数理解析研究所講究録
第213巻 1974年 180-209

Regular Expression の同値性判定プログラム

電通大　金山　裕
鈴木康年

0. まえがき

Salomaaによって regular expression の同値性に対する無矛盾で、かつ完全な公理系がつくられた．この完全性の証明において，with $\alpha=\beta$ がこの公理系の定理であるか否かを判定するアルゴリズムが示されている[1].

我々は公理系を計算機で扱う1つの例として，このアルゴリズムに着目しregular expressionの同値性を判定するプログラム compar について述べる．我々はcompareをLISP言語[2],[3]で表現し，HITAC 8350によって実行した[4]．我々はSalomaaが示したアルゴリズムにしたがうが，やや異なるものを用いたのでcompareにおいてはregular expression α に対して α の方程式系をつくる（αを受理する有限オートマトンをつくる）ための独立なサブプログラムを用意してのない．

したがって α を受理するオートマトンをつくるプログラム

amについても簡単に述べる．amはLISP言語との比較のためSNOBOL3言語[5]で表現し，HEWLETT PACKARD 2100で実行したものについて示す．

またスタックを用いて同様のプログラムを表現する方法についても述べる．

1. 準備

$\Sigma = \{A, B, \cdots Z\}$ とする．Σ 上のregular expression（以下reと略記する）を次のように定義する．

(1) $0 \notin \Sigma$ はreである．

(2) $a \in \Sigma$ ならば a はreである．

(3) α，β が共にreならば $(\alpha \cdot \beta)$，$(\alpha + \beta)$，$(\alpha *)$ はそれぞれreである．この第3のものは（，）でくくってある点がSalomaaのものとは異なる．

例 1.1　A，$(0*)$，$(A + ((B \cdot C)*))$ はそれぞれreである．

Salomaaの公理系は文献[1]にあるが再掲すると次のようになる．axiom shema A1～A11 と推論規則 R1, R2 より構成される．

A1　$(\alpha + (\beta + \gamma)) = ((\alpha + \beta) + \gamma)$

A2　$(\alpha \cdot (\beta \cdot \gamma)) = ((\alpha \cdot \beta) \cdot \gamma)$

A3　$(\alpha + \beta) = (\beta + \alpha)$

A4　$(\alpha \cdot (\beta + \gamma)) = ((\alpha \cdot \beta) + (\alpha \cdot \gamma))$

A7　$((0*) \cdot \alpha) = \alpha$

A8　$(0 \cdot \alpha) = 0$

A9　$(\alpha + 0) = \alpha$

A10　$(\alpha *) = ((\alpha \cdot (\alpha *)) + (0*))$

A5 $((\alpha+\beta)\cdot\gamma) = ((\alpha\cdot\gamma)+(\beta\cdot\gamma))$ A11 $(\alpha^*)=((\alpha+(0^*))^*)$

A6 $(\alpha+\alpha) = \alpha$

R1: 代入 R2: 方程式の解

(A10, A11 は Salomaa のものと異なる.)

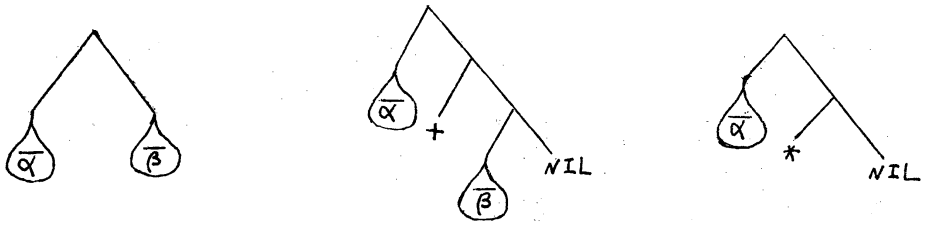
re α は次の形をしているとき normal form をしていると言い, normal form をしている re のことを normal re (以下 nre と略記する) と言う.

$\alpha \equiv (\alpha_m + \cdots + (\alpha_1 + \alpha_0)\cdots)$, $n \geq 0$, $\alpha_0 \equiv 0$ 又は $\alpha_0 \equiv (0^*)$, $1 \leq i \leq n$ に対して $\alpha_i \equiv (a_i \cdot \beta_i)$ 又は $\alpha_i \equiv a_i$ $a_i \in \Sigma$, β_i は re, ここで α_i を linear term, a_i をその handle, また $\alpha_i \equiv (a_i \cdot \beta_i)$ に対して β_i をその derivative, $\alpha_i \equiv a_i$ に対しては (0^*) をその derivative という. $n \geq 2$ ならば $a_n, \cdots, a_1$ は重複がなくかつアルファベット順になっている. このことを sort されていると言う.

<u>例 1.2</u> (0^*), $(A+0)$, $(A+((C\cdot(B^*))+((D\cdot(A+B))+0)))$ はそれぞれ nre で (0^*) の linear term は存在しない. $(A+0)$ の linear term は A でその handle は A, derivative は (0^*) である. 第3番目の nre の linear term は A, $(C\cdot(B^*))$, $(D\cdot(A+B))$ で, A, C, D がそれぞれの handle, (0^*), (B^*), $(A+B)$ がそれぞれの derivative である.

re の LISP における内部表現について述べる. re α の内部表現を $\overline{\alpha}$ とかく, $\overline{\alpha}$ は re α を S式とみなした値とする. (ただし

+ の前後と * の前には空白を入れるものとする.) したがって re α, β に対して $\overline{(\alpha\cdot\beta)}$, $\overline{(\alpha+\beta)}$, $\overline{(\alpha\,*)}$ を図式表現すると次のようになる.



nre α に対しては処理能率を向上させるために特別の内部表現 $\widetilde{\alpha}$ を用いる. nre $\alpha \equiv (\alpha_n + \cdots + (\alpha_1 + \alpha_0)\cdots)$ $n \geq 0$ に対して $\widetilde{\alpha}$ を次のようにする. $\widetilde{\alpha} = ((\overline{\alpha_n} \sqcup \cdots \sqcup \overline{\alpha_1})\cdot \widetilde{\alpha_0})$

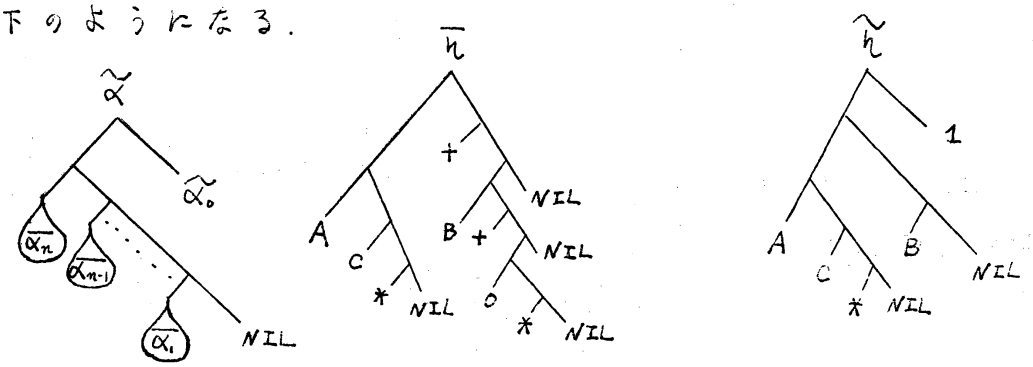
$$\widetilde{\alpha_0} = \begin{cases} 0 & : \alpha_0 \equiv 0 \text{ のとき} \\ 1 & : \alpha_0 \equiv (0\,*) \text{ のとき} \end{cases}$$

$car[\widetilde{\alpha}] = (\overline{\alpha_n} \sqcup \cdots \sqcup \overline{\alpha_1})$ を $\widetilde{\alpha}$ の linear part, $cdr[\widetilde{\alpha}] = \widetilde{\alpha_0}$ を $\widetilde{\alpha}$ の constant part と言う. $\widetilde{\alpha}$ の図式表現は下の左図のようになる.

例 1·3 $\eta \equiv ((A\cdot(C\,*)) + (B + (0\,*)))$ のとき

$\overline{\eta} = ((A\cdot(C \sqcup *)) \sqcup + \sqcup (B \sqcup + \sqcup (0 \sqcup *)))$

$\widetilde{\eta} = (((A\cdot(C \sqcup *)) \sqcup B)\cdot 1)$ でこれらは図式表現すると下のようになる.



2. プログラム compare

Salomaaは前節に示した公理系の完全性の証明のなかで次のことを示している[1].

(1)任意のre α に対して$\vdash \alpha=\beta$ なるnre β が存在する.(Lemma 4)

(2)任意のre α は方程式系を有す. (Lemma 4)

(3)任意の2つのre α,β の同値性は決定可能である. (Theorem 2)

この事実にもとずいて任意のre α,β に対して$\vdash \alpha=\beta$ (すなわちαとβが同値)であるか否かを判定するプログラムcompareをつくる. Salomaaは上の(1),(2)をアルゴリズムを示すことで同時に証明した. 我々がcompareにおいて用いたアルゴリズムはこれとはいくらか異なる. プログラムはLISPのM式によって示す. S式で表現したプログラムは付録1に示す.

式の読みやすさのために⟦,⟧,⟦,⟧,を[,]と同じ意味に用いることにする. 以下説明のない関数(述語)はすべてLISPシステムの基本関数又は組込み関数である. ただしeq1は, 2つのS式が等しいことを判定する述語で我々が定義したものである.

(1) handle　任意のlinear termに対してそのhandleを対応させる関数

```
handle[x]=[atom[x]→x;T→car[x]]
```

例2·1 handle[A]=A , handle[(A.(B.+.C))]=A

(2) drv (derivative)　　linear term に対してその derivative を対応させる関数

drv[x]=[atom[x]→list[0;*];T→cdr[x]]

例2.2　drv[A]=(0∘*) , drv[(A.(B∘+∘C))]=(B.+∘C)

(3) order　　linear part $(\alpha_n \circ \cdots \circ \alpha_1)$ に対して α_n の handle の alphabetical order を対応させる関数. linear part = ( ) のときは order[( )]=27 とする.

order[x]=[null[x]→27;T→cdr[assoc[handle[car[x]];((A.1) ...(Z.26))]]]

例2.3　order[(B)]=2 , order[((A.(B∘*))∘B∘C)]=1

(4) member　　x がリスト y の要素であるか否かを判定する述語

member[x;y]=[null[y]→NIL;eql[x;car[y]]→T;T→member[x;cdr[y]]]

例2.4　member[A;(B∘A)]=T , member[(A);(B∘A)]=NIL

(5) form　　S式をその形によって分類する関数で次をみたす.

$$\mathrm{form}[x]=\begin{cases} 0 & : x=0 \\ 1 & : \exists a[x=a \wedge a\varepsilon\Sigma] \\ 2 & : \exists\alpha[x=(\alpha\circ *)] \\ 3 & : \exists\alpha,\exists\beta[x=(\alpha\circ+\circ\beta)] \\ 4 & : \exists\alpha,\exists\beta[x=(\alpha.\beta)\wedge\sim\exists\gamma[\beta=(*.\gamma)\vee\beta=(+.\gamma)]] \\ \mathrm{NIL} & : \text{otherwise} \end{cases}$$

form[x]=[atom[x]→[eq[x;0]→0;member[x;(A∘...∘Z)]→1;T→NIL];
　　atom[cdr[x]]→4;
　　eql[cadr[x];*]→[null[cddr[x]]→2;T→NIL];
　　eql[cadr[x];+]→[atom[cddr[x]]→NIL;T→[null[cdddr[x]]→3;T→NIL]];
　　T → 4]

例2.5　form[*]=NIL , form[(A)]=4 , form[((A∘+∘B)∘*)]=2

(6) elementary　　文 $=\overline{\alpha}$ に対して α が $(\beta+\gamma)$ なる形をしていないことを示す述語

elementary[x]=not[eq[form[x];3]]

例 2·6　elementary[(A␣+␣(B␣*))]=NIL , elementary[((A␣+␣B)␣*)]=T

(7) mergel

$$\text{mergel}[\overline{(\alpha_n+\ldots+(\alpha_2+\alpha_1)\ldots)};\bar{\beta}]=\begin{cases}\overline{(\alpha_n+\ldots+(\alpha_2+(\alpha_1+\beta))\ldots)} & :(\forall i)[1\le i\le n\to\alpha_i\neq\beta]\\ \overline{(\alpha_n+\ldots+(\alpha_2+\alpha_1)\ldots)} & :(\exists i)[1\le i\le n\ \alpha_i\equiv\beta]\end{cases}$$

mergel[x;y]=[elementary[x]→[eql[x;y]→x;T→list[x;+;y]];

T→[eql[car[x];y]→x;T→list[car[x];+;mergel[caddr[x];y]]]]

例 2·7　mergel[$\overline{(B+C)}$;$\overline{(A+(B+C))}$]=$\overline{(B+(C+(A+(B+C))))}$

mergel[$\overline{(A+((A^*)+B))}$;$\overline{(A^*)}$]=$\overline{(A+((A^*)+B))}$

(8) merge　$x=\overline{(\alpha_m+\ldots+(\alpha_2+\alpha_1)\ldots)}$, $y=\overline{(\beta_n+\ldots+(\beta_2+\beta_1)\ldots)}$ $m,n\ge1$ に対して

$\overline{(\alpha_m+\ldots+(\alpha_1+(\beta_{i_k}+\ldots+(\beta_{i_2}+\beta_{i_1})\ldots)}$ $i_k\ge0, i_k>\ldots i_2>i_1$ を対応させる関数.

これは次の(9)の中で用いられ, compare の停止性に対して本質的な役わりをもっている. これについては次節で述べる.

merge[x;y]=[elementary[y]→mergel[x;y];

T→merge[mergel[x;car[y]];caddr[y]]]

例 2·8　merge[$\overline{(B+C)}$;$\overline{(A+(B+C))}$]=$\overline{(B+(C+A))}$

(9) addl (add linear parts)　2つのそれぞれ sort されている linear part に対して1つの sort された linear part を対応させる関数.

addl[x;y]=[lessp[order[x];order[y]]→cons[car[x];addl[cdr[x];y]];

greaterp[order[x];order[y]]→cons[car[y];addl[x;cdr[y]]];

T→[null[x]→NIL;T→cons[cons[handle[car[x]];merge[drv[car[x]];

drv[car[y]]]];addl[cdr[x];cdr[y]]]]]

例 2·9　addl[(A);()]=(A)

addl[(A␣(C.(A␣+␣B)));((B.A)␣(C.((A␣*)␣+␣B)))]

=(A␣(B.A)␣(C.(A␣+␣(B␣+␣(A␣*)))))

(10) addc (add constant parts)　　2つの constant parts に対して1つの constant part を対応させる関数.

addc[x;y]=[eq[x;0]→y;T→1]

addc, addl は nre α, $\tilde{\alpha}=((\overline{\alpha_l}\sqcup\cdots\sqcup\overline{\alpha_1})\cdot\tilde{\alpha_0})$ と nre β, $\tilde{\beta}=((\overline{\beta_m}\sqcup\cdots\sqcup\overline{\beta_1})\cdot\tilde{\beta_0})$ から $(\alpha+\beta)$ と同値な nre γ, $\tilde{\gamma}=((\overline{\gamma_n}\sqcup\cdots\sqcup\overline{\gamma_1})\cdot\tilde{\gamma_0})$, $n\leq l+m$, $\tilde{\gamma_0}=$addc$[\tilde{\alpha_0};\tilde{\beta_0}]$ をもとめるのに用いられる.

(11) dtbt (distribute)　　1つの sort されている linear part $(\overline{\alpha_m}\sqcup\cdots\sqcup\overline{\alpha_1})$ と re $\overline{\beta}$ に対して sort された linear part $(\overline{\gamma_n}\sqcup\cdots\sqcup\overline{\gamma_1})$ を対応させる関数で $((\alpha_m+\cdots+(\alpha_2+\alpha_1)\cdots)\cdot\beta)$ なる形の re からこれと同値な re $(\gamma_n+\cdots+(\gamma_2+\gamma_1)\cdots)$,

$$\gamma_i\equiv\begin{cases}(a_i\cdot\beta) & : \alpha_i\equiv a_i\\ (a_i\cdot(\eta_i\cdot\beta)) & : \alpha_i\equiv(a_i\cdot\eta_i)\end{cases}$$ をもとめるのに用いられる.

dtbt[x;y]=[null[x]→NIL;

atom[car[x]]→cons[cons[car[x];y];dtbt[cdr[x];y]];

T→cons[cons[caar[x];cons[cdar[x];y]];dtbt[cdr[x];y]]]

例 2·10　dtbt[(A⊔(B.C));(A⊔*)]=((A.(A⊔*))⊔(B.(C.(A⊔*))))　で これは

((A+(B.C)).(A*)) より ((A.(A*))+(B.(C.(A*))))　をもとめることに相当する.

(12) nmlz　　標準化関数といい, 任意の α に対して $\vdash\alpha=\beta$, β は nre なる β を対応させる関数で compare の中で用いられる主要な関数である. nmlz[x] を x の形で場合わけして定義すると次のようになる.

0) x=0 すなわち form[x]=0　のとき

nmlz[0]=(().0)

1) x=a, a∈Σ すなわち form[x]=1　のとき

nmlz[a]=((a).0)

2) x=$\overline{(\alpha^*)}$ すなわち form[x]=2　のとき

nmlz[$\overline{(\alpha^*)}$]=cons[dtbt[car[nmlz[$\overline{\alpha}$]];$\overline{(\alpha^*)}$];1]　……(i)

$nml_z[\bar{\alpha}] = \widetilde{\mu}$, $\vdash \alpha = \mu$, μ は nre で $\mu \equiv (\mu_m + \cdots + (\mu_1 + \mu_0)\cdots)$ $m \geq 0$ なる仮定のもとで (i) 式の右辺は, $\vdash (\alpha *) = \nu$, ν は nre , なる $\widetilde{\nu}$ であることを示そう.

$\mu_0 \equiv 0$ なら A9 により $\mu_0 \equiv (0 *)$ なら A11 により

$$\begin{aligned}
\vdash (\alpha *) &= ((\mu_m + \cdots + (\mu_2 + \mu_1)\cdots) *) \\
&= (((\mu_m + \cdots + (\mu_2 + \mu_1)\cdots) \cdot ((\mu_m + \cdots + (\mu_2 + \mu_1)\cdots) *)) + (0 *)) \quad [\because A10] \\
&= ((((\mu_m + \cdots + (\mu_2 + \mu_1)\cdots) \cdot (\alpha *)) + (0 *) \\
&= (\nu_m + \cdots + (\nu_1 + (0 *))\cdots)
\end{aligned}$$

ただし $\nu_i \equiv \begin{cases} (a_i \cdot (\alpha *)) & : \mu_i \equiv a_i \text{ のとき} \\ (a_i \cdot (\beta_i \cdot (\alpha *))) & : \mu_i \equiv (a_i \cdot \beta_i) \text{ のとき} \end{cases}$

一方 $car[nml_z[\bar{\alpha}]] = (\bar{\mu}_m \cup \cdots \cup \bar{\mu}_1)$, したがって dtbt の機能により (i) 式の右辺は $((\bar{\nu}_m \cup \cdots \cup \bar{\nu}_1) \cdot 1)$ でこれは $\widetilde{\nu}$ にほかならない.

3) $x = \overline{(\alpha+\beta)}$ すなわち form[x]=3 のとき

nmlz[$\overline{(\alpha+\beta)}$]=cons[addl[car[nmlz[$\bar{\alpha}$]];car[nmlz[$\bar{\beta}$]]];
addc[cdr[nmlz[$\bar{\alpha}$]];cdr[nmlz[$\bar{\beta}$]]]] ------ (ii)

この場合には $nml_z[\bar{\alpha}]$, $nml_z[\bar{\beta}]$ をもとめて, addl を用いて2つの linear parts より $\overline{(\alpha+\beta)}$ の linear part をつくり, addc を用いて2つの constant parts より $\overline{(\alpha+\beta)}$ の constant part をつくり, それらを cons でつなげる. $nml_z[\bar{\alpha}] = \widetilde{\mu}$, $nml_z[\bar{\beta}] = \widetilde{\nu}$, $\vdash \alpha = \mu$, $\vdash \beta = \nu$, μ, ν は nre, なる仮定のもとで (ii) 式の右辺は, $\vdash (\alpha+\beta) = \nu$, ν は nre , なる $\widetilde{\nu}$ であることは addl, addc の機能より明らか.

4) $x = \overline{(\alpha.\beta)}$ すなわち form[x]=4 のとき. このときは cdr[nmlz[$\bar{\alpha}$]] すなわち $nml_z[\bar{\alpha}]$ の constant part の 0, 1 によって更に場合わけする.

4.0) cdr[nmlz[$\bar{\alpha}$]]=0 のとき

nmlz[$\overline{(\alpha.\beta)}$]=cons[dtbt[car[nmlz[$\alpha$]];$\bar{\beta}$];0]

4.1) cdr[nmlz[$\bar{\alpha}$]]=1 のとき

nmlz[$\overline{(\alpha.\beta)}$]=cons[addl[dtbt[car[nmlz[$\bar{\alpha}$]];$\bar{\beta}$];car[nmlz[$\bar{\beta}$]]];cdr[nmlz[$\bar{\beta}$]]] ---- (iii)

$nml_z[\bar{\alpha}] = \widetilde{\mu}$, $\vdash \alpha = \mu$, $\mu \equiv (\mu_m + \cdots + (\mu_1 + (0 *))\cdots)$ $nml_z[\bar{\beta}] = \widetilde{\nu}$, $\vdash \beta = \nu$ $\nu \equiv (\nu_n + \cdots + (\nu_1 + \nu_0)\cdots)$ なる仮定のもとで (iii) 式の右辺は, $\vdash (\alpha \cdot \beta) = \nu$, ν は nre , なる $\widetilde{\nu}$ であることは addl, dtbt の機能と次の事実よりわかる.

$$\begin{aligned}
\vdash (\alpha \cdot \beta) &= ((\mu_m + \cdots + (\mu_1 + (0 *))\cdots) \cdot \beta) \\
&= (((\mu_m + \cdots + (\mu_2 + \mu_1)\cdots) \cdot \beta) + ((\rho *) \cdot \beta)) \\
&= ((((\mu_m + \cdots + (\mu_2 + \mu_1)\cdots) \cdot \beta) + (\nu_n + \cdots + (\nu_1 + \nu_0)\cdots))
\end{aligned}$$

実際のプログラムは効率の点からLISPのprog形式を用いて次のように定義する. また form[x]=NIL のときは nmlz[x]=NIL とする.

```
nmlz[x]=prog[(alpha beta frm);
        [null[setq[frm;form[x]]]→return[NIL]];
        [eq[frm;0]→return[cons[NIL;0]]];
        [eq[frm;1]→return[cons[list[x];0]]];
        [null[setq[alpha;nmlz[car[x]]]]→return[NIL]];
        [eq[frm;2]→return[cons[dtbt[car[alpha];x];1]]];
        [eq[frm;3]→[null[setq[beta;nmlz[caddr[x]]]]→return[NIL];
          T→return[cons[add1[car[alpha];car[beta]];addc[cdr[alpha];cdr[beta]]]]]];
        [eq[cdr[alpha];0]→return[cons[dtbt[car[alpha];cdr[x]];0]]];
        [null[setq[beta;nmlz[cdr[x]]]]→return[NIL]];
        return[cons[add1[dtbt[car[alpha];cdr[x]];car[beta]];cdr[beta]]] ]
```

例は付録1で示す.

(13) compare　比較関数といいnmlzを利用して, 2つのre α, βに対してt α=β (すなわちαとβは同値) か否かを決定する関数である. はじめに次の2つの関数を定義する.

1) nconc1　リストxの最後にS式をつけ加えてできるリストを値とする関数で

nconc1[$(S_n \cdot \ldots \cdot S_1)$;S]=$(S_n \cdot \ldots \cdot S_1 \cdot S)$　$n \geq 0$ をみたす.

nconc1[x;y]=nconc[x;list[y]]

2) drvpair　2つのlinear part $(\bar{\alpha}_\ell \cdot \ldots \cdot \bar{\alpha}_1)$, $(\bar{\beta}_m \cdot \ldots \cdot \bar{\beta}_1)$ に対してderivativeのペアのリスト $(\bar{P}_n \cdot \ldots \cdot \bar{P}_1)$ n=max[l;m] を対応させる関数

例2.11　drvpair[$(A \cdot (B.\bar{\beta}) \cdot (C.\bar{\gamma}))$; $((A.\bar{\alpha}) \cdot (C.\bar{\gamma}))$]

=($((0 \cdot *).\bar{\alpha})$ $(\bar{\beta}.0)$ $(\bar{\gamma}.\bar{\gamma})$)

```
drvpair[x;y]=[lessp[order[x];order[y]]→cons[cons[drv[car[x];0];drvpair[cdr[x];y]];
              greaterp[order[x];order[y]]→cons[cons[0;drv[car[y]]];drvpair[x;cdr[y]]];
                null[x] → NIL;
                T→cons[cons[drv[car[x]];drv[car[y]]];drvpair[cdr[x];cdr[y]]]]
```

re α,β に対して compare は次のうごきをする.

(A) u1 を $((\overline{\alpha}\cdot\beta))$, u2 を ( ) として (B) へすすむ.

(B) nr1 を nmlz[caar[u1]], nr2 を nmlz[cdar[u1]] として, nr1 と nr2 の constant part が等しいか否かをみる. 等しくなければ $\vdash\alpha=\beta$ であり処理を終える. 等しければ u2 を nconc1[u2;car[u1]], u1 を cdr[u1], pairlist を drvpair[car[nr1];car[nr2]] として (C) へすすむ.

(C) pairlist が ( ) なら (D) へすすむ. ( ) でなければ apair を car[pairlist] とし, pairlist を cdr[pairlist] とし, apair と等しい要素が u1 又は u2 の中にあれば (C) へすすむ, なければ u1 を nconc1[u1;apair] として (C) へすすむ.

(D) u1 が ( ) ならば $\vdash\alpha=\beta$ であり, 処理を終える. ( ) でなければ (B) へすすむ.

compare に用いたアルゴリズムは Salomaa [1] に従ったが次の点が異なる.

1) Salomaa では α,β についてそれぞれ方程式系をつくる. そのつくり方は re α に対してその Sub re のなかでもっとも小さなものについて方程式系をつくり順次より大きな Sub re に対する方程式系をつくり最終的に α に対する方程式系をつくる.
そして2つの方程式系を比較して $\alpha=\beta$ の validity に対する予告があるか否かをみる.
一方 compare では上に述べたように, α,β のペアに対してそれらと同値な nrc のペアをもとめ, その constant part を比較して validity に対する予告がなければ derivative のペア達に対してまだ nre のペアをもとめていないものについて順次 nre のペアをもとめ constant part を比較していく. したがって compare では α と β の方程式系をつくる (間接的であるが) ことと2つの方程式系を比較することを全く並行して行う.

2) Salomaa では各方程式の右辺にはアルファベットのすべての元が handle として現われる. compare においてはアルファベットの元のうちで必要なものだけが handle として現われる.

例 2.12　　$\alpha \equiv ((A\cdot(B*))+C)$ に対して Salomaa では

$\vdash \alpha = ((A\cdot(B*))+((B\cdot 0)+((C\cdot(0*))+((D\cdot 0)+\cdots+((Z\cdot 0)+0)\cdots)$

を方程式とみなし compare では

$\vdash \alpha = ((A\cdot(B*))+(C+0))$ を方程式とみなす.

$$\text{compare}[x;y]=\begin{cases}((\bar{\alpha}.\bar{\beta})\ (\bar{\alpha}_1.\bar{\beta}_1)\ \ldots\ (\bar{\alpha}_n.\bar{\beta}_m)) & : x=\bar{\alpha}, y=\bar{\beta},\ \vdash\alpha=\beta \\ \text{NOT-EQUIVALENT} & : x=\bar{\alpha}, y=\bar{\beta},\ \nvdash\alpha=\beta \\ \text{SYNTAX-ERROR} & : X\text{は}Y\text{が re の表現でない}\end{cases}$$

ただし $(\bar{\alpha}_1.\bar{\beta}_1),\ldots,(\bar{\alpha}_n.\bar{\beta}_m)$ derivative のペア

となるように compare を prog 形式で定義すると次のようになる.

```
compare[x;y]=prog[(u1 u2 nr1 nr2 pairlist apair);
            setq[u1;list[cons[x;y]]];
      COMP1 [or[null[setq[nr1;nmlz[caar[u1]]]]
               ;null[setq[nr2;nmlz[cdar[u1]]]]]→return[SYNTAX-ERROR]];
            [not[eq[cdr[nr1];cdr[nr2]]]→return[NOT-EQUIVALENT]];
            setq[u2;nconc1[u2;car[u1]]];
            setq[u1;car[u1]];
            setq[pairlist;drvpair[car[nr1];car[nr2]]];
      COMP2 [null[pairlist]→go[COMP3]];
            setq[apair;car[pairlist]];
            setq[pairlist;cdr[pairlist]];
            [or[member[apair;u1];member[apair;u2]]→go[COMP2]];
            setq[u1;nconc1[u1;apair]];
            go[COMP2];
      COMP3 [null[u1]→return[u2]];
            go[COMP1]    ]
```

3. compare の停止性

まずプログラム compare の中で用いられる各サブプログラムの停止性が問題になる。しかしこれらのほとんどのものの停止性は自明である。nml2はやや複雑な形をしているが、その停止性は、正当性も含めて、re α 中に現われる演算子 $(\cdot, +, *)$ の個数に関する帰納法によって証明される。その証明の一部は nml2 の説明のところで示した。

次に compare の停止性は re α の相異なる derivative の個数が有限（α に対する方程式系が存在する）か否かによる。derivative が有限個になることの証明は Salomaa [1] の Lemma 4 における証明と並行して、やはり α 中の演算子の個数に関する帰納法によってできるが次のことに注意しなくてはならない。

derivative $(\alpha_1 + (\alpha_2 + \cdots + (\alpha_{m-1} + \alpha_m)\cdots)$ $n \geq 1$ において $\alpha_1, \cdots, \alpha_m$ の中に相等しいものがあるか否かを調べて derivative をそれと同値な $(\alpha_{i_1} + \cdots + (\alpha_{i_{m-1}} + \alpha_{i_m})\cdots)$, $i_m \geq 1$, $i_1 < i_2 < \cdots < i_m \leq n$, $\alpha_{i_1}, \cdots, \alpha_{i_m}$ は相異なる に変形しなくてはならない。（merge はこれを行なう）これは $(\alpha\cdot\beta)$, (α^*) なる形の re の相異なる derivative の個数を有限に抑える上で必要である。merge を単に

$$\mathrm{merge}[x;y] = \mathrm{list}[x;+;y]$$

とすると上で述べたことを行なわなくなるので、$((A+((A\cdot B)+(B\cdot A)))^*)$, $((A^*)\cdot(A^*))$ 等の derivative の個数は有限にならなくなる。例えば

$((A*)\cdot(A*))$ に対しては $((A*)\cdot(A*))$, $(((A*)\cdot(A*))+(A*))$, $(((A*)\cdot(A*))+((A*)+(A*)))$, $\cdots$ と無限に derivative が現われる.

4. 有限オートマトンの生成

この節では re α に対して $|\alpha|$ を受理する有限オートマトンを生成するプログラムについて簡単に述べる. プログラムは LISP との比較のために SNOBOL 3 で表現した[5]. 全プログラムと実行例は付録2 に示す.

re α に対して α の方程式系; $\alpha_i = \beta_i$, $i = 1, \cdots, n$, $\alpha_1 \equiv \alpha$, 各 β_i は nre で β_i の各 derivative γ_{ij} に対して $\gamma_{ij} \equiv \alpha_k$, $1 \le k \le n$ なる k が存在する.; は $|\alpha|$ を受理する有限オートマトンの構造を完全に表現している. この事実にもとずいてプログラムを SNOBOL 3 で表現する. SNOBOL 3 と LISP の主なちがいはその扱うデータ構造である. LISP では binary tree 構造をもつデータを扱うのに対して SNOBOL 3 では記号列そのものをデータとして扱う. recursive call は LISP と同様 SNOBOL 3 にも許されている.

我々はこの節では, re に対して Salomaa のものと同じ定義を採用する. ($(\alpha*)$ でなく α^* を用いる.) re α に対して $\overline{\alpha}$ は α そのものとし, nre $\alpha \equiv (\alpha_n + \cdots + (\alpha_1 + \alpha_0)\cdots)$ に対して $\widetilde{\alpha}$ は次のものとする. $\widetilde{\alpha} = ((\alpha_n + \cdots + (\alpha_1 + NIL)\cdots) + \alpha_0)$ この様に定義しておくと LISP のときと全く同様にして SNOBOL 3 で nml 等が

表現できる．SNOBOL 3 で表現した nmlg の中で用いられるサブプログラムのプログラム名は LISP でのそれらと多少異なったので，機能別にそれらの対応を下に示す．

SNOBOL 3	LISP
left	car $\lfloor\{x \mid x=\overline{\alpha}\}$
right	cdr $\lfloor\{x \mid x=\overline{(\alpha\cdot\beta)}\}$ または caddr $\lfloor\{x \mid x=\overline{(\alpha+\beta)}\}$
oper	form $\lfloor\{x \mid x=\overline{(\alpha*)} \vee x=\overline{(\alpha\cdot\beta)} \vee x=\overline{(\alpha+\beta)}\}$
handle	handle
drv	drv
order	order
cona	merge1
con	merge
mrg	addl
addcnst	addc
dister	dtbt

この表で同じ行のプログラムは同じ機能である．

re α に対して |α| を受理する有限オートマトンを生成するプログラム am は nmlg を用いて α に対する方程式系をつくることを行う．出力する方程式系をみやすくする為に各方程式の derivative を Si (i は自然数) で置き換えたものをも出力するようにしてある．実行例として $\alpha\equiv(((B\cdot A)+(A^*\cdot(A\cdot B)))^*\cdot A^*)$ と

$\beta \equiv (A+((A\cdot B)+(B\cdot A)))^*$ について付録2で示した.

5. スタックを用いて標準化関数 nmlz をつくる方法

この節では recursive call を使わないで nmlz をつくる1例を簡単に述べる. ここでも re の定義は Salomaa のものと同じものとする.

re そのものが入力されるとし, nre $\alpha \equiv (\alpha_n + \cdots + (\alpha_1 + \alpha_0)\cdots)$ $n \geq 0$ に対する $\widetilde{\alpha}$ は次のようにする.

$$\widetilde{\alpha} = \widetilde{\alpha_n} + \cdots + \widetilde{\alpha_1} + \widetilde{\alpha_0}$$

$$\widetilde{\alpha_i} = \begin{cases} a & : \alpha_i \equiv a \text{ のとき} \\ a\beta & : \alpha_i \equiv (a\cdot\beta) \text{ のとき} \end{cases} \quad (1 \leq i \leq n)$$

$$\widetilde{\alpha_0} = \begin{cases} 0 & : \alpha_0 \equiv 0 \text{ のとき} \\ 1 & : \alpha_0 \equiv 0^* \text{ のとき} \end{cases}$$

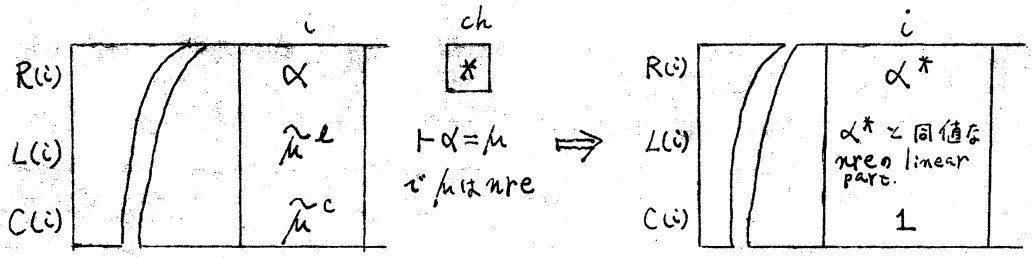
また $\widetilde{\alpha}^l = \widetilde{\alpha_n} + \cdots + \widetilde{\alpha_1} +$ を $\widetilde{\alpha}$ の linear part, $\widetilde{\alpha}^c = \widetilde{\alpha_0}$ を $\widetilde{\alpha}$ の constant part と言う. re α と同値な nre を逐次的にもとめるために3種のスタック R, L, C を用いる. R(i) には与えられた re α の sub re β が (か+か・) をしまう. L(i) には $\vdash \beta = \gamma$, γ は nre なる $\widetilde{\gamma}^l$, C(i) には $\widetilde{\gamma}^c$ をしまう.

例 5.1 $((B+A)\cdot C^*)$ を処理しているときのスタックの状態は次のとおりである.

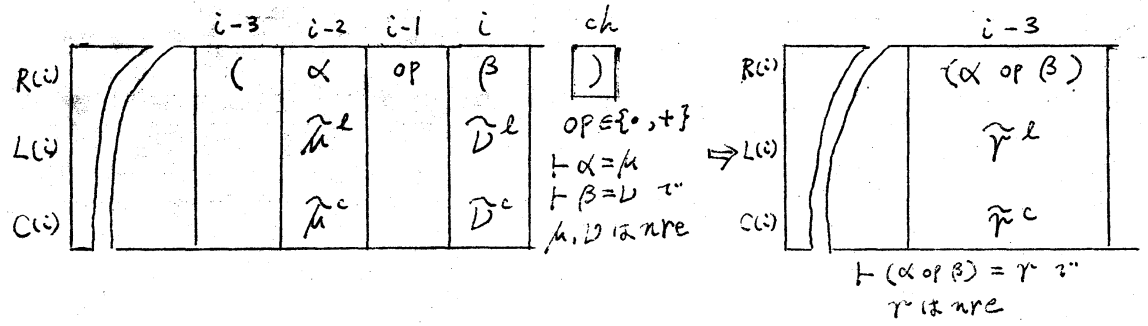
i	1	2	3	4	
R(i)	(	(B+A)	•	C	
L(i)		A+B+		C+	
C(i)		0		0	

以下にスタックのうごきを明確化するためにLISPメタ言語に類似な架空の言語でプログラムを示すが，条件式，go，return，:=はLISPのprog形式におけるそれらと同じ意味に使う．またaddl，addc，dtbtは第2節のそれらと同様の機能とする．oはstringのconcatenationを意味する．

プログラムにおいてNML24の次のステートメント以後のそれらはスタックの状態を次のようにかえる．



またNML25以後のステートメントでは次のようになる．



```
nmlz[x]=[  i:=0 ;
   NMLZ1   [xは空 → return[ERROR]];
           ch:= xの左端1字      ;
           x:= xより左端1字をのぞいたもの      ;
           i:= i+1  ;    R(i):=ch  ;
           [ch=( → go[NMLZ1]];
           [ch=0 → go[NMLZ2]];
           [ch は五つの元でない → return[ERROR]];
           L(i):=ch∘+      ;
   NMLZ2   C(i):=0  ;
   NMLZ3   [xは空 → [i=1 → return[L(1)∘C(1)];T→return[ERROR]];
           ch:= xの左端1字   ;
           x:= xより左端の1字をのぞいたもの    ;
           [ch≠. かつ ch≠+ →go[NMLZ4]];
           R(i):=ch  ;  go[NMLZ1];

   NMLZ4   [ch≠*→ go[NMLZ5]];
           L(i):=dtbt[L(i);R(i)];  C(i):=1; R(i):=R(i)∘*   ;  go[NMLZ3]  ;

   NMLZ5   [ch≠)または i< 4または R(i-3)≠(→return[ERROR]];
           [R(i-1)=+ →go[NMLZ7]];
           [R(i-1)≠. →return[ERROR]];
           L(i-3):=dtbt[L(i-2);R(i)];
           [C(i-2)=1→go[NMLZ8]];
           C(i-3):=0  ;
   NMLZ6   R(i-3):=(∘R(i-2)∘R(i-1)∘R(i)∘)  ;  i:=i-3  ;  go[NMLZ3];

   NMLZ7   L(i-3):=add1[L(1-2);L(i)];  C(i-3):=addc[C(i-2;C(i)];  go[NMLZ6];

   NMLZ8   L(i-3):=add1[L(i-3);L(i)];  C(i-3):=C(i);  go[NMLZ6]    ]
```

謝辞

LISPシステムに関してお世話になった通産省、電子技術総合研究所の田村浩一郎さんに感謝します。

文献

[1] Salomaa, A., Two Complete Axiom System for the Algebra of Regular Events, JACM, vol. 13, No. 1, PP. 158-169, 1966

[2] McCarthy, J., et al., LISP 1.5 Programmer's Manual, MIT Press, Cambridge, Mass., 1962

[3] 日立製作所, HITAC 8350 LISP 仕様書, 1972

[4] 日立製作所, HITAC 8350 LISP 操作仕様書, 1972

[5] Farber, D.J., et al., The SNOBOL 3 Programing Language, The Bell System Technical J., July-August, 1966

付録1


```
HITAC 8000 SERIES          ***EDOS***  *JOB CONTROL*          DATE 02/15/74  TIME 17:55:25

*JOB CLASS 8 *JOB NAME ZYS56   *JOB# 0560 *MEMORY 9/KB

RESERVED DEVICE C0

// VOC SYSUT3,,EDOS.SYSUT3A,EDOS11

// EXEC LISP
```

```
HITAC LISP                                        02/15/74        PAGE    1

*PACKET
C                                                                          REX00000
C  EQUIVALENCE PRUVER FOR REGULAR EXPRESSIONS                              REX00010
C                                                                          REX00020
 DEFINE( (                                                                 REX00030
C                                                                          REX00040
 (EQ1 (LAMBDA (X Y) (COND ((AND (ATOM X) (ATOM Y)) (EQ X Y))               REX00050
                          ((OR (ATOM X) (ATOM Y)) NIL)                     REX00060
                          ((EQ1 (CAR X) (CAR Y)) (EQ1 (CDR X) (CDR Y)))    REX00070
                          (T NIL))))                                       REX00080
C                                                                          REX00090
 (HANDLE (LAMBDA (X) (COND ((ATOM X) X) (T (CAR X)))))                     REX00100
C                                                                          REX00110
 (DRV (LAMBDA (X) (COND ((ATOM X) (LIST (QUOTE 0) (QUOTE *)))              REX00120
                        (T (CDR X)) )))                                   REX00130
C                                                                          REX00140
 (ORDER (LAMBDA  (X) (COND ((NULL X) (QUOTE 27)) (T (CDR (ASSOC (HANDLE   REX00150
       (CAR X)) (QUOTE ((A.1) (B.2) (C.3) (D.4) (E.5) (F.6) (G.7) (H.8)    REX00160
       (I.9) (J.10) (K.11) (L.12) (M.13) (N.14) (O.15) (P.16) (Q.17)       REX00170
       (R.18) (S.19) (T.20) (U.21) (V.22) (W.23) (X.24) (Y.25)             REX00180
       (Z.26)))))))))                                                      REX00190
C                                                                          REX00200
 (MEMBER (LAMBDA (X Y) (COND ((NULL Y) NIL) ((EQ1 X (CAR Y)) T)           REX00210
                             (T (MEMBER X (CDR Y))) )))                    REX00220
C                                                                          REX00230
 (FORM (LAMBDA (X) (COND ((ATOM X) (COND ((EQ X (QUOTE 0)) X)              REX00240
       ((MEMBER X (QUOTE (A B C D E F G H I J K L M N O P Q R S T U V      REX00250
       W X Y Z))) (QUOTE 1)) (T NIL))) ((ATOM (CDR X)) (QUOTE 4))         REX00260
       ((EQ1 (CADR X) (QUOTE *)) (COND ((NULL (CDDR X)) (QUOTE 2))        REX00270
       (T NIL))) ((EQ1 (CADR X) (QUOTE +)) (COND ((ATOM (CDDR X)) NIL)    REX00280
       (T (COND ((NULL (CDDDR X)) (QUOTE 3)) (T NIL)))))                  REX00290
       (T (QUOTE 4)) )))                                                  REX00300
C                                                                          REX00310
 (ELEMENTARY (LAMBDA (X) (NOT (EQ (FORM X) (QUOTE 3))) ))                  REX00320
C                                                                          REX00330
 (MERGE1 (LAMBDA (X Y) (COND ((ELEMENTARY X) (COND ((EQ1 X Y) X)           REX00340
       (T (LIST X (QUOTE +) Y)))) (T (COND ((EQ1 (CAR X) Y) X)            REX00350
       (T (LIST (CAR X) (QUOTE +) (MERGE1 (CADDR X) Y))))) )))            REX00360
C                                                                          REX00370
 (MERGE (LAMBDA (X Y) (COND ((ELEMENTARY Y) (MERGE1 X Y))                  REX00380
       (T (MERGE (MERGE1 X (CAR Y)) (CADDR Y))) )))                       REX00390
C                                                                          REX00400
 (ADDL (LAMBDA (X Y) (COND ((LESSP (ORDER X) (ORDER Y)) (CONS (CAR X)      REX00410
       (ADDL (CDR X) Y))) ((GREATERP (ORDER X) (ORDER Y)) (CONS (CAR Y)    REX00420
       (ADDL X (CDR Y)))) ((NULL X) NIL) (T (CONS (CONS (HANDLE            REX00430
       (CAR X)) (MERGE (DRV (CAR X)) (DRV (CAR Y)))) (ADDL (CDR X)         REX00440
       (CDR Y)))))))                                                      REX00450
C                                                                          REX00460
 (ADDC (LAMBDA (X Y) (COND ((EQ X (QUOTE 0)) Y) (T (QUOTE 1)) )))          REX00470
C                                                                          REX00480
 (DTBT (LAMBDA (X Y) (COND ((NULL X) NIL)                                  REX00490
       ((ATOM (CAR X)) (CONS (CONS (CAR X) Y) (DTBT (CDR X) Y)))           REX00500
       (T (CONS (CONS (CAAR X) (CONS (CDAR X) Y)) (DTBT (CDR X) Y)))))))  REX00510
C                                                                          REX00520
 (NMLZ (LAMBDA (X) (PROG (ALPHA BETA FRM)                                  REX00530
       (COND ((NULL (SETQ FRM (FORM X))) (RETURN NIL)) )                   REX00540
       (COND ((EQ FRM (QUOTE 0)) (RETURN (CONS NIL (QUOTE 0)))) )          REX00550
       (COND ((EQ FRM (QUOTE 1)) (RETURN (CONS (LIST X) (QUOTE 0)))) )     REX00560
```

```
        (COND ((NULL (SETQ ALPHA (NMLZ (CAR X)))) (RETURN NIL)) )          REX00570
        (COND ((EQ FRM (QUOTE 2)) (RETURN (CONS (DIST (CAR ALPHA) X)       REX00580
                                                (QUOTE 1)))) )             REX00590
        (COND ((EQ FRM (QUOTE 3))                                          REX00600
        (COND ((NULL (SETQ BETA (NMLZ (CADDR X)))) (RETURN NIL))           REX00610
              (T (RETURN (CONS (ADDL (CAR ALPHA) (CAR BETA))               REX00620
                               (ADDC (CDR ALPHA) (CDR BETA))))))) ))       REX00630
        (COND ((EQ (CDR ALPHA) (QUOTE 0))                                  REX00640
               (RETURN (CONS (DIST (CAR ALPHA) (CDR X)) (QUOTE 0))) ))     REX00650
        (COND ((NULL (SETQ BETA (NMLZ (CDR X)))) (RETURN NIL) ))           REX00660
        (RETURN (CONS (ADDL (DIST (CAR ALPHA) (CDR X)) (CAR BETA))         REX00670
                      (CDR BETA))) )))                                     REX00680
C                                                                          REX00690
 (NCONC1 (LAMBDA (X Y) (NCONC X (LIST Y)) ))                               REX00700
C                                                                          REX00710
 (DRVPAIR (LAMBDA (X Y) (COND ((LESSP (ORDER X) (ORDER Y))                 REX00720
        (CONS (CONS (DRV (CAR X)) (QUOTE 0)) (DRVPAIR (CDR X) Y)))        REX00730
        ((GREATERP (ORDER X) (ORDER Y)) (CONS (CONS (QUOTE 0)              REX00740
        (DRV (CAR Y))) (DRVPAIR X (CDR Y)))) ((NULL X) NIL)                REX00750
        (T (CONS (CONS (DRV (CAR X)) (DRV (CAR Y)))                        REX00760
        (DRVPAIR (CDR X) (CDR Y)))))))                                     REX00770
C                                                                          REX00780
 (COMPARE (LAMBDA (X Y) (PROG (U1 U2 NR1 NR2 PAIRLIST APAIR)               REX00790
        (SETQ U1 (LIST (CONS X Y)))                                        REX00800
 COMP1 (COND ((OR (NULL (SETQ NR1 (NMLZ (CAAR U1))))                       REX00810
                  (NULL (SETQ NR2 (NMLZ (CDAR U1)))) )                    REX00820
              (RETURN (QUOTE SYNTAX-ERROR)) ))                            REX00830
        (COND ((NOT (EQ (CDR NR1) (CDR NR2)))                              REX00840
               (RETURN (QUOTE NOT-EQUIVALENT)) ))                          REX00850
        (SETQ U2 (NCONC1 U2 (CAR U1)))                                     REX00860
        (SETQ U1 (CDR U1))                                                 REX00870
        (SETQ PAIRLIST (DRVPAIR (CAR NR1) (CAR NR2)))                      REX00880
 COMP2 (COND ((NULL PAIRLIST) (GO COMP3)))                                 REX00890
        (SETQ APAIR (CAR PAIRLIST))                                        REX00900
        (SETQ PAIRLIST (CDR PAIRLIST))                                     REX00910
        (COND ((OR (MEMBER APAIR U1) (MEMBER APAIR U2)) (GO COMP2)))       REX00920
        (SETQ U1 (NCONC1 U1 APAIR))  (GO COMP2)                            REX00930
 COMP3 (COND ((NULL U1) (RETURN U2)) (T (GO COMP1)) )   )))       ))       REX00940
C   EXAMPLES
 NMLZ(  (((B.C).D) + (((A.B) + C) + A))  )
 NMLZ( ((A + ((A.B) + (B.A))) *) )
 NMLZ( ((((B.A) + ((A *).(A.B))) *).(A *)) )
 COMPARE(  A   (A + ((((B.C).D).E).0))   )
 COMPARE(  (A *)   (((((((A *) *) *) *) *) *) *)  )
 COMPARE(  (((A.B) *).A)  (A.((B.A) *))  )
 COMPARE(  ((A + B) *)  ((A *).((B.(A *)) *))  )
 COMPARE(  (((A.(B *)).A) *)  ((((A.(B *)).A) *).(A.(B *)))  )
 COMPARE(  (((A + (B.A)) *).C)  ((((A *).(B.A)) *).((A *).C))  )
 COMPARE(  ((A + ((A.B) + (B.A))) *)
           ((((B.A) + ((A *).(A.B))) *).(A *))  )
 COMPARE(  (((B + (C.((A *).B))) *).((C.((A *).C)) + A))
           ((((((B *).C).(A *)).B) *).(((B *).((C.(A *)).C)) + ((B *)
                                                .A)))  )
 COMPARE(    (F.(((B.(G.(((C.G) *).D))) *).(A.G)))
          (F.((((B.(G.D)) + (B.(G.(C.(G.(((C.G) *).D)))))) *).(A.G)))  )
 STOP

END OF SOURCE LISTING.
```

```
FUNCTION EVALQUOTE HAS BEEN ENTERED.ARGUMENTS...
DEFINE
(((EQ1 (LAMBDA (X Y) (COND ((AND (ATOM X) (ATOM Y)) (EQ X Y)) ((OR (ATOM X) (ATOM Y)) NIL) ((EQ1 (CAR X) (CAR Y)) (EQ1 (CDR X) (
CDR Y))) (T NIL)))) (HANDLE (LAMBDA (X) (COND ((ATOM X) X) (T (CAR X))))) (DRV (LAMBDA (X) (COND ((ATOM X) (LIST (QUOTE  0) (QUOTE
 *))) (T (CDR X))))) (ORDER (LAMBDA (X) (COND ((NULL X) (QUOTE  27)) (T (CDR (ASSOC (HANDLE (CAR X)) (QUOTE ((A. 1) (B. 2) (C. 3)
(D. 4) (E. 5) (F. 6) (G. 7) (H. 8) (I. 9) (J. 10) (K. 11) (L. 12) (M. 13) (N. 14) (O. 15) (P. 16) (Q. 17) (R. 18) (S. 19) (T. 20)
(U. 21) (V. 22) (W. 23) (X. 24) (Y. 25) (Z. 26))))))))) (MEMBER (LAMBDA (X Y) (COND ((NULL Y) NIL) ((EQ1 X (CAR Y)) T) (T (MEMBER
X (CDR Y)))))) (FORM (LAMBDA (X) (COND ((ATOM X) (COND ((EQ X (QUOTE  0)) X) ((MEMBER X (QUOTE (A B C D E F G H I J K L M N O P Q
R S T U V W X Y Z))) (QUOTE  1)) (T NIL))) ((ATOM (CDR X)) (QUOTE  4)) ((EQ1 (CADR X) (QUOTE *)) (COND ((NULL (CDDR X)) (QUOTE  2)
) (T NIL))) ((EQ1 (CADR X) (QUOTE +)) (COND ((ATOM (CDDR X)) NIL) (T (COND ((NULL (CDDDR X)) (QUOTE  3)) (T NIL))))) (T (QUOTE  4)
)))) (ELEMENTARY     (LAMBDA (X) (NOT (EQ (FORM X) (QUOTE  3))))) (MERGE1 (LAMBDA (X Y) (COND ((ELEMENTARY     X) (COND ((EQ1 X Y)
 X) (T (LIST X (QUOTE +) Y)))) (T (COND ((EQ1 (CAR X) Y) X) (T (LIST (CAR X) (QUOTE +) (MERGE1 (CADDR X) Y))))))) (MERGE (LAMBDA
(X Y) (COND ((ELEMENTARY     Y) (MERGE1 X Y)) (T (MERGE (MERGE1 X (CAR Y)) (CADDR Y)))))) (ADDL (LAMBDA (X Y) (COND ((LESSP (ORDER
 X) (ORDER Y)) (CONS (CAR X) (ADDL (CDR X) Y))) ((GREATERP (ORDER X) (ORDER Y)) (CONS (CAR Y) (ADDL X (CDR Y)))) ((NULL X) NIL) (T
 (CONS (CONS (HANDLE (CAR X)) (MERGE (DRV (CAR X)) (DRV (CAR Y)))) (ADDL (CDR X) (CDR Y))))))) (ADDC (LAMBDA (X Y) (COND ((EQ X (
QUOTE  0)) Y) (T (QUOTE  1))))) (DTBT (LAMBDA (X Y) (COND ((NULL X) NIL) ((ATOM (CAR X)) (CONS (CONS (CAR X) Y) (DTBT (CDR X) Y)))
 (T (CONS (CONS (CAAR X) (CONS (CDAR X) Y)) (DTBT (CDR X) Y)))))) (NMLZ (LAMBDA (X) (PROG (ALPHA BETA FRM) (COND ((NULL (SETQ FRM
(FORM X))) (RETURN NIL))) (COND ((EQ FRM (QUOTE  0)) (RETURN (CONS NIL (QUOTE  0))))) (COND ((EQ FRM (QUOTE  1)) (RETURN (CONS (
LIST X) (QUOTE  0))))) (COND ((NULL (SETQ ALPHA (NMLZ (CAR X)))) (RETURN NIL))) (COND ((EQ FRM (QUOTE  2)) (RETURN (CONS (DTBT (
CAR ALPHA) X) (QUOTE  1))))) (COND ((EQ FRM (QUOTE  3)) (COND ((NULL (SETQ BETA (NMLZ (CADDR X)))) (RETURN NIL)) (T (RETURN (CONS
(ADDL (CAR ALPHA) (CAR BETA)) (ADDC (CDR ALPHA) (CDR BETA))))))))) (COND ((EQ (CDR ALPHA) (QUOTE  0)) (RETURN (CONS (DTBT (CAR
ALPHA) (CDR X)) (QUOTE  0)))))) (COND ((NULL (SETQ BETA (NMLZ (CDR X)))) (RETURN NIL))) (RETURN (CONS (ADDL (DTBT (CAR ALPHA) (CDR
X)) (CAR BETA)) (CDR BETA)))))) (NCONC1 (LAMBDA (X Y) (NCONC X (LIST Y)))) (DRVPAIR (LAMBDA (X Y) (COND ((LESSP (ORDER X) (ORDER Y
)) (CONS (CONS (DRV (CAR X)) (QUOTE  0)) (DRVPAIR (CDR X) Y))) ((GREATERP (ORDER X) (ORDER Y)) (CONS (CONS (QUOTE  0) (DRV (CAR Y)
)) (DRVPAIR X (CDR Y)))) ((NULL X) NIL) (T (CONS (CONS (DRV (CAR X)) (DRV (CAR Y))) (DRVPAIR (CDR X) (CDR Y))))))) (COMPARE (
LAMBDA (X Y) (PROG (U1 U2 NR1 NR2 PAIRLIST APAIR) (SETQ U1 (LIST (CONS X Y))) COMP1 (COND ((OR (NULL (SETQ NR1 (NMLZ (CAAR U1))))
(NULL (SETQ NR2 (NMLZ (CDAR U1))))) (RETURN (QUOTE SYNTAX-ERROR)))) (COND ((NOT (EQ (CDR NR1) (CDR NR2))) (RETURN (QUOTE
NOT-EQUIVALENT)))) (SETQ U2 (NCONC1 U2 (CAR U1))) (SETQ U1 (CDR U1)) (SETQ PAIRLIST (DRVPAIR (CAR NR1) (CAR NR2))) COMP2 (COND ((
NULL PAIRLIST) (GO COMP3))) (SETQ APAIR (CAR PAIRLIST)) (SETQ PAIRLIST (CDR PAIRLIST)) (COND ((OR (MEMBER APAIR U1) (MEMBER APAIR
U2)) (GO COMP2))) (SETQ U1 (NCONC1 U1 APAIR)) (GO COMP2) COMP3 (COND ((NULL U1) (RETURN U2)) (T (GO COMP1)))))))))

END OF EVALQUOTE.VALUE IS...
(EQ1 HANDLE DRV ORDER MEMBER FORM ELEMENTARY     MERGE1 MERGE ADDL ADDC DTBT NMLZ NCONC1 DRVPAIR COMPARE)

FUNCTION EVALQUOTE HAS BEEN ENTERED.ARGUMENTS...
NMLZ
((((B.C).D) + (((A.B) + C) + A)))

END OF EVALQUOTE.VALUE IS...
(((A B + ( 0 *)) (B C.D) C). 0)

FUNCTION EVALQUOTE HAS BEEN ENTERED.ARGUMENTS...
NMLZ
(((A + ((A.B) + (B.A))) *))

END OF EVALQUOTE.VALUE IS...
(((A (( 0 *) + B) (A + ((A.B) + (B.A))) *) (B A (A + ((A.B) + (B.A))) *)). 1)

FUNCTION EVALQUOTE HAS BEEN ENTERED.ARGUMENTS...
NMLZ
(((((B.A) + ((A *) A.B)) *) A *))

END OF EVALQUOTE.VALUE IS...
(((A (((((A *) A.B) + B) ((B.A) + ((A *) A.B)) *) A *) + (A *)) (B (A ((B.A) + ((A *) A.B)) *) A *)). 1)

FUNCTION EVALQUOTE HAS BEEN ENTERED.ARGUMENTS...
COMPARE
(A (A + ((((B.C).D).E). 0)))
```

```
END OF EVALQUOTE.VALUE IS..
((A A + ((((B.C).D).E). 0)) (( 0 *)  0 *) ( 0 ((C.D).E). 0) ( 0 (D.E). 0) ( 0 E. 0) ( 0. 0))

FUNCTION EVALQUOTE HAS BEEN ENTERED.ARGUMENTS..
COMPARE
((A *) (((((((A *) *) *) *) *) *) *))

END OF EVALQUOTE.VALUE IS..
(((A *) ((((((A *) *) *) *) *) *) *) ((A *) ((((((A *) (A *) *) ((A *) *) *) (((A *) *) *) *) ((((A *) *) *) *) *) ((((((A *) *) *) *) *) *) *))

FUNCTION EVALQUOTE HAS BEEN ENTERED,ARGUMENTS..
COMPARE
((((A,B) *),A) (A (B,A) *))

END OF EVALQUOTE.VALUE IS..
(((((A.B) *),A) A (B,A) *) ((((B (A,B) *),A) + ( 0 *)) (B,A) *))

FUNCTION EVALQUOTE HAS BEEN ENTERED,ARGUMENTS..
COMPARE
(((A + B) *) ((A *) (B A *) *))

END OF EVALQUOTE.VALUE IS..
((((A + B) *) (A *) (B A *) *))

FUNCTION EVALQUOTE HAS BEEN ENTERED.ARGUMENTS..
COMPARE
((((A B *),A) *) ((((A B *),A) *) A B *))

END OF EVALQUOTE.VALUE IS..
NOT-EQUIVALENT

FUNCTION EVALQUOTE HAS BEEN ENTERED.ARGUMENTS..
COMPARE
((((A + (B,A)) *),C) (((A *) B,A) *) (A *),C))

END OF EVALQUOTE.VALUE IS..
(((((A + (B,A)) *),C) (((A *) B,A) *) (A *),C) ((((A + (B,A)) *),C) ((((A *) B,A) ((A *) B,A) *) (A *),C) + ((A *),C)) (((A (A + (
B,A)) *),C) (A ((A *) B,A) *) (A *),C) (( 0 *)  0 *))

FUNCTION EVALQUOTE HAS BEEN ENTERED.ARGUMENTS..
COMPARE
(((A + ((A,B) + (B,A))) *) ((((B,A) + ((A *) A,B)) *) A *))

END OF EVALQUOTE.VALUE IS...
((((A + ((A,B) + (B,A))) *) (((B,A) + ((A *) A,B)) *) A *) ((((  0 *) + B) (A + ((A,B) + (B,A))) *) (((((A *) A,B) + B) ((B,A) + ((
A *) A,B)) *) A *) + (A *)) ((A (A + ((A,B) + (B,A))) *) (A ((B,A) + ((A *) A,B)) *) A *) ((((A + ((A,B) + (B,A))) *) + (A (A + ((
A,B) + (B,A))) *)) (((B,A) + ((A *) A,B)) *) A *) (((((  0 *) + B) (A + ((A,B) + (B,A))) *) + ((A + ((A,B) + (B,A))) *)) (((((A *)
A,B) + B) ((B,A) + ((A *) A,B)) *) A *) + (A *)))

FUNCTION EVALQUOTE HAS BEEN ENTERED.ARGUMENTS...
COMPARE
((((B + (C (A *),B)) *) (C (A *),C) + A) ((((((B *),C) A *),B) *) ((B *) (C A *),C) + ((B *),A)))

END OF EVALQUOTE.VALUE IS...
(((((B + (C (A *),B)) *) (C (A *),C) + A) (((((B *),C) A *),B) *) ((B *) (C A *),C) + ((B *),A)) (( 0 *)  0 *) ((((B + (C (A *),B)
) *) (C (A *),C) + A) ((((((B *),C) A *),B) ((((B *),C) A *),B) *) ((B *) (C A *),C) + ((B *),A)) + (((B *) (C A *),C) + ((B *),A)
)) ((((((A *),B) (B + (C (A *),B)) *) (C (A *),C) + A) + ((A *),C)) ((((A *),B) ((((B *),C) A *),B) *) ((B *) (C A *),C) + ((B *).
```

```
 HITAC LISP                                    02/15/74      PAGE    5

A)) + ((A *),C)))

FUNCTION EVALQUOTE HAS BEEN ENTERED.ARGUMENTS...
COMPARE
((F ((B G ((C,G) *),D) *) A,G) (F (((B G,D) + (B G C G ((C,G) *),D)) *) A,G))

END OF EVALQUOTE,VALUE IS...
(((F ((B G ((C,G) *),D) *) A,G) F (((B G,D) + (B G C G ((C,G) *),D)) *) A,G) ((((B G ((C,G) *),D) *) A,G) (((B G,D) + (B G C G ((C
,G) *),D)) *) A,G) (G,G) ((((G ((C,G) *),D) (B G ((C,G) *),D) *) A,G) (((G,D) + (G C G ((C,G) *),D)) ((B G,D) + (B G C G ((C,G) *)
,D)) *) A,G) (( 0 *)  0 *) ((((((C,G) *),D) (B G ((C,G) *),D) *) A,G) ((D + (C G ((C,G) *),D)) ((B G,D) + (B G C G ((C,G) *),D)) *
) A,G) (((((G (C,G) *),D) (B G ((C,G) *),D) *) A,G) ((G ((C,G) *),D) ((B G,D) + (B G C G ((C,G) *),D)) *) A,G) ((((((C,G) *),D) (B
 G ((C,G) *),D) *) A,G) ((((C,G) *),D) ((B G,D) + (B G C G ((C,G) *),D)) *) A,G) (((((G (C,G) *),D) (B G ((C,G) *),D) *) A,G) (((G
 (C,G) *),D) ((B G,D) + (B G C G ((C,G) *),D)) *) A,G))

END OF EVALQUOTE OPERATOR,

***STATISTICS***

RUN TIME(SEC)...          58.

CONS COUNTER ...       53755.

RECLAIMER    ...          17.

*FIN


// END
```

```
**********************************************************************************************
*******   ACCOUNTING INFORMATION   ***************************DATE 02/15/74*******
**                                                                                          **
**  JOB NAME--ZYS56      JOB#--0560   JOB CLASS--B                                         **
**                                                                                          **
**  CPU TIME--000100   START TIME--175526   END TIME--175639   MEMORY--100000              **
**                                                                                          **
**  SYSIPT--000115   SYSLST--000237   SYSOPT--000000                                       **
**                                                                                          **
**  CO--000192                                                                            **
**                                                                                          **
**********************************************EDOS V10 R00*****************************
**********************************************************************************************
```

```
0001  SNOBOL
0002  *
0003              ALPHBT = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' /(MIN1)
0004  *
0005  *
0006  *
0007              DEFINE('LEFT(X)','LEFT1')
0008  LEFT1       LEFT = '#' X '#'
0009              LEFT '#(' *(X1)* '+' *(X2)* ')#' = X1 /S(RETURN)
0010              LEFT '#(' *(X1)* '.' *(X2)* ')#' = X1 /S(RETURN)
0011              LEFT '#' *(X1)* '*#' = X1 /(RETURN)
0012  *
0013  *
0014  *
0015              DEFINE('RIGHT(X)','RIGHT1')
0016  RIGHT1      RIGHT = '#' X '#'
0017              RIGHT '#(' *(X1)* '+' *(X2)* ')#' = X2 /S(RETURN)
0018              RIGHT '#(' *(X1)* '.' *(X2)* ')#' = X2 /(RETURN)
0019  *
0020  *
0021  *
0022              DEFINE('OPER(X)','OPER1')
0023  OPER1       OPER = '#' X '#'
0024              OPER '#(' *(X1)* '+' *(X2)* ')#' = '+' /S(RETURN)
0025              OPER '#(' *(X1)* '.' *(X2)* ')#' = '.' /S(RETURN)
0026              OPER '#' *(X1)* '*#' = '*' /(RETURN)
0027  *
0028  *
0029  *
0030              DEFINE('HNDL(X)','HNDL1')
0031  HNDL1       HNDL = .EQ(SIZE(X),'1') X /S(RETURN)
0032              HNDL = LEFT(X) /(RETURN)
0033  *
0034  *
0035  *
0036              DEFINE('DRV(X)','DRV1')
0037  DRV1        DRV = .EQ(SIZE(X),'1') '0*' /S(RETURN)
0038              DRV = RIGHT(X) /(RETURN)
0039  *
0040  *
0041  *
0042              DEFINE('ORDER(X)','ORDER1')
0043  ORDER1      ORDER = EQUALS(X,'NIL') '26' /S(RETURN)
0044              X1 = EQUALS(OPER(X),'+') HNDL(LEFT(X)) /S(ORDER2)
0045              X1 = HNDL(X)
0046  ORDER2      ALPHBT *X2* X1
0047              ORDER = SIZE(X2) /(RETURN)
0048  *
0049  *
0050  *
0051              DEFINE('CONA(X,Y)','CONA1')
0052  CONA1       CONA = Y
0053              X1 = EQUALS(OPER(Y),'+') LEFT(Y) /F(CONA2)
0054              CONA = UNEQL(X,X1) '(' X1 '+' CONA(X,RIGHT(Y)) ')'
0055  .                 /(RETURN)
0056  CONA2       CONA = UNEQL(X,Y) '(' Y '+' X ')' /(RETURN)
0057  *
0058  *
0059  *
0060              DEFINE('CON(X,Y)','CON1')
0061  CON1        CON = UNEQL(OPER(X),'+') CONA(X,Y) /S(RETURN)
0062              CON = CON(RIGHT(X),CONA(LEFT(X),Y)) /(RETURN)
0063  *
```

207

```
0064  *
0065  *
0066              DEFINE('MRG(X,Y)','MRG1')
0067  MRG1        ORD1 = ORDER(X)
0068              ORD2 = ORDER(Y)
0069              MRG = .LT(ORD1,ORD2)
0070  .                 '(' LEFT(X) '+' MRG(RIGHT(X),Y) ')' /S(RETURN)
0071              MRG = .GT(ORD1,ORD2)
0072  .                 '(' LEFT(Y) '+' MRG(X,RIGHT(Y)) ')' /S(RETURN)
0073              MRG = .EQ(ORD1,'26') 'NIL' /S(RETURN)
0074              MRG = '((' HNDL(LEFT(X)) '.'
0075  .                  CON(DRV(LEFT(X)),DRV(LEFT(Y))) ')+'
0076  .                  MRG(RIGHT(X),RIGHT(Y)) ')' /(RETURN)
0077  *
0078  *
0079  *
0080              DEFINE('DISTR(X,Y)','DISTR1')
0081  DISTR1      DISTR = EQUALS(X,'NIL') X /S(RETURN)
0082              DLFT = LEFT(X)
0083              DISTR = .EQ(SIZE(DLFT),'1')
0084  .                  '((' DLFT '.' Y ')+' DISTR(RIGHT(X),Y) ')'
0085  .                   /S(RETURN)
0086              DISTR = '((' LEFT(DLFT) '.(' RIGHT(DLFT)
0087  .                  '.' Y '))+' DISTR(RIGHT(X),Y) ')' /(RETURN)
0088  *
0089  *
0090  *
0091              DEFINE('ADDCNST(X,Y)','ADDCNST1')
0092  ADDCNST1    ADDCNST = EQUALS(X,'0') Y /S(RETURN)
0093              ADDCNST = '0*' /(RETURN)
0094  *
0095  *
0096  *
0097              DEFINE('NML(X)','NML11','ALPHA')
0098  NML11       OPE = .NE(SIZE(X),'1') OPER(X) /F(NML12)
0099              EQUALS(OPE,'*') /S(NML21)
0100              EQUALS(OPE,'+') /S(NML31)
0101              EQUALS(OPE,'.') /S(NML41)F(FRETURN)
0102  *
0103  NML12       NML = EQUALS(X,'0') '(NIL+0)' /S(RETURN)
0104              ALPHBT X /F(FRETURN)
0105              NML = '((' X '+NIL)+0)' /(RETURN)
0106  *
0107  *
0108  NML21       ALPHA = NML(LEFT(X)) /F(FRETURN)
0109              NML = '(' DISTR(LEFT(ALPHA),X) '+0*)' /(RETURN)
0110  *
0111  NML31       ALPHA = NML(LEFT(X)) /F(FRETURN)
0112              BETA = NML(RIGHT(X)) /F(FRETURN)
0113              NML = '(' MRG(LEFT(ALPHA),LEFT(BETA)) '+'
0114  .                  ADDCNST(RIGHT(ALPHA),RIGHT(BETA)) ')' /(RETURN)
0115  *
0116  NML41       ALPHA = NML(LEFT(X)) /F(FRETURN)
0117              NML = EQUALS(RIGHT(ALPHA),'0')
0118  .            '(' DISTR(LEFT(ALPHA),RIGHT(X)) '+0)' /S(RETURN)
0119              BETA = NML(RIGHT(X)) /F(FRETURN)
0120              NML = '(' MRG(DISTR(LEFT(ALPHA),RIGHT(X)),LEFT(BETA))
0121  .                 '+' RIGHT(BETA) ')' /(RETURN)
0122  *
0123  *
```

```
0124  *
0125              DEFINE('SUBST(X,Y)','SUBST1')
0126  SUBST1      SUBST = X
0127  SUBST2      X '(' *(TRM)* '+' *(X1)* ')' = X1 /F(RETURN)
0128              HL = HNDL(TRM)
0129              DV = DRV(TRM)
0130              I = '1'
0131  SUBST3      I = UNEQL($(Y I),DV) I + '1' /F(SUBST4)
0132              $(Y I) = .GT(I,$('LH' Y)) DV /F(SUBST3)
0133              $('LH' Y) = I
0134  SUBST4      SUBST TRM = '(' HL '.' Y I ')' /(SUBST2)
0135  *
0136  *
0137  *
0138              DEFINE('PRINT(X)','PRINT1')
0139  PRINT1      X *SYSPOT/'70'* *X* /S(PRINT1)
0140              SYSPOT = X /(RETURN)
0141  *
0142  *
0143  *
0144              DEFINE('AM(X)','AM1')
0145  AM1         J = '1'
0146              S1 = X
0147              LHS = '1'
0148  AM2         NR = NML($('S' J)) /F(AM3)
0149              SYSPOT =
0150              PRINT('S' J ' = ' $('S' J))
0151              PRINT('    = ' NR)
0152              PRINT('    = (' SUBST(LEFT(NR),'S') '+' RIGHT(NR) ')')
0153              J = .NE(J,LHS) J + '1' /S(AM2)F(RETURN)
0154  *
0155  AM3         SYSPOT = 'NOT RE' /(RETURN)
0156  *
0157  *
0158  *
0159  *
0160  MIN1        X1 = SYSPIT
0161              X1 = UNEQL(X1,'END') AM(X1) /S(MIN1)
0162  END
**** LIST END ****
@
```

```
209 :PR,SNOB,2,99
      1  SNOBOL

  FIN DE COMPILATION . NOMBRE D'ERREURS:      Ø

  NOMBRE DE MOTS OCCUPES PAR LE PROGRAMME:  2006

(((B.A)+(A*.(A.B)))*.A*)


S1 = (((B.A)+(A*.(A.B)))*.A*)
   = (((A.(A*+(((B+(A*.(A.B))).((B.A)+(A*.(A.B)))*).A*)))+((B.((A.((B.
A)+(A*.(A.B)))*).A*))+NIL))+Ø*)
   = (((A.S2)+((B.S3)+NIL))+Ø*)

S2 = (A*+(((B+(A*.(A.B))).((B.A)+(A*.(A.B)))*).A*))
   = (((A.((((B+(A*.(A.B))).((B.A)+(A*.(A.B)))*).A*)+A*))+((B.(((B.A)+
(A*.(A.B)))*.A*))+NIL))+Ø*)
   = (((A.S4)+((B.S1)+NIL))+Ø*)

S3 = ((A.((B.A)+(A*.(A.B)))*).A*)
   = (((A.(((B.A)+(A*.(A.B)))*.A*))+NIL)+Ø)
   = (((A.S1)+NIL)+Ø)

S4 = ((((B+(A*.(A.B))).((B.A)+(A*.(A.B)))*).A*)+A*)
   = (((A.(A*+(((B+(A*.(A.B))).((B.A)+(A*.(A.B)))*).A*)))+((B.(((B.A)+
(A*.(A.B)))*.A*))+NIL))+Ø*)
   = (((A.S2)+((B.S1)+NIL))+Ø*)


(A+((A.B)+(B.A)))*


S1 = (A+((A.B)+(B.A)))*
   = (((A.((B+Ø*).(A+((A.B)+(B.A)))*))+((B.(A.(A+((A.B)+(B.A)))*))+NIL
))+Ø*)
   = (((A.S2)+((B.S3)+NIL))+Ø*)

S2 = ((B+Ø*).(A+((A.B)+(B.A)))*)
   = (((A.((B+Ø*).(A+((A.B)+(B.A)))*))+((B.((A.(A+((A.B)+(B.A)))*)+(A+
((A.B)+(B.A)))*))+NIL))+Ø*)
   = (((A.S2)+((B.S4)+NIL))+Ø*)

S3 = (A.(A+((A.B)+(B.A)))*)
   = (((A.(A+((A.B)+(B.A)))*)+NIL)+Ø)
   = (((A.S1)+NIL)+Ø)

S4 = ((A.(A+((A.B)+(B.A)))*)+(A+((A.B)+(B.A)))*)
   = (((A.(((B+Ø*).(A+((A.B)+(B.A)))*)+(A+((A.B)+(B.A)))*))+((B.(A.(A+
((A.B)+(B.A)))*))+NIL))+Ø*)
   = (((A.S5)+((B.S3)+NIL))+Ø*)

S5 = (((B+Ø*).(A+((A.B)+(B.A)))*)+(A+((A.B)+(B.A)))*)
   = (((A.((B+Ø*).(A+((A.B)+(B.A)))*))+((B.((A.(A+((A.B)+(B.A)))*)+(A+
((A.B)+(B.A)))*))+NIL))+Ø*)
   = (((A.S2)+((B.S4)+NIL))+Ø*)

END

SNOBOL HP FRANCE -  FIN DE L'INTERPRETATION
                    00075 GARBAGE COLLECTION(S)


$END SNOB
```