ペトリネットの新しい解析法とそのプログラム自動生成への応用に関する研究

Studies on the New Analysis Method of Petri Nets and Its Application to Automatic Program Generation

2012 年 9 月

恐神　正博

内容梗概

本論文では,離散事象システムのモデルとして理論/応用の両面で有用とされるペトリネットについて，その数学的解析の基本となる状態方程式について考察し，Fourier-Motzkin 法およびその改良によって，ペトリネットの状態方程式の解の表現ならびに従来の方法では得られなかった解が新たに求められることを明らかにするほか，本来，離散事象に対するものとして提案されたペトリネットの概念を一般化した時間なし連続ペトリネットについても解析を行っている．また，IT 社会の急速な進展に伴い，ますます重要度が増しているプログラムの生産性向上の技術のうち，プログラムの自動生成について取り上げ，その試作システムの構築を行っている．さらに試作したプログラム自動生成システムの運用において必須となるモジュールの入出力条件のチェックにペトリネットを利用する手法を開発し，その有用性を実証している．

本論文は，序論（第 1 章），本論（第 2 章～第 4 章），結論（第 5 章）から構成されている．

ペトリネットとは，1962 年に C. A. Petri によって提案された有向グラフを用いた手法で，離散事象システムのモデル化や解析に用いられるなど，数学的にも解析が可能なツールとして知られている．ペトリネットには，可達性，有界性，安全性，活性などの性質があり，モデル化されたシステムについてこれらの性質を解析することで，システムの構造と動的な挙動について評価・変更・改良をすることが可能となる．また，グラフィックなツールとして，フローチャートやブロックダイヤグラム・ネットワークと同じようにシステム構造の可視的な表現手段として使用できるだけでなく，ペトリネットの中のトークンと呼ばれるものを使用することにより，システムの並行的でダイナミックな事象をシミュレートすることができる．一方，その数学的な解析の手法として状態方程式を用いた方法があげられる．状態方程式には一般に無限個の解が存在するが，無限個の解は有限個の解を組み合わせることによっても表現することができる．本論文ではペトリネットの状態方程式について考察し，通常 T-インバリアントを求める手法である Fourier-Motzkin 法において接続行列を拡張することで，状態方程式の解（すなわち発火回数ベクトル）を求めることができることを示す．また，解析において有用な概念となる無限個の解（発火回数ベクトル）を有限個の解（T-インバリアントと特解）で表す際の展開係数の導出にも Fourier-Motzkin 法のアルゴリズムがそのまま適用できることを示す．さらに，Fourier-Motzkin 法を拡張し従来の Fourier-Motzkin 法では求めきれなかった解について，それらの一部を求められるアルゴリズムの提案も行う．一方，本来，離散事象に対するものとして提案されたペトリネットの概念を一般化した連続ペトリネット，および，プレース/トランジションペトリネットを組み合わせたハイブリッドペトリネットの状態方程式においても，Fourier-Motzkin 法が有効であることを示す．

また，IT 社会の急速な進展に伴い，コンピュータは各種産業構造の中に深く浸透し，それに伴いソフトウェアの需要激増，および，その複雑化が進んできている．これは，1968 年ドイツの Garmisch における NATO の国際会議で既に指摘されていた，いわゆる「ソフトウェア危機」であるが，これに対し，各方面でソフトウェアの生産性を向上させるため，自然言語によるプログラミング，モジュールの再利用化，プログラミングの自動化など，様々な研究が行われて来た．しかしながら，現在でも組込み系ソフトウェアの技術者不足が強く叫ばれるなど，ソフトウェア危機に対する解決策はいまだ見つかっていないのが現状である．本論文ではソフトウェアの生産性を向上させる技術とプログラムの自動生成をとりあげ，汎用性の高い再利用可能なモジュールを用意し，それらを効率的に検索・生成することを考え，そのための実験システム MAPP (Module Aided Programming system by Prolog)を構築した. 実験システム MAPP は, Prolog 言語を用いて構築されており，これは単一化を中心とするリスト処理などに優れ，情報の参照や変換の処理を比較的簡潔にプログラム化できるほか，その推論能力により，モジュールの論理チェックや補填・生成などを行う潜在能力を持ち合わせている．実験システム MAPP ではモジュールごとに持たせた入出力条件により，必要なモジュールをシステムが誤りをできるだけ補う形でプログラムを設計することができるなど，自動補填の考えを新たに導入している．また，プログラムの自動生成システムにおいては，でき上がったプログラムにおける実行可能性の検証を事前に行うことが必要である． このため，ここでは自動生成されたプログラムのモジュールに対しペトリネットによるモデル化を行い，ペトリネットの性質の一つである可達性の判定条件を応用することにより，モデル化されたプログラムの実行可能性について検証できることを示している．

本論文では，これらペトリネットについての新たな知見を示すとともに，自動生成されたプログラムに対しペトリネットによるモデル化を行うことで，ペトリネットの性質の一つである可達性の判定条件を用い，モデル化されたプログラムの実行可能性について検証することを提案している．

第 1 章では，ソフトウェア技術者不足の背景や現状，それに対するソフトウェア生産性向上のための技術についての経緯，および，その中の１つであるプログラムの自動生成技術について述べるとともに，ペトリネットと呼ばれる数学的に解析が可能なグラフィック的なツールについて述べている．

第 2 章では，ペトリネットと呼ばれるモデル化の手法について述べ，数学的解析を行うことで，システムの安全性や可動性が検証できることを示すとともに，解析の際，重要な情報を求める手法の Fourier-Motzkin 法について述べている．また，解析の際重要な情報となる展開係数の導出にも Fourier-Motzkin 法のアルゴリズムがそのまま適用できることや，Fourier-Motzkin 法の改良を行い従来の方法では求めきれていなかった解の一部も導出できること，さらに時間な

し連続ペトリネットに対する状態方程式に対しても Fourier-Motzkin 法が有効であることなどを新たに示している．

第 3 章では，プログラムの生産性を向上させる手法の一つである，プログラムの自動生成について，その一構成法を示し，実験システム MAPP の構築をとおし，その検討を行っている．また，各モジュールの入出力条件を用いた自動補填の概念を新たに導入し，実験によってその効果を確認している．

第 4 章では，プログラムの自動生成における入出力条件のチェックに，第 2 章で述べたペトリネットにおける可達判定が有効であることを述べるとともに，例題をとおし，その検証を数学的手法を用いて行っている．

第 5 章は結論であり，プログラムの生産性を向上させる手法の一つである，プログラムの自動生成において，本論で述べた，生成されたプログラムの検証方法としてのペトリネットによる可達条件を用いた数学的検証法が有効であることについてまとめている．

目　次

１章　序　論

１．１　研究の背景

ペトリネットは，ドイツ人，C.A. Petri によって 1962 年に提案された概念であり，システムにおける条件と事象の概念を用いてシステムをグラフモデル化したものである．ペトリネットは，多くのシステムに適用可能なグラフィカルなモデル化ツールであるが，特に，並列非同期同時進行する複数のプロセスからなる，いわゆる離散事象システムを表現することに適している．また，システムの動作を考える上で最も基本的な要求である，発散しないか（安全であるか），目的の状態に到達できるか，デッドロックを生じないか（停止してしまわないか），などについてはペトリネットにおける，有界性問題，可達問題，活性問題として扱われており，ペトリネットではこれらの性質を数学的に解析可能であるという性質をあわせ持っている．

この数学的に解析可能という部分について，各方面で様々な研究がなされてきており，特に可達問題については，被覆木，状態方程式，ペトリネットの構造と性質に着目する 3 つの手法があるが，被覆木を用いた方法は一般に膨大な計算量を必要とする一方，状態方程式を用いる方法は，ペトリネットの性質を代数方程式の解の存在として考察できる利点があるため，その解法や，解の表現方法などについての研究が進められてきている[1].

一方，1946 年にアメリカのペンシルバニア大学において世界初の電子計算機 ENIAC が開発されて以来，電子計算機はわずか数十年の間にめざましい発展を遂げ，現在，産業構造の根幹を支える重要な役割を担うまでに至っている．これに伴い，ソフトウェアの需要は増加の一途をたどり，現在では，需要に供給が追いつかない状態に陥ろうとしている．これは，1968 年ドイツの Garmisch における NATO の国際会議において既に指摘されており，いわゆる「ソフトウェア危機」と呼ばれるものである．このソフトウェア危機を回避しようと，ソフトウェアの生産性を向上させるための様々な技術が研究・開発されてきた．例えば，構造化プログラミング，オブジェクト指向プログラミング，自然言語によるプログラミング，モジュールの再利用化，プログラミングの自動化などがあげられる．

この中で，プログラミングの自動化，いわゆるプログラムの自動生成技術とは，その昔，産業革命において手工業から機械工業に変わることで，劇的に生産性を向上させてきたように，ソフトウェアも機械的に生産させるという分野である．このソフトウェアの生産を機械化する（自動化する）自動生成の技術については，C.Green らによる Refine などがあるが，システムを零から作り上げるというよりはリエンジニアリングの方向として成功している．また他にも対象を絞った形での成功例もいくつかあるが[2][3]，ソフトウェア全般の需要を満たすまでには至っていない．また，近年では図 1-1 に示すように組み込みソフトウェア技術者の不足も懸念されており，いわゆるソフトウェアの生産性向上の技術はますます必要になってきている．

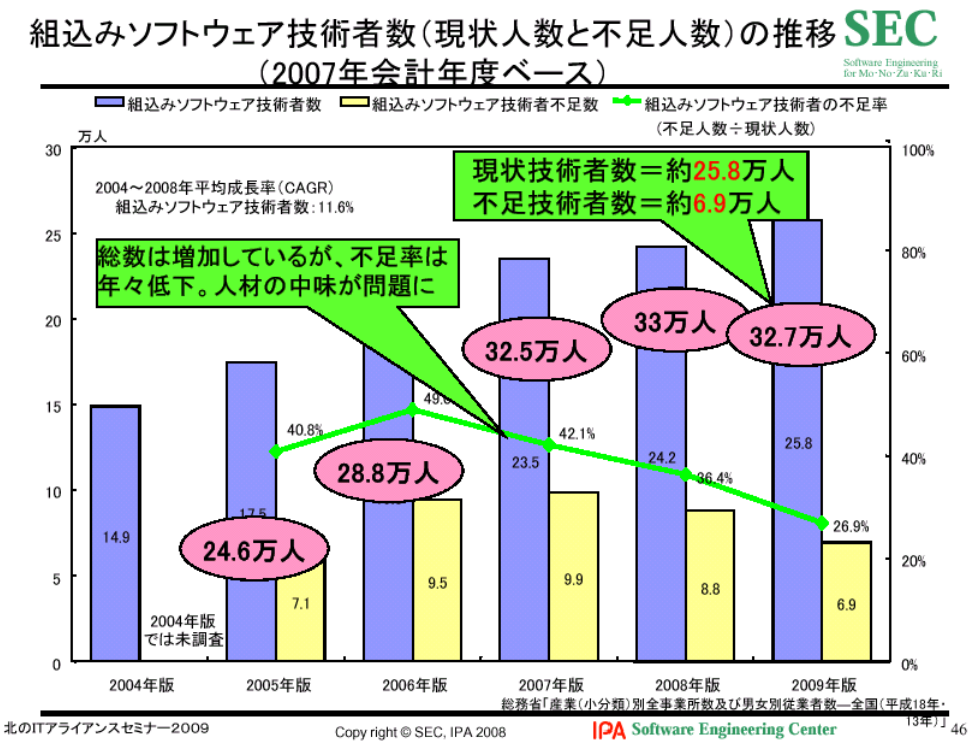


図 1-1　組込みソフトウェア技術者不足の現状

（IPA Softwere Engineering Center 主催 北の IT アライアンスセミナーにおける資料より，2009.）

１．２　研究の目的

ペトリネットにおけるシステムの解析において，可達問題における状態方程式の解法には，一つの状態方程式の非負整数解が無限個存在するということが知られている[1]．本研究では，これら無限個の解は有限個の解を用いて表現できる[4]ことから，この有限個の解の導出法の一つである Fourier-Motzkin 法について検討を行う．その結果，この Fourier-Motzkin 法におけるアルゴリズムが，有限個の解から無限個の解を表現するための展開係数導出にもそのまま適用できることや，本来，離散事象に対するものとして提案されたペトリネットの概念を一般化した時間なし連続ペトリネットについても同様にこの Fourier-Motzkin 法が適用できることを示す．さらに，Fourier-Motzkin 法におけるアルゴリズムの改良を行うことで，従来の Fourier-Motzkin 法では得られなかった解の一部が得られることを示すなど，ペトリネットにおける新たな解析法の提案を行う．

一方，プログラムの生産性を向上させる手法の一つとして，汎用性の高い再利用可能なモジュールを用意し，それらを効率的に検索・生成することでプログラムを自動的に生成する，いわゆるプログラムの自動生成についても考え，その実験システム MAPP を構築している．実験システム MAPP は Prolog を用いて構築しており，情報の参照や変換の処理を比較的簡潔にプログラム化できるほか，その推論能力により，モジュールの論理チェックや補填・生成などを行

う潜在能力を持ち合わせている．このことにより，必要なモジュールを簡単なキーワードを用いて呼び出すことができ,それらのカスタマイズをメニューに沿って行うことができる.また，設計中のプログラムの構造化図を日本語などの自然言語を用いて表示しながら設計を進めることができ，システムが誤りをできるだけ補う形でプログラムを設計することができる．

ところで，従来より行われてきた多くのプログラムの自動生成において，生成されるプログラムの検証は，変換規則の中に確からしさを含め，その変換規則を経て生成されることで行ってきた．本論では，組み合わせたモジュールの入出力条件チェックに焦点をあて，ここにペトリネットを用いることにより，数学的な裏づけを付加することで生成されるプログラムの検証を行う方法について検討することを目的としている．

１．３　論文の構成

本論文の第 1 章は序論であり，研究の背景，研究の目的について示した後，各章の概要について記述している．

第 2 章では，離散事象システムのモデル化および解析の手段として有用なペトリネットの新しい解析法について述べている．まず，ペトリネットの基本的な説明と，その解析に用いられる Fourier-Motzkin 法の説明を行い，それを用いた解の導出の他，Fourier-Motzkin 法で用いられているものと同一のアルゴリズムで，その他の解を表現するための展開係数の導出ができることや，今までの Fourier-Motzkin 法では導出できなかった解を新たに求めるための改良アルゴリズムを導くなど，Fourier-Motzkin 法が当初の目的である T-インバリアントの導出以外にも様々に活用できることを示している．

第 3 章では，プログラムの生産性向上の手法の一つとして Prolog を用いたプログラムの自動生成システムの構成法について，その実験システム MAPP(Module Aided Programming system by Prolog)の試作・検討について報告を行っている．システムおよび各機能ごとの構成について述べ，自動生成の例をあげるとともに，生成の際行われている，モジュールの入出力条件のチェック法についても説明を行っている．また，MAPP では生成の際，設計した仕様に不足等があった場合，システムができるだけ自動的にそれらを補いつつ生成する自動補填の考えを持たせており，この自動補填の手法についても説明を行っている．

第 4 章では,自動生成におけるモジュールの入出力条件のチェックにペトリネットを用いて，数学的な検証を行うことについての検討を行っている．生成されたプログラムをモデル化したペトリネットの可達判定を行うことで，そのプログラムの実行可能性を検証できることを具体的な事例を通して明らかにしている．

第 5 章は結論であり，本研究で得られた結果を総括するとともに，今後の研究の方向性についてまとめている．

第２章　ペトリネットとその解析における Fourier-Motzkin 法の利用

本章では，離散事象システムのモデル化などによく用いられる，ペトリネットと呼ばれる数学的に解析が可能なモデル化ツールの解析における新しい解析法について述べている．離散事象システムとは，並行的，分散的なふるまいを持つシステムを表し，生産システムをはじめ，通信システム，電力システム，通信プロトコルなど，情報化・分散化社会で見受けられるほとんどのシステムを離散事象として捉えることができる．このように多くのシステムに適用できる離散事象システムの解析を行うことは重要であり，それらのシステムの数学的な解析を可能にするペトリネットの解析については多方面で研究が行われてきている[1].

２．１節から２．２節においては，ペトリネットの分類，性質，およびそれらを解析するための状態方程式について述べる．２．３節においては，状態方程式の解を求める手法としての Fourier-Motzkin 法に触れ，通常 T-インバリアントを求める手法である Fourier-Motzkin 法において接続行列を拡張することで，状態方程式の解（すなわち発火回数ベクトル）を求めることができることを示す．さらに解析の際有用な概念となる，任意の発火回数ベクトルを T-インバリアントと特解で表す際の展開係数の導出にも Fourier-Motzkin 法のアルゴリズムがそのまま適用できることを示す．また，Fourier-Motzkin 法を拡張し従来の Fourier-Motzkin 法では求めきれなかった解について，それらの一部を求められるアルゴリズムの提案も行う．一方，本来，離散事象に対するものとして提案されたペトリネットの概念を一般化した時間なし連続ペトリネットおよびそれらを組み合わせたハイブリッドペトリネットの状態方程式においても，Fourier-Motzkin 法が有効であることを示す.

２．１　ペトリネットとそれらの基本的性質

ペトリネットとは，ドイツ人，C.A. Petri によって 1962 年に提案された概念であり，システムにおける条件と事象の概念を用いてシステムをグラフモデル化したものである.

ペトリネットには，可達性，有界性，安全性，活性などの性質があり，モデル化されたシステムについてこれらの性質を解析することで，システムの構造と動的な挙動について評価・変更・改良をすることが可能となる．また，グラフィックなツールとして，フローチャートやブロックダイヤグラム・ネットワークと同じようにシステム構造の可視的な表現手段として使用できるだけでなく，ペトリネットの中のトークンと呼ばれるものを使用することにより，システムの並行的でダイナミックな事象をシミュレートすることができる．以下にペトリネットの各分類および性質について説明を行っていく.

２．１．１　ペトリネットの分類

ペトリネットには，図 2-1 に示すようないくつかの種類が存在する．基本となるのは，事象生起の論理的関係のみを記述したプレース/トランジションペトリネット（P/T ペトリネット）であり，本論文においては，この P/T ペトリネットについて主に論じている．また，P/T ペトリネットには構造的，あるいは，挙動的な制限を加えたサブクラスが存在する．時間的概念を導入した時間ペトリネット（TPN：timed Petri Net）および確率ペトリネット（SPN：stochastic Petri Net）と呼ばれるもので，TPN は確定的な時間を，SPN は確率的な時間を導入している．また，論理的な構造と時間的な構造を融合したモデルとして一般化確率ペトリネット（GSPN：generalized stochastic Petri Net）もある．さらに，P/T ペトリネットの表現は単純であるため，実用的な対象を記述しようとした場合ネットのサイズが大きくなる欠点を持つことから，P/T ペトリネットを時間的関係は持たせず論理的関係についてのみ拡張した高水準ペトリネット（HPN：high level Peti Net）も提案されている．図 2-1 にこれらの分類を示す．

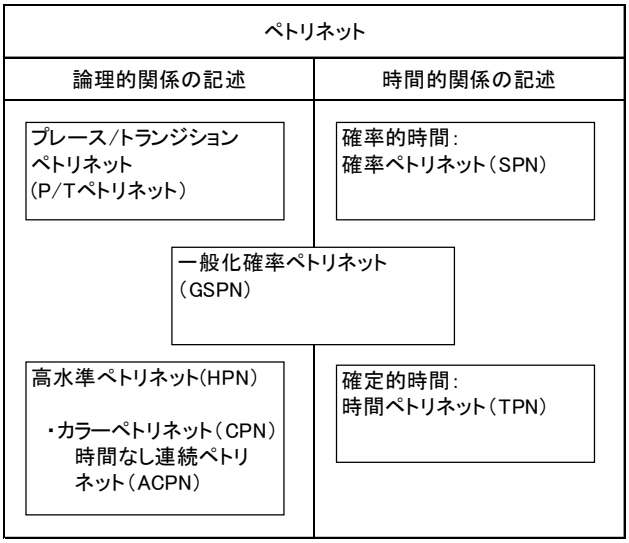


図 2-1　ペトリネットの分類

２．６節で，HPN の中に含まれる時間なし連続ペトリネット（ACPN)）や，それらを組み合わせたハイブリッドペトリネットについて述べているが，それ以外，本論文ではすべて P/T ペトリネットについて論じており，単にペトリネットと表記しているものは P/T ペトリネットを意味している．

２．１．２　トランジション発火可能性と発火則

ペトリネットは, 有向グラフ（direct graph）の１つであり, 初期マーキング（Initial marking）M_0 と呼ばれる初期状態を持つ．ペトリネットを作る基本的なグラフ N は，重みつき有向 2 部グラフであって，プレースおよびトランジションと呼ばれる 2 種類のノードからなる．ここでアークは，プレースからトランジションに接続するものあるいはトランジションからプレースへのものいずれかである. グラフ的な表現では, プレースは丸“○”, トランジションは棒“｜”または箱“■”で描かれる．アークには重み（正整数）が付記され，重みが k であるアークは，k 重アーク（多重アーク）と呼ばれ，k 本の平行なアークの集合と解釈できる．k が 1 の時の重みの付記は，通常，省かれる．プレース p が非負整数 k を割り当てられている時，p は k 個のトークン（token）でマーキングされているという．図示する時は，プレース p 内に k 個の黒い点（トークン）を描く．m 個のプレースからなるペトリネット全体のマーキングは，m 次元ベクトル M で表される．M の p 番目の成分は $M(p)$で表されプレース p 内のトークンの数を表す.

条件（condition）と事象（event）の概念を用いるモデル化においては，プレースが条件を表し，トランジションが事象を表す．トランジション（事象）は，それぞれ事象の前提条件（pre-condition）と後提条件（post-condition）とを表しているいくつかの入力プレースと出力プレースを持つ．あるプレースにトークンが存在していれば，そのプレースに関係付けられた条件が成立していると解釈できる．ペトリネットの定義を表 2-1 に示す.

表 2-1　ペトリネットの定義

ペトリネット $PN=(P, T, W, M_0)$ ，ここで

$P=\{ p_1, p_2, \cdots, p_m \}$ は，プレースの有限な集合.

$T=\{ t_1, t_2, \cdots, t_n \}$ は，トランジションの有限な集合.

$F \subseteq (P\times T) \cup (T\times P)$ は，アークの集合.（流れ関係）

$W: F \rightarrow \{1, 2, 3, \cdots\}$ は，重み付け関数.

$M_o: P \rightarrow \{0, 1, 2, 3, \cdots\}$ は，初期マーキング.

$P \cap T = \phi$ でかつ $P \cup T \neq \phi$ である.

特定の初期マーキングが規定されていないペトリネット構造は N で表される.

初期マーキングが規定されているペトリネットは (N, M_0) で表される.

多くのシステムの挙動は，システムの状態とその遷移によって記述できる．システムの動的な挙動をシミュレートするために，状態すなわちペトリネットのマーキングは，次のトランジション発火則（transition firing rule）に従って変化する.

（1）トランジション t のすべての入力プレース p に，少なくとも $w(p,t)$ 個のトークンがあれば，トランジション t は発火可能であるという．ここで，$w(p,t)$ は，入力プレース p からトランジション t へのアークの重みである．

（2）発火可能なトランジションは，発火しても発火しなくてもよい（事象は起こりえる状態にあっても，実際に起きることと起きないことがあるため）．

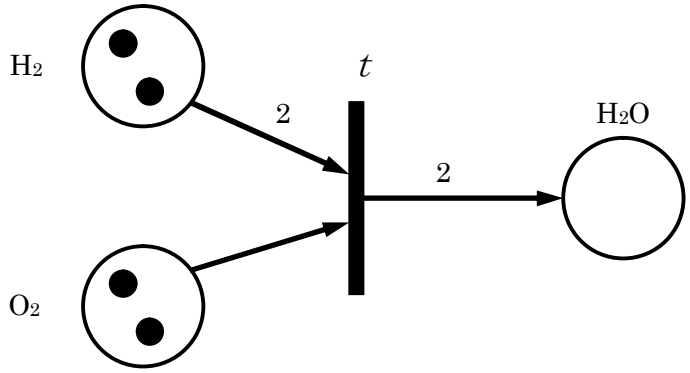
（3）トランジション t が発火すると，t の各入力プレース p から $w(p,t)$ 個のトークンが取り去られ，t の各出力プレースへ $w(p,t)$ 個のトークンが加えられる．ここで，$w(p,t)$ は，入力プレース p からとトランジション t へのアークの重みである．

入力プレースを持たないトランジションをソーストランジション（source transition）と呼び，出力プレースを持たないトランジションをシンクトランジション（sink transition）と呼ぶ．ソーストランジションは，入力プレース（前提条件）が無いので無条件に発火可能である．シンクトランジションの発火はトークンを消費するだけである．

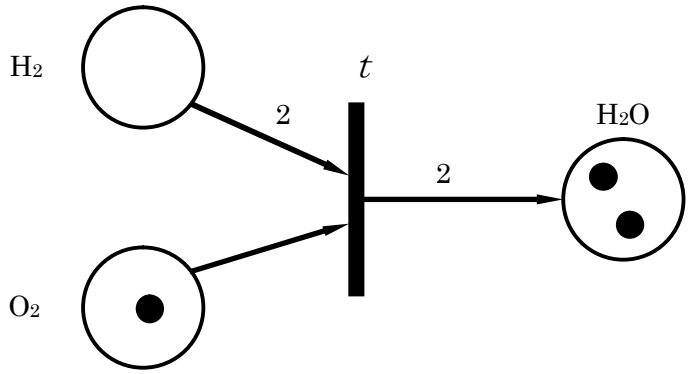
プレース p が，トランジション t の入力かつ出力プレースとなっている場合，p と t との対を自己ループ（self-roop）と呼ぶ．自己ループを持たないペトリネットを，純粋（pure）であると呼ぶ．．すべてのアークの重みが 1 であるペトリネットを，正規（ordinary）であると呼ぶ．

２．１．３　ペトリネットのトランジション発火則の例題

ここで，２．１．２項で述べたペトリネットのトランジション発火則の例題を，よく知られた化学反応：$2H_2 + O_2 \rightarrow 2H_2O$ を用いて図 2-2 で説明する．H_2 および O_2 のそれぞれ 2 分子が反応可能であることを図 2-2(a)の各プレースの 2 つのトークンは示しており，トランジション t は発火可能である．t の発火後，マーキングは図 2-2(b)で示すものとなり，トランジション t はもはや発火可能ではなくなる．



（ａ）　発火可能なトランジション t を発火させる前のマーキング



（ｂ）　t を発火させた後のマーキング
（t は発火不能となっている）

図 2-2　トランジション発火則の例

このように，様々な事象をペトリネットを用いて表すことができる．

２．１．４　マーキングに依存する並行システムの性質

ペトリネットでシステムをモデル化した際，探求できる性質としては，初期マーキングに依存する，マーキング依存の性質(Marking-dependent property)あるいは動的性質(behavioral property)と，初期マーキングとは独立な，構造的性質(structural property)の２つのタイプに分類される．

ここでは基本的な動的性質について述べる．

・可達性

可達性(reachabikity)はあらゆるシステムのダイナミックな性質を研究する基盤である．発火可能なトランジションの発火は２．１．２項で述べたトランジション発火則に従って，トークンの分布（マーキング）を変える．発火の系列は，マーキングの系列に帰着する．マーキング M_0 を M_n へ変換する発火の系列が存在するとき，マーキング M_n は M_0 から可達(reachable)であるという．発火系列（firing sequence）または生起系列（occurrence sequence）は，$\sigma = M_0 t_1 \ M_1 t_2 \cdots t_n M_n$ で表すか，あるいは $\sigma = t_1 t_2 \cdots t_n$ と略記して表す．この場合，M_n は，σ により M_0 から可達であり，$M_0[\sigma > M_n$ と記す．ネット（N, M_0）において，M_0 から可達なすべてのマーキングの集合を $R(N, M_0)$ あるいは単に $R(M_0)$ と記す．ネット（N, M_0）において，M_0 から始まるすべての発火系列の集合を $L(N, M_0)$，あるいは単に $L(M_0)$ と記す．

ペトリネットの可達問題とは，ネット（N, M_0）において，M_n に対し，$M_n \in R(M_0)$ であるかどうかを求める問題である．いくつかの応用では，プレースの部分集合のみに関心が持たれ，ネットの残りのプレースについては無関心でよいことがある．これは部分マーキング可達問題として，次のように定義される．つまりあるマーキング M_n' が与えられたとき，$M_n' \in R(M_0)$ であるかどうかを検証するものであり，ここで M_n' は，対象とするプレースの部分集合 P' に属する p' に対し，$M_n'(p') = M_n(p')$ であるが その他のプレース p については $M_n'(p) \neq M_n(p)$ であってもよいようなマーキングである．なお，可達問題は決定可能であることが示されている．

・有界性

ペトリネットは，M_0 から可達な任意のマーキングにおいて各プレース内のトークンの数がある有限な数 k を越えなければ，k−有界（k-bounded），あるいは単に有界（bounded）であると呼ばれる．このとき，すべてのプレース p およびすべてのマーキング $M \in R(M_0)$ に対して $M(p) \leqq k$ が成り立つ．L−有界の場合，ペトリネット（N, M_0）を安全（safe）であると呼ぶ．プレースはバッファや中間データを蓄えるレジスタを表すことがある．ネットが有界あるいは安全であることを検証することによって，どの発火系列がとられようと，バッ

ファやレジスタでオーバーフローが起きないことを保障できる.

・活性

活性（liveness）の概念は，オペレーティングシステム（OS）内で，デッドロックがまったく起きないことと密接に関係している. M_0 からどのようなマーキングに達していようと,ネット内の任意のトランジションを，そのマーキングから何らかの発火系列を通して発火可能にできるならば, そのペトリネット（N, M_0）は活性である（あるいは同値的に M_0 は N に対し，活性マーキング（live marking）である）と呼ぶ. これは，どのような発火系列が選ばれようと，活性ペトリネットがデッドロックのない操作を保証することを意味する.

活性は多くのシステムの理想的な性質である．しかし，大型コンピュータの OS のように複雑で大規模なシステムにとって，この厳しい性質を保障することは，コストの面からもほとんど不可能である．それゆえ，活性条件をゆるめ，次のような活性のレベルを定義する.

(0) トランジション t が，$L(M_0)$ のどの発火系列においても，発火できないならば，t は不活性（dead）（$L0$－活性）である.

(1) トランジション t が，$L(M_0)$ のある発火系列において少なくとも 1 回は発火可能であるならば，t は $L1$－活性（$L1$－live）（潜在的に発火可能）である.

(2) 任意の整数 k に対し，トランジション t が，$L(M_0)$ のある発火系列において，少なくとも k 回は発火可能であるならば，t は $L2$－活性（$L2$－live）である.

(3) トランジション t が，$L(M_0)$ のある発火系列において，無限回現れるならば，t は $L3$－活性（$L3$－live）である.

(4) トランジション t が，$R(M_0)$ のすべてのマーキングに対して，$L1$－活性ならば，t は $L4$－活性（$L4$－live）あるいは単に活性である（すなわち活性なトランジション t は，M_0 からいかなるマーキングに達しても, そこからある発火系列により発火することができる.

ネット内のすべてのトランジションが，Lk －活性，k = 0, 1, 2, 3, 4, ならば，ペトリネット（N, M_0）は，Lk －活性であると呼ばれる．$L4$－活性は最も強いものであり，先に定義した活性に相当する. 次の包含関係は容易にわかる. $L4$－活性 → $L3$－活性 → $L2$－活性 → $L1$－活性，ここで→は「包含する」ことを意味する．トランジションが Lk －活性であって，$L(k+1)$－活性で無い場合（$k = 0, 1, 2, 3$），強 Lk －活性（strictly Lk －live）であると呼ぶ.

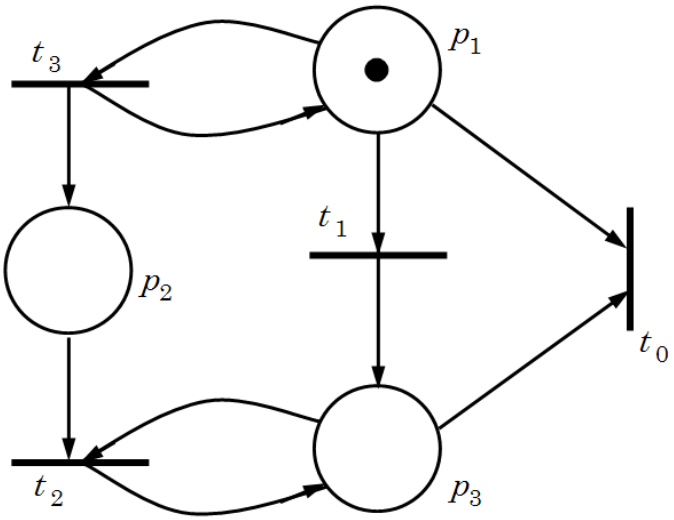


図 2-3　活性を表すペトリネットの例

図 2-3 の場合，トランジション t_0, t_1, t_2, t_3 はそれぞれ不活性（$L0$－活性），$L1$－活性，$L2$－活性，$L3$－活性となっている．

２．２　解析法について

ペトリネットの解析法は次の３つのグループに分けることができる．

（１）被覆（可達）木法（coverability (reachability) tree method）．
（２）行列方程式法（matrix equation approach）．
（３）縮約あるいは分割手法（reduction or decompsition techniques）．

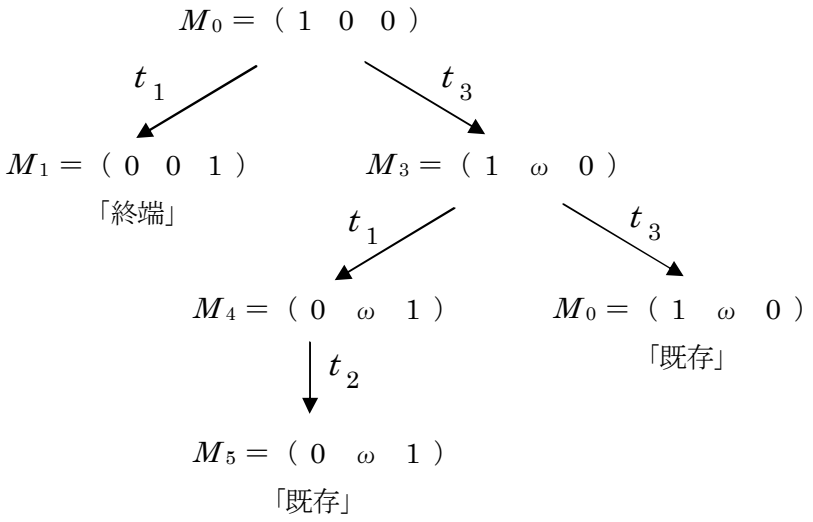
（１）の被覆木法は本質的にすべての可達マーキングあるいは被覆マーキングを数え上げるものである．これはネットの全クラスに適用可能であるが，状態空間発散（state-space explosion）の複雑さの問題のため，「小規模な」ネットに限定される．一方，状態方程式や縮約手法は強力であるが，多くの場合，ペトリネットの特定のサブクラスあるいは特殊な条件でのみ適用可能である．

２．２．１　被覆木

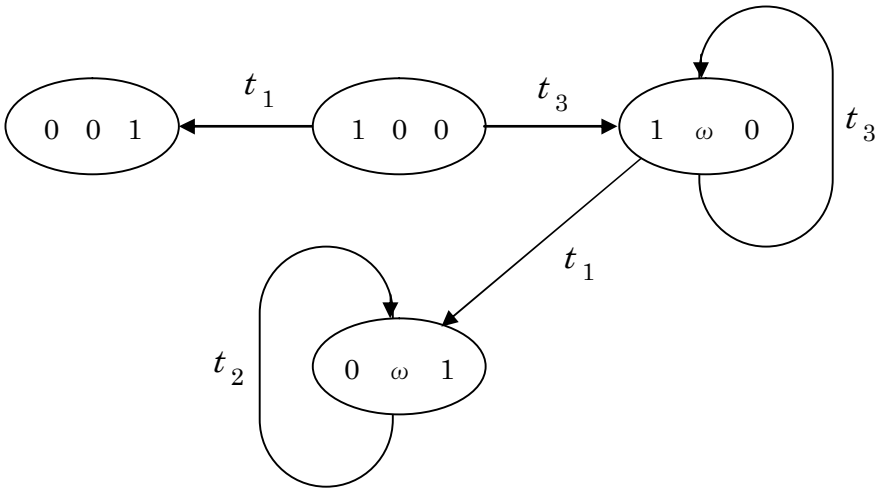
ペトリネット（N, M_0）を考える．初期マーキング M_0 から発火可能なトランジションを１回発火することにより，発火可能なトランジションと同数の「新しい」可達マーキングを得ることができる．それぞれの新しいマーキングから，またさらに新しいマーキングを得ることができる．このプロセスは，マーキングの木表現に帰着する．ノードは，初期マーキング M_0（根（root））およびそれから生成されるマーキング（子孫）を表し，各アークはトランジションの発火を表している．

しかし上記の木表現は，ネットが有界でなければ，無限に大きくなってしまう．木を有限に抑えるために，特殊な記号ωを導入する．ここでωはすべての整数 n に対して，$\omega > n$，$\omega \pm n = \omega$，$\omega \geqq \omega$ という性質を持つ．

例として，図 2-3 で示したネットを考える．初期マーキング M_0 ＝（１　０　０）に対して，２つのトランジション t_1 および t_2 が発火可能である．t_1 の発火は M_0 を M_1 ＝（０　０　１）に変える．これは，M_1 において発火可能なトランジションがないため，「終端」である．t_3 の発火により，M_0 は M_3’＝（１　１　０）となり，これは，M_0 ＝（１　０　０）を被覆しているため，新しいマーキングは M_3 ＝（１　ω　０）である．ここでは，２つのトランジション t_1 および t_3 が再び発火可能である．t_1 の発火で M_3 から M_4 ＝（０　ω　１）を得る．ここから t_2 が発火可能となり，M_4 と等しい「既存」ノード M_5 を得る．M_3 における t_3 の発火は，M_3 と等しい「既存」ノード M_6 を得る．したがって，図 2-4（a）に示す被覆木が生成される．



（a） 図 2-3 のネットの被覆木



（b） 図 2-3 のネットの被覆グラフ

図 2-4 図 2-3 のネットの被覆木および被覆グラフ

与えられたペトリネット（N, M_0）に対して，被覆木 T を用いて調べることのできる性質には，次のものがある.

（1）T のどのノードの記述にもωが現れなければ，かつそのときに限って，ネット（N, M_0）は有界である．すなわち，R（M_0）は有限である．
（2）T のどのノードの記述にも0および1しか現れなければ，かつそのときに限って，ネット（N, M_0）は安全である．
（3）T のどのアークの記述にもトランジション t が現れなければ，かつそのときに限って，そのトランジション t は不活性である．
（4）M が M_0 から可達であれば，$M \leqq M'$ であるようなノード M' が存在する．

有界なペトリネットに対しては，被覆木は，すべての可達なマーキングを含むため，可達木（reachability tree）と呼ばれる．この場合，すべての解析問題は解くことができるが，「シラミつぶし」的な方法なので非常に小規模なシステムにしか使用できないという欠点がある．一般に，記号ωを用いることによる情報の損失（ωは偶数あるいは奇数のみを表しているかもしれないし，また増減していることがわからないなど）のため，一般的に可達問題および活性問題をこの被覆木法のみを用いて解くことはできない．

２．２．２　接続行列と状態方程式

工学で研究される多くのシステムのダイナミックな性質は，微分方程式あるいは代数方程式で表現できる．方程式でペトリネットの性質を完全に解析できれば理想的である．ここでは，ペトリネットでモデル化されたシステムの性質を表す行列方程式（matrix equation）を考えてみる．これらの方程式によりある程度システムの性質を解析できるが，ペトリネットモデルの本質的な非決定的性質，および解が非負整数でなければならないという性質のため,その可解性は幾分制限されている．ここで行列方程式を取り上げる際には，常にペトリネットは純粋であるか，または，トランジションとプレースのダミー対を加えることにより純粋にすることができると仮定する．

接続行列

m 個のプレースと n 個のトランジションを持ったペトリネット N に対して，接続行列 $A=[a_{ij}]$ は $m \times n$ の整数行列であり，各成分は次の式で与えられる．

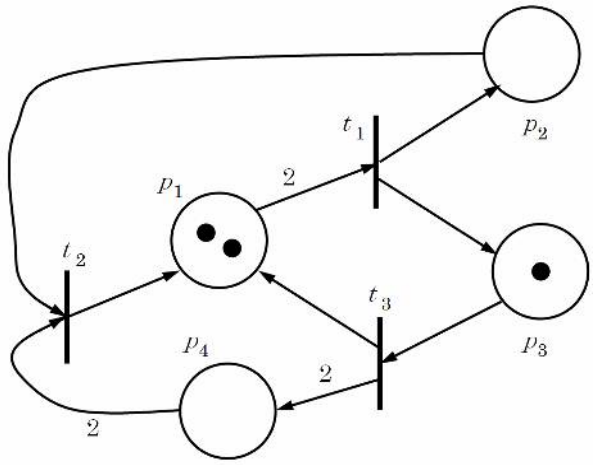
$$a_{ij} = a_{ij}^{+} - a_{ij}^{-} \qquad (2\text{-}1)$$

ここで，$a_{ij}{}^{+} = w(i,j)$ は，トランジション t からその出力プレース j へ向かうアークの重みであり，$a_{ij}{}^{-} = w(i,j)$ は，トランジション t の入力プレース j からトランジション t へ

向かうアークの重みである．２つの $m \times n$ 行列 $A^+ = [a_{ij}^+] = F$ および $A^- = [a_{ij}^-] = B$ はそれぞれ前向き接続行列（forward incidence matrix）および後ろ向き接続行列（backward incidence matrix）と呼ばれる．式（2-1）から，$A = A^+ - A^- = F - B$ を得る．もしプレース i とトランジション j において，$A_{ij}^+ = a_{ij}^-$ となるような自己ループがあれば，$a_{ij} = 0$ となる．これはプレース i とトランジション j の間にアークが無い場合と同じ意味である．このため，接続行列を取り扱う場合には，ネットが純粋であると仮定しなければならない．

２．１．２項で述べた，トランジションの発火則により，トランジション t が１回発火したときに，$a_{ij}^-, a_{ij}^+, a_{ij}$，がそれぞれプレース i から除かれる，加えられる，変化させられるトークンの数を表していることがわかる．また，トランジション j の発火条件は次式で表されることが容易にわかる．

$$a_{ij}^- \leqq M(j),\ j = 1, 2, \ldots, n \tag{2-2}$$



（ａ）　ペトリネットの例

$$A = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{bmatrix} -2 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & -2 & 2 \end{bmatrix} \end{array}$$

（ｂ）　（ａ）の接続行列

図 2-5　ペトリネットの例とその接続行列

状態方程式

以下の行列方程式において，マーキング M_k は，$m\times1$ 列の列ベクトルとして表す．M_k の i 番目の成分は, ある発火系列における k 番目の発火直後のプレース i 内のトークンの数を表す．k 番目の発火ベクトル（firing vector or control vector）u_k は，$n\times1$ 列ベクトルであり，k 番目の発火においてトランジション j が発火することを表すため，j 番目の成分は 1 でその他の成分は 0 となっている．接続行列 A の j 番目の行は，発火トランジション j によるマーキングの変化を表すから，ペトリネットの状態方程式を，次のように書くことができる．

$$M_k = M_{k-1} + Au_k,\ k = 1, 2, \ldots \tag{2-3}$$

２．２．３　無制限ネットにおける可達必要条件

目標とするマーキング M_d が, 発火系列 $\{u_1, u_2,\ \cdots,\ u_d\}$ を通して初期マーキング M_0 から可達であると仮定する．j = 1, 2, ･･･, , に対する状態方程式（2-3）を求め，和をとると

$$M_d = M_0 + A\sum_{k=1}^{d} u_k \tag{2-4}$$

を得る．これは次のように書きなおすことができる．

$$Ax = \Delta M \tag{2-5}$$

ここで，$\Delta M = M_d - M_0$ および $x = \sum_{k=1}^{d} u_k$ である．

この x は非負整数の $n\times1$ 列ベクトルであり，発火回数ベクトル（firing count vector）と呼ばれる．x の j 番目の成分は，M_0 から M_d へ変わるためにトランジション j が発火しなければならない回数を表す．線形代数方程式（2-5）は ΔM がその転置同次方程式(2-6)の解 y と直行しているとき，かつ，そのときに限って解 x を持つことは，よく知られている[5].

$$A^T y = 0 \tag{2-6}$$

ここで，接続行列 A ，その rank を r ，行数を m ，列数を n とした時,

$$A = \begin{matrix} \\ m-r \\ r \end{matrix}\!\!\begin{matrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \end{matrix}\!\!\begin{matrix} r & n-r \end{matrix} \tag{2-7}$$

で表される A_{11}， A_{21} および $n-r$ 次の単位行列を I_μ を用いて，

$$B_f = [I_\mu : -A_{11}(A_{21})^{-1}] \tag{2-8}$$

で表わされる B_f を求め，さらに，初期マーキングを M_0，最終マーキングを M_d とした場合のマーキング差 $\triangle M = M_d - M_0$ と B_f を用いると，B_f と $\triangle M$ 間には，M_0 から M_d が可達であるなら，

$$B_f \Delta M = 0 \tag{2-9}$$

が成立つことが言える[5]. そこでシステムを表すペトリネットにおいて式(2-9)が成立つかどうかを調べることで，システムの可動性を検証することができる.

２．２．４　状態方程式の解法

ここで，状態方程式（2-5）を解くことは，すなわち，発火回数ベクトルである x を求めることに他ならない．そこで，ΔM を b ，と置き直すと，式（2-5）は，次のように書き換えられる．

$$Ax = b \qquad (2\text{-}10)$$

すなわち，ペトリネットの状態方程式を解くことは，式（2-10）の $Ax = b$ の解 x を求めることと同じである．

状態方程式はペトリネットのふるまいを調べる代数的手法であるが，その際有用な手がかりを与えるのがインバリアントである．接続行列 A が与えられた場合，$Ax = 0$ ，$A^T y = 0$ を満足する整数ベクトル x および y をそれぞれ，T-インバリアント（T-invariant），S-インバリアント（S-invariant）と呼び，動的挙動を特徴付ける量である．

T-インバリアント $x \leqq 0$ または S-インバリアント $y \geqq 0$ の非ゼロ成分に対応する，トランジション または プレース の集合は，インバリアントの台集合と呼ばれ，$\|x\|$ または $\|y\|$ で表される．台集合は，その台集合の空でない真部分集合がいずれも台集合でないとき，極小であると呼ばれる．インバリアント（ベクトル）x は，すべての t に対して，$x_1(t) \leqq x(t)$ であるような他の異なるインバリアント x_1 が存在しない時，極小であると呼ばれる．インバリアントのある極小台集合を考えると，その極小台集合に対応する一意の極小インバリアントが存在する．そのようなインバリアントを極小台集合インバリアント（minimal-support invariant）と呼ぶ．すべての極小インバリアントの集合を用いて，インバリアントを生成することができる．つまり任意のインバリアントは，極小台集合インバリアントの線形結合集合として書くことができる．

なお，これら $Ax = 0$ の整数解 x（T-インバリアント）や $A^{\mathrm{T}} y = 0$ の整数解 y（S-インバリアント）は，ペトリネットの構造的性質を研究する上で有力なツールであり，解法としては，Fourier-Motzkin 法や線形計画法（LP 法）などがよく知られている[3].

２．３　Fourier-Motzkin 法

ここでは Fourier-Motzkin 法について説明を行う．Fourier-Motzkin 法は初等的ベクトル解のすべて（$Ax = 0^{m\times 1}$ の非負整数解）を含むようなベクトル解の集合を算出する方法であり，アルゴリズムは次のように与えられる[3]．ここで，$Z^{m\times n}$ は 整数を要素とする $m\times n$ 次行列の集合，　$Z_+^{m\times n}$ は非負整数を要素とする $m\times n$ 次行列の集合を表す．

＜Fourier-Motzkin（FM）法のアルゴリズム＞

入　力：接続行列　$A \in Z^{m\times n}$

出　力：初等的ベクトル解のすべてを含むベクトル解の集合

初期化：接続行列　$A \in Z^{m\times n}$ の下に単位行列　$E^{n\times n}$ を置いた

$B = [A \quad \mathrm{E}]^T \in Z^{(m+n)\times n}$ とする．

Step1；行列 B の第 i 行に対し，B の第 i 行のすべての要素が零ならば，$i = i+1$ とし，Step2 へ，少なくとも 1 つ零でない要素があれば，Step3 へ．

Step2；もし，$i \leqq m$ なら，Step1 へ，そうでなければ Step 4 へ．

Step3；行列 B の第 i 行に対し，B の第 i 行の 0 でない要素を 0 にするような B の 2 つの列の正係数一次結合（ただし係数は最小のものを選ぶ）をすべて B の列に加え，これを新しく B とする．さらに，第 i 行に 0 でない要素を持つすべての列を B から削除し，これを新しく B とする．$i = i+1$ とし，Step2 へ．

Step 4；行列 B の第 1 行から第 m 行までを取り除いた部分行列の各列は $Ax = 0^{m\times 1}$ の非負整数解を表している．

この Fourier-Motzkin 法は通常，$Ax = 0$ の解 x を求めるため，状態方程式 $Ax = b$ において $b = 0$ とした場合，すなわち，T−インバリアントを求めるための手法であるが，接続行列 A を $\widetilde{A} = [A \quad -b] \in Z^{m\times(n+1)}$ とし，$\widetilde{A}\widetilde{x} = 0$ の解 $\widetilde{x}$ を，通常と全く同じアルゴリズムにより求めることで，$b \neq 0$ の場合における $Ax = b$ の解，すなわち発火回数ベクトルを求めることができる．以下でこれを示す[6][7].

そこで，以後 $A = \widetilde{A}, x = \widetilde{x}$ とし，通常の Fourier-Motzkin 法を用いて発火回数ベクトル x を求める．

まず，例題として，図 2-6 のペトリネットを考える．

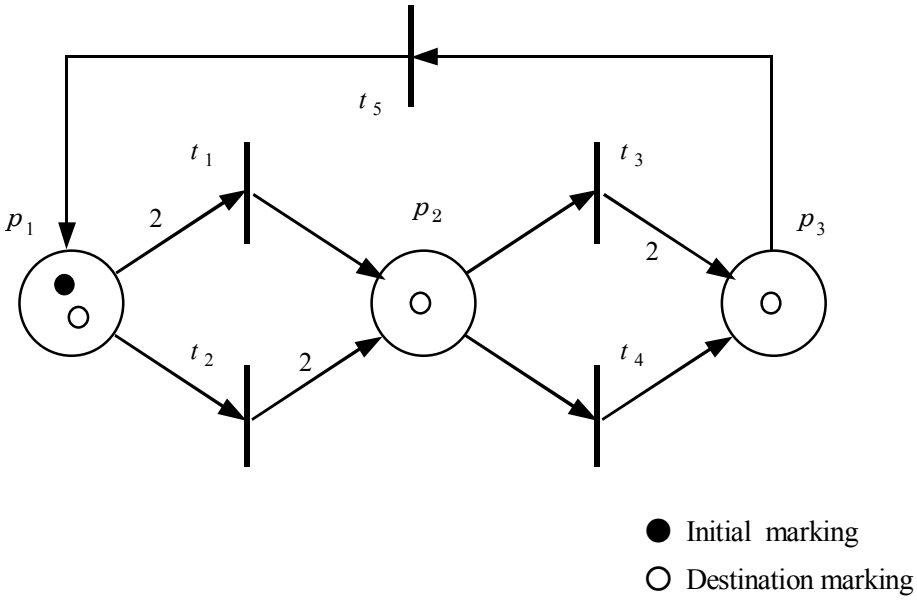


図 2-6　ペトリネットの例（その２）

図のように，最初 p_1 のみにトークンが１つある状態（Initial Marking）から，p_1, p_2, p_3 それぞれに１つずつトークンがある状態（Destination Marking）になる発火回数ベクトルを Fourier-Motzkin 法によって求める．

ここで，この接続行列 A は

$$A = \begin{bmatrix} -2 & -1 & 0 & 0 & 1 \\ 1 & 2 & -1 & -1 & 0 \\ 0 & 0 & 2 & 1 & -1 \end{bmatrix} \tag{2-11}$$

であり，また，マーキング差 b は，

$$b = M_d - M_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \tag{2-12}$$

となる．$Ax = b$ において，$\widetilde{A} = [\, A \quad -b \,] \in Z^{m\times(n+1)}$ とし，$\widetilde{A}\widetilde{x} = 0$ の解 $\widetilde{x}$ を，通常の Fourier-Motzkin 法と同じアルゴリズムにより求める．

$B = [\ \widetilde{A}\ \ E\ ]^T \in Z^{m\times n}$ より，

$$B = \left[\begin{array}{cccccc} -2 & -1 & 0 & 0 & 1 & 0 \\ 1 & 2 & -1 & -1 & 0 & -1 \\ 0 & 0 & 2 & 1 & -1 & -1 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{2-13}$$

第 1 行目の要素をすべて 0 にするように，2 つの列の正係数一次結合をつくり，新しく B に加える．

$$B = \left[\begin{array}{cccccc|cc} -2 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 2 & -1 & -1 & 0 & -1 & 1 & 2 \\ 0 & 0 & 2 & 1 & -1 & -1 & -2 & -1 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \tag{2-14}$$

第 1 行に 0 でない要素を持つすべての列を削除し，これを新しく B とする．

$$B = \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 1 & 2 \\ 2 & 1 & -1 & -2 & -1 \\ \hline 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{array}\right] \tag{2-15}$$

第 2 行以降から，3 行目まで上の操作を繰り返し，最終的に次の B を得る．

$$B = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 2 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 3 & 3 \\ 2 & 1 & 3 & 2 \\ 0 & 1 & 0 & 1 \end{array}\right] \qquad (2\text{-}16)$$

ここで，最終の行の要素が 0 の列が T-インバリアントとなり，1 の要素を持つ列が，
$Ax = b$ の解 x つまり，発火回数ベクトルとなる．

実際，第 2 列目について順に発火系列をたどってみる．$M_0(1,0,0)$ から t_2, t_3, t_5 と発火させていくと，（ 1　0　0 ）→ （ 0　2　0 ）→（ 0　1　2 ）→（ 1　1　1 ）と，M_d が得られることを確認できる．

	p_1	p_2	p_3		
	1	0	0	→	M_0
t_2	0	2	0		
t_3	0	1	2		
t_5	1	1	1	→	M_d

図 2-7　発火系列とそれぞれのマーキング（その 1）

この場合の，発火回数ベクトルは，x ＝（ 0　1　1　0　1 ）^{T} であり，式（2-16）の第 2 列目と同じであることが確認できる．また，第 4 列についても同様に見てみると，

	p_1	p_2	p_3		
	1	0	0	$\rightarrow$	M_0
t_2	0	2	0		
t_4	0	1	1		
t_4	0	0	2		
t_5	1	0	1		
t_2	0	2	1		
t_4	0	1	2		
t_5	1	1	1	$\rightarrow$	M_d

図 2-8　発火系列とそれぞれのマーキング（その２）

M_0(1　0　0) から $t_2, t_4, t_4, t_5, t_2, t_4, t_5$ と発火させていく．すると，（1　0　0）→（0　2　0）→（0　1　1）→（0　0　2）→（1　0　1）→（0　2　1）→（0　1　2）→（1　1　1）となり，この場合の式（2-16）の第 4 列目である，発火回数ベクトル x ＝（0　2　0　3　2）^{T} も M_d を得ることができていることが確認できる．

ここで，式(2-16)において求めた，$Ax = b$ の解 x において，式(2-16)の行列の n 行目の解を $x_{(n)}$ と表記した場合，発火回数ベクトルは，

$x_{(2)}$ ＝（0　1　1　0　1）^{T}，および，
$x_{(4)}$ ＝（0　2　0　3　2）^{T}

の 2 つで与えられる．一方，式(2-16)で同時に求まった解のうち，1 列目および 3 列目の解は，最後の行が 0 となっている．これは図 2-6 のペトリネットにおける T-インバリアントとして与えられる．

実際，式(2-16)における第 1 列目の解 $x_{(1)}$ ＝（1　0　1　0　2）^{T} について発火系列をたどってみる．ただし，T-インバリアントの場合，必ずしも与えられた初期マーキング M_0 から発火できるとは限らない．このため，発火可能なあるマーキング M_n(2　0　0) を考える．

M_n(2　0　0) から t_1, t_3, t_5, t_5 と発火させていくと，（2　0　0）→（0　1　0）→（0　0　2）→（1　0　1）→（2　0　0）と，再び M_n に戻ってくることが確認できる．

次に，式(2-16)における第 3 列目の解 $x_{(3)}$ ＝（1　1　0　3　3）^{T} について発火系列をたどってみる．ただし，ここでも与えられた初期マーキング M_0 から必ずしも発火でき

るとは限らない．このため，先ほどと同様の発火可能なあるマーキング $M_n(2\ \ 0\ \ 0)$ を考える．

$M_n(2\ \ 0\ \ 0)$ から $t_1, t_4, t_5, t_2, t_4, t_4, t_5, t_5$ と順に発火させていくと，（ 2　0　0 ）→（ 0　1　0 ）→（ 0　0　1 ）→（ 1　0　0 ）→（ 0　2　0 ）→（ 0　1　1 ）→（ 0　0　2 ）→（ 1　0　1 ）→（ 2　0　0 ）と，再び M_n に戻ってくることが確認できる．

また，T-インバリアントとして与えられた解 $x_{(1)}$ ＝（ 1　0　1　0　2 ）^{T} もしくは，$x_{(3)}$ ＝（ 1　0　1　2　0 ）^{T} の 他の発火系列（発火回数ベクトルの組み合わせによる別の発火系列）においても，同様にあるマーキング M_n から再び M_n に戻ってこれる．つまり，発火可能なすべての状態から $x_{(1)}$ もしくは $x_{(3)}$ に準じた発火回数をそれぞれのトランジションで行うと，必ず，もとのマーキング M_n に戻ってくることになる．

さらに別の発火回数ベクトル x' ＝（ 2　1　1　3　5 ）^{T} を考えてみると，$M_n(2\ \ 0\ \ 0)$ から $t_1, t_4, t_5, t_2, t_4, t_4, t_5, t_5, t_1, t_3, t_5, t_5$ と順に発火させていくことで，（ 2　0　0 ）→（ 0　1　0 ）→（ 0　0　1 ）→（ 1　0　0 ）→（ 0　2　0 ）→（ 0　1　1 ）→（ 0　0　2 ）→（ 1　0　1 ）→（ 2　0　0 ）→（ 0　1　0 ）→（ 0　0　2 ）→（ 1　0　1 ）→（ 2　0　0 ）のように，再び M_n に戻ってくることが確認できる．すなわち，この x' も T-インバリアントであることになるが，この x' については，$x' = x_{(1)} + x_{(3)}$ で表すことができ，同様に，$x_{(1)}$ と $x_{(3)}$ を組み合わせた任意の発火回数ベクトルも M_n から再び M_n に戻ってこれることは容易に理解できる．

ここで，他の T-インバリアントの非負有理数の線形結合では表せない T-インバリアントのことを初等的 T-インバリアントと呼ぶ．これは，任意の T-インバリアントが初等的 T-インバリアントの線形結合で表せることを意味する．

次に，$M_0(1\ \ 0\ \ 0)$ から M_d（ 1　1　1 ）に至る適当な発火回数ベクトル x'' を考えた場合，x'' ＝（ 1　1　2　0　3 ）^{T} などをあげることができる．ここで，$M_0(1\ \ 0\ \ 0)$ から $t_2, t_3, t_3, t_5, t_5, t_1, t_5$ と順に発火させていくと，（ 1　0　0 ）→（ 0　2　0 ）→（ 0　1　2 ）→（ 0　0　4 ）→（ 1　0　3 ）→（ 2　0　2 ）→（ 0　1　2 ）→（ 1　1　1 ）と，M_d を得ることができる．しかし，この発火回数ベクトル x'' は，$x'' = x_{(1)} + x_{(2)}$ で表すことができ，上で示した T-インバリアントと同様に図 2-6 で示す例題における，$M_0(1\ \ 0\ \ 0)$ から M_d（ 1　1　1 ）への到達可能な発火回数ベクトルは，式(2-16)で示したT-インバリアントの $x_{(1)}$ および $x_{(3)}$ で表しうるあらゆるT-インバリアントと，式(2-16)で示した $Ax = b$ の解（特解）である $x_{(2)}$ もしくは $x_{(4)}$ との組み合わせによって表すことができる．

２．４　発火回数ベクトルをT-インバリアントと特解で表現するときの展開係数の求め方

ここでは，Fourier-Motzkin 法を用いて発火回数ベクトルを T-インバリアントと特解で表現するときの展開係数の求め方について考える．展開係数 α とは，ある状態方程式の解が，一般に複数存在する初等的 T-インバリアントおよび特解を用いて，どのように組み合わされ表現されているかを表すものである．

一般的に，ペトリネットの状態方程式，$A \in Z^{m\times n}$，$b \in Z^{m\times 1}$，のときの，$Ax = b$ の任意の非負整数解 $x \in Z_+^{n\times 1}$ は，次式で与えられる．

$$x = \sum_{i=1}^{l} \alpha_i u_i + v_j \qquad (2\text{-}17)$$

なお，

$\alpha_i \in Z_+^{1\times 1}$,

$u_i \in U := \{u_i \in Z_+^{n\times 1};$

$Ax = b$ の初等的 T-インバリアント，$i = 1, 2, \ldots, l\}$,

$v_j \in V := \{v_j \in Z_+^{n\times 1};$

$Ax = b$ の特解，$j = 1, 2, \ldots, k\}$

である．

すなわち，ペトリネットにおける初等的 T-インバリアントと特解のすべてを求めれば，それらの組み合わせで，状態方程式 $Ax = b$ のあらゆる非負整数解 $x \in Z_+^{n\times 1}$（発火回数ベクトル）が求められる．

また，ペトリネットを解析する上で，ある状態方程式の解（非負整数解発火回数ベクトル）が，一般に複数存在する初等的 T-インバリアントおよび特解を用いて，どのように組み合わさって表現されているのか，いわゆるその展開係数を求めることは，ペトリネットの挙動検証を効率よく行う上でも有用とされている[7]．

今，式(2-17)で，U, V, $x \in Z_+^{n\times 1}$ がそれぞれ与えられていたとき，式(2-17)は，

$$\sum_{i=1}^{l} \alpha_i u_i = x - v_j$$

と変形でき，

$$[u_1, u_2, \ldots, u_l]\alpha = [x - v_j] \in Z_+^{n\times 1} \qquad (2\text{-}18)$$

となる．式(2-18)において，

$$[u_1, u_2, \ldots, u_l] \rightarrow A',$$
$$\alpha \longrightarrow x', \quad [x - v_j] \rightarrow b'$$

とおくと，式(2-18)は，

$$A'x' = b' \qquad (2\text{-}19)$$

となり，式(2-17)と同じ $Ax = b$ 型の方程式となるため，式(2-19)における x’ を求めることで，

$$\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_l]^T \in Z_+^{l\times 1} \qquad (2\text{-}20)$$

が求められる．

すなわち，ある可達な発火回数ベクトルの展開係数を求める方法の 1 つとして，Fourier-Motzkin 法のアルゴリズムをそのまま用いることが可能であることを示している．

ここで，図 2-6 で示したペトリネットにおける初等的 T-インバリアントおよび特解は，式(2-16)で示したとおり，

$$u_1 = (1\ 0\ 1\ 0\ 2)^T, \quad v_1 = (0\ 1\ 1\ 0\ 1)^T,$$
$$u_2 = (1\ 1\ 3\ 3\ 0)^T, \quad v_2 = (0\ \ 2\ \ 0\ \ 3\ \ 2)^T.$$

ただし，

$$u_i \in U := \{u_i \in Z_+^{n\times 1};\ Ax = b \text{ の初等的 T-インバリアント},$$

$$v_j \in V := \{v_j \in Z_+^{n\times 1};\ Ax = b \text{ の特解}$$

であるので，ある適当な発火回数ベクトル $x = (3\ 3\ 2\ 2\ 5)^T$ と，特解 $v_1 = (0\ 1\ 1\ 0\ 1)^T$ を考えたときの展開係数 α を，Fourier-Motzkin 法を用いて求める．

$x - v_1 = (\ 3\ \ 2\ \ 1\ \ 2\ \ 4\ )^T$となるので，式(2-17)より Fourier-Motzkin 法を用いて，

$$B = \left[\begin{array}{ccc} 1 & 1 & -3 \\ 0 & 1 & -2 \\ 1 & 0 & -1 \\ 0 & 1 & -2 \\ 2 & 1 & -4 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \tag{2-21}$$

において，第 1 行目の要素をすべて 0 にするように 2 つの正係数一次結合を作り，新しく B に加える．

$$B = \left[\begin{array}{ccc|cc} 1 & 1 & -3 & 0 & 0 \\ 0 & 1 & -2 & -2 & 1 \\ 1 & 0 & -1 & 2 & -1 \\ 0 & 1 & -2 & -2 & 1 \\ 2 & 1 & -4 & 2 & -1 \\ \hline 1 & 0 & 0 & 3 & 0 \\ 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 1 & 1 & 1 \end{array}\right] \tag{2-22}$$

第 1 行に 0 でない要素を持つすべての列を削除し，これを新しく B とする．

$$B = \left[\begin{array}{cc|c} 0 & 0 & 0 \\ -2 & 1 & 0 \\ 2 & -1 & 0 \\ -2 & 1 & 0 \\ 2 & -1 & 0 \\ \hline 3 & 0 & 3 \\ 0 & 3 & 6 \\ 1 & 1 & 3 \end{array}\right] \tag{2-23}$$

第 1 行目から第 5 行目までの要素をすべて 0 になった列について，最終行（基底）の数が 1 以上の場合，その列における各要素を基底の数で割る．

$$B = \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hline 1 \\ 2 \\ 1 \end{array}\right] \qquad (2\text{-}24)$$

ここで，展開係数 α が求められたので，確認してみると式(2-17)より

$$x = \sum_{i=1}^{l} \alpha_i u_i + v_j$$

また，

$$\alpha_1 = 1\ ,\ \alpha_2 = 2\ ,$$

$$u_1 = (\ 1\ 0\ 1\ 0\ 2\ )^T,\ u_2 = (\ 1\ 1\ 0\ 1\ 1\ )^T,$$

$$v_1 = (\ 0\ 1\ 1\ 0\ 1\ )^T$$

から確かに

$$x = (\ 3\ 3\ 2\ 2\ 5\ )^T$$

が求められる．つまり，Fourier-Motzkin 法のアルゴリズムをそのまま用いて，展開係数を求めることができることを示している．

２．５　改良 Fourier-Motzkin 法

図 2-6 における例題について，条件（M_d）を変えて Fourier-Motzkin 法を考えてみる．すなわち，M_d ＝（ 1　 1　 1 ）ではなく，M_d ＝ （ 1　 1　 2 ）とした場合，接続行列 A は変わらないため，式(2-11)はそのまま，式(2-12)が以下の式(2-25)のように変わる．

$$b = M_d - M_0 = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \qquad (2\text{-}25)$$

ここで，Fourier-Motzkin 法により，

$$B = \left[\begin{array}{cccccc} -2 & -1 & 0 & 0 & 1 & 0 \\ 1 & 2 & -1 & -1 & 0 & -1 \\ 0 & 0 & 2 & 1 & -1 & -2 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \qquad (2\text{-}26)$$

Fourier-Motzkin 法のアルゴリズムにより最終的に

$$B = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 1 & 0 \\ 0 & 4 & 1 & 3 \\ 1 & 5 & 0 & 0 \\ 0 & 0 & 3 & 5 \\ 2 & 4 & 3 & 3 \\ 0 & 3 & 0 & 1 \end{array}\right] \qquad (2\text{-}27)$$

が得られる.

ここで，基底の数が 1 以上の場合，式(2-23)のときと同じように，基底の数で割ってやる必要があるので，第 2 列目は非負整数解とはならず，特解は第 4 列目のみとなり，初等的 T-インバリアントおよび特解は,

$$u_1 = (1\ 0\ 1\ 0\ 2\ )^T,\ v_1 = (0\ \ 3\ \ 0\ \ 5\ \ 3)^T,$$

$$u_2 = (1\ 1\ 3\ 3\ 0\ )^T,$$

ただし,

$$u_i \in U := \{u_i \in Z_+^{n\times 1};\ \ Ax = b \text{ の初等的 T-インバリアント},$$

$$v_j \in V := \{v_j \in Z_+^{n\times 1};\ \ Ax = b \text{ の特解}$$

となる.

しかしながら，ある発火回数ベクトル $x = (0\ \ 2\ \ 1\ \ 2\ \ 2)^T$ を考えた時,

	p_1	p_2	p_3		
	1	0	0	$\rightarrow$	M_0
t_2	0	2	0		
t_3	0	1	2		
t_5	1	1	1		
t_5	2	1	0		
t_2	1	3	0		
t_4	1	2	1		
t_4	1	1	2	$\rightarrow$	M_d

図 2-9　発火系列とそれぞれのマーキング（その３）

$M_0(1\quad 0\quad 0)$ から $t_2, t_3, t_5, t_5, t_2, t_4, t_4$ と発火させていく．すると，（ 1　0　0 ）→（ 0　2　0 ）→（ 0　1　2 ）→（ 1　1　1 ）→（ 2　1　0 ）→（ 1　3　0 ）→（ 1　2　1 ）→（ 1　1　2 ）となり，この発火回数ベクトル $x = (0\quad 2\quad 1\quad 2\quad 2)^T$ でも M_0 から M_d が得られることがわかる．なおかつ，この発火回数ベクトル x は初等的 T-インバリアントと特解の組み合わせによって表されたものでもない．なぜならば，Fourier-Motzkin 法によって得られた $v_1 = (0\quad 3\quad 0\quad 5\quad 3)^T$ において，$x = (0\quad 2\quad 1\quad 2\quad 2)^T$の各要素 k を比べた場合，$v_1(k) > x(k)$ となる要素が 1 つ以上存在するためである．つまり，従来の Fourier-Motzkin 法ではすべての特解を求めきれていないことがわかる．

この理由としては, 列操作において第 i 行目の要素をすべて 0 にするように 2 つの正係数一次結合を作っているが，必ずしも正負それぞれの要素を持つ 2 つの列によってのみ，第 i 行目の要素を 0 にするわけではないことが上げられる．

例えば，式(2-26)の第 2 行目に着目する．従来の Fourier-Motzkin 法では正負の 2 列の組み合わせを持って正係数一次結合を作ることで第 i 行目の要素を 0 にしているため, 第 2 列と第 3 列を 2 倍した 2 つの列だけの組み合わせから新たな列を作り，追加していた．しかしながら，第 2 行目の要素を 0 にするには，2 つ以上の列の組み合わせを用いても行うことができる. 例えば, 第 2 列と第 3 列と第 4 列の組み合わせである. 従来の Fourier-Motzkin 法では，それらの列操作の情報が抜けてしまうため，アルゴリズムにおいてこの部分の改良を加えて，以下の改良 Fourier-Motzkin 法を得る．

＜改良(Modulated) Fourier-Motzkin（MFM）法のアルゴリズム＞

入　力：接続行列　$A \in Z^{m\times n}$

出　力：初等的ベクトル解のすべてを含むベクトル解の集合

初期化：接続行列　$A \in Z^{m\times n}$ の下に単位行列　$E^{n\times n}$ を置いた

$B = [A \quad \mathrm{E}]^T \in Z^{(m+n)\times n}$ とする．

Step1；行列 B の第 i 行に対し，B の第 i 行のすべての要素が零ならば，$i = i+1$ とし，Step2 へ，1 つでも零でない要素があれば，Step3 へ．

Step2；もし，$i \leqq m$ なら，Step1 へ，そうでなければ Step 8 へ．

Step3；行列 B の第 i 行において，正と負の要素の対を 1 つでも作れるならば，Step4 へ，そのような対が作れなければ，Step7 へ．

Step4；行列 B の第 i 行に対し，B の第 i 行の要素が正の列と負の列を，重みを付けずに加える．この操作で新しくできた列と第 i 要素がすでに零となっている旧行列 B (新しい列ができる前の行列）に対して，極小ベクトルの判定をする．残った新しい列をすべて旧行列 B に加え，これを新しく B とする．

Step5；加えられた新しい行列のすべての第 i 行要素が 0 であるならば，Step7 へ，1 つでも 0 でない要素を持つ列があるならば Step6 へ．

Step6；行列 B の第 i 行に対し，B の第 i 行の要素が正の列と負の列を，重みを付けずに加える．ただし，一度組み合わせた列は除く．この操作で新しくできた列と第 i 要素がすでに零となっている旧行列 B に対して，極小ベクトルの判定をする．残った新しい列をすべて旧行列 B に加え，これを新しく B とし Step5 へ．

Step7；第 i 行に 0 でない要素を持つすべての列を B から削除し，これを新しく B とする．$i = i+1$ とし，Step2 へ．

Step 8；行列 B の第 1 行から第 m 行までを取り除いた部分行列の各列は $Ax = 0^{m\times 1}$ の非負整数解を表している．

なお，この改良 Fourier-Motzkin 法のプログラムリストを付録 A に与える．

このアルゴリズムに基づき，式(2-26)から列操作を行っていく．まず，第 1 行目において重みを付けずに正と負の列の組み合わせにより列を追加する．

$$B = \left[\begin{array}{cccccc|cc} -2 & -1 & 0 & 0 & 1 & 0 & -1 & 0 \\ 1 & 2 & -1 & -1 & 0 & -1 & 1 & 2 \\ 0 & 0 & 2 & 1 & -1 & -2 & -1 & -1 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \tag{2-28}$$

さらに新たに追加された第 7 列と第 5 列の正と負の組み合わせについても列の追加を行う.

$$B = \left[\begin{array}{cccccccc|c} -2 & -1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 1 & 2 & -1 & -1 & 0 & -1 & 1 & 2 & 1 \\ 0 & 0 & 2 & 1 & -1 & -2 & -1 & -1 & -2 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right] \tag{2-29}$$

さらに第 9 列目に新たな列が追加されるが，第 7 列と第 9 列を比べた場合，第 4 行目から第 9 行目のそれぞれの要素の値が，追加された第 9 列は第 7 列より等しいかさらに大きな値を持っているため，この第 9 列は削除される.

これ以上追加できる正と負の組み合わせが存在しないため, 第 1 行目における操作が終了し，第 1 行目が 0 の値を持つ列のみを取り出し，新たに B を作り，第 2 行目に対し同様の操作を行う.

$$B = \left[\begin{array}{cccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 2 & 1 & 1 & 1 \\ 2 & 1 & -2 & -1 & 1 & 0 & -3 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \tag{2-30}$$

第 2 行目について，第 1 列から第 4 列までの正と負の組み合わせによる新たな列について，第 5 列から第 7 列を追加する．さらに新たに追加された列も含めて，正と負の組み合わせによる列の追加を行うが，すでに同じ要素の列がある場合は重複するため追加されない．

$$B=\left[\begin{array}{rrrrrrr|rrrrrr}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
-1 & -1 & -1 & 2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
2 & 1 & -2 & -1 & 1 & 0 & -3 & 3 & 2 & -1 & 1 & -2 & -5\\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 1 & 1 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 2 & 1 & 0\\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 2
\end{array}\right] \tag{2-31}$$

すると式(2-31)が得られる．これ以上第 2 行目における列操作はできないため，第 2 行目の要素が 0 の列だけを取り出し新たにこれを B として，第 3 行目に対し同様の操作を行う．

$$B=\left[\begin{array}{rrrrrrr|rrrrrrr}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
3 & 2 & -1 & 1 & -2 & -5 & 2 & 1 & -2 & 0 & -3 & -1 & -4\\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2\\
2 & 1 & 1 & 0 & 0 & 0 & 3 & 2 & 2 & 1 & 1 & 0 & 0\\
0 & 1 & 0 & 2 & 1 & 0 & 0 & 1 & 0 & 2 & 1 & 3 & 2\\
1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2\\
0 & 0 & 1 & 0 & 1 & 2 & 1 & 1 & 2 & 1 & 2 & 1 & 2
\end{array}\right] \tag{2-32}$$

式(2-32)では，正と負の組み合わせによる新たな列について，得られた列の 4 行目（$m+1$ 行目）から 10 行目（$m+n$ 行目）までの各要素のいずれかが追加する以前のすべての列の対応する各要素より小さい場合を除き，得られた列を B の列に追加する．しかしながら，追加された列ともともとの列との正と負の組み合わせを取っていくと，さらに，列が追加され，候補ベクトルの数がどんどん増えて行く．増えれば増えただけ，さらに新たな組み合わせによる列の追加が行われ，計算に膨大な時間がかかるなど，この問題については実際問題として最終的な結果まで得ることが難しい．

$$B = \left[\begin{array}{cccccccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ldots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ldots \\
3 & 2 & -1 & 1 & -2 & -5 & 2 & 1 & -2 & 0 & -3 & -1 & -4 & 1 & 0 & -1 & -2 & 0 \ldots \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ldots \\
1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 3 \ldots \\
2 & 1 & 1 & 0 & 0 & 0 & 3 & 2 & 2 & 1 & 1 & 0 & 0 & 4 & 3 & 2 & 1 & 0 \ldots \\
0 & 1 & 0 & 2 & 1 & 0 & 0 & 1 & 0 & 2 & 1 & 3 & 2 & 0 & 1 & 2 & 3 & 5 \ldots \\
1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 3 \ldots \\
0 & 0 & 1 & 0 & 1 & 2 & 1 & 1 & 2 & 1 & 2 & 1 & 2 & 2 & 2 & 2 & 2 & 1 \ldots
\end{array}\right] \quad (2\text{-}33)$$

しかしながら，計算の途中段階までで得られた特解を見てみると，第 10 列目と第 18 列目に得られていることがわかる. 第 15 列目にも 3 行目までの要素がすべて 0 となった列が存在しているが，10 行目の基底の数が 2 となっており 1 よりも大きいため非負正数解とはならない.

ここにおいて, 式(2-27)で得られた特解 $v_1 = (0 \quad 3 \quad 0 \quad 5 \quad 3)^T$ は第 18 列目で得られており，また，第 10 列目において，従来の Fourier-Motzkin 法では得られなかった，新たな $v_2 = (\ 0 \quad 2 \quad 1 \quad 2 \quad 2\ )^T$ が得られていることが確認できる. この発火回数ベクトルは図 2-9 であげた発火系列から得られるものと同じであることが確認できる.

Fourier-Motzkin 法では，計算の途中段階で，すべての候補ベクトルを記憶しておかなければならない．そこで，候補ベクトル数が非常に多くなった場合に計算機のオーバーフローが生じる問題が以前から指摘されている[3]．改良 Fourier-Motzkin 法では，従来の Fourier-Motzkin 法より多くの候補ベクトルを記憶しなければならないことから，さらにオーバーフローは生じやすくなる問題がある．ここで取り上げた例においてもオーバーフローの問題により，すべての解を求めるところまでは至ってはいないが，従来の Fourier-Motzkin 法では得られなかった解が，改良 Fourier-Motzkin 法を用いることで，新たに得ることができている.

２．６　時間なし連続ペトリネットにおける状態方程式の解および展開係数

前節までで論じていた P/T（プレース/トランジション）ペトリネットでは，その状態を表すトークンの個数が非負整数値をとり，トランジションの発火が一瞬に起こるということを前提にしている．ペトリネットを拡張するものの一つとして，本来，離散事象に対するものとして提案された P/T ペトリネットの概念を一般化した時間なし連続ペトリネットを含む，ハイブリッドペトリネットと呼ばれる非負実数のトークンと連続的に発火するトランジションを一部に許容するものが提案されている[8]．ここでは，ハイブリッドシステムのモデルとして P/T ペトリネットを拡張するときに，連続ダイナミクスを記述するための基本構造として重要である時間なし連続ペトリネット(ACPNs) の状態方程式の解を考える．ACPNs の状態方程式は通常は初期マーキングを与えたもとで，到達可能なすべての状態（マーキング）を求めるための表現として使われている．他方，初期マーキング M_0 ，目標マーキング M_d を与え，さらに $M_d - M_0$ を固定したもとでの状態方程式の解を考えることも重要であり，このとき，周知の T-インバリアントの他に，一般に複数個存在する特解を効率よく求める必要がある．そこで，時間なし連続ペトリネット(ACPNs) に対するこれらの解の導出についても Fourier-Motzkin 法が有効であること示す[9].

また，時間なし連続ペトリネットの表記については，連続で非負実数を扱うプレースを二重丸 “◎”で表し，トランジションは中抜きの箱”□”，アークにおいては，矢印に斜線を入れて表現することで区別している．

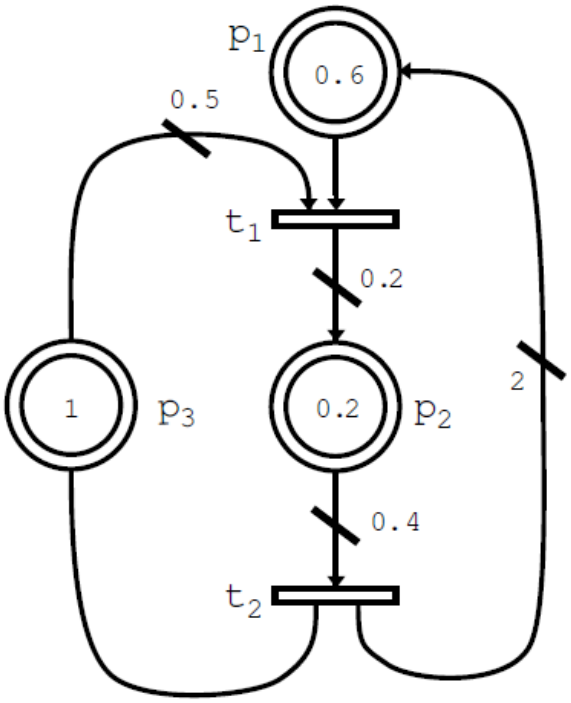


図 2-10　初期マーキング M_0 を与えられている時間なし連続ペトリネットの例

図 2-10 の時間なし連続ペトリネットを考えた場合, 状態方程式 $Ax = b$ の接続行列 A は,

$$A = \begin{bmatrix} -1 & 2 \\ 0.2 & -0.4 \\ -0.5 & 1 \end{bmatrix} = \begin{bmatrix} -1/1 & 2/1 \\ 1/5 & -2/5 \\ -1/2 & 1/1 \end{bmatrix} \in Q^{3\times 2} \tag{2-34}$$

となる．ここで，$Q^{m\times n}$ は 有理数を要素とする $m\times n$ 次行列の集合，$Q_+^{m\times n}$ は非負有理数を要素とする $m\times n$ 次行列の集合を表し，$R^{m\times n}$ は 実数を要素とする $m\times n$ 次行列の集合，$R_+^{m\times n}$ は非負実数を要素とする $m\times n$ 次行列の集合を表す．

今， t_1 ， t_2 を 1 回ずつ発火させて得られたマーキングを M_d とすると，
M_0（ 0.6　0.2　1 ） → （ -0.4　0.4　0.5 ）→（ 1.6　0　1.5 ）となり，

$$b = M_d - M_0 = (1.0 \ \ -0.2 \ \ 0.5)^T \in R^{3\times 1} \tag{2-35}$$

から，拡大接続行列

$$\widetilde{A} := [A, -b] = \begin{bmatrix} -1/1 & 2/1 & -1/1 \\ 1/5 & -2/5 & 1/5 \\ -1/2 & 1/1 & -1/2 \end{bmatrix} \in Q^{3\times(2+1)} \tag{2-36}$$

が得られ，ここにおいて Fourier-Motzkin 法を用い初等的 T-インバリアントおよび特解を求めるが，時間なし連続ペトリネットにおいては実数を扱うことになり，もともと整数を扱うことを前提としていた２．３節のアルゴリズムではそのまま用いることができない，このため，各要素の分母の最小公倍数をすべての要素に掛け合わせ，各要素を整数にしたうえで Fourier-Motzkin 法を適用することを考える．すると，式(2-36)は,

$$\widetilde{A} = \begin{bmatrix} -10 & 20 & -10 \\ 2 & -4 & 2 \\ -5 & 10 & -5 \end{bmatrix} \in Z^{3\times(2+1)} \tag{2-37}$$

となり，これを用いて Fourier-Motzkin 法を適用する．

$$B = \left[\begin{array}{ccc|cc} -10 & 20 & -10 & 0 & 0 \\ 20 & -4 & 2 & 0 & 0 \\ -5 & 10 & -5 & 0 & 0 \\ \hline 1 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 2 \end{array}\right] \tag{2-38}$$

が得られる．ここから，行列 B から T-インバリアントと特解を表している部分行列 C を取り出すと，

$$C = \left[\begin{array}{cc} 2 & 0 \\ 1 & 1 \\ 0 & 2 \end{array}\right] \in Z_+^{3\times 2} \tag{2-39}$$

となり，特解を表す C の 2 列目において基底の数で各要素を割ってやり，

$$\begin{aligned} u_1 &= (\ 2 \quad 1\ )^T, \\ v_1 &= (\ 0 \quad 1/2\ )^T = (\ 0 \quad 0.5\ )^T \end{aligned} \tag{2-40}$$

が得られる．

一方，ここでは t_1 ， t_2 を 1 回ずつ発火させて得られたマーキングを M_d とし，T-インバリアントと特解を求めたが，式(2-17)より，実際に式(2-40)で得られた u_1, v_1 を展開係数 α を用いてどのように表せるかを，２．４節で述べた Fourier-Motzkin 法を適用して求めてみる．

式(2-19)における A'， b' は，

$$\begin{aligned} A' &= [\ 2 \quad 1\ ]^T, \\ b' &= x - v_1 = [\ 1-0 \quad 1-0.5\ ]^T \end{aligned} \tag{2-41}$$

なので，

$$B = \left[\begin{array}{cc} 20 & -10 \\ 10 & -5 \\ \hline 1 & 0 \\ 0 & 1 \end{array}\right] \tag{2-42}$$

よって,

$$B = \left[\begin{array}{cc|c} 20 & -10 & 0 \\ 10 & -5 & 0 \\ \hline 1 & 0 & 1 \\ 0 & 1 & 2 \end{array}\right] \tag{2-43}$$

から，展開係数αは

$$\alpha = 1/2 \tag{2-44}$$

と得られ，今回の例で t_1 ， t_2 を１回ずつ発火させて得られた $x = (\ 1 \quad 1\ )^T$ は式(2-17)より，

$$\begin{aligned} x &= \sum_{i=1}^{l} \alpha_i u_i + v_j \\ &= 1/2(\ 2 \quad 1\ )^T + (\ 0 \quad 0.5\ )^T \\ &= (\ 1 \quad 1\ )^T \end{aligned} \tag{2-45}$$

と，確かに展開係数が得られていることがわかる.

すなわち，時間なし連続ペトリネットにおける状態方程式および展開係数導出においても，Fouire-Motzkin 法が有効であることを示すことができた.

さらに，P/T ペトリネットと時間なし連続ペトリネットを組み合わせたハイブリッドペトリネットについても同様に Fourier-Motzkin 法を適用してみる.

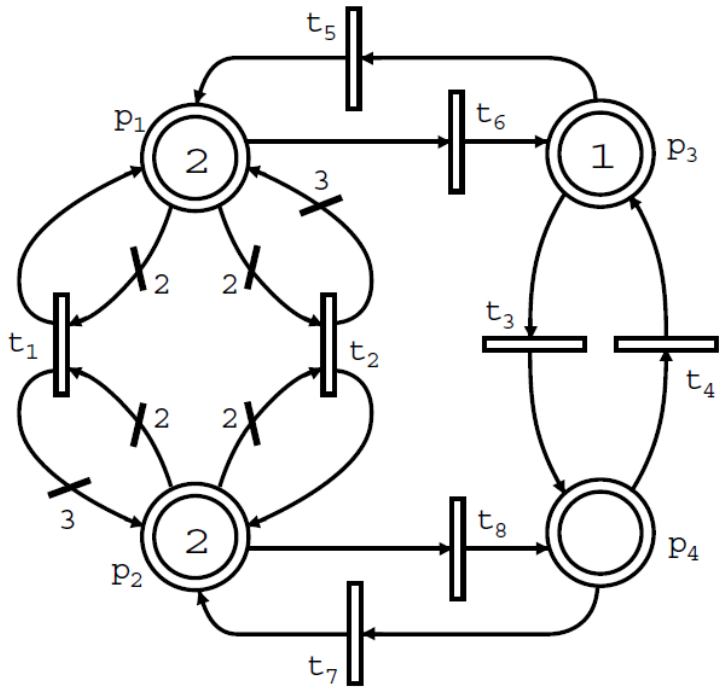


図 2-11　初期マーキング M_0 を与えられているハイブリッドペトリネットの例

図 2-11 の例題を考え，$M_0 = (\ 2\ \ 2\ \ 1\ \ 0\ )^T$，$M_d = (\ 4\ \ 0\ \ 0\ \ 1\ )^T$ の場合，すなわち $b = (\ 2\ \ -2\ \ -1\ \ \ 1\ )^T$ の場合について，状態方程式の解，および，展開係数のそれぞれが Fourier-Motzkin 法によって求まることを示す.

図 2-11 における拡大接続行列 $\tilde{A} = [\ A\ \ \ -b\ ] \in Z^{4\times 9}$ は,

$$\tilde{A} = \begin{bmatrix} -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & -2 \\ 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 2 \\ 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 & -1 \end{bmatrix} \tag{2-46}$$

から，行列 B が

$$B = \left[\begin{array}{rrrrrrrrr} -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 & -2 \\ 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 2 \\ 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 & -1 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \in Q^{13\times 9} \tag{2-47}$$

となり，最終的に得られる行列 B から，先ほどと同じように T-インバリアントと特解を表している部分行列 C を取り出すと，

$$C = \left[\begin{array}{ccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 2 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}\right] \in Q^{9\times 9} \tag{2-48}$$

となるため，ここから $U = \{u_1, \ldots, u_6\}$ と $V = \{v_1, v_2, v_3\}$ が

$$\begin{aligned} u_1 &= (11000000)^T,\ u_2 = (00110000)^T \\ u_3 &= (00001100)^T,\ u_4 = (00000011)^T, \\ u_5 &= (01100110)^T,\ u_6 = (10011001)^T. \\ v_1 &= (02100000)^T,\ v_2 = (00012002)^T, \\ v_3 &= (01001001)^T \end{aligned} \tag{2-49}$$

のように得られる．

さらに，式(2-49)で得られた $U = \{u_1, \ldots, u_6\}$ と，$v_1 = (0\ 2\ 1\ 0\ 0\ 0\ 0\ 0)^T$ を用いて，$x = (2\ 4\ 3\ 2\ 2\ 2\ 2\ 2)^T$ を表す展開係数を Fourier-Motzkin 法を用いて求める．

式(2-19)における A'，b' は，

$$A' = \begin{bmatrix} 1\,0\,0\,0\,0\,1 \\ 1\,0\,0\,0\,1\,0 \\ 0\,1\,0\,0\,1\,0 \\ 0\,1\,0\,0\,0\,1 \\ 0\,0\,1\,0\,0\,1 \\ 0\,0\,1\,0\,1\,0 \\ 0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,1\,0\,1 \end{bmatrix},$$

$$\begin{aligned} b' &= x - v_1 \\ &= (2\ 4\ 3\ 2\ 2\ 2\ 2\ 2)^T - (0\ 2\ 1\ 0\ 0\ 0\ 0\ 0)^T \\ &= (2\ 2\ 2\ 2\ 2\ 2\ 2\ 2)^T \end{aligned} \tag{2-50}$$

となり，行列 B が

$$B = \left[\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 1 & -2 \\ 1 & 0 & 0 & 0 & 1 & 0 & -2 \\ 0 & 1 & 0 & 0 & 1 & 0 & -2 \\ 0 & 1 & 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 1 & 0 & 0 & 1 & -2 \\ 0 & 0 & 1 & 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 & 0 & 1 & -2 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \in Q^{15\times 7} \tag{2-51}$$

より，最終的に得られる行列 B から，先ほどと同じように T-インバリアントと特解を表している部分行列 C を取り出すと，

$$C = \begin{bmatrix} 0\ 2 \\ 0\ 2 \\ 0\ 2 \\ 0\ 2 \\ 2\ 0 \\ 2\ 0 \\ 1\ 1 \end{bmatrix} \in Q^{7\times 2} \tag{2-52}$$

となり，$(\alpha_1, \ldots, \alpha_6) = (\ 0\ 0\ 0\ 0\ 2\ 2\ )^T$ と$(\ 2\ 2\ 2\ 2\ 0\ 0\ )^T$が得られた.

よって，ハイブリッドペトリネットにおける状態方程式および展開係数導出においても，Fouire-Motzkin 法が有効であると言える.

第３章　Prolog を用いたプログラム自動生成システムの構築

ソフトウェア危機が叫ばれて以来，プログラムの生産性を向上させるためのさまざまな技術が研究されてきた．例えば，構造化プログラミング，オブジェクト指向プログラミング，自然言語によるプログラミング，モジュールの再利用化，プログラミングの自動化などである．

ここでは，プログラム生産性向上の手法の一つであるプログラム自動生成において，開発言語に Prolog を用いた，プログラム自動生成システムの構成について取り上げる．Prolog を用いたプログラムの自動生成システムは Prolog 自体のプログラムを自動生成するシステム[10]や分野や規模を絞った形での事例しかない．しかしながら，Prolog はその強力な記号処理能力や推論機能から，複数言語へのソースコードの展開をはじめ，プログラムの自動補填など自動生成を考える上で様々な長所を持ち合わせており，それらの機能を用いることで拡張性の高いシステムの構築を行うことができる．ここでは，Prolog を用いた自動生成システムの一つの構成法として，その実験システム MAPP(Module Aided Programming system by Prolog)の試作・検討を行う[11][12].

図 3-1 に MAPP の構成と処理の流れ図を示す．MAPP では，あらかじめ用意された辞書を用い，必要なモジュールの検索，パラメータ等の情報の付加を行い，概略仕様を作成する．

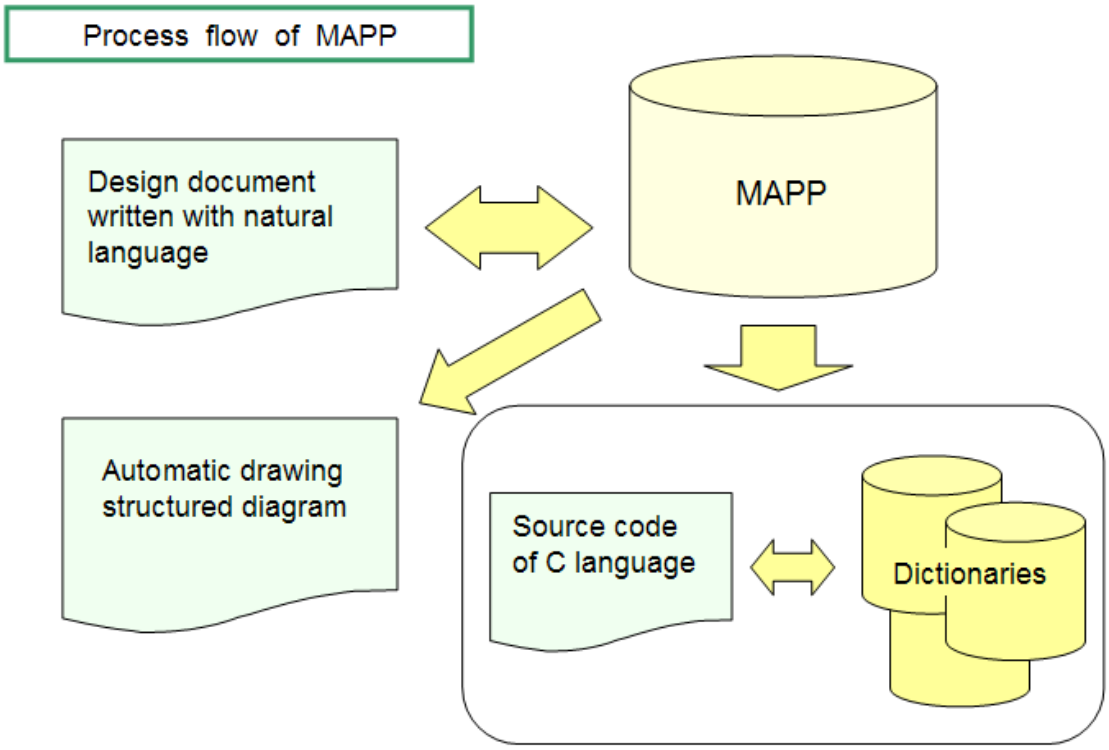


図 3-1　MAPP の大まかな構成と処理の流れ

この際，プログラムの構造化図を確認しながら作業を行うことができる．整った概略仕様から目的のプログラムを生成する．以上の基本的な流れについて，順次説明を行う．

３．１　モジュールの検索

ソフトウェアの生産性向上の方法として，ソフトウェアのモジュール化（部品化），標準化，再利用化といったことがあげられる．機能ごとに部品化されたモジュールを，標準のインターフェース上で，何度も再利用できるようになればかなりの生産性向上が期待できる．そのためには，細かく分割されたモジュールをそれらの機能，属性，対象，構造等から効率的に検索できるシステムが不可欠である．しかしながら，これらの属性項目はそれぞれのモジュールごとに存在し，全てのモジュールに対し，存在する全ての項目を持たせるのは現実的ではない．検索のための手法としてはモジュールごとに付けている名前（ファイル名）で行うのが最も簡単な方法であるが，今述べたとおりこれは非現実的である．そこで，Prolog の持つリスト処理を利用した検索を考える[14].

３．１．１　リスト処理を用いたモジュールの検索

全てのモジュールに対し式(3-1)のような辞書を設けておく．ここで process_*n* は一つ一つのモジュールに対応している処理仕様の一般形である．次に，式(3-2)の述語を設けておくことで曖昧なキーワードからも関連するモジュールを推論して検索することができる．

```
dict([[head([Key_1,Key_2,･･･,Key_n]),
       body([process_1,process_2,･･･process_n])],  ･･･  ]).    (3-1)

retr(H1):-dict(DICT),member([head(KEY),body(T)],DICT),member(H1,KEY),
          writelist_nl(T).                                      (3-2)
```

式(3-1)の Key_*n* には，process_*n* に関連するキーワードを指定する．このキーワードはリスト上に書かれているため，数字，数式，２バイト形文字，１バイト形文字を問わず必要に応じいくらでも拡張することができる．body 別に全て違うキーワードを指定しなければならないということもなく，必要であれば同じキーワードが複数の body に対して存在していてもよい．したがって，関連すると思われる全てのキーワードを body ごとに記述しておけば曖昧なキーワードからの検索も行なうことができる．

ここで process_*n* は一つ一つのモジュールに対応している処理仕様の一般形となっているため，関連するキーワードを式(3-2)の H1 に指定すれば，必要なモジュールの処理仕様がいくつ

か選び出される．writelist_nl は単に body の内容を記述するための述語である．そこで，その中から必要な処理仕様を選び，一般形からカスタマイズを施しながら設計中の処理仕様に付け加えていけばよい.

次に式(3-1)の例を示す.

```
dict([
  [head([読み込む,入力,read,input,prompt,io]),
   body([['X',に読み込む],
         ['プロンプトにより','X',に読み込む],
         ['プロンプトにより','X',の各変数に読み込む],
         [プロンプトにより１次元配列,'X',に読み込む],
         [プロンプトにより２次元配列,'X',に読み込む],
         [ファイルから,'X',に読み込む],
         [ファイル名,'FN',の第,' K' ,番目のファイルから,
              ２次元配列,'A',の第,'J',列に読み込む]
  ])],
  [head([書き出す,出力,print,write,output,io]),
    body([['X','O を,'OP_TYPE',でプリントする],
          ['X','OX',をプリントする],
               ..........
  ])],
               .......... ]).                                    (3-3)
```

ここで，式(3-2)の H1 に関連するキーワードを指定するとそのキーを持つ処理仕様の一般形が表示される.

処理仕様を呼び出すキーワードを，１バイト形，２バイト形，及び複数の関連する項目にまたがる場合とに分け，それぞれの探索結果を図 3-1,図 3-2,図 3-3 に示す.

```
?:-retr(read).

['X',に読み込む],
['プロンプトにより','X',に読み込む],
['プロンプトにより','X',の各変数に読み込む],
[プロンプトにより１次元配列,'X',に読み込む],
[プロンプトにより２次元配列,'X',に読み込む],
[ファイルから,'X',に読み込む],
[ファイル名,'FN',の第,' K' ,番目のファイルから,
２次元配列,'A',の第,'J',列に読み込む]
```

図 3-1　キーワードに１バイト形のキーを使用した場合の探索結果

```
?:-retr(読み込む).

['X',に読み込む],
['プロンプトにより','X',に読み込む],
['プロンプトにより','X',の各変数に読み込む],
[プロンプトにより１次元配列,'X',に読み込む],
[プロンプトにより２次元配列,'X',に読み込む],
[ファイルから,'X',に読み込む],
[ファイル名,'FN',の第,' K' ,番目のファイルから,
２次元配列,'A',の第,'J',列に読み込む]
```

図 3-2　キーワードに２バイト形のキーを使用した場合の探索結果

```
?:-retr(io).

['X',に読み込む],
['プロンプトにより','X',に読み込む],
['プロンプトにより','X',の各変数に読み込む],
[プロンプトにより１次元配列,'X',に読み込む],
[プロンプトにより２次元配列,'X',に読み込む],
[ファイルから,'X',に読み込む],
[ファイル名,'FN',の第,' K' ,番目のファイルから,
２次元配列,'A',の第,'J',列に読み込む]
['X',を,'OP_TYPE',でプリントする]
[１次元配列,'X',をプリントする]
[２次元配列,'X',をプリントする]
[ファイル,'X',に,'Y',を,TYPE',でプリントする]
['X','OP_TYPE'],['X',の各値を,'OP_TYPE',
で名前に続き各行にプリントする]
['X',の各値の見出し行をタブによりプリントする]
['X',の各値を,'OP_TYPE_LIST',で一行にプリントする]
[定記号列,'S',をプリントする]
[定記号パターン列,'S',をプリントする]
[定記号列,'S',をリスト要素ごとに改行してプリントする]
[定記号列,'S',と変項,'X',を,'OP_TYPE',でプリントする]
```

図 3-3　キーワードに複数の項目にまたがるキーを使用した場合の探索結果

図 3-1，図 3-2 の場合は，どちらも同じ処理に関連するキーワードを指定しているため，そのキーワードが１バイト形，２バイト形を問わず同じ処理仕様が表示された．ところが，図 3-3 においては，指定したキーワードが読み込みに関連する処理と，書き出しに関連する処理の両方にまたがっていたため，そのキーワードを持つ全ての処理仕様が表示されている．

３．２　仕様の整備

仕様は処理関連対象の型やデータ構造を指定するデータ構造仕様と，処理方法を指定する処理仕様に大別される．前者は，

objlist(spec(NUM)):-obj(x1),obj(x2),･･･,obj(x*n*).　　(3-4)

の形で与え，x*i*(*i*=1,2,･･･)が例えば初期値を持つ配列の場合，

```
obj([name(xi),type(float),array([dim(1),size(n),max(200)]),
      val([v1,v2,･･･,vn])]).                                (3-5)
```

の様な形で与える．データ構造に関する機械可読な仕様は，上式のような単一の定型的な形でほぼ完全に与えることができる．

次に処理の仕様であるが，これは概略仕様をユーザが日本語文などで箇条書風に適当に作る．算術式 AL1,AL2,･･･,AL*n* などは，簡単のため computelist(AL1,AL2,･･･,Al*n*)のように述語 computelist の引数部に書いておくものとする．

一方，日本語文からなる処理の仕様は短いものでも語彙・順序の違いなどにより，同じ意味内容の文が多くの異なる文で表されることになる．的確にユーザの欲する処理を指定するには，その処理を行なうモジュールや命令を，見出し表のコメント項目の日本語文などを参照して欲しいものを指定すればよい．

そのためには，３．１節で述べた方法により，処理の中心となる事項をキーワードで指定して処理の検索を行い，要求に見合う処理手続き文を選択したら，それに必要なカスタマイズを施しながら，設計文書に付け加えていけばよい．

次に処理対象のデータタイプなどは無視し，この種類を行なうモジュールや関数の中心機能を表す日本語文を表示させ，必要なパラメータや引数を指定して作成する．

つまり，一般的な処理仕様の形としては，一つ一つの処理項目を process_1, process_2, ･･･, process_*n* としたとき

```
process(spec(NUM)):-proc_list([ process_1,
                                process_2,
                                   ...    ,
                                process_n ]
                               ]).                              (3-6)
```

で与える．ここで process_*n* は自然言語を用いて表記していく．この時の spec(NUM)は式(3-4)で与えられるスペック番号に対応しており，また，それぞれの処理項目に対応して後述する式(3-9)のようなモジュール辞書が与えられている．

３．３　プログラムの自動生成

以上述べてきた方法で，必要なモジュールを検索し，カスタマイズを施しながら設計文書を作成することができる．ここからは，それらの設計文書からCのソースコードを生成するために必要ないくつかの規則・辞書の構成について説明する．

リスト処理を用いることで，モジュールごとに持たせるべき属性項目は一定である必要がなく，また，データ構造のユニフォーミティの規制の枠に因われることなく定義することができる[15]．このことは，新しい属性項目を定義する必要が生じた際にも，全てのモジュールに対してそれらの属性項目を再定義する必要がなく，フレキシビリティに富んだモジュールの構成を可能にする．以下にこれらのリスト処理を用いて効率的な生成を可能にするステートメント及びモジュールの構成についての説明を行なう．

３．３．１　ステートメントの構成

C 言語などのプログラミング言語の命令や付属の基本関数をステートメント，これらを組み合わせてユーザが作ったプログラム単位をモジュールと呼ぶ．ステートメントは，呼び出し文をシステム定義の一つの関数形に変換したもので，呼び出せばそのまま機能するのに対し，モジュールは，原則としてユーザ側が定義してプログラムモジュールを作成し，これを呼び出し文で呼び出して使用する．これらのステートメントやモジュールを適用するために，ステートメントやモジュールごとにステートメント述語，モジュール述語を作り，その引数部に辞書的情報を格納する．また，ステートメントやモジュールを検索するために，３．６節で述べるように非制御処理と制御処理別にステートメントやモジュールを一括して，見出し辞書を作成する．式(3-7)にステートメント述語の構成を示す．

```
statement([jp(JE,PE),arg_type([ARG_TYPE_LIST]),
           in([IN]),out([OUT])]).                               (3-7)
```

引数部には次のような情報を格納している．項 jp の引数 JE はこのステートメントの日本語表現などの呼び出し文で PE はその述語的表現であり，arg_type 項の引数リストは呼び出し文の中に含まれる変数の名前とデータ構造やタイプの組からなるリスト，IN と OUT はそれぞれこのステートメントの適用条件と適用後の出力データや出力状態を記述する．

```
statement([jp([プロンプトにより,OBJ,を読み込む],prompt_read(OBJ)),
          arg_type([name(OBJ),type(TYPE)]),
          in([]),out([read(OBJ)])].                    (3-7a)

statement([jp([if(T),then,S1,else,S2,end_if]),
          if_else(T,S1,S2)]).                          (3-7b)
```

式(3-7a),(3-7b)にステートメント述語の例を示す．式(3-7a)はキーボードからのプロンプト表示付きの読み込みの生成ステートメントであるが，マクロ命令として取扱い，入力要求表示と読み込みの２個のＣコードの生成を行う．また TYPE は変数で任意のものでよいことを表している．in の引数が空リストであるのは，キーボードから読み込むのに必要な前提条件が何も設けていないことを示している．式(3-7b)は式(3-7)の JE に分りやすさ，見やすさの点から日本語文でなく，常用の英語表現を当てている．T や S1 や S2 などの変数は個体変数でなく文を表すものであるので，タイプ指定や入出力条件の指定を行なっていない．

３．３．２　モジュールの構成

一般にモジュールは，次のような構成を持つ．

```
module(Head,Tail).                                     (3-8)
```

ここで Head や Tail の引数部はリスト形式を取る．Head はモジュールの機能の種別を表す呼び出し文項目や処理対象のデータ構造の項目などからなる頭部であり，Tail はリスト形式により不定長・不定数の属性項目からなる尾部である．尾部の項目としては，モジュールの関数の型と機能を日本語文で説明するコメント項目，この関数を C 言語プログラムでコールするときの関数表現の項目，引数部の処理対象の変数の型や，関数の戻り値の型を記述する対応項目，このモジュールを適用するのに必要な入力データやその条件を記述する入力項目，適用後，出力するデータの名前や性質を記述する出力項目，このモジュールを収納するファイル名項目などである．先ほども述べたように，これらの属性項目の記述は，リストを用いているため不定長でよく，また属性項目の種類はモジュールを通じて同じである必要はない．

次に一つの例として式(3-9)に１次元配列 OBJ にキーボードからデータを読み込むモジュールの構成を示す.

```
module([
        jp([プロンプトにより大きさ,N,の１次元配列１次元配列,
            OBJ,にデータを読み込む],
        prompt_read(OBJ)),                                    ①
        argl([[name(OBJ),type(TYPE),
        array([dim(1),size(N),max_size(NMAX)])]])             ②
       ],[
        funct([[int,[read1adi(OBJ,N)],
                float,[read1adf(OBJ,N)]]]),                   ③
        file_name([[int,'read1adi.c']
                   [float,'read1adf.c']])                     ④
        funct_type([]),
        in([read(N)]),                                        ⑤
        out([read([OBJ,N])])                                  ⑥
       ]).                                                          (3-9)
```

①はこのモジュールの見出し部を表す．すなわち，これは「読み込む」に関するモジュールであり，日本語による呼び出し文とこれに対応する述語形を表している．

②はこのモジュールにおいて処理に関連する対象の名前とデータ構造やタイプ名を示しており，このモジュールの適用条件をチェックするのに用いられる．

③の funct 部はこのモジュールがＣのソースコードに展開されたときの C の関数形及び引数を表している．関数の型に合わせて int の場合，float の場合のそれぞれに分け read1adi，read1adf のように別名を設けている．

④の file_name 部は③で表した C の関数形がどこのファイルに格納されているのかを表しており，このモジュールがリンクされる際，ヘッダ文にインクルードされる．

⑤の in([read(N)])はこのモジュールが適用される際の入力条件を表しており，このモジュールを適用する際，この条件に示されているとおり配列の大きさ N が予め与えられていなければならないことを示している．

⑥の out([read([OBJ,N])])はこのモジュールが適用される際の出力条件を表しており，このモジュールに実行によって，求められる出力や状態変化などの出力条件を表している．この例では，モジュールの実行によって大きさ N の１次元配列 OBJ にデータが読み込まれたことを示している．

ここで①②が頭部 Head であり，③④⑤⑥が尾部 Tail になっている．しかしながらこれはあ

くまで一例に過ぎず，リスト処理を施している関係で，特に尾部については必要に応じていくらでも拡張していくことができる．

３．４　ソースコードの生成

３．４．１　Cのソースコードへの展開

式(3-4),(3-6)が準備された後，以下の述語式(3-10)を実行することでCのソースコードの生成が行なわれる．

```
prog(spec(NUM)):-lang(c),objlist(spec(NUM)),
                    main_head,process(spec(NUM)),main_end.        (3-10)
```

ここで lang(c)は対象言語を指定するための述語であり，また，main_head,main_end は３．５節で述べるが，メイン関数の頭部を完成させるための述語である．

式(3-4)により，処理関連対象のデータ構造などの宣言部のほか，データタイプを手軽に参照するための述語を生成する．

次に，process 述語によりプログラムの処理部を作成する．処理設計文書の内容は process 述語の本体の proc_list 述語の引数部に式(3-11)のようなリスト形式[P1,P2,･･･,P*n*]で記録されている．

```
process(spec(NUM)):-proc_list([P1,P2,･･･,Pn]).                   (3-11)

proc_list([H|T]):-proc(H),proc_list(T).                          (3-12)

proc_list([]).                                                   (3-13)
```

これらの設計文書の内容は式(3-12)により，proc_list 上の設計文書ごとに一つずつ処理されて行く．

呼び出し文 X を引数に含む jp(X,Y)が式(3-14)に示すようにステートメント S の要素であり，かつ，このステートメントで処理する対象のデータタイプが，データ設計文書の obj タイプに一致して適用可能なステートメントであればXの述語形Yから述語c(Y)により後述のようにCの関数形の命令を生成する．

```
proc(X):-statements(S),member(jp(X,Y),S),
          member(arg_type([name(OBJ),type(TYPE)]),S),
          obj([name(OBJ),,type(TYPE)]),c(Y).                     (3-14)
```

呼び出し文 X を引数に含むステートメント述語へのアクセスに失敗すれば，式(3-15)のような述語によりモジュールを検索し，モジュール関数の呼び出し文を生成する．

```
proc(X):-
        module(H,T),member(jp(X,Y),H),                    ①
        member(arg_type([name(OBJ),type(TYPE),
                array([dim(D),size(N),max(NMAX)])]),H),   ②
        p_obj([name(OBJ),type(TYPE),
                array([dim(D),size(N),max(NMAX)])]),      ③
        member(funct(M),T),member([TYPE,PROC],M),         ④
                writelist(PROC),write(';'),nl,            ⑤
        member(file_name(FNL),T),member(TYPE,FN],FNL),
                include(FN),                              ⑥
       (member(funct_type([]),T);member(funct_type(FT),T),
        member([TYPE,FUNCT_NAME],FT),
        funct_type_decl([TYPE,FUNCT_NAME])).              ⑦    (3-15)
```

①において呼び出し文 X を引数に持つ項 jp(X,Y)が module の Head の要素であり，かつ，②③においてモジュールの arg_type がデータ設計文書から作成した p_obj の引数のタイプ TYPE と一致すれば，④⑤によりこのモジュールの Tail 部の funct の引数部 M の中で TYPE に対応して表記されている C での関数表現を書き出す．

また⑥により Tail 部の file_name の引数部 FNL の中で対象の TYPE に対応するファイル名 FN を include の中に記録する．また，⑦において Tail 部の funct_type の引数部 FT が[ ]でなく，宣言しなくてはならない場合には，対象の type に適合する関数名 FUNCT_NAME を登録する．

MAPP では制御命令の多くは，処理対象のデータタイプや構造のチェックをしない．この際は，式(3-16)によりタイプチェック無しに c(Y)を実行する．

```
proc(X):-statement(S),member(jp(X,Y),S),c(Y).                    (3-16)
```

また，タイプ不一致などにより，変換不可能な呼び出し文が現れた場合にメッセージを出すため，式(3-17)の規則を設け proc(X)を頭部に持つ規則の最後に配置する．

```
proc(X):-writelist([X,はプログラムに変換できません]).           (3-17)
```

設計文やその述語形にはC言語の関数表現作成に必要な対象の名前やタイプなどの情報を含

ませているため,これらの情報を用いれば,Cのステートメントやモジュールの関数表現はwrite述語を用いて比較的簡単に生成することができる.

設計文を述語形に変換すると，引数部の長さが不定の場合にもC言語への変換を一般的に簡単に表すことができて便利である．これに関連して，以下に述語形の設計文からC言語に変換する２，３の例を示す.

数個の文字列 S1,S2,･･･S*n* を与え文字列ごとに改行して印刷する処理の見出し文の述語形をprint_string_nl_list([S1,S2,･･･S*n*]).とするとき，これをC言語の命令に変換するには

```
c(print_string_nl_list([H|T]):-
        c(print_string(H)),c(print_string_list(T)).
c(print_string_nl_list([]).
c(print_string(S)):-writelist('printf("',S,'");']),nl.          (3-18)
```

のような述語形を定義すればよい.

同様に一般形の処理文を変数に持つ制御の C 言語への変換も，式(3-18)のように述語形を用いて簡潔に示すことができる.

```
c(CASE(INDEX_NAME,BODY)):-writelist(['swtch(',INDEX_NAME,'){']),nl,
        CASE_body(BODY),write('}'),nl.                          (3-19a)

CASE_body([item(NUM),H,end_CASE]|T):-
    writelist(['   CASE ',NUM,'   :']),
    proc_list(H),writelist(['    ','break;'],nl,CASE_body(T).  (3-19b)

CASE_body([]).                                                  (3-19c)
```

上記のスイッチ文(CASE 文）においては，body 部の場合の数が一般的には不定であり，処理内容も不定であるが，述語の再帰的定義表現を用いて式(3-18)のようにC への変換を簡潔に示すことができる．if_else_if 文などにおいても同様であり，不定個の場合の処理を，引数部に処理内容をトップダウン的に逐次指定することにより，目的言語表現に変換することができる.

３．４．２　リンクによるプログラムの部分生成

モジュール見出し表には式(3-9)の⑤⑥に示したように，モジュールの適用条件である入力項目 in(IN)，処理後の状態や出力データを記述する出力項目 out(OUT)を設けている．従って，これらの知識を用いることにより，モジュールを組み合わせて作ったプログラムの実行可能性に

関するチェックや，逆に入出力条件を与えこれを満たすプログラムを何個かのモジュールをリンクして生成することができる．ここでは後者について考え，入力データに関する入力条件 input(IN)と typep(TYPE)なる条件の下に出力条件 output(OUT)を満たすプログラムを生成する問題を考える．このためには次に示した出力述語 ouput(OUT)の本体部が真となるように，本体部の述語を逐次実行していけばよい.

```
output(OUT):-
     module(Head,Tail),member(out(OUT),Tail),          ①
     member(in(IN),Tail),input(IN),                    ②
     member(type(TYPE),Tail),typep(TYPE),              ③
     member(funct_type(FUNCTTYPE),Tail),               ④
     funct_declared(FUNCTTYPE),
    (memebr(para_obj(OBJ,type(OBJTYPE)),Tail),         ⑤
     para_declared_list(OBJ,OBJTYPE);none),
    (member(tmpvar_type(VARTYPE),Tail),                ⑥
     tmpvar_declared(VARTYPE);none),
    (member(file_name(FILE_NAME),Tail),                ⑦
     include(FILE_NAME);none),,
     member(funct(FUNCT),Tail),                        ⑧
    (state(STATE),member(FUNCT,STATE);                 ⑨
     nl,writelist(FUNCT),write(';'),                   ⑩
     member(com(COM),Tail),com_set(SET),               ⑪
     add_com_set(COM,SET),
     addstate(FUNCT,STATE)).                           ⑫    (3-20)
```

まず，モジュールの尾部 Tail の出力項目 out(OUT)の引数が仕様の出力条件 OUT と一致するモジュールを探索する.成功すればそのモジュールの入力項目 in(IN)やタイプ項目の引数 TYPE が，仕様で与えた入力述語 input(IN)の引数 IN やタイプ述語 typep(TYPE)の引数とそれぞれ一致するかどうかを調べる．一致すれば④～⑥で型宣言すべき関数やパラメータ変数，一時置換変数をモジュール尾部の各項目から type_member 述語の引数に登録し⑦でインクルードすべきファイルがあればこれを登録し，⑩で OUT を満たす手続きを与える関数表現を出力し，⑪でその関数表現に対する日本語文を注釈用にモジュール尾部の com 項目から com_set 述語の引数部にコメント用として収録する.

もし，このような入力データが含まれていなければ,

```
input(X):-output(X).                                   (3-21)
```

なる関係を用いて，X を出力項目に含むモジュールの探索を繰り返す．探索に失敗すればプログラム作成は失敗に終わる．他方，このような探索を繰り返して，仕様で与えられた入力条件を満たすモジュールに達すれば，これまでたどってきた探索の形路とは逆方向に初めの出力条件 OUT を含むモジュールに向かって，各モジュール記載の④以降の処理，すなわち，各種の変数や関数の型の登録，ファイルの登録，関数 FUNCT の出力を前述のように繰返し，MAPP はメイン関数のプログラムを作成する．なお，このモジュールの入出力条件のチェックは，自動補填に有効に働くばかりでなく，生成されるプログラムの正しさを検証するためにも利用可能である．

次に，以下の問題に対する MAPP による自動生成の例を示す.

＜問題の概要＞　大きさ n の 1 次元配列 a にキーボードよりデータを入力し，それらの配列要素の偏差，和，平均をそれぞれ出力印字する．その際，整数 n，配列 a は入力条件として与えている.

スペック

```
objlist:-obj(a,type(float),array(size(n)),
         obj(n,type(int)).

spec:-output(funct_var_printed(dev([[a,n]]))),
      output(funct_var_printed(sum([[a,n]]))),
      output(funct_var_printed(av([[a,n]]))).
```

自動生成された C のメインプログラム

```
#include    "dev1adm.c"
#include    "av1adm.c"
#include    "sum1adf.c"
#include    "read1adf.c"
main(){
    float   dev_a,dev_1ar_f(),av_a,av_1ar_f(),sum_a,sum_1ar_f(),a[];
    int     read_1ar_df(),n;
    printf("n=");
    scanf("%d", &n);
    read_1ar_df(a,n,"a");
    sum_a=sum_1ar_f(a,n);
    av_a=av_1ar_dm(sum_a,n);
    dev_a=dev_1ar_dm(a,n,av_a);
    printf("dev_a=%f", dev_a);
    printf("sum_a=%f", sum_a);
    printf("av_a=%f", av_a);
}
```

図 3-4　スペックで示した仕様から，MAPP において自動生成された C のメインプログラム

ここは，配列のデータを処理する 2,3 のモジュールをリンクする他，変数 n のデータをキーボードから読み込むマクロ命令を自動的にプログラムに取り込んでいる.また,式(3-20)により，平均を求めるモジュールのリンクの際には変数 n の読み込みを自動的に補填したり，その際行なった合計及び平均の計算を，その後に要求されたそれぞれの仕様（合計の計算，平均の計算）に対して用い，一旦行なわれた計算は，できるだけ重複しないようにし，モジュールの再利用はもちろん，内部計算処理においてもその再利用を行なっている.

３．５　メイン関数頭部の作成

前節では処理仕様から仕様の詳細化やプログラムへの変換について述べたが，C プログラムの場合，初めにインクルード文やメイン関数などの関数記号部，型宣言部などを作成しなければならない．MAPP はこれらを処理対象に関する仕様や，検索したモジュールに含まれる情報を用いて，式(3-22)の述語 main_head により自動的に作成する.

```
main_head:-(include([]);include(FILE),include_sent_gen(FILE)),
            main_header,type_decl.                              (3-22)
```

右辺の本体部はインクルード述語の引数部が空であれば何もせずに次の述語 main_header へ行き，そうでない時には引数部の FILE をインクルードする文を include_sent_gen(FILE)という述語で作成し，その後述語 main_header で main()などを作り，述語 type_decl でタイプ宣言部を作ることを示す.

プログラムへの変換時に，適用するライブラリモジュール関数を含むファイルのファイル名が，まだ include(OLD_FILE)の引数部に含まれていなければ，次のような述語 include(FILE)によりこれを述語 include の引数部に加える.

```
included(FILE):-((include(OLD_FILE),member(FILE,OLD_FILE));
                  add_file(FILE,OLD_FILE)).                     (3-23)
```

main などの関数記号部は次の述語 main_header により作成される.

```
main_header:-main_arg(ARG),writelist(['main(',ARG,')']),nl,
              write('{'),nl.                                    (3-24)

main_arg([]).                                                   (3-25)
```

main 関数記号部に続く型宣言部は，処理対象に置けるデータ構造の仕様 obj(X)や，検索した

モジュールから関連情報を type_member 述語の引数部に抽出保持した情報を type_decl 述語により書き出して作成する.

まず図 3-4 の objlist 述語により各 obj のデータ構造情報を呼び出し，次に obj 述語により，引数部に記述した型に関する情報を type_member 述語の引数部などに移してタイプ宣言文作成の準備をする.

```
obj([name(NAME),type(TYPE),array([dim(D),size(S),max(M),]),val(VAL)]):-
      assert(p_obj([name(NAME),type(TYPE),
      array([dim(D),size(S),max(M)])]),                    ①
      type_member(TYPE,MEMBER),                            ②
      (member(NAME,MEMBER);
       add_type_entity([NAME,val(VAL)],TYPE,MEMBER)).      ③    (3-26)
```

式(3-26)は処理対象が配列でかつ初期値を持つ場合の,データ構造仕様の読み込みに置ける処理ルールである．処理対象がスカラであったり，初期値が指定されていない場合に対しては，array や val の項目を省略した処理ルールを設けている.

さて，処理対象の型やデータ構造の設計文書は一つの設計課題に対して図 3-4 のように見出しとして一つの objlist でまとめ，その本体部にいくつかの obj 述語を並べて記述している．従って，式(3-26)の obj 述語では仕様のデータ構造に関する情報を直接参照するのは不便である．このために，式(3-26)の obj 述語を用いて，図 3-4 の objlist 述語の本体部から各処理対象のデータ構造を取り込み，式(3-26)の①によりこれを assert 文で p_obj 述語の引数部に記憶し，演算や入出力命令の作成における処理対象の型名などの作成やチェックに対処している.

②の type_member 部ではこの処理対象が，その TYPE の MEMBER に登録されているかどうかを調べ，続いて③において登録されていなければ，[NAME,val(VAL)]を add_type_entity 述語によりその TYPE の MEMEBR に加える.

この他，MAPP は適用するモジュール述語から関数値や一時変数などの各種変数の名前やタイプを type_member に登録収納した後，次の type_decl 述語によりＣの型宣言の様式に従って印字する.

```
type_decl:-
    (type_member(int,[]);type_member(int,INT),,
     write('int    '),write_namelist_ic(INT),write(';'),nl).   (3-27)
```

式(3-27)はデータが整数型の場合のプログラム作成部分で，この型に属するメンバーが[ ]で変数が存在しないときには，何も印刷せず，そうでないときには，引数部の INT の変数のリストを，write_memberlist_ic(INT)述語により，要素間に ‘,’ を挟んで write_name により印字する.

write_name 述語は対象のスカラや配列の別，初期値の有無などにより異なる型宣言を出力するように設定してあり，MAPP は対象のデータ構造や初期値の有無などに従って，設定した型宣言部を出力する.

次に，write_name 述語の中で，１次元配列で大きさと初期値を指定した宣言を出力するためのプログラム作成部分を示す.

```
write_name(NAME,max(M),val(VAL)):-nl,write('          '),
        writelist([NAME,'[',M,'[={']),
        writelist_ic(VAL),write('}'),nl.                    (3-28)
```

ただし writelist(X)は引数 X のリスト要素を並べて印字し，writelist_ic(X)は，間に‘,’を挟んでリスト要素を印字するための規則である.

以上述べてきたように，図 3-4 のスペックで示した仕様を整備することで，前述した数々のメリットを受けプログラムの生成を行なうことができる．しかしながら，日本語文を仕様の中に用いることによりいくらかは読み易い形になっているとはいえ，この記述を行なうことを考えると，決して扱い易いものとは言い難い.

そこで今までの経緯をたどりつつ，ユーザ側に立った観点から，できるだけ扱い易い形で図 3-4 のスペックで示したような仕様を整備できるよう，メニュー選択方式を採用し，関連する項目に関する機能の一覧表から必要なモジュールを選択していく方法について次節で述べる.

３．６　設計文書から構造化図の自動作成

CASE（Computer Aided System Engineering）[13]においてはプログラム設計を構造化図上で行なうことで，作業効率を向上させさらに開発後の保守作業においても大幅な作業効率の向上をもたらした．そこで MAPP においてもこの方法を採用し，設計作業を行ないながら同時にその構造化図を表示させ，でき上がった手続き文からプログラムの自動生成を行なうことを考えた.

構造化図を考える上で，構造化の概念は必須項目である．構造化とはジャンプ命令の安易な多用によるソースコードの繁雑化を防ぐために，処理をできるだけブロックごとに固め，一見して全体の処理の流れがわかるようにすることである．シリアルな処理の場合は非常に簡単であるが，複雑に絡み合った制御構造を含む処理を理解する上で特に構造化の考え方が役に立ってくる．よって構造化図を取り入れるためには制御構造を MAPP で扱えることが必要になってくる．そこで次には，制御構造を含ませるために行なったいくつかの拡張とその中での非制御・制御両方の設計文書の作成について説明を行なう.

３．６．１　非制御文の場合

処理の呼び出し文を検索するために見出し辞書を作成する．呼び出し文を処理機能別に分類し，"読み込む", "read"のような日本語や英語の見出しのキーワードを設けて入力し，このキーワードをもつ呼び出し文の一覧を表示させる．ユーザは適用したい呼び出し文を番号で指定して利用する．これらの辞書の書式を式(3-29)に示す．ここにおける辞書は，式(3-6)を拡張した形で構成されており，辞書の情報は式(3-4)の内容 process_*n* を引数部で与えている．

```
dict1([
  [head([J_KEY1,E_KEY1]),
    body([p(1,V11,S11),...,p(n1,V1n1,S1n1)]),
  [head([J_KEY2,E_KEY2]),
    body([p(1,V21,S21),...,p(n2,V2n2,S2n2)]),
    ..........................])  ]).                    (3-29)
```

式(3-29)の各 J_KEY*i*,E_KEY*i* (*i*=1,2,･･･)はそれぞれ日本語,英語などの１バイト形，２バイト形によらず関連するキーワードを表す．これらのキーワードはリスト上に記述されているため関連すると思われる複数の項目を指定すれば曖昧なキーワードからも目的の処理を探索することができる．

ただし，これらのキーワードは式(3-4)で説明した場合と少し異なり，キーワードに複数の関連する項目を持つキーを設けることはできない．これは，後述する図 3-7 のような辞書を設けた際，必要な処理に対する文番号が定まらなくなってしまうためである．

body 部の引数部にこの見出しキーに属する呼び出し文の p の組を与えている．各組は

```
p(NUM,V,S)                                              (3-30)
```

の項をもっている.式(3-30)の p の引数の中,NUM は呼び出し文の選択のための番号を表わす.S は日本語文などによる呼び出し文，V は S に含まれるカスタマイズ用の変数リストである.呼び出し文はリスト形で書き，カスタマイズ用の変数の前後をコンマで区切る.

図 3-5 に見出し辞書の実例を示す. 図 3-5(a)には"書き出す","読み込む"などの非制御呼び出し文に対する辞書 dict1 の一部を示し, 図 3-5(b)には選択などの制御呼び出し文に対する辞書 dict2 の一部を示している.

```
dict1([head(書き出す,print),
   body([p(1,X,[X,をプリントする]),
         p(2,X,[１次元配列,X,をプリントする]),
         p(3,X,[２次元配列,X,をプリントする]),
         ·····························])],
  [head(読み込む,read),
  body([·····························])],
       ·····························
```

(a)非制御呼び出し文の場合

```
dict2([head(もし,if),
   body([p(1,T,if(T),c([if(T),then,b(S),end_if])),
         p(2,T,if_else(T),c([if(T),then,b(S1),else,b(S2),end_if])),
         p(3,OBJ,when(OBJ),c([when(OBJ),b(SP),end_when_list])),
         p(4,INDEX_LABEL,CASE(INDEX_LABEL),
              c([CASE(INDEX_LABEL),b(SP),end_CASE_list])),...
```

(b)　制御呼び出し文の場合

図 3-5　見出し辞書の一部

求める処理機能の種類の呼び出し文を検索して処理設計文に加えるには，式(3-31)のadd_spec(X)述語を用いる．

```
add_sp(X):-
  (retr(X,CUR_SP),out_sp(CUR_SP);                    ①
  retrc(X,XC),spc(XC,CUR_SP)),                       ②
  sp_all(SP_ALL),                                    ③
  append(SP_ALL,[CUR_SP],N_SP_ALL),                  ④
  retract(sp_all(SP_ALL)),                           ⑤
  assert(sp_all(N_SP_ALL)),...                       ⑥    (3-31)
```

所要の処理のカテゴリキーX を引数として述語 add_sp(X)を入力する．すると，X が非制御関連のキーか制御関係の手続きのキーかによって,retr 述語かretrc 述語が作動して図 3-5 の dict1, dict2 のような辞書の中のキーX に属する呼び出し文の組が表示される．利用者は，この呼び出し文の組から所要のものを以下のように選定する．

非制御モジュールや命令の呼び出し文の場合，式(3-31)の①で式(3-32)のようなretr(X,CUR_SP)述語によりキーX に属する呼び出し文一覧を表示させる．

```
retr(X,STX):-
  dict1(DICT1),
  member([head(H1),body(T)],DICT1),member(X,H1),
  writelist_nl(T),                                    ①
  write('適用しようとする命令やモジュールが
        なければ０を
        あれば番号を入力して下さい. '),
  read(N),
  (N=:=0,break;                                       ②
  member(p(N,V,ST),T),nl,
  writelist_nl([ST,V]),nl,                            ③
  writelist_nl([変数列,V,を決めて下さい]),
  sp_var(V,SV),qsp(H2,V,SV,STX)).                     ④    (3-32)
```

ここに式(3-32)の①の writelist_nl(T)はリスト T をリスト要素ごとに改行して表示する述語で，head 部に見出しキーX をもつ項の tail 部の呼び出し文を表示する．希望する呼び出し文がないときには0 を入力すれば②により停止する．そして他の基本的な呼び出し文を用いた設計へ移行する．ここに';'は「または］という意味の OR 論理記号である．

③でキーボードから入力した番号の呼び出し文を表示した後，④の述語 sp_var(V,SV)により，プログラムで用いる変数名 V を指定する入力プロンプトを表示する．利用者が変数名リストをキーボードから SV に入力すると，式(3-32)の④の qsp 述語はカスタマイズした呼び出し文 STX を出力する．

このようにして式(3-31)の add_sp 述語は，out_sp 述語により，カスタマイズした呼び出し文 CUR_SP を画面に表示する．つづいてこの呼び出し文を③の sp_all 述語の引数部のこれまで作成した設計文 SP_ALL に④により追加し，⑤⑥で SP_ALL を更新する．

３．６．２　制御文の場合

制御文の呼び出し文は非制御文と同様に英語文などの一つの文の形をとるが，処理本体部のカスタマイズ用変数は一般に呼び出し文の集まりである．ここでは制御フレームへ呼び出し文をトップダウン的にはめ込むことにより制御文の呼び出し文を作成する．

制御の呼び出し文は一般に条件部と処理本体部とからなる．図 3-5(b)の dict2(X)に制御関係の関数の呼び出し文の一部を示す．body 部の項 p は

```
p(NUM,T,COND,S)                                       (3-33)
```

の形の 4 個の引数をもっている．NUM は呼び出し文番号，T は条件式，COND は呼び出し文の条件部，S は呼び出し文の全体である．条件式は，ここでは，簡単のため文字列として取り扱う．先ず始めに式(3-31)の②の retrc 述語により制御文の種類の番号と条件式を指定すると，MAPP は条件部 COND を作成する．

条件部の指定を終わると，引き続き「本体部の指定をして下さい」という spc(XC,CUR_SP) のプロンプトにより,利用者は処理部の指定を行う.本体部は例えばif文の処理部が１個,if_else 文の処理部が２個,CASE 文の処理部が２個以上というように制御文の種類により異なるので,制御文の種類ごとに spc 述語を設けている．

式(3-34)は図 3-5(b)に示すように C 言語のスイッチ文に対する spc 述語である．式(3-31)の add_sp(if)を入力して，②の retrc 述語により選択関係の呼び出し文の一覧を表示させ，呼び出し文番号 4 と条件部のインデクス変数名を指定した後，spc 述語によりインデクス変数の取る指標値リストを場合の数だけ式(3-34)の①に従って P に入力する．指標値リストは②により先頭のものから一つずつ取り出され，式(3-36)の CASE 述語に従って各場合の処理の指定が行われる．

```
spc(CASE(INDEX_NAME),
    c([CASE(INDEX_NAME),b(SP),end_CASE_list]) ):-
    write(CASE(INDEX_NAME)),nl,
    write('CASE 文の本体部の指標値リストを入力して
          下さい. '),nl,
    read(P),                                          ①
    CASE_list(P,SP),                                  ②
    out_sp(c([ CASE(INDEX_NAME),b(SP),
               end_CASE_list] )),nl).                 ③    (3-34)

CASE_list([H|T],[HS|TS]):-
   CASE(H,HS),CASE_list(T,TS).                              (3-35)
   CASE_list([],[]).
```

式(3-36)の CASE 述語の主要部は②の spc_list 述語である．すなわちスイッチ文のそれぞれの一つの場合において，順次に行う処理の一覧を見出しキーリストの形で(3-36)の①により P に入力し，続いて②により式(3-37)に従って順次，retrc 述語や retr 述語を再帰的に用いて，処理のそれぞれについて呼び出し文を適用し，設計をトップダウン的・組織的に進める．

```
CASE(ITEM,c([item(ITEM,b(SP),end_CASE])):-
  writelist(['CASE 文',ITEM,' に対する処理の
        見出しキーリストを入力して下さい']),nl,
  read(P),                                          ①
  spc_list(P,SP),                                   ②
  write(c([item(ITEM),b(SP),end_CASE])),nl,
    ..........................                          (3-36)

spc_list([H|T],[HS|TS]):-
   (retrc(H,HC),spc(HC,HS);
    retr(H,HS)),spc_list(T,TS).                         (3-37)
spc_list([],[]).
```

以上はスイッチ文を用いた設計の場合であるが，どの制御文も，処理の本体部は，spc 述語により，各枝ごとに処理の最上位のキーリスト[K1,K2,...,K*n*]の指定を要求する．ユーザは例えば[if,read,print]を処理のキーとして入力する．spc 述語は最初のキー'if'に属する命令やモジュールの呼び出し文の表示を行い，ユーザはその中から適当な呼び出し文を指定する．呼び出し文が制御文のような場合には必要に応じて第２レベルの処理部の設計に移る．最初のキーK1 に対する設計指定が終わると第 2 のキーK2 に対する設計に移る．

設計の進行にともない，式(3-33)の第 4 引数部 S の見出し文の諸パラメータは指定されたものから逐次埋められていく．なお，図 3-5(b)のように第 4 引数には c や b の記号を付ける．c はその引数部が制御文であること，b はその引数部が処理本体部のあるまとまった部分であることを構造化図作成ルーチンやプログラムへの変換ルーチンに指示する．このように利用者は制御構造明示のための枠組みを準備することにより，引数部に変数列を入力するだけで複雑な制御構造の設計文書も簡単に作成し，３．７節に述べるが，構造化図上においても，設計文書をチェックすることができる．

なお，述語 sp_all(X)の引数 X に記憶した処理設計文書は，文書番号を付けてファイルに保存する．

３．７　構造化図の作成

プログラムの構造は構造化図の方が構造化文より見易く，このため構造化図によるプログラムの作成や表示が各方面で研究開発されている[16]．この節では構造化のための制御ステートメントを if,then などのタイプ情報を用いて見出し文にレベル線や制御マークを付けて構造化図を自動的に描き，視覚性を増加する一つの手法について述べる．構造化図には視覚性をよくするために機能やタイプ別にいろいろな形の枠を用い，この中に文字情報を押し込めるものもあ

るが，文字情報の長さの制限が強いので，SPD 図に準拠して制御の種類の概要は特殊記号を組み合わせて表し，文字情報は横線の上側や右端の所定の位置におく手法をとっている．

各プログラム単位はレベル 0 のステートメント start と end で自動的にはさむ．呼び出し文は前の呼び出し文から，縦線分を指定した個数だけ隔てて下方に書くが，仕様文の文頭に制御を表す語（if,then,else,while,for など）が現れた場合には，次の仕様文のレベルを 1 レベル増し，文頭を 1 レベル右に移動して書く．

呼び出し文の集合である設計文書のリストは述語 pic_sp 述語の引数部に与えられ，各呼び出し文は(3-38)のように statement_p 述語により処理される．

```
pic_sp([H|T]):-statement_p(H),pic_sp(T).
pic_sp([]).                                                    (3-38)
```

処理 P が非制御処理の場合には，(3-39)の①に示すように，これ以前に then,else,CASE などの語が出て，P を書く接続方向を示す述語が con(right)であるときには，P を右方向に書き次に接続方向述語を con(down)として次に書く方向を下方向に設定する．接続方向が右方向でなく下方向であった場合には，②に示すように，write_level_line 述語により第 0 レベルから現在の第 L レベルまで L+1 個のレベル線を下方向に 2 単位伸ばした後，(3-40)の arg 述語により第 L レベルの右側に P を書く．

```
statement_p(P):-level(L),space(IS),
  ( con(right),arg(P),retract(con(right)),
              assert(con(down) ) ;                    ①
   write_level_line(L),nl,write_level_line(L),arg(P)).   ②   (3-39)

arg(P):-writelist(['-- :',P]),nl.                          (3-40)
```

また，制御文に関しては，そのテスト条件部などの図的表現のためにそれぞれ述語を設ける．(3-41)は if 文の場合で，write_symbol 述語で第 1 引数に示す'-'記号を第 2 引数に示す回数の 3 回分，繰り返し書き，レベルを 1 だけ増して level 述語の引数に記憶し，その右に分岐を表す記号'<>'と条件文 T を書く．

```
arg(if(T)):-
       write_symbol('-',3),
       level(L),L1 is L+1,retract(level(L)),assert(level(L1)),
       writelist(['◇---(IF:',T,')']),nl.                   (3-41)
```

if 文の場合，then や else によりさらにレベルが 1 レベル増すので，end_if が現れたときには，レベルを 2 レベル減少させる．繰り返し文の場合も同様で繰り返しを表す記号として ( ) などの記号を文頭に置き，右端に繰り返しの種類と条件を文字列で表して簡単であるが視認性の向上を図っている．

構造化図はディスプレイ上に適宜表示させ，これまでに作成した設計文書の構造を視覚的に検討しながら作成を進めることができる．図 3-6 にこのような方法で作成した構造化図を示す．

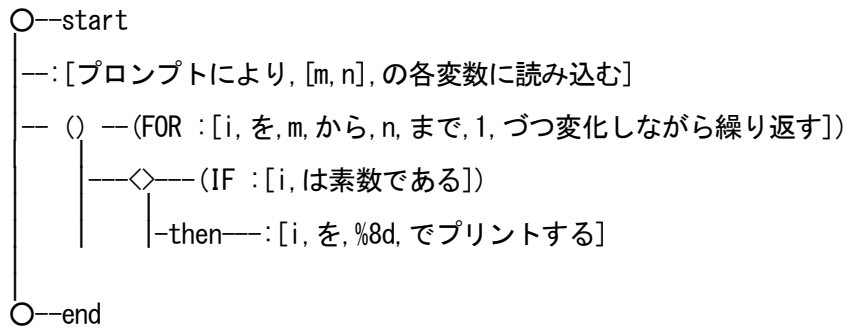


図 3-6　構造化図の例

ファイル FN の設計文書の spec(NUM)の内容を構造化図により図示するにはプロンプトに続き，draw_pic_sp(NUM,FN).を入力する．

```
draw_pic_sp(NUM,FN):-[FN],sp_rec(NUM,X),tell('pic.pl'),
      writelist([pic(NUM),':-pic_sp([start,']),expand_list(X),
      write('end ]).'),told,['pic.pl'],pic(NUM).                (3-42)
```

上式はファイル FN の sp_rec 述語の引数部の処理設計文リスト X を expand_list 述語で整形処理した後，その前後に start と end を付け，ファイル pic.pl に書き込み，これを読み出して，pic(NUM)のタグにより(3-38)の pic_sp 述語の引数部を構造化図として図示するものである．

３．８　MAPP における実験

前節までで述べてきた手法を用い，データと処理に関する簡単な設計文書から，C プログラムを生成する実験例を示す．表 3-1, 3-2 は実験設備を示し，図 3-7 は実験例における設計文書を示している．また，図 3-8 では設計時に自動的に展開される構造化図を示す．図 3-9 は図 3-7 で設計された仕様から生成された C プログラムを示している．

［実験設備］

表 3-1　機器構成

機器名	メーカー名	型番
NEWS ワークステーション	SONY	WS-1750
17 インチカラーディスプレィ	SONY	WP-513

表 3-2　ソフトウェア構成

ソフトウェア	型番等
OS	NEWS-OS Release 4.1C
X ウィンドウシステム	X11R2
Prolog インタプリタ	C-Prolog V1.5+
手続文書設計	MAPP：付録 C（C-1： 975 Step)
C プログラム生成	MAPP：付録 C（C-2：1610 Step)

※ 但し，開発環境として NEC PC-9801 NS/E，NEC MS-DOS V5.0
岩崎技研 Prolog KABA および MEGASOFT MIFES V5 を使用している.

※ 付録 C に MAPP のプログラムリストを載せているが，Step 数の記載は
あくまで参考である.

（実験例）ｎの値を読み込み０からｎ－１までの整数ｉの２乗の和を求めプリントする.

```
obj_list(spec(1)):-
obj([name(sum),type(int),val(0),rem([])]),
obj([name(n),type(int),rem([])]).

process(spec(1)):-proc_list([
[プロンプトにより,n,に読み込む],
[for([i,を,0,から,minus(n,1),まで,1,づつ変化しながら繰り返す]),
[compute(sum=sum+i*i)],
end_for],
[sum,を,d,でプリントする] ]).
```

図 3-7　データ構造仕様および処理仕様

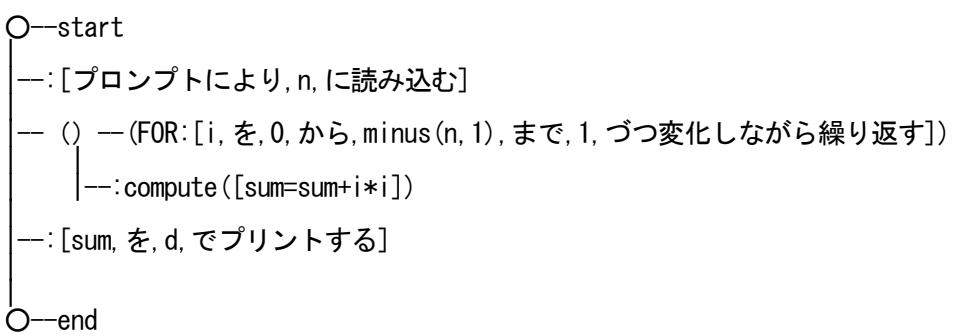


図 3-8　設計時において表示される構造化図

```
#include <stdio.h>
main( )
  {
      int     sum=0,n,i;
      printf("n=");
      scanf("%d",&n);
      for(i=0;i<n;i+=1){
          sum=sum+i*i;
      }
      printf("sum=%d",sum);
  }
```

図 3-9　自動生成されたＣのソースプログラム

実験例においては，MAPP により図 3-7 から図 3-9 で示したようにプログラムの自動生成を行っている．図 3-7 で示す概略仕様には，データ構造仕様および処理仕様が含まれ，ここではあらかじめ必要と思われるデータを定義しておく必要があるが，MAPP ではプログラム中で必要になってくる変数等は，可能な範囲において生成が行われる際補填される．例えば for ループはカウンタとして *i* を使用している．しかし，*i* はループ上で用いられるだけのロジック上考慮に入れる必要のない一時変数であり，ここでは for 文をリンクする際に入力条件として宣言部ヘッダのなかに自動的にインクルードされているため，実際のコード展開時において宣言文に補填されているのが図 3-9 において確認できる．もちろん for 分以外にも if 文や CASE 文など様々な制御構造を扱える．一方，CASE ツールなどコードの一部生成を機能として持つアプリケーションにおいては，通常，扱える制御構造の階層数に制限があるが，MAPP ではリスト構造を用いることにより，メモリが許す範囲で何階層にでもネストさせることが可能であり，理論的には制御構造の階層数に制限は無い．さらに制御構造の内部は，全体の構造化図を表示するウィンドウとは別のウィンドウを設けて表示させているため，設計文が長くなってきても見やすさを保っている．なお，付録 B に MAPP 操作時の表示画面を示しておく．

３．９　リスト処理を用いた入出力条件における課題

上述したように，プログラムを自動補填しながら自動生成し，なおかつ，構造化図も表示を行うことができた．しかしながら，入出力条件のチェックはおのおののモジュールごとに入出力条件を設け，モジュールがリンクする際，その入力条件がすでに存在するという前提で行ってきている．

確かに，モジュールの適用条件として，その前提となる入力条件をもつ他のすべてのモジュールを事前に，もしくは同時に適用するこの方法は，小さなプログラムを生成する場合は十分有効であると思われるが，規模の大きなプログラムを生成していく場合，条件が複雑に絡み合うことで，特に自動補填を行った場合等，予期せぬ障害を生じる可能性が考えられる．

そこで，入出力条件のチェック法において，数学的な解析が可能なペトリネットを用いることを考え，このペトリネットによって記述されたモデルの可達性を数学的に検証することで，同じくシステムの実行可能性を検証する．

第４章　ペトリネットを用いたプログラム自動生成における入出力条件のチェック

第 3 章において，予め用意してあるモジュールを組み合わせ，自然言語を用いた概略仕様から目的のプログラムを生成する手法について説明を行い，その実験システムである MAPP におけるいくつかの例題を扱った．MAPP における入出力条件のチェックについては，各モジュールごとに持たせた入力条件がそれぞれ満たされているかどうかにより行い，モジュールの適用後はその出力条件を条件群に加える方法により行っていた. MAPP では入出力条件のチェックを行う際，自動補填とプログラムのチェックを実現している．このようなモジュールの入出力条件のチェックを行うことは，生成されたプログラムの実行可能性を検証する上でも重要なことであるが, より厳密な入出力条件の検証を行うため, これら入出力条件のチェックについて，第 2 章で述べたペトリネットにおける可達判定条件を用いて，数学的な検証を与えることができないか検討を行う．

４．１　MAPP における入出力条件のチェック法

まず，従来より MAPP において行われていたチェック法について説明を行う．MAPP では，３．４．２項で述べた式（3-20）および（3-21）により以下に示すような入出力条件のチェックや自動補填を行っている．

具体的には，図 4-1 で示すように，予め条件を蓄えてあるリスト（apply conditions）を準備しておき，その上で，この場合モジュール function_i が適用される際，この入力条件（input condition）が apply conditions の中に含まれているかどうかをチェックする．そして，条件が含まれていたら，function_i を目的のプログラムに対し適用（リンク）する．それと同時に，function_i の出力条件（output condition）を apply conditions に追加する．次にモジュール function_n がリンクされるとし，同様に function_n の input condition が apply conditions の中に既に含まれているかチェックされるが，今回の場合，function_i の持つ output condition と function_n の input condition が同じであり，なおかつ，function_i はすでに適用済みのため，function_n の input condition は満たされていることになり，function_n もリンクされる．それと同時に，function_n の output condition も apply conditions に追加される．さらに，モジュール function_n+1 がリンクされるとすると， function_n+1 の input condition は 今と同様に，function_n により apply conditions の中に既に含まれているため, function_n+1 もリンクされる．このように，出来上がったプログラムにおけるすべてのモジュールの input condition は満たされている状態でプログラムが生成される．

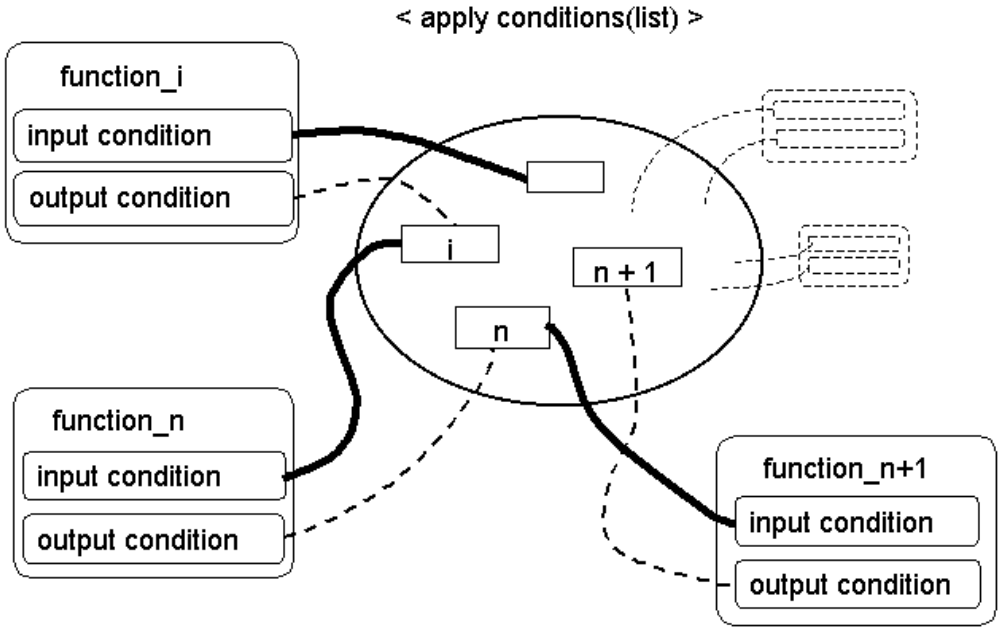


図 4-1　入出力条件のチェックにおけるイメージ図

次に，自動補填の考え方について図 4-2 を用いて説明する．

今，概略仕様（Specification）には，Function_A の記述しかなかったとする．MAPP は対応するモジュールの Module_A をモジュール群の中から検索し，これを目的のプログラム（Generated Source Code）に適用しようとする．その際，前述のように，この Module_A の入力条件がすでに満たされているかどうかチェックを行う．（具体的には，Module_A の入力条件が apply conditions の中に含まれているかどうか確認する．）しかしながら，もし Module_A の入力条件が満たされていなかった場合は，この Module_A の入力条件と同じ条件を出力条件として持つ，別の Module （この場合 Module_A’ ）をモジュール群の中から検索し，たとえ概略仕様に Module_A’ に対応する記述がなくとも，MAPP 自身によりこの Module_A’ の適用（リンク）を Module_A の適用前に試みる．さらに，Module_A’ を適用することにより，新たな入力条件が必要となってきた場合は，さらに，その条件を満たすための別のモジュール（この場合は Module_A” ）を検索し，先ほどと同様に，Module_A’ の適用前に，この Module_A” を適用しよう試みる．

これらの試みを繰り返し，最終的には概略仕様に記述がなくとも条件を満たすために必要なモジュールを自動的に補填し，すべての条件が満たされた形で目的のプログラム（Generated Source Code ）が生成される．もし，条件が最終的に満たされない場合にはこの生成は失敗に終わる．

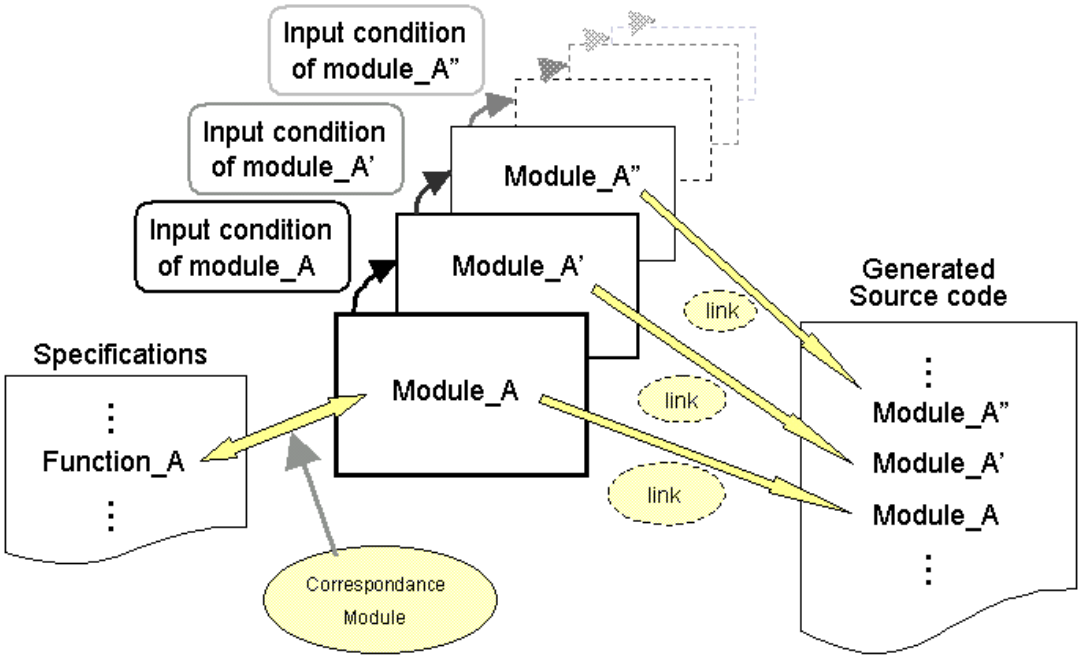


図 4-2　モジュールの自動補填のイメージ図

４．２　ペトリネットによる入手力条件チェック法

ここで，ペトリネットを用いた入出力条件のチェックについて検証を行う．まず，MAPP における自動化にはまだ組み込まれていないため，MAPP によって生成されるプログラムを一旦ペトリネットでモデル化し[17]，このモデルを用いて検証を行う．モデル化されたペトリネットの可達性を検証することが，プログラムの実行可能性を検証することと同じであることを，次の例題をとおして考えてみる．

今，簡単のために，以下の問題を考える．

［問　題］

整数型変数 n にプロンプトから値を入力し，入力された値を標準出力（ディスプレイ）に表示させる．

この問題を考えた時，処理を以下に示す２つの部分に分けることがでる．

ａ）整数型変数 n にプロンプトから値を入力する

ｂ）入力された値を標準出力（ディスプレイ）に表示する．

ここで，問題におけるプログラムおよび対応するペトリネットは，図 4-3 のようになる．

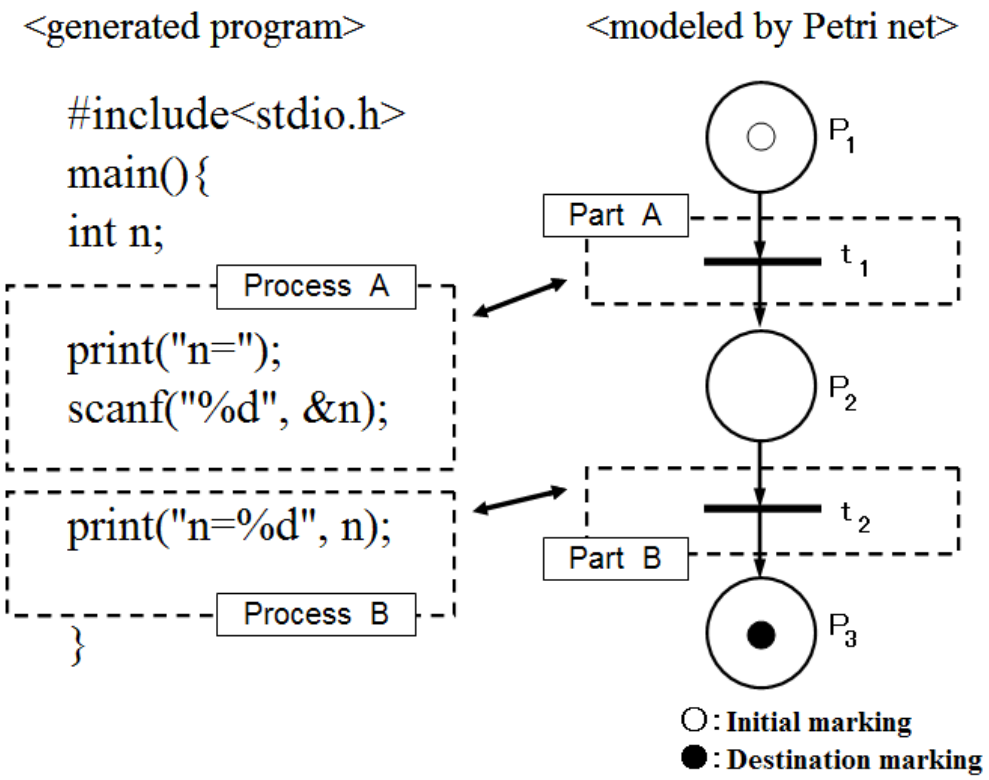


図 4-3 例題におけるプログラムおよびペトリネット

また，ここにおいて破線で囲まれた Process A および Process B は，上で示したａ），ｂ）２つの部分にそれぞれ対応している．

ここで，これらのプログラムとペトリネットの動作の対応を考えると，以下のようになる．

図 4-4 で示すように，最初プログラムは初期状態として start の部分にあり，また，ペトリネットでは，p_1 にトークンが存在している Initial marking : M_0 ＝［　1　　0　　0　］^{T} を持っている．

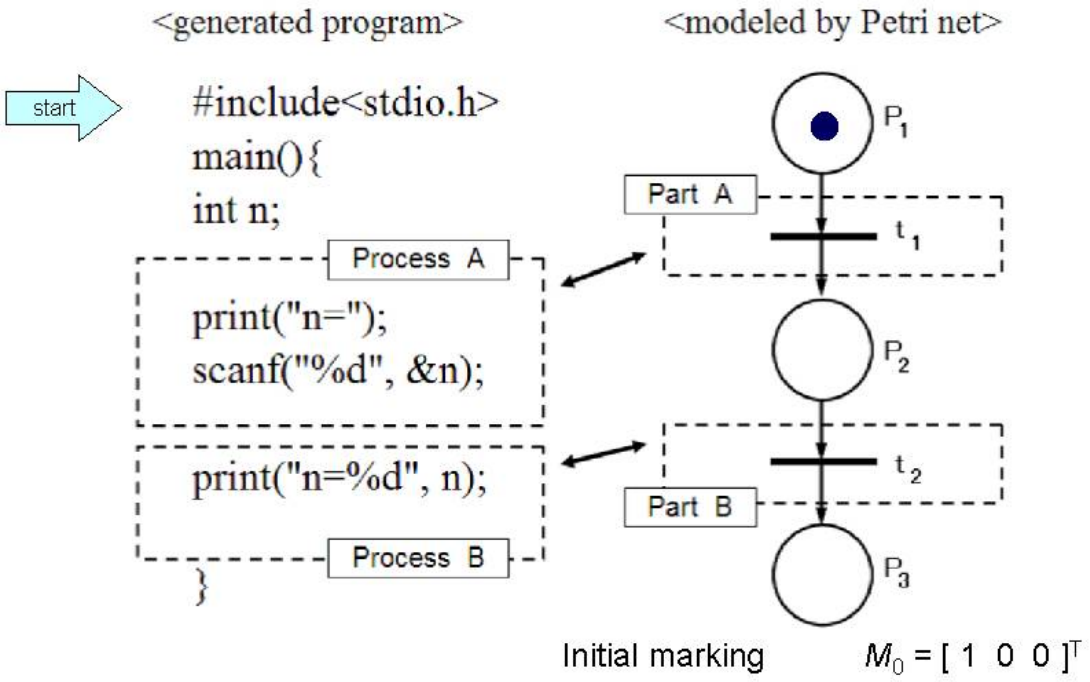


図 4-4　プログラムとペトリネットの対応（その１）

次に図 4-5 で示すように，プログラムは Process A の部分が実行される．ここでペトリネットにおいては，t_1 が発火（fire）される．つまり，Process A と Part A のそれぞれが対応をしている．

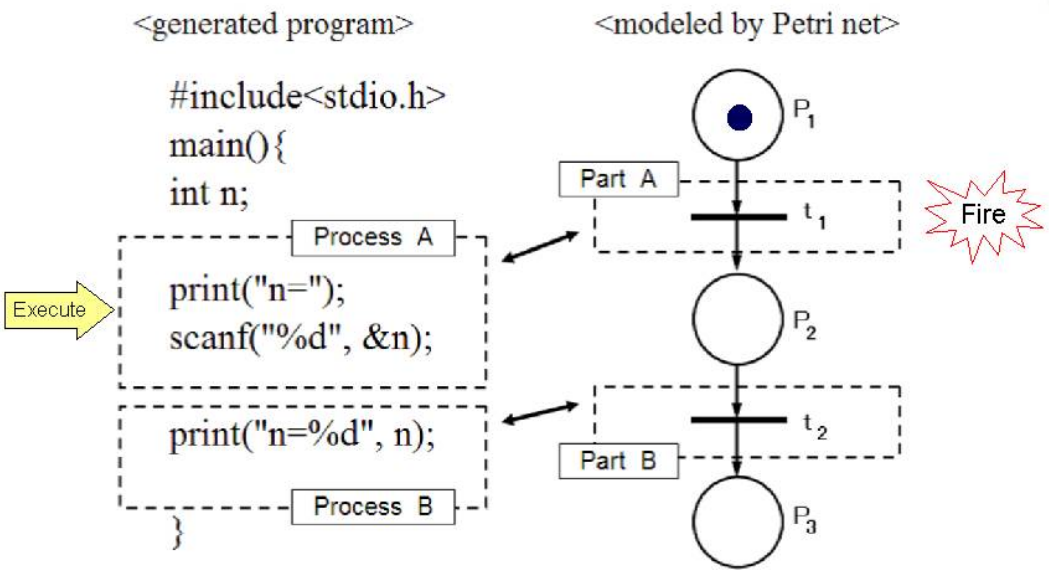


図 4-5　プログラムとペトリネットの対応（その２）

t_1 の発火により，図 4-6 で示すようにペトリネットにおいては，トークンの移動が生じ，p_1 から p_2 にトークンが移り，マーキングが $M_0 = [\ 1\quad 0\quad 0\ ]^T$ から次のマーキング $M_1 = [\ 0\quad 1\quad 0\ ]^T$ に変わる．

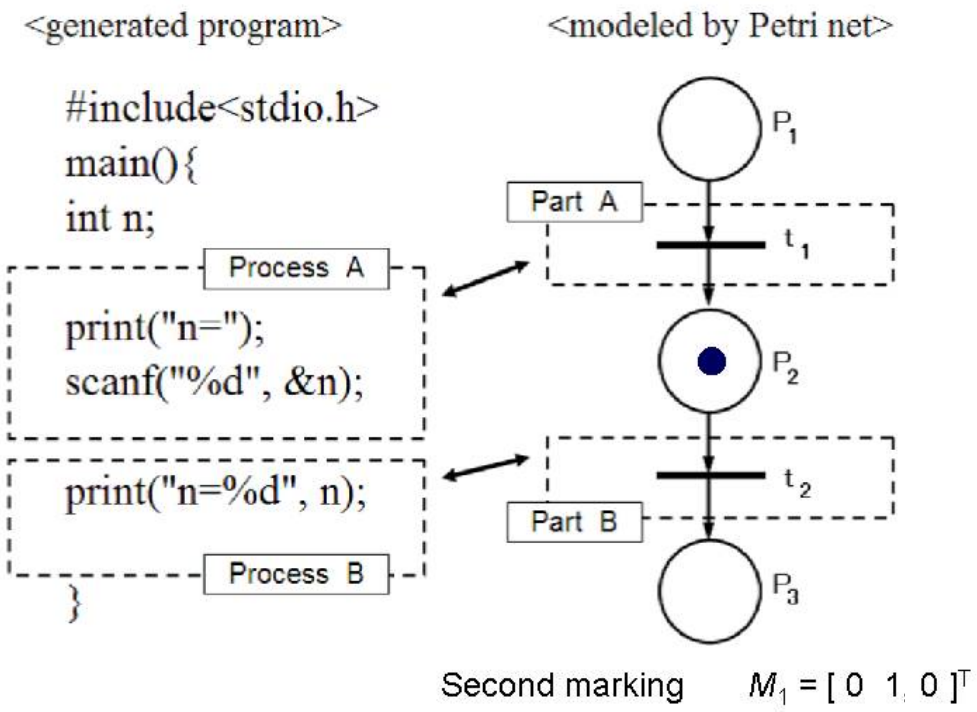


図 4-6　プログラムとペトリネットの対応（その３）

次に図 4-7 で示すように，プログラムは Process B が実行され，ペトリネットにおいては，対応する t_2 が発火（fire）される．

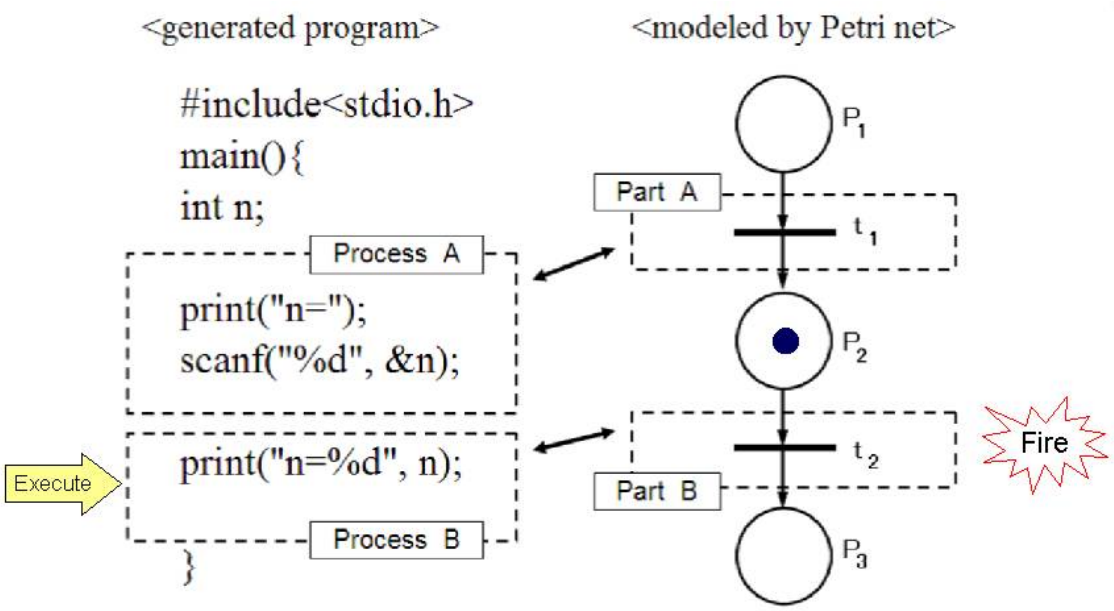


図 4-7　プログラムとペトリネットの対応（その４）

最終的に，図 4-8 で示すように，プログラムは終了（end）し，ペトリネットにおいては t_2 の発火により，トークンが p_2 から p_3 に移動する．このため，ペトリネットの最終マーキング M_d は，図 4-8 でも示すとおり $M_d = [0 \quad 0 \quad 1]^T$ となる．

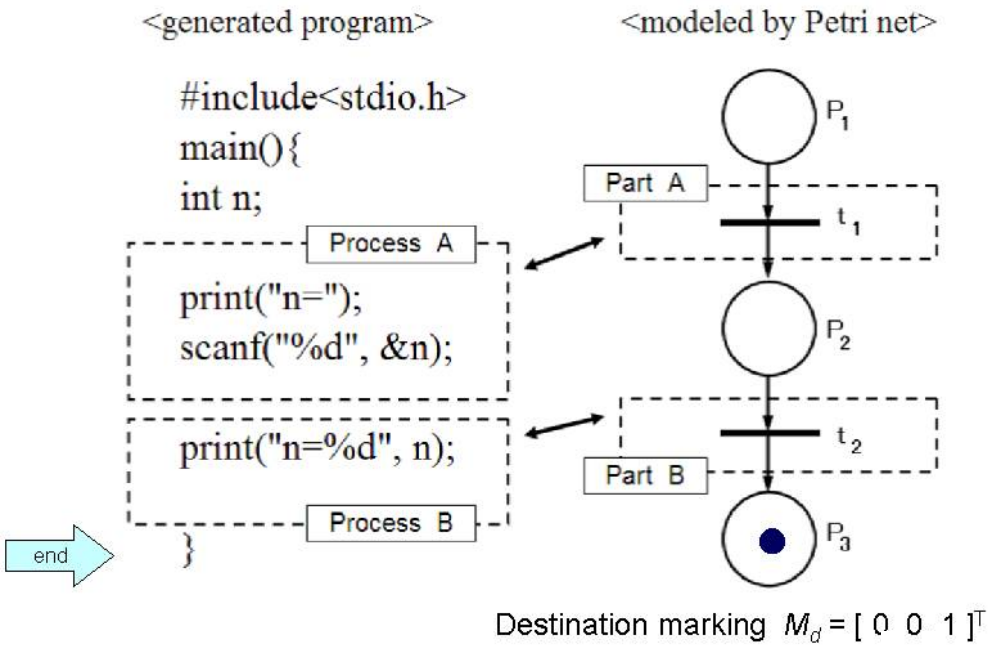


図 4-8　プログラムとペトリネットの対応（その５）

すなわち，プログラムが start から end まで動くことを証明するためには，対応するペトリネットにおいて, Initial marking の M_0 から Destination marking である M_d までの可達性を証明できればよいことになる.

$$
\begin{aligned}
M_0 &= [\ 1 \quad 0 \quad 0\ ]^T \\
&\downarrow t_1 \\
M_1 &= [\ 0 \quad 1 \quad 0\ ]^T \\
&\downarrow t_2 \\
M_d &= [\ 0 \quad 0 \quad 1\ ]^T
\end{aligned}
$$

図 4-9　例題におけるマーキングの遷移

もちろん，図 4-5 から図 4-8 で示したとおり，t_1，t_2 と発火させればよいことはすぐにわかり，また，図 4-9 で示すように，マーキングの遷移を見ても確認できる.

ただ，ここでは数学的手法を用いて証明を行うことを検討しているため，可達木を使わず，２．２．３項で述べた，数学的手法を用いて検討を行う.

ここで，このペトリネットにおける接続行列 A は

$$
A = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \tag{4-1}
$$

であり，このとき $rank(A) = 2$ より

$$
A_{11} = \begin{bmatrix} -1 & 0 \end{bmatrix} \tag{4-2}
$$

$$
A_{21} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \tag{4-3}
$$

である．また，式(2-8)から,

$$
B_f = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{4-4}
$$

となり，初期マーキング M_0 ＝ ［ 1　0　0 ］^{T}，最終マーキング M_d＝［ 0　0　1 ］^{T}，

であることから，そのマーキング差 ΔM は，

$$\begin{aligned} \Delta M &= M_d - M_0 \\ &= [\ -1 \quad 0 \quad 1\ ]^T \end{aligned} \tag{4-5}$$

ここで確かに

$$B_f \Delta M = 0 \tag{4-6}$$

となり，式(2-9)が言え M_0 から M_d に至るための状態方程式の解が存在することは数学的に証明できた．しかしながら，２．２．３項で述べたように，あくまで解の存在が証明できたのみであり，実際のペトリネットにおいて，トランジションの発火によりトークンの移動を行うためには，この解は非負整数解である必要がある．このため，解の存在の証明はあくまで可達性を証明するための必要条件になる．

そこで，可達木を使わず，状態方程式の解を求める数学的手法の Fourier-Motzkin 法を用いて解を求めてみる．なお，ここでは，特にすべての特解を求める必要まではないため，改良 Fourier-Motzkin 法は使わず従来の Fourier-Motzkin 法を用いる．

接続行列 $A \in Z^{m \times n}$ の下に単位行列 $E^{n \times n}$ を置いて行列
$B = [\ A,\ E\ ]^T \in Z^{(m+n) \times n}$ を作る．

$$B = \left[\begin{array}{ccc} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \tag{4-7}$$

式(4-7)の行列 B の第 1 行目に着目し，0 でない要素を 0 にするような B の 2 つの列の正係数 1 次結合（但し係数は最小のものを選ぶ）をすべて B の列に加え，これを新しく B とする．

$$B = \left[\begin{array}{ccc|c} -1 & 0 & 1 & 0 \\ 1 & -1 & 0 & 1 \\ 0 & 1 & -1 & -1 \\ \hline 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right] \tag{4-8}$$

具体的には，1 列目と 3 列目の各要素を足した正係数 1 次結合を新たに B に加え，これを B とする．次に，第 1 行の要素が 0 以外のの列，つまり式(4-8)における 1 列目と 3 列目を削除し，これを新しく B とする．

$$B = \left[\begin{array}{cc} 0 & 0 \\ -1 & 1 \\ 1 & -1 \\ \hline 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{array}\right] \tag{4-9}$$

この場合 $m = 2$ であるので，式(4-9)の行列 B において第 2 行から第 3 行目まで第 1 行目と同様の操作を行い，最終的に式（4-10）が得られる．

$$B = \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \hline 1 \\ 1 \\ 1 \end{array}\right] \tag{4-10}$$

これにより発火回数ベクトル $x = [\ 1 \quad 1\ ]^T$ が得られる．これは，トランジション t_1，t_2 をそれぞれ 1 回ずつ発火させることを示しており，図 4-7 で示したマーキングの遷移（発火系列）とも一致している．

解の存在自体は，式（4-6）により既に証明済であったが，実際の発火回数ベクトルを求めることによって，得られる解が非負整数解であることを示すことができ，まちがいなく可達，すなわちシステムが実行可能であることが検証できた．

実験システム MAPP で行ったプログラムの自動生成において，生成されたプログラムに対し，そのペトリネットモデルにおける可達性を判定することで，プログラムの実行可能性を数学的検証した．ただ，このモデル化は簡単な順次処理においてのみ行っており，分岐，繰り返しなどを含む複雑なプログラムへの適用はまだできていない．今後の課題として適用できる対象のプログラムを広げるためにも，分岐，繰り返しにおけるモデル化を行っていく必要がある．

第５章　結　論

本研究では，プログラムの自動生成システムにおけるモジュールの組み合わせにおいて，おのおのに持たせた入出力条件のチェックに,ペトリネットによる数学的解析法を用いることで,生成されたプログラムの動作可能性を数学的に証明する手法の提案を行った.

ペトリネットとは，ドイツ人，C.A. Petri によって 1962 年に提案された概念であり，システムにおける条件と事象の概念を用いてシステムをグラフモデル化したものである．ペトリネットには，その有界性，活性，可達性などの性質を数学的解析が可能であるという特徴があるため，対象モデルにおける安全性や可動性について検証を行うことが可能となってくる．このことから，ペトリネットの数学的な解析法は各方面で研究されている．特に可達性に関しては，被覆木，状態方程式，ペトリネットの構造と性質に着目する 3 つの手法があるが，被覆木を用いた方法は一般に膨大な計算量を必要とする一方，状態方程式を用いる方法は，ペトリネットの性質を代数方程式の解の存在として考察できる利点があるため，その解法や，解の表現方法などについての研究が進められてきている[1]．本研究では，この状態方程式の解の導出法として Fourier-Motzkin 法を取り上げ，状態方程式の解の導出はもちろん，無限個の解を有限の解で表現する際の展開係数の導出，および，拡張ペトリネットの一つである時間なし連続ペトリネットやハイブリッドペトリネットへの Fourier-Motzkin 法の適用，さらに，Fourier-Motzkin 法のアルゴリズムを改良し従来の Fourier-Motzkin 法では求め切れなかった解を得るなど，ペトリネットの新しい解析法の提案を行った.

一方，1968 年ドイツの Garmisch で開催された NATO の国際会議で「ソフトウェア危機」が問題となって以来，ソフトウェアの生産性と品質向上の技術が研究されて続けている．しかしながら今もなお組込み系ソフトウェアの技術者の不足が叫ばれ続けているなど，ソフトウェア技術者の不足は拡大し続けている．通産省の調べによれば，1992 年時点において我が国で 50 万人の技術者を抱えており，ソフトウェアの市場は 2000 年まで年 8%で拡大を続け，その時点ではあと 50 万人，つまり現在の 2 倍の技術者が必要といわれてきた[13]経緯もある．このような現状の中でプログラムの生産性を上げるために，物理的な技術者の数を向上させるための技術者育成法から，プログラミングの効率を上げるためのプログラミング技法，プログラミングの自動化などあらゆる方法が研究され続けているが決定的な解決法が見つかっていないのが現状となっている．そこで本研究では，プログラムの生産性を上げる手法の一つであるプログラムの自動生成を取り上げ，その実験システムの構築および検討を行っている．試作システム MAPP では生成中にモジュールの入出力条件のチェックを行い，その中で，自動補填等の新たな手法を展開しているが，このモジュールの入出力条件のチェック部分に，ペトリネットの可達判定を用いることで，モジュールの入出力条件のチェックをより厳密に行い，システムの可動性を検証するための手法を提案している.

第 2 章では，ペトリネットと呼ばれるモデル化の手法について述べ，その性質である，可達性，活性などを数学的に解析することで，モデル化したシステムの安全性や可動性について検証できることを紹介するとともに，解析の際重要な情報となる T-インバリアントや可達のための発火回数ベクトルを求める状態方程式の解の導出方法としての Fourier-Motzkin 法について述べ，システム解析に重要な展開係数の導出にも Fourier-Motzkin 法が有効であることを示すとともに，従来の Fourier-Motzkin 法の改良を行い，従来では求め切れていなかった解の一部も新たに求められることを示した．さらに，本来，離散事象に対するものとして提案されたペトリネットの概念を一般化した時間なし連続ペトリネットに対する状態方程式にも Fourier-Motzkin 法が有効であることを示すなど，Fourier-Motzkin 法に関する新たな知見を示した．

第 3 章では，プログラムの生産性向上の一つの手法として，プログラムの自動生成についてその一構成法を提案し，実験システム MAPP の構築及び検討について示した．また MAPP において，各モジュールの入出力条件を用いた自動補填という新しい概念を組み込み，実験によってその効果を確認できた．

第 4 章では，実験システム MAPP で行った自動生成において，でき上がったプログラムに対するペトリネットモデルを用いて，その解析を行うことでシステムの実行可能性を検証する方法について述べた．プログラムを正しくペトリネットに置き換える手法の確立はいまだ途中段階であり未確定要素も強いが，ペトリネットによるモデル化を行うことで，今まで経験的に正しいと判断していた入出力条件のチェックの部分を数学的に証明することができた．

今後については，改良 Fourier-Motzkin 法についてのアルゴリズムを確立し，大きな問題についても扱えるようしていく必要があるとともに，ペトリネットを用いたプログラムの実行可能性の検証については，扱える問題の範囲を順次処理にとどまらず，分岐，繰り返しについても検討を進めていく必要があり，この検証法を MAPP に組み込んで自動的に行えるよう改善していくことが，これからの研究課題としてあげられる．

参 考 文 献

[1] 市川，熊谷，薦田：ペトリネットとその応用，計測自動制御学会(1992).

[2] 稲葉：“仕様記述に基づくシーケンスプログラムの自動生成”，情報処理学会第 45 回全国大会, pp.5-357-358(1992.10).

[3] 田村，稲野，尾関，皆川，加地：“エレベータ制御用ソフトウェアの自動生成”，情報処理学会第 47 回全国大会, pp.5-261-262(1993.10). “”

[4] 青山, 内平, 平石：ペトリネットの理論と実践，朝倉書店(1995).

[5] 村田：ペトリネットの解析と応用，近代科学社(1992).

[6] 松本，茂呂，恐神：“P/T ペトリネットの発火回数ベクトルを T インバリアントと特解で表すときの展開係数導出法”，信学技法, IEICE Technical Report, CAS2006-45, CST-21, pp. 19-24, (2006.11)

[7] 松本，恐神，茂呂：“事象駆動型離散事象システムの挙動検証に関する一考察”，平成 18 年度電気関係学会北陸支部連合会, H-13. (2006.9).

[8] R. David, and H. Alla, Discrete, Continuous, Hybrid Petri Nets, Springer, 2005.

[9] Matsumoto，Osogami and Moro：“On Paticular Solutions for State Equation of Autonomous Continuous Petri Nets”, Proc. of The 2008 International Symposium on Nonlinear Theory and its Applications, NOLTA'08, in Budapest, Hungary, pp. 668-671, (2008.9).

[10] Pelin, A and Mollow, P.：“Automatic program generation from specifications using PROLOG”, NASA Conf Publ, NASA-CP-2492-PT2, pp.53-57, 1988.

[11] Osogami, Nishida:：“A Method of Automatic Program Designing and Soft and Code Generation using Informal Procedure Call Sentences", Proc. of IASTED -ASC'98, pp.161-164, May 1998.

[12] 恐神: “プログラム生成におけるリスト処理を用いた入出力条件のチェック”，福井工業大学研究紀要第 29 号, pp.289-296(1999.3).

[13] 原田　実 : CASE のすべて，オーム社(1991).

[14] 恐神，西田：“プロログによるモジュールの検索とプログラムの合成”，福井工業大学研究紀要 第 23 号, pp. 1－313－320(1993.3).

[15] 恐神，西田：“プロログによるモジュール援用プログラミングシステム”，福井工業大学研究紀要 第 22 号, pp.1－313－320(1992.3).

[16] 遠藤：構造化プログラム設計図法 SPD，共立出版(1992).

[17] 青山，内平，平石：ペトリネットの理論と実践，朝倉書店(1995).

公開論文リスト

第 2 章

１．Matsumoto，Koido and Osogami：“Initial-Marking Based Liveness on Dissynchro nous Choice Petri Nets”, Proc. of The 1996 International Technical Conference on Circuits/Systems, Computers and Communications, in Seoul, pp.713-716, (1996.7).

２．Matsumoto，Moro and Osogami：“How to Obtain Coeffecients for a Firing Count Vector Expanded by T-Invariants and Particular Solutions in P/T Petri Nets”, Proc. of The 2006 International Symposium on Nonlinear Theory and its Applications, NOLTA'06, in Bologna, Italy, pp. 407-410, (2006.9).

３．Matsumoto，Osogami and Moro：“On Paticular Solutions for State Equation of Autonomous Continuous Petri Nets”, Proc. of The 2008 International Symposium on Nonlinear Theory and its Applications, NOLTA'08, in Budapest, Hungary, pp. 668-671, (2008.9).

４．松本，恐神，茂呂：“P/T ペトリネットの状態方程式の非負整数解の代数的構造に関する基礎的考察”, 福井工業大学研究紀要 第 39 号, pp. 1-23-30(2009.8).

５．Matsumoto，Osogami and Moro：“Reachability Judgment in P/T Petri Nets by Approximate Algebraic Approach”, Proc. of The 9th WSEAS International Conference on SIGNAL PROCESSING, ROBOTICS and AUTOMATION， ISPRA'10, in Cambridge, UK, pp. 318-322. (2010.2).

第 3 章

６．恐神，西田：“プロログによるモジュール援用プログラミングシステム”，福井工業大学研究紀要 第 22 号, pp.1-313-320(1992.3).

７．恐神，西田：“プロログによるモジュールの検索とプログラムの合成”，福井工業大学研究紀要 第 23 号, pp. 1-313-320(1993.3).

８．恐神，西田：“プログラム設計と構造化図作成の自動化”，福井工業大学研究紀要 第 24 号, pp. 1-261-268(1994.3).

９．恐神，西田：“概略設計文からのＣソースコードの生成”, 福井工業大学研究紀要 第 25 号, pp. 1-315-322(1995.3).

１０．恐神，西田：“構造体処理汎用モジュールの作成”, 福井工業大学研究紀要 第 26 号, pp. 1-311-318(1996.3).

１１．恐神，西田：“プログラム生成におけるモジュールの自動リンク”，福井工業大学研究紀要 第 27 号, pp. 1-305-312(1997.3).

１２．恐神，西田：“クラスモジュールの一構成法”，福井工業大学研究紀要 第 28 号, pp. 1-261-268(1998.3).

１３．Osogami, Nishida:：“A Method of Automatic Program Designing and Soft and Code Generation using Informal Procedure Call Sentences", Proc. of IASTED -ASC'98,

pp.161-164, May 1998.
１４．恐神：“プログラム生成におけるリスト処理を用いた入出力条件のチェック”, 福井工業大学研究紀要第 29 号, pp.289-296(1999.3).
１５．Osogami ：“A Method of Automatic Program Generation and Stractured Diagram using Informal Procedure Call Sentences", Proc. of IASTED -SEA'99, pp.104-108, Oct. 1999.
１６．恐神：“入出力条件を用いたプログラムの自動生成”, 福井工業大学研究紀要 第 30 号, pp. 1-301-308(2000.3).
１７．Osogami ：“A Method of Automatic Program Generation using Informal Procedure Call Sentences", Proc. of ACIS -SNPD'01, pp.969-976, Aug. 2001.

第 4 章

１８．恐神：“プログラムの自動生成における入出力条件に関する一考察”, 福井工業大学研究紀要 第 37 号, pp. 1-273-278, (2007.5).
１９．Osogami, Yamanishi and Uosaki：“Input-Output Conditions for Automatic Program Generation Using Petri Nets”, Proc. of The KES2011, in Germany, Part I, LNAI 6881, pp. 296-305, (2011.9).
２０．Osogami, Yamanishi and Uosaki：“A Method of Input-Output Conditions for Automatic Program Generation Using Petri Nets”, Proc. of The SICE Annual Conference 2011, in Tokyo, pp. 2415-2420, (2011.9).

発表講演論文リスト

第 2 章

２１．Matsumoto, Osogami：“Necessary and Sufficient Condition for structural liveness of Dissynchronous Choice Petri Nets”, Technical Report of IEICE, vol.96, No.204, pp.39-46, CST96-13, (1996.7).

２２．大野，恐神，松本：“新しい挙動的トラップを有するペトリネットの最簡クラス (AC∩DC ﾈｯﾄ) とその構造活性条件”，電子情報通信学会 1996 年基礎・境界ソサイエティー大会講演論文集, SA-8-2. pp.316-317, (1996.9).

２３．大野，恐神，松本：“挙動的トラップを許容する最簡ペトリネットとその活性条件”, SICE，システム／情報合同シンポジウム '96 講演論文集, pp.183-188, (1996.10).

２４．Ohno, Osogami. Matsumoto：“Relationships between Structure and Behavior of Partially Ordered Petri Nets and their Typical Subclasses: Liveness Conditions, - Part I -”, SICE, 第 19 回離散事象システム研究会論文集, pp.1-8, (1996.12).

２５．松本，恐神，茂呂：“事象駆動型離散事象システムの挙動検証に関する一考察”, 平成 18 年度電気関係学会北陸支部連合会, H-13. (2006.9).

２６．松本，茂呂，恐神：“P/T ペトリネットの発火回数ベクトルを T インバリアントと特解で表すときの展開係数導出法”，信学技法, IEICE Technical Report, CAS2006-45, CST-21, pp. 19-24, (2006.11)

２７．松本，恐神，茂呂：“P/T ペトリネットの初等的 T インバリアントの導出法の比較，検討”，平成 19 年度電気関係学会北陸支部連合会, E-43, (2007.9).

２８．松本，恐神，茂呂：“P/T ペトリネットの可到達性の代数的解析の試み”, 平成 19 年度電気関係学会北陸支部連合会, E-44, (2007.9).

２９．松本，恐神，茂呂：“P/T ペトリネットの可到達性判定の代数的解析の試みについて”, 信学技報, IEICE Technical Report, CAS2007-78, CST2007-29, pp.27-30, (2007.11).

３０．松本，恐神，茂呂：“時間なし連続ペトリネットの状態方程式の解について”, 信学技報, IEICE Technical Report, CAS2008-21, pp.53-57, (2008.8).

３１．松本，恐神，茂呂：“定常連続システムから離散状態システムを得るための一手法”, 信学技報, IEICE Technical Report, CST2010-34, pp.13-18, (2010.8).

第 3 章

３２．恐神，西田：“プロログによるモジュールの検索と合成”，情報処理学会第 43 回全国大会, pp.4-293-294, (1991.10).

３３．恐神，西田：“仕様の記述とプログラムの作成”，情報処理学会第 44 回全国大会, pp.5-191-192, (1992.3).

３４．恐神，西田：“ファイル処理を含むプログラム作成の自動化”，情報処理学会第 45 回全国大会, pp.5-359-360, (1992.10).

３５．恐神，西田：“仕様からの構造化図の簡易作成とプログラムの合成”，情報処理学会第 46 回全国大会, pp.5-327-328, (1993.3).
３６．恐神，西田：“構造化図の作成・チェックとコードの生成”，情報処理学会第 47 回全国大会, pp.5-265-266, (1993.10).
３７．恐神，西田：“モジュールの拡張について”，情報処理学会第 48 回全国大会, pp.5-353-354, (1994.3).
３８．恐神，西田：“フレームワークの一構成法”，情報処理学会第 49 回全国大会, pp.5-183-184, (1994.9).
３９．恐神，西田：“フレキシブルなモジュールの構成について”，情報処理学会第 50 回全国大会, pp.5-219-220, (1995.3).
４０．恐神，西田：“クラスモジュールの一作成法”，情報処理学会第 51 回全国大会, pp.5-157-158, (1995.9).
４１．恐神，西田：“入出条件によるプログラムの合成”，情報処理学会第 52 回全国大会, pp.5-47-48, (1996.3).
４２．恐神：“プロローグを用いたプログラム合成システム”，電子情報通信学会ソフトウェア研究会講演論文集, ss98-11, pp.1-8, (1998.7).

第 4 章

４３．恐神：“プログラムの自動生成における入出力条件について”，電子情報通信学会 FIT2006 第 5 回情報科学技術フォーラム講演論文集, B-023, pp.121-122, (2006.9).
４４．恐神，山西，魚崎：“プログラム合成における入出力条件のチェックでのペトリネットの利用”，情報処理学会第 74 回全国大会，pp.1-253-254, (2012.3).

付　　録

付　録Ａ

Fourier-Matzkin 法のプログラムリスト

```
#include    <stdio.h>
#include    <math.h>
#include    <string.h>
#include    <stdlib.h>

void init_parameter(void);                          // パラメータの初期化、各初期値代入
void init_Matrix_B_Array(void);                     // 配列の初期化
void draw_matrix_B(long);                           // 受け取った変数の配列（行と列）をファイルに出力
void comp_col(void);                                // サブメイン
long check_tmp_row_all_0(long);                     // 第 tmp_row_num 行の 各要素がすべて０かどうか調べる
                                                    //(すべて０＝＞　０、すべて０でない＝＞９９９）
long check_min_vector(long);                        // 以下、式の元列から、その持っている式を追加した列に作成
long check_b_i_bigger_b_j(long, long, long);// 極小ベクトルかどうかの判定＜重要＞！
long check_b_i_eq_b_j(long, long);      // ベクトルを比べる（ｂ_i　＝　ｂ_j）かどうか？　この場合消す //　 DEL = 1, OK = 0 (return)
void del_comp_col(long);                            // comp_col 列を削除する
void add_to_next_sheet(long);                       // 受け取った matrix_B_old 列(tmp_col_num)を次のシート(matrix_B)にコピーする
                                                    //＊＊＊　追加　＊＊＊
void check_row_matrix_B_old(long);  // 行番号を受け取り、その行の要素が０の列を新しいシートに追加する。
void check_row_matrix_B_old_for_MFM(long, long);// 行番号と、旧Ｂ行列の max_列数を受け取り、　（ＭＦＭ 対応モジュール）
                                                    //？？？正しいチェック方法は不明 ！！　？？？//　旧Ｂ行列の第 tmp_row_num 行（が０の要素）
                                                    //すべてと、追加されたすべての列を加え新Ｂ行列とし、新しいシートに追加する。
                                                    //なおかつ、新しく追加した列のすべてに含まれない列のみ Matrix_B へコピーするﾁｪｯｸを追加 ***
void check_row_matrix_B_old_for_MFM_old(long, long);// 行番号と、旧Ｂ行列の max_列数を受け取り、　（ＭＦＭ 対応モジュール）旧Ｂ行列の第 tmp_row_num 行が ０ の要素と、追加されたすべての列を加え新Ｂ行列とし、新しいシートに追加する。
long no_included_all_added_cols(long, long, long); // 追加された列のすべてに、もともとの列が含まれていなければ、新しいシートに加えていけるため、
                                                    //その条件をチェックする。// OUT = 0; OK = 1
long gcd(long, long);                               // 最大公約数を求める（GCD）
long gcd_2(long, long);                             // 最小公倍数を求める


// ****************************************
//　 チェック２（配列バージョン）**************
//　　 列追加の終了後、新しいシートに移る際、（たとえば＋の要素の）絶対値の最大値を超える、逆符号（－の要素の）の接待値を持つ列は、削除できる。
//　 列の制限を越えるため、配列で定義を行う。　sheets(Matrix_B_i).Cells(j, k) → Matrix_B(i, j, k) に置き換え
// ****************************************

// ****************************************
//　 修正_1（配列バージョン）**************
//　 FM、MFM の両方とも、配列で計算を行えるように変更
//　 列の制限を越えるため、配列で定義を行う。　sheets(Matrix_B_i).Cells(j, k) → Matrix_B(i, j, k) に置き換え
//　 FM MFM の ラジオボタンにより、計算方法を選択可能に変更
// ****************************************

// ****************************************
//  global 変数の宣言・定義
// ****************************************
    long max_row_num, max_col_num, max_col_num_old, max_col_num_new;
    long kijun_row, kijun_col, sheet_count, sheet_count_old;
    char matrix_B[256], matrix_B_old[256];
    long eq_Array[200], vector_Array[100];
    long f_mfm;                                     // old_FM = 2, MFM = 1 振り分けるためのフラグ
//    long row_num, col_num;
//    long Matrix_B_Array[200][200][3000];          //（シート番号，行，列）を配列で宣言
```

```
// ********** 配列の定義部分（はじめ） **********
    long row_num = 3;                   // 行数：入力値 ＊＊＊
    long col_num = 6;                   // 列数：入力値 ＊＊＊
//              Matrix_B_Array = new long[200][200][3000] =
                long Matrix_B_Array[24][12][512] =
                {{
                        {  0,  0,  0,  0,  0,  0,  0},          // ０行目は０で埋める（エクセルとの整合性を取るため）
                        {  0, -1, -1,  0,  0,  1,  1},          // ０列目も０で埋める（エクセルとの整合性を取るため）<以下同>
                        {  0,  1,  1, -1, -1,  0,  1},
                        {  0,  0,  0,  1,  1, -1, -2},
                /*  ------------------------------ */
                        {  0,  1,  0,  0,  0,  0,  0},
                        {  0,  0,  1,  0,  0,  0,  0},
                        {  0,  0,  0,  1,  0,  0,  0},
                        {  0,  0,  0,  0,  1,  0,  0},
                        {  0,  0,  0,  0,  0,  1,  0},
                        {  0,  0,  0,  0,  0,  0,  1}
                }};
// ********** 配列の定義部分（おわり） **********

// ****************************************
//      メイン
// ****************************************
void main(void){
    init_parameter();
    comp_col();
//      after_all_0_end_prosess();              //すべて０になった後、最小公約数等の処理を行い、お互いともに、共有結合でないものだけ残す
}
// ****************************************
//    パラメータの初期化、各初期値代入
// ****************************************
void init_parameter(void){
//      row_num = 3;                            // 行数：入力値 ＊＊＊
//      col_num = 6;                            // 列数：入力値 ＊＊＊
    max_row_num = row_num + col_num;            // 接続行列の下に単位行列を加えた際の行数
    max_col_num = col_num;                      // 計算途中で使用していく最大列数(変数)
    f_mfm = 2;                                  // old_FM = 2, MFM = 1 振り分けるためのフラグ
//      make_Matrix_B_Array();
}

/* ********** ooo **********

// ****************************************
//    エクセルシートから配列のへの移動 ＜＜いらない！＞＞
// ****************************************
void make_Matrix_B_Array(void)                  // エクセルシートから配列への移動（ここではいらない？！）
    long i, j, k, tmp_val;
    long i_sheet, j_sheet, k_sheet;

    init_Matrix_B_Array();
    matrix_B = "B0";
    sheet_count = 0; sheet_count_old = 0;
    i = sheet_count;
    for( j = 1 ; j <= max_row_num ; j++){
        for( k = 1 ; k <= max_col_num ; k++ ){
            Matrix_B_Array[i][j][k] = Sheets(matrix_B).Cells(j, k)                  // ********** //
        }
    }
}
```

```
// ****************************************
//    配列の初期化 ＜＜いらない！＞＞
// ****************************************
void init_Matrix_B_Array(void){
    long i, j, k;
    for( i = 0 ; i <= 200 ; i++){
        for( j = 0 ; j <= 200 ; j++ ){
            for( k = 0 ; k <= 3000 ; k++ ){
                Matrix_B _Array[i][j][k] = NULL;
            }
        }
    }
}

// ****************************************
//    受け取った変数の配列（行と列）をファイルに出力
// ****************************************
void draw_matrix_B(long tmp_received_sheet_count){
    long j, k, tmp_val;
    FILE *fp;
    char f_name[256] = "out_Matrix_";
            char tmp[256];

            ltoa(tmp_received_sheet_count, tmp, 10);
            strcat( f_name, "B_" );
            strcat( f_name, tmp );
            strcat( f_name, ".csv" );
            if( NULL == (fp = fopen( f_name, "w"))){
                    printf("¥7¥n Cannot Open the File : %s ¥n", f_name );
            }

            printf("Writing    sheet_count [%3d/%3d]th. max_col_num = %d ¥n", tmp_received_sheet_count, row_num, max_col_num );

    for( j = 1 ; j <= max_row_num; j++ ){
        for( k = 1 ; k <= max_col_num ; k++ ){
            tmp_val = Matrix_B_Array[sheet_count_old][j][k];
            fprintf(fp, "%5d", tmp_val);
            if( k != max_col_num ){
                    fprintf(fp, ", ");
            }
        }
        fprintf(fp, " ¥n");
    }
    fclose(fp);
}

****************** */

// ****************************************
//    受け取った変数の配列（行と列）をファイルに出力（旧）
// ****************************************
void draw_matrix_B(long tmp_received_sheet_count){
            int                 j=0, k=0;
            FILE        *fp;
            char        f_name[256] = "Matrix_B_Array.txt";

            if( NULL == (fp = fopen( f_name, "a"))){
                    printf("¥7¥n Cannot Open the File : %s¥n", f_name );
            }

            fprintf(fp, "sheet_count = %03d, row_num_now = ??? / %d, max_col_num = %d  ¥n", tmp_received_sheet_count, row_num,
max_col_num );
            printf("sheet_count = %03d, row_num_now = ??? / %d, max_col_num = %d ¥n", tmp_received_sheet_count, row_num, max_col_num );
```

```
            for(j = 1 ; j <= max_row_num ; j ++ ){
                        for( k= 1 ; k <= max_col_num ; k++ ){
                                    fprintf(fp, " %4d, ", Matrix_B_Array[tmp_received_sheet_count][j][k] );
                                    printf(" %4d, ", Matrix_B_Array[tmp_received_sheet_count][j][k] );
                        }
                        fprintf(fp, "¥n");
                        printf("¥n");
            }
            fclose(fp);
}
/* ************* ooo ************** */

// ***************************************
//    サブメイン
// ***************************************
void comp_col(void){
    long new_max_col_num, tmp_count, tmp_val;
    long tmp_row_num, tmp_col_num;                    // ループの変数（ネストの１番外側と２番目）それぞれ、行と列
    long now_row_num, now_col_num;                    // ループの変数（ネストの３番目）それぞれ、列と行
    long tmp_val_1st, tmp_val_2nd, alpha_1st, beta_2nd;
            long count;
            long root_data, comp_data;

    sheet_count = 0; tmp_count = 0;                   // 変数初期化

    for( tmp_row_num = 1 ; tmp_row_num <= row_num ; tmp_row_num++ ){
        new_max_col_num = max_col_num; max_col_num_old = max_col_num;        // 行列Ｂｊの当初の列数を、一時変数（new_max_col_num）
                                                                             //と大域変数(max_col_num_old)にコピー。大域変数で統一
                                                                             //すべき？？？　両変数は等しい
        count = 1; max_col_num_new = 1;

        for( tmp_col_num = 1 ; tmp_col_num <= new_max_col_num ; tmp_col_num ){
            tmp_val = Matrix_B_Array[sheet_count][tmp_row_num][tmp_col_num];
            tmp_val_1st = tmp_val;                                           // 当初の行における比較要素（元ネタ）をキープ
            if( tmp_val_1st != 0 ){                   // 0 以外の場合処理を行う（// tmp_row_num 行目の値が０の場合新しいＢn に移す）
                        for( now_col_num = tmp_col_num ; tmp_col_num <= new_max_col_num ; tmp_col_num++ ){

            printf(" *** test Debuging *** sheet_count:[%3d/%3d]th. max_col_num = %d, tmp_row_num = %d, tmp_col_num = %d, count = %d ¥n",
sheet_count, row_num, max_col_num, tmp_row_num, tmp_col_num, count );

                                    tmp_val = Matrix_B_Array[sheet_count][tmp_row_num][now_col_num];
                                    tmp_val_2nd = tmp_val;                  // 当初の行における比較先要素（比較ネタ）をキープ
                                    if( (tmp_val_2nd != 0) || ( (tmp_val_1st * tmp_val_2nd) < 0 ) ){
                                                                            // tmp_val_2nd が０でもなく、tmp_val_1st と同じ符号でもない場合
                                    for( now_row_num = tmp_row_num ; now_row_num <= max_row_num ; now_row_num++ ){
                                        root_data = Matrix_B_Array[sheet_count][now_row_num][tmp_col_num];          //基データ
                                        comp_data = Matrix_B_Array[sheet_count][now_row_num][now_col_num];          //比較先データ
                                        max_col_num = new_max_col_num + count;// ***** new_max_col_num を max_col_num（global）に代入！
                                        if( f_mfm == 2 ){                                   //FM の場合の重み付け（最小公倍数）
                                            alpha_1st = gcd( labs(tmp_val_1st), (labs(tmp_val_2nd) ) / labs(tmp_val_1st) );
                                            beta_2nd = gcd( labs(tmp_val_1st), (labs(tmp_val_2nd)) / labs(tmp_val_2nd) );
                                        }
                                        else if( f_mfm == 1 ){                              //MFM の場合の重み付け（すべて １）
                                            alpha_1st = 1; beta_2nd = 1;                    // 重みを” １”にして計算（MFM）
                                        }
                                        else{
                                            printf(" 最小公倍数計算エラー");                 // End
                                            break;
                                        }
                                        Matrix_B_Array[sheet_count][now_row_num][max_col_num] = root_data * alpha_1st + comp_data * beta_2nd;
                                    //各列を足したものを、最終列に追加 ///// 最小公倍数になるように計算！！//（old_ＦＭ）MFM の場合は、重みが”１”
                                    }
                                    tmp_count = check_min_vector( tmp_row_num );        //操作中の行を受けて、極小ベクトルの判定を行う ******
```

```
                                                                    //行を削除した場合、tmp_count の値が１（そうでないなら０）にする
                        max_col_num = max_col_num - tmp_count;
                        count = count + 1;              //追加の行を１右へずらす
                        count = count - tmp_count;                                  //count から tmp_count の値を引いて 行数の調整をする。
                    }
                                                                                    //break_1:    //基準の列が０の場合に飛ばす（break 処理）
                        tmp_count = 0;
                                                                                    // break 処理の際の tmp_count 初期化
            }
            if( now_col_num >= new_max_col_num ){
                max_col_num = max_col_num - tmp_count;          //列同士の＋－の組み合わせ計算の最後（）に列を削除した場合は、
                                                ///削除された列数のカウントが行われないので、max_col_num を１つ（減らした列数）減らす
            }
                    //＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
                    //  ０．max_col_num（global）を new_max_col_num に置換え
                    //  ☆．極小ベクトルの判定を行う。*** ★☆★☆★ ***
                    //  １．列の式番号をチェックして、重複分を削除。
                    //  ２．列の要素をチェックして、全部０なら、解として抽出・削除
                    //  ３．削除後の列数を、max_col_num に置換え（０と重複）
                    //＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
        }
        tmp_count = 0;                          // break 処理の際の tmp_count 初期化
    }                                           //tmp_col_num 列の tmp_row_num 行目の値が０の場合に飛ばす（break 処理）
// ***************************************
// * ここで、今操作していた行（tmp_row_num）の要素が０の列をすべて、新しいシートにコピーし、
// * 行列の大きさのパラメータも設定しなおす（そのあと、再び次の行の操作を行う。）
// ***************************************
    sheet_count_old = sheet_count;
    sheet_count = sheet_count + 1;
    draw_matrix_B(sheet_count_old);                 // 操作した行列を表示する。　　＜＝＝＝　タイミング注意！！
        if( f_mfm == 2 ){                           // old_FM の場合にのみ実行（基準の行が０のものだけを、次のシートへコピー）
            check_row_matrix_B_old(tmp_row_num);    // (old_FM) matrix_B_old の tmp_row_num 行の 0 の列を matrix_B へコピー
        }else if( f_mfm == 1 ){                     // MFM の場合にのみ実行（操作の行を変えずにそのままにするかどうか）
            check_row_matrix_B_old_for_MFM( tmp_row_num, new_max_col_num );
                                                    // (MFM) matrix_B_old の tmp_row_num 行の new_max_col_num 列までの
                                                    // 要素 0 の列と、new_max_col_num 以降のすべての列を matrix_B へコピー
                                                    //max_col_num 新しいシートの max_col_num を設定しなおし
            if( check_tmp_row_all_0( tmp_row_num ) != 0 ){  //"OK でないなら //(MFM) 新しいシートにコピーした tmp_row_num 行のすべての
                                //要素が０でなければ、tmp_row_num をそのまま（for 文で追加されるので、その前に －１ しておく）
                if( max_col_num_old == max_col_num ){       //(MFM) 新しいシートにコピーしても変わらなかったら、判定結果後の列の数が
                                //変わらなくなったら、現在の行の０要素のみ取り出し、新しい行列にし、次の行に移る
                    check_row_matrix_B_old_for_MFM_old( tmp_row_num, new_max_col_num );
                                // (MFM) matrix_B_old の tmp_row_num 行の new_max_col_num 列までの要素 0 の列と、
                                //new_max_col_num 以降のすべての列を matrix_B へコピー
                    tmp_row_num = tmp_row_num + 1;                  // 検証する行を　１つ上げる
                }
                tmp_row_num = tmp_row_num - 1;
            }
        }
    }
    draw_matrix_B(sheet_count);                                     //最後に操作した分の行列を表示する。
}

// ***************************************
//    第 tmp_row_num 行の 各要素がすべて０かどうか調べる(すべて０＝＞　０、すべて０でない＝＞９９９）
// ***************************************
long check_tmp_row_all_0(long tmp_row_num){
    long val_return;
    long i, tmp_val;
    val_return = 0;                                                 // OK
    for( i = 1 ; i <= max_col_num ; i++ ){
        tmp_val = Matrix_B_Array[sheet_count][tmp_row_num][i];
        if( tmp_val != 0 ){
```

```
            val_return = 999;
            break;                                  // １つでも０でない要素があれば、そこで終了
        }
    }
    return( val_return );
}

// ***************************************
//    以下、式の元列から、その持っている式を追加した列に作成
//    行を削除した（極小ベクトルでなかった）場合、このファンクションの値を１とし、行の追加カウント数から引いておいてもらう
//    DEL = 1, OK = 0;
// ***************************************
long check_min_vector(long tmp_row_num){
    long i, j, tmp_val_1st, tmp_val_2nd, del_count;
    long root_col, comp_col, tmp_val_of_sp_ans;
    long vector_f;                                  // DEL = 1, OK = 0;

    del_count = 0; vector_f = 0;
    for( i = 1 ; i <= max_col_num ; i++ ){
        tmp_val_1st = Matrix_B_Array[sheet_count][tmp_row_num][i];
        if( tmp_val_1st == 0 ){
            root_col = i;                           // 基準の列番号を root_col に代入
            for( j = 1 ; j <= max_col_num ; j++ ){
                comp_col = j;
                tmp_val_2nd = Matrix_B_Array[sheet_count][tmp_row_num][j];      //比べる列の対象行の要素
                tmp_val_of_sp_ans = Matrix_B_Array[sheet_count][max_row_num][j];
                    //特解が１になるための１番したの値を取得　＊＊＊＊　有理数（分数）に値にして、判定する　’　＊＊＊追加修正
                if(!(tmp_val_2nd == 0 || i == j)){          // 比べる列の要素が、０　または、基準列と同じ　列は　飛ばす
                    vect or_f = check_b_i_bigger_b_j(root_col, comp_col, tmp_val_of_sp_ans);
            //基準の列_root_col(=i)（要素が０）と残りの列_comp_col(=j)すべてを比べる
            //極小判定
            //第m要素以下の各要素を比較し、root_col 側に１つでも大きい数があれば、残し、それ以外は comp_col の列を削除する。
                    if( vector_f = 1 ){             // DEL
                        del_comp_col(comp_col);     //極小ベクトルでなかったら comp_col 列を削除する
                        del_count = del_count + 1;  // 削除した列数をカウントアップ
                        if( vector_f == 1 ){        //DEL

            printf(" *** test Debuging *** checking _ vector max_col_num = %d, del_count = %d ¥n", max_col_num,   del_count );

                            return( del_count );    //削除した列数を変数として返す
                        }
  //                        else{
  //                            return( 0 );        //0 を返す
  //                        }
                    }
                }
            }
        }
    }
    for( i = 1 ; i <= max_col_num ; i++ ){
        root_col = i;                               // 基準の列番号を root_col に代入
        for( j = 1 ; j <= max_col_num ; j++ ){
            comp_col = j;
            if( i != j ){                           //同じ列は　飛ばす
                vector_f = check_b_i_eq_b_j(root_col, comp_col); //基準の列_root_col(=i)（要素が０）と残りの列_comp_col(=j)すべてを比べる
                                //同じ値（列）
                                //各要素を比較し、root_col 側 と comp_col 側のすべての要素が同じ場合は、comp_col の列を削除する。
                if( vector_f == 1 ){    // DEL
                    del_comp_col(comp_col);         //極小ベクトルでなかったら comp_col 列を削除する
                    del_count = del_count + 1;      //削除した列数をカウントアップ
                    if( vector_f == 1 ){            //DEL
                        return( del_count );        //削除した列数を変数として返す
                    }
```

```
//                          else{
//                              return( 0 );                    //0 を返す
//                          }
                        }
                    }
                }
            }
        }
```

```
// ****************************************
//    ベクトルを比べる（ｂ_i　＞　ｂ_j）かどうか？　この場合のみ残す。あとは削除　　’特解を１にした、有理数（分数）で判定する
// ****************************************
long check_b_i_bigger_b_j(long root_col, long comp_col, long tmp_val_of_sp_ans){
    long i;
    double tmp_val, tmp_val_2;
    long val_return;                                    // DEL = 1, OK = 0;

    val_return = 1;
    for( i = row_num + 1 ; i <= max_row_num ; i++ ){    //第m行以下を比べる
        tmp_val = Matrix_B_Array[sheet_count][i][root_col];
        if( tmp_val_of_sp_ans == 0 ){
            tmp_val_2 = Matrix_B_Array[sheet_count][i][comp_col];
        }else{
            tmp_val_2 = Matrix_B_Array[sheet_count][i][comp_col] / tmp_val_of_sp_ans;
        }
        if( tmp_val > tmp_val_2 ){                      //１つでも大きいものがあれば OK（残す）を返す
            val_return = 0;
            break;                                      // 後の処理はしなくても OK
        }
    }
    return( val_return );
}
```

```
// ****************************************
//    ベクトルを比べる（ｂ_i　＞＝　ｂ_j）かどうか？　b_i が b_j を含んでいるかどうか
//    OUT = 0, OK = 1;
// ****************************************
long check_b_i_include_b_j(long root_col, long comp_col){
    long i;
    long tmp_val, tmp_val_2;
    long val_return;                                                    // OUT = 0, OK = 1;
    val_return = 1;
    for( i = row_num + 1 ; i <= max_row_num ; i++ ){                    //第m行以下を比べる
        tmp_val = Matrix_B_Array[(sheet_count - 1)][i][root_col];       //Matrix_old
        tmp_val_2 = Matrix_B_Array[(sheet_count - 1)][i][comp_col];     //Matrix_old
        if( tmp_val < tmp_val_2 ){              // １つでも大きいものがあれば OUT（残す）を返す
            val_return = 0;                                             // OUT = 0, OK = 1;
            break;                                                      // 後の処理はしなくても OK
        }
    }
    return( val_return );                                               // OUT = 0, OK = 1;
}
```

```
// ****************************************
//    ベクトルを比べる（ｂ_i　＝　ｂ_j）かどうか？　この場合消す
//    DEL = 1, OK = 0 (return)
// ****************************************
long check_b_i_eq_b_j(long root_col, long comp_col){
    long i;
    long tmp_val, tmp_val_2;
    long val_return;

    val_return = 1;
```

```
    for( i = 1 ; i <= max_row_num ; i++ ){                          //すべての要素を比べる
        tmp_val = Matrix_B_Array[sheet_count][i][root_col];
        tmp_val_2 = Matrix_B_Array[sheet_count][i][comp_col];
        if( tmp_val != tmp_val_2 ){                                 //１つでも異なる要素があれば OK（残す）を返す
            val_return = 0;                                         // OK
            break;                                                  // 後の処理はしなくても OK
        }
    }
    return( val_return );
}

// ****************************************
//    comp_col 列を削除する                          ＊＊＊＊＊＊　変更要　＊＊＊＊　　ososo
// ****************************************
void del_comp_col(long comp_col){
    long j, k, tmp_val;

    for( j = 1 ; j <= max_row_num ; j++ ){
        for( k = comp_col ; k <= max_col_num ; k ++ ){
            tmp_val = Matrix_B_Array[sheet_count][j][k + 1];
            Matrix_B_Array[sheet_count][j][k] = tmp_val;
        }
    }
}

// ****************************************
//    受け取った matrix_B_old 列(tmp_col_num)を次のシート(matrix_B)にコピーする　　＊＊＊　追加　＊＊＊
// ****************************************
void add_to_next_sheet(long tmp_col_num){
    long tmp_val, i;

    for( i = 1 ; i <= max_row_num ; i++ ){
        tmp_val = Matrix_B_Array[sheet_count_old][i][tmp_col_num];
        Matrix_B_Array[sheet_count][i][max_col_num_new] = tmp_val;
    }
    max_col_num_new = max_col_num_new + 1;
}

// ****************************************
//    行番号と、旧Ｂ行列の max_列数を受け取り、　（ＭＦＭ 対応モジュール）　？？？正しいチェック方法は不明 ！！　？？？
//    旧Ｂ行列の第 tmp_row_num 行（が０の要素）すべてと、追加されたすべての列を加え新Ｂ行列とし、新しいシートに追加する。
//    なおかつ、新しく追加した列のすべてに含まれない列のみ Matrix_B へコピーするﾁｪｯｸを追加 ***
// ****************************************
void check_row_matrix_B_old_for_MFM(long tmp_row_num, long old_max_col_num){
    long tmp_row_num_tmp, i, count;
    long check_add_able_col;                                        // OUT = 0; OK = 1

    check_add_able_col = 0;                                         // "OUT"
    count = 0;
    for( i = 1 ; i <= max_col_num ; i++ ){
        if( i <= old_max_col_num ){                                 // 旧Ｂ行列の範囲は ０要素のみ移動
            tmp_row_num_tmp = Matrix_B_Array[(sheet_co unt_old)][tmp_row_num][i];          //基準行が０の場合無条件追加
            if( tmp_row_num_tmp != 0 ){
                check_add_able_col = no_included_all_added_cols(tmp_row_num, old_max_col_num, i);
                                //元の列の値が、追加した列のすべてに含まれていない場合、新しいシートにコピーできる "OK"なら追加
            }
            if( tmp_row_num_tmp == 0 || check_add_able_col == 1 ){  // "OK"
                add_to_next_sheet(i);                               //matrix_B_old の tmp_row_num 行の i 列(要素 0)を matrix_B へコピー
                count = count + 1;
            }
                    }else{                                          //追加された列は、すべて移動
             add_to_next_sheet(i);    //matrix_B_old の old_max_col 列以降に追加された tmp_row_num 行 の i 列(すべて)を matrix_B へコピー
            count = count + 1;
```

```
        }
    }
    max_col_num = count;                        // 新しいコピー先の matrix_B の最大列数をセット
}

// ****************************************
//    追加された列のすべてに、もともとの列が含まれていなければ、新しいシートに
//    加えていけるため、その条件をチェックする。// OUT = 0; OK = 1;
// ****************************************
long no_included_all_added_cols(long tmp_row_num, long old_max_col_num, long comp_col){
    long i, tmp_val, root_col, count_1, count_2;
    long check_included;                                                    // OUT = 0, OK = 1;
//  long tmp_f;                                                             // OUT = 0, OK = 1;
    long return_val;                                                        // OUT = 0, OK = 1;

    count_1 = 0; count_2 = 0;
    return_val = 1;                                                         // OUT = 0, OK = 1;
    tmp_val = Matrix_B_Array[(sheet_count - 1)][tmp_row_num][comp_col];
            if( tmp_val != 0 ){
        for( i = (old_max_col_num + 1) ;  i <= max_col_num ; i++ ){
            check_included = 0; root_col = i;               // OUT = 0, OK = 1;
            check_included = check_b_i_include_b_j(root_col, comp_col);     //すべてに含まれていない場合"OUT"になる
            count_1 = count_1 + 1;
            if( check_included == 1 ){                                      // OK
                count_2 = count_2 + 1;
            }
        }
    }
    if( count_1 == count_2 ){                       //追加された列、すべてに含まれていたら、"OUT"とし、新しいシートには追加しない
        return_val = 0;                                                     // OUT
    }
    return( return_val );                           //OK＝新しいシートに追加、"OUT"＝comp_col を削除
}

// ****************************************
//    行番号と、旧Ｂ行列の max_列数を受け取り、　（ＭＦＭ 対応モジュール）
//    旧Ｂ行列の第 tmp_row_num 行が ０ の要素と、追加されたすべての列を加え新Ｂ行列とし、新しいシートに追加する。
// ****************************************
void check_row_matrix_B_old_for_MFM_old(long tmp_row_num, long old_max_col_num){
    long tmp_row_num_tmp, i, count;

    count = 0;
    for( i = 1 ; i <= max_col_num ; i++ ){
        if( i <= old_max_col_num ){                 //  旧Ｂ行列の範囲は ０要素のみ移動
            tmp_row_num_tmp = Matrix_B_Array[(sheet_count - 1)][tmp_row_num][i];
            if( tmp_row_num_tmp == 0 ){
                add_to_next_sheet(i);               //matrix_B_old の tmp_row_num 行の i 列(要素 0)を matrix_B へコピー
                count = count + 1;
            }
        }else{                                      //追加された列は、すべて移動
            add_to_next_sheet(i);     //matrix_B_old の old_max_col 列以降に追加された tmp_row_num 行 の i 列(すべて)を matrix_B へコピー
            count = count + 1;
        }
    }
            max_col_num = count;                    //新しいコピー先の matrix_B の最大列数をセット
}
```

```
// ****************************************
//    行番号を受け取り、その行の要素が０の列を新しいシートに追加する。
// ****************************************
void check_row_matrix_B_old(long tmp_row_num){
    long tmp_row_num_tmp, i, count;

    count = 0;
    for( i = 1 ; i <= max_col_num ; i++ ){
        tmp_row_num_tmp = Matrix_B_Array[(sheet_count - 1)][tmp_row_num][i];
        if( tmp_row_num_tmp == 0 ){
            add_to_next_sheet(i);   //matrix_B_old の tmp_row_num 行の i 列(要素 0)を matrix_B へコピー
            count = count + 1;
        }
    }
    max_col_num = count;                    //新しいコピー先の matrix_B の最大列数をセット
}


//****************************************
//    最大公約数を求める（GCD）
//****************************************
long gcd(long x, long y){

    if( y == 0 ){
        return( x );
            }else{
        return( gcd(y, (x % y)) );
            }
}
//****************************************
//    最小公倍数を求める
//****************************************
long gcd_2(long x, long y){
    long tmp, ans;
            tmp = gcd(x, y);
    ans = x * y / tmp;
            return( ans );
}
```

付　録 B

MAPP 操作時の表示画面

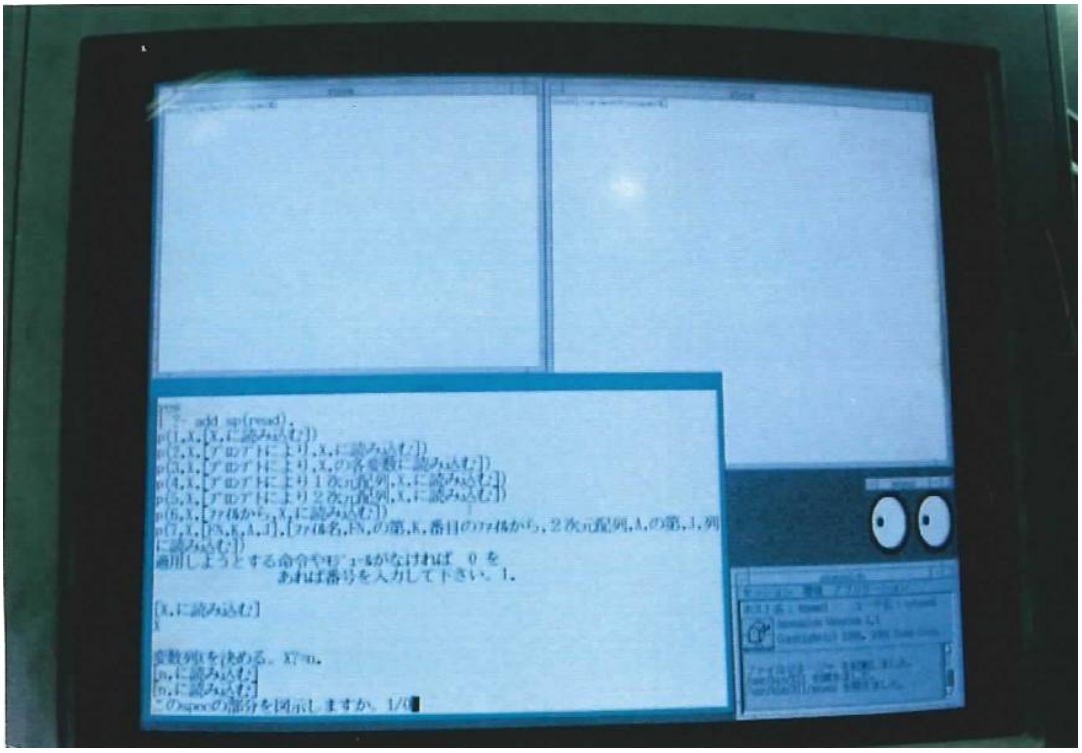


図 A-1　最初の手続き（読み込み）を指示するときの画面.

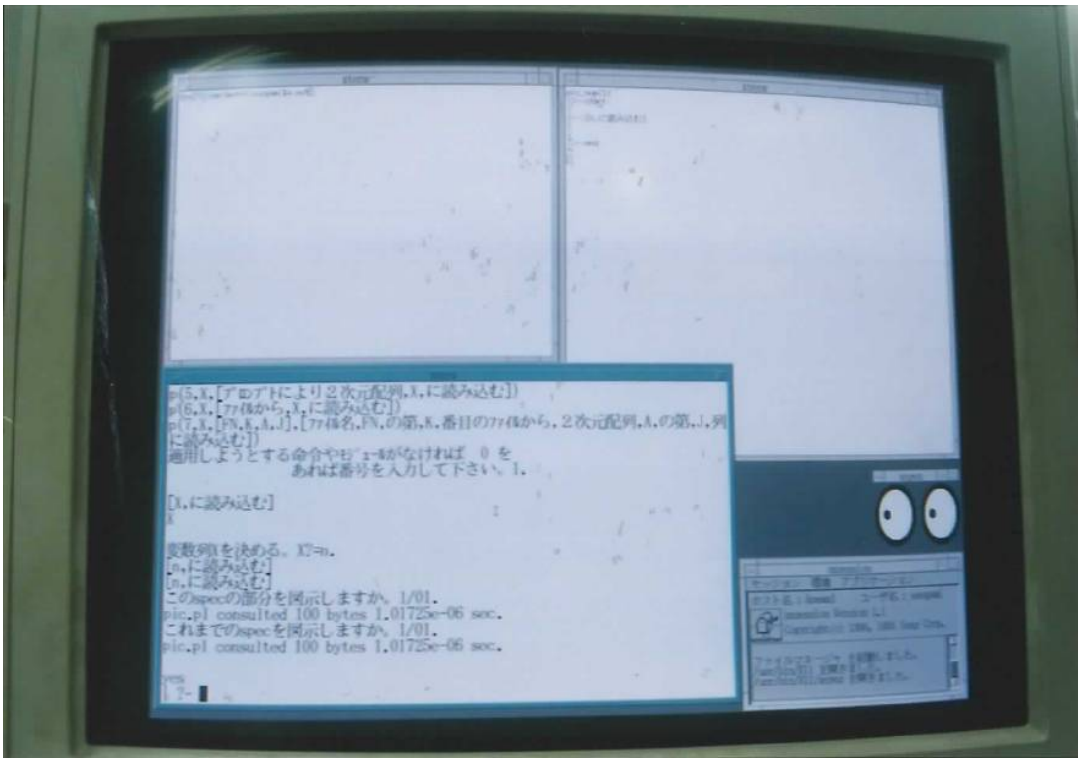


図 A-2　手続きを設計文書に追加し，その構造化図が表示された時の画面

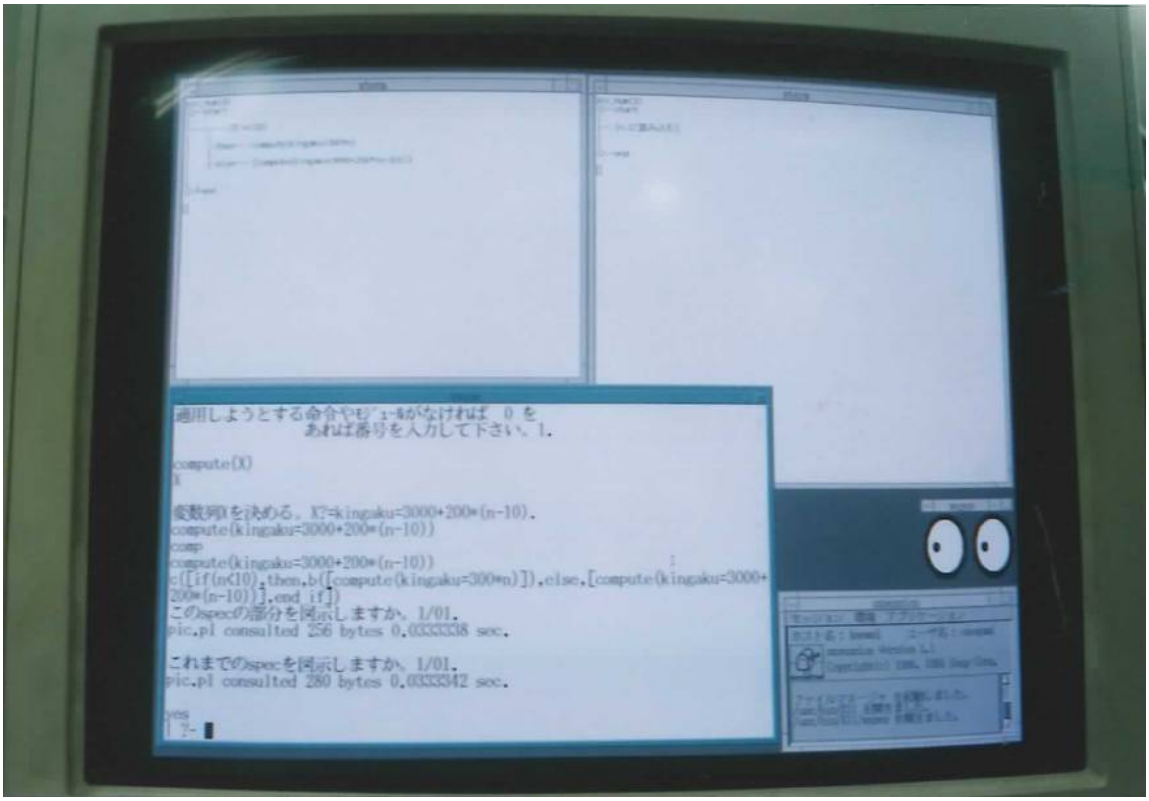


図 A-3　次の手続きの，制御文の設計について，全体の構造を表示しているウィンドウとは別のウィンドウ上（画面左上）で，指定を行った時の画面.

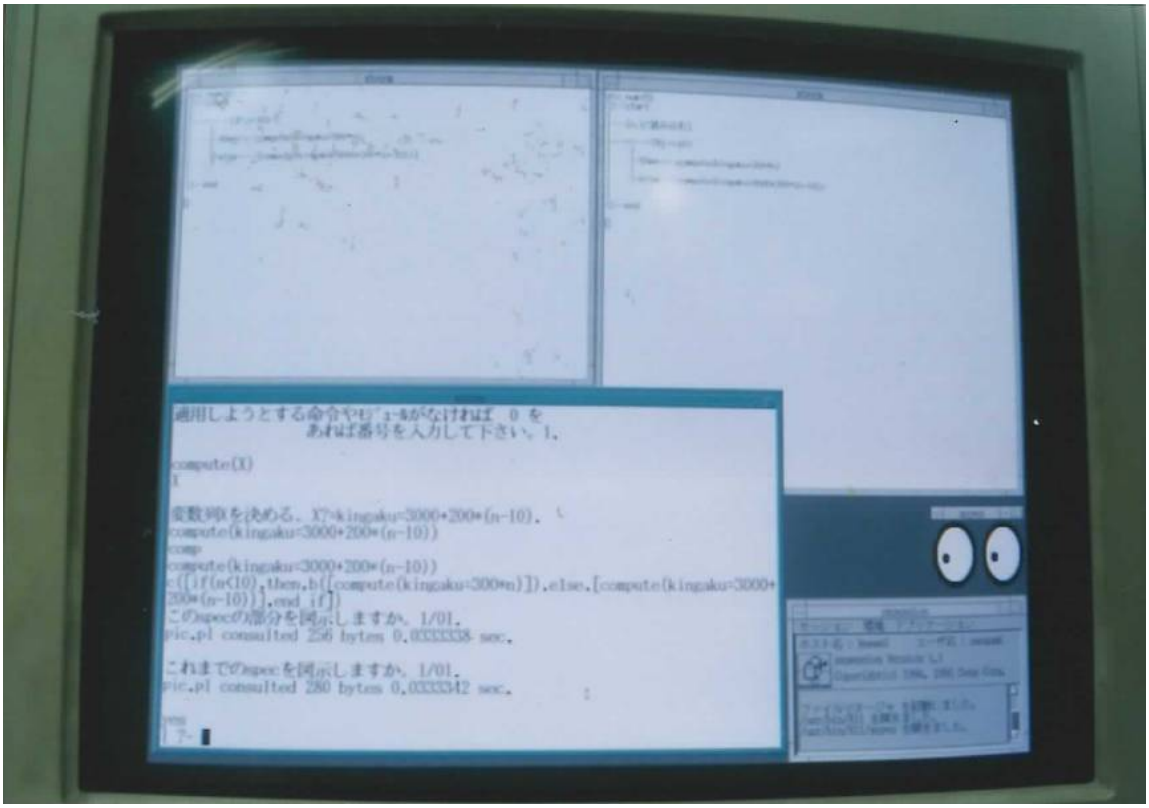


図 A-4　制御文の手続き文を全体の設計文書に追加した時の画面.

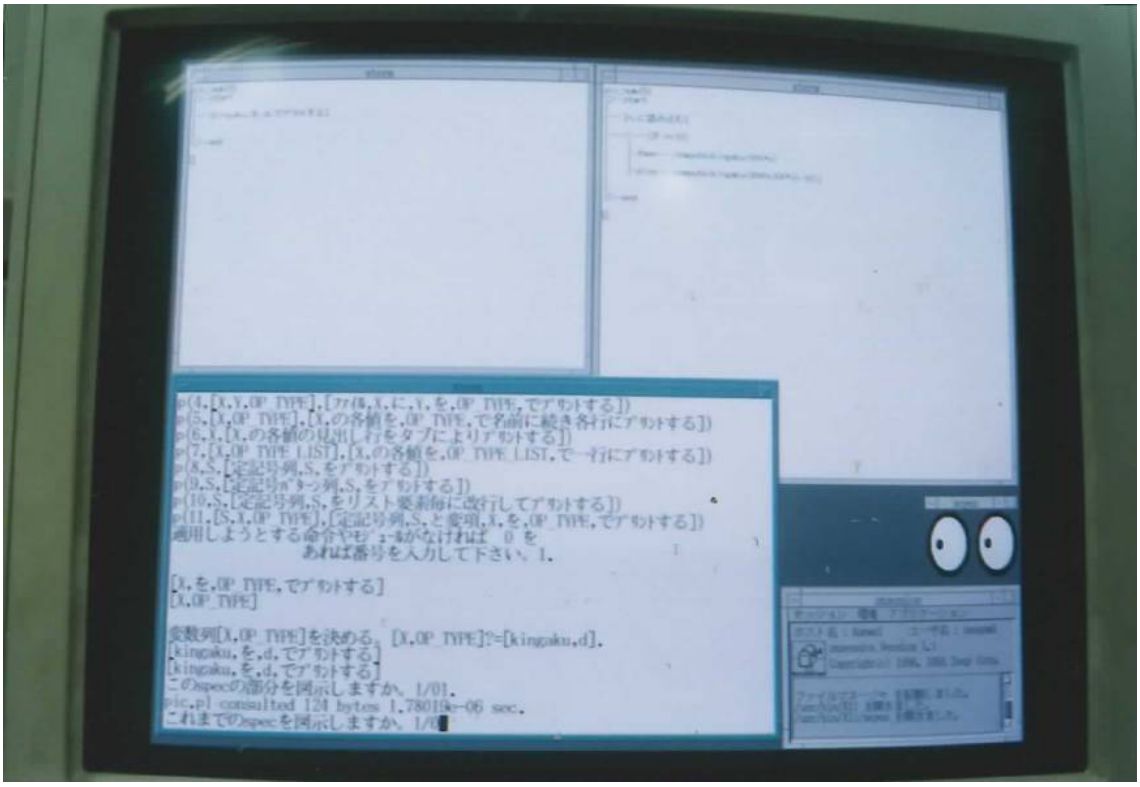


図 A-5　最後の手続き文（印字）を設計した時の画面.

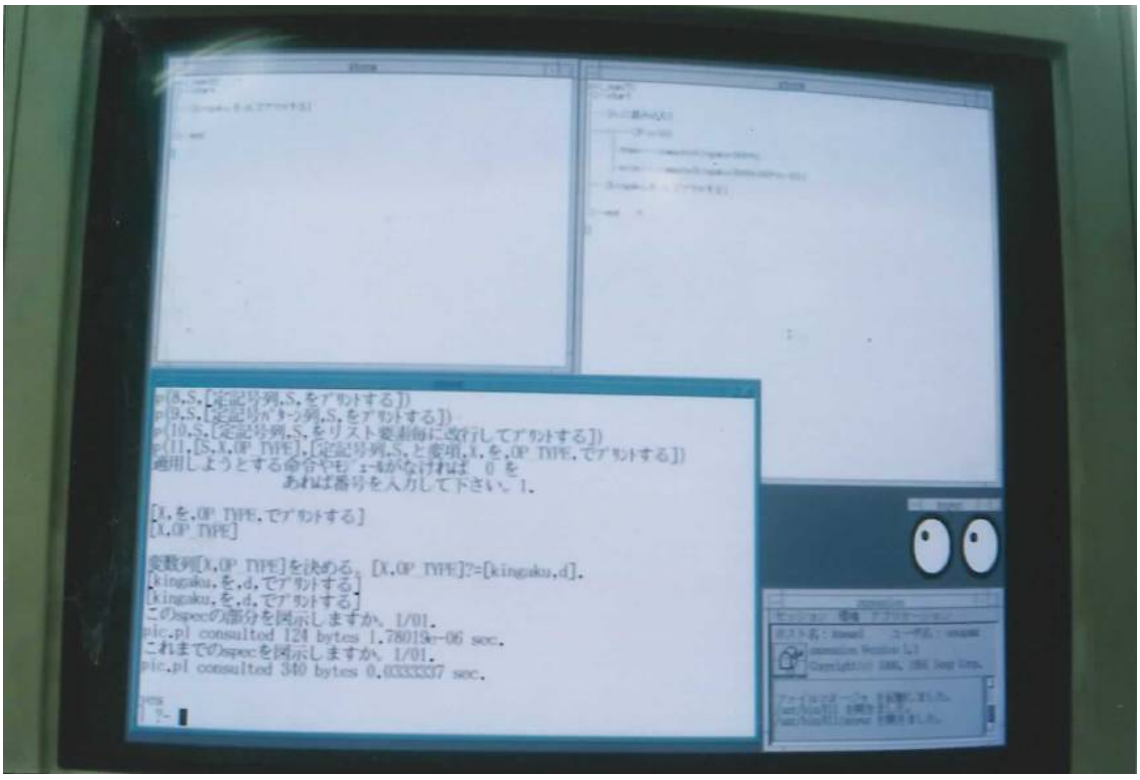


図 A-6　最終的に印字の手続き文が全体の設計文に追加された時の表示画面.

付　録 C

MAPP のプログラムリスト

C-1（手続文書設計）

```
/* 仕様、プログラムのダイアグラム化１ */

insert(0,X,L,[X|L]).
insert(N,X,[Y|L],[Y|XL]):-N >0,N1 is N-1,insert(N1,X,L,XL).
ins(X,H,[H|T],[X|[H|T]]).
ins(X,H,[Y|L],[Y|XL]):-ins(X,H,L,XL).
delete(1,[X|Y],Y).
delete(N,[X|Y],[X|Z]):-N>1,N1 is N-1,delete(N1,Y,Z).
del(X,[X|Y],Y).
del(X,[Y|Z],[Y|W]):-del(X,Z,W).
replace(1,X,[H|T],[X|T]).
replace(N,X,[Y|L],[Y|XL]):-N>1 -> N1 is N-1,replace(N1,X,L,XL).
replace_append(1,X,[H|T],XL):-append([ref(H)],X,R1),append(R1,[end_ref],R2),
                              append(R2,T,XL).
replace_append(N,X,[Y|L],[Y|XL]):-N>1 -> N1 is N-1,replace_append(N1,X,L,XL).

replace_append_list([NUM],X,SP,XT):-replace_append(NUM,X,SP,XT).
replace_append_list([1|TN],X,[[ref(H)|HR]|T],[XR2|T]):-
                        replace_append_list(TN,X,HR,XR),
                        append([ref(H)],XR,XR1),append(XR1,[end_ref],XR2).
replace_append_list([HN|TN],X,[Y|L],[Y|XL]):-HN>1 -> HN1 is HN-1,
                        replace_append_list([HN|TN],X,L,XL).
pic_list([H|T],[OH|OT]):-pic_ref(H,OH),pic_list(T,OT).
pic_list([],[]).
pic_ref(ref(S,R),R4):-pic_list(R,R2),
                      append([ref(S)],R2,R3),
                      append(R3,[end_ref],R4).
pic_ref(X,X).

refine(NSP):-spec(NSP,SP),out_sp(c(SP)),
             write('詳細化する番号を入れて下さい'),read(NUM),
             retr(NUM,SP,RETR),write(RETR),nl,
             write('詳細化する内容を入れて下さい'),read(REF),
             retr(NUM,SP,X),
             replace_append(NUM,REF,SP,RSP),out_sp(c(RSP)).
retr(1,[H|T],H).
retr(NUM,[H|T],RT):-NUM>1 -> NUM1 is NUM-1,retr(NUM1,T,RT).
retr_list([HN|TN],SP,RT):-retr(HN,SP,ref(RT1)),retr_list(TN,RT1,RT).
retr_list([N],SP,RT):-retr(N,SP,RT).

spec(1,[a,b,ref(c),c1,c2,end_ref,d]).
spec(2,[a,b,ref(c),c1,ref(c2),c21,c22,end_ref,c3,end_ref,d]).


qi(Z):-insert(3,[a1,a2,a3],[a,b,c,d],Z),
       write('insert(3,[a1,a2,a3],[a,b,c,d],Z)').
qins(Z):-ins([c1,c2],d,[a,b,d,e],Z),
         write('ins([c1,c2],d,[a,b,d,e],Z)').
qd(Z):-delete(3,[a,[b,c],[dd,d1,cd2],[e1,e2,e3]],Z),
       write('delete(3,[a,[b,c],[dd,d1,cd2],[e1,e2,e3]],Z)').
qdel(Z):-del([a,b],[c,d,x,[a,b],y],Z),
      write('del([a,b],[c,d,x,[a,b],y],Z)').
qr(Z):-replace(3,[c1,c2,c3],[a,b,c,d,e],Z),
       write('replace(3,[c1,c2,c3],[a,b,c,d,e],Z)').
qra1(R):-replace_append(3,[z11,z12],[x,y,z,u],R),
         write('replace_append(3,[z11,z12],[x,y,z,u],R)').

qpic1:-out_sp(c([a,b,ref(c),c1,c2,end_ref,d])),
       write('out_sp(c([a,b,ref(c),c1,c2,end_ref,d]))').
qpic2:-out_sp(c([a,b,ref(c),c1,ref(c2),c21,c22,end_ref,c3,end_ref,d])),
       write('out_sp(c([a,b,ref(c),c1,ref(c2),c21,c22,end_ref,c3,end_ref,d]))').

                                          replace_append_list([HN1|TN],X,L,XL).
qr1(R):-replace_append_list([3],[z11,z12],[x,y,z,u],R),
        write('replace_append_list([3],[z11,z12],[x,y,z,u],R)').
qr2(R):-replace_append_list([5],[z21,z22],[x,y,ref(z),z1,z2,end_ref,u],R),
        write('replace_append_list([5],[z21,z22],
                                   [x,y,ref(z),z1,z2,end_ref,u],R)').

qrp1(R):-replace_append_list([3],[z11,z12],[x,y,z,u],R),pic_list(R,PIC_R),
         out_sp(c(PIC_R)),
         write(' replace_append_list([3],[z11,z12],[x,y,z,u],R),
                 pic_list(R,PIC_R),out_sp(c(PIC_R))').
qrp2(R):-replace_append_list([5],[z21,z22],[x,y,ref(z),z1,z2,end_ref,u],R),
         pic_list(R,PIC_R),out_sp(c(PIC_R)),
         write('replace_append_list([5],[z21,z22],
                [x,y,ref(z),z1,z2,end_ref,u],R),
                pic_list(R,PIC_R),out_sp(c(PIC_R))').
qrp3(X):-qrp2(R),replace_append_list([6],[z211,z212],R,X),pic_list(X,XP),
         out_sp(XP),write('qrp2(R),replace_append_list([6],[z211,z212],R,X),
                           pic_list(X,XP),out_sp(XP)').


qpr1(R4):-pic_ref(ref(z,[z1,z2]),R4).
qpr2(R4):-pic_ref( ref(z,[ref(z1,[z11,z12]), z2]), R4 ).
qrefine(NUM).



help:-
 write('スペック'),nl,
 write('   手続きのスペツクを行うには     add_sp(KEY)       を入力  '),nl,
 write('   KEY,には, read,print,comp,return,sum,av,dev,max,min,sort,..'),nl,
 write('                if,repeat,... などがある'),nl,
 write('セーブ'),nl,
 write('   ﾃﾞｰﾀﾍﾞｰｽ上の        ﾊﾞｯﾌｧ sp_all の上の  '),nl,
 write('   一般, メイン,      関数            のプログラムにそれぞれ'),nl,
 write('   NUM,  NUM,           FUNCT_NAME   の名前をつけてファイル'),nl,
 write('   FN1,  main_pr0,    fn_pr0        にセーブするには,'),nl,
 write('                                      それぞれ'),nl,
 write('   reg_sp(NUM,FN),  reg_msp(NUM), reg_fsp(FUNCT_NAME)'),nl,
 write('                                      と入力する'),nl,
 write(整備),nl,
 write('   FN,       main_pr0, fn_pr0       のファイル上の設計文書を prog'),nl,
 write('                                      ﾌﾟﾛｸﾞﾗﾑ展開用に整備して、それ'),nl,
 write('   PFN,      main_pr,  fn_pr        それファイルに出力するには'),nl,
 write('   expnd_src(NUM,FN,PFN), expnd_m_src(NUM), expnd_f_src(FN_NAME)'),nl,
 write('                                      と入力する'),nl,
 write(構造化図表示),nl,
```

```
 write('      ファイル名    FN,  番号      NUM    の設計文書の内容を図示するには'),
                                                    nl,
 write('        draw_pic_sp(NUM,FN)                 '),nl,
 write('                                            と入力する'),nl,
 write('     FN は１０月現在では   main_prp,NUM は 13,14,15,16,17,18,19,20 '),nl.

/*   1.  基本述語   */
pic_cls:-tell('pic.pl'),d_clear,told.
expand_list([H|T]):-expand(H),nl,expand_list(T).
expand_list([]).
expand(c([H|T])):-writeq(H),write(','),expand_list(T).
expand(b([c([H|T])])):-writeq(H),expand_list(T).
expand(b([H|T])):-expand(H),expand_list(T).
expand(H):-writeq(H),write(','),nl.
append([],X,X).
append([A|X],Y,[A|Z]):-append(X,Y,Z).
member(X,[X|L]).
member(X,[Y|L]):-member(X,L).
reverse([],[]).
reverse([A|X],Z):-reverse(X,XR),append(XR,[A],Z).

writelist([]).
writelist([H|T]):-write(H),writelist(T).
writeqlist([]).
writeqlist([H|T]):-writeq(H),writeqlist(T).
writelist_hc([]).
writelist_hc([H|T]):-write(','),write(H),writelist_hc(T).
writelist_ic([]).
writelist_ic([H|T]):-write(H),writelist_hc(T).
writelist_com([H|T]):-write(H),write(','),writelist_com(T).
writelist_com([]).
writelist_nl([H|T]):-write(H),nl,writelist_nl(T).
writelist_nl([]).

writelist_hc_nl([H|T]):-write(','),nl,write_out(H),writelist_hc_nl(T).
writelist_hc_nl([]).
writelist_ic_nl([H|T]):-write_out(H),writelist_hc_nl(T).
writelist_ic_nl(X):-write_out(X).
write_out(c([H|T])):-write('['),writeq(H),write(','),nl,writelist_ic_nl(T),
                        write(']').
write_out(b([H|T])):-write('['),write_out(H),writelist_hc_nl(T),
                        write(']').
write_out(X):-writeq(X).

writelist_list([H|T]):-writelist(H),writelist_list(T).
writelist_list([]).
writelist_list_nl([H|T]):-(writelist_nl(H);write(H)),nl,writelist_list_nl(T).
writelist_list_nl([]).

writelist_list_hc_nl([H|T]):-write(','),nl,(writelist_ic_nl(H);write(H)),
                                    writelist_list_hc_nl(T).
writelist_list_hc_nl([]).
writelist_list_ic_nl([H|T]):-(writelist_ic_nl(H);write(H)),
                                    writelist_list_hc_nl(T).


/*  2.  スペックのファイルへのセーブ、整備、表示 */

pic_sp([]).

ra_sp:-retract(sp_all(X)),assert(sp_all([])).
sp_all([]).

reg_msp(NUM):-
        tella(main_pr0),sp_all(X),writelist(['sp_rec(',NUM,',']),
        writeq(X),write(').'),nl,told.
reg_fsp(FN_NAME):-
        tella(fn_pr0),sp_all(X),writelist(['sp_rec(',FN_NAME,',']),
        writeq(X),write(').'),told.

reg_sp(NUM,FN):-
        tella(FN),sp_all(X),writelist(['sp_rec(',NUM,',']),
        writeq(X),write(').'),told.

expnd_m_src(NUM):-
    [main_pr0],sp_rec(NUM,X),
    writelist(['process(spec(',NUM,'))':-proc_list([']),
    writelist_ic_nl(X),write(']).'),nl,
    tella(main_prc),writelist(['process(spec(',NUM,'))':-proc_list([']),
    writelist_ic_nl(X),write(']).'),nl,told,nl.
expnd_f_src(FN_NAME):-[fn_pr0],sp_rec(FN,X),
    writelist(['process(',FN_NAME,')':-proc_list([']),
    writelist_ic_nl(X),write(']).'),nl,
    tella(fn_pr),writelist(['process(',FN_NAME,')':-proc_list([']),
    writelist_ic_nl(X),write(']).'),told,nl.
expnd_src(NUM,FN,PFN):-[FN],sp_rec(NUM,X),
    writelist(['process(spec(',NUM,'))':-proc_list([']),
    writelist_ic_nl(X),write(']).'),nl,
    tella(PFN),writelist(['process(spec(',NUM,'))':-proc_list([']),
    writelist_ic_nl(X),write(']).'),told,nl.

draw_pic:-['pic.pl'],pic_num(N),write(pic_num(N)),nl,pic(N),nl,nl,N1 is N+1,
          retract(pic_num(N)),assert(pic_num(N1)).

draw_pic_sp(NUM,FN):-[FN],sp_rec(NUM,X),tell('pic.pl'),
        writelist([pic(NUM),':-pic_sp([start,']),expand_list(X),
        write('sp_end]).'),told,['pic.pl'],pic(NUM).
/*
sp_pic(N):-tell('pic.pl'),
        writelist([pic(N),':-pic_sp([start,']),
        proc_sp(N),write('sp_end]).'),told,
        ['pic.pl'],pic(N).
*/
pic_head(N):-tell('pic.pl'),
        writelist([pic(N),':-pic_sp([start,']).

pic_end:-write('sp_end]).'),told.
pic_num(0).

/*  3.  制御文スペック   */
ra_t(X):-retract(tmp(Y)),assert(tmp(X)).
qrf(O,N,SP):-out_sp(c(SP)),repl(O,N,SP,RSP),out_sp(c(RSP)).
refine_pic(NUM,N,SP,RSP):-replace(NUM,N,SP,RSP),out_sp(c(RSP)),
                          retract(tmp(X)),assert(tmp(RSP)).
spc(if(T),c([if(T),then,b(SP),end_if]) ):-
        pic_num(N),pic_head(N),expand(c([if(T)])),
        pic_end,
        write('条件部を図示しますか。1/0'),read(COND),(COND=:=1,
        tell('spec_ctr.pic'),draw_pic,told;write(' ')),
        write('条件文の then の本体部を入力して下さい'),nl,read(P),
        out_sp(c([ if(T),then,b(P),end_if])),nl,
        spc_list(P,SP),
        out_sp(c([ if(T),then,b(SP),end_if])),nl.

 out_sp(OUT_SP):-
        write(OUT_SP),nl,
```

```
        pic_num(N1),pic_head(N1),
        expand(OUT_SP),nl,
        pic_end,
        write('この spec の部分を図示しますか。1/0'),read(COND),(COND=:=1,
        tell('spec_ctr.pic'),draw_pic,told;write(' ')).   /*   ???   question */

spc(if_else(T),c([if(T),then,b(SP1),else,b(SP2),end_if]) ):-
        pic_num(N),pic_head(N),expand(c([if(T)])),
        pic_end,write('条件部を図示しますか。1/0'),read(COND),(COND=:=1,
        tell('spec_ctr.pic'),draw_pic,told;write(' ')),
        write('条件文の then の本体部を入力して下さい'),nl,read(P1),
        spc_list(P1,SP1),
        write(c([ if(T),then,b(SP1),end_if] )),nl,
        pic_num(N1),pic_head(N1),
        expand(c([ if(T),then,b(SP1),end_if] )),nl,
        pic_end,
        pic_end,write('if 文部を図示しますか。1/0'),read(IF),(COND=:=1,
        tell('spec_ctr.pic'),draw_pic,told;write(' ')),
        write('条件文の else の本体部を入力して下さい'),nl,read(P2),
        spc_list(P2,SP2),
        out_sp(c([ if(T),then,b(SP1),else,SP2,end_if] )),nl.



spc(case(INDEX_NAME),c([case(INDEX_NAME),b(SP),end_case_list]) ):-
        write(case(INDEX_NAME)),nl,
        write('case 文の本体部を入力して下さい'),nl,read(P),
        case_list(P,SP),
        out_sp(c([case(INDEX_NAME),b(SP),end_case_list] )),nl.

spc(if_else_list,c([if_else_list,b(SP),end_if_else_ilist]) ):-
        pic_num(N),pic_head(N),expand(c([case(INDEX)])),
        pic_end,write('条件部を表示しますか。1/0'),read(COND),(COND=:=1,
        tell('spec_ctr.pic'),draw_pic,told;write(' ')),
        write('if_else_list 文の本体部を入力して下さい'),nl,read(P),
        write(P),
        if_else_list(P,SP),
        write(c([if_else_list,b(SP),end_if_else_list] )),nl,
        pic_num(N1),pic_head(N1),
        expand(c([if_else_list,b(SP),end_if_else_list] )),
        pic_end.

spc(when(OBJ_NAME),c([when(OBJ_NAME),b(SP),end_when_list]) ):-
        write(when(OBJ_NAME)),nl,
        write('when_list 文の本体部を入力して下さい'),nl,read(P),
        when_list(P,SP),
        out_sp(c([when(OBJ_NAME),b(SP),end_when_list] )),nl.

spc(for([INDEX,INIT_VAL,TERM_VAL,STEP]),c([for([INDEX,を,INIT_VAL,
        から,TERM_VAL,まで,STEP,づつ変化しながら繰り返す]),b(SP),end_for])):-
        pic_num(N),pic_head(N),
        expand(c([for([INDEX,を,INIT_VAL,から,TERM_VAL,まで,STEP,
                づつ変化しながら繰り返す])])),
        pic_end,write('条件部を図示しますか。1/0'),read(COND),(COND=:=1,
        tell('spec_ctr.pic'),draw_pic,told;write(' ')),
        write('for の繰り返しの本体部を入力して下さい'),nl,
        read(P),out_sp(c([for([INDEX,を,INIT_VAL,
        から,TERM_VAL,まで,STEP,づつ変化しながら繰り返す]),b(P),end_for])),nl,
        spc_list(P,SP),
        out_sp(c([for([INDEX,を,INIT_VAL,から,TERM_VAL,まで,STEP,
                づつ変化しながら繰り返す]),b(SP),end_for])),nl.

spc(for([INIT,REP_COND,STEP]),
        c([for([初期条件を,INIT,とし,STEP,の処理をしながら,REP_COND,
            の継続条件が成立する間繰り返す]),b(SP),end_for]) ):-
        pic_num(N),pic_head(N),
        expand(c([for([初期条件を,INIT,とし,STEP,の処理をしながら,REP_COND,
                の継続条件が成立する間繰り返す])]) ),
        pic_end,write('条件部を図示しますか。1/0'),read(COND),(COND=:=1,
        tell('spec_ctr.pic'),draw_pic,told;write(' ')),
        write('for の繰り返しの本体部を入力して下さい'),nl,
        read(P),spc_list(P,SP),
        out_sp(c([for([初期条件を,INIT,とし,STEP,の処理をしながら,REP_COND,
                の継続条件が成立する間繰り返す]),b(SP),end_for]) ),nl.

spc(while(T),c([while([T,の間繰り返す]),b(SP),end_while]) ):-
        pic_num(N),pic_head(N),
        expand(c([while([T,の間繰り返す])])),
        pic_end,write('条件部を図示しますか。1/0'),read(COND),(COND=:=1,
        tell('spec_ctr.pic'),draw_pic,told;write(' ')),
        write('while の繰り返しの本体部を入力して下さい'),nl,
        read(P),spc_list(P,SP),
        out_sp(c([while([T,の間繰り返す]),b(SP),end_while])),nl.

spc(until([X,T]),c([until([X,を読み込み,T,が成立するまで繰り返す]),
    b(SP),end_until]) ):-
        pic_num(N),pic_head(N),
        expand(c([until([X,を読み込み,T,が成立するまで繰り返す])])),
        pic_end,write('条件部を図示しますか。1/0'),read(COND),(COND=:=1,
        tell('spec_ctr.pic'),draw_pic,told;write(' ')),
        write('until の繰り返しの本体部を入力して下さい'),nl,
        read(P),spc_list(P,SP),
        out_sp(c([until([X,を読み込み,T,が成立するまで繰り返す]),
        b(SP),end_while])),nl.

body(X,S):-sp(X,S).
body(X,S):-retr(X,S).

retr_list([H|T],[SH|ST]):-retr(H,SH),retr_list(T,ST).
retr_list([],[]).
retr_list(X,SX):-retr(X,SX).
spc_list([H|T],[HS|TS]):-(retrc(H,HC),spc(HC,HS);retr(H,HS)),
                write(H),nl,(write(HS);writelist(HS)),nl,spc_list(T,TS).
spc_list([],[]).
spc_list(X,XS):-retrc(X,XC),spc(XC,XS).

sp_list(X,XS):-retr_list(X,XS),out_sp(b(XS)).
sp_list([H|T],[HS|TS]):-sp(H,HS),sp_list(T,TS).
sp_list([],[]).
sp_list(X,XS):-sp(X,XS).
sp(X,S):-retr(X,S),out_sp(S).



case_list([H|T],[HS|TS]):-case(H,HS),case_list(T,TS).
case_list([],[]).
case(ITEM,c([item(ITEM),b(SP),end_case])):-
        writelist(['case 文 ',ITEM,' の本体部を入力して下さい']),nl,read(P),
        spc_list(P,SP),
        write(c([item(ITEM),b(SP),end_case] )),nl,
        pic_num(N1),pic_head(N1),
        expand(c([item(ITEM),b(SP),end_case] )),
        pic_end,
        write('case 文を図示しますか。1/0'),read(COND),(COND=:=1,
        tell('spec_ctr.pic'),draw_pic,told;write(' ')).
when_list([H|T],[HS|TS]):-when(H,HS),when_list(T,TS).
when_list([],[]).
```

```
when(COND,c([cond(COND),b(SP),end_when])):-
        writelist(['when 文の条件 ',COND,' の本体部を入力して下さい'),nl,read(P),
        spc_list(P,SP),
        write(c([cond(COND),b(SP),end_when] )),nl,
          pic_num(N1),pic_head(N1),
          expand(c([cond(COND),b(SP),end_when] )),
          pic_end,
          write('case 文を図示しますか。1/0'),read(YN),(YN=:=1,
          tell('spec_ctr.pic'),draw_pic,told;write(' ')).


/*  4．手続き文の検索と指定  */

add_sp(X):-(retrc(X,XC),spc(XC,CUR_SP);spc(X,CUR_SP);sp(X,CUR_SP)),
          sp_all(SP_ALL),append(SP_ALL,[CUR_SP],N_SP_ALL),
          retract(sp_all(SP_ALL)),assert(sp_all(N_SP_ALL)),
          pic_num(N2),pic_head(N2),
          sp_all(N_SP_ALL),
          expand_list(N_SP_ALL),nl,
          pic_end,
          write('これまでの spec を図示しますか。1/0'),read(COND),(COND=:=1,
          tell('spec_all.pic'),draw_pic,told;write(' ')).

retrc(H2,CX):-
          dict2_retr(DICT2_R),member([head(H1,H2),body(T)],DICT2_R),
          writelist_nl(T),
          write('適用しようとする命令やモジュールがなければ 0 を,
                  あれば番号を入力して下さい。'),
          read(N),(N=:=0;
                    member(p(N,X,C,CST),T),nl,
                    writelist_nl([C,X]),nl,
                    writelist([変数列,X,を決める。]),
                    write('テスト述語を検索適用するときには,1,を,
                            しないときには,0,を入力して下さい'),nl,read(YN),
                    (YN=:=1,retr(test,SX);
                    sp_var(X,SX)),qspc(H2,N,SX,CX)    ),
                    dict2(DICT2),member([head(H11,H12),body(T1)],DICT2),
                    (member(p(N,SX,CX,c([macro(CH1),B,end_macro])),T1),
                    write(macro(CH1)),nl,macro(CH1);write(' ')).


qspc(H,N,SX,CX):-
            dict2(DICT2),member([head(H1,H),body(T)],DICT2),
            member(p(N,SX,CX,CST),T),write(CX),nl.

retr(H2,STX):-
            dict1_retr(DICT1),member([head(H1,H2),body(T)],DICT1),
            writelist_nl(T),
           write('適用しようとする命令やモジュールがなければ  0  を
                  あれば番号を入力して下さい。'),
           read(N),(N=:=0;
                    member(p(N,X,ST),T),nl,
                    writelist_nl([ST,X]),nl,
                    writelist([変数列,X,を決める。]),
                    sp_var(X,SX),qsp(H2,N,SX,STX)).

retr(H2,ST2):-
            (dict1_retr(DICT1),member([head(H1,H2),body(T)],DICT1);
            dict2_retr(DICT2),member([head(H1,H2),body(T)],DICT2)),
            writelist_nl([H2,'は見つかりません。入力ミス？']),nl.
sp_var(X,SX):-writelist([X,'?=']),read(SX).
sp_var([],[]).

qsp(H,N,SX,STX):-
            dict1(DICT1),member([head(H1,H),body(T)],DICT1),
            member(p(N,SX,STX),T),(write(STX);writelist(STX)),nl.


/*  5．  見出し辞書 */

dict1_retr([
    [head(読み込む,read),
      body([p(1,'X',['X',に読み込む]),
            p(2,'X',['プロンプトにより','X',に読み込む]),
            p(3,'X',['プロンプトにより','X',の各変数に読み込む]),
            p(4,'X',[プロンプトにより１次元配列,'X',に読み込む]),
            p(5,'X',[プロンプトにより２次元配列,'X',に読み込む]),
            p(6,'X',[ファイルから,'X',に読み込む]),
            p(7,'X',['FN','K','A','J'],[ファイル名,'FN',の第,'K',
                番目のファイルから,２次元配列,'A',の第,'J',列に読み込む]) ])],
    [head(書き出す,print),
      body([p(1,['X','OP_TYPE'],['X',を,'OP_TYPE',でプリントする]),
            p(2,['X','OP_TYPE'],[１次元配列,'X',をプリントする]),
            p(3,['X','OP_TYPE'],[２次元配列,'X',をプリントする]),
            p(4,['X','Y','OP_TYPE'],[ファイル,'X',に,'Y',を,'OP_TYPE',
                でプリントする]),
            p(5,['X','OP_TYPE'],['X',の各値を,'OP_TYPE',
                で名前に続き各行にプリントする]),
            p(6,'X',['X',の各値の見出し行をタブによりプリントする]),
            p(7,['X','OP_TYPE_LIST'],['X',の各値を,'OP_TYPE_LIST',
                で一行にプリントする]),
            p(8,'S',[定記号列,'S',をプリントする]),
            p(9,'S',[定記号パターン列,'S',をプリントする]),
            p(10,'S',[定記号列,'S',をリスト要素毎に改行してプリントする]),
            p(11,['S','X','OP_TYPE'],[定記号列,'S',と変項,'X',を,'OP_TYPE',
                でプリントする]) ])],
    [head(和,sum),
      body([p(1,['X','Y','Z'],['X',に,'Y',を加えた結果を,'Z',におく]),
            p(2,['OBJ','SUMOBJ'],['OBJ',の和を,'SUMOBJ',に求める]),
            p(3,['OBJ','C','SUMOBJ'],
                [２次元配列,'OBJ',の第,'C',列の和を,'SUMOBJ',に求める]),
            p(4,['OBJ','FROM_C','TILL_C','ROW','SUMOBJ'],
                [２次元配列,'OBJ',の,'FROM_C',列から,'TILL_C',列までの,'ROW',
                 行の要素の和を,'SUMOBJ',に求める]) ])],
    [head(平均値,av),
      body([p(1,['X','Y'],['X',の平均値を,'Y',に求める]),
            p(2,['X','C','Y'],
                [２次元配列,'X',の第,'C',列の平均値を,'Y',に求める]) ])],
    [head(偏差値,dev),
      body([p(1,['OBJ','DEVOBJ'],['OBJ',の偏差値を,'DEVOBJ',に求める]),
            p(2,['OBJ','C','DEVOBJ'],
                [２次元配列,'OBJ',の第,'C',列の偏差値を,'DEVOBJ',に求める]) ])],
    [head(最大値を求める,max),
      body([p(1,['X','Y'],['X',の最大値を,'Y',に求める]),
            p(2,['X','C','Y'],
                [２次元配列,'X',の第,'C',列の最大値を,'Y',に求める]) ])],
    [head(最小値を求める,min),
      body([p(1,['OBJ','MINOBJ'],['OBJ',の最小値を,'MINOBJ',に求める]),
            p(2,['OBJ','C','MINOBJ'],
                [２次元配列,'OBJ',の第,'C',列の最小値を,'MINOBJ',に求める]) ])],
    [head(返す,return),
      body([p(1,'X',['X',を返す]) ])],
    [head(ブレークする,break),
      body([p(1,[],[ブレークする]) ])],
    [head(設定する,set),
      body([p(1,['Y','X'],set('Y','X')) ])],
```

```
    [head(計算する,comp),
        body([p(1,'X',compute('X')),
              p(2,'X',compute_list('X')) ])],
    [head(テストする,test),
        body([p(1,'N',['N',は素数である]),
              p(2,'Y',['Y',は閏年である]) ])],
    [head(乱数発生,random),
        body([p(1,['LOWER','UPPER','NUM','RAN','A'],
                  ['LOWER',から,'UPPER',までの一様乱数を,'NUM',個,'RAN',
                    を seed として発生し配列,'A',に入れる]),
              p(2,['LOWER','UPPER','RAN','A'],['LOWER',
                  から,'UPPER',までの一様乱数を,'RAN',を seed として発生し,
                  'N',行,'M',列の２次元配列,'A',に入れる]) ])],
    [head('ソート,sort),
        body([p(1,'[A,ORD]',['A',を,'ORD',順にソートする]),
              p(2,'[A,ORD,SORT]',
                  [２次元配列,'A',を,ORD,順に第,'K',
                   列の要素をキーとしてソートする]    ])]
]).

 dict1([
    [head(読み込む,read),
        body([p(1,X,[X,に読み込む]),
              p(2,X,[プロンプトにより,X,に読み込む]),
              p(3,X,[プロンプトにより,X,の各変数に読み込む]),
              p(4,X,[プロンプトにより１次元配列,X,に読み込む]),
              p(5,X,[プロンプトにより２次元配列,X,に読み込む]),
              p(6,X,[ファイルから,X,に読み込む]),
              p(7,X,[FN,K,A,J],[ファイル名,FN,の第,K,番目のファイルから,'２次元
                  配列',A,の第,J,列に読み込む]) ])],
    [head(書き出す,print),
        body([p(1,[X,OP_TYPE],[X,を,OP_TYPE,でプリントする]),
              p(2,[X,OP_TYPE],[１次元配列,X,をプリントする]),
              p(3,[X,OP_TYPE],[２次元配列,X,をプリントする]),
              p(4,[X,Y,OP_TYPE],[ファイル,X,に,Y,を,OP_TYPE,でプリントする]),
              p(5,[X,OP_TYPE],[X,の各値を,OP_TYPE,
                  で名前に続き各行にプリントする]),
              p(6,X,[X,の各値の見出し行をタブによりプリントする]),
              p(7,[X,OP_TYPE_LIST],[X,の各値を,OP_TYPE_LIST,
                  で一行にプリントする]),
              p(8,S,[定記号列,S,をプリントする]),
              p(9,S,[定記号パターン列,S,をプリントする]),
              p(10,S,[定記号列,S,をリスト要素毎に改行してプリントする]),
              p(11,[S,X,OP_TYPE],[定記号列,S,と変項,X,を,OP_TYPE,
                  でプリントする]) ])],
    [head(和,sum),
        body([p(1,[X,Y,Z],[X,に,Y,を加えた結果を,Z,におく]),
              p(2,[OBJ,SUMOBJ],[OBJ,の和を,SUMOBJ,に求める]),
              p(3,[OBJ,C,SUMOBJ],
                  [２次元配列,OBJ,の第,C,列の和を,SUMOBJ,に求める]),
              p(4,[OBJ,FROM_C,TILL_C,ROW,SUMOBJ],
                  [２次元配列,OBJ,の,FROM_C,列から,TILL_C,列までの,ROW,
                   行の要素の和を,SUMOBJ,に求める]) ])],
    [head(平均値,av),
        body([p(1,[X,Y],[X,の平均値を,Y,に求める]),
              p(2,[X,C,Y],
                  [２次元配列,X,の第,C,列の平均値を,Y,に求める]) ])],
    [head(偏差値,dev),
        body([p(1,[OBJ,DEVOBJ],[OBJ,の偏差値を,DEVOBJ,に求める]),
              p(2,[OBJ,C,DEVOBJ],
                  [２次元配列,OBJ,の第,C,列の偏差値を,DEVOBJ,に求める]) ])],
    [head(最大値を求める,max),
        body([p(1,[X,Y],[X,の最大値を,Y,に求める]),
              p(2,[X,C,Y],
                  [２次元配列,X,の第,C,列の最大値を,Y,に求める]) ])],
    [head(最小値を求める,min),
        body([p(1,[OBJ,MINOBJ],[OBJ,の最小値を,MINOBJ,に求める]),
              p(2,[OBJ,C,MINOBJ],
                  [２次元配列,OBJ,の第,C,列の最小値を,MINOBJ,に求める]) ])],
    [head(返す,return),
        body([p(1,X,[X,を返す]) ])],
    [head(ブレークする,break),
        body([p(1,[],[ブレークする]) ])],
    [head(設定する,set),
        body([p(1,[Y,X],set(Y,X)) ])],
    [head(計算する,comp),
        body([p(1,X,compute(X)),
              p(2,X,compute_list(X)) ])],
    [head(テストする,test),
        body([p(1,N,[N,は素数である]),
              p(1,Y,[Y,は閏年である]) ])],
    [head(乱数発生,random),
        body([p(1,[LOWER,UPPER,NUM,RAN,A],[LOWER,から,UPPER,までの一様乱数を,
                  NUM,個,RAN,を seed として発生し配列,A,に入れる]),
              p(2,[LOWER,UPPER,RAN,A],[LOWER,
                  から,UPPER,までの一様乱数を,RAN,を seed として発生し,
                  N,行,M,列の２次元配列,A,に入れる]) ])],
    [head(ソート,sort),
        body([p(1,[A,ORD],[A,を,ORD,順に,ソートする]),
              p(2,[A,ORD],[２次元配列,A,を,ORD,順に第,K,
                      列の要素をキーとしてソートする]) ])]
]).


dict2_retr([
[head(もし,if),
    body([
          p(1,'T',if('T'),c([if('T'),then,b('S'),end_if])),
          p(2,'T',if_else('T'),c([if('T'),then,b('S1'),else,b('S2'),end_if])),
          p(3,'OBJ_NAME',when('OBJ_NAME'),c([when('OBJ_NAME'),b('SP'),
                                              end_when_list])),
          p(4,'INDEX_NAME',case('INDEX_NAME'),c([case('INDEX_NAME'),b('SP'),
                                              end_case_list]))
])],
[head(繰り返す,repeat),
    body([
          p(1,['INDEX','INIT_VAL','TERM_VAL','STEP'],
          for(['INDEX','INIT_VAL','TERM_VAL','STEP']),
          c([for(['INDEX',を,'INIT_VAL',から,'TERM_VAL',まで,'STEP',
                  づつ変化しながら繰り返す]),b('SP'),end_for])),
          p(11,['A','I','S'],
          macro(['A',':[A,[I]]',':S',':[A,[0]]']),
          c([macro(['A',のすべての要素,'[A,[I]]',に対して,'S',を更新しながら,
          繰り返す,ただし,'S',の初期値を,'[A,[0]]',とする]),
          b('SP'),end_macro])),
          p(12,['A','S','INIT'],
          for(['A','S','INIT']),
          c([for([キーボードからデータを変数,'A',に読み込み,'S',の計算更新を,
           'EOF',を読み込むまで繰り返す,
            ただし,'S',の初期値を,'INIT',とする]),b('SP'),end_for])),
          p(2,['INIT','REP_COND','STEP'],for(['INIT','REP_COND','STEP']),
              c([for([初期条件を,'INIT',とし,'STEP',の処理をしながら,'REP_COND',
                      の継続条件が成立する間繰り返す]),b('SP'),end_for])),
          p(3,'T',while('T'),
              c([while([条件,'T',が成立する間繰り返す]),b('SP'),end_while])),
          p(4,['X','T'],until(['X','T']),c([until(['X',を読み込み,'T',
                          なる条件が成立するまで繰り返す]),b('SP'),end_until]))
```

```
])] ]).

dict2([
[head(もし,if),
    body([p(1,T,if(T),c([if(T),then,b(S),end_if])),
          p(2,T,if_else(T),c([if(T),then,b(S1),else,b(S2),end_if])),
          p(3,OBJ_NAME,when(OBJ_NAME),c([when(OBJ_NAME),b(SP),
                                         end_when_list])),
          p(4,INDEX_NAME,case(INDEX_NAME),c([case(INDEX_NAME),b(SP),
                                         end_case_list]))
])],
[head(繰り返す,repeat),
    body([p(1,[INDEX,INIT_VAL,TERM_VAL,STEP],
               for([INDEX,INIT_VAL,TERM_VAL,STEP]),
            c([for([INDEX,を,INIT_VAL,から,TERM_VAL,まで,STEP,づつ変化しながら,
               繰り返す]),b(SP),end_for])),
          p(11,[A,I,S],
          macro([A,[A,[I]],S,[A,[0]] ]),
          c([macro([A,のすべての要素,[A,[I]],に対して,S,を更新しながら,繰り返す,
            ただし,S,の初期値を,[A,[0]],とする]),b(SP),end_macro])),
          p(12,[A,S,INIT],
          for([A,S,INIT]),
          c([for([キーボードからデータを変数,A,に読み込み,S,の計算更新を,
           'EOF',を読み込むまで繰り返す,
            ただし,S,の初期値を,INIT,とする]),b(SP),end_for])),
          p(2,[INIT,REP_COND,STEP],for([INIT,REP_COND,STEP]),
            c([for([初期条件を,INIT,とし,STEP,の処理をしながら,REP_COND,
                  の継続条件が成立する間繰り返す]),b(SP),end_for])),
          p(3,T,while(T),
            c([while([条件,T,が成立する間繰り返す]),b(SP),end_while])),
          p(4,[X,T],until([X,T]),c([until([X,を読み込み,T,
                                 なる条件が成立するまで繰り返す]),b(SP),
              end_until]))
])] ]).

/*  6．スペックの構造化図表示*/
buff([]).
tmp([]).
proc_sp_list([]).

 sp_reg(PROC_NUM):-tell(PROC_NUM),writelist([PROC_NUM,':-proc_list(['),
          proc_sp_list(X),writelist_ic_nl(X),write(']).'),told.

sp_pic(N):-tell('pic.pl'),
          writelist([pic(N),':-pic_sp([start,']),
          proc_sp_list(X),writelist_ic_nl_pic(X),write(',sp_end]).'),told,
          ['pic.pl'],pic(N).
pic_cls:-tell('pic.pl'),d_clear,told.
pic_sp([H|T]):-statement_p(H),pic_sp(T).
pic_sp([]).
pic_sp_e([H|T]):-je(H,HE),statement_p(HE),pic_sp_e(T).
pic_sp_e([]).
pic_sp_p([H|T]):-je(H,HE),writelist([HE,',']),nl,pic_sp_p(T).
pic_sp_p([]).
reset:-retract(level(U)),assert(level(0)),
          retract(con(X)),assert(con(down)),
          retract(else(Y)),assert(else(n)).


/*  6.24配布資料　仕様、プログラムのダイアグラム化に続く */

level(0).
space(4).
con(down).
else(n).
ral:-retract(level(L)),assert(level(0)).
statement_p(start):-writelist(['○','--']),write('start'),nl.
arg(if(T)):-
          write_symbol('-',3),
          level(L),L1 is L+1,retract(level(L)),assert(level(L1)),
          writelist(['<>---(IF:',T,')']),nl.
arg(ref):-
          /* write_symbol('-',5),
           */   level(L),L1 is L+1,retract(level(L)),assert(level(L1)),
            retract(con(X)),assert(con(right)).
arg(shift):-write_symbol(' ',3),
          level(L),L1 is L+1,retract(level(L)),assert(level(L1)).
arg(case(ITEM)):-
          write_symbol('-',3),
          level(L),L1 is L+1,retract(level(L)),assert(level(L1)),
          writelist(['<>---(CASE:',ITEM,')']),nl.
arg(item(ITEM)):-
          writelist([' ','(OF:',ITEM,')']),nl,
          level(L),space(IS),
          write_level_line(L),
          L1 is L+1,retract(level(L)),assert(level(L1)),
          write_symbol('-',6),
          retract(con(X)),assert(con(right)).
arg(when(WHEN_ITEM)):-
          write_symbol('-',3),
          level(L),L1 is L+1,retract(level(L)),assert(level(L1)),
          writelist(['<>---(WHEN:',WHEN_ITEM,')']),nl.
arg(cond(COND)):-
          writelist([' ','(COND:',COND,')']),nl,
          level(L),space(IS),
          write_level_line(L),
          L1 is L+1,retract(level(L)),assert(level(L1)),
          write_symbol('-',6),
          retract(con(X)),assert(con(right)).
arg(then):-
          write_symbol('-',1),write('then'),write_symbol('-',1),
          level(L),L1 is L+1,retract(level(L)),assert(level(L1)),
          retract(con(X)),assert(con(right)).
arg(while(T)):-
          write_symbol('-',2),
          level(L),L1 is L+1,retract(level(L)),assert(level(L1)),
          writelist([' ()  --(WHILE:',T,')']),nl.
arg(for(T)):-
          write_symbol('-',2),
          level(L),L1 is L+1,retract(level(L)),assert(level(L1)),
          writelist([' ()  --(FOR:',T,')']),nl.
arg(until(T)):-
          write_symbol('-',1),write('until'),write_symbol('-',1),
          level(L),L1 is L+1,retract(level(L)),assert(level(L1)),
          writelist([' ()  -?:',T]),nl.
arg(P):-writelist(['--:',P]),nl.
statement_p(else):-retract(else(YN)),assert(else(y)),
          level(L),L1 is L-1,retract(level(L)),assert(level(L1)),
          level(L1),write_level_line(L1),nl,
          write_level_line(L1),
          write_symbol('-',1),write('else'),write_symbol('-',1),
          level(L1),L2 is L1+1,retract(level(L1)),assert(level(L2)),
          retract(con(X)),assert(con(right)).
statement_p(end_if):-level(L),
   (else(y),L2 is L-2,retract(level(L)),assert(level(L2)),
    retract(else(y)),assert(else(n));
```

```
    else(n),L1 is L-2,retract(level(L)),assert(level(L1))).
statement_p(end_case):-level(L),L1 is L-1,retract(level(L)),assert(level(L1)).
statement_p(end_ref):-level(L),L1 is L-1,retract(level(L)),assert(level(L1)).
statement_p(end_case_list):-
  level(L),L1 is L-1,retract(level(L)),assert(level(L1)).
statement_p(end_when):-level(L),L1 is L-1,retract(level(L)),assert(level(L1)).
statement_p(end_when_list):-
  level(L),L1 is L-1,retract(level(L)),assert(level(L1)).
statement_p(end_while):-level(L),L1 is L-1,retract(level(L)),assert(level(L1)).
statement_p(end_for):-level(L),L1 is L-1,retract(level(L)),assert(level(L1)).
statement_p(end_until):-level(L),L1 is L-1,retract(level(L)),assert(level(L1)).
statement_p(sp_end):-retract(level(X)),assert(level(0)),
                    write('|'),nl,write('|'),nl,write('〇--end').
statement_p(ref(P)):-level(L),space(IS),
 (con(right),arg(P),retract(con(right)),assert(con(down));
  write_level_line(L),nl,write_level_line(L),arg(P)),
  level(L),L1 is L+1,retract(level(L)),assert(level(L1)).
statement_p(P):-level(L),space(IS),
 (con(right),arg(P),retract(con(right)),assert(con(down));
  write_level_line(L),nl,write_level_line(L),arg(P)).
write_level_line(N):-write('|'),write_level_line_seq(N).
write_level_line_seq(N):-N>0,N1 is N-1,space(IS),
                    write_symbol(' ',IS),write('|'),write_level_line_seq(N1).
write_level_line_seq(0).
write_symbol(X,N):-N>0,write(X),N1 is N-1, write_symbol(X,N1).
write_symbol(X,0).
write_symbol_nl(X,N):-N>0,write(X),nl,N1 is N-1,write_symbol_nl(X,N1).
write_symbol_nl(X,0).


/*  簡易日英表現変換１ */

je([A,をプリントする],print(A)).
je([A,を読む],read(A)).
je([A,を,B,から読む],read(A,B)).
je(for([M,から,N,までの,I,に対して]),for([I,M,N])).
je([A,を,B,から,C,で読む],read(A,B,C)).
je(while([COND,の間]),while(COND)).
je(X,X).


          /* チェック入力  */
/*   () 《》【】@〇◎□△▽※① ［］〔〕 「」    */

add:-read(X),tm(TM),retract(tm(TM)),assert(tm([X|TM])).
disp:-tm(TM),reverse(TM,R),pic_sp(R).
tm([]).
q20:-pic_sp([start,p,ref,p1,p2,p3,end_ref,q,sp_end]).
q3:-pic_sp([start,a,if(t),then,b,else,c,end_if,d,e]).
q4:-je(X,read(c,d)),writelist(X).

q5:-pic_sp([
              start,
              [a,を読む],
              if(a>0),
              then,
                [a,をプリントする],
              end_if,
              end
             ]).
q6:-pic_sp_e([
              start,
              [a,を読む],
              if(a>0),
                then,
                 [a,をプリントする],
              end_if,
              end
      ]).
q7:-pic_sp_p([
              start,
              [a,を読む],
              if(a>0),
                then,
                 [a,をプリントする],
              end_if,
              end
      ]).
q8:-je(for([n1,から,n2,までの,i,に対して]),X),write(X).
q9:-pic_sp([
              start,
              [a,を読む],
              while(a>0),
                 [a,をプリントする],
              end_while,
              end
      ]).

q10:-pic_sp([
              start,
              [a,を読む],
              for('i=0 から i<n まで i++して繰り返す'),
                 if('x!=eof'),
                    then,
                       comp('a+=x'),
                 end_if,
                 [a,をプリントする],
              end_for;
              end
      ]).
q11:-pic_sp([
              start,
              [a,を読む],
              for('i=0 から i<n まで i++して繰り返す'),
                 if('x!=eof'),
                      then,comp('a+=x'),comp('b+=y'),
                 else,
                    comp('c+=z'),
                 end_if,
                 [a,をプリントする],
              end_for;
              end
      ]).
q12:-pic_sp([start,
              [プロンプトにより,n,を読み込む],
              if(n<10),
                 then,
                    compute(kingaku=300*n),
                 else,
                    compute(kingaku=3000+270*(n-10)),
              end_if,
              [kingaku,をプリントする],
              end]).
q13:-pic_sp([start,
              [プロンプトにより,kosuu,を読み込む],
                              [プロンプトにより,shotoku,を読み込む],
```

```
                    [shotoku,の平均値を av_shotoku,に求める],
                    [av_shotoku,をプリントする],
                    [shotoku,の最大値を max_shotoku,に求める],
                    [max_shotoku,をプリントする],
        end]).


writelist_hc_nl_pic([H|T]):-
            write(','),nl,write_out_pic(H),writelist_hc_nl_pic(T).
writelist_hc_nl_pic([]).
writelist_ic_nl_pic([H|T]):-write_out_pic(H),writelist_hc_nl_pic(T).
write_out_pic(c([H|T])):-write(H),write(','),nl,writelist_ic_nl_pic(T).
write_out_pic(X):-write(X).
writelist_ic_nl_pic(X):-write_out_pic(X).


/*  *************   *************   addition   *************   ***********  */

/*      変更点，構造化図上で修正（挿入）ができるようになった.

        rep_sp(Key) で作成した仕様を rep(N) (N は文番号) で
        入れ換える.

          修正箇所には ******* addition ******** をコメントしてある.
            （検索キーに指定すれば，OK.）

        また，構造化図は  spec_all.pic が  全体 の構造化図
                          spec_ctr.pic が 制御文の構造化図
                          r_sp_all.pic が 修正用の構造化図

        但し，制御文のファイル表示は再規定な定義のため  ネストは不可？
*/


/*  2．スペックのファイルへのセーブ、整備、表示 (addition)  */

pic_head_rep(N):-tell('pic_rep.pl'),
        writelist([pic(N),'.'-pic_sp([start,'])).


/*  3．1  ｇｒｏｂａｌ文 スペック    addition   */

glbl(X,[GL_SP]):-write('対応するキーワードが存在しないため、拡張モジュールとして入力しますか。
1/0'),
      read(COND),(COND=:=1,nl,write('【拡張モジュール入力】 : '),write(X),nl,
                  (read(GL_SP))).


/*  4．手続き文の検索と指定  (addition)  */

add_sp(X):-(retrc(X,XC),spc(XC,CUR_SP);spc(X,CUR_SP);sp(X,CUR_SP);glbl(X,CUR_SP)),
        sp_all(SP_ALL),append(SP_ALL,[CUR_SP],N_SP_ALL),
        retract(sp_all(SP_ALL)),assert(sp_all(N_SP_ALL)),
        pic_num(N2),pic_head(N2),
        sp_all(N_SP_ALL),
        expand_list(N_SP_ALL),nl,
        pic_end,
        write('これまでの spec を図示しますか。1/0'),read(COND),(COND=:=1,
        tell('spec_all.pic'),draw_pic,told;write(' ')).
                                                /*   ***   addition   ***  */
rep_sp(X):-
      (retrc(X,XC),spc(XC,CUR_SP);spc(X,CUR_SP);sp(X,CUR_SP);glbl(X,CUR_SP)),
        rep_sp_add(REP_SP_ADD),append(REP_SP_ADD,[CUR_SP],N_REP_SP_ADD),
        retract(rep_sp_add(REP_SP_ADD)),assert(rep_sp_add(N_REP_SP_ADD)),
        pic_num(N2),pic_head_rep(N2),
        rep_sp_add(N_REP_SP_ADD),
        expand_list(N_REP_SP_ADD),nl,
        pic_end,
        write('これまでの spec を図示しますか。1/0'),read(COND),(COND=:=1,
        tell('r_sp_all.pic'),draw_pic,told_rep;write(' ')).

replace(1,X,[H|T],[X|T]).
replace(N,X,[Y|L],[Y|XL]):-N>1 -> N1 is N-1,replace(N1,X,L,XL).

rep(N):-sp_all(SP_ALL),rep_sp_add(REP_SP_ADD),
                              replace(N,REP_SP_ADD,SP_ALL,N_SP_ALL),
        retract(sp_all(SP_ALL)),assert(sp_all(N_SP_ALL)),
        pic_num(N2),pic_head(N2),
        sp_all(N_SP_ALL),
        expand_list(N_SP_ALL),nl,
        pic_end,
        write('これまでの spec を図示しますか。1/0'),read(COND),(COND=:=1,
        tell('spec_all.pic'),draw_pic,told;write(' ')),
        write('置き換えた spec をクリアします。1/0'),read(COND),(COND=:=1,
        r_rep;write('    r_rep でクリア'),nl).

r_rep:-retract(rep_sp_add(REP_SP_ADD)).

/*  *** addition ** sp_all のｎ番目の項目を rep_sp_add で置き換える。*****  */
/*  replace(N,X,L,Y).    リストＬの第Ｎリスト要素をＸで置き換えた結果がＹ    */
replace(1,X,[H|T],[X|T]).
replace(N,X,[Y|L],[Y|XL]):-N>1 -> N1 is N-1,replace(N1,X,L,XL).


/**** addition ** sp_all のｎ番目の項目を rep_sp_add で置き換える。******/
rep(N):-sp_all(SP_ALL),rep_sp_add(REP_SP_ADD),
        retract(sp_all(SP_ALL),
                              replace(N,REP_SP_ADD,SP_ALL,N_SP_ALL),
                              assert(sp_all(N_SP_ALL))
        write('置き換えた spec をクリアします。1/0'),read(COND),(COND=:=1,
        retract(rep_sp_add(REP_SP_ADD));write('    r_rep. でクリア'),nl).
        write('これまでの spec を図示しますか。1/0'),read(COND),(COND=:=1,
        tell('spec_all.pic'),draw_pic,told;write(' ')).

/*   ****  addition  ****   修正分の  設計仕様のクリア  */
r_rep:-retract(rep_sp_add(REP_SP_ADD)).
```

C-2（C プログラム生成）

```
/*    プログラムの生成    */

system('stdio.h').
help:-
    write('初めに   ac↓    入力'),nl,
    write('Ｃプログラムへの変換'),nl,
    write(' MPR,          MDT          のﾌﾟﾛｾｼﾞｭｱ・ﾌｧｲﾙ とﾃﾞｰﾀ・ﾌｧｲﾙ に対しては'),nl,
    write(' cons_main(MPR,MDT).↓ に続いて,   cg_main(NUM).↓    と入力.'),nl,nl,

    write(' main_pr,     main_dt      のﾌﾟﾛｾｼﾞｭｱ・ﾌｧｲﾙ とﾃﾞｰﾀ・ﾌｧｲﾙ に対しては'),nl,
    write(' cons_main.↓                に続いて,   cg_main(NUM).↓    と入力   '),nl,nl,

    write('                                 してプログラムに変換する。'),nl,
    write(' 問題番号 NUM は             13,14,15,16,17,18,19,20'),nl,
    write(' 問題番号が変わる度に          ra.↓を入力'),nl.


/*  4．手続き文の検索と指定   */

retr(H2,STX):-
            dict4_retr(DICT4),member([head(H1,H2),body(T)],DICT4),
            writelist_nl(T),
            write('適用しようとする命令やモジュールがなければ   0  を
                  あれば番号を入力して下さい。'),
            read(N),(N=:=0;
                  member(p(N,X,ST),T),nl,
                  writelist_nl([ST,X]),nl,
                  writelist([変数列,X,を決める。]),
                  sp_var(X,SX),qsp(H2,N,SX,STX)).
proc1(X):-statement(S),member(jp(X,Y),S),Y.
proc2(X):-statement(S),member(jp(X,Y),S),write(Y).
qproc1:-proc1([構造体配列,item1,に kb から,index,を用いて,n,
            回繰り返し読み込む]).
qproc2:-proc2([構造体配列,item1,に kb から,index,を用いて,n,
            回繰り返し読み込む]).

str_mod:-retr(struct,STX),statement(S),member(jp(STX,Y),S),write(Y).
gen_str_mod:-retr(struct,STX),statement(S),member(jp(STX,Y),S),Y.

sp_var(X,SX):-writelist([X,'?=']),read(SX).
sp_var([],[]).
qsp(H,N,SX,STX):-
            dict4(DICT4),member([head(H1,H),body(T)],DICT4),
            member(p(N,SX,STX),T),(write(STX);writelist(STX)),nl.

dict4_retr([
  [head(構造体,struct),
   body([
   p(1,['A,N'],[構造体配列,'A',に kb から,'N',回繰り返し読み込む]),
   p(2,['FILE,A,N'],[構造体配列,'A',に,'FILE',から,'N',回繰り返し読み込む]),
   p(3,['A,N'],[構造体配列,'A',から,display に,'N',回繰り返し書き出す]),
   p(4,['FILE,A,N'],[構造体配列,'A',から,'FILE',に,'N',回繰り返し書き出す]),
   p(5,['A,N,KEY_ITEM,ORDER'],[大きさ,'N,の構造体配列,'A',を,'KEY_ITEM',
            をキーとして,'ORDER',順にソートする]),
   p(6,['FILE,N,COND,RETR_AR'],['N',個のﾚｺｰﾄﾞをもつ,'FILE',から,条件,'COND',
            を満たす構造体を表示し構造体配列,'RETR_AR',にコピーする]),
   p(7,['FILE,N,VAL,STRUCT'],['N',個のﾚｺｰﾄﾞをもつ,'FILE',を,メンバの値,'VAL',
            に対して条件,'CODE_LIST',を満たすﾀｸﾞ,'STRUCT',
            の構造体のデータ部分を表示し更新する]) ])] ]).

dict4([
[head(構造体,struct),
   body(
  [p(1,[A,N],[構造体配列,A,に kb から,N,回繰り返し読み込む]),
   p(2,[FILE,A,N],[構造体配列,A,に,FILE,から,N,回繰り返し読み込む]),
   p(3,[A,N],[構造体配列,A,から,display に,N,回繰り返し書き出す]),
   p(4,[FILE,A,N],[構造体配列,A,から,FILE,に,を用いて,N,回繰り返し書き出す]),
   p(5,[A,N,KEY_ITEM,ORDER],[大きさ,N,の構造体配列,A,を,KEY_ITEM,をキーとして,
            ORDER,順にソートする]),
   p(6,[FILE,N,COND,RETR_AR],[N,個のﾚｺｰﾄﾞをもつ,FILE,から,条件,COND,
            を満たす構造体を表示し構造体配列,RETR_AR,にコピーする]),
   p(7,[FILE,N,VAL,STRUCT],[N 個のﾚｺｰﾄﾞをもつ,FILE,を,メンバの値,VAL,
            に対して条件,CODE_LIST,を満たすﾀｸﾞ,STRUCT,
            の構造体のデータ部分を表示し更新する]) ])] ]).

statement([jp([構造体配列,A,に kb から,N,回繰り返し読み込む],
            gen_funct(read_struct_array(A,N)) )]).
statement([jp([構造体配列,A,に,FILE,から,N,回繰り返し読み込む],
            gen_funct(readfile_struct_array(FN,NAME,SIZE)) )]).
statement([jp([構造体配列,A,から,display に,N,回繰り返し書き出す],
            gen_funct(print_struct_array(A,N)) )]).
statement([jp([構造体配列,A,から,FILE,に,N,回繰り返し書き出す],
            gen_funct(printfile_struct_array(A,N,FILE)) )]).
statement([jp([大きさ,N,の構造体配列,A,を,KEY_ITEM,をキーとして,
            ORDER,順にソートする],
            gen_funct(sort_struct_array_a(A,N,KEY,ORDER)) )]).
statement([jp([N,個のﾚｺｰﾄﾞをもつ,FILE,から,条件,COND,
            を満たす構造体を表示し構造体配列,RETR_AR,にコピーする],
            gen_funct(retrfile_to_str_parr(FILE,N,COND,RETR_AR)) )]).
statement([jp([N 個のﾚｺｰﾄﾞをもつ,FILE,を,メンバの値,VAL,に対して条件,CODE_LIST,
            を満たすﾀｸﾞ,STRUCT,の構造体のデータ部分を表示し更新する],
            gen_funct(updatefile_from_trans(FILE,N,STRUCT,CODE_LIST,VAL)) )]).

qupdt_m:-gen_funct(updatefile_from_trans(ret1,n,meibo,
                  ['strcmp(tmp.name,cn)','==',0],cn)).



gen_funct(read_struct(NAME)):-obj([name(NAME),type(struct(STRUCT))]),
            struct_member(STRUCT,L),
            writelist(['void    read_struct(struct,',STRUCT,',',NAME,');']),nl,
            write('{'),nl,c(read_struct(NAME)),write('}'),nl.
ra(X):-retract(Y),assert(X).
point_var(0).
obj([name(item1),type(struct(item))]).
obj([name(item1),type(struct(item)),point_var(0)]).
struct_member(item,[[[name,[20]],string],[price,int],[number,int]]).
/* obj([name(tmp),type(struct(item))]).*/
obj([name(tmp),type(struct(STRUCT))]).
obj([name(retr_ar),type(struct(item))]).
obj([name(retr_ar_m),type(struct(meibo))]).

obj([name(meibo1),type(struct(meibo)),point_var(0)]).
obj([name(meibo1),type(struct(meibo))]).
struct_member(meibo,[[[name,[20]],string],[code,int],[age,int],
    [[tel,[20]],string],[[address,[30]],string] ]).
```

```
gen_funct(read_struct_array(NAME,SIZE)):-
   obj([name(NAME),type(struct(STRUCT))]),
   struct_member(STRUCT,L),write_sp,(point_var(0),
   writelist(['void     read_struct_array(struct ',STRUCT,' ',NAME,'[],int ',SIZE,
   ')']);
   writelist(['void     read_struct_array(struct ',STRUCT,' *',NAME,',int ',SIZE,
   ')']),
   write('{'),nl,add_sp,
/* funct_def([[read_struct_array,void],[[STRUCT,struct],[[NAME,[]],STRUCT],
     [SIZE,int]] ]), begin_body, var_def([[i,int]]),
   [for,[i,を,0,から,SIZE-1,まで,1,づつ変え繰り返す]],
     [構造体タグ,STRUCT,の構造体の配列要素,[NAME,[i]],に読む込む],
   end_for,end_body. */

   type_decl(int,[i]),
   (point_var(0),c(for(i,minus(SIZE,1),1,c(read_struct(NAME,i)) ));
                  c(for(i,minus(SIZE,1),1,
                   [read_struct(NAME),increase(NAME)] ) ),
    subt_sp,write_sp,write('}'),nl.


gen_funct(readfile_struct_array(FN,NAME,SIZE)):-
   obj([name(NAME),type(struct(STRUCT))]),
   struct_member(STRUCT,L),write_sp,(point_var(0),
   writelist(['void     readfile_struct_array(char *',FN,',
                struct ',STRUCT,' ',NAME,'[],int ',SIZE,')']);
   writelist(['void     readfile_struct_array(char *',FN,',
                struct ',STRUCT,' *',NAME,',int ', SIZE,')']),
   write('{'),nl,add_sp,
   type_decl('FILE',[' *fp']),type_decl(int,[i]),
   file_open(FN,fp,"r"),
   (point_var(0),c(for(i,minus(n,1),1,c(readfile_struct(fp,NAME,i)) ));
                  c(for( i,minus(n,1),1,
                       [readfile_struct(fp,NAME),increase(NAME)] ) ),
   subt_sp,write_sp,write('}'),nl.
/* funct_def([[readfile_struct_array,void],
                [[FN,'char*'],[STRUCT,struct],[[NAME,[]],STRUCT],[SIZE,int]] ]),
    begin_body, var_def([[*fp',FILE],[i,int]]),
      [fp,により,FN,を,read,用にオープンする],
      [for,[i,を,0,から,SIZE-1,まで,1,づつ変え繰り返す]],
        [fp,を用いてﾌｧｲﾙから構造体の配列要素,[NAME,[i]],に読む込む],
      end_for,end_body. */


qpr_str:-obj([name(tmp),type(struct(item))]),c(print_struct(tmp)),nl.


c(add(X,N)):-write_sp,writelist([X,'+=',N,';']),nl.
c(increase(C)):-write_sp,writelist([C,'++',';']),nl.
type_decl(TYPE,OBJLIST):-write_sp,writelist(['TYPE,'          ']),
                        writelist_ic(OBJLIST),write(';'),nl.
struct_type_decl(TYPE,OBJLIST):-write_sp,writelist(['struct   ',TYPE,'    ']),
                        writelist_ic(OBJLIST),write(';'),nl.
struct_type_def(STRUCT):-obj([name(X),type(struct(STRUCT))]),
                        struct_member(STRUCT,L),write_sp,
                        writelist(['struct    ',STRUCT,'{']),nl,add_sp,
                        write_struct_def_list(L),subt_sp,write_sp,write('};'),nl.
write_struct_def_list([HN,HT]|T]):-
                        write_struct(HN,HT),write_struct_def_list(T).
write_struct_def_list([]).
write_struct([HN,[N]],string):-write_sp,writelist(['char     ',HN,[N],';']),nl.
write_struct(HN,HT):-write_sp,writelist([HT,'     ',HN,';']),nl.
c(read_struct(X,I)):-obj([name(X),type(struct(STRUCT))]),
                        struct_member(STRUCT,L),
                        c(read_memberlist(X,I,L)).
c(readfile_struct(FILE_PNTR,X,I)):-obj([name(X),type(struct(STRUCT))]),
                        struct_member(STRUCT,L),
                        c(readfile_memberlist(FILE_PNTR,X,I,L)).
c(read_struct(X)):-obj([name(X),type(struct(STRUCT))]),struct_member(STRUCT,L),
                    c(read_memberlist(X,L)).
c(read_struct(STRUCT,X)):-obj([name(X),type(struct(STRUCT))]),
                    struct_member(STRUCT,L),c(read_memberlist(X,L)).

c(readfile_struct(FILE_PNTR,X)):-obj([name(X),type(struct(STRUCT))]),
                        struct_member(STRUCT,L),
                        c(readfile_memberlist(FILE_PNTR,X,L)).
c(readfile_struct(FILE_PNTR,STRUCT,X)):-obj([name(X),type(struct(STRUCT))]),
                        struct_member(STRUCT,L),
                        c(readfile_memberlist(FILE_PNTR,X,L)).
c(read_memberlist(X,I,[[HN,HT]|T])):-c(prompt_read_member([X,I,HN],HT)),
                                        c(read_memberlist(X,I,T)).
c(read_memberlist(X,I,[])).
c(readfile_memberlist(FILE_PNTR,X,I,[[HN,HT]|T])):-
                                    c(readfile_member(FILE_PNTR,[X,I,HN],HT)),
                                        c(readfile_memberlist(FILE_PNTR,X,I,T)).
c(readfile_memberlist(FILE_PNTR,X,I,[])).
c(prompt_read_member([X,I,MEM],TYPE)):-lang(c),
    /*   included('stdio.h'),   */
    write_sp,writelist(['printf("',X,'[',I,']',',MEM,'= in ' ,TYPE,' ");']),nl,
            c(read_member([X,I,MEM],TYPE)).
c(read_member([X,I,MEM],TYPE)):-lang(c),
            /* included('stdio.h'),*/
            ptype(TYPE,PTYPE),
            write_sp,writelist(['scanf("',PTYPE,"',']),
            write_type_struct_obj(TYPE,[X,I,MEM]),write(');'),nl.
c(readfile_member(FILE_PNTR,[X,I,MEM],TYPE)):-lang(c),
             /* included('stdio.h'),*/
            ptype(TYPE,PTYPE),
            write_sp,writelist(['fscanf(',FILE_PNTR,',"',PTYPE,"',']),
            write_type_struct_obj(TYPE,[X,I,MEM]),write(');'),nl.
c(read_memberlist(X,[[HN,HT]|T])):-c(prompt_read_member([X,HN],HT)),
                                        c(read_memberlist(X,T)).
c(read_memberlist(X,[])).

c(readfile_memberlist(FILE_PNTR,X,[[HN,HT]|T])):-
                                    c(readfile_member(FILE_PNTR,[X,HN],HT)),
                                      c(readfile_memberlist(FILE_PNTR,X,T)).
c(readfile_memberlist(FILE_PNTR,X,[])).

c(prompt_read_member([X,MEM],TYPE)):-lang(c),
             /* included('stdio.h'),*/
          write_sp,writelist(['printf("',X,'->',MEM,'= in ' ,TYPE,' ");']),nl,
              c(read_member([X,MEM],TYPE)).
c(read_member([X,MEM],TYPE)):-lang(c),
            /* included('stdio.h'), */
            ptype(TYPE,PTYPE),
            write_sp,writelist(['scanf("',PTYPE,"',']),
            write_type_struct_obj(TYPE,[X,MEM]),write(');'),nl.
c(readfile_member(FILE_PNTR,[X,MEM],TYPE)):-lang(c),
            /* included('stdio.h'), */
            ptype(TYPE,PTYPE),
            write_sp,writelist(['fscanf(',FILE_PNTR,',"',PTYPE,"',']),
            write_type_struct_obj(TYPE,[X,MEM]),write(');'),nl.
write_type_struct_obj(int,[X,MEM]):-(point_var(0),
            writelist(['&(',X,'.',MEM,')']); writelist(['&(',X,'->',MEM,')']) ).
write_type_struct_obj(float,[X,MEM]):-(point_var(0),
```

```
                writelist(['&(',X,',',MEM,')']); writelist(['&(',X,'->',MEM,')']) ).
write_type_struct_obj(char,[X,MEM]):-(point_var(0),
                writelist(['&(',X,',',MEM,')']); writelist(['&(',X,'->',MEM,')']) ).
write_type_struct_obj(string,[X,[MEM,[N]]]):-(point_var(0),
                writelist([X,',',MEM]); writelist([X,'->',MEM]) ).
write_type_struct_obj(string,[X,MEM]):-(point_var(0),
                writelist([X,',',MEM]); writelist([X,'->',MEM]) ).
write_type_struct_obj(int,[X,I,MEM]):-writelist(['&(',X,'[',I,'].',MEM,')']).
write_type_struct_obj(float,[X,I,MEM]):-
                                writelist(['&(',X,'[',I,'].',MEM,')']).
write_type_struct_obj(char,[X,I,MEM]):-writelist(['&(',X,'[',I,'].',MEM,')']).
write_type_struct_obj(string,[X,I,[MEM,[N]]]):-writelist([X,'[',I,']',MEM]).




raip:-retract(point_var(0)),assert(point_var(1)).
rapi:-retract(point_var(1)),assert(point_var(0)).
qrd_i_k:-gen_funct(read_struct_array(item1,n)).
qrd_i_f:-gen_funct(readfile_struct_array(fn,item1,n)).
qrd_m_k:-gen_funct(read_struct_array(meibo1,n)).
qrd_m_f:-gen_funct(readfile_struct_array(fn,meibo1,n)).

qrdst2:-gen_funct(read_struct_array(meibo1,n)).
qstruct1_1:-c(read_struct(item1)).
qstruct1_2:-gen_funct(read_struct(item1)).
qstruct1_3:-c(prompt_read_member([item1,'name[']],char)).
qstruct1_4:-c(read_struct(item1,i)).

make_spec:-writelist(['次の spec を入れてください。']),read(NEXT_SP),
                spec(SP),add_spec(NEXT_SP,SP,NEW_SP),prog(NEW_SP).



struct_member(meibo,[[[name,[20]],string],[code,int],[age,int],
    [[tel,[20]],string],[[address,[30]],string] ]).
table_head_format(item1,['%10s','%8s','%10s']).
table_head_format(meibo1,['%10s','%5s','%5s','%15s','%24s']).
table_data_format(item1,['%10s','%8d','%10d']).
table_data_format(tmp,['%10s','%8d','%10d']).
table_data_format(meibo1,['%10s','%5d','%5d','%15s','%24s']).



gen_funct(print_struct_array(NAME,SIZE)):-
   obj([name(NAME),type(struct(STRUCT))]),
   struct_member(STRUCT,L),write_sp,(point_var(0),
   writelist(['void print_struct_array(struct ',STRUCT,' ',NAME,'[],int ',SIZE,
   ')']);
   writelist(['void print_struct_array(struct ',STRUCT,' *',NAME,',int ',SIZE,
   ')']),
   write('{'),nl,add_sp,
   type_decl(int,[i]),
   c(print_table_headline(NAME)),
   (point_var(0),c(for(i,minus(n,1),1,c(print_struct(NAME,i)) ));
                c(for(i,minus(n,1),1,
                  [print_struct(NAME),increase(NAME)] ) ),
                subt_sp,write_sp,write('}'),nl.
/* funct_def([[print_struct_array,void],
                [[STRUCT,struct],[[NAME,[]],STRUCT],[SIZE,int]] ]),
    begin_body, var_def([[i,int]]),
      [NAME,の表の見出しをプリントする],
      [for,[i,を,0,から,SIZE-1,まで,1,づつ変え繰り返す]],
        [構造体の配列要素,[NAME,[i]],をプリントする],
      end_for,end_body. */
gen_funct(printfile_struct_array(NAME,SIZE,FN)):-
   obj([name(NAME),type(struct(STRUCT)),point_var(PV)]),
   struct_member(STRUCT,L),write_sp,(point_var(0),
   writelist(['void printfile_struct_array(struct ',STRUCT,' ',NAME,
                '[],int ',SIZE,',char *',FN,')']);
   writelist(['void printfile_struct_array(struct ',STRUCT,' *',NAME,
                ',int ',SIZE,',char *',FN,')']) ),
   write('{'),nl,add_sp,type_decl('FILE',[*fp']),nl,type_decl(int,[i]),
   file_open(FN,fp,"'w'"),
   (point_var(0),c(for(i,minus(n,1),1,
                        c(writefile_struct(fp,NAME,i)) )) ;
                c(for(i,minus(n,1),1,
                        [writefile_struct(fp,NAME),increase(NAME)] ))   ),
   subt_sp,write_sp,write('}'),nl.
   file_open(FN,FL_PNTR,RW):-write_sp,
                writelist([FL_PNTR,'=fopen(',FN,',',RW,');']),nl.
/* funct_def([[printfile_struct_array,void],
                [[STRUCT,struct],[[NAME,[]],STRUCT],[SIZE,int],[FN,'char*']] ]),
    begin_body, var_def([[*fp',FILE],[i,int]]),
      [fp,により,FN,を,write,用にオープンする],
      [for,[i,を,0,から,SIZE-1,まで,1,づつ変え繰り返す]],
        [fp,を用いてﾌｧｲﾙに構造体の配列要素,[NAME,[i]],を書き出す],
      end_for,end_body. */



c(print_struct(X,I)):-obj([name(X),type(struct(STRUCT))]),
                struct_member(STRUCT,L),c(print_memberlist(X,I,L,NAME,TYPE)),
                c(print_struct_list(X,TYPE,NAME)).

c(writefile_struct(FP,X,I)):-obj([name(X),type(struct(STRUCT))]),
                struct_member(STRUCT,L),c(print_memberlist(X,I,L,NAME,TYPE)),
                c(writefile_struct_list(FP,X,TYPE,NAME)).

c(print_memberlist(X,I,[[HN,HT]|T],[[[X,[I]],',',HN]|TN],[HT|TT])):-
                c(print_member(X,I,[HN,HT],[[X,[I]],',',HN],HT)),
                c(print_memberlist(X,I,T,TN,TT)).
c(print_member(X,I,Y,Z,U)).
c(print_memberlist(X,I,[],[],[])).

c(print_struct(STRUCT,X)):-obj([name(X),type(struct(STRUCT))]),
      struct_member(STRUCT,L),c(print_memberlist(X,L,NAME,TYPE)),
      c(print_struct_list(X,TYPE,NAME)).
c(print_struct(X)):-obj([name(X),type(struct(STRUCT))]),
      struct_member(STRUCT,L),c(print_memberlist(X,L,NAME,TYPE)),
      c(print_struct_list(X,TYPE,NAME)).

c(writefile_struct(FP,X)):-obj([name(X),type(struct(STRUCT))]),
                struct_member(STRUCT,L),c(print_memberlist(X,L,NAME,TYPE)),
                c(writefile_struct_list(FP,X,TYPE,NAME)).
c(writefile_struct(FP,STRUCT,X)):-obj([name(X),type(struct(STRUCT))]),
                struct_member(STRUCT,L),c(print_memberlist(X,L,NAME,TYPE)),
                c(writefile_struct_list(FP,X,TYPE,NAME)).

c(print_memberlist(X,[[HN,HT]|T],[[X,HN]|TN],[HT|TT])):-
                c(print_member(X,[HN,HT],[X,',',HN],HT)),
                c(print_memberlist(X,T,TN,TT)).
c(print_member(X,Y,Z,U)).
c(print_memberlist(X,[],[],[])).
c(print_struct_list(X,TYPE_LIST,OBJ_LIST)):-
      lang(c),
       /*   included('stdio.h'), */
```

```
    write_sp,write('printf("'),
    (table_data_format(X,P_DATAFORM),writelist(P_DATAFORM);
      std_print_ptype_list(TYPE_LIST,PTYPE_LIST),writelist(PTYPE_LIST)),
    write('¥n"'),writelist_list_hc_nl_a(OBJ_LIST),write(');'),nl.
c(writefile_struct_list(FP,X,TYPE_LIST,OBJ_LIST)):-
    lang(c),
    /* included('stdio.h'), */
     write_sp,writelist(['fprintf(',FP,',"']),
    (table_data_format(X,P_DATAFORM),writelist(P_DATAFORM);
     ptype_list(TYPE_LIST,PTYPE_LIST),writelist(PTYPE_LIST)),
     write('¥n"'),writelist_list_hc_nl_a(OBJ_LIST),write(');'),nl.

writelist_list_hc_nl_a([H|T]):-
    write(','),(write_struct_obj(H);writelist(H);write(H)),
                                writelist_list_hc_nl_a(T).
writelist_list_hc_nl_a([]).

write_struct_obj([X,[Y,[N]]]):-(write(X);writelist(X)),
                    (point_var(0),writelist(['.',Y]);writelist(['->',Y])).
write_struct_obj([X,Y]):-(write(X);writelist(X)),
                    (point_var(0),writelist(['.',Y]);writelist(['->',Y])).

write_struct_obj([X,M_SYMB,[Y,[N]]]):-writelist(X),writelist([M_SYMB,Y]).
write_struct_obj([X,M_SYMB,Y]):-writelist(X),writelist([M_SYMB,Y]).
write_struct_obj([X,M_SYMB,[Y,[N]]]):-writelist([X,M_SYMB,Y]).
write_struct_obj([X,M_SYMB,Y]):-writelist([X,M_SYMB,Y]).

c(print_table_headline(X)):-table_head_format(X,L),
                    obj([name(X),type(struct(STRUCT))]),struct_member(STRUCT,NT),
                    write_sp,write('printf("'),writelist(L),write('¥n"'),
                    member_name_list(NT,NAMELIST),
                    writelist_hc_dquote(NAMELIST),write(');'),nl.
writelist_hc_dquote([[H,[N]]|T]):-writelist(['","',H,'"']),
                                writelist_hc_dquote(T).
writelist_hc_dquote([H|T]):-writelist(['","',H,'"']),writelist_hc_dquote(T).
writelist_hc_dquote([]).

member_name_list([[HN,HT]|T],[HN|TN]):-
                    member_name([HN,HT],HN),member_name_list(T,TN).
member_name_list([],[]).
member_name(X,Y).
qnm1(X):-member_name_list([[a1,t1],[a2,at2],[a3,t3]],X).
qnm2(L,X):-struct_member(item,L),member_name_list(L,X).

qpr_i_k:-gen_funct(print_struct_array(item1,n)).
qpr_i_f:-gen_funct(printfile_struct_array(item1,n,fn)).
qpr_m_k:-gen_funct(print_struct_array(meibo1,n)).
qpr_m_f:-gen_funct(printfile_struct_array(meibo1,n,fn)).
qstruct2_1:-c(print_table_headline(item1)).
qstruct2_2:-c(print_struct(item1)).


gen_funct(sort_struct_array(NAME,SIZE,KEY_N)):-
  obj([name(NAME),type(struct(STRUCT)),point_var(PV)]),
  struct_member(STRUCT,L),(point_var(0),
  (member([[KEY_N,[S]],KEY_T],L);member([KEY_N,KEY_T],L)),
  write_sp,writelist(['void     srt_struct_array(struct ',STRUCT,' ',NAME,'[]',
                    ',int ',SIZE,')']);
  write_sp, writelist(['void     srt_struct_array(struct ',STRUCT,' *',NAME,
                    ',int ',SIZE,')']),
  write('{'),nl,add_sp,
    type_decl(char,['tmp[20]']),
    type_decl(int,[i,m,tmp_num]),
    (point_var(0),nl,sort_struct_array(NAME,i,L,SIZE,KEY_N);
     sort_struct_array(NAME,i,L,SIZE,KEY_N)),
    subt_sp,write_sp,write('}'),nl.
/* funct_def([[srt_struct_array,void],
                [[STRUCT,struct],[[NAME,[]],STRUCT],[SIZE,int] ] ]),
    begin_body, var_def([['tmp[20]',char],[[i,m,tmp_num],int]]),
      [添字,i,ﾊﾝﾊﾞｰﾘｽﾄ,L,をもち大きさ,SIZE,の構造体配列,NAME,を,KEY_N,
       をキーとして,ソートする],
      end_for,end_body. */

gen_funct(sort_struct_array_a(NAME,SIZE,KEY_N,ORDER)):-
  obj([name(NAME),type(struct(STRUCT)),point_var(PV)]),
  struct_member(STRUCT,L),
  write_sp,writelist(['void     srt_struct_array(struct ',STRUCT,' ',NAME,'[]',
                    ',int ',SIZE,',int ',ORDER,')']),
  write('{'),nl,add_sp,
    type_decl(char,['tmp[20]']),
    type_decl(int,[i,m,tmp_num]),
    sort_struct_array_a(NAME,i,L,SIZE,KEY_N,ORDER),
    subt_sp,
    write_sp,write('}'),nl.
sort_struct_array(X,I,L,SIZE,KEY_N):-
                c(for(conv([m,'=',minus(SIZE,1)]),conv([m,'>=',1]),step([m,-1]),
                    for(conv([I,'=',0]),conv([I,'<',m]),step([I,1]),
                    sort_struct(X,I,L,KEY_N) ) ) ).
sort_struct_array_a(X,I,L,SIZE,KEY_N,ORDER):-
                c(for(conv([m,'=',minus(SIZE,1)]),conv([m,'>=',1]),step([m,-1]),
                    for(conv([I,'=',0]),conv([I,'<',m]),step([I,1]),
                    sort_struct_a(X,I,L,KEY_N,ORDER) ) ) ).
c(for(conv(INIT),conv(END),step(S),P)):-write_sp,
  write('for('),conv(INIT),write(';'),conv(END),write(';'),step(S),write('){'),
  nl,add_sp,(proc_list(P);P),subt_sp,write_sp,write('}'),nl.
c(if(conv(T),X)):-write_sp,write('if('),conv(T),write('){'),nl,add_sp,
                    proc_list(X),nl,
                    subt_sp,write_sp,write('}'),nl.
c(if(conv(T),X1,X2)):-write_sp,write('if('),conv(T),write('){'),nl,add_sp,
                    proc_list(X1),
                    subt_sp,write_sp,write('} else {'),nl,add_sp,
                    proc_list(X2),
                    subt_sp,write_sp,write('}'),nl.
conv([I,REL,minus(V,N)]):-writelist([I,REL,V,'-',N]).
conv([I,REL,V]):-writelist([I,REL,V]).
step([I,1]):-writelist(['++',I]).
step([I,-1]):-writelist(['--',I]).
c(sort_struct(X,I,L,KEY_N)):-
              ( member([[KEY_N,[S]],string],L),
                c(if(conv(greater(t(strcmp([X,I,KEY_N],[X,nxt(I),KEY_N]),t(0))),
                swaplist(X,I,nxt(I),L)));
                c(if(conv(greater(t([X,I,KEY_N]),t([X,nxt(I),KEY_N]))),
                swaplist(X,I,nxt(I),L))) ).

c(sort_struct_a(X,I,L,KEY_N,ORDER)):-
                c(if(conv(equal(ORDER,1)),
                if(conv(greater(t(strcmp([X,I,KEY_N],[X,nxt(I),KEY_N]),t(0))),
                swaplist(X,I,nxt(I),L) ),
                if(conv(smaller(t(strcmp([X,I,KEY_N],[X,nxt(I),KEY_N]),t(0))),
                swaplist(X,I,nxt(I),L) ) )).
conv(greater(t(X),t(Y))):-t(X),write('>'),t(Y).
conv(smaller(t(X),t(Y))):-t(X),write('<'),t(Y).
conv(equal(X,V)):-writelist([X,'==',V]).
t(strcmp([X,I,KEY],[X,nxt(I),KEY])):-write('strcmp('),
                t([X,I,KEY]),write(','),t([X,nxt(I),KEY]),write(')').
t([X,nxt(I),KEY]):-writelist([X,'[',I,'+1].',KEY]).
```

```
t([X,I,KEY]):-writelist([X,'[',I,']',KEY]).
t(0):-write(0).
c(swaplist(X,I,J,[[LNH,LTH]|LT])):-
                swap(X,I,J,LNH,LTH),
                c(swaplist(X,I,J,LT)).
c(swaplist(X,I,J,[])).
swap(X,I,nxt(I),[LNH,[LSH]],string):-
   write_sp,writelist(['strcpy(tmp,',X,'[',I,'],',LNH,');']),nl,
   write_sp,writelist(['strcpy(',X,'[',I,'],',LNH,',',X,'[',I,'+1].',LNH,');']),
   nl,write_sp,writelist(['strcpy(',X,'[',I,'+1].',LNH,',tmp,');']),nl.

swap(X,I,nxt(I),LNH,int):-
write_sp,writelist(['tmp_num = ',X,'[',I,'].',LNH,';']),nl,
write_sp,writelist([X,'[',I,'].',LNH,' = ',X,'[',I,'+1].',LNH,';']),nl,
write_sp,writelist([X,'[',I,'+1].',LNH,' = ',tmp_num,';']),nl.


qsrt_i_a:-gen_funct(sort_struct_array_a(item1,n,'name[]',order)).
qsrt_m_a:-gen_funct(sort_struct_array_a(meibo1,n,'name[]',order)).
qsrt_i_n:-gen_funct(sort_struct_array(item1,n,name)).
qsrt_m_n:-gen_funct(sort_struct_array(meibo1,n,name)).
qsrt_i_num:-gen_funct(sort_struct_array(item1,n,number)).
qsort2:-gen_funct(sort_struct_array(meibo1,n,name)).
qsorta:- struct_member(item,L),
   c(sort_struct_a(item1,i,L,name,order)).
qswapa:- struct_member(item,L),
   c(swaplist(item1,i,nxt(i),L)).
qifa:-struct_member(item,L),
        c(if(conv(greater(t(strcmp([item1,i,name],[item1,nxt(i),name])),t(0))),
        swaplist(item1,i,nxt(i),L) )).
qsort:-
  c(sort_struct(item1,i,[['name[]',char],[price,int],[number,int]],'name[]')).
qswap:-c(swaplist(item,i,nxt(i),[['name[]',char],[price,int],[number,int]])).
qif:-c(if(t,swaplist(item,i,nxt(i),[['name[]',char],[price,int],
                                        [number,int]]))).
qgrtrn:-conv(greater( t(strcmp([item,i,'name[]'],[item,nxt(i),'name[]'])),
                        t(0) )).
qgrtri:-conv(greater( t([item,i,number]),
                        t([item,nxt(i),number]) )).

qrdfl_str:-obj([name(tmp),type(struct(item))]),c(readfile_struct(fp,tmp)),nl.

gen_funct(retrfile_to_str_parr(FN,SIZE,RETR_COND,RETR_AR)):-
   include_list(['stdio.h','string.h']),
   obj([name(RETR_AR),type(struct(STRUCT))]),

   struct_member(STRUCT,L),write_sp,
   (point_var(0),
   writelist(['int     retrfile_to_str_parr(char   *',FN,',int ',SIZE,
                ',struct ',STRUCT,' ',RETR_AR,'[])']);
   writelist(['int     retrfile_to_str_parr(char   *',FN,',int ',SIZE,
                ',struct ',STRUCT,' *',RETR_AR,')']),
   write('{'),nl,add_sp,type_decl('FILE',[' *fp']),
   (point_var(0),struct_type_decl(STRUCT,['tmp']);
   struct_type_decl(STRUCT,['*tmp'])),
   type_decl(int,[i,cnt,j]),
   file_open(FN,fp,"r"),
   set(cnt,0),
   obj([name(tmp),type(struct(STRUCT)) ]),
   (point_var(0),set(j,0),
   c(for(i,minus(SIZE,1),1,
      [readfile_struct(fp,STRUCT,tmp),
        if(conv(RETR_COND), [print_struct(STRUCT,tmp),
          copy_struct_array(tmp,RETR_AR),
            add(j,1),add(cnt,1)]) ]));
   c(for(i,minus(SIZE,1),1,
      [readfile_struct(fp,STRUCT,tmp),
        if(conv(RETR_COND), [print_struct(STRUCT,tmp),
        copy_struct_array(tmp,RETR_AR),
            add(RETR_AR,1),add(cnt,1)]) ]) ),
   c(return(cnt)),
   subt_sp,write('}').


include_list([H|T]):-include(H),include_list(T).
include_list([]).
include(H):-write_sp,writelist(['#include <',H,'>']),nl.

c(copy_struct_array(STR_NAME,ARRAY_NAME)):-
                obj([name(STR_NAME),type(struct(STRUCT))]),
                obj([name(ARRAY_NAME),type(struct(STRUCT))]),
                struct_member(STRUCT,L),
                copy_list(STR_NAME,ARRAY_NAME,L),pointer_add(ARRAY_NAME).
copy_list(STR,ARRAY,[[HN,HT]|T]):-
              copy(STR,ARRAY,[HN,HT]),copy_list(STR,ARRAY,T).
copy_list(STR,ARRAY,[]).
copy(STR,ARRAY,[[HN,[S]],string]):-(point_var(0),
        write_sp,writelist(['strcpy(',ARRAY,'[j].',HN,',',STR,'.',HN,');']);
        write_sp,writelist(['strcpy(',ARRAY,'->',HN,',',STR,'->',HN,');'])),
nl.
copy(STR,ARRAY,[HN,TX]):-(point_var(0),
        write_sp,writelist([ARRAY,'[j].',HN,'=',STR,'.',HN,';']);
        write_sp,writelist([ARRAY,'->',HN,'=',STR,'->',HN,';'])),
nl.
pointer_add(P):-write_sp,(point_var(0);writelist(['++',P,';'])),nl.
qret_i:-
   gen_funct(retrfile_to_str_parr(fn,n,['tmp.number','>',750],retr_ar)).
qret_m:-
   gen_funct(retrfile_to_str_parr(fn,n,['tmp.age','>',50],retr_ar_m)).

qupdt_i:-gen_funct(updatefile_from_trans(ret1,n,item,
                        ['strcmp(tmp.name,cn)','==',0],cn)).
qupdt_m:-gen_funct(updatefile_from_trans(ret1,n,meibo,
                        ['strcmp(tmp.name,cn)','==',0],cn)).

gen_funct(updatefile_from_trans(FN,SIZE,STRUCT,CODE_LIST,cn)):-
   include_list(['stdio.h','string.h']),
   obj([name(tmp),type(struct(STRUCT))]),
   struct_member(STRUCT,L),write_sp,
   (point_var(0),
   writelist(['int updatefile_from_trans(char *',FN,',int ',SIZE,',char *cn)']);
  writelist(['int updatefile_from_trans(char *',FN,',int ',SIZE,',char *cn)'])),
   write('{'),nl,add_sp,type_decl('FILE',[' *fp']),
   (point_var(0),struct_type_decl(STRUCT,['tmp']);
   struct_type_decl(STRUCT,['*tmp'])),
   type_decl(int,[i]),
   type_decl(long,[fp_val]),
   file_open(FN,fp,"r+"),
   obj([name(tmp),type(struct(STRUCT)) ]),
   (point_var(0),
   c(for(i,minus(SIZE,1),1,
        [tell_flpt_val(fp,fp_val),readfile_struct(fp,STRUCT,tmp),
        if(conv(CODE_LIST),
          [print_struct(STRUCT,tmp),read_struct(STRUCT,tmp),
           seek_flpt(fp,fp_val,0),
           writefile_struct(fp,STRUCT,tmp)]) ]));
```

```
  c(for(i,minus(SIZE,1),1,
      [tell_flpt_val(fp,fp_val),readfile_struct(fp,STRUCT,tmp),
      if(conv(CODE_LIST),
        [print_struct(STRUCT,tmp),read_struct(STRUCT,tmp),
          seek_flpt(fp,fp_val,0),
          writefile_struct(fp,STRUCT,tmp)])
    ])),
  subt_sp,write_sp,
  write('}').

c(tell_flpt_val(FP,VAL)):-write_sp,writelist([VAL,'=ftell(',FP,');']),nl.
c(seek_flpt(FP,VAL,0)):-write_sp,write(fseek(FP,VAL,0)),write(';'),nl.


spec([]).

 dict3([
 [head(繰り返し,repeat),
     body([
          p(0,[[FUNCT_NAME,FUNCT_TYPE],PNAME_PTYPE_PAIRLIST],
               funct_head([[FUNCT_NAME,FUNCT_TYPE],PNAME_PTYPE_PAIRLIST])),
          p(1,OBJ_TYPE_PAIRLIST,obj_type_pairlist(OBJ_TYPE_PAIRLIST)),
          p(2,REP,rep(REP)),
          p(3,INDEX,index(INDEX)),
          p(31,VAR_INITVAL_PAIRINITLIST,
                var_initval_pairlist(VAR_INITVAL_PAIRLIST)),
          p(32,PRE_PROC,pre_proc(PRE_PROC)),
          p(331,REPETITION_COND,repetition_cond(REPETITION_COND)),
          p(332,TERMINATION,termination(TERMINATION)),
          p(34,REP_UPDATE,rep_update(REP_UPDATE)),
          p(4,PROC_LIST,rep_proc_list(PROC_LIST)),
          p(5,POST_PROC,post_proc_list(POST_PROC)),
          p(6,VAR_VAL,return(VAR_VAL))
        ])] ]).

spec_list([
          funct_head([sum_a,float],[['a[]',float]]),
          obj_type_pairlist([[[sum,'a[]'],float] ]),
          rep(while),index(i),
          var_initval_pairlist([[sum,0],[i,0]]),
          pre_proc(PRE_PROC),
          repetition_cond('a[i]>=0'),
          termination(TERMINATION),
          rep_update(compute('i+=1')),
          rep_proc_list( compute(sum=sum+'a[i]') ),
          post_proc_list([ print(sum) ]),
          return(sum) ]).


funct_head([FUNCT_NAME,FUNCT_TYPE],PNAME_PTYPE_PAIRLIST):-
    write_sp,writelist([FUNCT_TYPE,'    ',FUNCT_NAME,'(']),
    paralist(PNAME_PTYPE_PAIRLIST),write(')').
h_paralist([[HN,HT]|TNT]):-write(','),writelist([HT,' ',HN]),h_paralist(TNT).
h_paralist([]).
paralist([[HN,HT]|TNT]):-writelist([HT,' ',HN]),h_paralist(TNT).
begin_def:-write('{'),nl,add_sp.
end_def:-subt_sp,nl,write_sp,write('}').
obj_type(sum,float).


ex1:-
funct_head([sum_a,float],[['a[]',float]]),begin_def,
type_decl(float,[sum,'a[]']),
set(sum,0),
set(i,0),
c(while('a[i]>=0',
  [compute(sum=sum+'a[i]'),
   compute('i+=1') ])),
print(sum),
c(return(sum)),
end_def.

ex2:-
  spec_list([
  funct_head([FUNCT_NAME,FUNCT_TYPE],PNAME_PTYPE_PAIRLIST),
  obj_type_pairlist(OBJ_TYPE_PAIRLIST),
  rep(while),index(i),
  var_initval_pairlist(VAR_VAL_PAIRLIST),
  pre_proc(PRE_PROC),
  repetition_cond(REPETITION_COND),
  termination(TERMINATION),
  rep_update(REP_UPDATE),
  rep_proc_list(REP_PROC_LIST),
  post_proc_list(POST_PROC_LIST),
  return(VAR_VAL) ]),
  funct_head([FUNCT_NAME,FUNCT_TYPE],PNAME_PTYPE_PAIRLIST),begin_def,
  type_decl_list(OBJ_TYPE_PAIRLIST),
  initial_set_list(VAR_VAL_PAIRLIST),
  c(while(REPETITION_COND,[REP_PROC_LIST,REP_UPDATE])),
  post_proc_list(POST_PROC_LIST),
  c(return(VAR_VAL)),
  end_def.
type_decl_list([H|T]):-obj_type_pair(H),type_decl_list(T).
type_decl_list([]).
obj_type_pair([O,T]):-type_decl(T,O).
initial_set_list([[HL,HR]|T]):-set(HL,HR),initial_set_list(T).
initial_set_list([]).
post_proc_list(X):-proc_list(X).
rep_proc_list(X):-proc_list(X).
obj_type_list([a,b,c],[int,float,char]).

statement([OBJLIST,を,TYPELIST,に宣言],type_decl(TYPELIST,OBJLIST)).

add_spec(X,Y,Z):-spec(Y),reverse([X|Y],Z),retract(spec(Y)),assert(spec(Z)),
                  writelist(Z).
writelist([]).
writelist([H|T]):-write(H),writelist(T).
writeqlist([]).
writeqlist([H|T]):-writeq(H),writeqlist(T).
member(X,[X|L]).
member(X,[Y|L]):-member(X,L).
ret_sp([]).
ret_pic_sp([]).
rasp:-retract(ret_sp(X)),assert(ret_sp([])).
rapsp:-retract(ret_pic_sp(X)),assert(ret_pic_sp([])).
buff([]).
proc([]).
reverse([],[]).
reverse([A|X],Z):-reverse(X,XR),append(XR,[A],Z).
append([],X,X).
append([A|X],Y,[A|Z]):-append(X,Y,Z).
writelist_list([H|T]):-writelist(H),writelist_list(T).
writelist_list([]).
writelist_list_nl([H|T]):-(writelist_nl(H);write(H)),nl,writelist_list_nl(T).
writelist_list_nl([]).
writelist_nl([H|T]):-write(H),nl,writelist_nl(T).
```

```
writelist_nl([]).
writelist_hc_nl([H|T]):-write(','),nl,write(H),writelist_hc_nl(T).
writelist_hc_nl([]).
writelist_ic_nl([H|T]):-write(H),writelist_hc_nl(T).
writelist_list_hc_nl([H|T]):-write(','),nl,(writelist_ic_nl(H);write(H)),
                                writelist_list_hc_nl(T).
writelist_list_hc_nl([]).
writelist_list_hc_nl_a([H|T]):-write(','),nl,(writelist(H);write(H)),
                                writelist_list_hc_nl_a(T).
writelist_list_hc_nl_a([]).
writelist_list_ic_nl([H|T]):-(writelist_ic_nl(H);write(H)),
                                writelist_list_hc_nl(T).

writelist_com([H|T]):-write(H),write(','),writelist_com(T).
writelist_com([]).
write_spec1:-ret_sp(SPEC),reverse(SPEC,R_SPEC),
        writelist_ic_nl(R_SPEC).
write_pic_spec1:-ret_pic_sp(SPEC),reverse(SPEC,R_SPEC),
        writelist_ic_nl(R_SPEC).
write_spec2:-ret_sp(SPEC),reverse(SPEC,R_SPEC),
        writelist_list_ic_nl(R_REC).


/*    3．1　仕様の概要　*/

fd_gen(N):-spec([N,fd],[[FN,FT],ARGLIST]),writelist([FT,'    ',FN,'(']),
    gen_arg(ARGLIST),write(')').
begin_body:-write('{'),nl.
end_body:-write('}'),nl.
gen_arg([[HN,HT]|T]):-writelist([HT,' ',HN]),gen_arg_hc(T).
gen_arg_hc([[HN,HT]|T]):-writelist([',',HT,' ',HN]),gen_arg_hc(T).
gen_arg_hc([]).
vd_gen(N):-spec([N,vd],VARLIST),var_decl(VARLIST),nl.
var_decl([[HN,HT]|T]):-
    write_sp,writelist([HT,' ']),(writelist_ic(HN);write(HN)),write(';'),nl,
    var_decl(T).
var_decl([]).


pre_genc(FN,N,FNC):-[FN],spec([N,all],[FD,VD,PROC]),tell(FNC),
        write(cg(N)),write(':-proc_list(['),preconv_list(PROC),
        write(']).'),told,[FNC],cg(N).

spec([11,all],[[[read_struct_array,void],[[STRUCT,struct],[[NAME,[]],STRUCT],[SIZE,int]]],[[[i,in
t]]],[[for,[i,を,0,から,'SIZE'-1,まで,1,づつ変え繰り返す]],[構造体の配列要素,[NAME,[i]],に読み込
む],end_for]]).


c_gen(FN,N,NAME,STRUCT):-
  obj([name(NAME),type(struct(STRUCT)),point_var(PV)]),
  struct_member(STRUCT,L),write_sp,point_var(0),
  read_spec(FN,N),fd_gen(N),begin_body,vd_gen(N),
        spec([N,proc],PROC),tell(tmp),write(cg(N)),write(':-proc_list(['),
        preconv_list(PROC),write(']).'),told,[tmp],cg(N),end_body.


read_spec(FN,N):-[FN],spec([N,all],[FD,VD,PROC]),
    ra_a(spec([N,fd],X),spec([N,fd],FD)),
    ra_a(spec([N,vd],Y),spec([N,vd],VD)),
    ra_a(spec([N,proc],Z),spec([N,proc],PROC)).
ra_a(X,Y):-(retract(X),assert(Y);assert(Y)).
proc_gen(N):-spec([N,proc],PROC),proc_list(PROC).

pre_genc(FN,N,FNC):-[FN],spec([N,all],[FD,VD,PROC]),tell(FNC),
        write(cg(N)),write(':-proc_list(['),preconv_list(PROC),
        write(']).'),told,[FNC],cg(N).

preconv_list([H|T]):-preconv(H),preconv_hc_list(T).
preconv_hc_list([H|T]):-write(','),preconv(H),preconv_hc_list(T).
preconv_hc_list([]).
preconv([for,COND]):-write('['),writeq([for,COND]).
preconv([if,COND]):-write('['),writeq([if,COND]).
preconv([while,COND]):-write('['),writeq([while,COND]).
preconv(end_for):-write('end_for]').
preconv(end_if):-write('end_if]').
preconv(end_while):-write('end_whil]').
preconv(X):-writeq(X).


proc_gen(N):-spec([N,proc],PROC),proc_list(PROC).
job([]).
proc_list([H|T]):-(c(H);proc(H)),proc_list(T).
proc_list([]).
proc_list(P):-(c(P);proc(P)).
cons_main(MPR,MDT):-[MPR],[MDT].
cons_main:-[main_pr],[main_dt].
cg_main(NUM):-prog(spec(NUM)).
cons_fn:-[fn_pr],[fn_dt].
prog(spec(NUM)):-lang(c),    obj_list(spec(NUM)),
                main_head,process(spec(NUM)),main_end.
prog(spec(NUM)):-lang(cob),obj_list(spec(NUM)),
                id_env_division,data_division(spec(NUM)),process(spec(NUM)).
argl_decl([H|T]):-arg_decl(H),argl_decl(T).
argl([]).
arg_decl([NAME,TYPE]):-writelist([TYPE,'          ',NAME]),nl.

main_header:-main_arg(ARG),writelist(['main(',ARG,')']),nl,write('  {'),nl.
main_arg(' ').
fn_end:-write('}').

acob:-assert(lang(cob)).
ac:-assert(lang(c)).
rcob_ac:-retract(lang(cob)),assert(lang(c)).
rc_acob:-retract(lang(c)),assert(lang(cob)).
ra:-retract(type_member(int,INT)),assert(type_member(int,[])),
    retract(type_member(float,FLOAT)),assert(type_member(float,[])),
    retract(type_member(char,CHAR)),assert(type_member(char,[])),
    retract(include(INC)),assert(include([])).


/*  3．2　Cの型宣言作成部　*/

obj([name(NAME),type(TYPE),rem(REM)]):-lang(c),
        assert(p_obj([name(NAME),type(TYPE)])),
        assert(obj_type(NAME,TYPE)),
        type_member(TYPE,MEMBER),
        (member(NAME,MEMBER);
        add_type_entity(NAME,TYPE,MEMBER)).
obj([name_list([H_NAME|T_NAME]),type(TYPE),rem(REM)]):-lang(c),
        obj([name(H_NAME),type(TYPE),rem(REM)]),
        obj([name_list(T_NAME),type(TYPE),rem(REM)]).
obj([name_list([]),type(TYPE),rem(REM)]).
obj([name(NAME),type(TYPE),val(VAL),rem(REM)]):-lang(c),
        assert(p_obj([name(NAME),type(TYPE)])),
        assert(obj_type(NAME,TYPE)),
        type_member(TYPE,MEMBER),
```

```
        (member(NAME,MEMBER);
        add_type_entity([NAME,val(VAL)],TYPE,MEMBER)).
obj([name(NAME),type(string),array([dim(D),size(S),max_size(M)]),rem(REM)]):-
        lang(c),assert(p_obj([name(NAME),type(string),
        array([dim(D),size(S),max_size(M)])])),
        assert(obj_type(NAME,string)),
        type_member(char,MEMBER),
        (member([NAME,max_size(M)],MEMBER);
        add_type_entity([NAME,max_size(M)],char,MEMBER)).
obj([name(NAME),type(TYPE),array([dim(D),size(S),max_size(M)]),rem(REM)]):-
        lang(c),assert(p_obj([name(NAME),type(TYPE),
        array([dim(D),size(S),max_size(M)])])),
        assert(obj_type([NAME,max_size(M)],TYPE)),
        type_member(TYPE,MEMBER),
        (member([NAME,max_size(M)],MEMBER);
        add_type_entity([NAME,max_size(M)],TYPE,MEMBER)).
obj([name(NAME),type(TYPE),
        array([dim(D),size(S),max_size(M)]),val(VAL),rem(REM)]):-
        assert(p_obj([name(NAME),type(TYPE),
        array([dim(D),size(S),max_size(M)])])),
        lang(c),assert(obj_type(NAME,TYPE)),type_member(TYPE,MEMBER),
       (member(NAME,MEMBER);add_type_entity([NAME,val(VAL)],TYPE,MEMBER)).

add_type_entity(VAR,TYPE,MEMBER):-
            type_member(TYPE,MEMBER),retract(type_member(TYPE,MEMBER)),
            assert(type_member(TYPE,[VAR|MEMBER])).

type_decl:-
/*      (type_member([struct,X],[]):type_member([struct,STR_TYPE],STR),
        writelist(['struct    ',STR_TYPE,'   ']),
        writelist_ic(STR),write(';'),nl),
*/      (type_member(char,[]):type_member(char,CHAR),write_sp,
        write('char          '),
        write_namelist_ic(CHAR),write(';'),nl),
       (type_member(float,[]):type_member(float,FLOAT),write_sp,
        write('float       '),
        write_namelist_ic(FLOAT),write(';'),nl),
       (type_member(int,[]):type_member(int,INT),write_sp,
        write('int           '),
        write_namelist_ic(INT),write(';'),nl).


write_namelist_hc([H|T]):-write(','),write_name(H),write_namelist_hc(T).
write_namelist_hc([]).
write_namelist_ic([H|T]):-write_name(H),write_namelist_hc(T).
write_name([NAME,max_size([M,N])]):-writelist([NAME,'[',M,'][',N,']']).
write_name([NAME,max_size(M)]):-writelist([NAME,'[',M,']']).
write_name([NAME,val(VAL)]):-writelist([NAME,=,VAL]).
write_name([NAME,max_size(M),val(VAL)]):-nl,write('                    '),
                                writelist([NAME,'[',M,']={']),
                                writelist_ic(VAL),write('}'),nl.
write_name([NAME,max_size([M,N]),val(VAL)]):-nl,write('                    '),
                                writelist([NAME,'[',M,'][',N,']={']),
                                writellist_ic_nl(VAL),write('}'),nl.

write_name(NAME):-write(NAME).
type_member(char,[]).
type_member(str,[]).
type_member(float,[]).
type_member(int,[]).
member_check(X,TYPE):-type_member(TYPE,MEMBER),member(X,MEMBER).
```

```
/*  3．3  手続き述語からC命令文への変換 */

writelist([H|T]):-write(H),writelist(T).
writelist([]).
writelist_hc([]).
writelist_hc([H|T]):-write(','),write(H),writelist_hc(T).
writelist_ic([]).
writelist_ic([H|T]):-write(H),writelist_hc(T).
member(X,[X|L]).
member(X,[Y|L]):-member(X,L).
obj(x,type(int)).
fl_pntr('tstfl.dat',fp).


/*  3．3ａ  C命令文への変換  */

set(Y,X):-write_sp,writelist([Y,'=']),write_term(X),write(';'),nl.
write_term(X):-write(X).
read_from(keyboard,OBJ,type(TYPE)):-prompt_read_l(OBJ,type(TYPE)).

prompt_read_l([H_OBJ|T_OBJ],type([H_TYPE|T_TYPE])):-
            prompt_read(H_OBJ,type(H_TYPE)),
            prompt_read_l(T_OBJ,type(T_TYPE)).
prompt_read_l([],type([])).
prompt_read_l(OBJ,type(TYPE)):-prompt_read(OBJ,type(TYPE)).

prompt_read(OBJ,type(TYPE)):-lang(c),writelist(['printf("',OBJ,'=");']),nl,
            ptype(TYPE,PTYPE),writelist(['scanf("',PTYPE,"',',OBJ,');']),nl.
prompt_read(OBJ):-lang(c),included('stdio.h'),obj_type(OBJ,TYPE),
            writelist(['printf("',OBJ,'=");']),nl,
            ptype(TYPE,PTYPE),writelist(['scanf("',PTYPE,"',',OBJ,');']),nl.
c(print(OBJ)):-print(OBJ).
print(OBJ):-lang(c),obj_type(OBJ,TYPE),ptype(TYPE,PTYPE),
            write_sp,writelist(['printf("',OBJ,=,PTYPE,"',',OBJ,');']),nl.
ptypel([H_TYPE|T_TYPE],[H_PTYPE|T_PTYPE]):-
                ptype(H_TYPE,H_PTYPE),write(H_PTYPE),ptypel(T_TYPE,T_PTYPE).
ptypel([],[]).
ptypel(TYPE,PTYPE):-ptype(TYPE,PTYPE),write(PTYPE).

ptype(int,'%d').
ptype(float,'%f').
ptype(char,'%c').
ptype(string,'%s').
std_print_ptype(int,'%8d').
std_print_ptype(float,'%8f.3').
std_print_ptype(string,'%8s').
std_print_ptype(char,'%8c').
record('TSTREC',file('tstfl.dat')).
main_head:-(include([]);include(FILE),include_sent_gen(FILE)),
          main_header,type_decl.
include_sent_gen([H|T]):-
                (system(H), writelist(['#include <',H,'>']);
                file_drive(D),writelist(['#include <',D,":",H,'>'])),
                nl,include_sent_gen(T).
include_sent_gen([]).
included(FILE):-((include(OLD_FILE),member(FILE,OLD_FILE));
            add_file(FILE,OLD_FILE)).
include([]).
add_file(X,Y):-include(Y),retract(include(Y)),assert(include([X|Y])).
main_header:-main_arg(ARG),writelist(['main(',ARG,')']),nl,write('  {'),nl.
main_arg(' ').
main_end:-write('}').
```

```
compute_list([H|T]):-compute(H),compute_list(T).
compute_list([]).

c(compute_list([H|T])):-c(compute(H)),c(compute_list(T)).
c(compute_list([])).
compute(A):-lang(c),write_sp,write(A),write(';'),nl.
c(compute(A)):-lang(c),swrite(A),write(';'),nl.
j_dic(読み込む,[['OBJ',を読み込む],
                [プロンプトにより,'OBJ',を読み込む],
                ['LOC',から,'OBJ',を読み込む]]).
qj_dic:-j_dic(読み込む,X),writelist_ic_nl(X).
writelist_ic_nl([H|T]):-writelist_ic(H),nl,writelist_ic_nl(T).
writelist_ic_nl([]).
j([プロンプトにより,OBJ,を読み込む]):-c(prompt_read(OBJ)).
j([OBJ,を読み込む]):-c(read1(OBJ)).

prompt_read(OBJ):-lang(c),included('stdio.h'),
        (obj_type(OBJ,int,array(dim(1),size(N))),
        writelist(['read1adi(',OBJ,',',N,');']),nl;
        obj_type(OBJ,float,array(dim(1),size(N))),
        writelist(['read1adf(',OBJ,',',N,');']),nl).
obj_type(x,float).


/*  入出力ステートメントのコードに対する C 生成部  */

obj_type(n,int).
obj_type('a[i]',float).
c(prompt_read(OBJ)):-      /*lang(c),included('stdio.h'),*/
        writelist(['printf("',OBJ,'=");']),nl,
        c(read1(OBJ)).
c(prompt_read(OBJ,TYPE)):-lang(c),included('stdio.h'),
        writelist(['printf("',OBJ,'= in ',TYPE,");']),nl,
        c(read1(OBJ,TYPE)).

c(prompt_read_list(OBJ_LIST)):-lang(c),included('stdio.h'),
        write('printf("'),writelist_ic(OBJ_LIST),write('=");'),nl,
        c(read_list(OBJ_LIST)).
c(read1(OBJ)):-                    /*lang(c),included('stdio.h'),*/
    obj_type(OBJ,TYPE),ptype(TYPE,PTYPE),
    writelist(['scanf("',PTYPE,"',']),
    write_type_obj(TYPE,OBJ),write(');'),nl.
c(read1(OBJ,TYPE)):-lang(c),included('stdio.h'),
    ptype(TYPE,PTYPE),
    writelist(['scanf("',PTYPE,"',']),
    write_type_obj(TYPE,OBJ),write(');'),nl.
c(read_list(OBJ_LIST)):-lang(c),included('stdio.h'),
    obj_type_list(OBJ_LIST,TYPE_LIST),
    ptype_list(TYPE_LIST,PTYPE_LIST),
    write('scanf("'),writelist(PTYPE_LIST),write('"'),
    write_type_obj_list(TYPE_LIST,OBJ_LIST),write(');'),nl.
obj_type_list([HOBJ|TOBJ],[HTYPE|TTYPE]):-obj_type(HOBJ,HTYPE),
                                           obj_type_list(TOBJ,TTYPE).
obj_type_list([],[]).
c(print(OBJ)):-lang(c),included('stdio.h'),obj_type(OBJ,TYPE),
    ptype(TYPE,PTYPE),
    writelist(['printf("',PTYPE,'¥n",',OBJ,');']),nl.
c(print(OBJ,d)):-lang(c),obj_type(OBJ,TYPE),ptype(TYPE,PTYPE),
     writelist(['printf("',OBJ,'=',PTYPE,"',',OBJ,');']),nl.
c(print(OBJ,OP_TYPE)):-lang(c),
     writelist(['printf("',OBJ,'=',OP_TYPE,"',',OBJ,');']),nl.
c(print_headname_list(OBJ_LIST)):-write('printf("'),
    (p_headname_list(OBJ_LIST,HEADNAME_LIST),writelist(HEADNAME_LIST);
                writelist_it(OBJ_LIST)),write('¥n");'),nl.
writelist_ht([H|T]):-writelist(['¥t',H]),writelist_ht(T).
writelist_ht([]).
writelist_it([H|T]):-write(H),writelist_ht(T).
writelist_it(H):-write(H).

c(print_list(OBJ_LIST,d)):-lang(c),included('stdio.h'),
        (p_objlist(OBJ_LIST,PTYPE_LIST);
        obj_type_list(OBJ_LIST,TYPE_LIST),
        ptype_list(TYPE_LIST,PTYPE_LIST)),
        write('printf("'),writelist(PTYPE_LIST),write('¥n"'),
        writelist_hc(OBJ_LIST),write(');'),nl.
c(print_list(OBJ_LIST,OP_TYPE_LIST)):-lang(c),included('stdio.h'),
        write('printf("'),writelist(OP_TYPE_LIST),write('¥n"'),
        writelist_hc(OBJ_LIST),write(');'),nl.
c(print_name_value(OBJ)):-lang(c),included('stdio.h'),obj_type(OBJ,TYPE),
        ptype(TYPE,PTYPE),
        writelist(['printf("',OBJ,'=',PTYPE,'¥n",',OBJ,');']),nl.
c(print_name_value_list(OBJ_LIST,d)):-lang(c),included('stdio.h'),
        obj_type_list(OBJ_LIST,TYPE_LIST),
        ptype_list(TYPE_LIST,PTYPE_LIST),
        repeat_write_l1(OBJ_LIST,PTYPE_LIST).
c(print_name_value_list(OBJ_LIST,OP_TYPE_LIST)):-lang(c),included('stdio.h'),
        repeat_write_l1(OBJ_LIST,OP_TYPE_LIST).
repeat_write_l1([HOBJ|TOBJ],[HPTYPE|TPTYPE]):-
        writelist(['printf("',HOBJ,'=',HPTYPE,'¥n",',HOBJ,');']),nl,
        repeat_write_l1(TOBJ,TPTYPE).
repeat_write_l1([],[]).
ptype_list([HTYPE|TTYPE],[HPTYPE|TPTYPE]):-ptype(HTYPE,HPTYPE),
                                            ptype_list(TTYPE,TPTYPE).
ptype_list([],[]).
std_print_ptype_list([HTYPE|TTYPE],[HPTYPE|TPTYPE]):-
        std_print_ptype(HTYPE,HPTYPE),std_print_ptype_list(TTYPE,TPTYPE).
std_print_ptype_list([],[]).

write_type_obj(int,OBJ):-writelist(['&',OBJ]).
write_type_obj(float,OBJ):-writelist(['&',OBJ]).
write_type_obj(char,OBJ):-write(OBJ).
write_type_obj(string,OBJ):-write(OBJ).
write_type_obj_list([HTYPE|TTYPE],[HOBJ|TOBJ]):-
                write(','),write_type_obj(HTYPE,HOBJ),
                write_type_obj_list(TTYPE,TOBJ).
write_type_obj_list([],[]).
c(print_string_list([H|T])):-c(print_string(H)),c(print_string_list(T)).
c(print_string_list([])).
c(print_pattern_string(S)):-
        (pattern_name(PAT_NAME);writelist(['printf("',S,");'])),nl.
c(print_string(S)):-
        writelist(['printf("',S,");']),nl.
pattern_name(dotted_line):-
        write('printf("----------¥n");').
c(print_const_obj(S,OBJ,d)):-
        pobj_type(OBJ,TYPE),ptype(TYPE,PTYPE),
        writelist(['printf("',S,PTYPE,'¥n",',OBJ,');']),nl.
c(print_const_obj(S,OBJ,OP_TYPE)):-
        writelist(['printf("',S,OP_TYPE,'¥n",',OBJ,');']),nl.
obj_type(hinmei,string).
obj_type(tanka,int).
obj_type(suuryou,int).
p_objlist([hinmei,tanka,suuryou],[%10s','%6d','%8%']).
obj_type(sum,int).
```

```
/* 制御ステートメントのコードに対する C 生成部 */

c(if(rel(L,REL,R),X)):-write('if('),rel(L,REL,R),write(')'),nl,
                        write('    '),proc_list(X),nl.
c(if(conv(T),X)):-write_sp,write('if('),conv(T),write('){'),nl,add_sp,
                write_sp,proc_list(X),
                subt_sp,write_sp,write('}'),nl.
c(if(T,X)):-write_sp,write(if(T)),write('{'),nl,add_sp,proc_list(X),
            subt_sp,write_sp,write('}'),nl.
c(if(T,X1,X2)):-write(if(T)),nl,write('    '),proc_list(X1),write('else'),nl,
                write('    '),proc_list(X2).

c(for(I,minus(N,1),S,P)):-write_sp,
                writelist(['for(',I,'=0;',I,'<',N,';',I,'+=',S,'){']),nl,
                add_sp,(proc_list(P):P),
                subt_sp,write_sp,write('}'),nl.
c(for(I,M,minus(N,1),S,P)):-
                swritelist(['for(',I,'=',M,';',I,'<',N,';',I,'+=',S,'){']),nl,
                add_sp,proc_list(P),subt_sp,write_sp,swrite('}'),nl.

sp(0).
add_sp:-sp(SP),retract(sp(SP)),SP1 is SP+3,assert(sp(SP1)).
subt_sp:-sp(SP),retract(sp(SP)),SP1 is SP-3,assert(sp(SP1)).
set_sp(X):-retract(sp(Y)),assert(sp(X)).
write_sp(0).
write_sp(N):-write(' '),M is N-1,write_sp(M).
write_sp:-sp(N),write_sp(N).
swrite(X):-write_sp,write(X).
swritelist(X):-write_sp,writelist(X).
/*c(for(I,II,T,S,P)):-writelist(['for(',I,'=',II,';',T,';',I,'+=',S,'){']),nl,
                write('    '),proc(P),write('}'),nl.
*/
c(for(I,II,T,S,P)):-writelist(['for(',I,'=',II,';',T,';',I,'+=',S,'){']),nl,
                write('    '),proc_list(P),write('}'),nl.
c(while(T,P)):-write_sp,writelist(['while(',T,'){']),nl,add_sp,proc_list(P),
                subt_sp,write_sp,write('}'),nl.
c(until(OBJ,T,P)):-write('while(1) {'),nl,
            c(read1(OBJ)),
            writelist(['    ','if(',T,') break;']),nl,proc_list(P),write('}'),nl.
c(return(X)):-write_sp,writelist(['return(',X,');']),nl.
c(case(INDEX_NAME,BODY)):-writelist(['switch(',INDEX_NAME,'){']),nl,
                        case_body(BODY),write('}'),nl.
case_body([[item(NUM),H,end_case]|T]):-writelist(['    case ',NUM,'    :']),
            proc_list(H),writelist(['                ','break;']),nl,case_body(T).
case_body([]).
c(when(OBJ_NAME,BODY)):-if_else_if(BODY).
/* c(X):-write(X).   */
if_else_if([[HT,HP,end_when]|T]):-c(if(HT,HP)),else_if(T).

else_if([[else,HP,end_when]|[]]):-write('else '),nl,write('    '),proc(HP).
else_if([[HT,HP,end_when]|T]):-write('else '),c(if(HT,HP)),else_if(T).
else_if([]).
c(if_then_else_if([[HT,HP]|T],DEFAULT)):-c(if(HT,HP)),else_if(T),
                                        if_default(DEFAULT).
else_if([[else,HP]|[]]):-write('else '),nl,write('    '),proc(HP).
if_default(D):-write('else '),nl,write('    '),proc(D).
default(D):-write('    default :'),proc_list(D).


/* 制御ステートメントからの　C生成のチェック */

qc0:-c(for(i,minus(n1,1),1,[for(j,minus(m1,1),1,compute('x[i][j]=i*j'))])).
qc1:-c(if(x>0),[[compute(z=x+a)],[compute(z=x-a)]]).
qc2:-c(if(x>0,[compute(z=x+a)])).
qc3:-c(case(i,[[item(1),compute(z=a),end_case],
                [item(2),compute(z=log(a)),end_case],
                [item(3),compute(z=0),end_case]])).
qc3a:-c(case_read_run(i,[[1,compute(z=a)],[2,compute(z=log(a))]],
        compute(z=0))).
qc4:-c(if_then_else_if([['x>=a1',[compute(z=y+b1)]],
                        ['x>=a2',[compute(z=y+b2)]],
                        ['x>=a3',[compute(z=y+b3)]]],[compute(z=y+b4)])).
qc41:-c(when(x,[['x>=a1',compute(z=y+b1)],['x>=a2',compute(z=y+b2)],
                        ['x>=a3',compute(z=y+b3)],[else,compute(z=y+b4)] ])).
qc5:-c(for(i,minus(n,1),1,compute('z=z+a[i]*b[i]'))).
qc6:-c(for(i,n,'i<=m',1,compute('z=z+a[i]*b[i]'))).
qc7:-c(while(x>0,compute(x=f(x,a)))).
qc8:-c(until(hinmei,'hinmei=="¥n"',
        [read1(tanka),read1(suuryou),print(hinmei),
          compute(kingaku=tanka*suuryou),compute(sum=sum+kingaku)])).
qc9:-c(read_list([hinmei,tanka,suuryou])).
qc10:-c(print_name_value_list([hinmei,tanka,suuryou])).
qc11:-c(print_list([hinmei,tanka,suuryou])).
qc12:-c(print_headname_list([hinmei,tanka,suuryou])).
qc13:-c(prompt_read_list([hinmei,tanka,suuryou])).
qs1:-proc([if_then_else_if['最初に条件に合う処理を実行'],
                    [['x>=a1',compute(z=y+b1),end_when],
                     ['x>=a2',compute(z=y+b2),end_when],
                     ['x>=a3',compute(z=y+b3),end_when]],
                    compute(z=y+b4),
                   end_if]).
qs2:-proc([case(index),
                    [[item(1),compute(z=a),end_case],
                     [item(2),compute(z=log(a)),end_case],
                     [item(3),compute(z=0),end_case]],end_case_list]).
qs3:-proc([for['0  から',minus(n,1),'まで',1,'づつ増加する',i,
                'に対して'],compute('z=z+a[i]*b[i]'),end_for]).

qs4:-proc([for[i,を,0,から,m,まで,1,づつ変化しながら繰り返す],
            compute('z=z+a[i]*b[i]'),end_for]).

qs5:-proc([while[（前に）条件,x>0,が成立する],compute(x=f(x,a)),end_while]).
qs6:-proc[until[hinmei,を読み込み,'hinmei == ¥n',が成立するまで繰り返す],
        [read1(tanka),read1(suuryou),print(hinmei),
          compute(kingaku=tanka*suuryou),compute(sum=sum+kingaku)],end_until]).

qs4a:-proc[[[for,[i,を,0,から,m,まで,1,づつ変え繰り返す]],
            compute('z=z+a[i]*b[i]'),end_for]).



qpr1:-proc[[プロンプトにより１次元配列,x,に読み込む]].
qpr1a:-write('qpr1a:-proc([a,に読み込む]).:-'),nl,proc([a,に読み込む]).
qpr1b:-write('qpr1b:-proc([プロンプトにより,a,に読み込む]).:-'),nl,
        proc([プロンプトにより,a,に読み込む]).
qpr1c:-write('qpr1c:-proc([a,を,%10.3,でプリントする]).:-'),nl,
        proc([a,をプリントする]).
qpr2:-write('qpr2:-proc([x,の和を,sum_x,に求める]).:-'),nl,
        proc([x,の和を,sum_x,に求める]).
qpr3:-write('qpr3:-proc([x,の平均値を,av_x,に求める]).:-'),nl,
        proc([x,の平均値を,av_x,に求める]).
qpr4:-write('qpr4:-proc([x,の偏差値を,dev_x,に求める]).:-'),nl,
        proc([x,の偏差値を,dev_x,に求める]).
qpr5:-write('qpr5:-proc([x,の最大値を,max_x,に求める]).:-'),nl,
        proc([x,の最大値を,max_x,に求める]).
qpr6:-write('qpr6:-proc([x,の最小値を,min_x,に求める]).:-'),nl,
        proc([x,の最小値を,min_x,に求める]).
```

```
qpr7:-write('qpr9:-proc(for([n,から,m,までの,i,に対して])).:-'),nl,
        proc([for([i,を,n,から,m,まで,1,づつ変化しながら繰り返す]),
              compute('sum=sum+x[i]'),end_for]).
obj_type(index,int).


statement([jp([FILE_PNTR,により,FN,を,RW,用にオープンする],
           file_open(FN,FILE_PNTR,RW)),
           arg_type([name([FN,FILE_PNTR,RW]),type(char)]),in([]),out([]) ]).
c(file_open(FN,FILE_PNTR,RW)):-
   write_sp,writelist([FN,'=fopen(',FILE_PNTR,',"',RW,'");']),nl.
statement([jp(['プロンプトにより,OBJ,に読み込む],prompt_read(OBJ)),
         arg_type([name(OBJ),type(TYPE)]),in([]),out([])).
statement([jp(['プロンプトにより',OBJ_LIST,の各変数に読み込む],
              prompt_read_list(OBJ_LIST)),
              arg_type([name(OBJ_LIST),type(TYPE_LIST)]),in([]),out([])]).

statement([jp(['プロンプトにより',OBJ_LIST,の各変数に一つ宛読み込む],
              prompt_read_list_seq(OBJ_LIST)),
              arg_type([name(OBJ_LIST),type(TYPE_LIST)]),in([]),out([])]).

statement([jp([OBJ,に読み込む],read1(OBJ)),
           arg_type([name(OBJ),type(TYPE)]),in([]),out([])).
statement([jp([OBJ,を,OP_TYPE,でプリントする],print(OBJ,OP_TYPE)),
           arg_type([name(OBJ),type(TYPE)]),in([]),out([])).
statement([jp([OBJ_LIST,の各値を,OP_TYPE_LIST,で名前に続き各行にプリントする],
              print_name_value_list(OBJ_LIST,OP_TYPE_LIST)),
              arg_type([name(OBJ_LIST),type(TYPE_LIST)]),in([]),out([])]).
statement([jp([OBJ_LIST,の各値の見出し行をタブによりプリントする],
              print_headname_list(OBJ_LIST)),
              arg_type([name(OBJ_LIST),type(STRING)]),in([]),out([])]).
statement([jp([OBJ_LIST,の各値を,OP_TYPE_LIST,で一行にプリントする],
              print_list(OBJ_LIST,OP_TYPE_LIST)),
              arg_type([name(OBJ_LIST),type(TYPE_LIST)]),in([]),out([])]).
statement([jp([定記号パターン列,S,をプリントする],print_pattern_string(S)),
              arg_type([name(S),type(string)]),in([]),out([])]).
statement([jp([定記号列,S,をプリントする],print_string(S)),
              arg_type([name(S),type(string)]),in([]),out([])]).
statement([jp([定記号列,S,をリスト要素毎に改行してプリントする],
              print_string_list(S)),
              arg_type([name(S),type(STR_LIST)]),in([]),out([])]).
statement([jp([定記号列,S,と変項,X,を,OP_TYPE,でプリントする],
              print_const_obj(S,X,OP_TYPE)),
              arg_type([[name(S),type(string)],[name(X),type(TYPE)]]),
             in([]),out([])]).
statement([jp([X,に,Y,を加えた結果を,Z,におく],plus(X,Y,Z)),
           in([]),out([])).
statement([jp([if(T),then,X,end_if],if(T,X)),in([]),out([])]).

statement([jp(if(T),if(T)),in([]),out([])).
statement([jp([if(T),then,X1,else,X2,end_if],if(T,X1,X2)),
           in([]),out([])).
statement([jp([[if,T],then,X1,end_if],if(T,X1)),
           in([]),out([])).
statement([jp([[if,T],then,X1,else,X2,end_if],if(T,X1,X2)),
           in([]),out([])).
statement([jp([if_then_else_if([最初に条件に合う処理を実行]),L,DEFAULT,
           end_if],if_then_else_if(L,DEFAULT)),in([]),out([])]).
statement([jp([case(INDEX_NAME),L,end_case_list],
           case(INDEX_NAME,L)),in([]),out([])]).
statement([jp([case_read_run([CASE]),L,DEFAULT,end_case],
           case_read_run(CASE,L,DEFAULT)),in([]),out([])]).
statement([jp([when(OBJ_NAME),L,end_when_list],
           when(OBJ_NAME,L)),in([]),out([])]).
statement([jp([for([初期条件を,INIT,とし,STEP,の処理をしながら,REP_COND,
           の継続条件が成立する間繰り返す]),P,end_for],
           for(INIT,REP_COND,STEP,P)),in([]),out([])]).
statement([jp([for([INDEX,を,INIT_VAL,から,TERM_VAL,まで,STEP,
           'づつ変化しながら繰り返す]),P,end_for],
           for(INDEX,INIT_VAL,TERM_VAL,STEP,P)),in([]),out([])]).
statement([jp([for([INDEX,を,INIT_VAL,から,TERM_VAL,まで,STEP,
           'づつ変え繰り返す]),P,end_for],
           for(INDEX,INIT_VAL,TERM_VAL,STEP,P)),in([]),out([])]).
statement([jp([[for,[INDEX,を,INIT_VAL,から,TERM_VAL,まで,STEP,
           'づつ変え繰り返す']],P,end_for],
           for(INDEX,INIT_VAL,TERM_VAL,STEP,P)),in([]),out([])]).

statement([jp([for([M,から,minus(N,1),まで,S,'づつ増加する',I,
            'に対して']),P,end_for],for(I,M,minus(N,1),S,P)),in([]),out([])]).
statement([jp([while([（前に）条件,T,が成立する]),P,end_while],while(T,P)),
           in([]),out([])]).
statement([jp([until([ITEM,を読み込み,T,が成立するまで繰り返す]),P,end_until],
           until(ITEM,T,P)),in([]),out([])]).
statement([jp(set(Y,X),set(Y,X)),in([]),out([])]).
statement([jp([X,を返す],return(X)),in([]),out([])]).
statement([jp(then([このとき]),then),in([]),out([])]).
statement([jp(else([他方]),else),in([]),out([])]).
statement([jp(X,X),in([]),out([])]).
statement([jp(X,X),arg_type([name(OBJ),type(TYPE)]),in([]),out([])]).
obj_type(a,int).

p_obj([name(x),type(float),array([dim(1),size(n),max_size(200)])]).
p_obj([name(sum_x),type(float)]).
p_obj([name(av_x),type(float)]).
p_obj([name(dev_x),type(float)]).
p_obj([name(max_x),type(float)]).
p_obj([name(min_x),type(float)]).
p_obj([name(a),type(float)]).
p_obj([name(kosuu),type(int)]).

qproc1a:-proc([[for,[i,を,0,から,'n-1',まで,1,'づつ変え繰り返す']],[compute(x=x+a)],end_for]).
qproc2a:-proc([[if,'x>0'],then,[compute(y=x+a)],else,[compute(y=x+b)],end_if]).
qproc3a:-proc([[if,'x>0'],then,[compute(y=x+a)],end_if]).


proc(X):-statement(S),member(jp(X,Y),S),
   member(arg_type([name(OBJ),type(TYPE)]),S),p_obj([name(OBJ),type(TYPE)]),
   c(Y).
p_obj([name(n),type(int)]).
p_obj([name('a[i]'),type(float)]).


type_check_argl([H|T]):-type_check(H),type_check_argl(T).
type_check_argl([]).
type_check(X):-p_obj(X).

proc(X):-
   module(H,T),member(jp(X,Y),H),
   member(argl(ARGL),H),
    type_check_argl(ARGL),
    member(HA,ARGL),member(type(TYPE),HA),
   member(funct(M),T),member(['TYPE,PROC'],M),writelist(PROC),write(';'),nl,
   member(file_name(FNL),T),member(['TYPE,FN'],FNL),included(FN),
   (member(funct_type([]),T);
            member(funct_type(FT),T),
            member(['TYPE,FUNCT_NAME],FT),funct_type_decl(['TYPE,FUNCT_NAME])).
```

```
proc(X):-statement(S),member(jp(X,Y),S),c(Y).
proc_a(X):-statement(S),member(jp(X,Y),S),write(Y).
read_memberlist(X,[H|T]):-read_member(X,H),read_memberlist(X,T).
read_memberlist(X,[]).
read_member([X,I],[MEMBER,TYPE]):-
                prompt_read([X,I,OBJ],TYPE).
read_member(X,[MEMBER,TYPE]):-
                prompt_read([X,[MEMBER]],TYPE).

/*
module([
 jp([LOWER,から,UPPER,までの一様乱数を,NUM,個,RAN,を seed として発生し配列,A,
        に入れる],ran1a(A,NUM,LOWER,UPPER,RAN)),
 argl([[name(A),type(TYPE),array([dim(1),size(N),max_size(NMAX)]) ],
       [name(NUM),type(int)],[name(LOWER),type(TYPE)],[name(UPPER),type(TYPE)],
       [name(RAN),type(int)]])],
 [funct([int,[ran1adi(A,NUM,LOWER,UPPER,RAN)]],[float,
       [ran1adf(A,NUM,LOWER,UPPER,RAN)]])],
  file_name([int,'ran1adi.c'],[float,'ran1adf.c']),
  funct_type([[int,'ran1adi()'],[float,'ran1adf()']],in([]),out([]) ]).

module([
 jp([LOWER,から,UPPER,までの一様乱数を,RAN,を seed として発生し,N,行,M,列の,
       ２次元配列,A,に入れる],ran1a(A,N,M,LOWER,UPPER,RAN)),
 argl([[name(A),type(TYPE),array([dim(2),size([N,M]),max_size([NMAX,MMAX])]) ],
       [name(N),type(int)],[name(M),type(int)],
       [name(LOWER),type(TYPE)],[name(UPPER),type(TYPE)],
       [name(RAN),type(int)] ])],
 [funct([[int,[ran2adi(A,N,M,LOWER,UPPER,RAN)]],
        [float,[ran2adf(A,N,M,LOWER,UPPER,RAN)]]]),
  file_name([[int,'ran2adi.c'],[float,'ran2adf.c']]),
  funct_type([[int,'ran2adi()'],[float,'ran2adf()']],in([]),out([]) ]).


module([
 jp([プロンプトにより構造体,X,に読み込む],prompt_read_struct(X)),
 argl([name(X),type(struct(TYPE)) ]) ],
 [funct([read_struct(X)]),
  file_name(['rdstruct.c']),
  funct_type([]),in([]),out([])].
module([
 jp([プロンプトにより１次元配列,OBJ,に読み込む],prompt_read(OBJ)),
 argl([[name(OBJ),type(TYPE),array([dim(1),size(N),max_size(NMAX)])]])],
 [funct([[int,[read1adi(OBJ,N)]],[float,[read1adf(OBJ,N)]]]),
  file_name([[int,'rd1adi.c'],[float,'rd1adf.c']]),
  funct_type([]),in([]),out([])].
module([
 jp([プロンプトにより１次元配列,OBJ,に読み込む],prompt_read(OBJ)),
 argl([[name(OBJ),type(TYPE),array([dim(1),size(N),max_size(NMAX)])]])],
 [funct([[int,[read1adi(OBJ,N)]],[float,[read1adf(OBJ,N)]]]),
  file_name([[int,'rd1adi.c'],[float,'rd1adf.c']]),
  funct_type([]),in([]),out([])].
module([
 jp([プロンプトにより２次元配列,OBJ,に読み込む],prompt_read(OBJ)),
 argl([[name(OBJ),type(TYPE),
        array([dim(2),size([N,M]),max_size([NMAX,MMAX])]) ])],
 [funct([[int,[read2adi(OBJ,N,M)]],[float,[read2adf(OBJ,N,M)]]]),
  file_name([[int,'rd2adi.c'],[float,'rd2adf.c']]),
  funct_type([]),in([]),out([])].

module([
 jp([ファイル名,FN,の第,K,番目のファイルから,２次元配列,A,の第,J,列に読み込む
],read_col_fl(A,N,M,J,FN,K)),
 argl([[name(A),type(TYPE),array([dim(2),size([N,M]),max_size([NMAX,MMAX]) ])],
       [name(J),type(int)],[name(FN),type(char)],[name(K),type(int)]  ])],
 [funct([[int,[rdcolfli(A,N,M,J,FN,K)]],[float,[rdcolflf(A,N,M,J,FN,K)]]]),
  file_name([[int,'rdcolfli.c'],[float,'rdcolflf.c']]),
  funct_type([]),in([]),out([]) ].

module([
 jp([１次元配列,OBJ,をプリントする],prt1ad(OBJ)),
 argl([[name(OBJ),type(TYPE),array([dim(1),size(N),max_size(NMAX)])]])],
 [funct([[int,[prt1adi(OBJ,N)]],[float,[prt1adf(OBJ,N)]]]),
  file_name([[int,'prt1adi.c'],[float,'prt1adf.c']]),
  funct_type([]),in([]),out([])].
module([
 jp([２次元配列,OBJ,をプリントする],prt2ad(OBJ)),
 argl([[name(OBJ),type(TYPE),array([dim(2),size(N,M),max_size(NMAX,MMAX)])]])],
 [funct([[int,[prt2adi(OBJ,N,M)]],[float,[prt2adf(OBJ,N,M)]]]),
  file_name([[int,'prt2adi.c'],[float,'prt2adf.c']]),
  funct_type([]),in([]),out([])].

module([
 jp([OBJ,の最大値を,MAXOBJ,に求める],max(OBJ,MAXOBJ)),
 argl([[name(OBJ),type(TYPE),array([dim(1),size(N),max_size(NMAX)])],
       [name(MAXOBJ),type(TYPE)]),
       default(MAXOBJ,[max_,OBJ])],
 [funct([[int,[max_,OBJ,=,max1adi(OBJ,N)]],
         [float,[MAXOBJ,=,max1adf(OBJ,N)]]]),
  file_name([[int,'max1adi.c'],[float,'max1adf.c']]),
  funct_type([[int,'max1adi()'],[float,'max1adf()']],in([]),out([])].

module([
 jp([２次元配列,OBJ,の第,C,列の最大値を,MAXOBJ,に求める],max(OBJ,C,MAXOBJ)),
 argl([[name(OBJ),type(TYPE),array([dim(2),size([N,M]),
 max_size([NMAX,MMAX])])],
       [name(C),type(int)],[name(MAXOBJ),type(TYPE)]),
       default(MAXOBJ,[max_,OBJ])],
 [funct([[int,[max_,OBJ,=,max2adi(OBJ,C,N)]],
         [float,[MAXOBJ,=,max2adf(OBJ,C,N)]]]),
  file_name([[int,'max2adi.c'],[float,'max2adf.c']]),
  funct_type([[int,'max2adi()'],[float,'max2adf()']],in([]),out([])].

module([
 jp([OBJ,の最小値を,MINOBJ,に求める],min(OBJ,MINOBJ)),
 argl([[name(OBJ),type(TYPE),array([dim(1),size(N),max_size(NMAX)])],
        [name(MINOBJ),type(TYPE)]),default(MINOBJ,[min_,OBJ])],
 [funct([[int,[MINOBJ,=,min1adi(OBJ,N)]],
         [float,[min_,OBJ,=,min1adf(OBJ,N)]]]),
  file_name([[int,'min1adi.c'],[float,'min1adf.c']]),
  funct_type([[int,'min1adi()'],[float,'min1adf()']],in([]),out([])].

module([
jp([２次元配列,OBJ,の第,C,列の最小値を,MINOBJ,に求める],min(OBJ,C,MINOBJ)),
argl([[name(OBJ),type(TYPE),array([dim(2),size([N,M]),max_size([NMAX,MMAX])])],
       [name(C),type(int)],[name(MINOBJ),type(TYPE)] ]),
      default(MINOBJ,[min_,OBJ])],
 [funct([[int,[MINOBJ,=,min2adi(OBJ,C,N)]],
         [float,[min_,OBJ,=,min2adf(OBJ,C,N)]]]),
  file_name([[int,'min2adi.c'],[float,'min2adf.c']]),
  funct_type([[int,'min2adi()'],[float,'min2adf()']],in([]),out([])].


module([
  jp([A,を,ORD,順にソートする],sort(A,ORD)),
  argl([[name(A),type(TYPE),array([dim(1),size(N),max_size(NMAX)])],
```

```
        [name(N),type(int)],
        [name(ORD),type(char)]])],
  [funct([[int,[sort1adi(A,N,ORD)]],
          [float,[sort1adf(A,N,ORD)]]]),
   file_name([[int,'sort1adi.c'],[float,'sort1adf.c']]),
   funct_type([[int,'sort1adi(0']],[float,'sort1adf(0']]),in([]),out([])).

module([
 jp([２次元配列,A,を,ORD,順に第,K,列の要素をキーとしてソートする],
 sort(A,K,ORD)),
 argl([[name(A),type(TYPE),array([dim(2),size([N,M]),max_size([NMAX,MMAX]) ])],
       [name(K),type(int)],[name(N),type(int)],
        [name(M),type(int)],[name(ORD),type(char)] ]) ],
 [funct([[int,[sort2adi(A,K,N,M,ORD)]],
        [float,[sort2adf(A,K,N,M,ORD)]]]),
   file_name([[int,'sort2adi.c'],[float,'sort2adf.c']]),
   funct_type([[int,'sort2adi(0']],[float,'sort2adf(0']]),in([]),out([])).

module([
 jp([OBJ,の和を,SUMOBJ,に求める],sum(OBJ,SUMOBJ)),
 argl([[name(OBJ),type(TYPE),array([dim(1),size(N),max_size(NMAX)])],
       [name(SUMOBJ),type(TYPE)]]),
       default(SUMOBJ,[sum_,OBJ])],
 [funct([[int,[sum_,OBJ,=,sum1adi(OBJ,N)]],
         [float,[sum_,OBJ,=,sum1adf(OBJ,N)]]]),
  file_name([[int,'sum1adi.c'],[float,'sum1adf.c']]),
  funct_type([[int,[]],[float,'sum1adf(0']]),in([]),out([])).


module([
 jp([２次元配列,OBJ,の第,C,列の和を,SUMOBJ,に求める],sum(OBJ,C,SUMOBJ)),
 argl([[name(OBJ),type(TYPE),
        array([dim(2),size([N,M]),max_size([NMAX,MMAX])])],
       [name(C),type(int)],[name(SUMOBJ),type(TYPE)] ]),
       default(SUMOBJ,[sum_,OBJ])],
 [funct([[int,[sum_,OBJ,=,sum2adi(OBJ,N,C)]],
         [float,[sum_,OBJ,=,sum2adf(OBJ,N,C)]]]),
  file_name([[int,'sum2adi.c'],[float,'sum2adf.c']]),
  funct_type([[int,'sum2adi(0']],[float,'sum2adf(0']]),in([]),out([])).

module([
 jp([２次元配列,OBJ,の,FROM_C,列から,TILL_C,列までの,ROW,
     行の要素の和を,SUMOBJ,に求める],sum(OBJ,FROM_C,TILL_C,ROW,SUMOBJ)),
 argl([[name(OBJ),type(TYPE),
        array([dim(2),size([N,M]),max_size([NMAX,MMAX]) ])],
       [name(SUMOBJ),type(TYPE)],[name(FROM_C),type(int)],
       [name(TILL_C),type(int)],[name(ROW),type(int)]]),
       default(SUMOBJ,[sum_,OBJ ])],
 [funct([[int,[SUMOBJ,=,sumrowi(OBJ,ROW,FROM_C,TILL_C) ]],
         [float,[SUMOBJ,=,sumrowf(OBJ,ROW,FROM_C,TILL_C)]] ]),
  file_name([[int,'sumrowi.c'],[float,'sumrowf.c']]),
  funct_type([[int,'sumrowi(0']],[float,'sumrowf(0']]),in([]),out([])).

module([
 jp([OBJ,の平均値を,AVOBJ,に求める],av(OBJ,AVOBJ)),
 argl([[name(OBJ),type(float),array([dim(1),size(N),max_size(NMAX)])],
       [name(AVOBJ),type(TYPE)]]),
       default(AVOBJ,[av_,OBJ])],
 [funct([[float,[AVOBJ,=,av1adf(OBJ,N)]]]),
  file_name([[float,'av1adf.c']]),
  funct_type([[float,'av1adf(0']]),in([]),out([])).

module([
 jp([２次元配列,OBJ,の第,C,列の平均値を,AVOBJ,に求める],av(OBJ,C,AVOBJ)),
 argl([[name(OBJ),type(float),
        array([dim(2),size([N,M]),max_size([NMAX,MMAX])])],
       [name(C),type(int)],[name(AVOBJ),type(TYPE)] ]),
       default(AVOBJ,[av_,OBJ])],
 [funct([[float,[AVOBJ,=,av2adf(OBJ,N,C)]]]),
  file_name([[float,'av2adf.c']]),
  funct_type([[float,'av2adf(0']]),in([]),out([])).

module([
 jp([OBJ,の偏差値を,DEVOBJ,に求める],dev(OBJ,DEVOBJ)),
 argl([[name(OBJ),type(float),array([dim(1),size(N),max_size(NMAX)])],
       [name(AVOBJ),type(TYPE)]]),
       default(DEVOBJ,[dev_,OBJ])],
 [funct([[float,[dev_,OBJ,=,dev1adf(OBJ,N)]]]),
  file_name([[float,'dev1adf.c']]),
  funct_type([[float,'dev1adf(0']]),in([]),out([])).

module([
 jp([２次元配列,OBJ,の第,C,列の偏差値を,DEVOBJ,に求める],dev(OBJ,C,DEVOBJ)),
 argl([[name(OBJ),type(float),array([dim(2),size([N,M]),max_size([NMAX,M])])],
       [name(C),type(int)],[name(AVOBJ),type(TYPE)] ]),
       default(DEVOBJ,[dev_,OBJ])],
 [funct([[float,[dev_,OBJ,=,dev2adf(OBJ,C,N)]]]),
  file_name([[float,'dev2adf.c']]),
  funct_type([[float,'dev2adf(0']]),in([]),out([])).

module([
 jp([N,は素数である],prime(N)),
 argl([[name(N),type(int)]])  ],
 [funct([[int,[prime(N)]]]),
  file_name([[int,'prime.c']]),
  funct_type([[int,'prime(0']]),in([]),out([])).

*/
funct_type_decl([int,X]).
funct_type_decl([TYPE,FUNCT_NAME]):-type_member(TYPE,MEMBER),
            (member(FUNCT_NAME,MEMBER);
              add_type_entity(FUNCT_NAME,TYPE,MEMBER)).
```

謝　辞

本研究を遂行するにあたり，充実した研究環境を与えて頂きました学校法人金井学園 金井 兼 理事長に深く御礼申し上げます.

また，論文作成に際し様々なご支援を頂きました福井工業大学学長 城野政弘教授， 副学長 森島洋太郎教授，いつも暖かい励ましを下さいました経営情報学科主任で諮問委員長の梅野正隆教授，終始手厚いご指導，ご鞭撻を賜りました審査委員主査の魚崎勝司教授，審査委員の山西輝也教授，金江春植教授，そして諮問委員の鹿間敏弘教授，古庄純次教授に対しまして深く感謝申し上げます．本当にありがとうございました.

さらに，福井工業大学に赴任後ずっとご指導を頂き研究者への道筋を付けて頂きました故 西田富士夫先生（元福井工業大学教授），大学時代からずっとご指導頂いた松本忠先生（元福井工業大学教授），そして福井大学の茂呂征一郎准教授，また数々のご助言を頂いた電気電子情報工学科の佐々木弘教授，ならびにいつも応援して下さった経営情報学科の各先生および福井工業大学のお世話になったすべての先生方に深く感謝申し上げます.

最後に，研究を進めるにあたり，時間的に多大なご協力を頂き暖かく見守って下さった事務局の皆さまに対し心より感謝申し上げます.

2012 年 9 月　　恐神正博