論　文

順序機械型プログラムの階層的設計法と在庫管理プログラムの開発例

正　員　岡野　浩三[†]　　正　員　北道　淳司[†]
正　員　東野　輝夫[†]　　正　員　谷口　健一[†]

Hierarchical Design of Abstract Sequential Machine Style Program and Its Application to Development of Stock Management Program

Kozo OKANO[†], Junji KITAMICHI[†], Teruo HIGASHINO[†]
and Kenichi TANIGUCHI[†], *Members*

あらまし　本論文では代数的手法による順序機械型プログラムの階層的設計法について提案する．我々の階層的設計法では要求仕様から実現プログラムまで同一の詳細化法に基づいて段階的に順次詳細化していく．各レベルでは，そのレベルでの関数，処理の性質等の要求記述が行われ，そのレベルで閉じた記述になっている．我々は，プログラム設計技法の共通問題として提供された在庫管理問題に対し本手法を適用し，他の文献では見られなかった入出力関係のみを指定した要求記述から，5段階にわたって逐次詳細化し，プログラムを開発した．本論文では記述の概略と共に，要求記述や詳細化の際に一般的に生じる問題点，解決法を述べ，また，他の方法論との比較も行っている．「拡張射影」と呼ばれる単純な枠組で最上位の満たすべき入出力関係を記述できること，記述は完全に階層的であること，各レベルの要求記述はいわゆるオーバスペックにならないように必要なことのみ記述していること，各レベルの要求記述に抜けがないよう記述スタイルを工夫していること，正しさの形式的な証明が可能であったこと，実現プログラムの実行時間はCプログラムのそれのたかだか2倍程度であること，等の結果より，本設計法および処理系の有効性が確かめられた．

キーワード　代数的仕様記述，階層的設計法，順序機械型プログラム，在庫管理問題

1. まえがき

フォーマルアプローチによるプログラム開発法の中で代数的手法は有望視されている．代数的言語は，意味定義が厳密で，任意の抽象度のレベルを同じ言語で記述することができ，記述の正しさの証明を形式的に行える等の利点をもつ．しかし実用上の見地からその有用性を調べる研究はほとんどなされていない．筆者らのグループでは代数的言語ASLを定め[3]，記述支援系，検証支援系，実行系からなるASLプログラム開発システムを作り，代数的手法の実用性，有用性について調べている[4]．

我々は以下のことを行った．(1)抽象的順序機械型プログラムの階層的設計法について新たに提案した．(2)その設計法に基づいて，例題(在庫管理問題[1])に対して入出力関係のみ指定した最上位の記述から，5段階にわたってプログラムまで詳細化を行った．なお在庫管理問題は，プログラム設計技法の共通問題として提供された問題であり，今までにいくつかのグループが，それぞれの言語，手法で適用例を示している．(3)上記の過程を通じて，実際の記述において，一般的に生じるであろう問題点と，その解決方法を調べた．(4)実現プログラムの評価，他手法との比較を行った．これらについて，本論文で発表する．

(1)に関連して，我々のグループでは関数型プログラムによる階層的設計法を既に提案している[4]．

在庫管理のように次々と与えられる入力に対する逐次的処理の記述を考える．順序機械型のモデルでは，各時点での内部レジスタや出力に相当する(複数の)状

† 大阪大学基礎工学部情報工学科，豊中市
Faculty of Engineering Science, Osaka University, Toyonaka-shi, 560 Japan

態成分関数値が，その時点の入力による状態遷移に対して，遷移前の各状態成分関数値や入力に依存してどのように変わるか，というスタイルで記述する．一方関数型の記述では，今までの入力列を引数にして今の出力（あるいは今までの出力列）が何であるかというスタイルになる．従って，逐次的処理は，順序機械型のモデルで記述する方が関数型の記述よりは自然に記述できる．そこで今回，逐次的処理に対する順序機械型記述を用いた階層的設計法を提案する．その階層的設計法では各レベルで，そのレベルでの関数や処理等の要求記述が行われ，そのレベルで閉じた記述になっている．それにより，プログラムの正しさの証明が各段階ごと独立に行える，（実行可能なプログラムまで至らない）どのレベルまでの詳細化であっても設計記述として役立つ，等の利点がある．また，詳細化においては一つの状態遷移を下位レベルの状態遷移列に置き換える．状態遷移列を定義する方法として，順次実行，選択，末尾再帰などの形に制限しているので，プログラムの正しさの証明が各処理ごと独立に行え，また統一した証明の論法が使える等の利点がある．このような状態遷移の展開だけに制限しても実用上は問題がない．我々は，処理の要求記述を「これさえ満たしていればどのように実現してもよい」という立場で行う．そのために，（3）で述べるような拡張射影などの新しい概念を用いている．設計方法等を **2.** で述べる．

（2）に関して，言語，設計方法が定まっても，記述を具体的にどのように書くべきかについては，今までに研究例がなく，困難な点，解決方法等は明確でない．これらを調べるための例題として在庫管理問題を取り上げ，要求記述レベルから，実現プログラムまで順次詳細化を行った．

（3）に関して，一般に問題となるのは，（i）処理の要求記述の抜けをどのようにして防ぐか，（ii）処理の前提条件をどのように表現するか，（iii）最上位レベルで本来の要求をどのように記述するか，である．

（i）に対する解決方法として，記述スタイルを工夫した **(3.1)**†．

また（ii）に関して，各処理の前提条件を統一的に取り扱う方法を考案した **(3.1)**．普通，前提条件（外部から与えられる入力や，その処理前の内部データが満たすべき仮定）は，公理ごと独立に陽に記述する．しかし我々は前提条件をまとめて，一つの述語記号 valid で表現する．この述語 valid は，外部入力が満たすべき条件（および各処理 T が実行できる条件）のみ陽に書いて定義されており，個々の処理前に仮定する内部データが満たすべき条件は陽に書かない．この方法による記述を採用すると，（1）処理の満たすべき条件と，前提条件の記述が明示的に区別できる．（2）成立しない前提条件（内部情報に関する具体的な仮定）を書いて記述全体が意味のないものになる，という心配はない．（3）公理ごと必要な前提条件をいちいち考慮する必要はない．（4）処理の満たすべき内容を変更しても，通常は前提条件の記述部を変更しなくてもよい，等の利点が生ずる．

（iii）に対しては従来発表されているもの[1),(6),(7)]には我々のような抽象度の高い要求記述を行った例はない．（一般的に）このような抽象度の高い記述をどのように書くべきかも明確でない．そこで，以下に留意し，最上位の要求仕様記述を行った．

実現プログラムの実現の自由度を制約するような指定はすべきでない．例えば，内部でどのような情報を覚えるべきかや，どのような内部処理を用意し，それをどのような順で実行するか等は指定すべきでない．そのためには，入出力関係だけを指定すべきであるが，そのことを従来の枠組み（公理，および射影[3)]）で記述することはできない．そこで，標準的と思われる内部情報を用いて対象となる処理の要求を記述し，この記述から，必要な入出力関係だけを取り出す仕組みとして，新たに拡張射影を導入した．これを用いて第1レベルの記述を作成した．拡張射影は合同関係のみに基づいて定義されており，条件を満たす入出力関係（合同関係）の集合を指定する．この枠組により，従来表現できなかったこと（関数値が一意でなく，また相互に関連している関数群の中での特定の関数に関する要求）が記述でき，しかも合同関係のみを用いているので，従来の実現の正しさの概念等と自然に適合する **(3.3)**．

（4）に関して，以下のような中間レベル（第3レベル）を設け，他手法との比較を行った．

この中間レベルでは在庫管理において主要な処理である「出庫処理」（1枚の出庫依頼書に対し，在庫があるときにその依頼書に対応する出庫指示書を作成し，内部情報を更新する処理）に対する要求記述を行い，これを用いて在庫管理全体を記述した．なお，我々と同様に各レベルで意味定義の閉じた記述を行っているイ

† なお，一般に要求仕様に抜けがないことを調べるには，その仕様記述において，対象の処理である関数の値が定義されていること，記述された性質を用いてより上位の仕様に対する詳細化の正しさの証明できることなどを確認する必要があるが，かなり困難なことである．

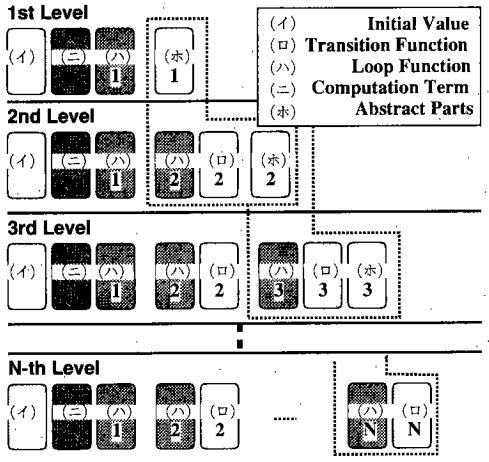

図1 ASMの階層的設計
Fig. 1 Hierarchical design of abstract sequential machine style program.

オタでは，これを最上位レベルの記述としている[1]．

また第5レベルの記述（プログラム）は，Cプログラムと比べそん色ない実行効率をもつことを確かめた(6.3)．

なお本論文では触れないが，我々はこの例プログラム（証明用の例題でない，実行効率を考慮に入れた実用的なプログラムとみなせる）に対して，プログラムの詳細化が正しいことを証明している[5],[8]．

2. 抽象的順序機械型プログラムの階層的設計の概略

在庫管理の記述およびプログラム作成には，ASLの部分クラスである抽象的順序機械型言語ASL/ASM[2]（以下ASMと呼ぶ）を用いる．

ASMでは抽象状態を表すデータ型を新たに導入し，状態初期化関数（初期状態に相当）と状態遷移関数の系列で抽象状態を表す．抽象状態は，各時点での内部情報（レジスタやファイルの内容）をすべて保持していると考え，この抽象状態からそれぞれの内部情報を取り出す関数として状態成分関数を導入する[2]．

状態遷移関数は内部情報（状態成分関数）を変化させる処理とみなすことができる．これらの処理を順次実行，繰返し，あるいは選択実行することによって，より複雑な処理（以下，「ループ関数」と呼ぶ）を作ることができる．

トップダウンによる階層的設計法を用いる．第1レベルでは，まず抽象レベルの記述に必要な処理（状態遷移関数）と内部情報（状態成分関数）を決定する．それらに対して，第1レベルの記述は，(イ)初期状態での各状態成分関数の値の指定（初期化），(ホ)処理の性質の記述と，(ハ)それらを用いたループ関数の記述，および(ニ)計算指定からなる（図1の1st Level）．第1レベルで設計者が考えている処理は抽象度が高いのが普通である．そこで(ホ)ではその処理に対応する状態遷移関数を適用する前後で各状態成分関数の値がどのような関係を満たすべきかを述語の形式で記述する．(ハ)では，いくつかの状態遷移を組み合わせてより複雑な処理を定義している．(ニ)では，初期状態から((ハ)で指定する）どんな一連の状態遷移を行った後のどの状態成分の値を計算結果として出力するかを指定する．

第2レベルでは，より具体的な（複数の）処理（状態遷移関数）や第1レベルの状態成分を計算するための新たな状態成分関数を導入し，それらを用いて第1レベルの性質記述の対象となった処理（これらは第2レベルのループ関数とみなす）を実現する．このため，第2レベルの記述は，第1レベルの(ホ)処理の性質の記述を第2レベルのより具体的な(ホ)処理の性質の記述と，(ハ)それらを用いたループ関数の記述で置き換えたものとなる（図1の2nd Level）．但し，性質の記述を書くまでもない簡単な処理については，(ホ)を，その処理を実行した後の各状態成分の値を具体的に定めた(ロ)処理の定義の記述で置き換えてもよい．また，新たに導入した状態成分関数については，今までに定義されている処理に対してその状態成分関数に関する記述を追加する必要がある．このように，性質記述の対象となる処理（抽象度の大きい状態遷移）を逐次（具体性の大きい状態遷移で）詳細化していく（図1の3rd Level）．最下位のレベルでは，処理の性質の記述は現れない（図1のN-th Level）．最下位まで詳細化した記述から(イ)〜(ニ)の記述を集めると実現プログラムとなる．

上述の実現法では上位レベルの各状態成分関数は下位レベルでも継承されると仮定しているが，下位レベルで新しい状態成分関数を導入し，それらを組み合わせて上位レベルの状態成分関数を実現してもよい．その場合，状態成分の対応関係や新たに導入した状態成分関数の記述などを追加する必要がある．

3. 階層的設計における記述方法

2.で順序機械型記述の階層的設計の概略について述べたが，ここで，その際に有効な記述スタイルやいくつかの概念についてまとめて述べる．これらの在庫管理問題への適用は4.で述べる．3.1で要求記述指針につ

いて簡単に述べた後，**3.2** において **1.** で述べた前提条件の取扱い方について説明し，**3.3** で仕様記述のための拡張射影の概念について述べる．

3.1 処理の要求記述指針

記述スタイルを工夫することにより，なるべく要求の抜けを生じさせないようにしたり，詳細化の正しさの証明をなるべく統一された手法で行えるようにすることは可能である．我々はいくつかの例題を通して得た経験から次のような記述方針をとっている．

処理の要求性質は処理の前後で(入力に対して)状態成分関数値がどのように変わるべきかという形で記述する．

そして(1)対象となる状態遷移を行った後の状態成分関数値がどうあるべきかを十分条件，必要条件の両方向について考え，記述する．(2)状態成分関数値の満たすべき性質が条件によって異なる場合はすべての条件について記述を行う．(3)自然語要求仕様に常識として陽に記述されていないことで要求仕様として必要な内容を決め，記述する．(1)，(2)の有用性は後述の出庫処理におけるコンテナリストの要求記述で現れる．この記述においては，指示書に現れた品目やコンテナがどうあるべきかに目を向けがちであるが，指示書に現れない品目やコンテナがどうであるべきかがすべての条件について考えることにより漏らさず記述できる．

3.2 前提条件の扱い方

1. で述べた理由により要求性質として，

$\mathrm{valid}(T(s,l))\mathrm{imply}p(F(T(s,l)),s,l)==\mathrm{TRUE}$

のように記述することを提案している．これは「状態 s で引数 l で処理 T を実行した後も valid が真である場合は，T に対する引数 s，l と，T 実行後の関数値 F の間には述語 p が成り立つ」ことを表している．

「s がそのレベルの記述の実行で実際に到達できる状態であり，かつ初期状態から現在に至るまでの各時点で，外部から与えられる入力が満たすべき条件を満足しているとき，valid が真になる」ように valid は(プログラムの外部入力とプログラムに現れる各処理について)公理式(1)，(2)のように定義することができる．

$$\mathrm{valid}(\mathrm{Init}(\cdots))==\mathrm{TRUE} \qquad (1)$$

$$\mathrm{valid}(s)\wedge\mathrm{Extern}_T(s,l)\wedge\mathrm{select}_T(s,l) ==\mathrm{valid}(T(s,l)) \qquad (2)$$

(s は状態を表す変数，l は処理 T の引数)

ここで，$\mathrm{Extern}_T(s,l)$ は，現状態 s に対して，処理 T の外部入力 l が満たすべき条件，$\mathrm{select}_T(s,l)$ は現状態 s で処理 T が選択される条件を表す．処理 T の引数 l がプログラムの外部から与えられるものでなければ，$\mathrm{Extern}_T(s,l)$ の項は省略される．Init はプログラムの初期状態である．各 T に対する select_T の内容はループ関数の記述の if 文の条件部から決められる．従って実際の記述では省略してよい．このように記述したテキストに対し，その詳細化を行ったり，詳細化の正しさを証明する際に，valid(s) が真である状態 s では内部データ(状態成分関数の値)の間に具体的にどのような関係が成立しているかを書く必要があるが，それらに関する議論はここでは省略する．

3.3 拡張射影を用いた要求記述

代数的言語ではその一つの記述(テキストとも言う)で，表現式集合およびその上の合同関係を指定する．我々の言語では表現式の構文および集合を文脈自由文法で指定し，変数付きの表現式(以降項と呼ぶ)の等式(公理と呼ぶ)で合同関係を指定する[3]．

抽象レベルの順序機械型記述では，その時点(抽象状態 s)での入力 a と処理 T の実行後の出力に相当する状態成分関数値 $f(T(s,a))$ との満たすべき関係を指定したい．一般に状態成分関数値 $f(T(s,a))$ は状態 s での他の状態成分関数値に依存する．そこで，T の実行後の各状態成分関数値が状態 s での状態成分関数値からどのように定まるかを記述しておき，そこから要求として指定したい関数 f の値のみ射影[3]を用いて取り出すことが考えられる．

例としてこのような形で，処理 $T_1(s,a)$，$T_2(s,b)$ について各状態成分関数値がどのように定まるかを記述したテキスト C を考える．変数 l を関数 f と同一のデータタイプの値のみ許す変数とする．次の二つの射影文

$f(T_1(s,a))==l$ **from** C　$f(T_2(s,b))==l$ **from** C

はテキスト C が表す合同関係から $f(T_1(s,a))$(あるいは $f(T_2(s,b))$)の変数 $s,a(s,b)$ に許される値を代入した表現式と合同関係にある値との等式集合(初期状態からの入力系列を含む各遷移列に対し，関数 f の値が何であるかを定義している)を取り出す†．よってこの射影文により，(1)関数 f 以外の状態成分関数は実現しなくてもよいこと，(2)要求として指定したい関数 f の値がそのような等式集合を満たすような T_1，T_2 でないといけない(そのような T_1，T_2 を実現せよ)，ということを指定したことになる．

しかしながら，普通は過剰指定(オーバスペック)に

† 公理も射影文も等式集合しか指定しない．これらの等式集合から定まる最小の合同関係をテキストの定める合同関係とする．

表1 性質記述すべき内容

性質1 未出庫依頼書情報の中から出庫できるものはすべて出庫すること
性質2 依頼書を選ぶ順番を陽に指定しないこと
性質3 作成された出庫指示書リスト中の任意の指示書がその出現位置で正しい記載内容をもつこと
記述1 その処理で作成される出庫指示書のリストを想定し，そのリストに対し，
a) 個々の指示書は 処理以前の未出庫依頼書リストに該当する依頼書があり，かつ
b) 該当する依頼書に対応した内容であること，
c) 出庫指示書のリストが，性質3を満たすこと．
記述2 処理後のコンテナリストは，出庫指示書のリストに書かれた分出庫されていること．
記述3 処理後の未出庫依頼書情報は，
a) 出庫指示書のリストに該当する依頼書の分を処理以前の未出庫依頼書情報から取り除いたものであること．
b) 処理後の在庫情報に対して出庫できるものがないこと．
記述1 b) の定義は以下の通り．
(S1) 指示書の番号が依頼書の番号に一致すること．
(S2) 指示書の送り先が依頼書の送り先に一致すること．
(S3) 指示書の品名が依頼書の品名に一致すること．
(S4) 指示書中のコンテナの記述は一度に限る (同じコンテナに関する記述が2回以上現れない) こと．
(S5) 指示書に書かれた本数の総数と依頼書の本数が一致すること．

指示書正当の内容
(S6a) 指示書に書かれた本数は正で，その本数はそれまでの仮想出庫量を差し引いても当該コンテナに存在すること．
(S6b) それまでの仮想出庫量に加え，指示書に書かれた本数だけ当該コンテナから出庫するとそのコンテナが空になるときそのときのみ，指示書に空きコンテナ指示が与えられること．

コンテナリストの性質
(C1) 出庫前のコンテナリストにあるコンテナで，かつ出庫指示書に現れかつ，空きコンテナ指示がないコンテナに関して，次の (a),(b) を満たすこと．
(a) そのコンテナ番号は，出庫後のコンテナリストにも，ただ一つあること．
(b) その番号のコンテナに関して，次の i,ii を満たすこと．
i 出庫品目は，出庫量だけ減少していること．
ii 出庫品目以外の品目に関して，出庫前後で品目及び数量に変化がないこと．
(C2) 出庫前のコンテナリストにあるコンテナで，かつ出庫指示書に現れないコンテナに関して，次の (a),(b) を満たすこと．
(a) そのコンテナ番号は，出庫後のコンテナリストにも，ただ一つあること．
(b) その番号のコンテナに関して出庫前後で品目及び数量に変化がないこと．
(C3) 出庫前のコンテナリストにあるコンテナで，かつ出庫指示書に空きコンテナ指示があれば，当該コンテナはコンテナリストから外されること．
(C4) 出庫前のコンテナリストにないコンテナは出庫後のコンテナリストに現れないこと．

ならぬよう関数値 $f(T_i(s,a))$ の値を一通りに定義せず，満たすべき条件のみ記述し，複数個の関数を許すべきである．このような記述を行ったテキスト C' では常に各状態成分関数について関数値が定まっているわけではないので従来の射影では等式集合を取り出せない．

そこで，テキスト C' から，要求として指定したい関数 f の C' を満たす関数表のみを取り出す概念として拡張射影を考える．拡張射影では関数 f の関数表に相当する等式集合のうち C' で許されるもの全体を表す．一般に C' の実現プログラム A は複数存在する．そのようなテキスト A を一つ定めればテキスト A から

$f(T_1(s,a))$==l **from** A　　$f(T_2(s,b))$==l **from** A

なる (従来の) 射影により等式集合が一つ取り出せる．これを $R(A,\{f\})$ で表す．拡張射影文

$f(T_1(s,a))$==l **Extndfrom** C'

$f(T_2(s,b))$==l **Extndfrom** C'

はそのような等式集合の集合 $\{R(A,\{f\}) \mid A$ は C' に対する f の実現プログラム$\}$ を表すと定義する†．これらの記述は，テキスト C' を満たすある実現と同じ f の値をとるような実現であれば，どのように T_1，T_2 を実現してもよいということを表す．

拡張射影を用いて本来必要な関係 (仕様) を記述する．拡張射影をもつテキスト U に対し，U の表す合同関係集合のうちの任意の一つの合同関係を満たすテキスト U' は U に対する正しい実現であると定義する．

4. 在庫管理問題への適用

3. で述べた概念の在庫管理問題への適用例を示す．

4.1 在庫管理記述のための基本データ型と基本関数

ここで在庫管理のプログラム，仕様記述に用いる基本データ型と基本関数について簡単に述べる．在庫管理の入力データである積荷表と出庫依頼書，出力データである出庫指示書 (以下単に依頼書，指示書とも記述する) のデータ型は，一般的な，文字列，整数および両方向リストなどの組として定義される．両方向リストはポインタを内部でもっており，抽象リストと呼ぶことにする．また，各抽象リストの要素をセルと呼ぶことにする．セルのデータ型が異なっていても共通のリスト演算子が用いられるものとする．抽象リストの基本演算として次のようなものがある．

新リスト，トップ，ボトム，前進，後退，削除＋，削除－，挿入，参照，参照_i，変更_i，先頭，後尾．

詳細は文献(5)，(8)を参照．

4.2 第1レベルの記述例 (被射影テキスト)

第1レベルで考えている状態遷移関数は依頼書入力に対する処理，積荷表入力に対する処理，保存終了処理の三つである．これらの処理の要求記述を行う際，例えばどのコンテナから処理していくか，あるいはどの未出庫依頼書から処理していくか，などの要求に自由度をもたせたい．そこでまず，これらの処理の要求記述を標準的と思われる状態成分関数を用いて行う．この記述を行ったテキストを $TL1$ とする．そして，テキスト $TL1$ からこれらの処理に関して関心ある出力 (出

† ASL では一般に条件付き公理を採用しているので，関数の引数に制約を付けるための条件付き射影および条件付き拡張射影を認めてもよい．

表2　積荷表入力に対する処理の要求記述（一部）(ホ)

```
define VALID :=
 'vaild(S 積荷表入力に対する処理(積荷表))' ;
define 処理後指示書リスト :=
 '差リスト(ファイルシステム(S),
  ファイルシステム(S 積荷表入力に対する処理(積荷表)))';
define 処理後未出庫依頼書リスト :=
'未出庫依頼書リスト(S 積荷表入力に対する処理(積荷表))' ;
I1: VALID imply( Memberof(id, 処理後指示書リスト)→
    Memberof(id, 未出庫依頼リスト(S)) and
  依頼書対応( 参照(id, 処理後未出庫依頼リスト),
              参照(id, 処理後指示書リスト))) == TRUE ;
I2: VALID imply 集合(処理後未出庫依頼リスト) =
    集合(未出庫依頼リスト(S)) -
    依頼書集合(処理後指示書リスト) == TRUE ;
I3: VALID imply(Memberof(id, 処理後未出庫依頼リスト)
 → not 出庫可能( 参照(id, 処理後未出庫依頼リスト),
コンテナリスト
     (S 積荷表入力に対する処理(積荷表))) == TRUE ;
                    ⋮
SS6: VALID imply 指示書リスト正当(コンテナリスト(S),
 積荷表, ボトム(処理後指示書リスト),CargoID) == TRUE ;
```

力用ファイル，情報保存用ファイル）だけを拡張射影で取り出す．拡張射影により，より実現の自由度の高い仕様を得ることができる．これに関しては**4.3**で述べる．ここでは拡張射影の対象となるテキストについて述べる．状態成分関数として，コンテナリスト(S)，未出庫依頼書リスト(S)，ファイルシステム(S)を用意する．それぞれ，状態 S での在庫情報，未出庫依頼書情報，ファイルの集合を表すのに使用する[8],[9]．コンテナリスト，未出庫依頼書リストのデータ型は抽象リストとする．

ここでは，特に積荷表入力に対する処理の要求記述について述べる．この処理は，(外部入力である)積荷表により，(以前の在庫情報と合わせて)出庫できるようになった未出庫依頼(出庫できず残っていた依頼書)をすべて出庫する処理である．表1中の性質1～3等の表現方法がポイントとなる．本論文では記述1～3等とする．

この記述に対応する各公理は前述の valid を用いて満たすべき性質がどうあるべきか，という形で記述されている(表2参照)．

なお「依頼書や，積荷表に現れる本数は正である」や，「番号は内部で記憶しているコンテナ番号や，依頼書番号と異なる」という，外部入力に関する条件は，**3.1**の述語 Extern の定義として与える．

公理Ｉ1が記述1a)に相当する．関数依頼書対応は記述1b)の内容に対応することを表す述語である．公理Ｉ2，Ｉ3，SS6がそれぞれ記述3a)，記述3b)，記述1c)に相当する．ここでは(関数ボトムでポインタを後尾にセットすることにより)出庫指示書リスト全体に対して関数指示書リスト正当が成り立つことを記述している．関数指示書リスト正当は別テキストで，リストの先頭からポインタまでのすべての位置の指示書に対して，述語指示書正当(表1の(S6a)，(S6b))が成立することと(再帰的に)定義されている．

関数依頼書入力に対する処理の要求記述は，例えば，在庫がある場合に表1の(S1)～(S6)等に相当すること等や，また在庫がない場合に未出庫依頼情報に引数の依頼書の情報が加わること，該当依頼書の出庫不可連絡が出力されること等を記述すればよい(省略)．

関数保存終了の満たすべき性質として，どのファイルにどの内部情報が保存されるべきかを記述する(省略)．

4.3　第1レベルの記述例(射影による抽出)

第1レベル記述テキスト本体では，上述の三つの状態遷移関数については，入力と処理後の(ファイルシステムの中で興味ある)ファイルの値に関する合同関係だけを拡張射影を用いて例えば以下のように取り出す記述を行う．

依頼書入力に対する処理，積荷表入力に対する処理，保存終了をそれぞれ T_1，T_2，T_3 と略記する．

出力ファイル(ファイルシステム(S T_1(irai))
  ==file **Extdfrom** TL1；
出力ファイル(ファイルシステム(S T_2(tumini))
  ==file **Extdfrom** TL1；
保存在庫ファイル(ファイルシステム(S T_3))
  ==file **Extdfrom** TL1；
保存依頼書ファイル(ファイルシステム S T_3))
  ==file **Extdfrom** TL1；

ここで S は“初期化(T_1(irai)|T_2(tumini))*”なる系列をとる変数である．

irai，tumini，file はそれぞれ依頼書入力に対する処理の引数である依頼書，積荷表入力に対する処理の引数である積荷表，ファイルの値を表す変数である†．保存終了 T_3 で終る処理系列に対しては保存すべき在庫情報，依頼書情報の内容を記述したファイルに興味があり，その他の系列については出力である出庫指示書や，出庫不可連絡を記載した出力ファイルに興味がある．テキスト $TL1$ の(T_1，T_2，T_3 に対する)正しい実現を決めればすべての遷移列に対するそれらの値が定まる．

† 変数に特定の表現式だけを代入するように制約を設けたい場合，本論文のように「irai は値だけが代入される変数である」というようにデータタイプ名で指定する．これだけでは対応できない場合例えばSについては条件付き拡張射影を用いて valid(S) を満たすようなSについてのみ拡張射影をとるというようにする．厳密には本例についてもこの方法を適用すべきであるが簡単のため省略する．

表3 出庫処理の要求記述(ホ)—一部—

```
define 前提条件 :='valid(S 出庫処理(依頼書))' ;
define 指示書 :='指示書(S 出庫処理(依頼書))' ;
  (* コンテナリストの性質 *)
define 処理後コンテナリスト :=
       'コンテナリスト(S 出庫処理(依頼書))' ;
define '搬出指示' :=
       '搬出指示コンテナ(指示書, CargoID)' ;
define '在庫減少量' :=
  '本数_在庫.番号.品名(コンテナリスト(S), CargoID,
                        品名(指示書)) -
 仮想数量( 品名(指示書),
   表_在庫.番号(処理後コンテナリスト, CargoID) )' ;
C1a: 前提条件 imply
( Memberof(CargoID , 指示書) and not 搬出指示
 and Memberof(CargoID, コンテナリスト(S)) →
 Uniq_id(処理後コンテナリスト) and
 Memberof(CargoID, 処理後コンテナリスト) ) == TRUE ;
C1bi: 前提条件 imply
( Memberof(CargoID, 指示書) and not 搬出指示 and
  and Memberof(CargoID, コンテナリスト(S)) →
  在庫減少量 =  本数_指示書.番号(指示書, CargoID) )  == TRUE ;
C1bii: 前提条件 imply
( Memberof(CargoID, 指示書) and not 搬出指示
 and Memberof(CargoID, コンテナリスト(S))
 and not 品名(指示書) = hin →
 コンテナ不変(hin,
     表_在庫.番号(処理後コンテナリスト, CargoID),
     表_在庫.番号(コンテナリスト(S), CargoID)) )== TRUE;
C2a: 前提条件 imply
( not Memberof(CargoID, 指示書)
  and Memberof(CargoID, コンテナリスト(S)) →
  Uniq_id(処理後コンテナリスト) and
  Memberof(CargoID, 処理後コンテナリスト) ) == TRUE ;
C2b: 前提条件 imply
( not Memberof(CargoID , 指示書)
  and Memberof(CargoID, コンテナリスト(S)) →
  コンテナ不変(hin,
      表_在庫.番号(処理後コンテナリスト, CargoID),
      表_在庫.番号(コンテナリスト(S), CargoID)) )== TRUE;
C3: 前提条件 imply
( Memberof(CargoID , 指示書) and  搬出指示
 and Memberof(CargoID, コンテナリスト(S)) →
 not Memberof(CargoID, 処理後コンテナリスト) ) == TRUE ;
C4: 前提条件 imply
( not Memberof(CargoID, コンテナリスト(S)) →
 not Memberof(CargoID, 処理後コンテナリスト) ) == TRUE ;
  (* valid に関する性質 *)
IN1: vaild(S) and 在庫有り(S, pr_3(依頼書), pr_4(依頼書))
  imply valid(S 出庫処理(依頼書)) == TRUE ;
                    ⋮
A → B は, not A or B の略記
```

T_1, T_2, T_3 はそのような合同関係のうちいずれと一致するように実現してもよいということを表している.

これらの拡張射影文と初期化(イ), および在庫管理全体の記述(ハ), 計算指定(ニ)で第1レベルの記述となる.

(ハ)では入力に応じて三つの状態遷移関数のいずれかを選択し, その後再び入力を得て繰り返す.

(ニ)においては, 入力系列に対して興味あるファイルの値に関する合同関係のみを指定する. 初期化, 計算指定, 在庫管理全体の記述の実際は省略[8].

4.4 第2レベルの記述例

第2レベルでは, 積荷表入力に対する処理を, 積荷表追加処理と, 全未出庫依頼書に対する処理を用いて詳細化した. その他の記述は, テキスト TL1 を継承する†.

4.5 第3レベルの記述例

第3レベルでは, まず, 状態成分関数として指示書(S), 在庫あり(S,hon,hin)を追加した. 指示書(S)は, 出庫指示書の内容を逐次求めていくために用いる状態成分関数である. 在庫あり(S,hon,hin)は, 出庫依頼書に記載されている本数 hon, 品名 hin に対して在庫があるかどうかを判定する述語関数である. 在庫がある場合, その依頼書に対応する(その商品をどのコンテナから何個ずつ出庫するか, また, その出庫によりそのコンテナが空になるかどうか等を記載した)出庫指示書を作成し内部情報等を更新する出庫処理の満たすべき性質(ホ)を記述する. また出庫処理以外の簡単な処理(作成された指示書を出力ファイルに書き出す指示書出力処理等)の定義(ロ), と併せて第2レベルの全未出庫依頼書に対する処理, 依頼書入力に対する処理の詳細化を行う(ハ)(これらの記述[5],[8]は省略).

4.6 状態遷移「出庫処理」の満たすべき性質

出庫処理の満たすべき性質を次のとおり定める.

指示書について表1における(S 1)〜(S 5)に加え, 表1の(S 6 a), (S 6 b)において, 仮想出庫量を0としたもの((S 6')とする)を記述する.

コンテナリストに関しては, 個々のコンテナに着目して処理後のコンテナリストに, (1)存在するかしないか, (2)存在するときは, 内容品はどうあるべきか, を記述すればよい. これを **3.1** の方針に注意してまとめると表1の(C 1)〜(C 4)となる. なお, (C 1), (C 2)は処理後にコンテナが存在する場合で, (C 3), (C 4)は存在しない場合である. これらの記述をほぼそのまま公理の形で記述する. 記述例については表3参照††.

状態遷移関数在庫ありの満たすべき性質として,「在庫ありが真であるのは, 状態 S でのコンテナリストに, 依頼書の品名が依頼書の本数以上あるときかつそのときに限る」ことを記述しておく.

† もちろんその他の実現方法もあり得るが, 本論文の記述では詳細化の正しさが議論しやすい.

†† イオタでは内容が0のコンテナを加えてもよいという解釈が可能でありあまり好ましくない. またこれらの関係を間接的に述べているため, わかりにくくまた間違いが生じやすい.

5. 第4，第5レベルの記述例

5.1 第4レベルの記述例——「在庫品名数量表」の導入と状態成分関数「在庫あり」の実現——

状態成分関数在庫ありをこのレベルで新たに導入した状態成分関数在庫品名数量表(S)を用いて実現する．在庫品名数量表は，品名から直ちに(状態 S における)在庫数量がわかるテーブルである．コンテナリストを用いて在庫ありを記述しているが，品名数量表を参照するように在庫ありを実現することにより，処理の高速化が行える(記述例は省略)．

第4レベルは，第3レベルの記述のうち在庫ありに関する性質の記述を取り除き，在庫ありの実現，新しい状態成分関数のための追加公理，在庫品名数量表の要求性質等の記述を加えたものである．

新しい状態成分関数在庫品名数量表の出庫処理，保存終了処理に対する要求性質として，例えば出庫処理後の在庫品名数量表は指示書に記載されている分だけ減っていることなどを記述する(記述例[8]は省略)．

5.2 第5レベルの記述例

ここで保存終了処理と出庫処理を詳細化する．

保存終了処理を，リストや抽象テーブルをファイルに書き込む基本関数を用いて実現する．

出庫処理を，リストのポインタの移動，リストのセルの削除，リストのセル内の本数の減少等を行う基本関数を用いて実現する(記述例[8]は省略)．

ここでは，コンテナリストの先頭(古い在庫)から順に調べ，コンテナリストをたどるという方針を採用する．処理の高速化のため，必要量出庫すればその時点で処理を終了するように記述している†．ループの脱出条件のためのパラメータ hon は出庫依頼本数に初期設定されており，現時点の着目コンテナの該当品目の数量分を毎回の繰返しで減少させる．このとき，高速化のためコンテナリストを新たに作り直すことを避けている††．以上のように高速化の工夫をしているが，それにもかかわらずこの実現が，出庫処理の要求性質を満たしていることの証明が行えている(一部)[5],[8]．

6. ASLによる在庫管理プログラム開発

6.1 プログラムのサイズ

ASLテキスト(各レベルの(イ)～(ホ))の各サイズを表4に示す．プログラムは，各レベルの(イ)～(ニ)を合わせたもの(公理は38個，140行)である．出庫処理の要求性質等の記述も含め記述のサイズは大きくはない．

表4 ASLテキストのサイズ

文法部		書換え規則 200個 300行			
レベル	イ	ロ	ハ	ニ	ホ
1	3/3		1/11	4/4	45/200
2		1/2	1/2		10/60
3	1/2	6/18	4/20		30/140
4	1/2	1/2	1/2		3/8
5		10/33	4/36		

表の要素はそれぞれ，公理数/行数 を表している．
状態成分関数の値が変わらない状態遷移に関する公理，及び上位から継承される公理は省略し，数えていない．

表5 在庫管理プログラムの実行時間

場合	依頼書数	積荷表数	出庫指示書数	延べ処理時間†
1	50枚	20枚	約50枚	約1.7秒
2	100枚	40枚	約100枚	約4.3秒
3	200枚	100枚	約300枚	約18秒

†Sun SPARC上で計測

基本関数として比較的高度な関数(もちろんその実現が問題の本質にかかわらない汎用の関数)を使用していることが理由の一つであるが，これにより，実現プログラムの正しさの証明を行うときに，基本関数の性質の証明に煩わされることなしに，本質的な部分の議論のみを行える利点も生じる．

6.2 コンパイル時間

この在庫管理プログラムの全体(上述のテキスト，Cで記述した基本関数，ASLで記述した基本関数(公理は80個，250行))のコンパイル(ASLのマクロ展開や語い解析，構文解析，Cへの言語変換(変換されたCコードは全体で約900行程度)，Cのコンパイル)に要した時間は，Sun SPARC ELCを使用して全体で約100秒ほどである．コンパイルするCのサイズは全体で約1,700行であった†††．

6.3 実行時間

表5にテストデータによる目的プログラムの実行時間を記す．在庫管理プログラムを起動し，コンテナ，未出庫依頼をあらかじめいくらかもたせた上(場合1はそれぞれ50個，50枚，場合2はそれぞれ50個，100枚，場合3はそれぞれ100個，200枚)で，出庫依頼

† イオタでは，ループの脱出条件をリストの最後かどうかで判定しており，出庫すべき本数を出し終えた後では無駄な繰返しを行っている．

†† イオタの記述では，リストを新たに作り直しており，リストの大域化による高速化は望めない．

††† ASLの目的語としては，ライブラリーとの結合の容易性，移植性を考慮に入れて，C言語にしている．

書，積荷表を複数枚(70～300枚(例えば場合1で出庫指示書が約50枚出力されるよう調節した))ランダムに与えたときの処理時間の合計である．

なお同じアルゴリズムで同じ基本関数を用いてすべてCで記述した場合，テキストサイズは計1,800行であり(基本関数を除いた処理に関する部分だけのサイズでは約200行)，実行時間は場合1，2，3でそれぞれ約1.0秒，2.8秒，10秒であった．実行時間の比はたかだか2倍程度であることが確かめられた†．

比較のため，第4レベルの工夫を行わず述語関数在庫ありを上位レベルの要求性質(定義式)のとおり，コンテナリストをたどりながら調べるように実現したときの実行時間は，場合1，2，3でそれぞれ約3秒，12秒，100秒であった．

7. 他手法との比較

実用性を考慮に入れてプログラム設計開発方法を提案する場合，言語，設計方法，詳細化の正しさ等の定義の明確さ，記述の自然さ，例題の適用結果，処理系の実行効率，プログラムの正しさの検証法(証明が計算機の支援のもとで行えること)等が大事であると考える．他手法との比較を，特に設計法，適用結果，プログラムの実行効率に焦点を絞って行う．

設計法に関しては，トップダウンによる階層的設計を提案しているものは多かった．しかしながらそのレベルで意味定義がきちんと閉じている階層化を行っているもの(例えばイオタ，HIPS[7])は少なかった．

実行環境システムに関しては，言及されているもの(例えば，Stella[1]，イオタ，Valid[1])はいくつかあった．これらのうち関数型言語の処理系はインタプリタであり，実行効率はあまり良くないと予想される．他の言語の処理系についても具体的な実行効率を示しているものはなかった．Cの実行時間の2倍程度という本論文の実行結果はこの種のフォーマルアプローチで作成されたプログラムの実行効率としては十分速いものと思われる．

我々はプログラムの検証も行なっているが[5],[8]，プログラムの検証可能性に関しては処理系がサポートされているのは，イオタだけであった．しかし，いくつかの脚注で触れたように，イオタでも我々の観点からでは好ましくない点があり，プログラムの正しさの証明も実例については，触れていなかった．また，我々の第1レベルに相当する本来の要求記述を行った例も見られなかった．特に仕様の「抜け」に関しては我々の記述は十分考慮したつもりである(**3.1** 参照)．

† なお，現バージョンの変換系では補助関数等の関数型記述のtail recursionの解消を行っていないため実行時間はやや遅い．この点を改良すれば更に良くなるものと思われる．

8. むすび

ASL/ASMの階層的設計法を提案し，その手法を用いて，在庫管理の記述を自然に行うことができた．我々は出庫処理の要求仕様のうち指示書に関するもの(**4.6** 中の(S 1)～(S 6′))や，第2レベルの要求記述(一部)について，本論文のプログラムがそれらを満たしていることを検証支援系を用いて証明した[5],[8]．これらについては，内容を整理した上で別途発表したい．実行速度は代数的言語としては十分実用に耐えることをこの例題でも確かめることができた．

文　献

(1) 二村良彦，雨宮真人，山崎利治，淵　一博："新しいプログラミングパラダイムによる共通問題の設計"，情報処理，**26**, 5, pp. 458-520 (1985-05).

(2) 大瀧雅弘，杉山裕二，谷口健一："代数的言語ASLにおける抽象的順序機械型プログラムとその処理系"，信学論(D-I)，**J73-D-I**, 12, pp. 971-978 (1990-12).

(3) 嵩　忠雄，谷口健一，杉山裕二，関　浩之："代数的言語ASL/*—意味定義を中心に"，信学論(D)，**J69-D**, 7, pp. 1066-1075 (1986-07).

(4) 東野輝夫，関　浩之，谷口健一："代数的仕様から関数型プログラムの導出とその実行"，情報処理，**29**, 8, pp. 881-896 (1989-08).

(5) 岡野浩三，北道淳司，東野輝夫，谷口健一："代数的言語ASLで記述した在庫管理プログラムとその正しさの証明"，信学技報，**SS90**-29 (1990-12).

(6) 山崎利治："共通問題によるプログラム設計技法解説"，他，情報処理，**25**, 9, pp. 934-962 (1984-09).

(7) 山崎利治："共通問題によるプログラム設計技法解説(その2)"，他，情報処理，**25**, 11, pp. 1219-1268 (1984-11).

(8) 岡野浩三："代数的手法による在庫管理プログラムの階層的設計とそのプログラムの正しさの証明"，平成3年度大阪大学大学院基礎工学研究科修士学位論文 (1992-03).

(9) 岡野浩三，東野輝夫，谷口健一："代数的言語ASLを用いた酒屋在庫管理の要求仕様記述"，平4情処春季全大，pp. 5-327-328 (1992-03).

(平成4年6月3日受付，5年1月13日再受付)

岡野　浩三

平2阪大・基礎工・情報卒．平5同大大学院博士後期課程中退．同年同大情報工学科助手，現在に至る．代数的手法によるプログラム開発，分散システムなどに興味をもつ．情報処理学会会員．

北道　淳司

昭63阪大・基礎工・情報卒．平3同大大学院後期課程中退．同年同情報工学科助手，現在に至る．代数的手法を用いた順序回路の仕様記述および設計に関する研究に従事．情報処理学会会員．

東野　輝夫

昭54阪大・基礎工・情報卒．昭59同大大学院博士課程了．同年同大助手，現在，同情報工学科助教授．工博．昭63～平2本会SS研幹事．平2モントリオール大客員研究員．ソフトウェア開発におけるフォーマルアプローチ，分散システム，通信プロトコルなどに関する研究に従事．情報処理学会，ACM各会員．

谷口　健一

昭40阪大・工・電子卒．昭45同大大学院博士課程了．同年同大・基礎工・助手，現在，同情報工学科教授．工博．この間，オートマトンと言語理論，計算の複雑さ，プログラム設計開発の代数的手法および支援システム，ハードウェアの仕様記述と詳細化などに関する研究に従事．