えることができる．即ち，プログラムは計算機のある内部状態（初期状態）から実行を開始し，ある最小単位となる動作を実行するごとに内部状態を更新していく．この様子は，初期状態を根とし，各節点に状態を持ち，その節点から出る枝に実行された動作が対応するような木として表される．例として上のP_1，P_2 が並行に走るプログラムについて，その実行を木

で表すと次ページ図１．１のようになる．これについては第４章で再び触れる．

並列プログラムの形式的セマンティクスを定義するということは，即ち，与えられた初期状態の集合に対してこのような木の集合またはその一部を抽象化して定義することと考えられる．こうしてプログラムの性質は，そのプログラムから定義される木の性質として議論することができる(Emerson Clarke 80).例えば，実行の速度独立性の問題などは，与えられたプログラムと初期状態の集合から定義されるいかなる木の，正常な実行をあらわすいかなるパスについても，プログラマの意図する条件が成り立つかどうか，という問題となる．

本論文ではこのような木の集合のすべてのパスまたは，すべての有限なパスの集合，及び，すべての初期状態すなわち根と，正常な終了をあらわす状態（葉のうちのあるもの）の対の集合等でセマンティクスを定義する．

本論文の最後の部分の結果は，並列プログラムを非決定性プログラムと考えて，検証体系を与えたことによって得られたものである．

（２）すべてのプロセスは有限時間内に計算を終了し，計算結果を出力するものとする．

この仮定は，本論文では，ユーザプログラムのように何らかのアルゴリズムを実現したプログラムを対象としていることによる．従って，オペレーティングシステムあるいはconcurrent pascal (Brinch Hansen 77)のようなcycle 文を含む言語によって書かれたプログラムのように，無限に動き続けることを意図したシステムを対象にするには不適当であるように思われるかもしれない．しかし，オペレーティングシステムの多くの部分が，有限時間で完了するアルゴリズムの繰り返しからなっていることからわかるように，無限に実行が続くシステムについて議論する場合も，ある時点から有限時間内での動作にのみ注目して考えることも多い（例えば，ある入力が入ってから，その結果が出力されるまで）．また無限に実行が

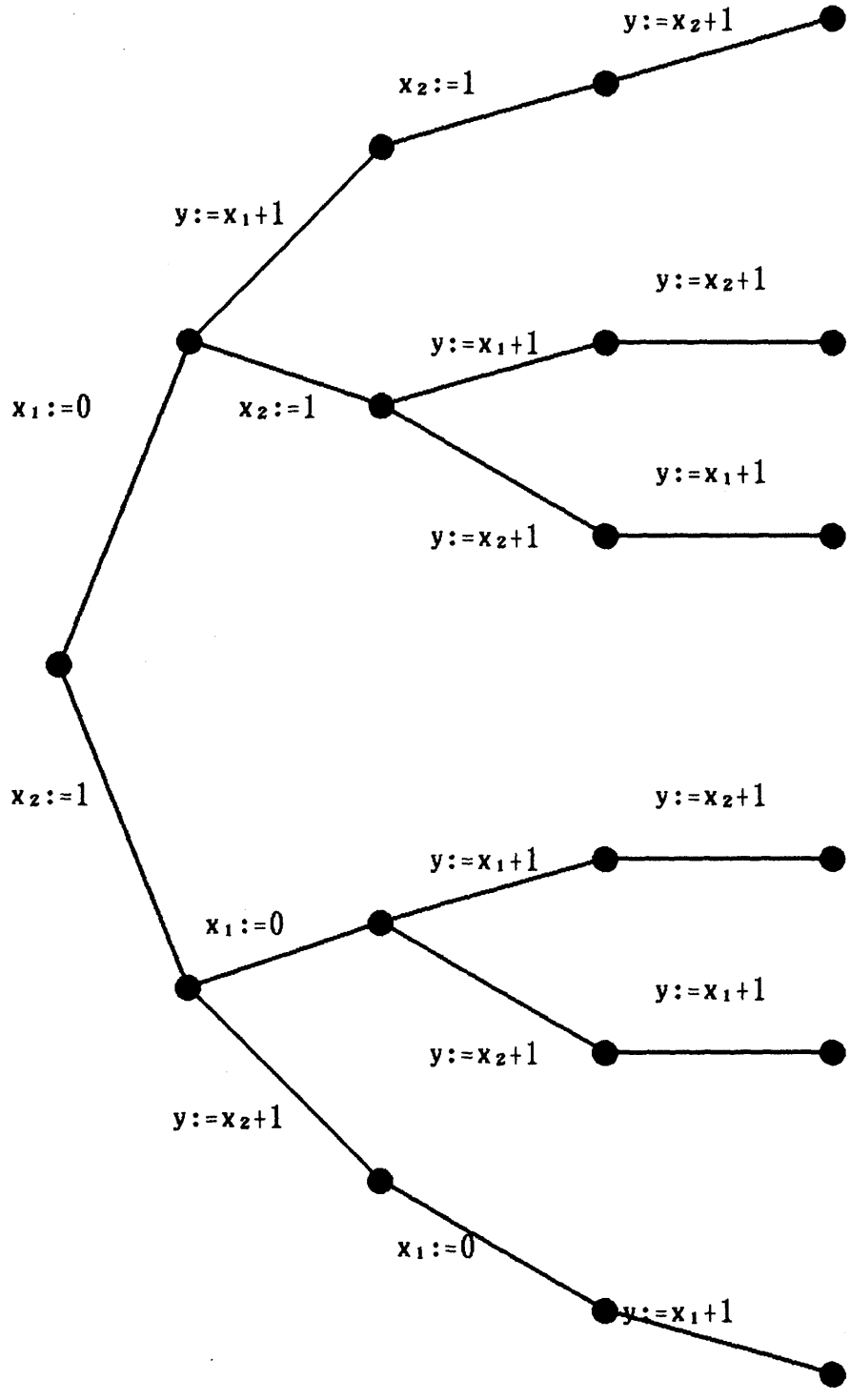

y:=x2+1
x2:=1
y:=x1+1
y:=x2+1
y:=x1+1
x1:=0
x2:=1
y:=x1+1
y:=x2+1
x2:=1
y:=x2+1
y:=x1+1
x1:=0
y:=x1+1
y:=x2+1
y:=x2+1
x1:=0
y:=x1+1

図　1．1　計算の木

続くシステムの多くの性質は，(fairness 等の適当な制限を考慮すれば) 有限時間で停止するプロセスの自然な拡張によって与えられることが多い．したがってこの仮定のもとでも，無限に動き続けることを意図したシステムに関する議論も，ある程度は可能である．

（３）本論文では共有変数型のモデルと，通信型のモデルの両方について，それぞれ形式的な検証体系をあたえる．

共有変数型の並列プログラムのモデルとしては，有限個のプロセスを有限状態のスケジューラによって同期をとるモデル：正規並列プログラム（ＲＧＣ）を導入する．プロセス間の同期機構を有限状態に制限することは，特に強い制限ではない．その根拠は，後に示すように，二値セマフォのような同期機構は有限状態のスケジューラによって記述できることによる．また山下らは（山下　稲垣　本多　昭55）で有限状態のスケジューラをもつ並列プログラム図式（ＣＰＳ）（ＣＳＰではない）を扱っている．ＣＰＳのスケジューラは，各プロセスの命令の一種である通信命令（ＣＳＰの通信命令とは別の機能を持つものである）を入力アルファベットとする有限オートマトンで表される．このような有限状態スケジューラを持つＣＰＳがConcurrent Pascal 風のモニタ機能で同期をとるようなプログラムを記述する能力があることを示し，実際的応用のうえで十分な記述能力があることを主張している．本論文で扱うＲＧＣは本質的に解釈付のＣＰＳと同等である．

また通信型のモデルとしては(Hoare 78)の相互通信逐次型プロセス系（ＣＳＰ）を採用する．ＣＳＰはプロセス間の情報の交換及び同期を，通信命令機能（先述のＣＰＳの通信命令とは異なる）により行うもので，Adaの待ち合わせ(rendezvous)，またはＩＮＭＯＳグループのoccam(INMOS 84)等のモデルとして取り入れられている．

本論文では，いずれのモデルでも並列動作の入れ子(nesting) は許さない．これは主に，理論が煩雑になるのを避けるためである．

（４）本論文で扱うモデルはいずれもプロセス数は有限個に固定されたものである．この点について，Brinch Hansenは，次のよう

に述べている(Brinch Hansen 77)．即ち，プロセスの動的な生成，抹消はプログラミング言語の意味付けや言語処理系の設計を複雑にするのみで，その長所を十分に引き出す場合はまれである．

Concurrent Prolog は並列性と再帰呼び出しを兼ね備えた言語でプロセスの動的な生成，抹消を含むが，このような論理型の言語については，本論文で扱うような手続き型の言語に対する公理的意味論とは別の異なる意味記述法を導入すべきと考えられる．

即ち，本研究で対象とする並列プログラムは通常のユーザプログラムのように，何等かのアルゴリズムを実現したようなものを想定している．

なお，本論文では，並行(concurrency) と並列(parallelism) の区別は特に行わず，いずれの呼びかたをされるものに対しても並列プログラムという用語を主に用いる．

１．２　プログラムの正当性検証の理論

プログラムの正当性検証の理論は，１９６０年代にＦloyd(Floyd 67) によって始められ，Hoare(Hoare 69)によって論理体系として確立された．その後，Hoare論理では扱えないプログラムの停止性等や，非決定性プログラム，再帰呼出を含むプログラム，また無限に実行が続くプログラム等をも議論の対象とするため，様相論理を基にした論理体系(Pnueli 77,Manna Pnueli 79,Harel 80)，あるいは述語変換系を基にした研究(Flon Suzuki 81)等が行われてきた．

これらの研究は当初は，プログラマ自らが設計し実現したプログラムが，プログラマが意図した仕様を満たすことを，自動的にあるいは会話的手法によって証明するためのシステムの開発を目標としていたが，数学的，技術的限界から，そのようなシステムが現実的でないなどの批判があった．また「プログラムの正しさの数学的証明」の，プログラムの信頼性を保証するものとしての有効性を疑問視する意見もあり(De Millo Lipton Perilis 79)，暫くはプログラムの正当性検証の理論は，その理論の意義について議論があった．

しかし，近年ソフトウェアの設計，開発の上でプログラムの形式的意味記述の重要性が認識されるようになり，またプログラミングにおける論理的アプローチの有効性が様々な形で認められるに至り，プログラム検証の理論の研究は，その意義を広く認められつつあるといえる．

プログラムの形式的意味記述の手法としては，プログラムの不動点理論から発展した表示的意味論（中島　昭57,Benson 82），ウィーン定義法のような操作的意味論(Lucas 81)，アトリビュートグラマ等の他に，抽象データ型を用いた方法に代表される代数的意味記述法（北　坂部　稲垣　本多　昭58）等がプログラミング言語及びその処理系の設計，実現への応用等を目的として研究されている．しかし，これらの手法はすべての局面において有効であるとは言えない．例えば，既に存在するプログラミング言語によって実現されたプログラムの働きをユーザに理解させる方法として，その言語の代数的意味記述のみから理解させようとする試みは適当とは思われない．即ち，代数的手法は，言語の設計者，インプリメンタにとっては優れていても，それだけではユーザに対して不親切であると考えられる．　プログラムの正当性検証の理論は，その研究開始当時の目的から，比較的ユーザに近い局面での応用に適した意味記述法であるといえる．即ちEuclid(Lampson et al 77)，ＩＯＴＡ(Nakajima　Yuasa Kojima 80)，Clear(Bustall Goguen 81)，Clu(Liskov Snyder Atkinson 77) などの検証性の高いプログラミング言語や抽象データ型の理論を基にした開発用ツールへの応用，プログラミングの初級者に正しいプログラムを構造的に作成する方法の教育(Alagic Arbib 78) などの応用の多くはプログラマのための試みといえる．またプログラムの証明から，ドキュメントを作成したり，プログラムリストにコメントつける際の参考とする等の面でも，他の意味記述法よりもユーザには利用しやすい．さらに，プログラムの中で使用したデータ型の性質を，抽象データ型を用いて記述してあった場合，プログラムの正当性の検証に抽象データ型の公理を

用いることも可能である(Nakasima Honda Nakahara 80)．即ち，代数的意味記述法も公理的意味記述法と組み合わせて用いることにより，より多くの局面で有効となるであろう．

したがって，プログラムの正当性の検証の理論は，特にユーザに近い局面での応用を考えて，議論すべきと考える．

例えばプログラムの設計，検証が容易であるようなプログラミング言語とその言語に対する検証システム，及び通常の高級プログラム言語へのコンパイラ等をもつプログラミング環境を考える等の応用は有効と思われる．

プログラムの検証の理論で最もよく知られているのは，Hoareの部分的正当性検証体系(Hoare 69)である．Hoareの体系では，プログラムPが条件Φを満たす入力に対し，もし実行が停止すれば，出力結果が条件Ψを満たすことを．Pが入力条件Φと出力条件Ψに関して部分的正当であるといい，

$$\Phi \ \{P\} \ \Psi$$

という式であらわす．これを論理式として公理と推論規則から形式的に証明することによって，Pの正当性を検証するのがHoareの方法である．この方法は言わばプログラムの計算の結果のみについての形式的議論であり，プログラムが暴走することはないか(停止性)，あるいは計算の途中で配列の添字が指定された範囲をこえたりしないか等，プログラムが正常に終了するか否かについて，即ち計算の制御については議論の対象とはならない．後に，プログラムの停止性または，再帰呼出を含むプログラム，手続き呼出の際の引き数の受け渡し，go to 文なども扱えるよう拡張された体系(Alasic Arbib 78) などが報告された．

一方，非決定性を含むプログラムについてその計算結果及び計算の制御について検証を行うために，Hoareの体系の拡張ではなく始から別の論理体系を用いる方法も提案された．即ち，様相論理をとりいれた方法として時制論理（Temporal Logic）(Pnueli 77)，

拡張されたHoareの体系と様相論理を結びつけたダイナミック論理（DynamicLogic）(Harel 79)，また述語変換系(predicate transformer) を用いることによりプログラムが正常に結果を出すための最弱前条件(weakest pre-condition) を求める方法(Flon Suzuki 81) である．

時制論理では通常の論理記号の他にG，F等の論理記号を用いる．これは様相論理の□，◇にあたるもので直観的にはGΨとは，ある時点から先は常にΨが成立ち続けることをいう．この記法を用いると，プログラムPのΦとΨに関する部分的正当性Φ｛P｝Ψは，Pの終了した時点にいることを$halt_P$ で表すことにすると，

$$\Phi \supset G(halt_P \supset \Psi)$$

によってあらわされる．

またダイナミック論理では□，◇の論理記号にプログラムPを対応させ，［P］Ψ，<P> Ψといった記法を用いる．これらの意味はそれぞれ「Pの終わった時点ではいつもΨが成り立つ」，「Pの終わった状態でΨの成り立つものが少なくとも一つ存在する」である．これらの記法を用いると，Φ｛P｝Ψは

$$\Phi \supset [P]\Psi$$

によって表される．

また最弱前条件では，プログラムPと出力条件Ψから，Pが停止して結果がΨを満たすために入力が満たすべき最も弱い条件をPとΨに関する関数wpの値と考え，wp(P，Ψ)　のように表す．即ち，

$$\Phi \supset wp(P, \Psi)$$

はΦ｛P｝ΨかつPが停止することを意味する．

これらの記述，検証の手段のお互いの関係はHarel(Harel 80)がダイナミック論理と他の体系との関連について論じている．また(Majster-Cederbaum 80)でもwpによる意味論と他の意味論との関係が議論されている．

並列プログラムの検証の体系のうちのいくつかは非決定性プログラムの検証体系に関する結果をもとにしたものである．非決定性プ

ログラムの性質が，そのプログラムが作る木の性質として表現されることは先に述べた通りである．事実，非決定性プログラムの検証に関する研究のいくつかは，その目的として並列プログラムの検証をあげている(Pnueli 77)．即ち並列プログラムの性質は，それを非決定性プログラムとみなした時に，その非決定性プログラムの実行によってつくられる木の性質として議論される．しかし，実行によってつくられる木の性質を表現する言語は，形式的検証にはそのままは使えない．プログラムの検証に用いられる言語は，プログラムを実行する以前に得られるもの，即ちプログラムの構文を用いて表現する言語でなければならず，なおかつ，推論規則は構文の構造にそって順次適用できるものでなければならない．以下にあげた検証体系はいずれもプログラムの構文をもちいて，その性質を記述し，推論するような体系となっている．

並列プログラムの意味論，正当性検証体系に関する研究として，共有資源型，通信型共に，Hoareの公理系の拡張(Owicki Gries 76, Lamport 80,Apt Francez De Roever 80,Gerth De Roever 84, Lamport Schneider 84,Soundararajan 84)，様相論理(Pnueli 77, Abrahamson 79,Manna Pnueli 79,Manna Pnueli 81,Manna Pnueli 83)，述語変換系(Flon Suzuki 81,Elrad Francez 82)を用いる方法が提案されている．これらの体系のうち，Hoareの体系を拡張したものとしては，通信型の並列プログラムに対しては(Apt Francez De Roever 80,Lamport Schneider 84,Soundararajan 84) 等がある．これらはいずれもプログラムのモデルとしてＣＳＰを採用している．また共有変数型の並列プログラムに対しては，(Owicki Gries 76,Lamport 80,Apt 81) 等がある．Hoareの体系は先に述べたように，計算の結果のみに着目した性質の検証体系であったため，これを並列プログラムに拡張したとき，次のような点が問題となる．即ち並列プログラムの正しさとして特に関心がもたれるデッドロックフリー性，共有資源の相互排除性などの計算の制御に関する性質についての検証が出来ないという点である．この点については様々

な工夫がなされ，(Owicki Gries 76) に見られるようにＣＳＰ，条件付臨界領域のような特定の制御方式については計算の制御に関する性質についての検証も一応可能である．

しかしさらに次のような問題点が残されている．上に述べた体系では，例えば，$P_1, P_2, \cdots\cdots, P_n$ というｎ個のプロセスが並列に走るプログラム

$$P_1 \parallel P_2 \parallel \cdots\cdots \parallel P_n$$

に対して，入力条件Φと出力条件Ψに関する部分的正当性の検証は次のようにして推論される．即ち，ある述語の列

$$\Phi_1, \Phi_2, \cdots\cdots, \Phi_n, \Psi_1, \Psi_2, \cdots\cdots, \Psi_n$$

に対し，各プロセスごとにある性質

$$\Phi_1 \{ P_1 \} \Psi_1$$
$$\Phi_2 \{ P_2 \} \Psi_2$$
$$\vdots$$
$$\Phi_n \{ P_n \} \Psi_n$$

を証明し，かつ$\Phi_1, \Phi_2, \cdots\cdots, \Phi_n$にある関係が成り立ち，かつ$\Psi_1, \Psi_2, \cdots\cdots, \Psi_n$にある関係が成り立つとき，

$$\bigwedge \Phi_i \{ P_1 \parallel P_2 \parallel \cdots\cdots \parallel P_n \} \bigwedge \Psi_i$$

が導かれ，そこから，

$$\Phi \{ P_1 \parallel P_2 \parallel \cdots\cdots \parallel P_n \} \Psi$$

を結論する，といった手順をふむ．

この方法では，プログラムの構文を各プロセスに分ける推論規則を与えている．Hoareの公理系は，プログラムをbegin-end,while do, if then 等の単独でまとまったモジュールの組合せ，即ち構造化された構文を用いて記述することによって，意味記述が簡潔になるというインスタンスと考えられる．しかし並列動作は意味記述の簡潔さのためより，高速化等の実現の都合上生じた構造と考えられるため，並列に走るプロセスの組合わさったプログラムの各プロセスをモジュールとして，全体を構造化された構文とみなすにはモジュール間の結合が複雑でありまた強すぎると考えられる．その結果，

$\Phi_1, \Phi_2, \cdots\cdots, \Phi_n$ に成り立つある関係（例えば$\Phi_1, \Phi_2, \cdots\cdots, \Phi_n$が共有変数を含まない等）のような付帯条件が数多く付くなど推論規則が複雑となる．このような体系のもとでの証明は，推論の各ステップがプログラムの各部分の直観的な意味を理解するために，証明全体をみわたす必要がある等ドキュメントとして読みづらい等の問題点を含む．Hoareの体系の証明が優れている点として，プログラムの働きを，実行時の動きを逐次追うことなく，その構文上からスタティックに理解することができるという点があげられるが，このようにプログラムの見かけ上の構造にとらわれた証明ではその長所は十分に生かされない場合が多い．Lamportは
Hoareの記法の意味を変更して，ある程度プログラムの動的な面が推論規則に現れるような体系をＣＳＰ(Lamport Schneider 84)，及び共有資源型(Lamport 80)のプログラムについてあたえている．

一方，時制論理を用いる体系(Manna Pnueli 83,Manna Pnueli 79)，被防護命令プログラムに対する最弱前条件を用いる体系(Flon Suzuki 81) 等は，プログラムの実行をある程度ステップワイズに追う形の証明を行うものといえる．即ち，並列プログラムは，動作的には非決定性プログラムとして扱うことが可能であることから，並列プログラムの検証を非決定性プログラムの検証として行う考えかたである．このような体系の特徴として，その適用範囲の広さがあげられる．即ち，並列プログラムを一旦非決定性プログラムに変形してから検証を行うため，公理と推論規則は非決定性プログラムに関するものを与える．従って，いかなる同期メカニズムを使う並列プログラムも，非決定性プログラムへの変形手続きさえ与えられれば，公理と推論規則は共通のもので十分である．さらに時制論理を用いる方法ではプログラムのデッドロックフリー性，相互排除性などの計算の制御に関する性質についても，直接的に表現し検証することが可能である．また被防護命令プログラムの性質は，実行時につくられる実行系列の木に関する性質として形式的に与えられれば，不動点を用いた表現に機械的に変換可能であること(Emerson

Clarke 80），及びその不動点が単調な関数の最大あるいは最小不動点で与えられれば容易に公理系が得られること(Flon Suzuki 81)より，最弱前条件による方法も表現能力に関しては強力である．

しかし，これらの方法はいずれもプログラムの変形を伴うため，既にある言語で実現されたプログラムのドキュメントとしては不適当であり，また会話型の半自動証明などにも向かないといえる．すなわち，時制論理による方法等は，先に述べたようなプログラムの設計，検証のための言語とその言語に対する検証システムのほかに，通常の並列プログラム言語へのコンパイラをも用意した環境への応用を考える場合に有効と思われる．Wolper のダイナミック論理に基づくスケジューラの合成法(Wolper 82) 等はその種の試みとみなすことも可能である．また，被防護命令プログラムに対する最弱前条件を用いる体系は，純粋に証明論的な議論の対象として考えるのが適当と思われる．

本論文では，ユーザが並列性を含むプログラムを設計するために適した言語を想定し，Hoare流の公理系の並列版の短所を補った検証体系，即ち非決定性プログラムへの変形を含むことなく，計算の制御に関する性質の検証も可能で，かつプログラムの見かけの構造にとらわれず，できあがった証明からプログラムの動作を追うことが容易であるような検証体系を与える．

即ち，まず共有資源型並列プログラムのモデルとして，正規並列プログラム（ＲＧＣ）を導入し，その検証体系をダイナミック論理の拡張として与える．次に，通信型の並列プログラムのモデルとしてＣＳＰを採用し，centralized approach(Elrad Francez 82)による検証体系を与える．

最後に，並列プログラムを非決定性プログラムとして検証を行う体系の証明論的性質としてAptの完備性(Apt 81)の概念を拡張して得られた結果をいくつか示す．

第２章　正規並行プログラムの検証体系

本章で扱う共有変数型の並列プログラムは次のようなものである．

①．すべてのプロセスは，分岐，連接，及び脱出条件が書かれた繰り返しの組合わさった構造により，計算を行う．

②．プロセス数は動的に変化することはなく，有限個に固定されている．

③．各プロセス間の同期の条件は，許される実行系列の集合を表す正規表現として与えられる．

並列プログラムの正当性として示すべき性質には次のようなものが考えられる．

(1)　速度独立部分的正当性…いかなる順序で実行されても計算結果は出力条件をみたす．

(2)　停止性…どのプロセスも無限ループに陥ることはない，

(3)　デッドロックフリー性…計算が途中で行き止まりになってしまうことはない．

(4)　相互排除性　…複数のプロセスが同時に臨界領域にはいることはない．

本章では，正規プログラム（ＲＧ）の形で書かれた非決定性プログラムの検証体系として(Harel 79)が提案した第一階ダイナミックロジック（ＲＧ－ＤＬ）を拡張し，上記の(1)，(3)，(4)　についての検証体系を与え，その無矛盾性及び完全性を示す．すなわち，ＲＧを拡張し，上記①，②，③を満たす並列プログラムのモデル（ＲＧＣ）を導入し，また論理記号といくつかの公理，推論規則を補うことによって，新しく論理体系ＲＧＣ－ＩＬを提案し，その無矛盾性と完全性を証明する．

２．１　正規並行プログラム

本節では上に述べたような条件①，②，③を満たす並列プログラムを記述できる言語として正規並行プログラム（ＲＧＣ）を定義する．このＲＧＣは，(Harel 79)によって導入された非決定性プログ

グラムの記述言語である正規プログラム（ＲＧ）の拡張として次のように定義される．

２．１．１ＲＧＣのシンタックス

プログラムに現れる変数記号をx, y, …等で，関数記号をf, g, …等であらわす．引数を持たない関数記号を定数記号という．それらから作られる項をτとし変数をｘとするとき，

$$x := \tau$$

を割当文という．

また，量記号のない述語論理式Ｐについて，

Ｐ？

を判定文という．代入文と判定文を基本命令という．

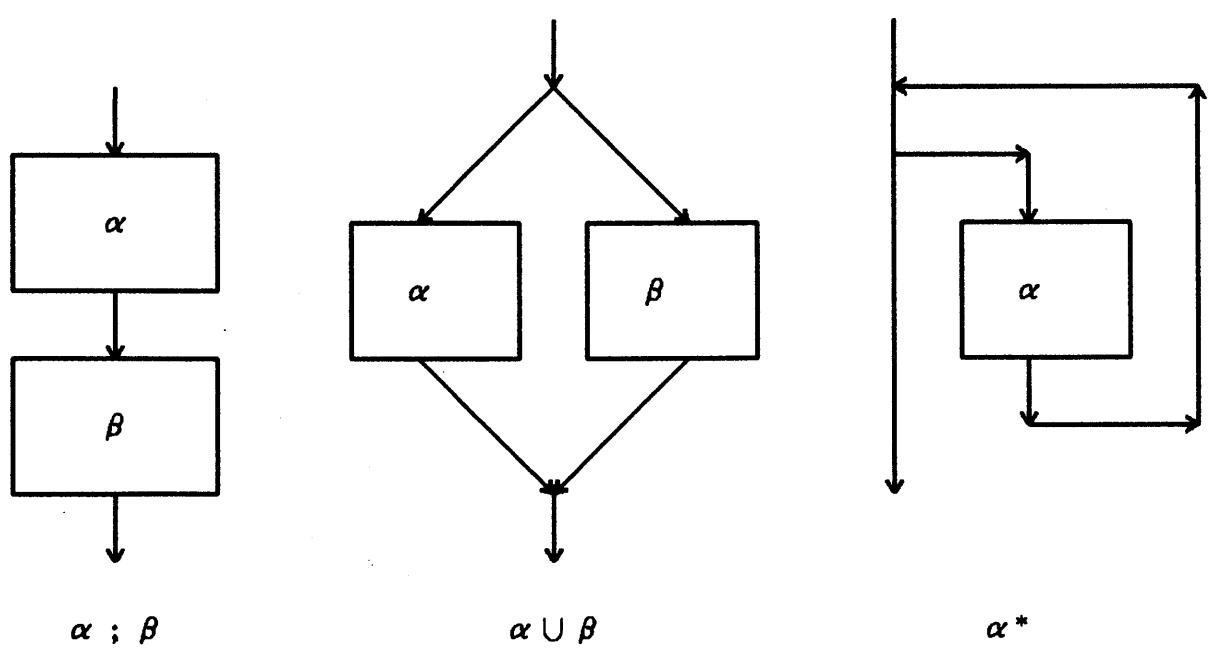
プログラムのシンタックスは，そのプログラムの実行系列を表す正規表現で与えられる．即ち，

〔定義２．１〕基本命令の集合Σの上の正規表現を正規プログラム（ＲＧ）という．即ち，

（ｉ）ε，ϕ，および基本命令はＲＧである．

（ii）α，βがＲＧなら，$\alpha ; \beta$，$\alpha \cup \beta$，α^* はＲＧである．

演算記号の結合の強さの順序は＊，；，$\cup$とする．必要に応じて括弧(,) を用いることもある．以下では正規表現とそれが表す系列の集合とを同一視する．直観的には$\alpha ; \beta$，$\alpha \cup \beta$，α^* はそれぞれ次の図２．１のような非決定性のフローチャートを表す．



$\alpha ; \beta$　　$\alpha \cup \beta$　　α^*

図２．１　ＲＧに対応するフローチャート

この表現を使うと

分岐，　　if P then A else B，

連接，　　A ; B，

繰り返し，while P do A

はそれぞれ，

$$P?;A \cup \neg P?;B,$$

$$A;B,$$

$$(P?;A)^{*};\neg P?$$

によって表すことができる，((Harel 79)ページ２１参照)

〔定義２．２〕$\alpha_1,\cdots,\alpha_n,\sigma$がＲＧのとき，次の形の表現

$$(\alpha_1 \parallel \alpha_2 \parallel \cdots \parallel \alpha_n) \cap \sigma_1 \cap \sigma_2 \cap \cdots \cap \sigma_n$$

を並列構造（ＰＳ）という．ＲＧならびにＰＳを正規並行プログラム（ＲＧＣ）という．

ここで，$\parallel$および$\cap$は，それぞれ，シャッフル演算，ならびに共通集合の演算を表す．よく知られているように，正規集合のクラスは，$\parallel$でも$\cap$でも閉じているので，ＰＳの表す集合もやはり正規集合である．

２．１．２　ＲＧＣのセマンティクス

ＲＧＣのセマンティクスは，領域$\mathfrak{D}$とその上での解釈を指定することによって定まる．まず状態の概念を導入する．

〔定義２．３〕空でない領域$\mathfrak{D}$が指定されているものとする．このとき，各関数，各述語および各定数の$\mathfrak{D}$上での解釈と変数のとる$\mathfrak{D}$上の値の組を状態という．以下では，関数，述語，定数の解釈を一つに固定して議論を進めるので，単に変数のとる値の組を状態という．以下で，$i \models Q$は，状態ｉで述語Ｑが成り立つことを表す．また，以下では，$i \not\models Q$と書いて，状態ｉで述語Ｑが成立しないことを表す．また任意の状態ｉで述語Ｑが成り立つとき単に$\models Q$と書く．

〔定義２．４〕ユニバースＵは，状態の集合である．

〔定義２．５〕ＲＧＣαに対して，ユニバースＵ上の関係$m(\alpha)$を以下のように定義する．

（ｉ）まず，基本命令に対してはつぎのように定義する．ｘ:=τに対しては，

$$m(x:=\tau)=\{(i,\ j)\mid i\in U,\ \text{かつ},\ j\text{は，状態}i\text{でのτの値が存在するときその値をｘに代入して得られる状態．}\}.$$

また判定文Ｑ？に対しては，

$$m(Q?)=\{(i,\ i)\mid i\in U,\ i\models Q\}.$$

（ii）次に，基本命令の系列ｗに対しては，ｍ（ｗ）を次のように定義する．

ｗ＝ε（空系列）に対しては，

$$m(\varepsilon)=\{(i,\ i)\mid i\in U\}.$$

ｅを基本命令とするとき，ｗ≡ｕ；ｅに対しては，

$$m(w)=m(u)\bigcirc m(e)$$

ここに，○は関係の合成を表す．

（iii）ＲＧＣαに対しては，

$$m(\alpha)=\bigcup_w m(w)$$

ここで，$\bigcup_w$ はαに含まれる各ｗについての和集合をあらわす．

〔定義２．６〕任意のｋ≧０について，e_1，…，e_k …，e_n を基本命令，i_0，…，i_k…，i_nを状態とする．

このとき０＜ｋ≦ｎについて，

$$(i_{k-1},i_k)\in m(e_k)$$

を満たすならば，

$$<\varepsilon,i_0><e_1,i_1>\cdots<e_n,i_n>\ \text{ないしは}$$

$$<\varepsilon,i_0><e_1,i_1>\cdots<e_n,i_n><\perp>$$

を計算履歴という．ただし⊥は計算がそれより先に進めないことを表す記号である．

〔定義２．７〕計算履歴ｕ，ｖを，

$$u=<\varepsilon,i_0><e_1,i_1>\cdots<e_n,i_n>$$

$$v=<\varepsilon,i_0'><e_1',i_1'>\cdots<e_m',i_m'>$$

とする．これらの連接uvは，$i_n=i_0'$ のときにのみ定義され，

$u\ v = \langle \varepsilon, i_0\rangle\langle e_1, i_1\rangle \cdots \langle e_n, i_n\rangle\langle e_1', i_1'\rangle \cdots \langle e_m', i_m'\rangle$

とする．

〔定義２．８〕 I（$\subset U$）を状態の集合，αをＲＧＣとする．このとき，αとＩに対して，計算履歴集合Ｈ（α，Ｉ）を次のように定義する．

(i) まず基本命令については次のように定義する．

割当文 $x := \tau$ については，

$$H(x := \tau, I) = \{\langle \varepsilon, i\rangle\langle \perp\rangle \mid i \in I,\ \text{かつ}\ (i, j) \in m(x := \tau)\ \text{となる}\ j\ \text{が存在しない．}\} \cup \{\langle \varepsilon, i\rangle\langle x := \tau, j\rangle \mid i \in I,\ (i, j) \in m(x := \tau)\} \cup \{\langle \varepsilon, i\rangle \mid i \in I\}.$$

判定文Ｑ？については，

$$H(Q?, I) = \{\langle \varepsilon, i\rangle\langle \perp\rangle \mid i \in I,\ i \not\models Q\} \cup \{\langle \varepsilon, i\rangle\langle Q?, i\rangle \mid i \in I,\ i \models Q\} \cup \{\langle \varepsilon, i\rangle \mid i \in I\}.$$

（ii）次に基本命令の系列ｗについては次のように定義する．

$w = \varepsilon$については，

$$H(\varepsilon, I) = \{\langle \varepsilon, i\rangle \mid i \in I\}.$$

次に，ｕを基本命令の系列，ｅを基本命令とするとき，

$w = u ; e$に対して

$$H(u ; e, I) = H(u, I)\ H(e, \{j \mid \exists i \in I, (i, j) \in m(u)\}) \cup H(u, I)$$

ここで，$H(u, I)\ H(e, \{j \mid i \in I, (i, j) \in m(u)\})$は計算履歴の連接をその集合に自然に拡張したものである．

（iii）一般に，ＲＧＣαに対しては，

$$H(\alpha, I) = \bigcup_w H(w, I)$$

Ｈ（α，Ｉ）は，Ｉに含まれる状態からαを実行したときにたどる計算履歴の集合である．なお定義からわかるようにそれにはαの

プレフィクスを実行したときの計算履歴も含まれることに注意されたい．　以上の定義より次の補題は容易に証明される．

〔補題２．１〕α，βを任意のＲＧとし，Ｉ（$\subset$Ｕ）を状態の集合とする．このとき

（ｉ）$H(\alpha \cup \beta, I) = H(\alpha, I) \cup H(\beta, I)$

（ii）$H(\alpha ; \beta, I) = H(\alpha, I) \cup H(\alpha, I)\ H(\beta, I')$

ここに，$I' = \{j \mid \exists i \in I \wedge (i,j) \in m(\alpha)\}$

（iii）$H(\alpha^*, I) = H(\alpha, I) \cup H(\alpha^*, I)\ H(\alpha, I'')$

ここに，$I'' = \{j \mid \exists i \in I \wedge (i,j) \in m(\alpha^*)\}$

（iv）$H(\alpha \cap \beta, I) = H(\alpha, I) \cap H(\beta, I)$

（ｖ）ＲＧγが存在して$H(\alpha \| \beta, I) = H(\gamma, I)$　□

<u>例</u>　αを，

```
(x> 1 ? ; (even(x)?;x:=x/2 ∪
                odd(x)?;x:=3x+1))* ;x=1?
```

とするとき，

Ｉ＝｛ｉ｜ｉ⊨ｘ＝５｝に対する計算履歴集合Ｈ（α，Ｉ）は，

```
{< ε , x=5 >,
 < ε , x=5 >< ⊥ >,
 < ε , x=5 >< x> 1 ?, x=5 >,
 < ε , x=5 >< x> 1 ?, x=5 >< ⊥ >,
 < ε , x=5 >< x> 1 ?, x=5 >< odd(x)? , x=5 >,
 < ε , x=5 >< x> 1 ?, x=5 >< odd(x)? , x=5 >
                          < x:=3x+1, x = 1 6 >, …}
```

となる．

２．２　ＲＧＣで表現される並行プロセス

前章で導入したＲＧＣは，第２章のはじめにで与えた条件①，②，③を満たす様な並列プログラムをモデル化する能力があることはその定義から理解できる．ここでは，その有効性について考えるために，従来考えられてきたいくつかの共有資源ないしは共有変数に基

礎をおく同期方式のうち２値セマフォと条件付臨界領域（ＣＣＲ）をＲＧＣで記述する方法を示す．

ＲＧＣでは，有限固定個数の並行に走るプロセスP_1，P_2,…，P_n はＲＧ，α_1,α_2,…，α_n のシャッフル

$$\alpha_1 \parallel \alpha_2 \parallel \cdots \parallel \alpha_n$$

で表され，それらの同期は，ＲＧσ_1,σ_2,…，σ_m の積

$$\sigma = \sigma_1 \cap \sigma_2 \cap \cdots \cap \sigma_m$$

で表現されている．すなわち$\alpha_1 \parallel \alpha_2 \parallel \cdots \parallel \alpha_n$ は，ｎ個のプロセスが並行に走るときに，考えることのできるすべての実行系列を表しており，そのなかからσに含まれる系列だけを可能な実行系列として許すことによって，並行プロセスの同期を実現している．このように，許される系列の集合ないしはそれらが満たすべき条件を与えることによって同期条件を表すという考えかたは，（山下，稲垣，本多　昭５５）の有限状態スケジューラを有する並列プログラム図式（ＣＰＳ）や(Campbell,Harberman 74) のパス式（path expression)などで採用されている．特に，パス式は，共有資源に対する操作の正しい系列の集合を規定するものであり，操作の名前の系列の集合を表している．このような意味で，ＲＧＣは，パス式による同期機構の自然なモデルであるといえよう．

共有変数に基礎をおく同期機構として，パス式の他に，セマフォ（semaphore），条件付臨界領域(conditional critical region，ＣＣＲと略記する），モニタ(monitor) などが考えられている(Andrews, Schneidr 83)．これらの同期機構のうち，２値セマフォ及びＣＣＲは次のように表現することができる．

〔１〕２値セマフォ　２値セマフォによって制御される実行系列は，「Ｐ操作が実行されるとＶ操作の実行後でなければ次のＰ操作は実行されない」という性質で特徴づけられる．問題にしているＰ操作，Ｖ操作で指定される臨界領域に含まれる命令の集合をΣ_cで，また，Σ_cの命令とＰ操作，Ｖ操作以外の命令の集合をΣとすれば，上の性質を持つ命令系列の集合は，

$$(\Sigma^* P (\Sigma \cup \Sigma_c)^* V)^*$$

というＲＧで表現される．なお，Ｐ，Ｖ操作は，セマフォ変数をｘとすれば，値0,1 のｘへの代入ｘ：＝０，ｘ：＝１によって実現される．

〔２〕条件付臨界領域（ＣＣＲ）共有資源Ｒを有するｎ個のプロセスP_1，P_2,…，P_nからなる並列プログラムを考える．各プロセスP_iが共有資源Ｒに関して，

$$\underline{\text{with}}\ \ R\ \ \underline{\text{when}}\ \ B_i\ \underline{\text{do}}\ \ S_i$$

というＣＣＲを含むものとする．ここに，B_iは条件文，S_iは命令系列である．まず上記のプロセスP_iの共有資源Ｒに関するＣＣＲの部分は

$$(\neg B_i\ ?)^*\ \ (B_i\ ?\ ;\ x:=0\ ;\ S_i\ ;\ x:=1)$$

と表されるので，P_iのその部分をこのように置き換えて得られるＲＧをα_iとする．ここにｘはＲに関する２値セマフォ変数である．

さて，ＣＣＲを含む上記の並列プログラムをＲＧＣで表現するには，共有資源Ｒに関する相互排除の条件を表すＲＧσ_1と通常要求される公平なプロセスの実行を保証する条件を表すＲＧσ_2を構成し，

$$(\alpha_1 \parallel \alpha_2 \parallel \cdots \parallel \alpha_n) \cap (\sigma_1 \cap \sigma_2)$$

とすればよい．

まず，σ_1は，次のように構成される．

$$\begin{aligned}\sigma_1 = (\ \Sigma^*\ ((&B_1\ ?\ ;\ x:=0\ ;\ \Sigma^*\ ;\ s_{11};\\ &\Sigma^*\ ;\ s_{12};\ \cdots,\ \Sigma^*\ ;\ s_{1k1};\Sigma^*\ ;\\ &\qquad\qquad x:=1)\ \cup \cdots\\ &\cup\ (B_n\ ?\ ;\ x:=0\ ;\ \Sigma^*\ ;\ s_{n1}\ ;\\ &\qquad \Sigma^*\ \cdots\Sigma^*\ ;\ s_{nkn}\ ;\ \Sigma^*\ ;\ x:=1)^*\ ;\\ &\Sigma^*\ )^*\end{aligned}$$

ここで，Σは，ＣＣＲに含まれない命令の集合である．また，各S_iは簡単に$S_i = s_{i1}\ ;\ s_{i2}\ ;\ \cdots s_{iki}$（ｉ＝１，…，ｎ）のような基本命令の系列であるとしている．S_iに∪や＊が含まれる場

合にも，上記の構成法は容易に拡張できる．

次に，σ_2 については，待ち行列として機能する（山下，稲垣，本多　昭 55)で用いられた形の有限状態スケジューラを構成し，この有限状態スケジューラで定まる有限オートマトンの受理する集合を表す正規表現としてσ_2 をもとめればよい．待ち行列のとり得る状態の数は，プロセス数がｎ個に固定されているので，たかだか，ｎ個からｋ個（$0 \leq k \leq n$）選ぶ順列の数の合計である．ｎ個のプロセス$P_1, P_2, \cdots P_n$ に対して，有限状態スケジューラの状態集合Ｑを次のように定義する．

$Q = \{ \langle \varepsilon \rangle \} \cup \{ \langle i_1 i_2 \cdots i_k \rangle \mid 1 \leq k \leq n,\ i_1 i_2 \cdots i_k$ は
１からｎまでの整数からｋ個選んで得られる順列$\}$．

ここで，$\langle i_1 i_2 \cdots i_k \rangle$ は，プロセスが$P_{i1}, P_{i2} \cdots, P_{ik}$の順に待ち行列に並んでいることを示している．また$\langle \varepsilon \rangle$ は，待ち行列が空であることを示している．

さて，待ち行列が$\langle i_1 i_2 \cdots i_k \rangle$ であるとき，そこに含まれないプロセスP_m が$\neg B_m$ ？を実行して待ち行列に加わると，待ち行列は，$P_{i1}, P_{i2} \cdots, P_{ik}, P_m$ となる．一方待ち行列が$P_{i1}, P_{i2} \cdots, P_{ik}$であるとき，その先頭の$P_{i1}$が$B_{i1}$？；ｘ：＝０を実行して臨界領域にはいると待ち行列は$P_{i2} \cdots, P_{ik}$になる．従って，有限状態スケジューラの状態遷移関数δは次のように定められる．

$\delta(\langle i_1 i_2 \cdots i_k \rangle, \neg B_m ?) = \langle i_1 i_2 \cdots i_k\ m \rangle$
（但し，$1 \leq m \leq n$かつ$m \neq i_1, \cdots, m \neq i_k$）
$\delta(\langle i_1 i_2 \cdots i_k \rangle, B_{i1} ?) = \langle i_2 \cdots i_k \rangle$
$\delta(\langle \varepsilon \rangle, \neg B_m ?) = \langle m \rangle$
$\delta(\langle m \rangle, B_m ?) = \langle \varepsilon \rangle$
$\delta(q, e) = q$

（但し$q \in Q$，かつｅは$\neg B_m ?$，$B_m ?$以外の命令．）

ＲＧＣσ_2 を構成するには，この有限状態スケジューラで，初期状態を$\langle \varepsilon \rangle$，最終状態集合を$\{ \langle \varepsilon \rangle \}$として定まる有限オートマトンの受理する集合を表す正規表現をもとめればよい．

２．３　検証体系ＲＧＣ－ＩＬ

並列プログラムの正当性として議論すべき性質としては，先に述べた(1)速度独立部分的正当性，(2)停止性，(3)デッドロックフリー性，(4)相互排除性がある．本章では，ダイナミックロジック(Harel 79)の体系ＲＧ－ＤＬを拡張して，上記の(1)，(3)，(4)の証明が可能な論理体系を与える．

ここで論理式Ｒの証明が可能とは，論理体系に含まれる公理から推論規則だけを用いてＲを推論できることである．Ｒが証明可能であることを⊢Ｒであらわす．

２．３．１　ＲＧ－ＤＬの概要

まず，Harelの非決定性プログラムの検証体系であるＲＧ－ＤＬ(Harel 79)について，その概要を述べる．

〔定義２．９〕ＲＧ－ＤＬの論理式を次のように定義する．

（ｉ）述語記号ｐと項$\tau_1, \cdots, \tau_n$から作られる素命題

$$p(\tau_1, \cdots, \tau_n)$$

はＲＧ－ＤＬの論理式である．

（ii）論理式Ｐ，Ｑ，ＲＧα，と変数ｘに対して，

$$\neg P,\ P \vee Q,\ \exists x P,\ [\alpha] P$$

は，いずれもＲＧ－ＤＬの論理式である．また，

$$\neg [\alpha] \neg P$$

を

$$\langle \alpha \rangle P$$

と略記する．その他に，

$$\neg P \vee Q,\quad \neg (\neg P \vee \neg Q),\quad \neg (\exists x \neg P(x))$$

をそれぞれ，

$$P \supset Q,\ P \wedge Q,\ \forall x A(x)$$

と略記する．

〔定義２．１０〕ＲＧ－ＤＬの論理式の真偽は次のようにして決められる．

（ｉ）素命題$p(\tau_1, \cdots, \tau_n)$について，

$i \models p(\tau_1, \cdots, \tau_n)$ iff

「状態 i で $p(\tau_1, \cdots, \tau_n)$ の解釈の結果が真．」

(ii) $i \models A \vee B$ iff 「$i \models A$ または $i \models B$．」

$i \models \neg A$ iff 「$i \models A$ が成り立たない．」

$i \models \exists x A(x)$ iff 「ある x が存在して $i \models A(x)$．」

(iii) P が RG－DL の論理式，α が RG のとき，

$i \models [\alpha] P$ iff 「$\forall j (i,j) \in m(\alpha)$ に対して $j \models P$．」

直観的には $[\alpha] P$ は α のすべての実行に対してもそれが終了すれば必ず P が成立することを，また $\langle \alpha \rangle P$ は α が停止して P が成立するような実行のしかたがすくなくとも一つ存在することをあらわす．

以上のセマンティクスのもとで，次の公理系 RG－DL が完全かつ無矛盾であることが，(Harel 79)で示された．

〔公理〕

命題論理におけるすべての公理(例えば，(細井 昭49) 138 頁参照) と次の 4 つの公理($\leftarrow$R),(?R),(;R)及び($\bigcup$R)．

($\leftarrow$R) $[x := \tau] P \equiv P\langle x / \tau \rangle$，

ここに，P は述語論理式，$P\langle x / \tau \rangle$ は P における変数 x のすべての自由な出現に τ を代入したもの．

(?R) $[Q?] P \equiv Q \supset P$

(;R) $[\alpha ; \beta] P \equiv [\alpha] [\beta] P$

($\bigcup$R) $[\alpha \bigcup \beta] P \equiv ([\alpha] P \wedge [\beta] P)$

〔推論規則〕

(MP) $\dfrac{P \qquad P \supset Q}{Q}$

(G)

$$\frac{P \supset Q}{[\alpha] P \supset [\alpha] Q} \qquad \frac{P(x) \supset Q(x)}{\exists x P(x) \supset \exists x Q(x)}$$

(*I) (*C)

$$\frac{P \supset [\alpha] P}{P \supset [\alpha^*] P} \qquad \frac{P(n+1) \supset \langle \alpha \rangle P(n)}{P(n) \supset \langle \alpha^* \rangle P(0)}$$

2．3．2　RG－IL

前節で述べたＲＧ－ＤＬに新たな論理記号$\overline{\ }$をつけ加える．こうして拡張された論理体系をＲＧ－ＩＬと呼ぶ．

〔定義２．１１〕ＲＧ－ＩＬの論理式は次のような式である．

（ｉ）ＲＧ－ＤＬの論理式はＲＧ－ＩＬの論理式である．

（ii）αがＲＧ，Ｐ，ＱがＲＧ－ＩＬの論理式であるとき，

$$\overline{\alpha}\,P,\quad [\alpha]\,P,\quad P \wedge Q,\quad \neg P$$

はＲＧ－ＩＬの論理式である．

〔定義２．１２〕ＲＧ－ＩＬの論理式$\overline{\alpha}\,P$の真偽は次のように定める．

$$i \models \overline{\alpha}\,P \text{ iff } 「\forall u \in H(\alpha, \{i\}) \text{ について，}$$
$$Last(u) \models P \text{ あるいは } Last(u) = \perp .」$$

ここで，

$$Last(u) = \begin{cases} j\ ;\ u = \langle \varepsilon, i\rangle \cdots \langle e, j\rangle \text{ のとき．} \\ \perp\ ;\ u = \langle \varepsilon, i\rangle \cdots \langle e, j\rangle\langle\perp\rangle \text{ のとき．} \end{cases}$$

直観的には，$\overline{\alpha}\,P$は，αの実行中は常にＰが成り立ち続けることをあらわす．

ＲＧ－ＩＬの公理系を次のようにあたえる．

〔公理系ＲＧ－ＩＬ〕

〔公理〕ＲＧ－ＤＬの公理ならびに次の５つの公理

（←Ｉ）　$\overline{x := \tau}\,P \equiv [x := \tau]\,P \wedge P$

（？Ｉ）　$\overline{Q\,?}\,P \equiv Q \supset P$

（$\cup$Ｉ）　$\overline{\alpha \cup \beta}\,P \equiv \overline{\alpha}\,P \wedge \overline{\beta}\,P$

（；Ｉ）　$\overline{\alpha\ ;\ \beta}\,P \equiv \overline{\alpha}\,P \wedge [\alpha]\,\overline{\beta}\,P$

（＊Ｉ）　$\overline{\alpha^*}\,P \equiv [\alpha^*]\,\overline{\alpha}\,P$

〔推論規則〕ＲＧ－ＤＬの推論規則．

〔定理２．１〕ＲＧ－ＩＬは無矛盾である．

（証明）ＲＧ－ＩＬの推論規則は，ＲＧ－ＤＬのそれと同じであるから，新たに付け加えた（←Ｉ），…，（＊Ｉ）の５つの公理が定義２．

１２のセマンティクスのもとで真であることを示せば十分である．

（←Ｉ）について；

まず，

$$i \models [x := \tau]\, P \wedge P \text{ ならば } i \models \overline{x := \tau}\, P$$

を示す．

$$(i,\ i') \in m(x := \tau)$$

なるｉ’が存在する場合には，

$$H(x := \tau,\ \{i\}) = \{<\varepsilon, i>, <\varepsilon, i><x := \tau, i'>\}$$

となる．いま

$$i \models [x := \tau]\, P \wedge P$$

であるから，定義２．１２より

$$i \models \overline{x := \tau}\, P$$

が成立する．一方

$$(i,\ i') \in m(x := \tau)$$

なるｉ’が存在しない場合は，

$$H(x := \tau,\ \{i\}) = \{<\varepsilon, i>, <\varepsilon, i><\perp>\}$$

であり，$i \models P$より，やはり定義２．１２によって

$$i \models \overline{x := \tau}\, P$$

が成立する．

逆に

$$i \models \overline{x := \tau}\, P$$

ならば，

$$i \models P$$

かつ

$$(i,\ i') \in m(x := \tau)$$

なるｉ’で

$$i' \models P$$

であるから，

$$i \models [x := \tau]\, P \wedge P$$

が成立する．

（？Ｉ）については，（←Ｉ）と同様にその妥当性が証明できる．

（$\cup$Ｉ）について；

$$i \models \langle\alpha\rangle P \wedge \langle\beta\rangle P$$

ならば

$$i \models \langle\alpha \cup \beta\rangle P$$

の証明．

任意の

$$u \in H(\alpha \cup \beta, \{i\})$$

について，

$$Last(u) = \perp \quad あるいは \quad Last(u) \models P$$

であることを示す．

補題２．１（ｉ）より，次の２つの場合を考えればよい．

$u \in H(\alpha, \{i\})$ の場合，

$$i \models \langle\alpha\rangle P$$

より

$$Last(u) = \perp \quad あるいは \quad Last(u) \models P$$

となる．

$u \in H(\beta, \{i\})$ の場合も同様である．

逆に，

$$i \models \langle\alpha \cup \beta\rangle P \quad ならば \quad i \models \langle\alpha\rangle P$$

となることは，

$$H(\alpha, \{i\}) \subseteq H(\alpha \cup \beta, \{i\})$$

より明らか．

$i \models \langle\beta\rangle P$ についても同様である．

（；Ｉ）について；

$$i \models \langle\alpha\rangle P \wedge [\alpha]\langle\beta\rangle P$$

ならば

$$i \models \langle\alpha ; \beta\rangle P$$

の証明．

補題２．１（ⅱ）より，任意の

$$u \in H(\alpha ; \beta, \{i\})$$

について，

$$Last(u) = \perp \quad あるいは \quad Last(u) \models P$$

であることを示す．まず，

$$u \in H(\alpha, \{i\})$$

の場合には，

$$i \models \overline{\alpha} P$$

より，明らかに

$$Last(u) = \perp \quad あるいは \quad Last(u) \models P$$

となる．一方，

$$u \in H(\alpha, \{i\}) H(\beta, \{j \mid (i,j) \in m(\alpha)\})$$

の場合は，u_1，u_2 が存在し

$$u = u_1 u_2,$$

$$u_1 \in H(\alpha, \{i\})$$

かつ

$$<\varepsilon, j> u_2 \in H(\beta, \{j \mid (i,j) \in m(\alpha)\})$$

となる．今，

$$i \models [\alpha] \overline{\beta} P$$

より，

$$(i, j) \in m(\alpha)$$

なる任意の j で

$$j \models \overline{\beta} P$$

である．

従って，

$$Last(u_2) \models P \quad または \quad Last(u_2) = \perp.$$

また

$$u = u_1 u_2$$

であるから

$$Last(u) = Last(u_2).$$

従って，

$$Last(u)=\perp \quad \text{あるいは} \quad Last(u) \models P$$

となる．逆に，

$$i \models \boxed{\alpha ; \beta} P$$

ならば

$$i \models \boxed{\alpha} P \wedge [\alpha] \boxed{\beta} P$$

であることは以下のように示される．仮定より，任意の

$$u \in H(\alpha ; \beta, \{i\})$$

に対して，

$$Last(u)= \perp \quad \text{または} \quad Last(u) \models P$$

である．まず，

$$H(\alpha ; \beta, \{i\}) \supset H(\alpha, \{i\})$$

より，

$$i \models \boxed{\alpha} P$$

がいえる．また，

$$i \models [\alpha] \boxed{\beta} P$$

が成り立たないとすると，

$$u' \in H(\alpha, \{i\})$$

が存在し，

$$(i, Last(u')) \in m(\alpha)$$

かつ

$$Last(u') \not\models \boxed{\beta} P .$$

すると，ある

$$u'' \in H(\beta, \{Last(u'')\})$$

について，

$$Last(u'') \neq \perp \quad \text{かつ} \quad Last(u'') \not\models P .$$

ここで，u'，u''に対して，

$$Last(u'\ u'') = Last(u'')$$

より

$$u'\ u'' \in H(\alpha ; \beta, \{i\})$$

について，

$$Last(u'\ u'') \neq \perp \text{かつ} Last(u'\ u'') \not\models P$$

となり，これは

$$i \models \overbrace{\alpha\ ;\ \beta}\, P$$

に反する．

（＊Ｉ）について，

$$i \models [\alpha^*]\ \overbrace{\alpha}\, P$$

ならば

$$i \models \overbrace{\alpha^*}\, P$$

の証明を以下に示す．補題２．１（iii）より，まず

$$u \in H(\alpha^*, \{i\})\ H(\alpha, \{j \mid (i,j) \in m(\alpha^*)\})$$

の場合には，ｕは

$$u = u'\ u''$$

のように分けられて，

$$u' \in H(\alpha^*, \{i\})$$

かつ

$$u'' \in H(\alpha, \{j \mid (i,j) \in m(\alpha^*)\}).$$

ここで，

$$i \models [\alpha^*]\ \overbrace{\alpha}\, P$$

より，

$$(i, j) \in m(\alpha^*)$$

なる任意のｊについて

$$j \models \overbrace{\alpha}\, P.$$

ゆえに，

$$Last(u') \models \overbrace{\alpha}\, P.$$

従って，

$$u'' \in H(\alpha, \{j \mid (i,j) \in m(\alpha^*)\})$$

より

$$Last(u) = \perp \quad \text{あるいは} \quad Last(u) \models P.$$

また，

$$u \in H(\alpha, \{i\})$$

の場合は，

$$(i, i) \in m(\alpha^*)$$

より

$$i \models \boxed{\alpha} P$$

となり，任意のuについて

$$Last(u) = \perp \quad \text{あるいは} \quad Last(u) \models P$$

は明らかである．逆に，

$$i \models \boxed{\alpha^*} P \text{ ならば } i \models [\alpha^*]\boxed{\alpha} P$$

であることは，次のように証明される．

$$H(\alpha^*, \{i\}) \supset H(\alpha^*; \alpha, \{i\})$$

より，

$$i \models \boxed{\alpha^*} P \text{ ならば } i \models \boxed{\alpha^*; \alpha} P$$

である．ゆえに，(;I) より，

$$i \models [\alpha^*]\boxed{\alpha} P .$$

（証明終）

〔定理２．２〕ＲＧ－ＩＬは完全である．

（証明）ＲＧ－ＩＬの任意の論理式は，新たに付加した　５つの公理(←I)，…，(*I) を用いて，それと同値なＲＧ－ＤＬの論理式に変換できることを示す．$\boxed{\ }$で囲まれている最も大きいＲＧαの構成に関する帰納法で証明する．

(i) αがＱ？または x ：＝τのときには，それぞれ公理(?I)，(←I)を用いれば，同値な$\boxed{\ }$を含まない論理式に変形できることは明らかである．

（ii）αが$\alpha_1 \cup \alpha_2$，$\alpha_1;\alpha_2$ または　α_1^* のときには，それぞれ公理(⋃I)，(;I) および(*I)と，帰納法の仮定とを用いて，$\boxed{\ }$を含まない式に変形できる．　よって，ＲＧ－ＤＬは完全であることが示されているので(Harel 79)，ＲＧ－ＩＬも完全となる．

（証明終）

２．３．３　ＲＧＣ－ＩＬ

前節で述べたＲＧ－ＩＬをＲＧＣに拡張し，ＲＧＣ－ＩＬを与え

る．

〔定義２．１３〕ＲＧＣ－ＩＬの論理式を次のように定義する．

（ｉ）ＲＧ－ＩＬの論理式は，ＲＧＣ－ＩＬの論理式である．

（ii）αをＲＧＣ，またＰ，ＱをＲＧ－ＩＬの論理式とするとき，

$$Q \supset \boxed{\alpha} P$$

はＲＧＣ－ＩＬの論理式である．

・論理式の意味はＲＧ－ＩＬの場合と同様に定義される．

先に述べた並列プログラムの部分的正当性，デッドロックフリー性，相互排除性はＲＧＣ－ＩＬの論理式を用いて次のように記述される．

Ａ．部分的正当性：αを$(\alpha_1 \parallel \cdots \parallel \alpha_n) \cap \sigma$とする．各プロセス$\alpha_i$の最後に，$halt_i$をブール値をとる変数として

$$halt_i := true$$

という割当文を付け加えて，各プロセスを，

$$\alpha_i ; halt_i := true$$

とすれば，プロセス全体が停止に至っていることは，

$$\wedge_i (halt_i = true)$$

で表される．

従って，αが，入力条件Ｑと出力条件Ｐについて部分的正当であることは，

$$Q \supset \boxed{\alpha} (\wedge_i (halt_i = true) \supset P$$

と表される．

Ｂ．デッドロックフリー性：e_1，…，e_mをＲＧＣαに含まれる命令とすると，αのデッドロックフリー性は，

$$\boxed{\alpha} ((\vee_{i=1,m} < e_i > true) \vee (\wedge_{i=1,n} halt_i))$$

と表現される．

Ｃ．相互排除性：プロセスP_1，P_2のそれぞれの臨界領域に，セマフォ変数x_1，x_2を対応させる．すなわちi=1,2 についてP_iが臨界領域内の命令を実行中であるとき

$$x_i = 0$$

となる．このとき，入力条件をQ，スケジューラをσとすると，相互排除の条件は，

$$Q \supset \overbrace{P_1 \parallel P_2 \cap \sigma} \quad (\neg (x_1 = 0 \wedge x_2 = 0)$$

と表せる．

ＲＧＣをαとし，ＲＧ－ＤＬの論理式をＰ，Ｑとすると以上の性質はいずれも，ＲＧＣ－ＩＬの論理式$Q \supset \overbrace{\alpha} P$の形をしている．

従って，以下で導入するＲＧＣ－ＩＬの公理系はその無矛盾性と完全性より，並列プログラムの正当性として示すべき性質のうち上記Ａ，Ｂ，Ｃの三つがＲＧＣ－ＩＬの論理体系のもとで形式的に証明可能であることがわかる．

〔公理系　ＲＧＣ－ＩＬ〕

〔公理〕ＲＧ－ＩＬの公理．

〔推論規則〕

ＲＧ－ＩＬの推論規則に，次の３つの推論規則（$\parallel$I)，($\cap$I)，(CI)をつけ加える．

($\parallel$I)

（ＲＧα_i（$1 \leq i \leq n$）の任意の命令ｅについて）

$$\frac{P \supset [e] P}{P \supset \overbrace{\alpha_1 \parallel \cdots \parallel \alpha_n} P}$$

($\cap$I)

$$\frac{\overbrace{\alpha} P \qquad \overbrace{\beta} Q}{\overbrace{\alpha \cap \beta} (Q \wedge P)}$$

(CI)

$$\frac{\overbrace{\alpha} P \qquad P \supset Q}{\overbrace{\alpha} Q}$$

〔定理２．３〕ＲＧＣ－ＩＬは無矛盾である．

(証明)($\cap$I)の妥当性については，補題２．１の（iv）より明らかである．また（ＣＩ）についても定義より明らかである．

($\parallel$I)が妥当であることは以下のように示される．いまｉ⊨Ｐであるとする．Ｈ（$\alpha_1 \parallel \cdots \parallel \alpha_n$，｛ｉ｝）のすべての要素について，その長さに関する帰納法を用いる．

（ｉ）仮定より，$<\varepsilon, i>$のｉで，ｉ⊨Ｐとなる．

（ii）帰納法の仮定として，

$H(\alpha_1 \| \cdots \| \alpha_n, \{i\})$ の長さn以下の任意のuについて，

$$Last(u) \neq \perp \text{ならば } Last(u) \models P$$

であるとする．

このとき，長さn+1 の

$$v \in H(\alpha_1 \| \cdots \| \alpha_n, \{i\})$$

について

$$Last(u) \neq \perp$$

であるとすると，長さnの

$$u \in H(\alpha_1 \| \cdots \| \alpha_n, \{i\})$$

が存在して

$$v = u\langle e, j\rangle$$

と書き表わされ，帰納法の仮定から

$$Last(u) \models P.$$

そこでP⊃［e］Pであるからj⊨P．　　（証明終）

〔定理２．４〕ＲＧＣ－ＩＬは完全である．

（証明）ＲＧ－ＩＬの完全性より，αが

$\alpha_1 \| \cdots \| \alpha_n$　あるいは　$\alpha_1 \cap \alpha_2$

の場合に，ＲＧＣαとＲＧ－ＩＬの論理式Ｑ，Ｐについて

$$Q \supset \boxed{\alpha} P$$

が成立するならば，その証明が可能であることを示せば十分である．

任意のＲＧ－ＤＬの論理式について，それと同値な自然数上の述語が存在すること(Harel 79)，および定理２．１，２．２より，任意のＲＧ－ＩＬの論理式に対して，それと同値な自然数上の述語が同様に存在する．従って，Ｑ，Ｐは述語論理式であるとして差し支えないので，以下ではそのように仮定する．

（ｉ）αが$\alpha_1 \| \cdots \| \alpha_n$の場合．

$$\models Q \supset \boxed{\alpha} P$$

ならば，

$$Q \supset R,\ R \supset P$$

であり，かつαに含まれる任意の基本命令eに対して，

$$R \supset [e]\, R$$

である述語Rの存在をしめす．

述語Xに対して状態の集合 $\{i \mid i \models X\}$ を $\pi(X)$ と書くことにする．また，sp（e，X）でXが成立する状態で基本命令eが実行可能ならば，それを実行した後で成立する最強条件，すなわち最強後条件（strongest post condition）をあらわす．

そこで R_k を次のようにおく．

$$R_1 = Q$$

$$R_{k+1} = R_k \vee \bigvee_i sp(e_i, R_k)$$

ただし，e_i はαに含まれる基本命令，$\bigvee_i$ はすべての基本命令についての論理和をあらわす．

明らかに，

$$R_k \supset R_{k+1}$$

である．また，

$\pi(R_k) = \{ Last(u) \mid u \in H(\alpha, \pi(Q)), Last(u) \neq \perp$
かつuの長さがk以下．$\}$

である．

$\{R_k \mid k \geq 0\}$ の最小上界をRとすると，

$\pi(R) = \{ Last(u) \mid u \in H(\alpha, \pi(Q)), Last(u) \neq \perp \}$

であることを以下のように示すことができる．

明らかに，

$$\models Q \supset R,$$

ゆえにRG－ILの完全性より，

$$\vdash Q \supset R \quad \cdots\cdots (1)$$

また，$\models Q \supset \boxed{\alpha} P$ より

$$\pi(R) \subset \pi(P)$$

したがって，$\models R \supset P$．ゆえに，

$$\vdash R \supset P \quad \cdots\cdots (2)$$

また，$i \models R$ なる任意のiについて，ある

$$u \in H(\alpha, \pi(Q))$$

が存在して，

$$Last(u) = i$$

とすることができる．ここで，$(i,j) \in m(e)$ なる j を考えると，

$$u\langle e,j\rangle \in H(\alpha, \pi(Q))$$

である．ゆえに，

$$j \in \pi(R)$$

言い替えると

$$j \models R.$$

したがって，

$$\models R \supset [e]R$$

となり

$$\vdash R \supset [e]R$$

が得られる．

ここで(‖I)により，

$$\vdash R \supset \boxed{\alpha} R.$$

式(1)，(2)及び，(CI) により，

$$\vdash Q \supset \boxed{\alpha} P.$$

（ii） α が $\alpha_1 \cap \alpha_2$ の場合．

$$\models Q \supset \boxed{\alpha} P$$

を仮定する．

（i）の場合と同様にして，α_1，α_2 について R^1，R^2 をつくる．ただし，i=1,2 に対して

$\pi(R^i) = \{ Last(u) \mid u \in H(\alpha_i, \pi(Q)), Last(u) \neq \perp \}$．

すると，

$$\models Q \supset \boxed{\alpha_i} R^i .$$

(⋂I)により，

$$\vdash Q \supset \boxed{\alpha_1 \cap \alpha_2} (R^1 \wedge R^2) \quad \cdots\cdots\cdots\cdots\cdots\cdots (3)$$

$$H(\alpha_1 \cap \alpha_2, \pi(Q)) = H(\alpha_1, \pi(Q)) \cap H(\alpha_2, \pi(Q))$$

であるから，

$$R \equiv R_1 \wedge R_2$$

とすると，

$$\pi(R) = \{ Last(u) \mid u \in H(\alpha_1 \bigcap \alpha_2, \pi(Q)) \}$$

を満たす．

$$\models Q \supset \boxed{\alpha} P$$

より，

$$\pi(R) \subset \pi(P),$$

ゆえに

$$\vdash R \supset P.$$

式(3)および(CI)により，

$$\vdash Q \supset \boxed{\alpha} P.$$ （証明終）

2．4　証明例

本節では，ＲＧＣ－ＩＬによるデッドロックフリー性の記述と証明の例を示す．よく知られた「５人の哲学者の食事」の問題を，２人の哲学者の場合にして考える．P_1，P_2 の２つのプロセスが F_1，F_2 という２つの資源を共有しているものとする．P_1，P_2 はいずれも F_1，F_2 を同時に使用して処理 e_1，e_2 を行って停止する．F_1，F_2 はいずれも P_i (i=1,2) に確保されると，e_i が終了するまで解放されることはないものとする．

このような並行処理系はＲＧＣでは次のように表現される．各 F_i に対して変数 f_i を対応させる．f_i は一種のセマフォ変数であるが，次のように動作を決めることにより通常の動作をするセマフォよりＲＧＣでの扱いが容易になる．すなわち f_i は０，１，２いずれかの値をとり，$f_i = 0$ のとき F_i は解放されており，$f_i = 1$，２のときは，F_i はそれぞれ P_1，または P_2 に確保されていることをあらわす．

P_j が F_i を確保，解放するという操作は，表２．１，２．２のよ

うな演算p_j, v_j をf_i にほどこすこととして表現される．表２．１，２．２で／は値が定義されないことを表す．P_j を記述するＲＧＣは次のようになる．

$$(f_1 := p_j(f_1); f_2 := p_j(f_2) \cup$$
$$f_2 := p_j(f_2); f_1 := p_j(f_1)) ;$$
$$e_j ;$$
$$(f_1 := v_j(f_1); f_2 := v_j(f_2) \cup$$
$$f_2 := v_j(f_2); f_1 := v_j(f_1)) ;$$
$$(\mathrm{halt}_j \leftarrow \mathrm{true})$$

σはＲＧで記述すると長くなるので，有限状態の同期機構として図２．２に示す．この機械の初期状態s_1 を受理状態として，それが受理する言語の正規表現を求めるとそれがＲＧσである．

表２．１　ｐ演算

f_i	j=1	j=2
0	1	2
1	／	／
2	／	／

表２．２　ｖ演算

f_i	j=1	j=2
0	／	／
1	0	／
2	／	0

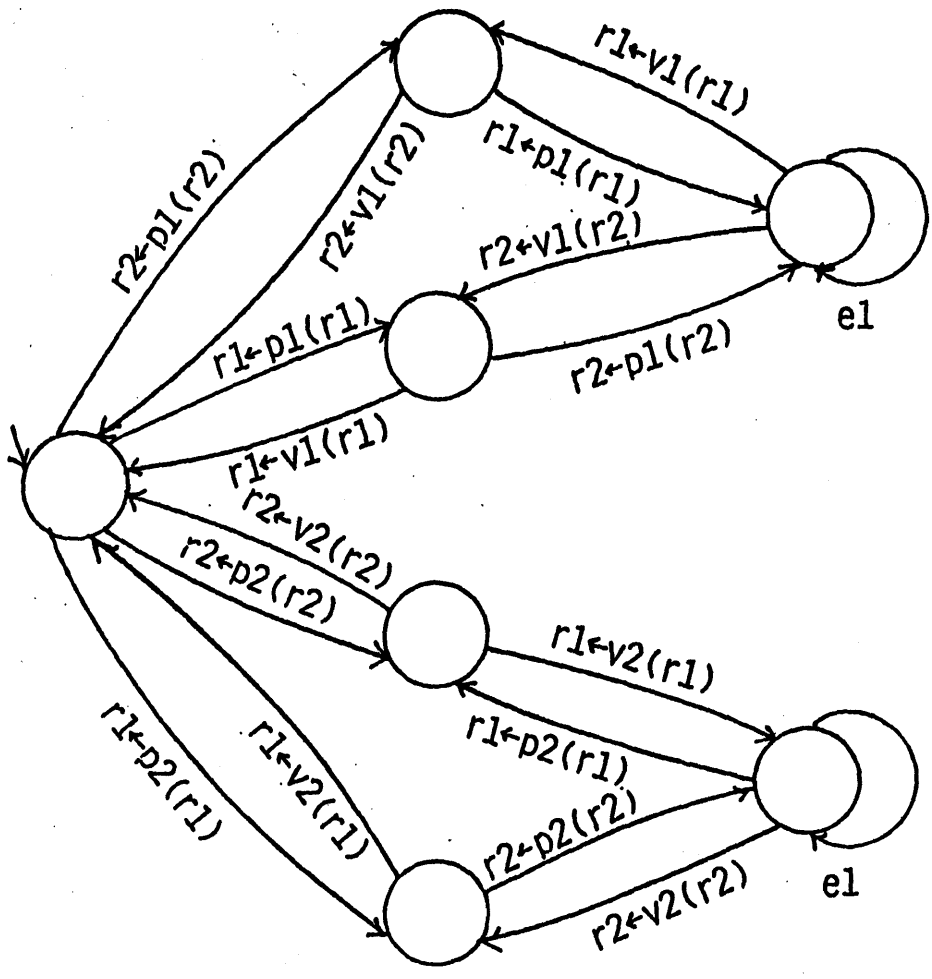

r1←v1(r1)
r1←p1(r1)
r2←p1(r2)
r2←v1(r2)
r2←v1(r2)
e1
r1←p1(r1)
r2←p1(r2)
r1←v1(r1)
r2←v2(r2)
r2←p2(r2)
r1←v2(r1)
r1←p2(r1)
r1←p2(r1)
r1←v2(r1)
r2←p2(r2)
e1
r2←v2(r2)

図２．２　スケジューラ　σ

ここで，m，n = 0,1,2 に対して

$f_1 = m \wedge f_2 = n$ を（m，n）と書くことにすると次式が成立する．

$$\vdash (0,0) \supset \overbrace{\sigma}\,(\neg(1,2) \wedge \neg(2,1)) \cdots\cdots\cdots\cdots\cdots\cdots\cdots (4)$$

ここで I_j を

$$I_j \equiv ((0,0) \wedge (0,j) \wedge (j,0) \wedge (j,j)) \supset (\bigvee_s < a_{js} > \mathrm{true} \vee \mathrm{halt}_j)$$

とする．　ここで a_{js} は P_j に含まれる基本命令であるとすると，任意の s，j について，

$$\vdash (I_1 \wedge I_2) \supset [a_{js}]\ (I_1 \wedge I_2) \cdots\cdots\cdots (5)$$

式(5)および（‖ I）により，

$$\vdash (I_1 \wedge I_2) \supset \overbrace{P_1 \parallel P_2}\ (I_1 \wedge I_2)$$

f_1，f_2 のとりうる値が0，1，2であることより，

$I_1 \wedge I_2$ は次のようになる．

$$(\neg(1,2) \vee \neg(2,1)) \supset (\bigvee_s < a_{1s} > \mathrm{true} \vee \bigvee_s < a_{2s} > \mathrm{true} \vee (\mathrm{halt}_1 \wedge \mathrm{halt}_2))$$

一方，

$$\vdash (0,0) \supset (I_1 \wedge I_2)$$

であるから，上式より

$$\vdash (0,0) \supset \overbrace{P_1 \parallel P_2}\ (I_1 \wedge I_2) \cdots\cdots\cdots\cdots\cdots (6)$$

そこで，式(4)，(6)および（⋂ I）により，

$$\vdash (0,0) \supset \overbrace{(P_1 \parallel P_2) \cap \sigma}\,(\bigvee_s < a_{1s} > \mathrm{true} \vee \bigvee_s < a_{2s} > \mathrm{true} \vee (\mathrm{halt}_1 \wedge \mathrm{halt}_2)) \cdots\cdots\cdots\cdots (7)$$

(7)式は（$P_1 \parallel P_2$）⋂ σ が停止しているか，いずれかのプロセスが常に動作できることをあらわしている．（4．3節，B参照）

（例終）

２．５　まとめ

本章では，共有変数をもつプロセスが並行に走るようなプログラムの，許される実行系列の集合を与えることによって，同期条件を定めるようなモデル，ＲＧＣについて，結果が速度独立に正しいこと，デッドロックフリーであること等を証明する公理系ＲＧＣ－ＩＬを導入し，その無矛盾性と完全性を示した．並列プログラムの性質として，本章で扱わなかった停止性等についても，ＲＧＣについての，ダイナミックロジックの拡張を進めることにより，扱うことが可能であると予想している．しかし，公理系が複雑なものとなることが予想され，それについては，そのような形式化が望ましいか否かを含めて，さらに議論を必要とするものと考える．

第３章　相互通信逐次型プロセス系の検証

Ｈoareが提案した相互通信逐次型プロセス系(Communicating Sequential Processes ：ＣＳＰ) (Hoare 78)は，共有変数を含まず，プロセス間の情報の交換及び同期を通信命令という機能によって行う，並列計算のモデルである．

ここで通信命令とは，プロセスＰから入力を受け取る命令

Ｐ？ｘ

または，プロセスＱに出力を送付する命令

Ｑ！τ

であり，前者を入力命令，後者を出力命令という．ここで，Ｐは発信元，Ｑは送付先と呼ばれる．又，ｘは標的変数，τは式である．二つのプロセスＰ，Ｑは通信命令によって次のように通信を行う．プロセスＰの実行が出力命令 Ｑ！τに至り，かつプロセスＱが入力命令 Ｐ？ｘに至り，τとｘの型が一致したとき，ｘにτの値が代入される．このような同期，通信の方式は，Ａｄａの待ち合わせ(rendezvous)，またはＩＮＭＯＳグループのｏｃｃａｍ(INMOS 84)等に取り入れられ，広く受け入れられている．

ＣＳＰの検証，意味論に関する報告は先に第１章で述べたように，Ｈoare流の部分的正当性検証の論理体系を拡張したもの(Apt Francez De Roever 80, Levin Gries 81, Lamport Schneider 84, Ossefort 83, Soundararajan 84)の他に，代数的あるいは表示的意味記述をあたえるもの(Brooks 83, Brooks Hoare Roscoe 84, De-Nicola 83, Francez Lehmann Pnueli 80, Francez Hoare Lehmann De Roever 79, Hoare 81, Milner 80)，Ｔemporal Ｌogicを用いるもの(Manna Pnueli 83),最弱前条件(weakest precondition)を与えるもの(Elrad Francez 82)等がある．本章では，ＲＧＣの場合と同様に，まず与えられたＣＳＰプログラムに対し状態の集合を考え，その上の二項関係を定義することによって，プログラムの実行中の詳細を無視し入出力関係のみに着目したセマンティクスを与える．

このセマンティクスを定める際に，多くの類似の論文で行われているような，個々のプロセスについてそれぞれの意味を定め，後にそれらをもとに，全体の意味を定めるといった方法をとらず，プログラム全体について，ある１ステップの実行の効果と，続いて実行されるプログラムの残りの部分の意味から，全体の意味をきめるという方法をとる．このような方法は，(Elrad Francez 82)の centralized approach と同じものであり，また，表示的意味論における接続法（continuation）の拡張とも考えられる．またこの定義のしかたは，各プロセスの制御構造に被防護命令の他に go to文等を加えた場合にも，容易に拡張できる．本章では次に，これらのセマンティクスをもとに，ＣＳＰの部分的正当性を証明する公理系を提案し，その無矛盾性を示す．この体系は，セマンティクスを定義したときと同様に，centralized approachに沿った推論の手順をたどる証明を可能にするものである．

しかし，二項関係を基本にした論理体系では，そのセマンティクスが命令の実行に関する情報を表現できないために，ＣＳＰのデッドロックフリー性を扱うことはできない．

そこで，本論文では，ＣＳＰの実行可能な命令系列と，その命令系列を実行途中に通る状態も含めて記録するセマンティクスとして，計算履歴集合を，centralized approachによって定義する．続いて，ＣＳＰの部分的正当性とデッドロックフリー性を，同時に検証する公理系を与え，その無矛盾性(soundness) を示す．

３．１　ＣＳＰのシンタックス

ＣＳＰの各プロセスは，通信命令をもつ被防護命令(guraded command) である．いくつかの論文では，通信命令はプログラム中の任意の位置に現れることはできないとしている．例えば，(Hoare 78)によれば防護の直後には入力命令のみが現れる．また(Apt Francez De Roever 80) では防護には通信命令は現れないとしている．さらに(Lamport Schneider 84,Ossefort 83)では，被防護命

令の任意の場所に通信命令が現れることを許している．

本論文では，(Levin Gries 81, Soundararajan 84, Elrad Francez 82, Manna Pnueli 83) 等と同様に，通信命令は防護の直後だけに現れるものとする．そのシンタックスは以下のように定義される．

xをプログラム変数，τを項とするとき，x:=τを割当文という．またP，Qをプロセス名とするとき，P?x および，Q! τを通信命令という．特に前者は入力命令，後者は出力命令という．またP，Qはそれぞれ発信元(source)，送付先(destination) という．また skipはスキップ命令とする．また，防護とは出現する変数が自由変数のみの述語である．

〔定義３．１〕ステートメント(以下では「文」と呼ぶ) は次の，ⅰ)，ⅱ)，ⅲ) で定義される．

ⅰ) 空系列Aは文である．

ⅱ) Aが文であるとき，x:=τ ; A は文である．

ⅲ) B_1, …, B_k が防護，C_1, …, C_k がskip文または通信命令，A_1, …, A_k 及び，Aが文であるとき，

$$\underline{\text{if}}\ B_1;C_1 \rightarrow A_1 \square \cdots \square\ B_k;C_k \rightarrow A_k\ \underline{\text{fi}}\ ;A$$

及び

$$\underline{\text{do}}\ B_1;C_1 \rightarrow A_1 \square \cdots \square\ B_k;C_k \rightarrow A_k\ \underline{\text{od}}\ ;A$$

は文である．

〔定義３．２〕 文A_1, …, A_k が次の条件ⅰ)，ⅱ)，ⅲ) を満たすとき，各文をプロセスとよぶ．

ⅰ) 各A_i は，P_i という名前をもつ．以下では，プロセス自身を表す記号A_i と，その名前P_i を同一視する．

ⅱ) 各P_i に現れる変数は全て各ｉについて定められた変数の集合V_i の元であり，$i \neq j$ ならば，$V_i \cap V_j = \phi$ とする．すなわち共有変数は存在しない．

ⅲ) 各P_i に現れる通信命令はいずれも通信相手，即ちその発信元または送付先として自分以外のプロセス，すなわち，P_1, …, P_{i-1}, P_{i+1}, …, P_n のいずれかを指定している．

〔定義３．３〕　$P_1, \cdots, P_n$ がプロセスであるとき $P_1 \parallel \cdots \parallel P_n$ をＣＳＰプログラムという．

定義から直ちにわかるように，任意のプログラムから，あるプロセスのプレフィクスを取り去ったものは，常にプログラムとなっていることに注意されたい．

〔定義３．４〕　２つの通信命令 C_1 ，C_2 について，C_1 がプロセスＱの入力命令Ｐ?x，かつ，C_2 がプロセスＰの出力命令Ｑ!τ（あるいはその逆）であるとき，すなわち C_1 と C_2 が通信をすることができるとき，

$$\mu(C_1 , C_2)$$

とかく．また，

$$target(P?x)=P, \quad target(Q!\tau)=Q$$

とする．また C_1 ，C_2 の実行を $C_1 \leftrightarrow C_2$ で表す．

３．２　ＣＳＰの部分的正当性の検証

３．２．１　状態集合上の二項関係

最初にＣＳＰプログラムの状態の集合を考え，その上の二項関係によってセマンティクスを定義する．状態及びその上の二項関係は基本的にはＲＧＣの場合と同様である．

〔定義３．５〕ＣＳＰ，$P \equiv P_1 \parallel \cdots \parallel P_n$ の状態とは，プロセス P_i に現れるすべての変数の集合 V_i $(1 \leq i \leq n)$ とすると，

$$V_1 \cup V_2 \cup \cdots \cup V_n$$

の中のすべての変数の値の組である．ＣＳＰ，Ｐのとりうるすべての状態の集合を U_P で表す．

〔定義３．６〕　ＣＳＰプログラムＰの各割当文 $x:=\tau$ および通信命令の対 $C_k \equiv P?x$, $C_h \equiv Q!\tau$ について U_P 上の二項関係

$$m(x:=\tau) \quad , \quad m(C_k \leftrightarrow C_h)$$

を次のように定義する．

$m(x:=\tau) = \{(i,j) \mid$ ｊは状態ｉにおける
ｘの値を τ の値に更新して得られる状態$\}$．

$m(C_k \in C_h)$ は，$\mu(C_k, C_h)$ の場合のみ定義され，

$$m(C_k \in C_h) = m(x := \tau).$$

以下，与えられたＣＳＰプログラムに対して，その初期状態と終了状態の対の集合となるような二項関係を定義する．

〔定義３．７〕　ＣＳＰ，$P \equiv P_1 \| \cdots \| P_n$ に対して，U_P 上の二項関係 $m(P_1 \| \cdots \| P_n)$ を次のように定義する．

(i) $m(\Lambda \| \cdots \| \Lambda) = \{(i,i) \mid i \in U_P\}$，

(ii) $x := \tau$ が $P_l(^{+} x := \tau ; P_l')$ の命令で $(i,i') \in m(x := \tau)$，かつ，

$$(i',j) \in m(P_1 \| \cdots \| P_l' \| \cdots \| P_n)$$

のとき，

$$(i,j) \in m(P_1 \| \cdots \| x := \tau ; P_l' \| \cdots \| P_n).$$

（iii）m番目のプロセスの先頭のif文:

$$\underline{\text{if}}\ B_1 ; C_1 \to S_1 \ \square \cdots \square\ B_k ; C_k \to S_k\ \underline{\text{fi}}$$

の $C_1, \cdots, C_k$ がいずれも通信命令で，その通信相手のプロセスが全てΛのとき，

$$m(P_1 \| \cdots \| \underline{\text{if}}\ B_1 ; C_1 \to S_1 \ \square \cdots \square\ B_k ; C_k \to S_k\ \underline{\text{fi}}; P_m' \| \cdots \| P_n) = \phi.$$

（iv）m番目のプロセスの先頭のdo文:

$$\underline{\text{do}}\ B_1 ; C_1 \to S_1 \ \square \cdots \square\ B_k ; C_k \to S_k\ \underline{\text{od}}$$

の，すべてのh$(1 \leq h \leq k)$で，$i \bigcup B_h$ または C_h の相手がΛであるとき（この条件を LE_m と書く），

$$(i,j) \in m(P_1 \| \cdots \| P_m' \| \cdots \| P_n)$$

ならば

$$(i,j) \in m(P_1 \| \cdots \| \underline{\text{do}}\ B_1 ; C_1 \to S_1 \ \square \cdots \square\ B_k ; C_k \to S_k\ \underline{\text{od}}; P_m' \| \cdots \| P_n)$$

（v）m番目のプロセスの先頭のif文のある防護 B_k，およびs番目のプロセスの先頭のif文のある防護 B_s について，　あるｉで，$i \cap B_k \in B_l$，かつ $\mu(C_k, C_l)$ のとき，

$$(i',j) \in m(P_1 \| \cdots \| S_k ; P_m' \| \cdots$$

$\| S_t ; P_s' \| \cdots \| P_n)$

かつ

$(i, i') \in m(C_k \leftrightarrow C_t)$

ならば，

$(i, j) \in m(P_1 \| \cdots \| \underline{if} \cdots \Box B_k ; C_k \rightarrow S_k \Box \cdots \underline{fi} ; P_m' \|$
$\cdots \| \underline{if} \cdots \Box B_t ; C_t \rightarrow S_t \Box \cdots \underline{fi} ; P_s' \|$
$\cdots \| P_n)$.

(vi) $(i, j) \in m(P_1 \| \cdots \| S_k ; P_m' \| \cdots \| P_n)$
かつ $i \models B_k$ のとき，
$(i, j) \in m(P_1 \| \cdots \| \underline{if} \cdots \Box B_k ; skip \rightarrow S_k \Box \cdots \underline{fi} ;$
$P_m' \| \cdots \| P_n)$.

(vii) $(i, j) \in m(P_1 \| \cdots \|$
$\underline{if} B_1 ; C_1 \rightarrow S_1 \Box \cdots \Box B_k ; C_k \rightarrow S_k \underline{fi} ;$
$\underline{do} B_1 ; C_1 \rightarrow S_1 \Box \cdots \Box B_k ; C_k \rightarrow S_k \underline{od} ;$
$P_m' \| \cdots \| P_n)$

であるとき，

〔すなわち，if文とそれに続くdo文の内部が字面の上で全く同じであるとき，〕

$(i, j) \in m(P_1 \| \cdots \|$
$\underline{do} B_1 ; C_1 \rightarrow S_1 \Box \cdots \Box B_k ; C_k \rightarrow S_k \underline{od} ;$
$P_m' \| \cdots \| P_n)$.

直観的には，$(i, j) \in m(P_1 \| \cdots \| P_n)$ であることは，RGCの場合と同様に，状態iからプログラム $P_1 \| \cdots \| P_n$ を実行したら，状態jで停止することが可能であることを示している．すなわち二項関係 $m(P_1 \| \cdots \| P_n)$ はプログラムの実行中の詳細を無視し，入出力関係のみに着目したセマンティクスであるといえる．

〔例3．1〕次のプロセス P_1，P_2 からなるCSPプログラム $P_1 \| P_2$ について考える．

$P_1 \equiv x := x_0 ; \underline{do}\ x > 1 ; P_2 ! x \rightarrow x := x - 1\ \underline{od}$
$(\equiv x := x_0 ; P_1'$ と書く$)$

$P_2 \equiv z := 1 ;$ do true; $P_1 ? y \rightarrow z := z * y$ od

（$\equiv z := 1 ; P_2'$ と書く）

このとき状態 i を

$$i \models x_0 = 3 \quad ,$$

とし，j を

$$j \models z = 6 \wedge x = 1 \wedge y = 2$$

とすると，

$$(i, j) \in m(P_1 \parallel P_2)$$

となることが，次のようにしてわかる．

定義３．７(i)〔以下(i) …(vii) は定義３．７中の番号〕より，

$$(j, j) \in m(\Lambda \parallel \Lambda).$$

$j \models LE_2$ より(iv) を適用して，

$$(j, j) \in m(\Lambda \parallel P_2').$$

$x = 1 \supset LE_1$ より $j \models LE_1$ ．

(iv) を適用して，

$$(j, j) \in m(P_1' \parallel P_2').$$

ここで，

$$j_1 \models z = 6 \wedge x = 2 \wedge y = 2$$

とすると，

$$(j_1, j) \in m(x := x - 1)$$

より，(ii) を適用して，

$$(j_1, j) \in m(x := x - 1; P_1' \parallel P_2').$$

また，

$$j_2 \models z = 3 \wedge x = 2 \wedge y = 2$$

とすると

$$(j_2, j_1) \in m(z := z * y)$$

より，(ii) を適用して，

$(j_2, j) \in m(x := x - 1; P_1' \parallel z := z * y; P_2')$ ．

また，

$$j_3 \models z = 3 \wedge x = 2 \wedge y = 3$$

とすると

$$j_3 \models x > 1 \wedge \mathrm{true},$$

かつ

$$\mu\ (P_2!x,\ P_1?y),$$

$$(j_3, j_2) \in m(P_2!x \leftrightarrow P_1?y)$$

より，（v）を適用して，

$(j_3, j) \in m$ (if $x > 1$; $P_2!x \rightarrow x := x - 1$ fi ; P_1'

$\|$ if true; $P_1?y \rightarrow z := z * y$ fi ; P_2') .

（vii）を二回適用して，

$$(j_3, j) \in m(P_1' \| P_2').$$

以上を繰り返すと

$$i' \models x = 3 \wedge z = 1$$

のとき，

$$(i', j) \in m(P_1' \| P_2')$$

が得られる．さらに，

$$i'' \models x = 3$$

とすると，

$$(i, i'') \in m\ (x := x_0)$$

かつ

$$(i'', i') \in m\ (z := 1)$$

より，（ii）を2回適用して，

$$(i, j) \in m\ (P_1 \| P_2)$$

となる．

（例終）

3．2．2　CSPの部分的正当性の検証

以下では，前節で定義したセマンティクスをもとにして，CSPの部分的正当性の検証のための公理系について述べる．ここで提案する検証体系はHoareの体系と同じ記法を用いる．

〔定義３．８〕プログラムＰが入力条件Φと出力条件Ψについて部分的正当であるとは，Φの成り立つ状態からＰを実行して，停止すれば常にΨが成り立つことを言う．すなわち，任意の

（ｉ，ｊ）∈ｍ（Ｐ）について，

$$i \models \Phi \quad \text{ならば} \quad j \models \Psi$$

である．

本論文ではＰのΦ，Ψに関する部分的正当性を

$$\Phi \ \{P\} \ \Psi$$

とあらわす．以下述語及びΦ｛Ｐ｝Ψの形の式を論理式という．

以下ではＣＳＰの部分的正当性の検証体系を示し，その無矛盾性を証明する．

ＣＳＰの部分的正当性を証明する体系は次に定める．

まず推論規則をしめす．

ＣＩ１）

$$\frac{\Gamma}{\Phi \ \{P\} \ \Psi}$$

ここで，Γは論理式の並びで次のｉ）～ⅳ）の規則にあてはまる式の並びをすべて含む．

ｉ）Ｐが

$$P_1 \parallel \cdots \parallel x := \tau ; \ P_t' \parallel \cdots \parallel P_n$$

のとき，χを適当な述語とすると，

$$\Phi \ \{x := \tau\} \ \chi ,$$

$$\chi \ \{P_1 \parallel \cdots \parallel P_t' \parallel \cdots \parallel P_n\} \ \Psi$$

ⅱ）Ｐが，

$$P_1 \parallel \cdots \parallel \underline{if} \cdots \square B_k ; \ C_k \rightarrow S_k \ \square \cdots \underline{fi}; \ P_m'$$
$$\parallel \cdots \parallel \underline{if} \cdots \square B_t ; \ C_t \rightarrow S_t \ \square \cdots \underline{fi}; \ P_s'$$
$$\parallel \cdots \parallel P_n$$

かつ$\mu(C_k, C_t)$のとき，

$$\Phi \wedge B_k \wedge B_t \ \{C_k \leftrightarrows C_t\} \ \chi ,$$

$$\chi \ \{P_1 \parallel \cdots \parallel S_k ; \ P_m' \parallel \cdots$$
$$\parallel S_t ; \ P_s' \parallel \cdots \parallel P_n\} \ \Psi$$

iii) Pが,

$P_1 \parallel \cdots \parallel$ if $\cdots \Box B_k$;skip $\rightarrow S_k \Box \cdots$ fi; $P_m' \parallel \cdots \parallel P_n$

のとき,

$$\Phi \wedge B_k \ \{P_1 \parallel \cdots \parallel S_k \ ; \ P_m' \parallel \cdots \parallel P_n\} \ \Psi$$

iv) Pが

$P_1 \parallel \cdots \parallel$ do $B_1; C_1 \rightarrow S_1 \Box \cdots \Box B_k \ ; \ C_k \rightarrow S_k$ od; $P_m' \parallel \cdots \parallel P_n$

のとき,

$$\Phi \wedge LE_m \ \{P_1 \parallel \cdots \parallel P_m' \parallel \cdots \parallel P_n\} \ \Psi,$$

$\Phi \wedge \neg LE_m$ $\{P_1 \parallel \cdots \parallel$

if $B_1; C_1 \rightarrow S_1 \Box \cdots \Box B_k \ ; \ C_k \rightarrow S_k$ fi;

do $B_1; C_1 \rightarrow S_1 \Box \cdots \Box B_k \ ; \ C_k \rightarrow S_k$ od;

$P_m' \parallel \cdots \parallel P_n\} \ \Psi$

CI2)

$$\frac{\Phi \supset \Psi \ [x/\tau]}{\Phi \ \{x := \tau\} \ \Psi}$$

ここで, $\Psi\ [x/\tau]$ は, Ψ の x の全ての自由な出現に, τ を代入したもの.

CI3) $C_k = P!\tau$, $C_l = Q?x$ として,

$$\frac{\Phi \supset \Psi \ [x/\tau]}{\Phi \ \{C_k \leftrightarrow C_l\} \ \Psi}$$

CI4)

$$\frac{\Phi \supset \Phi' \qquad \Phi' \ \{P\} \ \Psi}{\Phi \ \{P\} \ \Psi}$$

CI5)

$$\frac{\Phi \ \{P\} \ \Phi' \qquad \Phi' \supset \Psi}{\Phi \ \{P\} \ \Psi}$$

CI6)

$$\frac{\Phi \supset \Psi}{\Phi \ \{\Lambda \parallel \cdots \parallel \Lambda\} \ \Psi}$$

公理として次のものを採用する．

ＣＡ１）任意のＰ，任意のΨについて，

false ｛Ｐ｝ Ψ．

ＣＡ２）Ｐに少なくともひとつΛでないプロセスが存在し，かつ推論規則１）のｉ）～ｉｖ）のいずれの規則にもあてはまらないとき，即ち，任意のΛでないプロセスP_kについて，文の先頭がif文でかつ防護に続く命令がすべて通信命令で，かつこれらの通信命令のどの対C_k，C_lについても$\mu(C_k, C_l)$が成り立たないとき，任意のΦ，Ψについて，

Φ ｛Ｐ｝ Ψ．

ＣＡ３）自然数論の全ての定理．

ＣＡ２）はＰがこれ以上実行が進まない場合をあらわす．このような場合の扱いは，次節で詳しく触れる．

〔定義３．１０〕論理式Φ ｛Ｐ｝ Ψについて，次のように定義されるラベル付の木を，その証明図という．

⑴根のラベルはΦ ｛Ｐ｝ Ψである．

⑵すべての内部節点のラベルは，その子供の節点のラベルから推論規則によって直接に導出できる．

⑶すべての葉は，ある内部節点で使われた推論規則

ＣＩ１）ｉｖ）の前提

$\Phi \wedge \neg L E_m$ ｛P_1 ‖ … ‖

if B_1; C_1 → S_1 □ … □ B_k ; C_k → S_k fi;

do B_1; C_1 → S_1 □ … □ B_k ; C_k → S_k od; P_m'

‖ … ‖ P_n ｝ Ψ

の子孫であり，かつそのラベルは，

Φ ｛P_1 ‖ … ‖

do B_1; C_1 → S_1 □ … □ B_k ; C_k → S_k od; P_m'

‖ … ‖ P_n ｝ Ψ

であるか，または，公理をラベルとしてもつ．

この公理系について，すべての公理が妥当なものであり，またどの推論規則についてもその前提にふくまれる式がすべて成立すれば，その結論の式も成立することは，定義３．７から証明できる．しかし定義３．１０で与えた証明図では，証明の最上段に公理以外のものが現れうるので，そのような通常の無矛盾性の証明では証明図の定義の妥当性を保証するには不十分である．この推論体系の妥当性は次の定理で保証される．

〔定理３．１〕任意のＣＳＰプログラムＰと任意の述語Φ，Ψについて，Φ｛Ｐ｝Ψの証明図が存在するとき，任意の

$$(i, j) \in m(P)$$

について，

$$i \models \Phi \text{ ならば } j \models \Psi.$$

（証明）$(i, j) \in m(P)$であることが決定するまでに適用された，定義３．７の規則の適用回数に関する帰納法を用いる．

（１）$(i, j) \in m(P)$であることが，定義３．７の規則の１回の適用でわかったとする．これは，

$$P \equiv \Lambda \parallel \cdots \parallel \Lambda$$

の場合のみである．このときは定義３．７および推論規則ＣＩ６）より明らかに定理は成立する．

（２）帰納法の仮定として，定義３．７の規則のｔ回以下の適用で

$$(i', j) \in m(P')$$

がわかったとき，任意の述語χについて，$i' \models \chi$，かつ
χ｛Ｐ’｝Ψの証明図が存在すれば$j \models \Psi$であると仮定する．

$(i, j) \in m(P)$であることがｔ＋１回でわかったとする．このｔ＋１回目に適用されたのが定義３．７のii）～vi）のいずれの規則であったかによって場合分けをする．

ii）の場合：この場合，Ｐ’は，

$$P_1 \parallel \cdots \parallel P_t \parallel \cdots \parallel P_n$$

であり，

$$(i', j) \in m(P_1 \parallel \cdots \parallel P_t \parallel \cdots \parallel P_n),$$

$$(i, i') \in m\ (x := \tau),$$

かつ，

$$(i, j) \in m\ (P_1 \parallel \cdots \parallel x := \tau;\ P_t' \parallel \cdots \parallel P_n)$$

である．

今，$i \models \Phi$であったとする．そして，$\Phi\ \{P\}\ \Psi$の証明図は，規則ＣＩ１）のｉ）より次のような部分

$$\frac{\Phi\ \{x := \tau\}\ \chi \qquad \chi\ \{P_1 \cdots \parallel P_t' \parallel \cdots \parallel P_n\}\ \Psi}{\Phi\ \{P_1 \parallel \cdots \parallel x := \tau;\ P_t' \parallel \cdots \parallel P_n\}\ \Psi}$$

を持つことに注意する．この前提に現れる式のうち

$$\Phi\ \{x := \tau\}\ \chi$$

からは，これに規則ＣＩ２）を適用し，

$$\Phi \supset \chi\ [x/\tau]$$

が得られる．即ち，

$$i \models \Phi$$

ならば

$$i' \models \chi$$

となる．

また，もとの証明図の

$$\chi\ \{P_1 \cdots \parallel P_t' \parallel \cdots \parallel P_n\}\ \Psi$$

を根とする部分木は，根それ自身の証明図となっている．ここで帰納法の仮定により

$$i' \models \chi$$

ならば

$$j \models \Psi$$

が結論される．

iii）の場合：このときは，$m(P) = \phi$であるので，題意は空に満たされる．

iv）の場合：このときは，

$(i,j) \in m(P_1 \| \cdots \| P_m' \| \cdots \| P_n)$,

$(i,j) \in m(P_1 \| \cdots \| \underline{do} B_1 ; C_1 \to S_1 \Box \cdots$

$\Box B_k ; C_k \to S_k \underline{od}; P_m' \| \cdots \| P_n)$,

でありかつ,

$$i \models L E_m$$

となっている．そして，$\Phi \{P\} \Psi$の証明図は，規則ＣＩ１）のiv）より次のような部分

$$\frac{\Phi \wedge L E_m \{P_1 \| \cdots \| P_m' \| \cdots \| P_n\} \Psi}{\Phi \{P_1 \| \cdots \| \underline{do} B_1 ; C_1 \to S_1 \Box \cdots \Box B_k ; C_k \to S_k \underline{od}; P_m' \| \cdots \| P_n\} \Psi .}$$

をもつ．この前提に現れる式

$$\Phi \wedge L E_m \{P_1 \| \cdots \| P_m' \| \cdots \| P_n\} \Psi$$

を根とする部分木は，根それ自身の証明図となる．よって，帰納法の仮定より

$$i \models \Phi \wedge L E_m$$

ならば

$$j \models \Psi$$

である．

v）の場合：このときは，

$(i',j) \in m(P_1 \| \cdots \| S_k ; P_m' \| \cdots \| S_t ; P_s' \| \cdots \| P_n)$,

$(i,i') \in m(C_k \leftrightarrow C_t)$,

$(i,j) \in m(P_1 \| \cdots \| \underline{if} \cdots \Box B_k ; C_k \to S_k \Box \cdots \underline{fi}; P_m' \| \cdots \| \underline{if} \cdots \Box B_t ; C_t \to S_t \Box \cdots \underline{fi}; P_s' \| \cdots \| P_n)$,

そして，

$$i \models B_k \wedge B_t, \quad \text{かつ} \quad \mu(C_k, C_t)$$

となる．また，規則ＣＩ１）のii）より，$\Phi \{P\} \Psi$の証明図は次

のような部分

$$\frac{\Phi \wedge B_k \wedge B_l \ \{C_k \leftrightarrow C_l\}\ \chi \qquad \chi\ \{P_1 \parallel \cdots \parallel S_k\ ;\ P_m' \parallel \cdots \parallel S_l\ ;\ P_s' \parallel \cdots \parallel P_n\}\ \Psi}{\Phi\ \{P_1 \parallel \cdots \parallel \underline{if} \cdots \Box B_k\ ;\ C_k \rightarrow S_k\ \Box \cdots \underline{fi};\ P_m' \parallel \cdots \parallel \underline{if} \cdots \Box B_l\ ;\ C_l \rightarrow S_l\ \Box \cdots \underline{fi};\ P_s' \parallel \cdots \parallel P_n\}\ \Psi}$$

をもつ．その前提の式の一つ，

$$\Phi \wedge B_k \wedge B_l \ \{C_k \leftrightarrow C_l\}\ \chi$$

に規則２）を適用すると

$$\Phi \wedge B_k \wedge B_l \supset \chi\ [x/\tau]$$

が得られる．従って，

$$i \models \Phi$$

ならば

$$i' \models \chi$$

となる．また

$\chi\ \{P_1 \parallel \cdots \parallel S_k\ ;\ P_m' \parallel \cdots \parallel S_l\ ;\ P_s' \parallel \cdots \parallel P_n\}\ \Psi$

を根とする部分木は，根それ自身の証明図となる．帰納法の仮定より，

$$i' \models \chi$$

ならば

$$j \models \Psi.$$

vi）の場合：このとき，

$$i \models B_k,$$

$$(i,j) \in m(P_1 \parallel \cdots \parallel S_k\ ;\ P_m' \parallel \cdots \parallel P_n),$$

$$(i,j) \in m(P_1 \parallel \cdots \parallel \underline{if} \cdots \Box B_k\ ;\mathrm{skip} \rightarrow S_k \Box \cdots \underline{fi};\ P_m' \parallel \cdots \parallel P_n).$$

である．

規則ＣＩ1）iii）より証明図は次のようになる．

$$\frac{\Phi \wedge B_k \ \{P_1 \parallel \cdots \parallel S_k ; P_m' \parallel \cdots \parallel \cdots \parallel P_n\} \ \Psi}{\Phi \ \{P_1 \parallel \cdots \parallel \underline{if} \cdots \Box B_k \ ; skip \rightarrow S_k \ \Box \cdots \underline{fi}; \ P_m' \parallel \cdots \parallel P_n\} \ \Psi}$$

上の推論規則の前提を根とする部分木は，根それ自身の証明図となり，帰納法の仮定より

$$i \models \Phi \wedge B_k$$

ならば

$$j \models \Psi .$$

vii）の場合：このときは，

$(i,j) \in m($ $P_1 \parallel \cdots \parallel$
$\underline{if} B_1 ; C_1 \rightarrow S_1 \ \Box \cdots \Box B_k \ ; \ C_k \rightarrow S_k \ \underline{fi};$
$\underline{do} B_1 ; C_1 \rightarrow S_1 \ \Box \cdots \Box B_k \ ; C_k \rightarrow S_k \ \underline{od};$
$P_m' \parallel \cdots \parallel P_n)$,

〔if文とそれに続くdo文の内部が全く同じ〕

$(i,j) \in m($ $P_1 \parallel \cdots \parallel \underline{do} B_1 ; C_1 \rightarrow S_1 \ \Box \cdots \Box$
$B_k \ C_k \rightarrow S_k \ \underline{od}; \ P_m' \parallel \cdots \parallel P_n)$

となっている．規則1）iv）より証明図は次のような部分

$$\frac{\Phi \wedge \neg L E_m \ \{P_1 \parallel \cdots \parallel \underline{if} B_1 ; C_1 \rightarrow S_1 \ \Box \cdots \Box B_k \ ; \ C_k \rightarrow S_k \ \underline{fi}; \ \underline{do} B_1 ; C_1 \rightarrow S_1 \ \Box \cdots \Box B_k \ ; \ C_k \rightarrow S_k \ \underline{od}; \ P_m' \parallel \cdots \parallel P_n\} \ \Psi}{\Phi \ \{P_1 \parallel \cdots \parallel \underline{do} B_1 ; C_1 \rightarrow S_1 \ \Box \cdots \Box B_k \ ; \ C_k \rightarrow S_k \ \underline{od}; \ P_m' \parallel \cdots \parallel P_n\} \ \Psi}$$

を持つ．このとき，$i \models L E_m$ ならば，定義3．7 iv）より既に解決している．したがって $i \models \neg L E_m$ の場合を考える．このとき，

$\Phi \wedge \neg LE_m$ {$P_1 \parallel \cdots \parallel$

if $B_1; C_1 \rightarrow S_1 \square \cdots \square B_k; C_k \rightarrow S_k$ fi;

do $B_1; C_1 \rightarrow S_1 \square \cdots \square B_k; C_k \rightarrow S_k$ od;

$P_m' \parallel \cdots \parallel P_n$ } Ψ

を根とする部分木は，その証明図になるとは限らない．なぜなら，定義３．１０のiii）より，葉に

Φ {$P_1 \parallel \cdots \parallel$

do $B_1; C_1 \rightarrow S_1 \square \cdots \square B_k; C_k \rightarrow S_k$ od; P_m'

$\parallel \cdots \parallel P_n$ } Ψ

というラベルの節点をもつかもしれない．そのような場合は次のような変形を行うことにより，新たに証明図とすることができる．すなわちもとの証明図の

Φ {$P_1 \parallel \cdots \parallel$

do $B_1; C_1 \rightarrow S_1 \square \cdots \square B_k; C_k \rightarrow S_k$ od; P_m'

$\parallel \cdots \parallel P_n$ } Ψ

を根とする部分木（これはそれ自身の証明図となる）を，葉に出現するすべての同じ式の上にのせる．（図３．１a），b）参照）．

これによって，

$\Phi \wedge \neg LE_m$ {$P_1 \parallel \cdots \parallel$

if $B_1; C_1 \rightarrow S_1 \square \cdots \square B_k; C_k \rightarrow S_k$ fi;

do $B_1; C_1 \rightarrow S_1 \square \cdots \square B_k; C_k \rightarrow S_k$ od;

$P_m' \parallel \cdots \parallel P_n$ } Ψ

の証明図を得た後，帰納法の仮定を適用する．即ち，

$$i \models \Phi \wedge \neg LE_m$$

ならば，

$$j \models \Psi.$$

今，$i \models \neg LE_m$ の場合を考えているので，

$$i \models \Phi$$

ならば

$$j \models \Psi.$$ （証明終）

a)

$$\frac{\Phi\ \{P_1 \parallel \cdots\cdots \parallel \text{do}\cdots * \cdots \text{od}; P_m \parallel \cdots\cdots \parallel P_n\}\ \Psi}{\vdots}$$

$$\frac{\cdots\cdots\cdots\cdots\cdots\cdots}{\neg L E_m \wedge \Phi\ \{P_1 \parallel \cdots\cdots \parallel \text{if}\cdots * \cdots \text{fi}; \text{do}\cdots * \cdots \text{od}; P_m \parallel \cdots\cdots \parallel P_n\}\ \Psi}$$

b)

$$\frac{\Phi\ \{P_1 \parallel \cdots\cdots \parallel \text{do}\cdots * \cdots \text{od}; P_m \parallel \cdots\cdots \parallel P_n\}\ \Psi}{\vdots}$$

$$\frac{\cdots\cdots\cdots\cdots\cdots}{\neg L E_m \wedge \Phi\ \{P_1 \parallel \cdots\cdots \parallel \text{if}\cdots * \cdots \text{fi};}$$

$$\text{do}\cdots * \cdots \text{od}; P_m \parallel \cdots \parallel P_n\}\ \Psi \qquad \Phi \wedge L E_m\ \{P_1 \parallel \cdots \parallel P_m \parallel \cdots \parallel P_n\}\ \Psi \quad \text{規則1) iv}$$

$$\frac{\Phi\ \{P_1 \parallel \cdots\cdots \parallel \text{do}\cdots * \cdots \text{od}; P_m \parallel \cdots\cdots \parallel P_n\}\ \Psi}{\vdots}$$

$$\frac{\cdots\cdots\cdots\cdots\cdots\cdots}{\neg L E_m \wedge \Phi\ \{P_1 \parallel \cdots \parallel \text{if}\cdots * \cdots \text{fi}; \text{do}\ \cdots * \cdots \text{od}; P_m \parallel \cdots \parallel P_n\}\ \Psi}$$

図３．１　証明図の変形

〔例３．２〕

前節で述べた検証体系による，部分的正当性の検証例を示す．例３．１のプログラム$P_1 \parallel P_2$について考える．すなわち，P_1，P_2を次に与えられるものである．

$P_1 \equiv x := x_0$;<u>do</u> $x > 1$; P_2 ! $x \rightarrow x := x - 1$ <u>od</u>

（$\equiv x = x_0; P_1'$と書く）

$P_2 \equiv z := 1$;<u>do</u> true; P_1 ? $y \rightarrow z := z * y$ <u>od</u>

（$\equiv z := 1 ; P_2'$と書く）

そこで，

$$\Phi \equiv x_0 \geq 0,\ \Psi \equiv fact(x_0) = z$$

とするとき，

$$\Phi \{P_1 \parallel P_2\} \Psi$$

となることを証明する．

すなわち，

$$\chi \equiv (fact(x_0) = z * fact(x))$$

とすると，

$$\Phi \{x := x_0\} \Phi \wedge x = x_0$$

$$\Phi \wedge x = x_0 \{z := 1\} \Phi \wedge x = x_0 \wedge z = 1$$

〔規則ＣＩ１）ｉ）〕

上の推論は$x := x_0$をまず実行し続いて$z := 1$を実行した場合に対応する．$x := x_0$と$z := 1$を逆順に実行した場合に対応する推論も同様であるので省略する．次に，

$$\Phi \wedge x = x_0 \wedge z = 1 \supset \chi$$

より規則ＣＩ４）を適用して

$$\chi \{P_1' \parallel P_2'\} \Psi$$

とする．図３．２に

$$\chi \{P_1' \parallel P_2'\} \Psi$$

の証明を示す．

図中の＊＊については，推論規則ＣＩ２），ＣＩ３）を適用して公理に到達する．$x := x - 1$と$z := z * y$を逆順に実行した場合に対応する推論も同様である．

（例終り）

$$\frac{x>0\wedge x+1=y\wedge z*fact(x)*(x+1)=fact(x_0)*y\supset\chi \quad \chi\ \{P_1'\parallel P_2'\}\ \Psi}{x>0\wedge x+1=y\wedge z*fact(x)*(x+1)=fact(x_0)*y\ \{P_1'\parallel P_2'\}\ \Psi,}$$

$$x>0\wedge x+1=y\wedge z*fact(x)*(x+1)=fact(x_0)\ \{z:=z*y\}\ x>0\wedge x+1=y\wedge z*fact(x)*(x+1)=fact(x_0)*y,\ \cdots *$$

$$\frac{\chi\wedge x>1\wedge x=y\ \{x:=x-1\}\ x>0\wedge true\wedge x+1=y\wedge z*fact(x)*(x+1)=fact(x_0)\cdots *}{\chi\wedge x>1\wedge x=y\ \{x:=x-1;\ P_1'\parallel z:=z*y;\ P_2'\}\ \Psi}$$

$$\frac{\chi\wedge x>1\ \{P_2!x\leftrightarrow P_1?y\}\ \chi\wedge x>1\wedge true\wedge x=y\cdots **}{\chi\wedge x>1\wedge true\ \{if\cdots fi;\ P_1'\parallel if\cdots fi;\ P_2'\}\ \Psi\cdots *1}$$

$$\frac{\chi\wedge x>1_1\ \{if\cdots\cdots fi;\ P_1'\parallel P_2'\}\ \Psi \qquad \chi\wedge x\leq 1\ \{\Lambda\parallel P_2'\}\ \Psi\cdots}{\chi\ \{P_1'\parallel P_2'\}\ \Psi}$$

＊2の子孫

descendant of formula ＊2

$$\frac{\chi\wedge x\leq 1\wedge true\ \{\Lambda\parallel\Lambda\}\ \Psi \qquad \chi\wedge x\leq 1\wedge true\ \{\Lambda\parallel if\cdots fi;\ P_2'\}\ \Psi\ \cdots公理}{\chi\wedge x\leq 1\ \{\Lambda\parallel P_2'\}\ \Psi}$$

図3．2　$\chi\ \{P_1'\parallel P_2'\}\ \Psi$の証明図

上の例では，証明図中に全く同じ部分木が現れたのでその部分を省略した．このように自明な冗長部分は，割当文が何個所も現れる場合は常に現れる．それらはいつも省略可能である．

例えば次のようなＣＳＰの場合，

$$P_1 : x := 1 ; \underline{do} \cdots \underline{od}$$

$$P_2 : y := 1 ; \underline{do} \cdots \underline{od}$$

Φをｘもｙも含まない述語とすると，

$$\Phi \{P_1 \parallel P_2\} \Psi$$

の証明の最初のステップ，即ち，根の子孫は次のようになる．

(1)

$$\frac{\Phi \wedge x = 1 \{y := 1\} \Phi \wedge x = 1 \wedge y = 1 \quad \Phi \wedge x = 1 \wedge y = 1 \{do \cdots od \parallel do \cdots od\} \Psi \cdots\cdots ***}{\Phi \{x := 1\} \Phi \wedge x = 1 \quad \Phi \wedge x = 1 \{do \cdots od \parallel P_2\} \Psi}$$

$$\Phi \{P_1 \parallel P_2\} \Psi$$

(2)

$$\frac{\Phi \wedge y = 1 \{x := 1\} \Phi \wedge x = 1 \wedge y = 1 \quad \Phi \wedge x = 1 \wedge y = 1 \{do \cdots od \parallel do \cdots od\} \Psi \cdots\cdots ***}{\Phi \{y := 1\} \Phi \wedge y = 1 \quad \Phi \wedge y = 1 \{P_1 \parallel do \cdots od\} \Psi}$$

$$\Phi \{P_1 \parallel P_2\} \Psi$$

ここで，＊＊＊の子孫は全く同じ部分木となる．これはＣＳＰプログラム$P_1 \parallel P_2$が，ｘ：＝１を先に実行してもｙ：＝１を先に実行してもその後の計算に影響を及ぼさないことを意味している．このことを利用して，プログラム全体の証明図を小さくすることができる．

〔定義３．１１〕a_{kl}を割当文とする．ＣＳＰプログラムＰが次のような形をしているとき，

$$a_{11}; a_{12}; \cdots; a_{1t_1}; P_1' \parallel$$

$$a_{21}; a_{22}; \cdots; a_{2t_2}; P_2' \parallel \cdots \parallel$$

$$a_{n1}; a_{n2}; \cdots; a_{nt_n}; P_n'$$

ここで，

$$w_1 = a_{11}; a_{12}; \cdots; a_{1t_1},$$
$$w_2 = a_{21}; a_{22}; \cdots; a_{2t_2},$$
$$\vdots$$
$$w_n = a_{n1}; a_{n2}; \cdots; a_{nt_n}$$

とする．またw_1，w_2，…，w_nをシャフルした結果のひとつを，wであらわす．

〔定義３．１２〕プログラムPの割当文a_1，a_2，…，a_tの系列

$$u \equiv a_1; a_2; \cdots; a_t$$

に対して，U_P上の二項関係m（u）を次のように定義する．

$$m(u) = \begin{cases} m(a_1) & , \text{ uが} a_1 \text{のとき} \\ m(a_1) \bigcirc m(a_2; \cdots; a_{t1}) & , \\ & \text{uが} a_1; a_2; \cdots; a_t \text{のとき．} \end{cases}$$

ここで，○は関係の合成をあらわす．

〔補題３．１〕ｎ個の割当文系列

$$w_1 = a_{11}; a_{12}; \cdots; a_{1t_1},$$
$$w_2 = a_{21}; a_{22}; \cdots; a_{2t_2},$$
$$\vdots$$
$$w_n = a_{n1}; a_{n2}; \cdots; a_{nt_n}$$

をシャッフルした結果のひとつwに対し，

$$(i, i') \in m(w)$$

かつ，

$$(i', j) \in m(P_1' \parallel P_2' \parallel \cdots \parallel P_n')$$

ならば，

$$(i, j) \in m(a_{11}; a_{12}; \cdots; a_{1t_1}; P_1' \parallel a_{21}; a_{22}; \cdots; a_{2t_2}; P_2' \parallel \cdots \parallel a_{n1}; a_{n2}; \cdots; a_{nt_n}; P_n').$$

（証明）wの長さに関する帰納法を用いる．

$|w|=1$の場合には，定義３．７（ii）より明らか．

$|w|=l+1$の場合には，

$w \equiv a_{k1};w'$，$|w'|=l$とするとき，帰納法の仮定は次のようになる．

$$(i'', i') \in m(w'),$$
$$(i', j) \in m(P_1' \parallel P_2' \parallel \cdots \parallel P_n')$$

ならば，

$$(i'', j) \in m(a_{11};a_{12};\cdots;a_{1t1};P_1' \parallel a_{21};a_{22};\cdots;a_{2t2};P_2' \parallel \cdots \parallel a_{k2};\cdots;a_{ktk};P_k' \parallel \cdots \parallel a_{n1};a_{n2};\cdots;a_{ntn};P_n').$$

いま，

$$(i, i') \in m(w).$$

である．すると，定義３．１２より，あるi″が存在して，

$$(i, i'') \in m(a_{k1}),$$
$$(i'', i') \in m(w').$$

をみたす．ここで帰納法の仮定より，

$$(i'', j) \in m(a_{11};a_{12};\cdots;a_{1t1};P_1' \parallel a_{21};a_{22};\cdots;a_{2t2};P_2' \parallel \cdots \parallel a_{k2};\cdots;a_{ktk}\ P_k' \parallel \cdots \parallel a_{n1};a_{n2};\cdots;a_{ntn};P_n').$$

定義３．７（ii）より，

$$(i, j) \in m(a_{11};a_{12};\cdots;a_{1t1};P_1' \parallel a_{21};a_{22};\cdots;a_{2t2};P_2' \parallel \cdots \parallel a_{k1};a_{k2};\cdots;a_{ktk}\ P_k' \parallel \cdots \parallel a_{n1};a_{n2};\cdots;a_{ntn};P_n').$$

（証明終）

〔補題３．２〕割当文の系列$w_1, w_2, \cdots, w_n$をつないだ系列

$$w \equiv w_1 ; w_2 ; \cdots ; w_n$$

について，（これは，$w_1, w_2, \cdots, w_n$のひとつのシャッフル結果である）

$$(i, j) \in m(w)$$

であるならば，$w_1, w_2, \cdots, w_n$ の任意のシャッフル結果w'について，

$$(i, j) \in m(w')$$

である．

（証明）次の事実に着目する．

w_1 ，w_2 ，… ，w_n の任意のシャッフル結果は，w_1 ，w_2 ，… ，w_n をつないだ系列

$$w \equiv w_1 ; w_2 ; \cdots ; w_n$$

が与えられたとき，wから次の操作を繰り返すことにより得ることができる．即ち，割当文の系列

$$u \equiv a_1; a_2; \cdots ; a_k ; a_{k+1}; \cdots ; a_t$$

のa_k とa_{k+1} が異なるプロセスP_l ，$P_{l'}$に含まれる命令であるとき（即ち，異なるw_l ，$w_{l'}$に含まれる命令であるとき），それらをいれかえて

$$u' \equiv a_1; a_2; \cdots ; a_{k+1}; a_k ; \cdots ; a_t$$

とする．

今

$$w \equiv a_1; a_2; \cdots ; a_{k-1} ; a_k ; a_{k+1}; a_{k+2}; \cdots ; a_t$$

かつ，a_k とa_{k+1} が異なるプロセスに含まれる命令であるとする．

定義３．１２より，

$$(i, j) \in m(w) \text{ iff}$$
$$(i, i_1) \in m(a_1; a_2; \cdots ; a_{k-1})$$
$$(i_1, i_2) \in m(a_k ; a_{k+1})$$
$$(i_2, j) \in m(a_{k+2}; \cdots ; a_t)$$

ここで，a_k とa_{k+1} は同じ変数を含まないので，

$$(i_1, i_2) \in m(a_{k+1} ; a_k)$$

定義３．１２より

$$(i, j) \in m(a_1; a_2; \cdots ; a_{k-1} ; a_{k+1} ; a_k ; a_{k+2}; \cdots ; a_t)$$

以上の繰り返しで，任意のw'について

$$(i, j) \in m(w')$$

（証明終）

〔定義３．１３〕入力条件Φ，出力条件Ψ，割当文の系列wについて，

$$\Phi \{w\} \Psi$$

iff

$i \models \Phi$，$(i, j) \in m(w)$ ならば $j \models \Psi$

〔補題３．３〕割当文の系列 w_1，w_2，…，w_n をつないだ系列

$$w \equiv w_1 ; w_2 ; \cdots ; w_n$$

について

$$\Phi \{w\} \Psi$$

ならば，w_1，w_2，…，w_n の任意のシャッフル結果w'について，

$$\Phi \{w'\} \Psi$$

（証明）定義３．１３及び補題３．２より明らか．　（証明終）

〔補題３．４〕$t \geq 2$ とするとき，

$$\Phi \{a_1; a_2; \cdots; a_k; \cdots; a_t\} \Psi$$

であるための必要十分条件は，あるχが存在して，

$$\Phi \{a_1\} \chi$$

かつ

$$\chi \{a_2; \cdots; a_k; \cdots; a_t\} \Psi.$$

（証明）if方向の証明については，定義３．１２及び３．１３より明らか．

only if 方向については，次のようにして証明される．

a_k が

$$x_k := \tau_k$$

という割当文であったとする．任意のΨについて

$$\Psi\ [x_k/\tau_k]\ \{x_k := \tau_k\}\ \Psi$$

を満たす．

従って，$a_1;\ a_2;\ \cdots;\ a_k;\ \cdots;a_t$ について，

$$\Psi\ [x_t/\tau_t]\ \{x_t := \tau_t\}\ \Psi$$

$$\Psi\ [x_t/\tau_t]\ [x_{t-1}/\tau_{t-1}]\ \{x_{t-1} := \tau_{t-1}\}\ \Psi\ [x_t/\tau_t]$$

$$\vdots$$

$$\Psi\ [x_t/\tau_t]\ [x_{t-1}/\tau_{-1}]\cdots[x_2/\tau_2]\ [x_1/\tau_1]\ \{x_1 := \tau_1\}$$

$$\Psi\ [x_t/\tau_t]\ [x_{t-1}/\tau_{t-1}]\cdots\cdots[x_2/\tau_2]$$

となる．ここで先に証明したif方向を繰り返し使うと，

$$\Psi\ [x_t/\tau_t]\ [x_{t-1}/\tau_{t-1}]\cdots\cdots[x_2/\tau_2]\ \{a_2;\ \cdots;\ a_k;\ \cdots;a_t\}\ \Psi$$

かつ

$$\Psi\ [x_t/\tau_t]\ [x_{t-1}/\tau_{-1}]\cdots[x_2/\tau_2]\ [x_1/\tau_1]\ \{a_1;\ \cdots;\ a_k;\ \cdots;\ a_t\}\ \Psi.$$

となる．ここで，

$$\Phi\ \{a_1;\ a_2;\ \cdots;\ a_k;\ \cdots;a_t\}\ \Psi$$

であるならば，

$$\Phi \supset \Psi\ [x_t/\tau_t]\ [x_{t-1}/\tau_{t-1}]\cdots[x_2/\tau_2]\ [x_1/\tau_1]$$

となることは容易にわかる．従って，χ を

$$\Psi\ [x_t/\tau_t]\ [x_{t-1}/\tau_{t-1}]\cdots\cdots[x_2/\tau_2]$$

とすれば

$$\Phi\ \{a_1\}\ \chi$$

かつ

$$\chi\ \{a_2;\ \cdots;\ a_k;\ \cdots;\ a_t\}\ \Psi.$$

となる．（証明終）

補題３．４より次のような推論規則を考える．

ＣＩ７）

$$\frac{\Phi\ \{a_1;\ \cdots;\ a_k;\ \cdots;\ a_{t-1}\}\ \chi \qquad \chi\ \{a_t\}\ \Psi}{\Phi\ \{a_1;\ \cdots;\ a_k;\ \cdots;\ a_{t-1};\ a_t\}\ \Psi}$$

この推論規則を先に示した公理系に組み入れて使用するために，推論規則ＣＩ１）の一部を次のように変形する．即ち，ＣＩ１）の前提ｉ）に代えて，次の前提ｉ’）を採用した公理系を考える．

ｉ’）w_1，w_2，…，w_nをｎ個の割当文の系列とすると，ＣＳＰプログラムＰが次のような形をしているとき，

$$w_1;P_1' \parallel$$
$$w_2;P_2' \parallel \cdots \parallel$$
$$w_n;P_n'$$

w_1，w_2，…，w_nをつないだ系列

$$w \equiv w_1\ ;\ w_2\ ;\ \cdots\ ;\ w_n$$

について

$$\frac{\Phi\ \{w\}\ \chi \qquad \chi\ \{P_1' \parallel P_2' \parallel \cdots \parallel P_n'\}\ \Psi}{\Phi\ \{w_1;P_1' \parallel w_2;P_2' \parallel \cdots \parallel w_n;P_n'\}\ \Psi}$$

〔定理３．２〕推論規則ＣＩ１）のｉ）をｉ’）に置き換えた推論規則及びＣＩ２）～ＣＩ７），公理ＣＡ１）～ＣＡ３）からなる公理系は無矛盾である．

（証明）規則を置き換えた公理系の証明図から，もとの公理系の証明図が構成できることを示す．即ち，証明図中に出現するすべての

$$\Phi\ \{w\}\ \chi$$

という形の式について，これを取り除くことが可能であることを示せばよい．ｗの長さに関する帰納法を用いる．

$|w| = 1$のとき，この場合は，ｉ’）はｉ）そのものである．

$|w| = t + 1$のとき，

$$\Phi\ \{w\}\ \chi$$

が成立すれば，補題３．３より，あらゆるｗ’について

$$\Phi\ \{w'\}\ \chi$$

が成り立つ．即ち，

$$\Phi\ \{w\}\ \chi$$

を根とする部分木を，それ自身の証明図とみると，同様なあらゆる

$$\Phi\ \{w'\}\ \chi$$

の証明図がＣＩ７）を何回か適用することによってられる．ここであるｗ’を固定して考え，

$$w' \equiv a\ ;\ w''$$

とすると，補題３．４より，

$$\Phi\ \{a\}\ \chi'$$

かつ，

$$\chi'\ \{w''\}\ \chi$$

となるχ’がそれぞれのｗ’について存在する．すなわち，

$$w_1;P_1'\ \|\ w_2;P_2'\ \|\ \cdots\ \|\ w_n;P_n'$$

のw_1，w_2，…，w_nのそれぞれ先頭となるどのa_iについても

$$\Phi\ \{a_i\}\ \chi'_i$$

$$\chi'_i\ \{w''_i\}\ \chi$$

を成立させるχ'_iが存在する．したがって証明図の根の子供の部分

$$\frac{\Phi\ \{w\}\ \chi \qquad\qquad \chi\ \{P_1'\ \|\ P_2'\ \|\ \cdots\ \|\ P_n'\}\ \Psi}{\Phi\ \{w_1;P_1'\ \|\ w_2;P_2'\ \|\ \cdots\ \|\ w_n;P_n'\}\ \Psi}$$

を図３．３のように変形することができる．

ここで，w''_1，……，w''_nはいずれもあるｗ’の先頭からそれぞれa_iを取り去ったものであり，長さはｗ’より短い（即ちｗより短い）．ここですべての，

$$\chi'_i\ \{w''_i\}\ \chi$$

にそれぞれ帰納法の仮定を適用して，ＣＩ１），ｉ）だけを使った証明図に書き改めることができる．

$$\dfrac{\chi'_i\ \{w''_i\}\ \chi \qquad \chi\ \{P_1' \parallel \cdots \parallel P_n'\}\ \Psi}{\dfrac{\cdots\ \Phi\ \{a_i\}\ \chi'_i \qquad \chi'_i\ \{w_1; P_1' \parallel \cdots \parallel w_i'; P_i' \parallel \cdots \parallel w_n; P_n'\}\ \Psi \quad \cdots\cdots}{\Phi\ \{w_1;\ P_1' \parallel \cdots \parallel w_i'; P_i' \parallel \cdots \parallel w_n; P_n'\}\ \Psi}}$$

図３．３　　規則ＣＩ１）ｉ）を使った証明図への変形

3．3　ＣＳＰのデッドロックフリー性の検証

前節ではＣＳＰのセマンティクスとして，状態集合のうえの二項関係を定義し，部分的正当性の検証体系を与えた．しかし，この論理体系では，そのセマンティクスが命令の実行に関する情報を表現できないために，ＣＳＰのデッドロックフリー性を扱うことはできない．例えば次のようなＣＳＰを考える．

P_1 ≡ if true ; skip → if true ; P_2!1→ x := 1 fi
　　□ true ; skip→ x := 1 fi

P_1' ≡ if true ; skip → x := 1 fi

P_2 ≡ if true ; skip → z := 1 fi

このとき，$P_1 \parallel P_2$ と$P_1' \parallel P_2$ について，両者に現れるプログラム変数はいずれもｘとｚで同じであり，また，

$$m(P_1 \parallel P_2) = m(P_1' \parallel P_2) = \{(i, j) \mid j \models x = 1 \wedge z = 1\},$$

となる．しかし$P_1 \parallel P_2$ はif文で枝別れがあり，１行目の

true ; skip → if true ; P_2!1→ x := 1 fi

を実行時に選択した場合，P_2!1の実行ができず，デッドロックとなる．このようにプログラムがデッドロックを起こすか否かは，状態集合上の二項関係だけからは判断できない．

そこで，本節では，ＣＳＰの実行可能な命令系列と，その命令系列を実行途中に通る状態も含めて記録するセマンティクスとして，計算履歴集合を，やはり前節と同様にして，centralized approachによって定義する．続いて，ＣＳＰの部分的正当性とデッドロックフリー性を，同時に検証する公理系を与え，その無矛盾性(sound－ness）を示す．

ここで定義する計算履歴は，２章で定義したＲＧＣの計算履歴とは微妙に異なっている．即ち⊥の使い方について，ＲＧＣの場合は単に実行の選択が出来ないことを意味したが，ＣＳＰの計算履歴の場合は後にわかるように⊥はデッドロックを意味する．また計算履歴集合についてもＲＧＣの場合はプレフィクスをも含めた定義とな

っていたが，ＣＳＰではプレフィクスは含めない．記法については計算履歴に関しては，定義の都合上，状態とその状態から実行された命令を対にして＜，＞で囲むことに変えるが，集合については特に混乱はないと思われるので，同じ記法を用いる．

３．３．１　計算履歴集合

〔定義３．１４〕次のような形の系列を，計算履歴という．

$\langle i_1, e_1\rangle\langle i_2, e_2\rangle \cdots\cdots \langle i_{n-1}, e_{n-1}\rangle\langle i_n, \varepsilon\rangle$

$\langle i_1, e_1\rangle\langle i_2, e_2\rangle \cdots\cdots \langle i_{n-1}, e_{n-1}\rangle\langle i_n, \perp\rangle$

ただし，e_k は割り当て文または $\mu(C_k, C_l)$ を満たす C_k，C_l の通信 $C_k \leftrightarrow C_l$ で，あり，$0 < h \leq n$ について

$$(i_h, i_{h+1}) \in m(e_h).$$

〔定義３．１５〕ＣＳＰプログラム $P \equiv P_1 \parallel P_2 \parallel \cdots \parallel P_n$，および状態の集合 $I \subseteq U_P$ に対して，計算履歴の集合Ｈ（Ｐ，Ｉ）を次のように定義する．

（ｉ）$H(\Lambda \parallel \cdots \parallel \Lambda, I) = \{\langle i, \varepsilon\rangle \mid i \in I\}$．

（ii）$P \equiv P_1 \parallel \cdots \parallel x := \tau; P_l' \parallel \cdots \parallel P_n$ のとき，

$$i \in I,\ (i, i') \in m(x := \tau) \cdots\cdots\cdots\cdots\cdots\cdots ※$$

かつ，$\langle i', e\rangle$ で始まる計算履歴（以下 $\langle i', e\rangle$ ｗのように表す）が，

$$H(P_1 \parallel \cdots \parallel P_l' \parallel \cdots \parallel P_n, \{i'\})$$

の元であるとき，

$\langle i, x := \tau\rangle\langle i', e\rangle$ $w \in H(P_1 \parallel \cdots \parallel x := \tau;\ P_l' \parallel \cdots \parallel P_n,\ I)$

（iii）$P \equiv P_1 \parallel \cdots \parallel \underline{\text{if}} \cdots \square B_k; \text{skip} \rightarrow S_k \square \cdots \underline{\text{fi}};\ P_m' \parallel \cdots \parallel P_n$

のとき，$i \in I$ で，

$$i \models B_k \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots ※$$

であり，かつ，

$$w \in H(P_1 \parallel \cdots \parallel S_k;\ P_m' \parallel \cdots \parallel P_n,\ \{i\})$$

ならば，ｗは，

$$H(P_1 \parallel \cdots \parallel \underline{\text{if}} \cdots \square B_k; \text{skip} \rightarrow S_k \square \cdots \underline{\text{fi}}; P_m' \parallel \cdots \parallel P_n\ I)$$

の元である．

（iv） $P \equiv P_1 \| \cdots \| \underline{do} \cdots \square B_k ; skip \rightarrow S_k \square \cdots \underline{od} ; P_m' \| \cdots \| P_n$

のとき，$i \in I$ で

$$i \models B_k \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots ※$$

であり，かつ，

$w \in H(P_1 \| \cdots \| S_k ; \underline{do} \cdots \square B_k ; skip \rightarrow S_k \square \cdots \underline{od} ; P_m' \| \cdots \| P_n , \{i\})$

ならば，wは

$H(P_1 \| \cdots \underline{do} \cdots \square B_k ; skip \rightarrow S_k \square \cdots \underline{od} ; P_m' \| \cdots \| P_n , I)$

の元である．

（v） $P \equiv P_1 \| \cdots \| \underline{do} B_1 ; C_1 \rightarrow S_1 \square \cdots \square B_k ; C_k \rightarrow S_k \underline{od} ; P_m' \| \cdots \| P_n$

の場合，

$i \in I$，かつ $1 \leq h \leq k$ なるすべての B_h について $i \models B_h$ ならば C_h は通信命令でかつその通信相手のプロセスがすべてΛであるとき，（即ち，$i \models LE_m$ であるとき） ┐
……※ ┘

wが$H(P_1 \| \cdots \| P_m' \| \cdots \| P_n , \{i\})$の元ならば

$w \in H(P_1 \| \cdots \| \underline{do} B_1 ; C_1 \rightarrow S_1 \square \cdots \square B_k ; C_k \rightarrow S_k \underline{od} ; P_m' \| \cdots \| P_n , I)$

（vi）プロセス P_i が，

$$\underline{do} B_1 ; C_1 \rightarrow S_1 \square \cdots \square B_k ; C_k \rightarrow S_k \underline{od} ; P_i'$$

または

$$\underline{if} B_1 ; C_1 \rightarrow S_1 \square \cdots \square B_k ; C_k \rightarrow S_k \underline{fi} ; P_i'$$

のとき

$$[B_1 ; C_1 \rightarrow S_1 \square \cdots \square B_k ; C_k \rightarrow S_k] ; P_i'$$

と書く．

また，上のような形をしたP_iについて，記法$P_i\ /k$をそれぞれ次のように定める．

$(\underline{do}\, B_1; C_1 \to S_1 \,\square\cdots\square\, B_k;\ C_k \to S_k \,\square\cdots\square\, B_l;$
$C_l \to S_l\ \underline{od};\ P_i')\ /k$

は，

$S_k;\ \underline{do}\, B_1; C_1 \to S_1 \,\square\cdots\square\, B_k;\ C_k \to S_k \,\square\cdots$
$\square\, B_l;\ C_l \to S_l\ \underline{od};\ P_i'$

を意味する．また，

$(\ \underline{if}\, B_1; C_1 \to S_1 \,\square\cdots\square\, B_k; C_k \to S_k \,\square\cdots\square$
$B_l; C_l \to S_l \underline{fi}; P_i')\ /k$

は，

$S_k; P_i'$

を意味する．以上の記法のもとで，

$P \equiv P_1 \parallel \cdots \parallel [\cdots\square B_k; C_k \to S_k \square\cdots];\ P_m' \parallel$
$\cdots \parallel [\cdots\square B_l; C_l \to S_l \square\cdots];\ P_s' \parallel \cdots \parallel P_n$

の場合，

$i \in I,\ i \models B_k \wedge B_l,\ \mu(C_k, C_l)$
かつ，$(i,i') \in m(C_k \leftrightarrow C_l)$ のとき，　……※

$\langle i', e\rangle w$が

$H(P_1 \parallel \cdots \parallel$
$[\cdots\square B_k; C_k \to S_k \square\cdots];P_m'/k \parallel \cdots \parallel$
$[\cdots\square B_l; C_l \to S_l \square\cdots];P_s'/l \parallel \cdots \parallel P_n, \{i'\})$

の元ならば，

$\langle i, C_k \leftrightarrow C_l\rangle\langle i', e\rangle$ wは

$H(P_1 \parallel \cdots \parallel$
$[\cdots\square B_k; C_k \to S_k \square\cdots];P_m' \parallel \cdots \parallel$
$[\cdots\square B_l; C_l \to S_l \square\cdots];P_s' \parallel \cdots \parallel P_n, I)$

の元である．

（vii） $P \equiv \Lambda \parallel \cdots \parallel \Lambda$でなく，かつ$i \in I$について（ii）～（vi）のいかなる※の条件もあてはまらないとき，

$$\langle i, \perp \rangle \in H(P, I).$$

〔例〕次のP_1，P_2で定められるＣＳＰプログラム$P \equiv P_1 \parallel P_2$について，

$P_1 \equiv$ do $x > 1$; $P_2!x \rightarrow x := x-1$ od

$P_2 \equiv$ $z := 1$; do true; $P_1?y \rightarrow z := z * y$ do

（$\equiv z := 1$; P_2'と書く）

このとき，状態 i を，

$$i \models x = 3, \quad j \models z = 6 \wedge x = 1 \wedge y = 2$$

とすると，

$\langle j, \varepsilon \rangle \in H(\Lambda \parallel \Lambda, \{j\})$　〔定義３．１５（ｉ）〕

また，

$$j \models LE_2$$

より，

$\langle j, \varepsilon \rangle \in H(\Lambda \parallel P_2', \{j\})$　〔定義３．１５（ｖ）〕

さらに，

$$j \not\models x > 1 \quad (\models LE_1)$$

より

$\langle j, \varepsilon \rangle \in H(P_1 \parallel P_2', \{j\})$　〔定義３．１５（ｖ）〕

また

$$j_1 \models z = 3 \wedge x = 1 \wedge y = 2,$$

かつ，

$$j_2 \models z = 3 \wedge x = 2 \wedge y = 2$$

とすると，

$$(j_1, j) \in m(z := z * y),$$

かつ，

$$(j_2, j_1) \in m(x := x-1)$$

より

$$\langle j_1, z := z * y \rangle \langle j, \varepsilon \rangle \in H(P_1 \parallel P_2', \{j\})$$

$$\langle j_2, x := x-1 \rangle \langle j_1, z := z * y \rangle \langle j, \varepsilon \rangle \in H(P_1 \parallel P_2', \{j_2\})$$

〔定義３．１５（ⅱ）２回適用〕

ここで，

$$j_3 \models z = 3 \wedge x = 2 \wedge y = 3$$

とすると

$$j_3 \models x > 1,$$

$$(j_3, j_2) \in m\ (P_2!x \leftrightarrow P_1?y)$$

より，

$$\langle j_3,\ P_2!x \leftrightarrow P_1?y \rangle \langle j_2,\ x := x-1 \rangle$$

$$\langle j_1, z := z * y \rangle \langle j,\ \varepsilon \rangle \in H(P_1 \parallel P_2',\ \{j_3\})$$

以下同様の繰り返しで，

$$\langle i, z := 1 \rangle \cdots \langle j_3, P_2!\ x \leftrightarrow P_1?\ y \rangle \langle j_2, x := x-1 \rangle$$

$$\langle j_1, z := z * y \rangle \langle j,\ \varepsilon \rangle \in H(\ P_1 \parallel P_2,\quad \{i\})$$

となる．

3．3．2　デッドロックフリー性の検証

前節で定義した計算履歴集合を用いてデッドロックを形式的に特徴づけ，その定義のもとでのデッドロックフリー性の検証体系を示す．

〔定義３．１６〕あるＣＳＰプログラムが状態ｉでデッドロックであるとは，少なくともひとつΛでないプロセスが存在し，任意のΛでないプロセスP_kについて，ステートメントの先頭がif文かdo文で$i \models LE_k$かつｉで真となる防護に続く命令がすべて通信命令で，かつ，これらの通信命令のどの対C_l, C_kについても$\mu(C_l, C_k)$が成り立たないことをいう．

あるＣＳＰプログラム$P \equiv P_1 \parallel P_2 \parallel \cdots \parallel P_n$が入力条件Φのもとでデッドロックフリーであるとは，Φの成り立つ任意の状態ｉからＰを実行したとき，途中で通るいかなる状態でもプログラムの残りの部分がデッドロックにならないことをいう．すなわちすべてのw∈Ｈ（Ｐ，｛ｉ｜ｉ⊨Φ｝）についてｗの最後が<ｊ，⊥>でないことをいう．また，Ｐが入力条件Φと出力条件Ψについて部分的正当であることは次のように表せる．すなわち，任意の

$$\langle i,e\rangle \cdots\cdots \langle j,\ \varepsilon\rangle \in H\ (P,\ \{i \mid i \models \Phi\})$$

について，$i \models \Phi$ならば$j \models \Psi$である．

この定義は３．１節の部分的正当性の定義と等価になる．

以下では，PがΦ，Ψについて部分的正当かつΦについてデッドロックフリーであることを，

$$\Phi\ 【P】\ \Psi$$

とあらわす．前節同様，この形の式と述語をあわせて論理式と呼ぶ．

以下ではＣＳＰの部分的正当性及びデッドロックフリー性について，計算履歴集合を定義した手順と同様なcentralized approachにそって推論を進める公理系を以下に与える．この公理系は前節で示した部分的正当性の検証体系の推論規則の一部について，前提となる式を付け加えたものである．したがって以下に示す体系での証明は付け加えた前提以外の部分についてはそのまま３．１節に示した部分的正当性の証明体系での証明となっている．

以下に公理と推論規則を示す．

ＣＪ１）

$$\frac{\Gamma}{\Phi\ 【P】\ \Psi}$$

ここで，Γは論理式の並びで，次のｉ）～ｖ）にあてはまる式の並びをすべて含む．

ｉ）～iv）部分的正当性の推論規則ＣＩ１）のｉ）～iv）に示す式の｛，｝をすべて【，】に置き換えて得られる式の並び．

ｖ）PのΛでないすべてのプロセスについて，その先頭がif文であるとき，

$$\Phi \supset (\bigvee_{l,h} (B_h \wedge B_l)) \vee \bigvee_l B_l$$

ここで，$\bigvee_{l,h}$は$\mu(C_l, C_h)$なるl,hについての論理和，$\bigvee_l$はC_lがskip文であるようなlについての論理和を表す．

この式をブロッキングフリー条件という．

ＣＪ２）

$$\frac{\Phi \supset \Psi\,[x/\tau]}{\Phi\,【\,x := \tau\,】\,\Psi}$$

ここで，$\Psi\,[x/\tau]$ はΨのx のすべての自由な出現にτを代入した式である．以下，同様の記法を用いる．

ＣＪ３）C_k をＰ！τ，C_l をＱ？xのとき，

$$\frac{\Phi \supset \Psi\,[x/\tau]}{\Phi\,【\,C_k \rightsquigarrow C_l\,】\,\Psi}$$

ＣＪ４）

$$\frac{\Phi \supset \Phi' \qquad \Phi'\,【\,P\,】\,\Psi}{\Phi\,【\,P\,】\,\Psi}$$

ＣＪ５）

$$\frac{\Phi\,【\,P\,】\,\Phi' \qquad \Phi' \supset \Psi}{\Phi\,【\,P\,】\,\Psi}$$

ＣＪ６）

$$\frac{\Phi \supset \Psi}{\Phi\,【\,\Lambda \parallel \cdots \parallel \Lambda\,】\,\Psi}$$

公理として次のＡ’１），Ａ’２），Ａ’３）を採用する．

Ａ’１）任意のＣＳＰプログラムＰ，任意の述語Ψについて，

$$\mathrm{false}\;【\,P\,】\,\Psi.$$

Ａ’２）Ｐが推論規則ＣＪ１）のｉ）～ｉｖ）の条件のいずれもあてはまらないとき，任意の述語Φ，Ψについて，

$$\Phi\;【\,P\,】\,\Psi.$$

Ａ’３）自然数論のすべての定理．

〔定義３．１７〕論理式Φ【Ｐ】Ψに対し，次のような論理式を節点のラベルとして持つ木をその証明図という．

⑴根のラベルはΦ【Ｐ】Ψである．

⑵すべての内部節点のラベルは，その子供のラベルを前提として直接に推論される．

⑶すべての葉は，ある内部節点で使われた推論規則

ＣＪ１）のｉｖ）の前提

$\Phi \wedge \neg L E_m$ 【$P_1 \| \cdots \|$

$\underline{if} B_1; C_1 \rightarrow S_1 \Box \cdots \Box B_k; C_k \rightarrow S_k \underline{fi};$

$\underline{do} B_1; C_1 \rightarrow S_1 \Box \cdots \Box B_k; C_k \rightarrow S_k \underline{od}; P_m'$

$\| \cdots \| P_n$ 】Ψ

の子孫であり，かつそのラベルは

Φ 【$P_1 \| \cdots \| \underline{do} B_1; C_1 \rightarrow S_1 \Box \cdots$

$\Box B_k; C_k \rightarrow S_k \underline{od}; P_m' \| \cdots \| P_n$ 】Ψ

であるか，または公理をラベルとしてもつ．

この公理系は３．１節で示したＣＳＰの部分的正当性の検証体系に，推論規則ＣＪ１）の前提ｖ）をつけ加えたものとなっている．すなわち，Φ【Ｐ】Ψの証明図が存在することは，Φ｛Ｐ｝Ψの証明図が存在し，かつ，その証明図に現れるすべてのχ｛Ｐ'｝Ψという形の式について，Ｐ'の空でないすべてのプロセスの先頭がｉｆ文であるならばブロッキングフリー条件が成立すること（このような証明図をブロッキングフリーな証明図と呼ぶ）と等価である．

以下に，この体系が部分的正当性とデッドロックフリー性を同時に検証するものであることを示す．

〔補題３．５〕Φ【Ｐ】Ψの証明図が存在すればＰは入力条件Φと出力条件Ψについて部分的正当である．

（証明）Φ【Ｐ】Ψの証明図はそれ自身Φ｛Ｐ｝Ψの証明図である．定理３．１よりＰは入力条件Φと出力条件Ψについて部分的正当である．（証明終）

〔補題３．６〕あるｗ$\in$Ｈ（Ｐ，｛ｉ｜ｉ$\models\Phi$｝）について，ｗの最後が$\langle j, \perp \rangle$であれば，Φ｛Ｐ｝Ψのブロッキングフリーな証明図は存在しない．

（証明）$\langle j, \perp \rangle$で終わるｗのもっとも短いものについて，それがＨ（Ｐ，Ｉ）の元であることがわかるまでに定義３．１５の規則が適用された回数に関する帰納法を用いて証明する．

(1)　　$\langle j, \perp \rangle \in$ Ｈ（Ｐ，｛ｉ｜ｉ$\models\Phi$｝）

であるとき，定義３．１５よりＰのΛでないすべてのプロセスにつ

いて先頭はif文かdo文であり，k 番目のプロセスの先頭がdo文であるならばそのdo文についてＬＥ$_k$ はｊで成立しない（そうでなければ定義３．１５ｖ）が適用できる）．　Φ｛Ｐ｝Ψのいかなる証明図も次のような部分をもつ．すなわちすべてのdo文について推論規則ＣＪ１）のiv）（の【，】を｛，｝に置き換えた規則，即ちＣＩ１）iv））を何回か適用し，すべてのΛでないプロセスの先頭はif文となっている（図３．４参照）．このとき，ｉではこの図の最上段でのブロッキングフリー条件が成立しない．すなわち，μ（Ｃ$_l$，Ｃ$_h$）の成立するｌ，ｈで，$\bigvee_{l,h}$（Ｂ$_h$ ∧ Ｂ$_l$）が成立すれば，Ｐに関して定義３．１５のiv）が適用でき，また，Ｃ$_l$ がskip文であるｌについて$\bigvee_l$Ｂ$_l$ が成立するなら，定義３．１５の（iii）または（iv）が適用できる．ゆえに，Φ｛Ｐ｝Ψのブロッキングフリーな証明図は存在しない．

(2)　帰納法の仮定として，

「Ｈ（Ｐ’，｛ｉ｜ｉ⊨Ｘ｝）に$\langle j, \perp \rangle$ で終る元が含まれることが定義３．１５のｔ回以下の適用でわかったとする．そのとき，

Ｘ｛Ｐ’｝Ψ

のブロッキングフリーな証明図は存在しない．」と仮定する．

$$\frac{\neg LE_1 \wedge \cdots \wedge \neg LE_n \wedge \Phi \ \{\text{if}\cdots*\cdots\text{fi};\text{do}\cdots*\cdots\text{od};P_1 \parallel \text{if}\cdots*\cdots\text{fi};\text{do}\cdots*\cdots\text{od};P_2 \parallel \cdots \parallel \text{if}\cdots*\cdots\text{fi};\text{do}\cdots*\cdots\text{od};P_n\}\ \text{true}}{\cdots\cdots\cdots\cdots\cdots\cdots}$$

$$\frac{\neg LE_1 \wedge \neg LE_2 \wedge \Phi \ \{\text{if}\cdots*\cdots\text{fi};\text{do}\cdots*\cdots\text{od};P_1 \parallel \text{if}\cdots*\cdots\text{fi};\text{do}\cdots*\cdots\text{od};P_2 \parallel \cdots \parallel \text{do}\cdots*\cdots\text{od};P_n\}\ \text{true}}{\neg LE_1 \wedge \Phi \ \{\text{if}\cdots*\cdots\text{fi};\text{do}\cdots*\cdots\text{od};P_1 \parallel \text{do}\cdots*\cdots\text{od};P_2 \parallel \cdots \parallel \text{do}\cdots*\cdots\text{od};P_n\}\ \text{true}}$$

$$\Phi \ \{\text{do}\cdots*\cdots\text{od};P_1 \parallel \cdots \parallel \text{do}\cdots*\cdots\text{od};P_m \parallel \cdots \parallel \text{do}\cdots*\cdots\text{od};P_n\}\ \text{true}$$

図３．４　推論規則Ｉ１）iv）の適用

そこで，t＋1回目に適用されたのが（ii）～（vi）のいずれであったかによって場合わけをする．

（ii）の場合には，

$\langle i, x := \tau \rangle \langle i', e \rangle \cdots\cdots \langle j, \perp \rangle \in H(P_1 \parallel \cdots \parallel x := \tau ; P_l' \parallel \cdots \parallel P_n , \{ i \mid i \models \Phi \})$．

であり，証明図は次のような部分をもつ．

$$\frac{\Phi \{ x := \tau \} X \qquad \cdots\cdots\cdots\cdots \qquad X \{ P_1 \parallel \cdots \parallel P_l' \parallel \cdots \parallel P_n \} \Psi}{\Phi \{ P_1 \parallel \cdots \parallel x := \tau ; P_l' \parallel \cdots \parallel P_n \} \Psi}$$

ここで，

$$X \{ P_1 \parallel \cdots \parallel P_l' \parallel \cdots \parallel P_n \} \Psi$$

を根とする部分木は，それ自身の証明図となる．したがって，もし，

$$\Phi \{ P_1 \parallel \cdots \parallel x := \tau ; P_l' \parallel \cdots \parallel P_n \} \Psi$$

のブロッキングフリーな証明が存在したとすると，

$$X \{ P_1 \parallel \cdots \parallel P_l' \parallel \cdots \parallel P_n \} \Psi$$

のブロッキングフリーな証明図が存在する．しかし，定義3．15のii）より，

$\langle i', e \rangle \cdots \langle j, \perp \rangle \in H(P_1 \parallel \cdots \parallel P_l' \parallel \cdots \parallel P_n , \{ i' \})$

一方，

$$(i, i') \in m (x := \tau)$$

のとき，

$$i' \models X$$

より，

$\langle i', e \rangle \cdots \langle j, \perp \rangle \in H(P_1 \parallel \cdots \parallel P_l' \parallel \cdots \parallel P_n , \{ i' \mid i' \models X \})$

となり，これは帰納法の仮定に反する．

（iii）の場合には，

$\langle i , e \rangle \cdots \langle j, \perp \rangle \in H(P_1 \parallel \cdots \parallel \underline{if} \cdots \Box B_k ; skip \rightarrow S_k \Box \cdots \underline{fi}$

$$; P_m' \| \cdots \| P_n , \{ i \mid i \models X \})$$

定義３．１５のiii）より，

$$\langle i , e \rangle \cdots \langle j , \perp \rangle \in H(P_1 \| \cdots \| S_k ; P_m' \| \cdots \| P_n , \{ i \}).$$

であり，また

$$i \models \Phi \wedge B_k$$

より

$$\langle i , e \rangle \cdots \langle j , \perp \rangle \in H(P_1 \| \cdots \| S_k ; P_m' \| \cdots \| P_n , \{ i \mid i \models \Phi \wedge B_k \}).$$

である．このとき証明図は次のようになる．

$$\frac{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots}{\Phi \wedge B_k \{ P_1 \| \cdots \| S_k ; P_m' \| \cdots \| P_n \} \Psi}$$

$$\Phi \{ P_1 \| \cdots \| \underline{\text{if}} \cdots \Box B_k ; \text{skip} \rightarrow S_k \Box \cdots \underline{\text{fi}} ; P_m' \| \cdots \| P_n \} \Psi$$

ゆえに，

$$\Phi \{ P_1 \| \cdots \| \underline{\text{if}} \cdots \Box B_k ; \text{skip} \rightarrow S_k \Box \cdots \underline{\text{fi}} ; P_m' \| \cdots \| P_n \} \Psi$$

のブロッキングフリーな証明が存在したとすると，（ii）の場合と同様にして，

$$\Phi \wedge B_k \{ P_1 \| \cdots \| S_k ; P_m' \| \cdots \| P_n \} \Psi$$

のブロッキングフリーな証明図が得られることになり，これは，帰納法の仮定に反する．

（iv）の場合には，

$$\langle i , e \rangle \cdots \langle j , \perp \rangle \in H(P_1 \| \cdots \| \underline{\text{do}} \cdots \Box B_k ; \text{skip} \rightarrow S_k \Box \cdots \underline{\text{od}} ; P_m' \| \cdots \| P_n , \{ i \mid i \models X \})$$

である．定義３．１５のiv）より，

$$\langle i , e \rangle \cdots\cdots \langle j , \perp \rangle \in H(P_1 \| \cdots \| S_k ; \underline{\text{do}} \cdots \underline{\text{od}} ; P_m' \| \cdots \| P_n , \{ i \})$$

であり，また，

$$i \models B_k$$

である．

$$B_k \supset \neg L E_m$$

より，

$\langle i, e\rangle \cdots \langle j, \perp\rangle \in H(P_1 \| \cdots \| S_k ; \underline{do} \cdots \underline{od}; P_m' \| \cdots \| P_n, \{i \mid i \models \neg L E_m \wedge \Phi \wedge B_k\})$

である．一方，証明図は次のような部分をもつ．

$$\frac{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots}{\neg L E_m \wedge \Phi \wedge B_k \{P_1 \| \cdots \| S_k; \underline{do} \cdots \underline{od}; P_m' \| \cdots \| P_n\} \Psi}$$

$$\frac{\neg L E_m \wedge \Phi \{P_1 \| \cdots \| \underline{if} \cdots \square B_k ; skip \rightarrow S_k \square \cdots fi; do \cdots od; P_m' \| \cdots \| P_n\} \Psi}{\Phi \{P_1 \| \cdots \| \underline{do} \cdots \square B_k ; skip \rightarrow S_k \square \cdots \underline{od}; P_m' \| \cdots \| P_n\} \Psi}$$

ここで，

$\Phi \{P_1 \| \cdots \| \underline{do} \cdots \square B_k ; skip \rightarrow S_k \square \cdots \underline{od}; P_m' \| \cdots \| P_n\} \Psi$

のブロッキングフリーな証明図が存在したとする．この証明図で，

$\neg L E_m \wedge \Phi \wedge B_k \{P_1 \| \cdots \| S_k; \underline{do} \cdots \underline{od}; P_m' \| \cdots \| P_n\} \Psi$

を根とする部分木は，それ自身その証明図になるとは限らない．というのは，定義３．１０の⑶より，葉に

$\Phi \{P_1 \| \cdots \| \underline{do} \cdots \square B_k ; skip \rightarrow S_k \square \cdots \underline{od}; P_m' \| \cdots \| P_n\} \Psi$

というラベルを持つかもしれない．そのような場合は次のような操作を行うことによって，新たに，

$\neg L E_m \wedge \Phi \wedge B_k \{P_1 \| \cdots \| S_k ; \underline{do} \cdots \underline{od}; P_m' \| \cdots \| P_n\} \Psi$

の証明図を作ることができる．すなわち，定理３．１の証明のときと同様に．

$\Phi \{P_1 \| \cdots \| \underline{do} \cdots \square B_k ; skip \rightarrow S_k \square \cdots \underline{od}; P_m' \| \cdots \| P_n\} \Psi$

の証明図自身をその葉に出現する同じ式のうえにのせる（図３．１参照）．　このようにして得られた，

$\neg LE_m \wedge \Phi \wedge B_k$ $\{P_1 \| \cdots \| S_k; \underline{do} \cdots \underline{od}; P_m' \| \cdots \| P_n\}$ Ψ
の証明図はもとの証明図がブロッキングフリーならばやはりブロッキングフリーである．これは帰納法の仮定に反する．

（v）の場合には，

$\langle i, e \rangle \cdots\cdots \langle j, \perp \rangle \in H(P_1 \| \cdots \| \underline{do} \cdots \underline{od}; P_m' \| \cdots \| P_n, \{i \mid i \models \Phi\})$

である．定義3．15（v）により，

$\langle i, e \rangle \cdots\cdots \langle j, \perp \rangle \in H(P_1 \| \cdots \| P_m' \| \cdots \| P_n, \{i\})$，

又，

$$i \models LE_m$$

より，

$\langle i, e \rangle \cdots \langle j, \perp \rangle \in H(P_1 \| \cdots \| P_m' \| \cdots \| P_n, \{i \mid i \models LE_m \wedge \Phi\})$．

一方，証明図は次のような部分を含む．

$$\frac{LE_m \wedge \Phi \ \{P_1 \| \cdots \| P_m' \| \cdots \| P_n\} \ \Psi}{\Phi \ \{P_1 \| \cdots \| \underline{do} \cdots \underline{od}; P_m' \| \cdots \| P_n\} \ \Psi}$$

これがブロッキングフリーな証明図であったと仮定する．すると，

$LE_m \wedge \Phi \ \{P_1 \| \cdots \| P_m' \| \cdots \| P_n\} \ \Psi$

を根とする部分木から，

$LE_m \wedge \Phi \ \{P_1 \| \cdots \| P_m' \| \cdots \| P_n\} \ \Psi$

のブロッキングフリーな証明が得られ，帰納法の仮定に反する．

（vi）の場合には，

$\langle i, C_k \leftrightarrow C_l \rangle \langle i', e \rangle \cdots\cdots \langle j, \perp \rangle \in H(P_1 \|$
$\cdots \| [\cdots \Box B_k; C_k \rightarrow S_k \Box \cdots]; P_m' \|$
$\cdots \| [\cdots \Box B_l; C_l \rightarrow S_l \Box \cdots]; P_s' \|$
$\cdots \| P_n, \{i \mid i \models \Phi\})$．

である．ここで，

$$\mu\ (C_k\ ,\ C_l)\ ,\ i \models B_k \wedge B_l$$

かつ

$$(i,i') \in m\ (C_k \leftrightarrow C_l)$$

である．

C_k，C_lは，いずれもdo文に含まれる場合とif文に含まれる場合がある．今，C_kがif文，C_lがdo文の場合を考える．すると，証明図は図３．５のようになる．ここで，

$$\Phi \wedge L\ E_s \wedge B_k \wedge B_l\ \{C_k \leftrightarrow C_l\}\ X,$$

より

$$(i,i') \in m\ (C_k \leftrightarrow C_l)\ .$$

定義３．１５の（vi）より，

$\langle i', e\rangle \cdots\cdots \langle j, \perp\rangle \in H(\ P_1 \| \cdots \| S_k; P_m'$

$\| \cdots \| S_l;$ do$\cdots \square B_l; C_l \rightarrow S_l \square \cdots$od;

$P_s' \| \cdots \| P_n\ ,\ \{i'\}\ )$．

したがって，

$$i' \models X\ .$$

$$\frac{X\ \{P_1 \| \cdots \| S_k;\ P_m' \| \cdots \| S_l;\ \text{do}\cdots\text{od};\ P_s' \| \cdots \| P_n\}\ \Psi \qquad \Phi \wedge \neg L\ E_m \wedge B_k \wedge B_l\ \{C_k \leftrightarrow C_l\}\ X}{\Phi \wedge \neg L\ E_m\ \{P_1 \| \cdots \| \text{if}\cdots \square B_k; C_k \rightarrow S_k \square \cdots \text{fi}; P_m' \| \cdots \| \text{if}\cdots \square B_l; C_l \rightarrow S_l \square \cdots \text{fi}; \text{do}\cdots \square B_l; C_l \rightarrow S_l \square \cdots \text{od}; P_s' \| \cdots \| P_n\}\ \Psi}$$

$$\frac{\Phi\ \{P_1 \| \cdots \| \text{if}\cdots \square B_k; C_k \rightarrow S_k \square \cdots \text{fi}; P_m' \| \cdots \| \text{do}\cdots \square B_l; C_l \rightarrow S_l \square \cdots \text{od}; P_s' \| \cdots \| P_n\}\ \Psi}{}$$

図３．５　証明図

いま，

Φ {$P_1 \| \cdots \| \underline{\text{if}} \cdots \square B_k ; C_k \rightarrow S_k \square \cdots \underline{\text{fi}} ;$
$P_m' \| \cdots \| \underline{\text{do}} \cdots \square B_l ; C_l \rightarrow S_l \square \cdots \underline{\text{od}} ;$
$P_s' \| \cdots \| P_n$} Ψ

の証明で，ブロッキングフリーなものが存在したとすると，（iv）の場合と同様に，

X {$P_1 \| \cdots \| S_k ; P_m' \| \cdots \| S_l ; \underline{\text{do}} \cdots \underline{\text{od}} ; P_s' \| \cdots \| P_n$} Ψ

を根とする部分木から，ブロッキングフリーな

X {$P_1 \| \cdots \| S_k ; P_m' \| \cdots \| S_l ; \underline{\text{do}} \cdots \underline{\text{od}} ; P_s' \| \cdots \| P_n$} Ψ

の証明が得られる．これは帰納法の仮定に反する．

C_k，C_l がいずれもif文に含まれる場合は，

X {$P_1 \| \cdots \| S_k ; P_m' \| \cdots \| S_l ; P_s' \| \cdots \| P_n$} Ψ

の証明図がそのまま得られ上と同様の議論となる．

C_k，C_l がいずれもdo文に含まれる場合は，証明を図３．６のような変形を繰り返すことによって，

X {$P_1 \| \cdots \| S_k ; \underline{\text{do}} \cdots \underline{\text{od}} ; P_m' \| \cdots \| S_l ; \underline{\text{do}} \cdots \underline{\text{od}} ; P_s' \| \cdots \| P_n$} Ψ

の証明図を得ることができる．（証明終）

〔定理３．３　〕Φ【P】Ψの証明図が存在すれば，PはΦとΨについて部分的正当かつデッドロックフリーである．

（証明）Φ【P】Ψの証明図はブロッキングフリー条件の部分を除けば，そのままΦ {P} Ψの証明図として読める．従って，補題３．４より部分的正当が保証される．また，Φ【P】Ψの証明図が存在すれば，

Φ {P} Ψのブロッキングフリーな証明図が存在することになり補題３．５よりPはデッドロックフリーである．

（証明終）

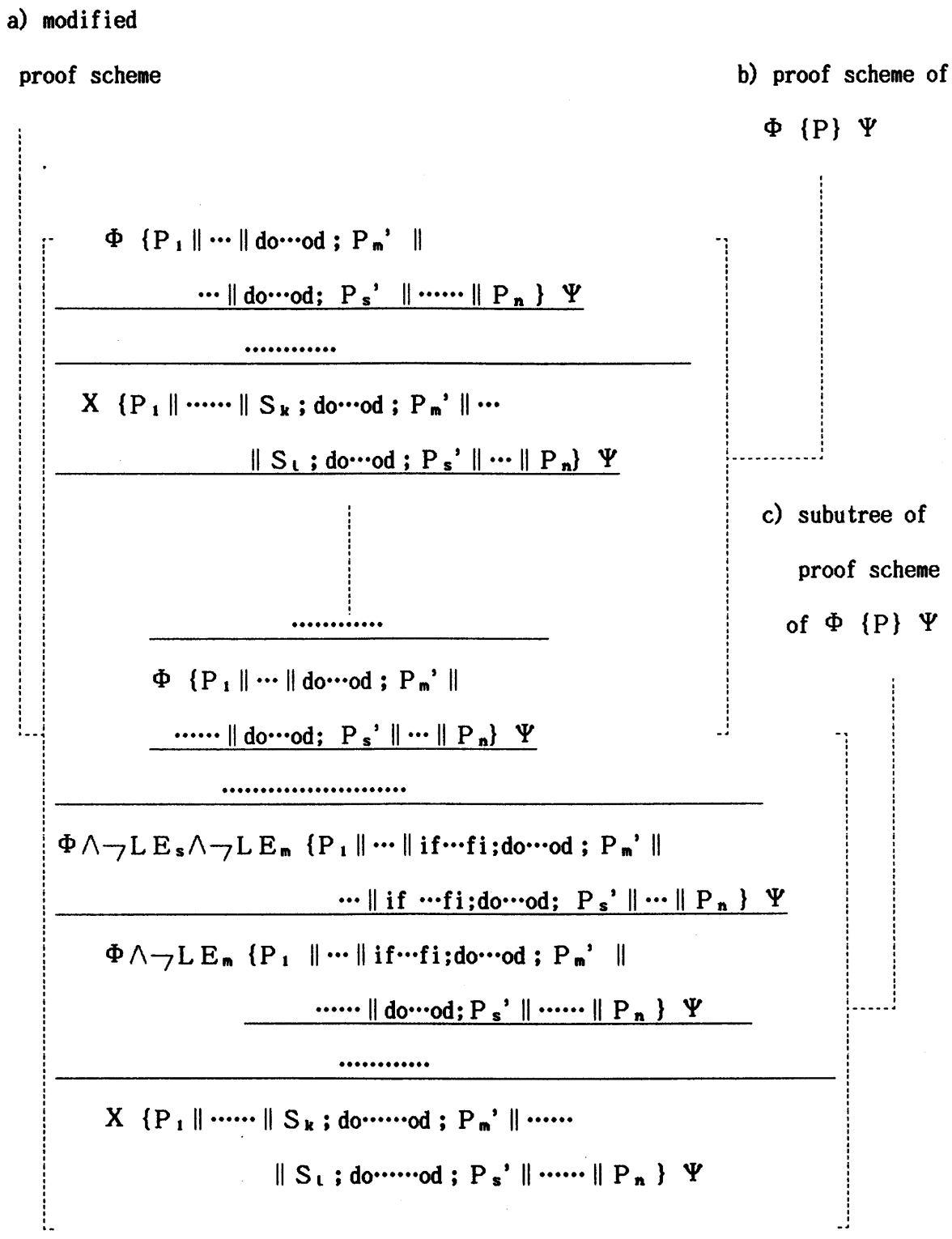


図　3．6　変形された証明図

〔例３．３〕

例３．２の証明において，空でないすべてのプロセスの先頭がif文となるのは，

$$\chi \wedge x > 1 \wedge true \ \{if\ \ x > 1 ; P_2!\, x \rightarrow x := x - 1\ \ fi; P_1' \parallel iftrue\ ; P_1?\, y \rightarrow z := z * y\ \ fi; P_2'\}\ \Psi$$

及び，

$$\chi \wedge x \leq 1 \wedge true \ \{\Lambda \parallel iftrue\ ; P_1?\, y \rightarrow z := z * y\ \ fi; P_2'\}\ \Psi$$

の２つの式である．

前者のブロッキングフリー条件は，

$$\chi \wedge x > 1 \wedge true \supset (x > 1 \wedge true)$$

となり，また後者のブロッキングフリー条件は，

$$\chi \wedge x \leq 1 \wedge false \supset false$$

となり，いずれも成立する．　（例終）

３．４　まとめ

本章ではＣＳＰのセマンティクスとして状態集合の上の二項関係，及び計算履歴の概念を定義し，それらをもとにＣＳＰの部分的正当性及びデッドロックフリー性を検証する論理体系を提案し，その妥当性を示した．ここで述べたセマンティクスでは停止性は扱われていない．すなわちここで定義したセマンティクスはいずれも停止した状態からさかのぼって定義されている．したがって実行時に停止しないパスを選択した場合については議論の対象になっていない．このような停止しないような実行をも含めたセマンティクス，ならびにそれに基づく検証方法については今後の課題である．

また公理系の完全性に関する議論も，今後の課題として残されている．

第４章　並列プログラムの検証と算術的階層の完備性

本章では，並列プログラムの検証体系について，その数学的性質を議論する．Aptらはプログラムの検証における述語のクラスについての重要な概念として，完備性(completeness)の概念を導入した(Apt Bergatra Meerteins 79)．直観的には，ある述語のクラス$\mathcal{A}$が完備であるとは，プログラムのクラスを$\mathcal{S}$とするとき，任意のP，$Q \in \mathcal{A}$，$s \in \mathcal{S}$についてP {s} Qが成立するならば，すべての中間表明を$\mathcal{A}$から選んで証明が可能であることをいう．(Apt Bergatra Meerteins 79) では$\mathcal{S}$としてwhile プログラムのクラスを選び，帰納的述語，帰納的可算述語，有限反証的述語等のクラスの完備性についていくつかの結果を示した．それによると，帰納的可算述語，有限反証的述語等のクラスは完備であるが両方の和をとると完備ではない．また帰納的述語については，そのままでは完備でなく，特別な変数を含めることを許すことによって完備となることがしめされた．

さらに(Apt 81)では$\mathcal{S}$として条件付臨界領域によって制御されるcobegin-coend 文を許す並列プログラムを選び，やはり帰納的述語のクラスの完備性について議論した．

以上の結果は，プログラムの性質のうち取扱いが可能であるものが部分的正当性に限られるHoare流の公理系，あるいはそれを拡張したOwiki(Owiki Gries 76)の公理系をもとにして得られたものである．一方，並列プログラムの正当性検証体系として第１章に述べたようにOwikiらの方法の他に，Flon, Suzuki (Flon Suzuki 81)の公理系，即ち並列プログラムを非決定性プログラムとして検証を行う体系が提案されている．この体系の特徴としては，第１章で述べたように，プログラムの制御に関する性質も含めて，様々な性質の議論が可能であることである．

一般に並列プログラムの制御に関する性質についての議論は，部分的正当性についての議論より困難であり，それらを扱う公理系も

複雑なものとなる．従ってそれによってなされる証明についての証明論的な議論はより重要なものとなると考えられる．

本章ではFlon, Suzuki の体系をもとに，完備性の概念を一般的なプログラムの性質の証明に拡張して議論する．Flon, Suzuki の体系の特徴として，次のような点があげられる．すなわち，そこで扱える性質は特定なものに固定されておらず，任意のプログラムｓの構文からつくられる連続ないしは単調な述語関数の最大ないしは最小不動点で表現される性質ならば，一般に取扱うことが可能である．また一般に並列プログラムの性質は，ある述語関数の不動点を用いてあらわすことができることが，(Emerson Clarke 80) によって知られている．

従って，プログラムの多様な性質の証明について考える際，個々の正当性についてそれぞれ公理系を個別に与え，それらを個々に議論する必要はなく，一般的な取扱いが可能である．

そこで本章では，不動点によって表現されたプログラムの性質の証明についての完備性を一般的に定義し，算術的階層の各クラスの完備性についていくつかの結果を示す．

４．１　被防護命令

本章では並列プログラムをDijkstra 流の被防護命令 (guarded comannd)(Dijkstra 76) のｄｏ文によって表現する．被防護命令は非決定性プログラムの一種で次のようなものである．

$$A_1 \ ; \underline{\text{do}}\ B_1 \rightarrow A_1 \ \square\ B_2 \rightarrow A_2 \ \square \cdots \square\ B_n \rightarrow A_n \ \underline{\text{od}}$$

ここで，各A_i (i=1, …,n) は作用(action)で，自然数の集合Ｎ上の代入文の並びである．また，B_i は防護(guard) と呼ばれ，Ｎ上の帰納的述語である．

A_1 は初期値の設定である．上のステートメントの実行は次のようにして行われる．すなわちB_i (i=1, …,n) 評価し，その結果が真であるもののうちから一つを非決定的に選び，対応するA_i を実行する．そして同様の動作を繰り返し，すべてのB_i が偽になったと

きに停止する．

cobegin-coend 文等の並列プログラムがいかにして被防護命令で表現されるかは文献(Emerson Clarke 80, Flon Suzuki 81) 等に述べられている．ここでは例を示すに留める．

次のようなプログラムを考える．

```
x:=0;
cobegin
repeat  produce;
        when true do x:=x+1 end  forever ‖
repeat  when x>0  do x:=x-1 end;
        consume   forever
coend
```

このプログラムは，被防護命令では次のように表される．

```
pc1:=pc2:=0; x:=0 ;
do
 pc1=0→produce; pc1:=(pc1+1)mod2    □
 pc1=1∧true→x:=x+1; pc1:=(pc1+1)mod2  □
 pc2=0∧x>0 →x:=x-1;  pc2:=(pc2+1)mod2   □
 pc2=1→consume ;  pc2:=(pc2+1)mod2
od
```

次にＣＳＰから被防護命令への変換例を示す．第３章で使用した例のプログラム $P_1 \| P_2$ は，被防護命令を用いて次のようにあらわせる．

```
pc1:= 0 ; pc2:= 0 ;  x := x0 ;  z := 1 ;
 do pc1= 0 ∧pc2= 0 ∧ x > 1 → y := x ; pc1:= 1 ; pc2:= 1   □
          pc1= 1   → x := x − 1 ; pc1:= 0   □
          pc2= 1   → z := z * y ; pc2:= 0   od
```

ここでは，第１章で述べた非決定性プログラムの実行を木で表す方法について形式的に扱うため，計算木(computation tree)を導入する．

〔定義４．１〕　状態ｉは各プログラム変数からＮ上の値への写像である．

〔定義４．２〕　プログラムｓと初期状態ｉに対して，計算木 $\mathcal{T}(s,i)$ を次のように定義する．初期状態とは初期値設定の結果，得られる状態である．

木の各節点は状態によってラベルづけられている．木の根のラベルは初期状態ｉである．各枝はｓの防護によってラベルづけられている．

各節点から出る枝は，その節点にラベルの状態で真となる防護によってラベルづけられており，その先に続く節点は各枝につけられた防護に対応する命令を実行した結果の状態でラベルづけられている．

$\mathcal{T}(s,i)$ の根から子孫に向かう各道は，ｓをｉから実行したときの実行系列に対応している．根から始まって，他の道の真のプレフィックスにならない道をフルパス(full path) と呼ぶ．

被防護命令プログラムの性質は，それを実行した時の計算木の性質として表すことができる．

４．２プログラムの性質と不動点理論

４．２．１　完備束上の関数と不動点

Ｎ上の述語全体からなる領域は，次のような半順序 $\sqsubseteq$ のもとで完備束となることが知られている．

$$P \sqsubseteq Q \quad \text{iff} \quad \pi(P) \subset \pi(Q)$$

ここで $\pi(P)$ は，述語Ｐの成立する状態の集合を表す．

記法　本章では以下のような記法を用いる．

$A \Rightarrow B$ …… ＡならばＢ，

$\bigvee_{k}^{n} B_k$ … $B_1 \vee B_2 \vee \cdots \vee B_n$ ，

$\bigwedge_{k}^{n} B_k$ … $B_1 \wedge B_2 \wedge \cdots \wedge B_n$ ，

$\exists x \leq_m p$ … $\exists x(p \wedge x \leq_m)$ ，

$\forall x \leq_m p$ … $\forall x(x \leq_m \rightarrow p)$ ，

$A^{-1}p$ …… 作用Aの実行後，述語pが成立するための最弱前条件．

$\pi(P)$ …… 述語Pの成立する状態の集合．

$\sqsubseteq$ …… 半順序．

$\sqcup S$ …… 半順序集合Sの最小上界．

$\sqcup_k a_k$ …… 鎖 $a_1 \sqsubseteq a_2 \sqsubseteq \cdots$ の上限．

$\sqcap S$ …… 半順序集合Sの最大下界．

$\sqcap_k a_k$ …… 鎖 $a_1 \sqsupseteq a_2 \sqsupseteq \cdots$ の下限．

〔定義４．３〕完備束をなす領域$\mathfrak{D}$について，

i）$\mathfrak{D}$上の関数fが単調(monotone)あるとは，すべてのX，Y$\in\mathfrak{D}$について

$$X \sqsubseteq Y$$

ならば，

$$f(X) \sqsubseteq f(Y)$$

となることである．

ii）$\mathfrak{D}$上の部分集合Aが有向集合であるとは，Aのすべての有限部分集合について，その上限および下限がAに含まれることである．

iii）$\mathfrak{D}$上の関数fが上から連続(continuous from above)であるとは，$\mathfrak{D}$の任意の有向集合Dについて

$$f(\sqcap D)=\sqcap \{f(X) \mid X \in D\}$$

となることである．

iv）$\mathfrak{D}$上の関数fが下から連続(continuous from below)であるとは，$\mathfrak{D}$の任意の有向集合Dについて

$$f(\sqcup D)=\sqcup \{f(X) \mid X \in D\}$$

となることである．

v）$\mathfrak{D}$上の関数fが連続であるとは，fが上かつ下から連続であることである．

vi）$\mathfrak{D}$上の関数fが上からω連続であるとは，任意の$\mathfrak{D}$上の鎖

$$d_1 \sqsubseteq d_2 \sqsubseteq \cdots$$

について

$$f(\sqcup\{d_i\})=\sqcup\{f(d_i) \mid i \geq 1\}$$

となることである．同様に下からω連続であるとは，任意の$\mathcal{D}$上の鎖

$$d_1 \sqsupseteq d_2 \sqsupseteq \cdots$$

について

$$f(\sqcap\{d_i\})=\sqcap\{f(d_i) \mid i \geq 1\}$$

となることである．

上からかつ下からω連続であるとき単にω連続であるという．

容易にわかるように連続ならばω連続であるが，ω連続であっても連続であるとは限らない．

〔定義４．４〕完備束をなす領域$\mathcal{D}$上の単調な関数ｆについて，

$$\{x \mid x \in \mathcal{D},\ x = f(x)\}$$

の最大元と最小元が常に存在する．最大限を最大不動点，最小限を最小不動点と呼び，それぞれ

$$gfp.Xf(X),$$
$$lfp.Xf(X)$$

で表す．以下ｆが１変数の場合，単にlfp.X ｆ，gfp.X ｆのように書く．

最大不動点および最小不動点は次のように特徴づけられる．

ｆが単調な場合，

$$lfp.Xf(X)=\sqcap\{X \mid f(X) \sqsubseteq X\}$$
$$gfp.Xf(X)=\sqcup\{X \mid X \sqsubseteq f(X)\}$$

また$\mathcal{D}$として述語全体からなる領域をとった場合，ｆがω連続ならば，

$$lfp.Xf(X)=\sqcup_k f^k(false)$$
$$gfp.Xf(X)=\sqcap_k f^k(true)$$

最大および最小不動点，単調あるいは連続な関数等に関する基本的な性質を次にまとめておく．いずれもよく知られた結果であるので証明は省略する．((Park 69)等参照)

１） ｆが連続であるならｆは単調である．

2） f(X,Y) がX およびY について下から(上から) 連続である場合，lfp.Xf(X,Y)(gfp.Xf(X,Y))も下から(上から) 連続である．

3） f(X,Y) がX およびY について下から(上から) 連続である場合，gfp.Xf(X,Y),lfp.Xf(X,Y) はいずれもY について単調である．

4） f(X,Y) がX およびY について単調である場合，gfp.Xf(X,Y)，lfp.Xf(X,Y) はいずれもY について単調である．

5） (不動点帰納法)

$$X \sqsubseteq f(X)$$

ならば

$$X \sqsubseteq gfp.Xf(X)$$

である．

述語の上の関数は，状態の集合の上の関数ともみなすことができる．即ち述語関数 f は述語 P をQに移すとすると，f は π(P) を π(Q) に移す状態集合上の関数である．以下述語関数とそれに対応する状態集合の上の関数を同じ記号を用いて表す．

$g_1(X)$，$g_2(X)$が連続な述語関数で，A^{-1}が帰納的関数の代入のみからなるとき，

$$f_1(X) = g_1(X) \wedge g_2(X)$$
$$f_2(X) = g_1(X) \wedge g_2(X)$$
$$f_3(X) = A^{-1}(g_1(X))$$

はいずれも連続である．

4．2．2 不動点論理式とその証明

被防護命令プログラムの性質は，前節で述べたように，その計算木の性質として表現することができる．しかし計算木は，プログラムを実行した後に得られるもので，実行前に木の性質を議論することは難しい．プログラムの性質を実行に先だって議論するには，その性質が例えばHoare流の記法のように，プログラムの構文を用いて表現されていることが望ましい．

本節では以下に，Emerson-Clarke の定理(Emerson Clarke 80)，およびFlon-Suzuki の定理(Flon Suzuki 80)についてその概要を

述べる．Emerson-Clarke の定理は被防護命令によって書かれたプログラムの性質を，N上の述語からなる領域上の関数（述語関数）の不動点を用いて表す方法について述べたものである．またFlon-Suzuki の定理は，単調な関数の最大不動点あるいは連続な関数の最小不動点で表現された性質の証明体系の作りかたについて述べたものである．

まずプログラムの性質を計算木についての性質として表現するための計算木式（computation tree formula:ＣＴＦ）を導入する．以下に示すのがＣＴＦの例とその意味である．

(1) ∀fullpath ∃nodeＲ．…ℸ(s,i) のすべてのフルパス上にＲが成立する状態でラベル付けされた節点が存在する．換言すれば，

「ｓをｉから実行したとき，いかなる道を選択してもいつかは必ずＲが成立する．」

となる．

(2) ∀fullpath ∀nodeＲ．…ℸ(s,i) のすべてのフルパス上のすべての節点でＲが成立する．すなわち

「ｓをｉから実行したとき，常にＲが成立し続ける．」

(3) ∃fullpath ∀∞nodeＲ．…ℸ(s,i) にあるフルパスが存在し，そのパス上の節点は有限個を除いて，すべてＲの成立する状態でラベル付けられている．すなわち

「ｓをｉから実行した時，ある時点からＲを常に成立し続けるようにすることが可能である．」

上のＲがさらにＣＴＦであってもかまわない．

上のようなＣＴＦで表現された性質を，ｓの構文を用いた表現にするために不動点論理式(fixpoint formula:ＦＰＦ）による記法を導入する．

〔定義４．５〕　次のような表現を不動点論理式という．

プログラムｓの述語R_1,…,R_k（=$\bar{R}$）についての正当性を表現するＦＰＦの構文はR_1,…，R_k，ｓの防護B_1，…，B_n，および変数X_1，X_2，…に

（ｉ）論理演算∧，∨，¬，⇒，…

（ii）ｓの各作用$A_1, A_2, \cdots A_n$の各最弱前条件演算

$$A_1^{-1}, A_2^{-1}, \cdots, A_n^{-1}$$

（iii）不動点演算子lfp,gfp

をほどこして得られるものである．

ＦＰＦの解釈はＮ上でその構文に従ってなされる．次の命題はＦＰＦがプログラムの性質について十分な表現力をもっていることを表す．

〔命題４．１〕（Emerson- Clarke)ＣＴＦで表現された任意の性質に対して，それと等価なＦＰＦに変換するアルゴリズムが存在する．又，∀∞，∃∞を含まないＣＴＦはω連続な関数のみからなるＦＰＦに変換される．

〔例４．１〕プログラム

ｓ：do$B_1 \rightarrow A_1 \square B_2 \rightarrow A_2 \square \cdots \square B_n \rightarrow A_n$ od

述語Ｒに対し，ＣＴＦ，

$$\forall \mathrm{fullpath} \forall \mathrm{node} R$$

は，

$$\Upsilon_R^n(X) = R \wedge \bigwedge_k^n (B_k \Rightarrow A_k^{-1} X)$$

とすると，

$$\mathrm{gfp}.X\ \Upsilon_R^n(X)$$

と等価になる．

また，次のＣＴＦ，

$$\exists \mathrm{fullpath} \forall \mathrm{node} R$$

は，「少なくともひとつ$\gamma(s,i)$にあるフルパスが存在し，そのパス上の節点は，すべてＲの成立する状態でラベル付けられている．」即ち，『ｓをｉから実行したとき，常にＲが成立し続けるような実行が可能である．』という意味になる．これは，

$$\Psi_R^n(X) = R \wedge \bigvee_k^n (B_k \Rightarrow A_k^{-1} X)$$

とすると，

$$\mathrm{gfp}.X\ \Psi_R^n(X)$$

と等価になる．　Υ_n，Ψ_nの最大不動点をそれぞれ

$$wip\forall\ (s,\ R)$$

$$wip\exists\ (s,\ R)$$

と書くことにする．以上の二つをプログラムｓの不変的性質と呼ぶ．

プログラムの性質を不動点によって特徴付けることの利点は，次に述べる命題に示されるように，その公理系が容易に得られることである．

〔命題４．２〕（Flon-Suzuki:最大不動点証明規則）

任意のプログラムｓと述語$\bar{R}$について，ｓの$\bar{R}$についての性質

$$pre(s,\ \bar{R})$$

がある単調な述語関数$\Phi_i(X)$の最大不動点gfp.X $\Phi_i(X)$と等価であり，かつ次の条件(1)，(2)を充たすとするとき，

(1)gfp.X $\Phi_i(X)$と等価な述語が中間表明として与えられる．

(2)述語Ｐ，Ｑに対して，$P \Rightarrow \Phi_i(Q)$という形の式を証明する公理系が存在する．

このとき，

$$\frac{P \Rightarrow \Phi_i(P)}{P \Rightarrow pre(s,\ \bar{R})} \quad \cdots F.\ I.$$

という規則を付け加えることによって，

$$P \Rightarrow pre(s,\ \bar{R})$$

に対する完全かつ無矛盾な公理系を得ることができる．

上の規則は不動点帰納法と同じ形をしている．

〔命題４．３〕（Flon-Suzuki:最小不動点証明規則）

任意のプログラムｓと述語$\bar{R}$について，ｓの$\bar{R}$についての性質

$$pre(s,\ \bar{R})$$

があるω連続な述語関数$\Phi_i(X)$の最小不動点lfp.X $\Phi_i(X)$と等価であり，かつ次の条件(1)，(2)を充たすとするとき，

(1)$\models \Phi_i^{\iota}(false) \equiv J(\iota)$となるＮ上の述語$J(\iota)$が中間表明として与えられる．

(2)述語Ｐ，Ｑに対して，$P \Rightarrow \Phi_i(Q)$という形の式を証明する公理

系が存在する．

このとき，

$$\frac{J(n+1) \Rightarrow \Phi_i(J(n)) \quad , \quad \neg J(0)}{\exists n\, J(n) \Rightarrow pre(s, \bar{R})}$$

という規則をつけ加えることによって，

$$P \Rightarrow pre(s, \bar{R})$$

に対する完全かつ無矛盾な公理系を得ることができる．

4．3　完備性

前節では，不動点で特徴付けられた性質の証明体系の一般的な形について述べているFlon-Suzuki の結果を紹介した．本節ではまずAptが導入したwhile プログラムの性質の部分的正当性についての完備性の定義を紹介する．続いてプログラムの一般的な性質の証明について，完備性の概念の拡張をおこなう．

本章の始めにも述べたように述語のクラス$\mathcal{A}$が性質

$$pre(s, \bar{R})$$

の証明について完備であるとは，

$$Q, \bar{R} \in \mathcal{A}$$

のとき，

$$Q \Rightarrow pre(s, \bar{R})$$

の証明が，$\mathcal{A}$からすべての中間表明を選ぶことによって可能であることをいう．

4．3．1　Hoare流の体系に基づく完備性

N上のプログラムのクラス$\mathcal{S}_w$として次のようなものを考える．

(1) N上の代入文は$\mathcal{S}_w$のプログラムである．

(2) $s_1, s_2 \in \mathcal{S}_w$であるとき，

$$s_1 ; s_2 \in \mathcal{S}_w ,$$

$$\underline{\text{while}}\ B\ \underline{\text{do}}\ s_1 \in \mathcal{S}_w$$

ただしBはN上の帰納的述語である．

$\mathcal{S}_w$の部分的正当性の証明についての完備性を次のように定義する．

〔定義４．６〕（Apt Bergatra Meerteins 79） 述語のクラス$\mathcal{A}$が$\mathcal{S}_w$の部分的正当性について完備であるとは，任意のs，s_1，$s_2 \in \mathcal{S}_w$，P，$Q \in \mathcal{A}$について次が成立することである．

(1) $\mathcal{A}$は，帰納的述語との論理積について閉じている．

(2) $\models P \{s_1 ; s_2\} Q$

ならば，ある$R \in \mathcal{A}$が存在して

$$\models P \{s_1\} R,$$
$$\models R \{s_2\} Q$$

である．

(3) $\models P \{\underline{\text{while}}\ B\ \underline{\text{do}}\ s_1\} Q$

ならば，ある$R \in \mathcal{A}$が存在して

$$\models P \Rightarrow R,$$
$$\models R \wedge B \{s\} R,$$
$$\models R \wedge \neg B \Rightarrow Q$$

である．

上の定義は部分的正当性についてのHoare流の体系の推論規則，

$$\frac{\vdash P \{s_1\} R \quad \vdash R \{s_2\} Q}{\vdash P \{s_1 ; s_2\} Q}$$

ならびに

$$\frac{\vdash P \Rightarrow R \quad \vdash R \wedge B \{s\} R \quad \vdash R \wedge \neg B \Rightarrow Q'}{\vdash P \{\underline{\text{while}}\ B\ \underline{\text{do}}\ s\} Q}$$

の結論に現れる述語が$\mathcal{A}$に入れば，前提に現れる述語も$\mathcal{A}$に含まれることを要求している．

このように，プログラムのある性質についての完備性とは，それを証明する公理系の形に依存して定義されることがわかる．

４．３．２ 完備性の一般化

上に述べたように，プログラムの様々な性質について完備性を定義するためには，個々の性質についてそれぞれ公理系を与え，それ

ぞれの公理系に従って完備性を定義しなければならない．しかし，前章の命題４．２，及び４．３より，被防護命令についての様々な性質について証明体系の一般的な形式が示されているので，それに基づいて完備性を一般的に定義することができる．

〔定義４．７〕述語のクラス$\mathcal{A}$が，ある単調な述語関数Φ_iの最大不動点と等価な性質

$$pre(s, \bar{R})$$

の証明について完備であるとは，任意のｓ，任意のＱ，$R_1, \cdots, R_k$（＝$\bar{R}$）$\in \mathcal{A}$に対して

$$\models Q \Rightarrow pre(s, \bar{R})$$

ならば，

$$\vdash Q \Rightarrow P,$$

$$\vdash P \Rightarrow \Phi_i(P)$$

となるＰ$\in \mathcal{A}$が存在することである．

同様にあるω連続な述語関数Φ_iの最小不動点と等価な性質

$$pre(s, \bar{R})$$

の証明について$\mathcal{A}$が完備であるとは，任意のｓ，任意のＱ，R_1，…，R_k（＝$\bar{R}$）$\in \mathcal{A}$に対して

$$\models Q \Rightarrow pre(s, \bar{R})$$

ならば，

$$\vdash Q \Rightarrow \exists l\, J(l),$$

$$\vdash J(l+1) \Rightarrow \Phi_i(J(l)),$$

$$\vdash \neg J(0)$$

である$\exists l\, J(l) \in \mathcal{A}$が存在することである．

上の定義は前に述べたwhile プログラムの場合と同様に，結論に現れた述語が$\mathcal{A}$に入れば前提に現れる述語も$\mathcal{A}$に含まれることを要求している．すなわちpre(ｓ，$\bar{R}$）が最大不動点の場合には，証明の形は，

$$\frac{Q \Rightarrow P \qquad \dfrac{P \Rightarrow \Phi_i(P)}{P \Rightarrow pre(s, \bar{R})} \quad \cdots\cdots \text{F．I})}{Q \Rightarrow pre(s, \bar{R})}$$

のようになり，中間表明としてPが出現する．

また最小不動点の場合には，証明の形は，

$$\dfrac{\dfrac{Q \Rightarrow J(\tau)}{Q \Rightarrow \exists \iota\, J(\iota)} \qquad \dfrac{J(\iota+1) \Rightarrow \Phi_i(J(\iota)),\ \neg J(0)}{\dfrac{J(\iota) \Rightarrow pre(s, \bar{R})}{\exists \iota\, J(\iota) \Rightarrow pre(s, \bar{R})}}}{Q \Rightarrow pre(s, \bar{R})}$$

のようになり，中間表明として，$\exists \iota\, J(\iota)$ が出現する．

4．4　算術的階層

N上の述語は，それが定義するN上の関係の複雑さによって階層構造をなすことが知られている(Hinman 78)．この階層構造は，各論理式に現れる束縛記号の入れ子の数によって特徴付けられる．

〔定義4．8〕　$r \geq 0$について，

(1) $\Sigma^0_0 = \Pi^0_0 =$ {帰納的に決定可能なN上の関係を定義する述語のクラス．}

(2) $\Sigma^0_{r+1} = \{ P(x_1, \cdots, x_n) \mid P(x_1, \cdots, x_n) = \exists y\ Q(x_1, \cdots, x_n, y),\ Q(x_1, \cdots, x_n, y) \in \Pi^0_r \}$．

(3) $\Pi^0_{r+1} = \{ P(x_1, \cdots, x_n) \mid P(x_1, \cdots, x_n) = \forall y\ Q(x_1, \cdots, x_n, y),\ Q(x_1, \cdots, x_n, y) \in \Sigma^0_r \}$．

(4) $\Delta^0_r = \Sigma^0_r \cap \Pi^0_r$．

Σ^0_1 は帰納的可算述語のクラスに，またΠ^0_1 は有限反証的（否定をとると帰納的可算となる）述語のクラスに一致することが知られている．

さらに次のことがいえる．

〔命題4．4〕すべての$r \geq 0$について，

(1)　$R \in \Sigma^0_r$ iff $\neg R \in \Pi^0_r$

かつ

$$R \in \Pi^0_r \text{ iff } \neg R \in \Sigma^0_r .$$

(2) $r \neq 0$ のとき，

$$\Delta^0_r \subsetneq \Sigma^0_r$$

かつ

$$\Delta^0_r \subsetneq \Pi^0_r .$$

(3) $$\Sigma^0_r \cup \Pi^0_r \subsetneq \Delta^0_{r+1} .$$

〔命題４．５〕算術的階層の各クラスは，すべての $r \geq 0$ で表４．１のような閉包性をもつ．

表４．１．算術的階層の閉包性

	Σ^0_r	Π^0_r	Δ^0_r
帰納関数の代入	○	○	○
$\wedge$ ，$\vee$	○	○	○
$\neg$	×	×	○
$\exists \leq$ ，$\forall \leq$	○	○	○
$\exists$	○($r \neq 0$)	×	×
$\forall$	×	○($r \neq 0$)	×

本節の内容はHinman(Hinman 78)によるものである．

4．5　一般化された完備性に関する結果

プログラムsが次のようなものであるとき，

$$\underline{\text{do}}\ B_1 \rightarrow A_1 \parallel B_2 \rightarrow A_2 \parallel \cdots \parallel B_n \rightarrow A_n\ \underline{\text{od}}$$

〔定義4．9〕述語関数Φ_iが正則(normal)であるとは次の条件を満たすことである．

(1) sのすべての作用A_t $(1 \leq t \leq n)$は帰納的関数の割当文だけからなる．

(2) sのすべての防護B_t $(1 \leq t \leq n)$はすべて帰納的述語である．

(3) Φ_iの変数X およびR$_j$ $(1 \leq j \leq k)$はすべて偶数回の否定であらわれる．

(4) Φ_iのなかに不動点演算子は含まれない．

〔補題4．1〕Φ_iが正則であるとき，$r > 0$について，

$$P \in \Sigma^0_r \quad (\Pi^0_r\ ,\ \Delta^0_r\ )$$

かつ，

$$R_j\ (1 \leq j \leq k) \in \Sigma^0_r \quad (\Pi^0_r\ ,\ \Delta^0_r\ )$$

ならば

$$\Phi_i\ (P) \in \Sigma^0_r \quad (\Pi^0_r\ ,\ \Delta^0_r\ )\ .$$

（証明）Σ^0_r $(\Pi^0_r\ ,\ \Delta^0_r)$は$\wedge$，$\vee$，偶数回の否定，帰納的関数の代入について閉じていることより明らか．

〔補題4．2〕Φ_iが正則ならばΦ_iは連続である．

（証明）$\Phi_i(X)=X$のときは明らか．連続関数は$\wedge$，$\vee$，偶数回の否定，帰納的関数の代入について閉じていることより，その他の場合も容易にしめせる．

〔定理4．1〕Φ_iが下記の条件をみたす正則な述語関数であるとき，Φ_iの最小不動点と等価な性質

$$\text{pre}\ (s,\ \bar{R})$$

の証明についてΣ^0_r $(r > 0)$は完備である．

条件：$\Phi_i^{\iota}(\text{false}) \equiv J(\iota)$

となる

$$J(\iota) \in \Sigma^0_r$$

が存在する．

（証明）$\models Q \Rightarrow pre(s, \bar{R})$

のとき，

Φ_iの連続性より，

$$lfp.X\ \Phi_i(X) \equiv \bigsqcup_\iota\ \Phi_i^\iota\ (false)\ .$$

$$\Phi_i^\iota\ (false) \equiv J(\iota)$$

より，

$$\models J(0) \equiv false,$$

ゆえに，

$$\vdash \neg J(0)\ ,$$

また，

$$\models J(\iota+1) \equiv \Phi_i(\ J(\ \iota))$$

より，

$$\vdash J(\iota+1) \Rightarrow \Phi_i(\ J(\ \iota))\ .$$

さらに，

$$\models Q \Rightarrow lfp.X\ \Phi_i(X)$$

より，

$$Q \sqsubseteq \bigsqcup_\iota\ \Phi_i^\iota\ (false)\ .$$

あるｍが存在して，

$$Q \sqsubseteq \Phi_i^m\ (false)\ \cdot$$

すなわち，

$$Q \sqsubseteq J(m).$$

ゆえに，

$$\vdash Q \Rightarrow \exists\iota\ J(\iota)$$

となる．

$$\exists\iota\ J(\iota) \in \Sigma^0_r$$

であることは，

$J(\iota) \in \Sigma^0_r$ および Σ^0_r が$\exists$について閉じていることより明らか．　（証明終）

〔定理４．２〕Φ_iが下記の条件をみたす正則な述語関数であるとき，

Φ_iの最大不動点と等価な性質

$$pre\ (s, R)$$

の証明についてΠ^0_r $(r > 0)$ は完備である.

条件： $\Phi_i^{\iota}\ (true) \equiv J(\iota)$

となる

$J(\iota) \in \Pi^0_r$

が存在する.

(証明) $\models Q \Rightarrow pre(s, R)$

のとき,

$$\vdash Q \Rightarrow P,$$

$$\vdash P \Rightarrow \Phi_i(P)$$

となるPを次のように与える.

$$P \equiv \bigsqcap_{\iota} \Phi_i^{\iota}\ (true)$$

Φ_iの連続性よりPはgfp.X $\Phi_i(X)$ と等しくなる. ゆえに

$$\vdash Q \Rightarrow P,$$

$$\vdash P \Rightarrow \Phi_i(P)$$

は明らか.

$$P \in \Pi^0_r$$

であることは次のようにしてしめされる. Pは, 次の鎖

$$true \sqsupseteq \Phi_i(true) \sqsupseteq \Phi_i^2(true) \sqsupseteq \Phi_i^3(true) \sqsupseteq \cdots$$

の下限である.

条件より

$$\Phi_i^{\iota}\ (true) \equiv J(\iota)$$

となる$J(\iota) \in \Pi^0_r$ が存在する. 次の鎖

$$J(0) \sqsupseteq J(1) \sqsupseteq J(2) \sqsupseteq \cdots$$

の下限は

$$\forall \iota\ J(\iota)$$

となり

$$P \equiv \forall \iota\ J(\iota).$$

Π^0_r は$\forall$について閉じているので

$$\forall \iota\ J(\iota) \in \Pi^0_r .$$

ゆえに

$$P \in \Pi^0_r .$$ （証明終）

〔定理４．３〕Φ_iが定理１と同様の条件を満たす正則な述語関数であるとき，Φ_iの最小不動点pre(s，$\bar{R}$）の証明について，

$$\models Q \Rightarrow pre(s, \bar{R}),$$

かつ，

$$Q, R \in \Pi^0_r$$

ならば，中間表明はΣ^0_{r+1}から選ぶことができる．

（証明）定理１の証明より$\exists \iota\ J(\iota)) \in \Sigma^0_{r+1}$を示せば十分であるが，これは定義より明らかである． （証明終）

最大不動点の証明についてΣ^0_rが完備であるか否かは一般には未解決である．しかし部分的な結果が次にしめされる．

〔補題４．３〕次のような関数の最小不動点を考える．

$$f^s_Q(X) = Q \vee X \vee \bigvee_t sp(X \wedge B_t, A_t)$$

ここで，

$$sp(X \wedge B_t, A_t)$$

は，$X \wedge B_t$が成立している時，A_tを実行した後の最強後条件を表すものとする．このとき，

$$lfp.X\, f^s_Q$$

を満たす状態の集合は，プログラムｓをＱが成立する状態から実行したときにできるすべての計算木の節点につけられたすべての状態の集合（これを$\mathcal{N}$（ｓ，Ｑ）で表す）と一致する．

（証明）まず，

$$\pi(lfp.X\, f^s_Q) \sqsubseteq \mathcal{N}(s, Q)$$

を示す．ここで

$$\pi(lfp.X\, f^s_Q)$$

はf^s_Qを状態集合の上の関数とみたとき，やはりその最小不動点になる．すなわち，

$$lfp.X\, f^s_Q = \bigsqcup_\iota f^{s\iota}_Q(false)$$

より，

$$\pi(\mathrm{lfp}.X\, f^{s}_{Q}) = \bigsqcup_{t} \pi(f^{s}_{Q}{}^{t}(\mathrm{false}))$$

となる．

したがって，$\mathcal{N}(s, Q)$ が f^{s}_{Q} の不動点であることをしめせばよい．

$$\mathcal{N}(s, Q) \sqsubseteq \pi(Q) \cup \mathcal{N}(s, Q) \cup \bigcup_{t} sp(\mathcal{N}(s, Q) \cap \pi(B_t), A_t)$$

より，

$$\mathcal{N}(s, Q) \sqsubseteq f^{s}_{Q}(\mathcal{N}(s, Q))$$

は明らか．

$$i \in f^{s}_{Q}(\mathcal{N}(s, Q))$$

については，次の3つの場合がある．

状態 i で Q が成立する場合 i は計算木の根である．ゆえに，任意の $i \in \pi(Q)$ について，

$$i \in \mathcal{N}(s, Q)$$

ゆえに，

$$\pi(Q) \sqsubseteq \mathcal{N}(s, Q)$$

次に状態 i が

$$i \in \mathcal{N}(s, Q)$$

の場合は，明らか．また，

$$i \in \bigcup_{i} sp(\mathcal{N}(s, Q) \cap \pi(B_i), A_i)$$

の場合，i はある計算木である状態 j の子供となっている．

したがって，やはり，

$$i \in \mathcal{N}(s, Q)$$

ゆえに，

$$f^{s}_{Q}(\mathcal{N}(s, Q)) \sqsubseteq \mathcal{N}(s, Q)$$

したがって $\mathcal{N}(s, Q)$ は f^{s}_{Q} の不動点である．

次に，

$$\mathcal{N}(s, Q) \sqsubseteq \pi(\mathrm{lfp}.X\, f^{s}_{Q})$$

を示す．これは，任意の

$$i \in \mathcal{N}\ (s,\ Q)$$

について，ｉがある計算木の深さｋ－１の節点であれば，

$$i \models f^{s}{}_{Q}{}^{k}\ (\mathrm{false})$$

となることを示せばよい．ｋに関する帰納法で示す．

ｋ＝１の場合：

ｉはある計算木の根である．ゆえに，

$$i \models Q$$

ここで，

$$f^{s}{}_{Q}\ (\mathrm{false}) = Q$$

より，

$$i \models f^{s}{}_{Q}\ (\mathrm{false})$$

ｋ＝ｍの場合：

ｊをある計算木の深さｍ－１の節点であるとすると，

$$j \models f^{s}{}_{Q}{}^{m}\ (\mathrm{false})$$

が成り立つとする．ここで，ｊの任意の子供ｉについてあるｔが存在し，

$$j \models B_{t}$$

かつｉはｊからA_tを実行した結果の状態である．従って，

$$i \models sp\ (f^{s}{}_{Q}{}^{m}\ (\mathrm{false}) \wedge B_{t},\ A_{t})$$

ゆえに，

$$i \models f^{s}{}_{Q}{}^{m+1}\ (\mathrm{false})$$

従って，任意の

$$i \in \mathcal{N}\ (s,\ Q)$$

について，

$$i \models \sqcup_{m}\ f^{s}{}_{Q}{}^{m}\ (\mathrm{false})$$

即ち，

$$i \in \pi(\mathrm{lfp.X}\ f^{s}{}_{Q})$$ （証明終）

〔定理４．４〕Ｒ及びＱが$\Sigma^{0}{}_{r}$の元であるとき，

$$f^{s}{}_{Q}{}^{l}\ (\mathrm{false}) \equiv F\ (l)$$

となるΣ^0_rの元F（ι）があるなら，wip∀（s，R）の証明について，Σ^0_rは完備である．

（証明）中間表明として，f^s_Qの最小不動点

$$lfp.X\,f^s_Q$$

を考える．このとき，

$$\Gamma^s_R(lfp.X\,f^s_Q) = R \wedge \bigwedge_t{}^n(B_t \Rightarrow A_t{}^{-1}lfp.X\,f^s_Q)$$

より

$$\vDash Q \Rightarrow wip\forall(s, R)$$

ならば，

$$\vdash Q \Rightarrow lfp.X\,f^s_Q$$

かつ，

$$\vdash lfp.X\,f^s_Q \Rightarrow R \wedge \bigwedge_t{}^n(B_t \Rightarrow A_t{}^{-1}lfp.X\,f^s_Q)$$

を示せばよいが，補題4．3より，

$$\pi(lfp.X\,f^s_Q) = \mathcal{N}(s, Q)$$

明らかに，

$$\vDash Q \Rightarrow lfp.X\,f^s_Q$$

また，

$$\vDash Q \Rightarrow wip\forall(s, R)$$

ならば，任意の

$$i \in \mathcal{N}(s, Q)$$

について，

$$i \vDash R$$

かつ，iは$\mathcal{N}$（s，Q）の元であるから，いかなるB_tについても，

$$i \vDash B_t$$

ならば，iからA_tを実行して到達する状態jについて，

$$j \in \mathcal{N}(s, Q)$$

ゆえに，

$$j \vDash lfp.X\,f^s_Q$$

すなわち，

$$i \models B_t \Rightarrow A_t^{-1} \mathrm{lfp}.X\, f^s_Q$$

ゆえに，

$$\mathcal{N}(s, Q) \subseteq \pi(R \wedge \bigwedge_t^n (B_t \Rightarrow A_t^{-1} \mathrm{lfp}.X\, f^s_Q)$$

即ち，

$$\models \mathrm{lfp}.X\, f^s_Q \Rightarrow R \wedge \bigwedge_t^n (B_t \Rightarrow A_t^{-1} \mathrm{lfp}.X\, f^s_Q)$$

よって，

$$\vdash \mathrm{lfp}.X\, f^s_Q \Rightarrow \Upsilon^s_R (\mathrm{lfp}.X\, f^s_Q)$$

が示される．また，

$$\mathrm{lfp}.X\, f^s_Q \equiv \exists l\, F(l)$$

より，

$$\mathrm{lfp}.X\, f^s_Q \in \Sigma^0_r$$

となることが，定理４．１の場合と同様に示される．　（証明終）

〔補題４．４〕関数g^s_Rを次のように定める．

$$g^s_R(X) = (R \wedge \bigwedge_t^n \neg B_t) \vee X \vee (R \wedge \bigvee_t^n (B_t \wedge A_t^{-1} X))$$

g^s_Rの最小不動点は，ある状態からプログラムｓを実行したときにたどる，Ｒが成立しつづける有限長のフルパス（これをｓのＲパスと呼ぶ）上の状態の集合を特徴づける．

（証明）ｓのあらゆる計算木のＲパス上の状態の集合を

$$M(s, R)$$

であらわす．補題４．３のときと同様にまず

$$\pi(\mathrm{lfp}.X\ g^s_R) \sqsubseteq M(s, R)$$

であることを，$M(s, R)$がg^s_Rの不動点であることを示すことによって証明する．まず，

$$M(s, R) \subseteq (\pi(R) \cap \bigcap_t \pi(\neg B_t)) \cup M(s, R) \cup (\pi(R) \cap \bigcup_t (\pi(B_t) \cap \pi(A_t^{-1} M(s, R))))$$

より，

$$M(s, R) \sqsubseteq g^s_R(M(s, R))$$

は明らか．一方，

$$g^s_R(M(s, R)) \sqsubseteq M(s, R)$$

については，次の３つの場合がある．

$$i \in (\pi(R) \cap \bigcap_t \pi(\neg B_t))$$

の場合，ｉはｉ自身を根とする計算木の長さ０のＲパス上の節点である．次に，

$$i \in M(s, R)$$

この場合は明らかにｉはＲパスの上にある．最後に，

$$i \in \pi(R) \cap \bigcup_t (\pi(B_t) \cap \pi(A_t^{-1} M(s, R)))$$

この場合，

$$i \in \bigcup_t (\pi(B_t) \cap \pi(A_t^{-1} M(s, R)))$$

より，１ステップ実行することにより，Ｒパス上の節点の至ることができる．また，

$$i \models R$$

より，ｉもＲパスの上にある．従って，Ｍ（ｓ，Ｒ）は，g^{s}_{R}の不動点である．

一方，

$$M(s, R) \sqsubseteq \pi(\mathrm{lfp}.X\ g^{s}_{R})$$

であることは，次のようにして証明される．

状態ｉがＲパスの上にあるなら，そこから計算木の葉（即ち，$\bigwedge_t{}^{n} \neg B_t$が成り立つ状態）に至までの最小のステップ数をｌとすると，

$$i \models g^{s}_{R}{}^{l+1}(\mathrm{false})$$

となることをｌに関する帰納法で示す．

ｌ＝０のとき：

この場合ｉは葉であり，

$$i \models \bigwedge_t{}^{n} \neg B_t$$

すなわち，

$$i \models g^{s}_{R}(\mathrm{false})$$

ｌ＝ｋのとき，

$$i \models g^{s}_{R}{}^{k+1}(\mathrm{false})$$

であるとする．ある状態ｉで

$$i \models R$$

かつ，葉までのステップ数がｋ＋１であるとする．ここでいかなるB_tも成立しなければ，ｉは葉であり既に解決している．従って，あるB_tが存在し，

$$i \models B_t$$

そこで，A_tを実行して状態ｊに至る．ここで帰納法の仮定より，

$$j \models g^{s}{}_{R}{}^{k+1}(\mathrm{false})$$

したがって，

$$i \models B_t \wedge A_t{}^{-1} g^{s}{}_{R}{}^{k+1}(\mathrm{false})$$

ゆえに，

$$i \models (R \wedge \bigwedge_t{}^{n} \neg B_t) \vee g^{s}{}_{R}{}^{k+1}(\mathrm{false}) \vee (R \wedge \bigvee_t{}^{n} (B_t \wedge A_t{}^{-1} g^{s}{}_{R}{}^{k+1}(\mathrm{false})))$$

すなわち，

$$i \models g^{s}{}_{R}{}^{k+2}(\mathrm{false}) \qquad \text{（証明終）}$$

〔定理４．５〕Ｒのすべての要素及びＱが$\Sigma^{0}{}_{r}$の元であるとき，

$$g^{s}{}_{R}{}^{l}(\mathrm{false}) \equiv G(l)$$

となる$\Sigma^{0}{}_{r}$の元Ｇ（ｌ）があり，かつ，入力条件Ｑについて，ｓが停止するなら，wip∃（ｓ，Ｒ）の証明について$\Sigma^{0}{}_{r}$は完備である．

（証明）中間表明として，

$$\mathrm{lfp.X}\, g^{s}{}_{R} \wedge \mathrm{lfp.X}\, f^{s}{}_{Q}$$

を用いる．

$$\models Q \Rightarrow \mathrm{wip}\,\exists(s, R)$$

ならば，

$$\models Q \Rightarrow \mathrm{lfp.X}\, g^{s}{}_{R} \wedge \mathrm{lfp.X}\, f^{s}{}_{Q}$$

かつ，

$$\models \mathrm{lfp.X}\, g^{s}{}_{R} \wedge \mathrm{lfp.X}\, f^{s}{}_{Q} \Rightarrow \Psi^{s}_{R}(\mathrm{lfp.X}\, g^{s}{}_{R} \wedge \mathrm{lfp.X}\, f^{s}{}_{Q})$$

を示す．任意の，

$$i \in \pi(Q)$$

について，補題４．３より，

$$i \in \pi(\mathrm{lfp}.X\, f^{s}{}_{Q})$$

ゆえに，

$$\models Q \Rightarrow \mathrm{lfp}.X\, f^{s}{}_{Q}$$

また，入力条件Ｑについて，ｓが停止することより，ｉを根とする計算木のパスはすべて有限長である．また，

$$\models Q \Rightarrow \mathrm{wip}\ \exists\ (s,\ R)$$

より，ｉを根とする計算木のパスの上では常にＲが成り立つ．従って，ｉはＲパス上の節点である．即ち，

$$i \in M\ (s,\ R)$$

補題４．４より，

$$i \models \mathrm{lfp}.X\, g^{s}{}_{R}$$

以上より，

$$\vdash Q \Rightarrow \mathrm{lfp}.X\, g^{s}{}_{R} \wedge \mathrm{lfp}.X\, f^{s}{}_{Q}$$

となる．また，

$$\models \mathrm{lfp}.X\, g^{s}{}_{R} \wedge \mathrm{lfp}.X\, f^{s}{}_{Q} \Rightarrow$$
$$R \wedge \bigvee_{t}{}^{n}(\ B_{t} \Rightarrow A_{t}{}^{-1}\ (\mathrm{lfp}.X\, g^{s}{}_{R} \wedge \mathrm{lfp}.X\, f^{s}{}_{Q})\ )$$

について，任意の

$$j \in \pi(\mathrm{lfp}.X\, g^{s}{}_{R} \wedge \mathrm{lfp}.X\, f^{s}{}_{Q})$$

について，補題４．４よりｊはＲパスの上にある．ゆえに，

$$j \models R$$

また，いかなるB_{t}についても，

$$j \not\models B_{t}$$

ならば，明らかに

$$j \models \bigvee_{t}{}^{n}(\ B_{t} \Rightarrow A_{t}{}^{-1}\ (\mathrm{lfp}.X\, g^{s}{}_{R} \wedge \mathrm{lfp}.X\, f^{s}{}_{Q})\ )$$

そうでなければ，補題４．４より，ｊはＲパスの上の状態であることより，あるB_{t}が存在して，

$$j \models B_{t}$$

かつ，$A_{t}{}^{-1}$を実行した後に到達する状態ｊ’について，

$$j' \in M\ (s,\ R)$$

さらに，補題４．３より

$$j \in \mathcal{N}\ (s,\ Q)$$

より，

$$j' \in \mathcal{N}\ (s,\ Q)$$

再び，補題４．３，４．４より，

$$j' \vDash \mathrm{lfp.X}\, g^{s}{}_{R} \wedge \mathrm{lfp.X}\, f^{s}{}_{Q}$$

従って，

$$j \vDash \bigvee_{t}{}^{n}(\ B_t \Rightarrow A_t{}^{-1}\ (\mathrm{lfp.X}\, g^{s}{}_{R} \wedge \mathrm{lfp.X}\, f^{s}{}_{Q})\ )$$

即ち，

$$\pi(\mathrm{lfp.X}\, g^{s}{}_{R} \wedge \mathrm{lfp.X}\, f^{s}{}_{Q}) \subseteq$$
$$\pi(R \wedge \bigvee_{t}{}^{n}(\ B_i \Rightarrow A_i{}^{-1}\ (\mathrm{lfp.X}\, g^{s}{}_{R} \wedge \mathrm{lfp.X}\, f^{s}{}_{Q})\ )$$

これより，

$$\vdash \mathrm{lfp.X}\, g^{s}{}_{R} \wedge \mathrm{lfp.X}\, f^{s}{}_{Q} \Rightarrow$$
$$R \wedge \bigvee_{t}{}^{n}(\ B_i \Rightarrow A_i{}^{-1}\ (\mathrm{lfp.X}\, g^{s}{}_{R} \wedge \mathrm{lfp.X}\, f^{s}{}_{Q})\ )$$

が示せる．

次に，

$$\mathrm{lfp.X}\, g^{s}{}_{R} \wedge \mathrm{lfp.X}\, f^{s}{}_{Q} \in \Sigma^{0}{}_{r}$$

について，

$$\mathrm{lfp.X}\, f^{s}{}_{Q} \in \Sigma^{0}{}_{r}$$

は定理４．４で既に解決している．

$$\mathrm{lfp.X}\, g^{s}{}_{R} \in \Sigma^{0}{}_{r}$$

についてはまた，

$$\mathrm{lfp.X}\, g^{s}{}_{R} \equiv \exists l\ G\ (l)$$

より，

$$\mathrm{lfp.X}\, g^{s}{}_{R} \in \Sigma^{0}{}_{r}$$

となることが，定理４．１の場合と同様に示される．$\Sigma^{0}{}_{r}$ は有限個の論理積について閉じているので（命題４．５），

$$\mathrm{lfp.X}\, g^{s}{}_{R} \wedge \mathrm{lfp.X}\, f^{s}{}_{Q} \in \Sigma^{0}{}_{r}$$

となる．　（証明終）

４．６　まとめ

本章では，Aptの完備性を，プログラムのより一般的な性質の証明に拡張し，算術的階層の各クラスについて，その完備性を議論した．

本章で扱ったのは，主に連続な関数の不動点であるが，それらは先に述べたように，次のような性質を含む．即ち，

「プログラムの実行中Ｒが成立し続ける．」

及び，

「プログラムの実行中にいつかはＲが成立する．」

である．これらは，様相論理における□Ｒ，◇Ｒに相当し，プログラムの有限時間内の動作を記述するための基本的な要素となっている．

本章で示した結果は，解決すべき問題のごく一部であり，最小不動点で示される性質について，Π^0_r が完備であるか，また最小不動点，最大不動点両方について，Δ^0_r はどうか，（最大不動点についてはΔ^0_0 が完備でないことは（Apt 81）の結果より明らか）などついては，更に研究を要する．

第５章　あとがき

本論文では，並列プログラムの形式的意味記述法として，公理的アプローチによる検証体系を，共有資源型のモデルと通信型のモデルについて与えた．本論文で提案した検証体系が対象とする性質はプログラムの部分的正当性，デッドロックフリー性などである．これらの性質はいずれも，第４章で述べたような表現を使うと，ある連続な関数の最大不動点で与えられるようなものである．それ以外の性質としては，例えばプログラムの停止性等があるが，このような性質についての検証体系を，本論文で与えた体系と同様なもの，即ち，ＲＧＣに対してはダイナミックロジックの拡張として，またＣＳＰに対しては，centralized approechに沿って検証を進めるような形で与えることは，今後の課題として残されている．

本論文では，並列プログラムの公理的意味論を，主にユーザのプログラムに近い形のものを対象として与えてきた．これは，まえがきにも述べたように，公理的意味論は比較的ユーザに近い局面で，その応用を考えるべきであるという立場からである．すなわち，今後このような研究をさらに進めるにあたって，次のような方向が考えられる．

公理的意味論以外の方法として提案され，研究が行われている様々なプログラムの意味記述法の幾つか，例えば抽象データ型の代数的意味記述法などについて，従来報告されている成果から，これらの意味記述法は，言語設計や言語処理系の実現のための工学的手法としては有望であるといえる．しかし，先にものべたように実際のあるプログラミング言語によって書かれたプログラムの，働きを，その言語の設計，実現に用いられた意味論から理解しようとするのは現実的ではない．一方，公理的意味論が言語設計などの手法の基礎として有力であるという報告はあまりなされてはいない．（僅かに，コンパイラ生成系ＰＱＣＣのコード生成部においてHoare流の記述の利用が試みられた）このように，現在までの研究の方向をみ

る限り，言語の設計，実現に有望視されている意味論と公理的意味論は，その応用分野を分担するような傾向がうかがえる．したがって，これらの意味論がそれぞれの分野で工学的手法として確立されるためには，公理的意味論と，言語の開発のための意味論との関係が明確になっていることが望ましい．それは，単に記述能力の比較といった理論的な考察に留まらず，例えばある種のセマンティクスーコンパイラ，即ち，言語開発のための意味記述から言語のユーザのための意味記述への変換系の開発につながるような方向を目指すべきであろう．

また，さらに進めて，プログラムの証明を，そのユーザに対するドキュメントとして利用できるような形で記述する方法に関する研究，及び，ドキュメントとしての証明を，実現されたプログラムあるいはそのプログラムを記述している言語の仕様から組織的に作る方法に関する研究などが考えられる．

もうひとつの方向として，前書にも述べたように，近年計算機システムの並列化がすすめられているが，並列性を含むプログラムに対する意味記述法としては，公理的意味論，ＣＳＰの表示的意味論以外には，多くは報告されてはいない．特に，並列計算のモデルは数多く提案されているにもかかわらず，並列プログラムに意味を定義する手法としては，逐次型，あるい非決定性プログラムの意味記述法を応用，拡張したものが多く，並列プログラムの意味記述のために新たに考案された手法はあまりみられない．

詳細な考察はここでは行わないが，近年論理型プログラミング，関係データベース，さらに広い意味ではオブジェクト指向プログラミング等もそうであるように，論理学的手法をとりいれた並列プログラミングの形式が注目されている．これらの事情を考慮すると，今後，当初から並列プログラムの意味記述を目的とした新たな意味論が考案されるとしたら，数理論理学的手法に基づくものである可能性は大きいと思われる．そのように考えると，新しい意味論が公理的意味論の研究から生ずる可能性が十分にあるといえる．

謝　辞

本論文は，著者が名古屋大学大学院工学研究科博士課程に在学中に行った研究をまとたものである．本研究を進めるにあたり，懇切な御指導を賜った名古屋大学稲垣康善教授，福村晃夫教授に心より感謝致します．また豊橋技術科学大学本多波雄学長，名古屋大学杉江昇教授に感謝致します．

またオートマトングループの阿曽弘具助教授及び三重大学坂部俊樹助教授，名古屋大学杉野花津江助手，広島大学山下雅史助教授には日頃熱心に御討論頂いたこと，名古屋大学吉田雄二助教授には日頃の御指導についてもここに感謝の言葉を表します．

また著者が日頃円滑に研究を進めることができたのは，伊藤慶子さんの働きによるところが大きい．謝辞を送ります．

本論文の原稿はワードプロセッサＯＡＳＹＳ－１００Ｇ及び１００ＧⅡを用いて清書された．また英文の参考文献の部分には，ＰＤＰ１１，バークレイ版ｕｎｉｘ２．９ｂｓｄのｎｒｏｆｆを用いた．

参考文献

Abrahamson,K.,"Modal Logic of Conncurrent Nondeterministic Programs" Lecture Notes in Computer Science no.70 (1979)

Adamsky,S.,"On Semantic Foundations for Applicative Multiprogramming" Proc. of ICALP 83 (1983)

Alagic,S.and M.A.Arbib,"The Design of Well-Structured and Correct Programs" Springer-Verlag(1978)

Andrews,G.R.,"Concepts and Notations for Concurrent Programming" ACM Computing Surveys,vol.15,no.1(1983)

Andrews,G.R.,"Parallel Programs: Proofs,Principles and Practice" CACM vol.24,no.3 (1981)

Apt,K.R., J.A.Bergstra and L.G.L.Meerteins,"Recursive Assertions Are Not Enough - Or Are They?" Theoretical Computer Science vol.8, (1979)

Apt,K.R. and C.Delporte,"An Axiomatization of the Intermittent Assertion Method Using Temporal Logic" Proc. of ICALP 83 (1983)

Apt,K.R.,"Formal Justification of a Proof System for Communicating Sequential Processes" JACM,vol.30,no.1(1983)

Apt,K.R.,"Recursive Assertions and Parallel Programs" Acta Informatica vol.15(1981)

Apt,K.R.,N.Francez and W.P.de Roever,"A Proof System for Communicating Sequential Processes" ACM Trans. on Programming Languages and Systems,vol.2,no.3,(1980)

Basu,S.K. and R.T.Yhe,"Strong Verification of Programs" IEEE Trans. on Software Engineering,SE-1,no.1,(1975)

Bell,D.H. and D.Simpson"Think Parallel" Comp Bul Series 2,28 (1981)

Bell,D.H.,J.M.Kerridge,D.Simpson and N.Willis,"Parallel Programming-A Bibliography" The British Computer Society,(1983)

Benson,D.B.,"In Scott-Strachey Style Denotational Semantics,Parallelism Implies Nondeterminism" Mathematical System Theory,vol.15,(1982)

Bergstra,J.A. and Tucker,J.V.,"Expressiveness and Completeness of Hoare's Logic" Journal of Computer and System Sciences, vol.25 (1982)

Boyer,R.S. and J.S.Moore,"A Verification Condition Generator for FORTRAN" The Correctness Problem in Computer Science, edited by R.S.Boyer and J.S.Moor, Academic Press, (1981)

Brinch Hansen,P."Operating System Principles" Prentice-Hall,(1973)

Brinch Hansen,P.,"Distributed Processes:A Concurrent Programming Concept" CACM vol.21,no.11 (1978)

Brinch Hansen,P.,"The Architecture of Concurrent Programs" Prentice-Hall (1977)

Brookes,S.D."A Model for Communicating Sequential Processes" D.Phil.Dissertation,Oxfard Univ.,Oxford,England(1983)

Brookes,S.D."A Semantic and Proof System for Communicating Processes" Research Report,Carnegie-Mellon University,CMU-CS-83-134(1983)

Brookes,S.D.,C.A.R.Hoare and A.W.Roscoe,"A Theory of Communicating Sequential Processes" JACM,vol.31,no.3(1984)

Buckley,G.N. and A.Silberschatz,"An Effective Implementation for The Generalized Input-Output Construct of CSP" ACM Trans. on Programming Language and Systems,vol.5,no.2,(1983)

Burstall,R.M. and J.A.Goguen,"An Informal Introduction to Specification Using Clear" The Correctness Problem in Computer Science, edited by R.S.Boyer and J.S.Moore, Academic Press, (1981)

Campbell,R.H. and A.N.Harbermann,"The Specification of Process Synchronization by Path Expressions" Lecture Notes in Computer Science no.16 (1973)

Chandy,K.M.,and J.Misra,"The Drinking Philosophers Problem" ACM Trans. on Programming Language and Systems,vol.6,no.4(1984)

Clarke,E.A.,E.A.Emerson and A.P.Sistla,"Automatic Verification of Finite State Concurrent Systems Using Temporal Logics Specifications:A Practical Approach" Proc.of ACM POPL 83,Research Report of Carnegie-Mellon University CMU-CS-83-152 (1983)

Clarke,E.M."Programming Language Constructs of Which It Is Impossible to Obtain Good Hoare Axiom Systems" JACM,vol.26,no.1 (1979)

Clarke,E.M.,"Proving Correctness of Coroutines without History Variables" Acta Informatica vol.13,(1980)

Clarke,E.M.,"Synthesis of Resource Invariants for Concurrent Programs" ACM Trans. on Programming Lounguages and Systems,vol.2,no.3,(1980)

Cook,S.A.,"Soundness and Completeness of an Axiom System for Program Verification" SIAM.J.Computing,vol.7,no.1 (1978)

De Bakker,J.W. and J.I. Zuker,"Denotational Semantics of Concurrency" Proc.of ACM STOC (1982)

De Bekker,J.W. and L.G.L.T.Meertens,"On the Completeness of the Inductive Assertion Method" J.C.S.S.,vol.11 (1975)

De Millo,R.A.,R.J.Lipton and A.J.Perilis,"Social Processes and Proofs of Theorems and Programs" CACM,vol.22,no.5,(1979)

De Nicola,R.,"A Complete Set of Axioms for A Theory of Communicating Sequential Processes" Proc.of The 1983 Conference on The Foundation of Computing Theory,Lecture Notes in Computer Science,no.158(1983)

Dijkstra,E.W.,"A Discipline of Programming" Prentice-Hall,(1976)

Dijkstra,E.W.,"Guarded Commands,Nondeterminacy and Formal Derivation of Programs" CACM vol.18,no.8 (1975)

Dijkstra,E.W.,"Introduction: Why Correctness Must Be a Mathematical Concern" The Correctness Problem in Computer Science, edited by R.S.Boyer and J.S.Moore, Academic Press,(1981)

Elrad,T. and N.Francez"A Weakest Precondition Semantics for Communicating Processes" Proc.of International Symp.on Programming 5th Colloquium,Lecture Notes in Computer Science,no.137(1982)

Elrad,T. and N.Francez,"A Weakest Precondition Semantics for Communicating Processes" IBM Research Report,RC9773,(1982)

Emerson,E.A. and E.M.Clarke,"Characterizing Correctness Properties of Parallel Programs Using Fixpoints" Proc.of ICALP 80 (1980)

Flon,L. and N.Suzuki,"The Total Correctness of Parallel Programs" SIAM.J.on Computing,vol.10,no.2 (1981)

Floyd,R.W.,"Assigning Meanings to Programs" Proc. Symp. Mathematical Aspects of Computer Science,(1967)

Francez,N, D.Lehmann and A Pnueli,"A Linear History Semantics for Distributed Language:Extended Abstract" Proc.of

FOCS,ACM(1980)

Francez,N.,"Distributed Termination" ACM Trans. on Programming Language and Systems,vol.2,no.1(1980)

Francez,N.,C.A.R.Hoare,D.J.Lehmann and W.P.de Roever, "Semantics of Nondeterminism,Concurrency,and Communication" JCSS vol.19(1979)

Gamatie,B.,"Systemes de Prosessus Communication Parallele de Languages Functionnels" Rapports de Recherche, no.320(1984)

Gergely,T. and L.Ury,"A Theory of Interactive Programming" Acta Informatica,vol.17(1982)

Gerth,R.,W.P.De Roever,"A Proof System for Concurrent ADA Programs" Science of Computer Programming,vol.4,no.2(1984)

Gourlay,J.S.,W.C.Rounds and R.Statman,"On Properties Preserved by Contractions of Conncurrent Programs" Lecture Notes in Computer Science no.70 (1979)

Harel,D., A.R.Meyer and V.R.Pratt,"Computability and Completeness in Logic of Programs:Preliminary Report" Proc.of the 9th ACM STOC, Revised version MIT.Lab.for Computer Science TM97,(1978)

Harel,D.,"First-Order Dynamic Logic" Lecture Notes in Computer Science,no.68,(1979)

Harel,D.,"On the Total Correctness of Nondeterministic Programs" Theoretical Computer Science,vol.13(1981)

Harel,D.,"Proving the Correctness of Reguler Deterministic Programs:A Unifying Survey Using Dynamic Logic" Theoretical Computer Science,vol.19,(1980)

Harel,D.,"Two Results on Process Logic" Information Processing Letters,vol.8,no.4 (1979)

Harel,D.,"Two Results on Process Logic" Information Processing Letters,vol.8,no.4,30 (1979)

Harel,D.,A.Pnueli and J.Stavi,"A Complete Axiomatic System for Proving Deductions about Recursive Programs" Proc.of the 9th ACM STOC (1977)

Harel,D.,A.Pnueli and J.Stavi,"Completeness Issues for Inductive Assertion and Hoare's Method" Technical Report,Dept.of Mathematical Sciences,Tel-Aviv Univ.Israel (1976)

Harel,D.,D.Kozen and R.Parikh,"Process Logic Expressiveness, Decidability,Completeness" Proc.of 21th IEEE Symp.on FOCS(1980)

Hinman,P.G.,"Recursion-Theoretic Hierarchies" Springer-Verlag, Berlin (1978)

Hoare,C.A.R.,"A Calculus of Total Correctness for Communicating Processes" Science of Computer Programming 1(1981)

Hoare,C.A.R.,"An Axiomatic Basis for Computer Programming" CACM,vol.12,no.10,(1969)

Hoare,C.A.R.,"Communicating Sequential Processes" CACM,vol.21,no.8(1978)

Howard,J.,"Proving Monitors" CACM,vol.19,no.5(1976)

INMOS Limited,"Occam Programming Manual" Prentice Hall,(1984)

Jorrand,P.,"Specification of Communicating Processes and Process Implementation Correctness" Proc.of International Symp. on Programming,5th Colloquium,Lecture Notes in Computer Science,no.137(1982)

Karp,R.A.,"Proving Failure-Free Properties of Concurrent

Systems Using Temporal Logic" ACM Trans.on Programming Languages and Systems,vol.6,no.2,(1984)

Keller R.M."Formal Verification of Parallel Programs" CACM,vol.19,no.7,(1976)

Lamport,.L. and R.Shostak and M.Pease,"The Bysantine General Problem" ACM Trans. on Programming Language and Systems ,to appear

Lamport,L. and F.B.Schneider,"The " Hoare Logic" of CSP, and All That" ACM Trans. on Programming Languages and Systems,vol.6,no.2(1984)

Lamport,L.,"Prooving The Correctness of Multiprocess Programs" IEEE Trans.Software Eng.SE3,2(1977)

Lamport,L.,"Specifying Concurrent Program Modules" ACM Trans. on Programming Languages and Systems,vol.5,no.2,(1983)

Lamport,L.,"The "Hoare Logic" of Concurrent Programs" Acta Informatica,vol.14,(1980)

Lamportt,L.,""Sometimes"is sometimes"Not Never":On the Temporal Logics of Programs" Proc. of the 7th ACM Symp. on POPL,(1980)

Lampson,B.W.,"Report on The Programming Language: Euclid" SIGPLAN Notice,ACM, vol.12,no.2 (1977)

Levin,G.M.,and D.Gries,"A Proof Technique for Communicating Sequential Processes" Acta Informatica,vol.15,(1981)

Lipton,R.J.,"A Necessary and Sufficient Condition for the Existence of Hoare Logics" Proc. of the 18th IEEE Symp. on FOCS(1977)

Liskov,B.,A.Snyder,R.Atkinson and C.Schaffert,"Abstraction Mechanisms in CLU" CACM vol.20,no.8 (1977)

Lucas,P.,"Formal Semantics of Programming Languages:VDL" IBM Journal of Res.Develop.,vol.25,no.5,(1981)

Majster-Cederbaum,M.E.,"Semantics :Algebras,Fixed Points, Axioms" Proc.of ICALP 80 (1980)

Makowsky,J.A.,"Measuring the Expressive Power of Dynamic Logics: An Application of Abstract Model Theory" Proc. of ICALP 80,Lecture Notes in Computer Science,no.85 (1980)

Manna,Z. and A.Pnueli,"Temporal Verification of Concurrent Programs: The Temporal Framework for Concurrent Programs" The Correctness Problem in Computer Science, edited by R.S.Boyer and J.S.Moore, Academic Press, (1981)

Manna,Z and A.Pnueli,"How to Cook A Temporal Proof System for Your Pet Language" Proc.of POPL ACM(1983)

Manna,Z. and A.Pnueli,"Proving Precedence Properties: The Temporal Way" Proc. of ICALP 83 (1983)

Manna,Z. and A.Pnueli,"The Modal Logic of Programs" Lecture Notes in Computer Science,no.71 (1979)

Manna,Z. and R.Waldinger "Is "Sometimes" Sometimes Better than "Always"? :Intermittent Assertions in Proving Program Correctness" CACM.vol.21,no.2(1978)

Manna,Z.,"Logic of Programs" Proc. of IFIP 80 (1980)

Manna,Z.,"Mathematical Theory of Computation"(1974)

Mckeag,R.M. and P.Milligan,"An Experiment in Parallel Program Design" Software Practice and Experience, vol.10,no.9(1980)

Meyer,A.R. and J.Y.Halpern,"Axiomatic Definitions of Programming Languages: A Theoretical Assessment" Journal of ACM,vol.29,no.2,(1982)

Meyer,A.R. and R.Parikh,"Definability in Dynamic Logic" JCSS,vol.23 (1981)

Milner,R.,"A Calculus of Communicating Systems" Lecture Notes in Computer Science,no.92(1980)

Misra,J., and K.M.Chandy,"Proofs of networks on Processes" IEEE Trans.on Software Eng.,SE-7,4(1981)

Nakajima,R.,M.Honda and Nakahara H.,"Hierarchical Program Specification and Verification - a Many-sorted Logical Approach" Acta Informatica no.14(1980)

Nakajima,R.,T.Yuasa and K.Kojima,"The IOTA Programming System - A Support System for Hierarchical and Modular Programming" Information Processing 80, North Holland (1980)

Nakamura,A.,"The Computational Complexity of Satisfiability in Systems of Modal Logic: Extended Abstract" IBM Symp.on Mathematical Foundation of Computer Science (1980)

Nielsen,M.,G.Plotokin and G.Winskel,"Petri Nets,Event Structures and Domains,Part 1" Theoretical Computer Science,vol.13 (1981)

O'Donnel,M.J.,"A Critique of the Foundations of Hoare Style Programming Logics" CACM vol.25,no.12,(1982)

Ossefort,M.,"Correctness Proofs of Communicating Processes:Three Illustrative Examples from The Literature" ACM Trans. on Programming Language and Systems,vol.5,no.4(1983)

Owicki,S. and D.Gries,"Verifying Properties of Parallel Programs: An Axiomatic Approach" CACM,vol.19,no.5 (1976)

Owicki,S.,"A Consistent and Complete Deductive system for the Verification of Parallel Programs" Proc. of 8th ACM STOC (1976)

Parikh,R."A Decidability Result for Second Order Process Logic" Proc. of the 19th IEEE Symp. of FOCS (1978)

Park,D.,"A Predicate Transformer for Weak Fair Iteration" Proc.of 6th IBM Symp. on Mathematical Foundation of Computer Science (1981)

Park,D.,"Concurrency and Automata on Infinite Sequence" Theory of Computation Report no.35 The University of Warwick (1981)

Park,D.,"Fixpoint Induction and Proofs of Program Properties" Machine Intelligence,no.5 (1969)

Park,D.,"On the Semantics of Fair Parallelism" Lecture Notes in Computer Science no.86 (1980)

Peterson,J.L.,"Perri Net Theory and The Modeling of Systems" Prentice-Hall,(1981)

Pnueli,A.,"The Temporal Logic of Programs" Proc.of the 18th IEEE Symp of FOCS (1977)

Pnueli,A.,"The Temporal Semantics of Concurrent Computation" Proc. of the Symp on Semantics of Concurrent Computation,Lecture Notes in Computer Science(1979)

Pnueli,A.,"The Temporal Semantics of Concurrent Program" Theoretical Computer Science,vol.13 (1981)

Polak,W.,"Program Verification Based on Denotational Semantics" Proc. of ACM POPL 81 (1981)

Pratt,V.R.,"Semantical Consideration on Floyd-Hoare Logic" Proc. of 17th IEEE Symp of FOCS (1976)

Ron,D.,F.Rosemberg and A.Pnueli"A Hardware Implementation of the CSP Primitives and Its Verification" Proc. of ICALP 84(1984)

Roper,T.J. and L.J.Barker,"A Communicating Sequential Process Language and Implementation" Software Practice and Experience,vol.11,no.11(1981)

Rosen,B.K.,"Correctness of Parallel Programs : The Church-Rosser Approach" Theoretical Computer Science,vol.2 (1976)

Salwicki,A. and T.Muldner,"On the Algorithmic Properties of Concurrent Programs" Lecture Notes in Computer Science no.125 (1979)

Schwartz,J.S.,"Denotational Semantics of Parallelism" Lecture Notes in Computer Science, no.70 (1979)

Shapiro,E.Y.,"A Subset of Concurrent Prolog and its Interpreter" TR-003,ICOT,(1983)

Shapiro,E.Y. and A.Takeuchi,"Object Oriented Programming in Concurrent Prolog" TR-004,ICOT,(1983)

Shaw,A.C.,"Software Descriptions with Flow Expressions" IEEE Trans. Software Eng.SE.4,3 (1978)

Shaw,A.C.,"The Logical Design of Operating Systems" Prentice-Hall (1974)

Silberschatz,A."Communication and Synchronization in Distributed Systems" IEEE Trans.Software Eng.,vol.5,no.6(1979)

Silberschatz,A."Port Directed Communication" Comp.J.,vol.24.1.(1981)

Soundararajan,N.,"Axiomatic Semantics of Communicating Sequential Processes" ACM Trans. on Programming Language and Systems,vol.6,no.4(1984)

Stark,E.W.,"Semaphore Primitives and Starvation-Free Mutual Exclusion" Journal of ACM,vol.29,no.4,(1982)

Van Lamsweerde,A. and M.Sintzoff,"Formal Derivation of

Strongly Correct Concurrent Programs" Acta Informatica vol.12,no.1,(1979)

Wolper,P.,"Specification and Synthesis of Communicating Processes Using An Extended Temporal Logic" Proc.of ACM POPL(1982)

Wolper,P.L.,"Synthesis of Communicating Processes from Temporal Logic Specification" Research Report,Stanford University (1982)

Van Emde Boas,P.,"The Connection Between Modal Logic and Algorithmic Logics" Lecture Notes in Computer Science,no.64 (1978)

Zedan,H.,"A Note on Deadlock-Free Proofs of Network of Processes" SIGPLAN Notice,v.19,no.10.(1984)

参考文献（和文）

泉　稲垣　本多，　正規集合のアドヒーレンスによって得られるω言語とω有限オートマトン，　電子通信学会論文誌（D）Ｊ６６－Ｄ，no．３（昭５８）

北　坂部　稲垣　本多，　抽象データタイプに基づく形式言語の意味記述，電子通信学会，技術報告ＡＬ８－２３　（昭５８）

中島，　数理情報学：スコット・プログラム理論，　朝倉書店　（昭５７）

細井，　論理数学，筑摩書房　（昭４９）

宮地，　並行プログラムの意味論，　情報処理　vol．２６，no.1　（昭６０）

村上　稲垣　本多，　並行プログラム検証のためのダイナミックロジックの拡張，　電子通信学会，部門別全国大会　４１（昭５６）

村上　稲垣　本多，　並行プログラム検証のためのダイナミックロジックの拡張：シャッフル操作に関する推論規則の導入，電気関係学会東海支部連合大会，３７７　（昭５６）

村上　稲垣　本多，　非決定性プログラムの全面的正当性，電子通信学会，全国大会　１２３７　（昭５６）

村上　稲垣　本多，　非決定性プログラムの全面的正当性，京都大学数理解析研究所講究録，４５８　（昭５７）

村上　稲垣　本多，　並行プログラムの検証のためのダイナミックロジックの拡張，電子通信学会，技術報告ＡＬ８２－１　（昭５７）

村上　稲垣　本多，　並行プログラムの検証体系に関する完備な述語のクラス，電気関係学会東海支部連合大会，４０２　（昭５７）

村上　稲垣　本多，　並行プログラムの検証と完備な述語のクラス，電子通信学会，技術報告ＡＬ８２－９６　（昭５８）

村上　稲垣　本多，　プログラムの検証における算術的階層の完備性情報処理学会　全国大会　１Ｋ－８　（昭５８）

村上　稲垣　本多，　２階述語論理に基づく並行プログラム検証体系についての考察，　電子通信学会　全国大会　１３４１　（昭５８）

村上　稲垣　本多，　並行プログラムの検証と完備な述語のクラス，京都大学数理解析研究所講究録，４９４　（昭５８）

村上　稲垣　本多，　被防護命令プログラムの検証における背理法に基づく推論規則について，電気関係学会東海支部連合大会，４０８　（昭５８）

村上　稲垣　本多，　並行プログラムの不変的性質の証明について，情報処理学会　全国大会　２Ｇ－２　（昭５８）

村上　稲垣　本多，　相互通信をもつ逐次型プロセスの停止性の検証電子通信学会　全国大会　１５１９　（昭５９）

村上　稲垣，　相互通信プロセス系のセマンティクス，電気関係学会東海支部連合大会，４０６　（昭５９）

村上　稲垣　本多，　並行プログラム検証体系のためのダイナミック論理の拡張，　電子通信学会論文誌（Ｄ）Ｊ６７－Ｄ，no．１０　（昭５９）

村上　稲垣，　相互通信プロセス系のKripke的セマンティクス，電子通信学会，技術報告ＡＬ８４－５３　（昭６０）

村上　稲垣，　相互通信逐次型プロセス系のKripke的意味論，　情報処理学会　全国大会　１Ｋ－３　　（昭６０）

村上　稲垣，　相互通信プロセス系の部分的正当性検証体系，　電子通信学会全国大会　１５１８　　（昭６０）

村上　稲垣，　相互通信逐次型プロセス系の検証，　京都大学数理解析研究所講究録，掲載予定

村上　稲垣，　相互通信逐次型プロセス系の部分的正当性検証体系，　電子通信学会論文誌（Ｄ）掲載予定

村上　稲垣，　相互通信逐次型プロセス系のデッドロックフリー性の検証，電子通信学会論文誌（Ｄ）投稿中

山下　稲垣　本多，　有限状態スケジューラを有する並行プログラム図式，電子通信学会論文誌（Ｄ）Ｊ６３－Ｄ，no．８　（昭５５）