1971 год ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ Выпуск 41

УДК 681.3.06:51

ЭР-I—ЯЗЫК ПРОГРАММИРОВАНИЯ ЗАДАЧ ПЕРЕРАБОТКИ СЛОВ В ПРОИЗВОЛЬНОМ АЛФАВИТЕ

Ю.Н. Корнев, С.В. Пискунов, С.Н. Сергеев

В данной работе кратко и на содержательном уровне описываются основные черты языка. Язык ЭР-I является языком программирования, предназначенным для преобразования слов в произвольном алфавите.

По носителю язык ориентирован на машины с однородной структурой [3] и на однородные сети конечных автоматов: итеративные сети, вычислительные среды и т.п. Такая ориентация требует простоты программы интерпретации этого языка. Поэтому авторы старались построить язык с минимумом изобразительных средств, который был продиктован назначением языка: описание задач интерпретации формальных языков, описание задач логического вывода и лингвистических задач.

Стремление сделать интерпретацию языка и его синтаксический контроль простыми и быстрыми явилось причиной того, что в ЭР-I введены ограничения на длину имен. В описываемом варианте языка приняты односимвольные имена.

Так же как в языках COMIT [1], SNOBOL (SNOBOL-3) [2], в ЭР-I основным средством преобразования слов является операция подстановки.

Набор операторов языка и их модификаций позволяет записывать алгоритмы задач в простой и удобной форме, близкой к естественной.

I. Содержание языка

Операторами языка являются:
1) оператор подстановки,
2) условный оператор,
3) оператор чтения,
4) оператор переписи,
5) оператор чистки,
6) оператор конца.

I.I. О п е р а т о р п о д с т а н о в к и предназначен для преобразования программ и состоит из трех частей: I-я часть – указатель границ слова, II-я часть – некое слово, именуемое левой частью оператора подстановки, III-я часть – слово, именуемое правой частью оператора подстановки.

П р и м е р : в М заменить АВС на ВАС;
I II III

Символы: в, заменить, на, ; – это ограничители, позволяющие определить тип оператора. Отметим, что далее все подчеркнутые слова являются символами – ограничителями языка.

Смысл оператора заключается в следующем. В слове, расположенном правее оператора, отыскивается подслово, началом которого является первое вхождение двухсимвольного слова (первый символ – Г, второй – символ алфавита, совпадающий с указателем границ слова в операторе подстановки), концом – второе. В этом подслове отыскивается первое вхождение левой части оператора и, если оно есть, заменяется подсловом – правая часть оператора.

Если в программе правее выписанного в примере оператора будет слово ГМ ААВС СА ГМ, то поиск вхождения левой части будет успешным, и слово в границах М будет преобразовано к виду ГМ АВАССА ГМ.

В качестве левой или правой части оператора подстановки может быть взято произвольное слово, например, слово, содержащее такие разделители языка, как: на, ; и т.п. В данном случае это

слово ограничивается слева и справа символами / с односимвольными именами. Например, слово:

в M заменить ABC на BAC;,

используемое в качестве правой части некоторого оператора подстановки, приобретает вид:

/ A в M заменить ABC на BAC; / A

Использование символов / с другими именами внутри левой (правой) части оператора подстановки допускается.

В левой и правой частях оператора могут использоваться переменные символы и слова.

Указатель границ в операторе подстановки также может быть переменным символом.

Переменный символ (слово) изображается парой символов, один из которых символ символ (слово), другой - имя этого символа.

Примеры переменных символов и слов: символ A, символ B, слово C.

Значением переменного символа может быть только однобук - венное слово, значением переменного слова - любое слово.

Использование переменных символов и слов, способы определения их значений во многом аналогичны применению и поиску значений для строчной переменной фиксированной длины (fixed - - lenght string variable), в нашем случае длины единица, и произвольной строчной переменной (arbitrary string variable) языка SNOBOL . Отметим только, что определению значений переменных символов и слов предшествует проверка, не были ли они определены ранее, а найденные значения переменных символов и слов хранятся в специальных подсловах, начинающихся символами значения переменных и заканчивающихся символами ;. Такие подслова называются областями значений переменных символов и слов. Проверка наличия значения переменного символа или слова производится в области значений переменных символов и слов, ближайшей справа к выполняемому оператору ; туда же заносятся значения определенных в операторе переменных символов и слов.

П р и м е р . Пусть левая часть некоторого оператора подстановки имеет вид:

A слово x B слово ρ символ z слово y C,

ближайшая область значений переменных символов и слов:

значения переменных слово ρ AA символ φ B ;

а слово в границах, перерабатываемое оператором - AABBAACCAC , тогда поиск левой части будет успешным, и область значений переменных символов и слов приобретает вид:

значения переменных слово x AB символ z C слово y CA слово AA символ B ;

Если указатель границы в операторе является переменным символом, то его значение обязательно должно быть в области значений переменных символов и слов, ближайшей справа к этому оператору. Это значение и является именем границ слова, которое перерабатывает оператор.

Если в правой части оператора подстановки есть переменные символы и (или) слова, то к моменту выполнения замены найденного вхождения левой части оператора подстановки правой частью оператора подстановки их значения должны быть определены, т.е. должны храниться в ближайшей справа к выполняемому оператору области значений переменных символов и слов.

Оператором типа подстановки является оператор определения вхождения. Этот оператор тождествене оператору подстановки с совпадающей левой и правой частями оператора подстановки.

П р и м е р : в A определитель вхождения AB;
в A вхождение AB ;
в символ z определить вхождение A
символ y ;

I.2. У с л о в н ы й о п е р а т о р предназначен для управления процессом чтения и преобразования программ.

Этот оператор состоит из двух частей: I-я часть - условие, II-я часть - слово условного оператора (им может быть, например, опять какой-нибудь оператор). Основной частью условия является оператор типа подстановки.

П р и м е р ы условных операторов.

а) если в A заменить СД на ОР применима то в A заменить A на B;
(I — II)

б) если в A заменить СД на ОР не применима то в A заменить A на B;

в) если в символ z вхождение СД есть то в A читать A;

г) если в A вхождение СД нет то читать слева B;

Если в операторе а) оператор подстановки из условия при-

менным, он выполняется, и дальнейшее чтение программы осуществляется с первого символа слова условного оператора, т.е. выполняется оператор:

в A заменить A на B; .

Если в операторе б) оператор подстановки из условия применим, он выполняется, и дальнейшее чтение программы осуществляется с символа, стоящего после символа;.

Если в операторе а) оператор подстановки из условия не применим (в перерабатываемом слове нет вхождений левой части этого оператора), то дальнейшее чтение программы осуществляется с символа, стоящего после символа;.

Если в операторе б) оператор подстановки из условия применим, он выполняется и дальнейшее чтение программы осуществляется с первого символа слова условного оператора.

В том случае, когда слово условного оператора содержит символ; (является, например, последовательностью операторов), оно, аналогично тому, как это делается для левой (правой) части оператора подстановки, ограничивается разделителем $/$ с односимвольным именем.

I.3. О п е р а т о р ч т е н и я предназначен для управления процессом чтения программ. Существуют две разновидности этих операторов. Первый тип имеет вид:

в A читать B; или в символ x читать B;

и интерпретируется так: в слове с границами A, либо с границами, являющимися значением переменного символа, определяемым так же, как значение переменной границы в операторе подстановки, отыскивается слово, помеченное меткой B:. Отметим, что оператор чтения обязательно является подсловом слова в границах, указанных в данном операторе. Дальнейшее чтение программы ведется с символа, стоящего справа от вхождения метки.

П р и м е р : $\Gamma\ \alpha$ B: AB$\Gamma\ \beta$ ABB:AB в символ x читать B; AB $\Gamma\ \beta\ \Gamma\ \alpha$ AB значения переменных символ $x\ \beta$; .

Дальнейшее чтение программы в этом случае будет продолжено с третьего вхождения символа A в слово примера.

Второй тип оператора имеет вид:

читать слева B ; или читать справа A ; .

Выполнение оператора этого типа состоит в отыскании метки

слева (справа) от оператора и продолжении чтения программы с символа, стоящего после этой метки.

П р и м е р : B: $\Gamma\ \alpha$ A$\Gamma\ \beta$ AB читать слева B; B: $\Gamma\ \beta$ AB $\Gamma\ \alpha$

Чтение программы будет продолжено с символа, стоящего после первого вхождения метки B: в слово.

Имя метки в операторе чтения может быть переменным символом. Например, в α читать символ x ;.

В этом случае значение слова символ x должно быть записано в ближайшей к оператору справа области значений переменных символов и слов. Выполнение оператора заключается в нахождении в подслове с границами α подслова, помеченного меткой, имя которой является значением этого переменного символа.

I.4. О п е р а т о р ы п е р е п и с и предназначены для преобразования программ и делятся на два типа: операторы переписи переменных и операторы переписи слов.

П р и м е р ы операторов переписи переменных:

переписать символ x к γ ;

переписать слово χ к γ ; .

Выполнение этих операторов заключается в том, что в программе отыскивается символ γ и к нему из ближайшей справа области значений переменных символов и слов переписывается значение переменного символа или слова.

П р и м е р оператора переписи слов:

переписать $\alpha\beta$ к γ ;

Выполнение этого оператора состоит в том, что в программе отыскивается подслово, ограниченное слева символом α, справа - β, и помещается после символа γ.

П р и м е р : AA γ ABC переписать $\alpha\beta$ к γ ; A α BBB β A

В результате выполнения оператора слово приобретает вид:

AA γ BBBABC переписать $\alpha\beta$ к γ ; A α BBB β A .

I.5. О п е р а т о р ы ч и с т к и предназначены для преоораования программ и имеют следующие разновидности: оператор чистки переменных, оператор чистки слов, левый оператор чистки слов, правый оператор чистки слов.

П р и м е р ы операторов. Оператор чистки переменных:

стереть слово x символ x слово y ;

Оператор чистки слов:

стереть $\alpha\beta$;

Левый оператор чистки слов;

стереть слева $\gamma\delta$;

Правый оператор чистки слов:

стереть справа $\mu\nu$;

Выполнение оператора чистки переменных состоит в том, что в ближайшем справа вхождении области значений переменных символов и слов стираются переменные символы и слова вместе с их значениями, перечисленные в списке оператора чистки.

Выполнение оператора чистки слов заключается в том, что в программе отыскивается подслово, началом которого является символ α , концом - β , и заменяется пустым словом.

Левый (правый) оператор чистки выполняется аналогично, только вхождение подслова ищется слева (справа) от оператора.

I.6. О п е р а т о р к о н ц а . Его вид: стоп;

Выполнение этого оператора состоит в прекращении чтения программы.

I.7. В языке предусмотрена возможность произвольные слова делать недоступными для операторов чтения и подстановки.

Слово, началом которого является пара символов $\leq\leq$, концом - $\geq\geq$, не содержащее других вхождений пары символов $\geq\geq$, недоступно для оператора чтения.

Аналогично, с заменой символа $\leq$ символом $<$, а символа $\geq$ символом $>$,определяется слово, закрытое для оператора подстановки.

П р и м е р : $\Gamma\alpha$ AB в α читать C; CC $\leq\leq$ C: C $\geq\geq$ C : $C\Gamma\alpha$.

Чтение слова будет продолжено с седьмого вхождения символа C в слово.

2. Структура программ, записанных на языке $\mathcal{ЭР}$ -I, и правила их выполнения

2.I. Пусть начало программы, алеф-символы языка. Программ называется слово, началом которого служит слово начало программы $<$ слово,не содержащее символа алеф $>$алеф,концом - конец программы. Других вхождений символов начало программы, конец программы в программу нет.

2.2. Введем понятие правильной структуры относительно оператора: это такое слово, только находясь в котором оператор может выполняться.

Определим правила чтения программы.

а) Программа читается слева направо, с первого вхождения символа алеф.

б) Если при чтении встречается оператор языка, который входит в слово, являющееся правильной структурой относительно этого оператора, то он выполняется.

Приведем п р и м е р ы правильных структур некоторых операторов.

а) Для оператора подстановки с постоянным указателем границ слова, левой и правой частями оператора, содержащими переменные символы и слова, правильная структура представляет собой слово

$$A_I\ A_2\ A_3\ A_4\ A_5\ A_6\ A_7\ A_{22}\ A_8,$$

где

A_I - слово вида начало программы $<$ слово, не содержащее символа алеф $>$ алеф;

A_2, A_{22} - произвольные слова;

A_3 - оператор подстановки;

A_4 - произвольное слово, не содержащее в подсловах,открытых для оператора подстановки, вхождений символа значения переменных и двубуквенных подслов Q , у которых первый символ Γ , второй - указатель границ слова оператора подстановки;

A_5 - описание области значений переменных символов и слов, содержащее все такие переменные символы и слова из правой части оператора подстановки, каких нет в левой части этого оператора;

A_6 - слово, не содержащее в подсловах, открытых для оператора подстановки, слов Q ;

A_7 - слово, ограниченное слева и справа словом Q и не содержащее других вхождений этого слова в подсловах, открытых для оператора подстановки;

A_8 – конец программы.

б) Для оператора чтения первого типа с постоянным именем границы (например, s) и именем метки t правильная структура представляет собой либо слово

$$A_1\ A_2\ A_9\ A_{13}\ A_{10}\ A_{12}\ A_{11}\ A_{23}\ A_9\ A_{22}\ A_8,$$

где

A_9 – слово Г s ,

A_{10} – слово t : ,

A_{11} – оператор чтения,

A_{12}, A_{23} – произвольные слова, не содержащие в подсловах, открытых для оператора чтения, слов t : , Г s ,

A_{13} – слово, не содержащее в подсловах, открытых для оператора чтения, подслов Г s ,

A_1, A_2, A_{22}, A_8 имеют тот же смысл, что и в а); либо слово

$$A_1\ A_2\ A_9\ A_{12}\ A_{11}\ A_{12}\ A_{10}A_{13}\ A_9\ A_2\ A_8.$$

в) Для оператора чистки переменных правильная структура представляет собой слово

$$A_1\ A_2\ A_{14}\ A_{15}\ A_{16}\ A_8,$$

где

A_{14} – оператор чистки переменных,

A_{15} – слово, не содержащее вхождений символа значения переменных,

A_{16} – описание области значений переменных символов и слов,

A_1, A_2, A_8 имеют тот же смысл, что и в а).

г) Для оператора чистки слов правильная структура представляет собой либо слово

$$A_1\ A_2\ A_{17}\ A_{18}\ A_{19}\ A_{24}\ A_{20}\ A_{25}\ A_8,$$

где

A_{17} – левая граница стирания оператора чистки слов,

A_{18}, A_{24}, A_{25} – слова, не содержащие вхождений ни левой, ни правой границы стирания оператора чистки слов,

A_{19} – правая граница стирания оператора чистки слов,

A_{20} – оператор чистки слов;

либо слово

$$A_1\ A_2\ A_{17}\ A_{18}\ A_{20}\ A_{24}\ A_{19}\ A_{21}\ A_8,$$

где

A_{21} – слово, не содержащее вхождений левой границы стирания оператора чистки слов; либо слово

$$A_1\ A_{18}\ A_{20}\ A_{18}\ A_{17}\ A_{18}\ A_{19}\ A_2\ A_8.$$

Программа, которая в процессе выполнения каждого оператора является правильной структурой относительно выполняемого оператора, называется п р а в и л ь н о й .

Результатом выполнения оператора является либо изменение программы, либо указание символа, с которого следует продолжить чтение программы, либо то и другое вместе.

Если характер чтения оператором не меняется, то после его выполнения программа читается либо начиная с символа, стоящего за просмотренным оператором, либо, если просмотренный оператор был оператором чистки слов, расположенным внутри стираемой области, с символа, стоящего за правой границей стирания стираемой области.

Отметим, что при чтении программы анализ правильности структуры, в которой находится оператор , производится только в момент чтения оператора. В какой структуре находился оператор до начала чтения, значения не имеет.

Если программа в момент чтения оператора не является правильной, то перед этим оператором ставится символ алеф. Перед первым вхождением символа алеф записывается слово, характеризующее нарушение структуры, затем или выполняется та часть программы, которая была записана между символами начало программы, алеф ("сбойная"), или, если между символами начало программы, алеф находится только пустое слово, выполняется оператор стоп.

Приведем п р и м е р ы программ:

а) начало программы алеф АВС в А заменить С на Д;
АВГА АВ ГА конец программы

б) начало программы алеф АВ в В заменить А на В;
АГА АВСД АВСД АВ ГА ВС конец программы

в) начало программы алеф АВ ; если в А заменить * на ГВ применима то читать слева В; в В заменить А на В; АГА АВСД * АВСД * АВГА ВС конец программы

В приведенных примерах а) - пример правильной программы, б) - неправильной, в) - правильной, так как к моменту выполнения второго оператора в программе уже будет слово с границами В.

3. Заключение

Отметим некоторые характерные черты языка.

3.1. Полнота языка. Пример построения по записи произвольного марковского алгоритма программы на языке $\mathcal{ЯР}$ - I программой, записанной на том же языке, является доказательством полноты (см.приложение).

3.2. Введение в язык именованных разделителей ($\perp$) и переменных символов и слов, позволяющее в качестве левых и правых частей операторов типа подстановки и слов условных операторов использовать произвольные слова (например, операторы,любые их части, или последовательности операторов языка $\mathcal{ЯР}$ -I), отказ от жесткой синтаксической структуры программы, введение для каждого оператора языка понятия правильной структуры -слова,в котором оператор выполняется, обеспечили возможность преобразования программы в процессе реализации. Это свойство названо самоприменимостью.

В приложении приводится построение одной программой другой средствами одного и того же языка $\mathcal{ЯР}$ - I.

Для проверки основных свойств и отработки основных средств языка была построена математическая модель реализации языка однородной цепочкой автоматов. Эта модель выполнена в виде программы в кодах БЭСМ-6 и по существу является интерпретатором языка $\mathcal{ЯР}$ - I. Основные свойства этого интерпретатора:

а) синтаксический контроль выполняемой программы осуществляется динамически, т.е. операторы программы интерпретируются до тех пор, пока не встретится нарушение синтаксической структуры слова, в котором выполняется оператор;

б) внешний алфавит не зафиксирован, определяется потребителем;

в) язык $\mathcal{ЯР}$ - I расширен операторами ввод-вывод.

4. Приложение

4.1. $\mathcal{ЯР}$ - I является алгоритмическим языком.

Будем рассматривать марковские алгоритмы в алфавите А,состоящем из строчных букв русского алфавита. В качестве алфавита $\mathcal{ЯР}$ - I возьмем объединение букв русского, греческого и латинского алфавитов, цифр и символов $\rightarrow$, $\cdot$, ∇ , $\wedge$, (,).

Докажем утверждение в такой формулировке: для любого марковского алгоритма α в алфавите А, перерабатывающего некоторое слово Q в этом алфавите, существует эффективная процедура построения алгоритма Σ, записанного на языке $\mathcal{ЯР}$ - I, эквивалентного α по действию, т.е. также перерабатывающего то же исходное слово Q .

Пусть

$$A_1 \rightarrow B_1 ,$$
$$A_2 \rightarrow B_2 ,$$
$$\dots\dots$$
$$A_i \rightarrow \cdot B_i ,$$
$$\dots\dots$$
$$A_j \rightarrow \cdot B_j ,$$
$$\dots\dots$$
$$A_n \rightarrow B_n ,$$

(где A_k , B_k , $k = 1,\dots, n$ - слова в алфавите А, а точкой отмечены заключительные правила), - схема некоторого марковского алгоритма α , перерабатывающего слово Q . Представим схему алгоритма α и слово Q так: $A_1 \rightarrow B_1$, $A_2 \rightarrow B_2$, ..., $A_i \rightarrow B_i$,..., $A_j \rightarrow \cdot B_j$,..., $A_n \rightarrow B_n \nabla Q$. Доказательством утверждения служит программа 4.2, записанная на языке $\mathcal{ЯР}$ - I.

Относительно каждого оператора, записанного в границах α, программа 4.2 является правильной. При ее выполнении правильность относительно этих операторов не нарушается. В результате выполнения операторов, записанных в границах α , программа приобретает вид 4.3. Последняя, в свою очередь, является правильной программой относительно каждого оператора в границах γ , выполнение операторов не нарушает ее правильности.

Эта программа является записью искомого алгоритма .Про-

цесс переработки слова, выполняемый программой 4.3, выглядит так. Читается первый условный оператор; если он применим, то происходит возврат на метку $\mathfrak{X}$ и повторное чтение программы, если нет - то чтение следующего оператора и т.д. Если читается условный оператор, построенный для заключительной подстановки из $\mathcal{A}$, и этот оператор применим, то процесс переработки слова Q прекращается; если он не применим, читается следующий оператор. Если ни один из операторов не применим, то переработка слова Q прекращается. Этот процесс совпадает с процессом переработки слова Q алгоритмом $\mathcal{A}$.

4.2. - Программа

Начало программы алеф $\mathfrak{z}$
граница α
в β заменить (слово $x\rightarrow$ слово y , на $\perp\xi$ граница $\gamma\mathfrak{x}$: если в μ заменить слово x на слово y применима то в γ читать $\mathfrak{x}$; $m\perp\xi$;
в β заменить (слово $x\rightarrow$. слово y , на $\perp\xi$ граница $\gamma\mathfrak{x}$: если в μ заменить слово x на слово y применима то стоп; $m\perp\xi$;
δ : стереть слово x слово y ;
если в β заменить m слово $x\rightarrow$ слово y , на $\perp\xi$ если в μ заменить слово x на слово y применима то в γ читать $\mathfrak{x}$; $m\perp\xi$ применима то в α читать δ ;
если в β заменить m слово $x\rightarrow$. слово y , на $\perp\xi$ если в μ заменить слово x на слово y применима то стоп; $m\perp\xi$ применима то в α читать δ ;
в β заменить m слово $x\rightarrow$ слово $y\ \nabla$ слово z)на $\perp\xi$ если в μ заменить слово x на слово y применима то в γ читать $\mathfrak{x}$; стоп; граница γ граница μ слово z граница $\mu\perp\xi$;
в β заменить m слово $x\rightarrow$. слово $y\ \nabla$ слово z) на $\perp\xi$ в μ заменить слово x на слово y ; стоп; граница γ граница μ слово z граница $\mu\perp\xi$;
в β заменить (слово $x\rightarrow$ слово $y\ \nabla$ слово z) на $\perp\xi$ граница $\gamma\mathfrak{x}$: если в μ заменить слово x на слово y применима то в γ читать $\mathfrak{x}$; стоп; граница γ граница μ слово z граница $\mu\perp\xi$;
в β заменить (слово $x\rightarrow$. слово $y\ \nabla$ слово z) на $\perp\xi$ граница γ в μ заменить слово x на слово y ; стоп; граница 4 граница z слово z граница $\mu\perp\xi$;
стереть $u v$;
стереть $\mathfrak{z}\,t$;
граница α
значения переменных ;
граница β

$$t(A_1\rightarrow B_1,\ldots,A_i\rightarrow\cdot B_i,\ldots,A_n\rightarrow B_n\ \nabla\ Q)\,u$$

граница $\beta\,v$
конец программы

4.3. Программа

Начало программы алеф
граница γ
$\mathfrak{x}$: если в μ заменить A_1 на B_1 применима то в γ читать $\mathfrak{x}$; если в μ заменить A_2 на B_2 применима то в γ читать $\mathfrak{x}$;
...
если в μ заменить A_i на B_i применима то стоп;
...
если в μ заменить A_j на B_j применима то стоп;
...
если в μ заменить A_n на B_n применима то в γ читать $\mathfrak{x}$;
стоп;
граница γ
граница μ
Q
граница μ
конец программы
ПРИМЕЧАНИЕ: Символы Γ и граница - синонимы.

Л И Т Е Р А Т У Р А

1. An Introduction to COMIT Programming. The Research Lab of Electronics and the Computation Center, M.I.T. 1961.
2. The SNOBOL 3 Programming Language. The Bell System Technical Journal, vol. XLV, N 6, 1966.
3. Ф.А. АТАМОВ, Ф.А. ГАДЖИЕВ, М.А. КАСНЕР. Вычислительные машины с однородной структурой.–Труды института кибернетики АН Азерб. ССР, 1967, том IУ.

Поступила в редакцию
5. У. 1970