ソフトウェア工学 46－3
（1986. 2. 6）

研究室内図書購入手続きの代数的仕様記述とその実現

田中哲雄　伊藤実　松浦敏雄　谷口健一
（大阪大学　基礎工学部）

1．まえがき

本稿では、筆者らの研究室における図書購入業務の代数的仕様記述、およびその実現について述べる。

まず、研究室の図書購入業務を分析し、本学の図書館で行われる業務やこの業務に関連する書店の業務も含めてモデル化した。このモデル化においては、業務の満たすべき条件のみに注目した。このモデルを代数的仕様記述言語ASL[1],[2]により記述した。ここで記述した仕様を計算機を用いて実現するために、計算機で処理する部分を指定し、それのより詳細化した仕様をASLで記述した。このとき、計算機で処理しない部分とのインタフェース及び手続きの実行順序を制御するスケジューラもASLで記述した。最後に、その詳細化した記述をASLの部分言語である関数型言語ASL/F[3]のプログラムに変換した。このプログラムはVAX11上で稼働する。

2．図書購入業務の概要

この章では「研究室の図書購入業務」を分析し、その結果得られた業務の本質的な部分の概要を述べる。ここでは計算機を用いてどう実現するかについては述べない。ここでいう図書購入業務はそれに関連する本学の図書館の業務や、書店の業務も含み、次の手続きからなる。

希望？：　購入を希望する図書（希望図書）の希望伝票を作成する。希望伝票には「書名」、「著者名」、「出版社」という項目がある。この手続きは、研究室の人間が図書の購入を希望したとき実行される。

必要？：希望図書が"既に、研究室にあるかどうか"などの判断情報によって必要か不要かをチェックする。この手続きは、希望伝票があるとき実行される。

発注？：　必要と判断された図書について、発注票とその控を作り、発注票は書店へ送る。発注票には「書名」、「著者名」、「出版社」、「発注日」、「書店」という項目がある。「書名」、「著者名」、「出版社」の項目には必要と判断された図書の希望伝票の各項目の値をそれぞれ書く。「発注日」の項目にはその日の日付を書く。「書店」の項目には発注する書店名を書く。控は発注票のコピーである。この手続きは、必要と判断された図書の希望伝票があるとき実行される。

納品／返答（書店の手続き）：　書店は、研究室によって発注された図書について、絶版でなければ、納品する。このとき納品は「納品書」、「納品書控」、「本」からなり、納品書には「書名」、「著者名」、「出版社」、「価格」、「書店」、「受入日」という項目がある。納品書控は納品書のコピーである。本には「書名」、「著者名」、「出版社」、「価格」という項目がある。「書名」、「著者名」、「出版社」、「書店」には発注票の各項目の内容を記入する。「価格」には図書の価格を記入する。また「受入日」にはその日の日付を書く。絶版または品切れならばそのことを研究室に知らせる。その通知には「書名」、「著者名」、「出版社」、「書店」、「絶版」、「品切れ」という項目がある。「書名」、「著者名」、「出版社」、「書店」には発注票の各項目を書く。「絶版／品切れ」には絶版か、品切れかを書く。この手続きは、研究室から発注票が届いたとき実行する。また発注されていない図書についても、書店の判断で研究室に必要と思われる図書は納品する。このときの納品は「納品書」、「納品書控」、「本」からなり、それらの項目は研究室によって発注された図書を納品する場合と同じである。「書名」、「著者名」、「出版社」、「価格」には納品する本の書名、著者名、出版社、価格を書く。「書店」には書店名を、「受入日」にはその日の日付を書く。

受入：　書店によって納品された図書について、納品書の控にサインして書店に渡す。絶版または品切れの通知がきたときには、その通知を保存する。この手続きは、書店から納品または絶版／品切れ通知が届いたとき実行される。

購入？：　納品された書籍が必要かどうかを判断情報によりチェックする。不要と判断された書籍は書店に返却する。この手続きは、納品された書籍があるとき実行される。

カード作成：　必要と判断された図書について、貸出カード、購入控、購入依頼書作成願（研究室が図書館に提出する購入依頼書を書店が研究室の代わりに作成するための書類）を作る。それらは、「書名」、「著者名」、「出版社」、「価格」という項目からなり、各項目には必要と判断された図書のそれぞれの項目を記入する。この手続きは、購入することが決った図書があるとき実行される。

依頼書作成（書店の手続き）：　書店は、研究室からの購入依頼書作成願にしたがって、購入依頼書を作成する。その項目は購入依頼書作成願と同じである。この手続きは、研究室から購入依頼書作成願が届いたとき実行される。

捺印：　書店からの購入依頼書に捺印し図書館に送る。この手続きは、書店から購入依頼書が届いたとき実行される。

整理（図書館の手続き）：　図書館は、研究室から購入依頼書と書籍（未整理本）が届くと、その書籍に「請求番号」、「登録番号」を付ける。この手続きは、研究室から購入依頼書が届いたとき実行される。

登録：　図書館で請求番号を付けられた図書（整理済図書）

について登録リストを作成する。登録リストには「書名」、「著者名」、「書店」、「価格」、「登録番号」、「請求番号」という項目がある。各項目には整理済図書のそれぞれの項目を記入する。この手続きは、図書館から請求番号を付けた図書が届いたとき実行される。

督促：　書店に対して督促状を出す。督促状には「書名」、「著者名」、「出版社」、「発注日」、「督促日」、「書店」という項目がある。「書名」、「著者名」、「出版社」、「発注日」の項目には、該当する図書の発注控のそれぞれの項目を記入する。「督促日」の項目には、その日の日付を記入する。発注控に督促日を記入する。この手続きは、発注または督促してから定められた期間経過しても図書が納品されないとき実行される。

3．抽象レベルの代数的記述

この章では、2章で述べた「研究室の図書購入業務」をASLで代数的に記述する。3．1で実際の業務をどのように抽象化するかについて述べ、3．2でその抽象化された業務を代数的に記述する方針を述べる。

3．1　業務の抽象化

研究室の図書購入業務を、本学の図書館で行われる業務やこの業務に関連する書店の業務も含めて、次のように抽象化する。（図3．1参照）

図書購入業務における個々の図書に関する書類や書籍（「希望伝票」、「発注票」、「発注控」、「本」、「納品書」、「納品書控」、「絶版通知」、「カード購入控」、「購入依頼書作成願」、「購入依頼書」、「目録」、「督促状」）の組をオブジェクトと呼ぶことにする。その書類や書籍は「書名」、「著者名」、「出版社」、「書店」、「発注日」、「督促日」、「価格」、「絶版」、「品切れ」、「受入日」、「サイン」、「捺印」、「登録番号」、「請求番号」、「日付」という項目の組である。すなわちオブジェクトObjは、

$$Obj = \langle s_1、\cdots、s_n \rangle$$
$$s_i = \langle t_1、\cdots、t_k \rangle$$

という形をしている。ここでs_iはオブジェクトの書類や書籍を表し、t_jはs_i項目を表す。オブジェクトの各書類や書籍の項目を表3.1に示す。各書類は表3.1の＊印のついた項目を持つ。

新しく書類を作成したり、書類に捺印したりする業務をすべて手続きと呼ぶ。手続きはオブジェクトの内容を変えるものである。オブジェクトにはその内容によって、購入の必要があるか、発注

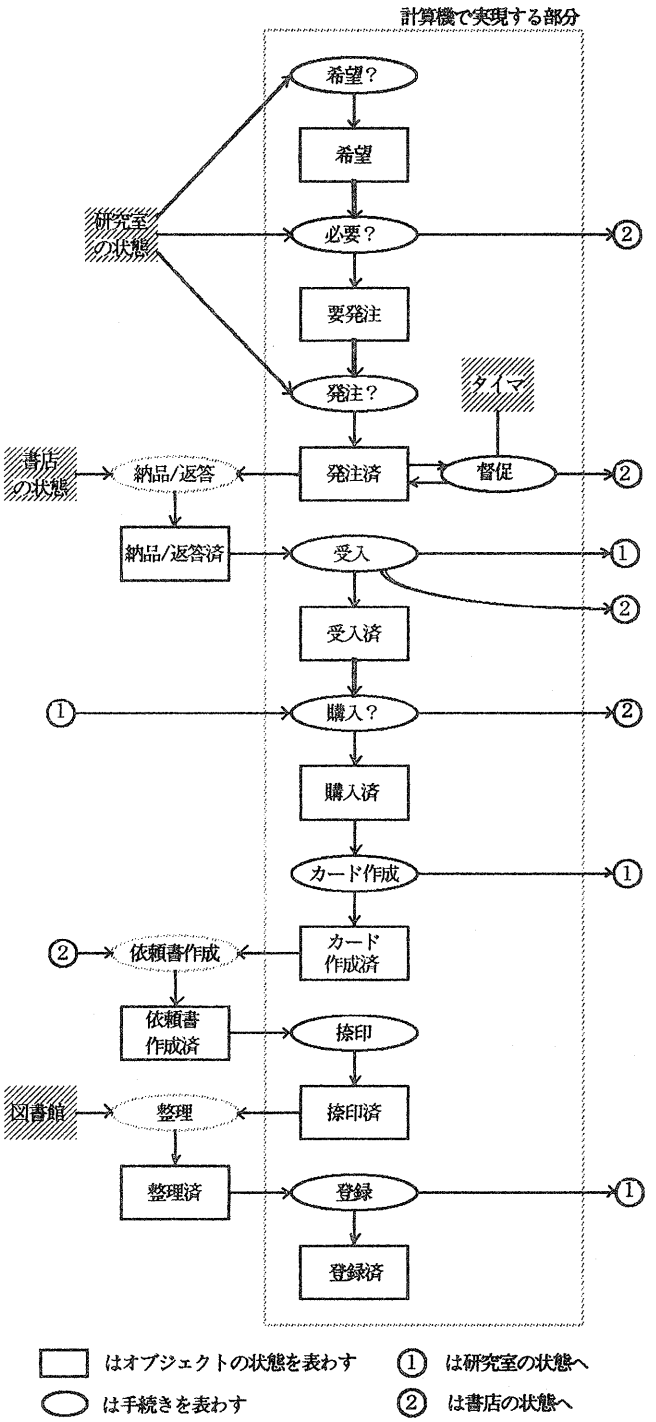

図 3．1　抽象レベルの記述の概念図

済か、登録済の図書か、などの状態がある。

手続きをP_1、…、P_nとする。例えば「手続きPjは手続きPiが終ったあとのオブジェクトに対してのみ意味がある」という関係が、手続きPiと手続きPjとの間にあるとき、あるオブジェクトに注目すれば手続きPiは手続きPjより先に実行されなければならない。すなわち、手続きPの実行が意味があるか否かの条件（実行可能条件）が定められている。例えば、発注手続きの実行可能条件は、その書籍に対するオブジェクトが発注許可の下りている状態にあるということである。

表3．1　Objectの各書類の項目

	書名	著者名	出版社	書店	発注日	督促日	価格	絶版品切れ	受入日	サイン	捺印	登録番号	請求番号	日付
希望伝票	*	*	*											
発注票	*	*	*	*	*									
発注控	*	*	*	*	*	*								
絶版通知	*	*	*	*				*						
納品書控	*	*	*	*			*		*	*				
納品書	*	*	*	*			*		*					
購入控	*	*	*				*							
カード	*	*	*				*							
依頼書作成願	*	*	*				*							
依頼書	*	*	*				*				*			
書籍	*	*	*				*					*	*	
目録	*	*	*				*					*	*	*
督促状	*	*	*	*	*	*								

3．2　代数的記述

3．1で抽象化した図書購入業務を代数的に記述する。その記述の1部を表3.2に示す。

［オブジェクトの状態］　手続きによってオブジェクトがどのような状態になるか（例えば発注済か、購入の必要があるか、登録済の図書か、など）を述語を用いて表す。オブジェクトの状態を表す述語には次のものがある。

希望　　：希望伝票がある。
要発注　　：購入する必要があると認められた図書の希望伝票がある。
発注済　　：発注票がある。
納品／返答　：納品または絶版／品切れ通知がある。
受入済　　：絶版／品切れでなく、サイン済の納品書控がある。
購入済　　：購入することが決った書籍がある。
カード作成済：貸出カードがある。
依頼書作成済：購入依頼書がある。
捺印済　　：購入依頼書に捺印済である。
整理済　　：請求番号付の書籍がある。
登録済　　：図書が登録済である。

表3．2　抽象レベルの代数的記述

```
SPEC  図書購入業務
  【  Obj      =  〈希望伝票，発注票，発注控，…〉
      Subobj   =  〈書名，著者名，出版社，…〉 】
CON 納品／返答:   Obj，S書店    →  Obj;
    購入？    :   Obj          →  Obj;
        :
OP  FOR P in   {発注済，納品／返答済，受入済…}
    P       :  Obj    →  bool
    For P1  in   {書名，著者名，出版社，書店，…}
    P1      :  Subobj →  項目;
    For P2  in   {希望伝票，発注票，発注控，…}
    P2      :  Obj    →  Subobj;
    invalid :  Obj    →  bool;
#   invalid関数
i1: invalid (納品／返答 (Obj，S書店) )
    == not 見計らい (S書店)  and not 発注済 (Obj) ;
i2: invalid (購入？ (Obj) )
    == not  受入済 (Obj) ;
      :
#   手続きによるオブジェクトの状態の変化
P1: 納品／返答済 (納品／返答 (Obj，S書店) )
    == true;
P2: 購入済 (購入？ (Obj，S研究室) )
    == need (Obj，S研究室) ;
      :
#   手続きによるオブジェクトの項目の変化
o1: 納品／返答 (Obj，S書店)
     == if  見計らい (S書店)   then
           納品1 (Obj，S書店)
        else if 絶版 (Obj)  or 品切れ (Obj)  then
           返答 (Obj)
        else
           納品2 (Obj，S書店) ;
o2: 書名   (本 (納品2 (Obj，S書店) ) )
    == 書名   (発注票 (Obj) )
o3: 著者名 (本 (納品2 (Obj，S書店) ) )
    == 著者名 (発注票 (Obj) )
o4:  出版社 (本 (納品2 (Obj，S書店) ) )
    == 出版社 (発注票 (Obj) )
o5: 価格   (本 (納品2 (Obj，S書店) ) )
    == get_価格 (S書店，Obj)
       :
```

本稿で記述した図書購入業務では、どのオブジェクトも高々1つの述語を成り立たせる。

［オブジェクトの項目］　各手続きにおいて、オブジェクトの各項目の値がオブジェクトのどの項目から、どのような演算により作られるかを記述することにより、手続きによるオブジェクトの変化を記述する。ここで手続きPによって変化したオブジェクトをP(Obj)で表すことにする。あるオブジェクトObjの書類s_iを取り出す関数s_i(Obj)、書類の各項目t」

の値を示す関数$t_j(s_i(Obj))$を用いて各手続きP(Obj)は次のような形で表されるとする。

$$t_j(s_i(P(Obj))) == f(t_1(s_1(Obj)),\cdots, t_k(s_n(Obj)));\quad \cdots(1)$$

または、

$$t_j(s_i(P(Obj))) == g(Sout)\ ;\quad \cdots(2)$$

(ただし、$1 \leqq i \leqq n, 1 \leqq j \leqq k$)

ここでSoutは研究室、書店、または図書館の状態であり、fやgは引数から項目の値を計算する関数（例えば整数や文字列の基本関数とか、公理で定義する関数の複合関数）である。本稿で記述した業務におけるfは、Pに対してあるs_mがあってs_mの項目を取り出す関数である。従って式(1)は

$$t_j(s_i(P(Obj))) == t_j(s_m(Obj))$$

と表すことができる。

例えば、納品という手続きにより、発注票と書店の状態から納品書が作られるとき、納品書の書名、価格は次のように定義される。

書名(納品書(納品(Obj))) == 書名(発注票(Obj))

価格(納品書(納品(Obj))) == get_価格(S書店))

ここでS書店は書店の状態である。

[手続きの実行]　手続きの実行が不可能な条件をinvalidという述語によって表す。invalid(P(Obj))が真ならば、Pは実行不可能、偽ならば実行可能である。例えばオブジェクトが受入済という状態になければ、購入？という手続きは実行出来ない。従ってinvalid関数は次のように書ける。

invalid(購入?(Obj)) == not受入済(Obj)

4．詳細化の方針

4．1　ボックスとレコード

ここで示した図書購入業務の抽象レベルの仕様記述では、個々のオブジェクトの各状態でオブジェクトの１つの書類成分の値だけが公理で定義されている。この書類成分をレコードと呼ぶと、１つのオブジェクトは、１つのレコードで表わすことができ、各手続きはオブジェクトの１つのレコードを引数として同じオブジェクトの他のレコードを出力するものとみなせる。

また、手続きの計算機上での実現を考えた場合、１つの手続きの引数は同じ形で同じ場所に格納した方が効率が良い。そこで、同じレコードの集まりをボックスと呼ぶと、手続きは、ボックスから１つのレコードを取出し、それをもとに同じオブジェクトの他のレコードを作り、そのレコードが入るべきボックスに移すものとみなせる。このように考えた場合の概念図は、図3.1中の長方形で囲まれた部分をボックスとみなしたものと等しい。このとき、例えば、発注という手続きは、要発注というボックスから１つのレコードを取り出し、発注済というボックスにレコードを入れると考えて良い。

4．2　計算機上で実現する部分およびそれとの入出力の指定

抽象レベルで記述した仕様のうち、研究室での業務を計算機を用いて実現するために、抽象レベルで記述された各手続きを、計算機で処理するものとそれ以外とに分けなければならない。以下に、図3.1に基づいて計算機で処理する部分とそれ以外の部分のインタフェースをどのように取り扱うかを説明する。（図4.1参照）

（１）書店や図書館の状態を引数とする手続き

納品／返答、依頼書作成、および整理の各手続きは、書店および図書館の状態によって決まる値を引数とするので、研究室の計算機では処理できない。従って、これらの手続きは書店および図書館で行うものとする。以下、これらの手続きおよびそれらとのインタフェースをどのように取り扱うかを説明する。

納品／返答手続きを書店が行えるように、発注手続きは発注リスト（図4.1参照）を計算機から出力する。発注リストは、発注済ボックス内のレコードと同じ項目からなるレコードである。書店は、この発注リストをもとに納品／返答手続きに対応する業務を行い、その出力である納品／返答リストを研究室に届けるものとする。このリストに基づいて、研究室のオペレータが端末から納品／返答リストの各項目を入力する。このために、端末からデータを入力する手続きが必要になる。この手続きは、抽象レベルでは記述されていなかったもので、このような手続きを一般に入力手続きと呼ぶ。入力手続きは端末から文字列を読込む関数と、端末に文字列を書出す関数を用いて記述されている。

納品／返答入力手続きは、納品／返答リストを計算機内に取り込むための手続きであり、この手続きの出力は、抽象レベルでの書店の手続き（納品／返答）の出力となっている。

依頼書作成手続きおよび整理手続きも同様に考えることができる。

（２）研究室の状態を引数とする手続き

希望？、必要？、発注？、および購入？の各手続きは、研究室の状態によって決まる値を引数とするので、計算機のみでは処理できない。従って、このような値を計算機内に取り込む手段が必要であり、前述の入力手続きと同じように端末を介して、この値を入力する。例えば、必要？手続きにおけるこの値は、その書籍を購入するか否かの研究室の判断に相当するものである。発注？および購入？の手続きの場合も同様である。希望？手続きの場合のこの値は、新しいオブジェクトを作り出すために用いられるが、入力方法等は同様である。

（３）書店や研究室の状態を変える手続き

必要？、督促、受入、購入？、カード作成および登録の各手続きは、書店や研究室の状態を変える。書店や研究室の状態は、計算機で取り扱えないので、このような手続きは、計

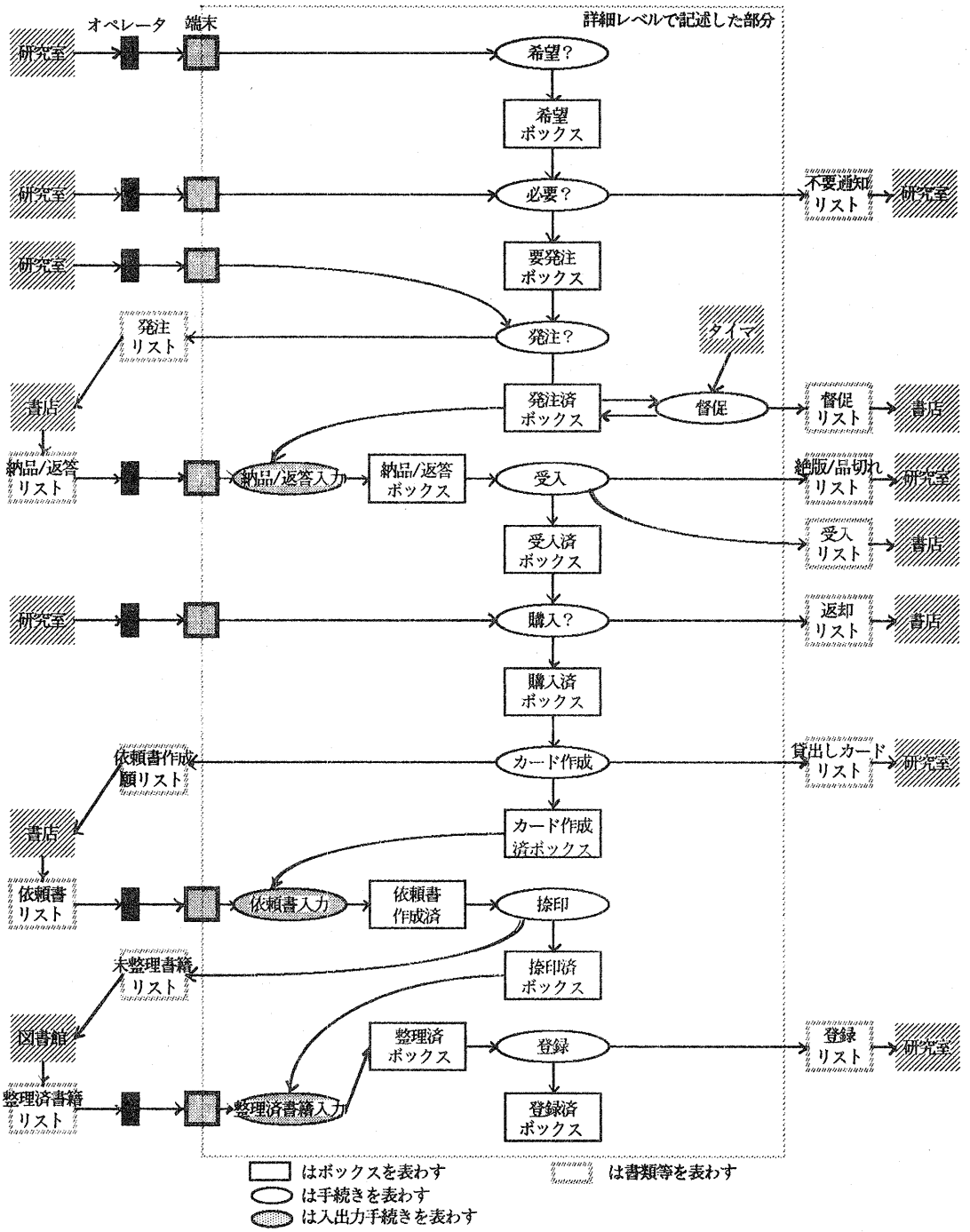

図4．1　詳細レベルの記述の概念図

算機外に、必要な情報を出力する。例えば、督促手続きは、その入力である発注済ボックスのレコードを参照し、督促リストを計算機外に出力する。書店は、督促状を見て、書店の状態を変えるものとする。

4．3　手続きの実行順序

単一プロセッサの計算機上で、抽象レベルの仕様記述に基づいた実現を行うためには、手続きの実行順序を陽に記述しなければならない。このために、手続きの実行順序を制御するスケジューラを記述する。スケジューラは抽象レベルで記述されたinvalid関数が真にならないような手続きを実行する。スケジューラは、入力手続きも含め、計算機上で処理するすべての手続きの実行順序を制御する。ここでの実現におけるスケジューラのアルゴリズムは、以下のように決める。

(1)手続きは、できる限り同じものを続けて実行する。

(2)オペレータの入力を必要としない手続きを優先する。

(3)オペレータの入力を必要とする手続きについては、オペレータが次に実行する手続きを指定する。

表5．1　詳細レベルの代数的記述

```
SPEC  図書購入業務
#  ボックスの性質
CON   init_B :                              →  ボックス
      put    :  ボックス, レコード          →  ボックス
      del    :  ボックス, レコード          →  ボックス
OP    rec    :  ボックス                    →  レコード
      empty  :  ボックス                    →  bool
B1: empty?(init_B)==true
B2: empty?(put(B, d))==false
B3: rec(init_B)==error
B4: rec(put(init_B, d))==d
B5: rec(put(put(B, d1), d2)=d2 or
      rec(put(put(B, d1), d2)=rec(put(B, d1))==true
B6: rec(del(B, d))≠d==true
B7: del(init_B, d)==init_B
B8: del(put(B, d1), d2)==
       if d1=d2 then B
       else put(del(B, d2), d1)
#  スケジューラ
CON   main     :  S        →  S
      next     :  S        →  S
OP    select   :  S        →  p_number
      e        :  S        →  bool
       :
S1: main(s)
   ==  if e(s) then
         s
       else
         main(next(s))
S2: next(s)
   ==  case select(s) of
         4:   納品(s)
         6:   チェック2(s)
                :
       endcase
    :
```

5．詳細レベルの仕様記述

この章では、抽象レベルで記述された仕様を計算機を用いて実現するために、4章で述べた方針に従って、計算機で処理する部分のより詳細な仕様をＡＳＬで記述する。その記述の一部を表5.1に示す。

5．1　手続き及び入力手続きの記述

ボックスに対する操作として

rec(B)　：ボックスＢ中のレコード１つを示す演算。

put(B. r)　：ボックスＢにレコードｒを入れたあとのボックスを示す演算。

```
#  手続き, 入力手続き
CON 【BOXES =
      〈発注ボックス, 納品1ボックス, 返却ボックス, …〉
      S  =
      〈BOXES, time, I/O 〉】
      購入?      : S      →S
      納品返答入力: S      →S
         :
P1: 購入?(受入済ボックス)
     ==  if  need?(rec(受入済ボックス))  then
           〈del(受入済ボックス, rec(受入済ボックス)),
            put(購入済ボックス, rec(受入済ボックス))〉
         else
           〈del(受入済ボックス, rec(受入済ボックス)),
            output(返却リスト, rec(受入済ボックス))〉
I1: 納品／返答入力(発注済ボックス)
    ==  if 見計らい(list_of(発注済ボックス), I/O )  then
          〈put(納品／返答ボックス, new_納品11(I/O ))〉
        else
          〈del(発注済ボックス,
               numtorec(発注済ボックス, I/O )),
           put(納品／返答ボックス,
               new_納品12(numtorec(発注済ボックス, I/O))〉
#  レコードの項目
SPEC  レコード(研究室図書業務)
CON   new_納品12: 発注済レコード, I/O  →納品1レコード
          :
R1: 書名   (new_納品12(発注済レコード, I/O ))
    ==  書名   (発注済レコード)
R2: 著者名(new_納品12(発注済レコード, I/O ))
    ==  著者名(発注済レコード)
R3: 出版社(new_納品12(発注済レコード, I/O ))
    ==  出版社(発注済レコード)
R4: 書店   (new_納品12(発注済レコード, I/O ))
    ==  書店   (発注済レコード)
R5: 価格   (new_納品12(発注済レコード, I/O ))
    ==  read_価格(I/O )
                :
```

del(B,r)　:ボックスBからレコードｒを取り除いた後のボックスを示す演算。

empty(B)　:ボックスBが空か否かを示す演算。

を考え、これらを用いて、手続きを記述する。

各ボックスの"状態"の並び(BOXES)と抽象的な時刻(time)、及びインタフェースの"状態"(I/O)の組Ｓ＝〈BOXES,time,I/O〉を"状態"とみなす。Ｓが各手続きでどのように変化するかを示すことによって手続きを記述する。すなわち各手続きは状態遷移関数である。ただし各手続きですべてのボックスが変化するとは限らないので、このレベルの仕様記述では引数としてその手続きが参照するボックスのみを書き、値域には、手続きを実行したとき"状態"を変えるボックスのみを書く。また、time、I/Oについては特に書かないが、すべての手続きで引数に入っているものとする

５．２　レコードの作り方の記述

レコードに対する操作として、レコードｒの各項目Aiの値を示す関数Ai(r)を考える。これを用いてレコードr_0と端末からの入力I/Oから新しいレコードｒ(型t)を作る関数new_t(r_0,I/O)を次のような形で定義する。

r_0の項目Aiが定義されているとき、

Ai(new_t(r_0,I/O)　==　Ai(r_0);

r_0の項目Aiが定義されていないとき、

Ai(new_t(r_0,I/O)　==　read_Ai(I/O);

ただしread_Aiは項目Aiを端末から読込む関数である。

５．３　スケジューラの記述

スケジューラも手続きと同様に"状態"Ｓから"状態"Ｓへの状態遷移関数として記述する。これは手続き及び入力手続きの実行順序を４．３で述べたように制御するものである。入力手続きは"オペレータの入力を必要とする手続き"と同様に扱うことができる。表5.1のスケジューラの記述の部分において、ｅは業務を終了するか否かの述語,selectは抽象レベルの記述におけるinvalid関数が真にならないように手続きを選択する関数である。

５．１、５．２、及び、５．３より、抽象レベルの仕様と詳細レベルの仕様の対応関係は明確である

６．ASL/Fによるプログラム

この章では、詳細レベルの記述をASL/Fプログラムに変換する。ボックス、レコードをASL/Fプログラムにおけるどのようなデータ構造に対応させるかを決定することにより、各手続きはプログラムにおける関数に機械的に変換できる。

６．１　データ構造の決定

プログラムにおけるデータ構造を次のように決定する。

［レコード］　現在のASL/Fコンパイラは、配列の要素としてレコードを直接扱えないので、レコードは項目を区切り記号で区切った文字列で表わす。項目の並びは、どの型のレコードも同じとし、定義されていない項目は長さ０の文字列とする。項目は、書名、著者名、出版社、書店、発注日、督促日、価格、絶版、受入日、登録番号、請求番号、日付が、この順に並んでいるものとする。

［ボックス］　全ボックスを、文字列を要素とする１つの２次元配列で表わし各ボックスを配列の各行で表わす。つまり配列のi（$1 \leq i \leq n$：nはボックス数）行の各要素がボックスBi中のレコードを表わす。すべてのボックスを一つの配列にまとめたのは、配列に対するオペレーションが１つだけで済むので、コーディングが効率的に行えるからである。

［端末の状態、時刻］　５章の記述における端末の状態(I/O)と時刻(time)は、それぞれ入出力ファイルの状態、計算機内部の時計が示す時刻で表わす。

６．２　レコードに対する操作の実現

レコードに対する操作にはレコードdの特定の項目Aiを表わす関数Ai(d)と，レコードd_0と端末の"状態"(I/O)から新たなレコードd（型t）を作る関数new_t(d_0,I/O)がある。Ai(d)は６．１により、レコードdの(i-1)番目の区切り記号からi番目の区切り記号までの文字列を取り出す関数として実現できる。d_0の項目Aiが定義されていればそれをnew_t(d_0,I/O)の項目Aiとし、定義されていなければ、端末からの入力をnew_t(d_0,I/O)の項目とし、それらを区切り記号で区切って連結することにより関数new_t(d_0,I/O)が実現できる。

６．３　ボックスに対する操作の実現

ボックスに対する４つの操作は、全ボックスを２次元配列で表わしたため配列から配列への関数として実現できる。

[rec(Bi)]　ボックスBiの中のレコードを取出す操作は、配列のi行目の要素の値を取出す関数として実現できる。このとき配列のインデックスが最小の要素を取出すことにする。ボックス中にレコードがない場合はnull（長さ０の文字列）を返す。

[put(Bi,d)]　ボックスBiにレコードを入れる操作は、配列のi行目の要素がnullである列に、レコードに相当する文字列を格納する関数として実現できる。

[del(Bi,d)]　ボックスBiからレコードを取り除く操作は、レコードdの格納されている配列上の場所にnullを格納する関数で実現できる。

[empty(Bi)]　"ボックスBiが空か否か"は、"rec(Bi)がnullか否か"で実現できる。

６．４　手続きおよびスケジューラの実現

詳細レベルでの仕様記述では、各手続き及びスケジューラは、各ボックスの状態の並び(BOXES)、時刻(time)、及び、端末の状態(I/O)の組ＳからＳへの関数として定義されている。従って、ボックスを表わす配列(BOXES)、計算機内の時計が示す(TIME)、入力ファイルノ状態(I/O)の組をＳとし、プログラムでの手続き、スケジューラを、Ｓからへの関数として表わすことにより詳細レベルでの記述とほぼ同じ形で関数が書ける。

7．記述の正しさ

詳細レベルの手続きは、キーボードからの読み込みや画面への書き出しのための基本関数(read,write)を用いて記述されている。詳細レベルの仕様が抽象レベルの仕様を満たしていることを言うためには、これらの基本関数のもつ意味を厳密に記述しなければならない。

また、詳細レベルでの記述では、オペレータが、研究室の状態から得られる値や研究室外の手続きの出力を、計算機へ正確に入力することを前提としている。つまり、オペレータが決められた動作をして、はじめて、詳細レベルの仕様は抽象レベルの仕様と同じ意味をもつ。

以下では、入出力のための基本関数の意味とオペレータの動作をASLで代数的に記述する（具体的な仕様記述は省略）。

7．1　入出力関数の代数的記述の方針

基本関数には、出力バッファに文字列を出力するwriteと、入力バッファから文字列（1行）を入力し、入力中の文字を出力バッファに出力する（エコーバックする）readがある。これらが、ASL/Fの組み込み関数(sgetl,sputl)と同じ意味をもつように記述する。

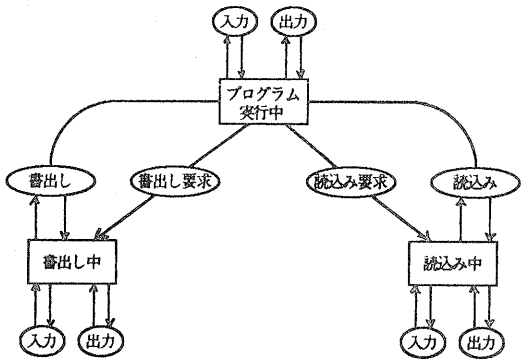

図7．1　入出力関数による状態遷移図

関数writeは書出し要求と書出しという手続きに分けられる。関数writeが呼び出されると、書出し要求が実行されプログラムは書出し中という状態になる。writeによって書出すべき文字列が書出しによって1文字ずつ出力バッファに書出され、書出すべき文字列がなくなったところでプログラムはもとの状態にもどる。関数readは、読込み要求と読込みという手続きに分けられる。関数readが呼び出されると、読込み要求が実行され、プログラムは読込み中という状態になる。読込みによってキーボードから入力バッファへ1文字読込み、同時にその文字を出力バッファへ書出す。1行の読込みが完了したら、もとの状態へ戻る（図7.1参照）。

キーボード入力を入力バッファに入力する操作（入力と呼ぶ）と出力バッファの値を画面に出力する操作（出力と呼ぶ）はASL/Fのプログラム外で行われる。またこれらは、1文字単位で行われる。入力バッファがfullのときの入力は無視され、出力バッファが空のとき出力は何もしない。

以上のことをASLで代数的に記述する。

7．2　オペレータ動作の代数的記述の方針

研究室外の手続きの出力はレコードの形でボックス中に存在する。したがって5章で定義した関数によってレコードの項目の値を知ることができる。また、計算機への入力は、7．1で記述した関数を使って記述できる。

8．あとがき

書店及び図書館の手続きが抽象レベルの仕様を満たしているならば、詳細レベルの仕様記述、端末の代数的記述、及びオペレータの動作の代数的記述をあわせたものは、抽象レベルの仕様で書かれた要求（公理で記述されている）を満たしている。

詳細レベルの仕様記述からASL/Fプログラムへの変換は直接的であるからASL/Fのプログラムが詳細レベルの仕様を満たしていることは容易にわかる。

図書購入業務の仕様、プログラムのサイズは次の通りである。仕様記述の部分において、関数名や変数名には漢字も使用し、それらは平均して3文字程度である。

抽象レベルの仕様・・・・約1万文字
詳細レベルの仕様・・・・約2万文字
ＡＳＬ/Ｆプログラム　・・約2.4万文字

業務の手続き数は12個である。

業務の抽象化や記述スタイルの決定及び仕様を記述するのに数ヶ月、プログラムの作製に約1週間かかった。

謝辞　適切な御指導を頂いた嵩忠雄教授に感謝します。

参考文献

(1)　嵩，谷口，杉山：”代数的言語の設計と処理系”，榎本編：”ソフトウェア工学ハンドブック”，オーム社.
(2)　嵩，谷口，杉山，関：”代数的言語ＡＳＬ／＊の意味定義について”電子通信学会論文誌(D)投稿中.
(3) 後藤信一　：”ASL/F1外部仕様書”，(1985-07).
(4) Kunin,J.S.：”Analysis and Specification of Office Procedure”, Doctoral dissertation, MIT (1982-02).
(5) 東野，工藤，縄田，杉山，谷口：”代数的仕様検証支援系　及びそれを用いた検証例”，電子通信学会論文誌(D), J67-D,4, pp.472-479 (昭59-04).
(6) 杉山，谷口，嵩：”基底代数を前提とする代数的仕様”，電子通信学会論文誌(D), J64-D, 4, pp.324-331(1981-04).
(7) 杉山，谷口，嵩：”代数的記述言語ASL/1－文法とその意味の定義－”，電子通信学会技術研究報告，AL82-62, (1982-11).
(8) 梶谷，伊藤，松浦，谷口，嵩：”オフィスワークの代数的記述と検証－図書購入業務の記述－”，電子通信学会技術研究報告，AL84-38, (1984-11).