数理解析研究所講究録
第470巻 1982年 46-56

PDL — Process-Data Language

瀬川 清(早大)

1. はじめに

PDL(Process-Data Language)は仕様記述法PDR(Process-Data Representation)[4,5]による仕様から算譜を作るための言語である。PDRの哲学に合致するものならば,複数のPDLが存在してもかまわない。むしろ,使用環境に応じたいくつかのPDLがあるべきだと言える。ここでは,DECSYSTEM-20上で実現した(一つの)PDL[3,7]について述べる。

2. PDRとPDL

PDRには具体的に決められていない点があるので,PDLを設計するときには,そのような点を決める必要がある。また,変更した点や実現の上で制限した点もある。

プロセスの種類と個数

PDRにはスタンダード,インタラプティブ,ポストルードの3種類のプロセスがある。PDLでは,スタンダード・プロセスをプレリュード・プロセスと呼び,一つのクラスタ内には高々一つあるものとする。起動条件はなく,常に実行

される。またインタラプティブ・プロセスをインタールード・プロセスと呼ぶ。ポストルード・プロセスは一つのクラスタ内には高々一つあり，起動条件はない。

3種類のプロセスは次のように実行される。

あるクラスタの実行が始まると，プレリュード・プロセスが（あるならば）起動される（このとき，クラスタはプレリュード状態にあると言う）。このプロセスが終了すると，インタールード・プロセスが（あるならば），PDRにおけるインタラプティブ・プロセスと同じように，起動される（このとき，クラスタはインタールード状態にあると言う）。起動されたすべてのインタールード・プロセスが終了し，すべてのインタールード・プロセスの起動条件が偽のとき，インタールード状態は終了する。そして，ポストルード・プロセスが（あるならば）起動される（このとき，クラスタはポストルード状態にあると言う）。このプロセスが終了したときに，クラスタの実行が終了する。

クラスタ間の共用変数

PDLでは，クラスタ間に共用変数はない。クラスタ間でデータのやりとりが必要な場合は，使用者の責任のもとで，外部ファイルを用いる。

強制式の右辺

強制式の右辺に現われる強制演算子は，$\langle d_1, \cdots, d_m \rangle : n$ の形のもののみで，また入れ子もないものとする。この制限のもとでは，必要なデータを確保することと際どい部分（critical section）に入いることは同一視できるので，強制式の右辺を実現する必要がなくなる。

順路式による記述

PDRでは，exec-path と ref-path の2種類の順路式を用いているが，PDLではexec-pathのみを用いる。

3. PDL算譜

PDL算譜の一般形を示す（次ページ）。

cluster path expression ── クラスタ順路式

クラスタの実行順序を順路式の記法により表わす。

cluster total condition ── クラスタ総合条件

クラスタ内のプロセスと共用データの関係を強制式により表わす。

declaration for shared data ── 共用データ宣言

クラスタ内の共用データを宣言する。また強制式により，参照のされ方を表わす。

```
program  program_name
   cluster path expression /* specify the relation
                              among clusters using
                              path expression notation */

   /* specification of a cluster */
   cluster  cluster_name
      cluster total condition  /* using forcing logic */

      /** data specification **/
         declaration for shared data /* using forcing logic */

      /** process specification **/
         process total condition /* specify the relation
                                    among processes using
                                    path expression notation */

         /* process declaration */
            prelude_process  process_name
              (or interlude_process
                  or postlude_process)
               activation condition /* Boolean expression */

               data requirement /* reference declaration
                                   for shared data using
                                   forcing logic */

                    program body  /* a complete SIMPL program
                                     corresponding to
                                     this process */
```

process total condition —— プロセス総合条件

プロセスの実行順序を順路式記法により表わす。この条件はプロセスの起動条件の一部とみなす。

activation condition —起動条件

プロセスが起動される時点を論理式により表わす。

data requirement — データ要求

プロセス内で参照する共用データを強制式により表わす。

program body —— プロセス本体

プロセスの本体の算譜をSIMPL-Iにより表わす。

強制式と順路式による条件の記述は必要な所のみでよい。ただし，よけいに記述しても，矛盾しなければよい（この種の冗長性はＰＤＲの哲学の一つである）。

算譜中では，強制式 "$P \xrightarrow{cs} D$" は "cs(P, D)" と記述され，隙どり部分 "cs"（の中味）は，"cs::(---)" と記述される。

ＰＤＬの算譜例を示す。

```
program  Dining_Philosophers
   cluster Main
      data specification
         INT FORK1 : eat ( [PH5,PH1]:1 , )
         INT FORK2 : eat ( [PH1,PH2]:1 , )
         INT FORK3 : eat ( [PH2,PH3]:1 , )
         INT FORK4 : eat ( [PH3,PH4]:1 , )
         INT FORK5 : eat ( [PH4,PH5]:1 , )
      end data specification
      process specification
         interlude process PH1
            forcing eat ( , <FORK1,FORK2>:2 )
            PROC PH1_MAIN
               INT TIME
               WHILE true DO
                  WRITE ('PH1 THINKING')
                  TIME:=RANDOM
                  WHILE TIME > 0 DO TIME:=TIME-1 END
                  WRITE ('PH1 HUNGRY')
                  TIME:=RANDOM
                  eat :: ( WRITE ('PH1 EATING')
                           WHILE TIME > 0
                              DO TIME:=TIME-1 END )
               END
            START PH1_MAIN
         end process PH1
         /* same specifications for
               processes PH2,...,PH5 */
            .
            .
            .
      end process specification
   end cluster Main
end program  Dining_Philosophers
```

共用データ宣言

一つのプロセス

データ要求

隙どり部分 eat

```
program READERS_WRITER
  cluster RAW
    forcing
      ACCESS( [ [READ1,READ2]:2 , WRITE1 ]:1 ,<FILE>:1 )
    data specification
       INT FILE
    end data specification
    process specification
      relation
        path READ1 end
        path READ2 end
        path WRITE1 end
      interlude process READ1
        body
        PROC MAIN
              .
              .
           ACCESS::( /* access to FILE */ )
              .
              .
        START MAIN
      end process READ1
      interlude process READ2
        body
        PROC MAIN
              .
              .
           ACCESS::( /* access to FILE */ )
              .
              .
        START MAIN
      end process READ2
      interlude process WRITE1
        body
        PROC MAIN
              .
              .
           ACCESS::( /* access to FILE */ )
              .
              .
        START MAIN
      end process WRITE1
    end process specification
  end cluster RAW
end program READERS_WRITER
```

クラスタ統合条件

強制式のない共用データ宣言

プロセス統合条件（実は必要ない）

隙ぐい部分 ACCESS

プロセス本体（ほぼ定型な SIMPL の算譜）

PDL(R)では，際どい部分に関する「同期操作」の記述（強制式による記述）と際どい部分の「中味」の記述（プロセス本体中にある）を分離することができる。このことにより，同期操作に関する検証が容易になることが予想される。

4. PDL処理系

PDL処理系はDECSYSTEM－20上で実現されている。処理系はSIMPLで記述され，大きさは約6000行である。PDL算譜を1パスで処理し，（SIMPLで記述された）目的算譜を生成する。

処理系はプロセス中に際どい部分をみつけると，対応する強制式から防護（guard）を作り，際どい部分の前後に置く。簡単な強制式に対する防護を示す（一般の強制式に対する防護については[6]参照のこと）。

高々演算子（at most operator）

高々演算子のために，次のような拡張2進セマフォを用意する。

```
type_extended_binary_semaphore =
   record
      b_sem : binary semaphore ;
      wait_queue : queue    ;
      event_queue : queue
   end
```

強制式 $[P_1, \cdots, P_m] : k \xrightarrow{CS}$ は次のように処理される：

S をカウンタ，op を拡張2進セマフォとする。

```
initialization :
    s := k ;
    op.b_sem := 1 ;
    op.wait_queue := empty ;
    op.event_queue := empty ;
```

← PDL 算譜全体を初期化する部分に置かれる

```
prologue :
    P(op) ;
       if s > 0 then s := s - 1
       else await(op)
       fi ;
    V(op) ;

code for the critical operation

epilogue :
    P(op) ;
       s := s + 1 ;
       cause(op) ;
    V(op) ;
```

際どい部分に対する防護

Brinch Hansen の cause/await とほぼ同じ。await したものは，その前にある P から，実行を再開する。

注．

前に述べた強制式の右辺に対する制限から，右辺（データ側）を実現する必要はない。

少なくとも演算子（at least operator）

少なくとも演算子を実現するため，次のような逆セマフォと逆セマフォ演算を用意する。

```
type inv_semaphore =
   record
      counter : integer ;
      init : integer ;
      active : boolean ;
      wait_queue : queue
   end
```

```
<P> (var s : inv_semaphore) =
   begin
      s.counter := s.counter - 1 ;
      if not s.active then
         if s.counter > 0 then
            block in s.wait_queue
         else
            if there are blocked processes
                  in s.wait_queue then
                    wake up them
            fi ;
            s.active := true
         fi
      fi
   end ;
```

```
<V> (var s : inv_semaphore ) =
   begin
      s.counter := s.counter + 1 ;
      if s.counter = s.init then
         s.active := false
      fi
   end ;
```

強制式 $\langle P_1, \cdots, P_m \rangle : k \xrightarrow{CS}$ は次のように処理される：

op を逆セマフォとする。

```
initialization :
     op.counter := k ;
     op.init := k ;
     op.active := false ;
     op.wait_queue := empty ;

prologue :
     <P> (op) ;

code for the critical operation

epilogue :
     <V> (op) ;
```

5. まとめ

ＰＤＬ処理系を実現した現在，ＰＤＲは並列処理の記述系であるといえる。今後の課題としては，検証法の確立および検証系（ＰＤＶ── Process-Data Verifier）の作製などがある。

謝辞

ＰＤＲ研究会の諸氏，特に慶瀬 健先生（早稲田大学）および土居 範久先生（慶応義塾大学），またＰＤＬ処理系作成に協力してくださった板東 浩之（慶応義塾大学，現在ソニー），野田 晴義（慶応義塾大学，現在日本ＩＢＭ）の両氏に感謝する。ＳＩＭＰＬ処理系を使用させていただいた慶応義塾大学の原田 賢一先生，三嶋 良武氏（現在三菱総合

研究所），鹿野　芳之氏に感謝する。

References

[1] Basili,V.R. and Turner,A.J., "SIMPL-T : A Structured Programming Language", University of Maryland, Computer Science Center, CN14.2, (1975).
[2] Brinch Hansen,P., "Operating System Principles", Prentice-Hall, (1973).
[3] Doi,N., Segawa,K., Bando,H. and Noda,H., "PDL --- Process-Data Language", Technical Report KIIS, Institute of Information Science, Keio University, 1982.
[4] Hirose,K., Saito,N., Doi,N., Segawa,K. et al., "Process-Data Representation", Proc. of 3rd USA-Japan Computer Conference, p.225-230 (1978).
[5] Hirose,K., Saito,N., Doi,N., Segawa,K., et al., "Specification technique for parallel processing : Process-Data Representation", Proc. of AFIPS 1981 National Computer Conference, AFIPS Conference Proceeding, Vol.50, p.407-413 (1981).
[6] Segawa,K., "A new synchronization primitives --- forcing logic and its implementation", to appear.
[7] 瀬川，板東，野田，土居，"並列型プロセスの仕様を記述するための言語とその実現"，情報処理学会ソフトウェア工学研究会資料 82-22，P.62-72 (1982).