1G－5

ASLプログラム開発システムにおける検証の自動化について

森岡 澄夫　岡野 浩三　北道 淳司　東野 輝夫　谷口 健一

大阪大学 基礎工学部 情報工学科

1 まえがき

筆者らは代数的言語 ASL を定め，それを用いた設計法を提案し，また検証支援系を作成して，その有用性について調べてきた．代数的手法を用いた検証は，基本的には項書き換え等の単純な作業を組み合わせて行えるが，一般に検証作業が繁雑になる．筆者らは現在，ASL 順序機械型プログラム [1] の検証作業の自動化を目指して検証支援系を改良している [2]．

本稿では，ソーティングを行う ASL 順序機械型プログラムを例に，その検証支援系の機能を説明する．また，検証結果についても述べる．

2 ASL による順序機械型プログラムの記述と設計

配列 ARY（0〜Nの範囲を持つ）の $I1, I2$（$ARY, I1, I2$ は状態成分）間のソーティングを行う ASL 順序機械型プログラムの要求仕様記述を表 1に示す．

遷移 sort が満たすべき性質を，配列 aryの i,j間が昇順に並んでいることを表す述語 $Ordered(ary,i,j)$，配列 $ary1$ の i,j間と $ary2$ の k,l間の要素集合が多重集合として等しいことを表す述語 $SameSet(ary1,i,j,ary2,k,l)$ を用いて記述した．$Ordered$,$SameSet$ は基本述語とする．

表 1: ソートプログラムの要求仕様

```
計算指定:ARY(sort(init(initial_data,from,to)))
(* 状態成分の初期化 (init) についての記述は省略 *)
(* 遷移 sort の要求記述 *)
define 'sort 前提条件':='0 ≦ I1(s) ≦ I2(s) ≦ N';
性質 SORT1:(* I1,I2 間が昇順に並ぶ *)
sort 前提条件 →
    Ordered(ARY(sort(s)),I1(s),I2(s)) == TRUE;
性質 SORT2:(* I1,I2 間は集合として不変 *)
sort 前提条件 →
    SameSet(ARY(s),I1(s),I2(s),
            ARY(sort(s)),I1(s),I2(s)) == TRUE;
```

遷移 sort をマックスソート法により実現することにしよう．レベル 2 の状態遷移として，最大値を探す範囲を全範囲（i,j 間）に初期設定する遷移 InitBound，最大要素の場所を見付ける遷移 findMaxPos，その最大要素をソート済みの領域の隣の要素と入れ換えてソート済み範囲を 1 つ広げる遷移 exchange を導入する．表 2にそれらの遷移の満たすべき性質を記す．$bound$ は $I1, I2$ 間のうちソート済範囲を示すためのポインタ，$maxpt$ は $I1$, $bound$ 間の最大要素の場所を指すポインタである．配列 aryの i,j間のどのデータよりも kの指すデータがより大きいか等しいことを示す述語 $isMaxPos(ary,i,j,k)$ や，配列 $ary1$ の i,j間と $ary2$ の k,l間が配列として長さも要素も等しいことを表す述語 $SameArray(ary1,i,j,ary2,k,l)$ を用いている．

レベル 1 の遷移 sort をレベル 2 の遷移を用いてどのように展開するかの記述を表 3に示す．図 1 に状態遷移図で表した実行制御を示す．

以降，遷移 findMaxPos や exchange をコンパイル・実行できるレベルまで詳細化する．

表 2: レベル 2 の各遷移の動作内容

```
(* ここでは，状態成分の内容が変わらないことの記述は省略．*)
(*** 遷移 nop ではどの状態成分も不変．***)
(*** 遷移 InitBound の動作内容 ***)
性質 initbd1: bound(InitBound(s)) == I2(s);
(*** 遷移 findMaxPos の動作内容 ***)
define 'fmax 前提条件':=
    '(0 ≦ I1(s) ≦ bound(s) ≦ N)';
性質 fmax1: fmax 前提条件 →
  I1(s) ≦ maxpt(findMaxPos(s)) ≦ bound(s) == TRUE;
性質 fmax2: fmax 前提条件 →
  isMaxPos(ARY(s),I1(s),
       bound(s),maxpt(findMaxPos(s))) == TRUE;
(*** 遷移 exchange の動作内容 ***)
性質 exchg1: bound(exchange(s)) == bound(s)-1;
define 'exchange 前提条件':=
  '(0 ≦ maxpt(s) ≦ N and 0 ≦ bound(s) ≦ N)';
性質 exchg2: exchange 前提条件 →
  iget(ARY(exchange(s)),bound(s))
    = iget(ARY(s),maxpt(s)) == TRUE;
性質 exchg3: exchange 前提条件 →
  iget(ARY(exchange(s)),maxpt(s))
    = iget(ARY(s),bound(s)) == TRUE;
性質 exchg4:
(* bound, maxpt の指す箇所以外は不変 *)
exchange 前提条件 → (
  (maxpt(s) ≦ bound(s) → (
    (0 < maxpt(s) →
      SameArray(ARY(s),0,maxpt(s)-1,
        ARY(exchange(s)),0,maxpt(s)-1)) and
    (maxpt(s)+1 < bound(s) →
      SameArray(ARY(s),maxpt(s)+1,bound(s)-1,
        ARY(exchange(s)),maxpt(s)+1,bound(s)-1)) and
    (bound(s) < N →
      SameArray(ARY(s),bound(s)+1,N,
        ARY(exchange(s)),bound(s)+1,N)) )
  ) and
  (bound(s) < maxpt(s) → (
    省略（上の maxpt と bound が逆になったもの)
  )) == TRUE;
```

表 3: 遷移 sort の展開

```
sort(s) == S1(InitBound(s));
S1(s)   == if (I1(s) < bound(s)) then
               S1(exchange(findMaxPos(s)))
           else
               nop(s);
```

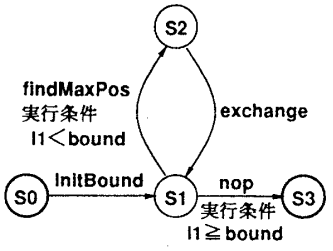

図 1: 遷移 sort を実現する実行制御の状態図

Facilities for Automatic Verification
in ASL Program Development System
Sumio MORIOKA, Kozo OKANO, Junji KITAMICHI,
Teruo HIGASHINO and Kenichi TANIGUCHI
Deperment of Information and Computer Sciences,
Faculty of Engineering Science, Osaka University
Toyonaka-shi, Osaka 560 Japan

3 検証支援系を用いた検証例

上記の詳細化の正しさを検証する．検証者はSORT1，SORT2の証明のために，図1のS1における不変式$R1(s)$, $R2(s)$（アサーション）を考案する（表4）．さらに，基本述語について，検証に必要と思われる性質（公理）を記述し（表5），証明に使うと思われる基本述語の公理の具体式（変数に状態成分値等を代入した式）を作る．

その後，検証者が検証実行を支援系に指示する．そうすると支援系は以下の操作を自動実行する．

(1) 実行制御の状態図において，始点，終点，不変式を割り付けた状態間の全てのパスを列挙する．

(2) 各パスについて以下を実行する．

(3) SORT1の検証を例に具体的に説明する．例えばパス$S1 \rightarrow S2 \rightarrow S1$について，証明すべき式

$$(R1(S)\ and\ I1(S) < bound(S)) \rightarrow R1(exchange(findMaxPos(S)))$$

を生成する．

(4) 公理SORT1の前提条件（$0 \leqq I1(S0) \leqq I2(S0) \leqq N$），レベル2の遷移findMaxPosの性質記述fmax1，fmax2等（変数sにはSを代入する），exchangeの性質記述exchg1〜exchg4等（変数sには$findMaxPos(S)$を代入する），および検証者が与えた基本述語の公理の具体式を証明すべき式の前提として付加する．

検証者は検証実行の前に，基本述語の公理の各具体式毎に，それをどのパスの証明で使うかを支援系に指示しておく．例えば表5の性質prm2の変数$a1$に$ARY\ (exchange\ (findMaxPos(S)))$，$i1$，$i2$にそれぞれ$bound\ (exchange\ (findMaxPos(S)))+1$，$I2\ (exchange\ (findMaxPos(S)))$を代入した式を，パス$S1 \rightarrow S2 \rightarrow S1$の証明で使用する．

(5) 得られた式中の項を変数に置き換え，ブール・整数上の論理式を得る（論理式中の整数の演算は加減算のみであるようにする）．

(6) その論理式の恒真性を判定する．恒真ならそのパスに対する証明は成功である．

配列の性質を$Ordered$のような一つの述語で表す記述スタイルを取れば，(6) での論理式が真となれば (4) の段階での論理式も真である．全てのパスに対する証明が成功すれば，性質SORT1が成り立つ（但し，検証者が導入した基本述語の性質が正しいという前提で）．

本支援系では，加減算のみからなる整数上の論理式の恒真性（全ての変数が全称記号で束縛された冠頭標準形の論理式が真であること）を高速に判定できるアルゴリズムを実現している（文献[3]の方法に基づいているが，変数の消去順等で工夫を行っている）．一般には配列を含むプレスブルガー算術は決定不能であり[4]，検証は困難であるが，本稿での例のように配列全体の性質を表す述語を導入することで証明に成功する場合がある．

表6にSORT1，SORT2の証明における，各パス毎の証明に要したCPU時間と，用いた基本述語の公理の具体式の数，および恒真性を判定した論理式の式長を示す．式長が1000トークン程度であっても，数秒で判定できる．

本稿では詳細は述べないが，遷移findMaxやexchange等の詳細化の正しさについても，それぞれ検証を行った．より下位レベルでの各遷移の性質記述は表2の性質initbd1のように状態成分への代入値を直接指定するものが多く，検証はレベル1−2間より容易で，各パスの証明も高速に行える．

4 あとがき

本稿では，ASL順序機械型プログラムの設計検証の自動化を目指した支援系について述べた．3.では述べなかったが，支援系は，(3) で生成した証明すべき式に対して自動項書き換えを行って式を簡単にすることにより，(6) での真偽判定をより高速に行うこともできる（項書き換えを行うかどうかは検証者が指定できる）．その代わり，基本述語の公理の変数への代入値を求める作業が難しくなる．

今後の課題としては，基本述語の性質の変数への代入値を求めるための支援や，証明失敗の原因を解析するための支援についての検討等が挙げられる．

参考文献

[1] 大蘆雅弘, 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 (DI),**J73-D-I**, No.12, pp.971-978(平2-12).

[2] 森岡澄夫, 北道淳司, 東野輝夫, 谷口健一:” 代数的手法を用いた順序機械型プログラムの設計検証”, 信学技報,**SS92-11** (1992-09) .

[3] D.C.Cooper: Theorem Proving in Arithmetic without Multiplication, Machine Intelligence, No.7(1972),pp.91-99.

[4] Suzuki,N. and Jefferson, D. : Verification Decidability of Presburger Array Programs, A Conference on Theoretical Computer Science(1977), Univ. of Waterloo, Canada, pp.202-212.

表4: レベル1−2間の検証で用いた不変式

```
＜SORT1の証明における不変式＞
R1(s):
0≦I1(S0)＝I1(s)≦bound(s)≦I2(S0)＝I2(s)≦N and
(bound(s)＜I2(s) →
  (Ordered(ARY(s),bound(s)+1,I2(S0)) and
   and isMaxPos(ARY(s),I1(S0),
            bound(s)+1,bound(s)+1)))
＜SORT2の証明における不変式＞
R2(s):
0≦I1(S0)＝I1(s)≦bound(s)≦I2(S0)＝I2(s)≦N and
SameSet(ARY(S0),I1(S0),I2(S0),ARY(s),I1(S0),I2(S0))
```

表5: レベル1−2間の検証で用いた基本述語の性質

```
prm1: (* 要素が1個の領域は Ordered *)
0≦i1＝i2≦N → Ordered(a1,i1,i2) == TRUE;
prm2: (* Ordered な領域を1つ拡大する *)
(0≦i1＜i2≦N and
   and Ordered(a1,i1+1,i2)
   and iget(a1,i1)≦iget(a1,i1+1))
→ Ordered(a1,i1,i2) == TRUE;
prm3: (* 要素が1個なら それは最大 *)
0≦i1＝i2＝i3≦N
→ isMaxPos(a1,i1,i2,i3) == TRUE;
prm4: (* isMaxPos の領域を1つ拡大する *)
(0≦i1＜i2≦N and 0≦i3≦N
    and isMaxPos(a1,i1,i2-1,i3)
    and iget(a1,i2)≦iget(a1,i3))
→ isMaxPos(a1,i1,i2,i3) == TRUE;
prm5: (* 同じ要素が1個の領域は SameSet *)
(0≦i1≦ N and 0≦i2≦N
    and iget(a1,i1)＝iget(a2,i2)
→ SameSet(a1,i1,i1,a2,i2,i2) == TRUE;
prm6: (* SameSet な領域の端に同じデータを
         付け足しても SameSet *)
(0＜i1≦i2≦N and 0≦i3≦i4＜ N
    and SameSet(a1,i1,i2,a2,i3,i4)
    and iget(a1,i1-1)＝iget(a2,i4+1)
→ SameSet(a1,i1-1,i2,a2,i3,i4+1) == TRUE;
等
prm7: (* 隣り合った SameSet な領域を
          マージしても SameSet *)
(0≦i1＜i2≦i3≦N and 0≦i4＜i5≦i6≦N
    and SameSet(a1,i1,i2-1,a2,i4,i5-1)
    and SameSet(a1,i2,i3,a2,i5,i6))
→ SameSet(a1,i1,i3,a2,i4,i6) == TRUE;
等
prm8: (* SameArray なら SameSet *)
(0≦i1≦i2≦N
    and SameArray(a1,i1,i2,a2,i1,i2))
→ SameSet(a1,i1,i2,a2,i1,i2) == TRUE;
.........
```

表6: 各パスの証明で恒真性を判定した論理式の大きさと判定時間 (Sony NEWS−5000)

公理	パス	CPU時間(秒)	基本述語の公理の具体式の数	式長
SORT1	S0 → S1	0.01	0	69
	S1 → S2 → S1	5.98	36	1647
	S1 → S3	0.05	6	202
SORT2	S0 → S1	0.03	3	173
	S1 → S2 → S1	1.73	19	871
	S1 → S3	0.03	5	254