БЕЗЪЯЗЫКОВОЕ ПРОГРАММИРОВАНИЕ

А.ван Вейнгаарден

Амстердам, Нидерланды

Эта статья показывает, что алгоритмы можно представлять с помощью двухуровневой грамматики, без вмешательства языка программирования, причем с той же примерно легкостью, что и при использовании языка.

I. Введение

Пусть программист желает задать некоторый процесс над данными. С этой целью он пишет на каком-то языке программирования, например на Алголе 68, некоторую собственно программу, например, такую:

```
(цел n;  чит (n); печ (("р(", целое (n, 0),
      ")_. - _. 25 _. = _. ", целое (простое (n)-25,0))))
```

Использованные здесь значки суть представления терминальных символов Алгола 68.

Эта собственно программа не вполне еще задает процесс, и тому есть две причины. Во-первых, ПС 68, то есть определение Алгола 68 [2] , не дает описания для идентификатора

```
простое .
```

Можно, однако, это описание дать в собственном вступлении. Допустим, что оно там в самом деле дано, и что вызов

```
простое (n)
```

выдает именно то, что и ожидалось от процедуры с таким идентификатором – n-ое простое число. Во-вторых, вызов

```
чит (n)
```

может присвоить переменной n и неположительное число. Из-за этого либо мы должны допустить, что процедура

простое

как-то справляется с этой ситуацией, либо же программисту придется исправить свою собственно программу, написав, например,

чит(n); n : = абс n + 1

вместо

чит (n).

Мы принимаем первое предположение.

Эта собственно программа может быть исполнена некоторым вычислителем. Допустим, что эффект вызова

чит (n)

таков же, какой был бы у оператора

n : = I25

Так как I25- м простым числом является, как хорошо известно, число 69I, то результат должен быть таким:

p (I 2 5) - 2 5 = 6 6 6

Значки здесь взяты из совершенно другого словаря, лишь частично определенного в ПС 68.

Очевидно, что определение языка программирования, в данном случае Алгола 68, входит в задание процесса, так как иначе смысл программы был бы неопределен. В ПС 68 определение языка дано частично на естественном (русском) языке, а частично с помощью некоторой конкретной двухуровневой грамматики с порождающими правилами вроде

программа: замкнутое предложение в новом сильно выдающее пустое значение.

Значки здесь взяты из еще одного, совершенно нового, словаря.

Очевидно далее, что определение концепции "двухуровневой грамматики" также входит в задание процесса, так как иначе смысл конкретной двухуровневой грамматики был бы неопределен. Это определение было впервые дано (неформальным образом) в [I] , а затем уточнено в [2] . Формальное операцион-

ное определение дано в [3] , оно приведено здесь в разделе 2.

Итак, задание процесса включает в себя, не считая определения на естественном языке, еще по меньшей мере четыре уровня определения:

- определение концепции двухуровневой грамматики (порожденное разработчиком инструментария);
- конкретную двухуровневую грамматику (порожденную автором языка);
- собственно программу на этом языке (порожденную программистом);
- выдачу этой собственно программы (порожденную вычислителем).

Это весьма внушительная иерархия и к тому же объемистая: ПС 68 содержит более 500 страниц. Вознаграждением служит только то, что программы на Алголе 68 могут исполняться с помощью современных вычислительных машин, но процесс все еще задается неудовлетворительно: так определение истинностного значения выражения $0 < 0$ сводится в ПС 68 к нестрогой формулировке "в обычном математическом смысле", а значение выражения $0 - 0 = 0$ в ПС 68 вообще не определено.

Причиной этого печального положения является то, что, с одной стороны, мы обладаем высокой определительной мощью двухуровневых грамматик, а с другой стороны, определению подлежит некоторый скромный процесс. Включить между ними большой язык программирования, не настроенный специально на этот процесс, это - если оставить в стороне исполнение на современных вычислительных машинах - все равно, что поручить слону приносить продукты из магазина: слон может часть их съесть по дороге.

В этой статье показано, как можно обойтись без языка, работая с конкретным примером. В действительности дано множество метаправил и гиперправил, мощь которых далеко превосходит нужды этого примера, и только лишь выбор конкретного гиперправила для исходного слова сводит все к этому примеру. В некотором смысле эти множества можно считать языком, но вот в чем существенная разница: "грамматик", то есть человек, задающий процесс с помощью двухуровневой грамматики, не будет писать представлений для терминальных букв; представления

для терминальных букв; представления эти, с другой стороны, эквивалентны значкам, которые будут написаны вычислителем. Язык, таким образом, устранен полностью.

Не следует преувеличивать значение критических замечаний по адресу Алгола 68; замечания по адресу других языков могли бы оказаться гораздо более критическими.

2. Двухуровневые грамматики

"Словарь" – это некоторое множество; его элементы называются "буквами". "Слово" над словарем V – это отображение $[1:n] \to V$ при произвольном $n \in N_0$, то есть множество, состоящее из n упорядоченных пар (i, v_i), где $i = 1,\ldots,n$, $v_i \in V$; v_i поэтому называется "i-й буквой", а n – "длиной" слова. Если $n = 0$, то слово является пустым множеством, которое называют также "пустым словом". Если V – словарь, то V^* обозначает множество слов над V, V^+ – множество непустых слов над V. "Предложение" над словарем V есть слово над словарем, именами своими буквами слова над V; таким образом, V^{**} есть множество предложений над V.

"Правило" есть упорядоченная пара (v, w), где v и w – слова над некоторыми словарями.

"Двухуровневая грамматика" ВВГ [*] – это упорядоченная шестерка $(V_m, V_o, V_t, R_m, R_h, W_s)$, где V_m, V_o и V_t – конечные словари, буквы которых называются соответственно "метабуквами", "ортобуквами" и "терминальными буквами", R_m R_h – конечные множества правил, называемых соответственно "метаправилами" и "гиперправилами", а W_s – некоторое слово над V_o, называемое "исходным словом".

Пусть $V_h := V_m \cup V_o$. Требуется, чтобы $V_m \cap V_o = \{\}$, $V_t \subset V_o^+$, $R_m \subset V_m \times V_h^*$, $R_h \subset V_h^+ \times V_h^{**}$, $W_s \in V_o^+$.

Грамматика ВВГ "порождает" некоторое множество предложений L, получаемое следующим образом.

Пусть $R_{mo} := V_m \times V_o^*$, $R_{oo} := V_o^+ \times V_o^{**}$, $R_{st} := \{W_s\} \times V_t^*$.

Вводятся множества R'_m, идентичные с R_m,

[*] То есть ван Вейнгаардена (Прим. переводчика)

и R'_h, идентичное с R_h; затем они расширяются применениями (в произвольном количестве и пока это возможно) Правила Расширения (ПР). При каждом применении ПР должны быть согласованным образом взяты либо первые, либо вторые, либо третьи альтернативы из каждой тройки альтернатив, разделенных значками ',' и заключенных между '(' и ')'.

ПР:К (R'_m, R'_h, R'_h) добавляется новое правило, которое получается – при наличии правила $(v', w') \in (R'_m \cap R_{mo}, R'_m \cap R_{mo}, R'_h \cap R_{oo})$ – заменой в копии некоторого правила $(v,w) \in (R'_m, R'_h, R'_h)$ (некоторого, каждого, некоторого) вхождения v' в (w, v и w, w) на w'.

Тогда $L := \{w|(w_s, w) \in R'_h \cap R_{st}\}$

Множество "терминальных метапорождений" метабуквы M есть множество $\{w|(M, w) \in R'_m \cap R_{mo}\}$.

В этой статье метабуквой служит обычная 'заглавная буква', за которой может следовать один или несколько ''' (апострофов); ортобуквой служит обычная 'малая буква', 'десятичная цифра', 'открывающая скобка' или 'закрывающая скобка'. Определенную роль играют еще четыре буквы: ':'(двоеточие), '.'(точка), ',' (запятая) и ';' (точка с запятой).

Буквы "выписываются" одна за другой так, чтобы было ясно, в каком порядке они выписаны. В этой статье принято, что очередная буква выписывается 'справа от' последней выписанной буквы или 'на следующей строчке' или 'на следующей странице', что бы это все ни значило.

Слово считается выписанным, если процесс его выписывания начался с выписывания первой буквы слова, при условии, что такая существует, и если после выписывания его i-й буквы процесс продолжился выписыванием (i + 1)-й буквы слова, при условии существования таковой.

Выписывание метаправила (v, w) состоит в последовательном выписывании v, двух двоеточий, w и точки.

Выписывание гиперправила (v, w) состоит в последовательном выписывании v, двоеточия, w и точки. Выписывание w создает, однако, ту проблему, что предложение над V_o может быть в дальнейшем прочтено неверно. Для избежания этого в естественных языках слова отделяются друг от друга пробелами.

Здесь же мы по традиции для отделения слов друг от друга будем выписывать запятую после выписывания одного слова и перед выписыванием следующего слова; это соглашение освобождает пробелы для использования их в целях удобства внутри самих слов.

Грамматический механизм, определенный выше, еще, однако, не полон. Терминальные буквы являются словами над V_0, но эта внутренняя структура не имеет никакого значения для пользователя. Поэтому V_t отображается на новый словарь W_t, являющийся множеством "представлений" терминальных букв. Эти представления - значки, выбранные пользователем по своему усмотрению, с тем, однако, чтобы они отличались от всех букв, упомянутых выше [ж].

Терминальное представление грамматики ВВГ, то есть элемент множества L, имеет представление, получающееся заменой каждой терминальной буквы в копии этого элемента на ее представление и удалением следующей за ней запятой, если таковая имеется.

Правила допускают полезную сокращенную форму представления: если два правила имеют одинаковую левую часть по двоеточие или двойное двоеточие включительно, то их можно объединить в одно правило, состоящее из первого правила, в котором точка заменена на точку с запятой, и - вслед за ним - правой части второго правила.

Таким образом ,

H :: 9; 8; 7.

заменяет собой

H :: 9. H : : 8. H : : 7 .

Другое полезное сокращение исходит из того, что зачастую нужно иметь метаправила, отличающиеся только левой частью, поскольку требуется ограничить эффект (в высшей степени необходимого) слова 'каждого' в ПР. Введем следующее условие: Пусть $\mathcal{A}$ обозначает любой элемент словаря V_m. Тогда наличие в ВВГ метабуквы $\mathcal{A}'$ подразумевает, что

[ж] Технические условия настоящего издания не позволяют соблюсти это условие, но все строчки, состоящие из представлений, помечены с помощью (ж). (Прим.переводчика)

$\mathcal{A}' :: \mathcal{A}.$

есть элемент R_m.

Таким образом, наличие V" влечет за собой метаправило

V" :: V'.

которое в свою очередь влечет за собой метаправило

V' :: V.

так что V, V' и V" имеют одинаковые терминальные порождения над V_0.

3. Пример двухуровневой грамматики для безъязыкового программирования

Пригодность безъязыкового программирования покажем на примере следующей двухуровневой грамматики.

Метабуквы суть, в первую очередь, H, J, K, L, M, N, P, Q, R, V, W, а кроме того, при наличии метабуквы $\mathcal{A}$ можно ввести метабукву $\mathcal{A}'$

Ортобуквы суть n, p, s, 9,8,7,6,5,4,3,2,I,0,(,).

Исходный набор метаправил таков:

МП 1 H :: 9;8;7;6;5;4;3;2;1;0.
МП 2 J :: HJ;.
МП 3 K :: n;p.
МП 4 N :: nN;.
МП 5 P :: pP;.
МП 6 M :: nN;pP;.
МП 7 Q :: HQ;KQ;.
МП 8 V :: sM.
МП 9 W :: sQ;(R).
МП 10 L :: VL;.
МП 11 R :: WR;.

Терминальные буквы – это терминальные метапорождения метабуквы V.

Исходным словом служит s0.

Гиперправила, наконец, таковы:

```
PII 1  L(R')R : LR'R.
PII 2  s0 : sp25sn8spPs1ns10sn9sn2sn41sn2sp25s1ns10sn2sn36sn2spPs50sp25s41s1ns10sps64ps2.
PII 3  Ls00R : .
PII 4  LsKHJR : Lss0KHJR.
PII 5  LVs0KHJR : LVsppppppppps42s00KH0123456789s40s0KJR.
PII 6  LVs0KR : LVR.
PII 7  LsM00KHH'JR : LsKM00KHJR.
PII 8  LsM00KHHJR : LsMR.
PII 9  LVs1MR : LVss24s4sn41sMVs03ps9s01s01R.
PII 10 Ls01PsP'L's01NR : LsP'sp10s34s4(sP'sp10s41s9(s01pP)L's01N)
         (sPss22s4(s01sPsP'L's01nN)(sP'L'snN))R.
PII 11 LVs2R : V , LR.
PII 12 LsKMs3M'R : LsMsKs4s3nM's3pM'R.
PII 13 Lss3M'R : LsM'R.
PII 14 LVs03KR : LVVs4( )s3sKs4( )s3R.
PII 15 LsPs4WW'R : LW'R.
PII 16 LsNs4WW'R : LW'R.
PII 17 LVs5(WW'R')R : LVss32s4W(Vsps41s5(W'R'))R.
PII 18 LVs5(W)R : LWR.
PII 19 Ls6pPVR : LVs6PR.
PII 20 Ls6pPs6HMWR : LR.
PII 21 LVV's6nNR : LVs6NV'R.
PII 22 Vs6nNLs6HMWR : VLR.
PII 23 Ls7WW'R : LWs4(W's7WW')( )R.
PII 24 LVV'V"s8WW'R : LV'ss36s4sp(VV"V's4s32s34)VV'V"s8pWW'R.
PII 25 LsMVV'V"s8KWW'R : LsMs4(sKs4(WVV'V"s8n)(W'VV's40V'V"s8)WW')sKR.
PII 26 LVs9WR : LWVR.
PII 27 LVs10R : LR.
PII 28 LVs11R : LVVR.
PII 29 LsMs12WR : LWMR.
PII 30 LVs13R : LVs4spsnR.
PII 31 LVV's20R : LVV's21s11s3s21R.
PII 32 LVV's21R : LVV's24s4VV'R.
PII 33 LVV's22R : LVV's41s022R.
PII 34 LspPs022R : LspPR.
PII 35 LsNs022R : LsnNR.
PII 36 LVV's23R : LVR.
PII 37 LVV's24R : LV'Vs22R.
PII 38 LVV's25R : LV'R.
PII 39 LVV's26R : LVV's22VV's24s27R.
PII 40 LVV's27R : LVV's22s4VV'R.
PII 41 LVV's3HR : LVV's2Hs3R.
PII 42 LsMsM's40R : LsMM'R.
PII 43 LsnNpPR : LsNPR.
PII 44 LspPnNR : LsPNR.
PII 45 LVV's41R : LVV's3s40R.
PII 46 LVV's42R : LV'ss36s4s(V's4(VV'sps41s42Vs40)(VV's3s42s3))R.
PII 47 LVV's43R : LsVs03pV's03ps043Vs4(V's4( )(sns64s3))(V's4(sns64ps3)s3)R.
PII 48 LsP"sPspP's043R : LsPspP's34s4(spP"sPspP's41spP's043)(sP"sP)R.
PII 49 LVsPs44R : LsPss36s4sp(VsPsps41s44Vs42)R.
```

```
ГП50  LVV'spPs45R : LVV'V's40s03pspPs22s4(V's13s40)( )R.
ГП51  LVV's40pPR : LVV's40R.
ГП52  LVV's41pPR : LVV's41R.
ГП53  LVV's42pPR : LVV's42spPs60s43spPs60s45R.
ГП54  LVV's43pPR : LVspPs60s42V's43spPs60s45R.
ГП55  LVV's44pPR : LV'ss36s4(spPs60)(V's4(VV'sps41s44pPVs42pP) (spPs60VV's3s44pPs43pP))R.
ГП56  LspPs50R : LspPsppspps050R.
ГП57  LspPsppP'sppP''s050R : LsppP'sppP''s43sns64s10ss22 s4(spPsppP'spppP''s050)
          (sppP'sppP''s22s4(spPspppP'spps050)(spPsps22s4(sPspppP'spps050)sppP'))R.
ГП58  LsM's6HMWR : sM'ss22s4(s6M'L)(Ls6M')s6HMWR.
ГП59  LVs6L's60R : LVL'VR.
ГП60  LVs6L'V's61R : LV'L'R.
ГП61  Ls6L'V's62R : LV'L'R.
ГП62  Ls6L's63KR : LsKs4L'LL'R.
ГП63  LVs6L's64MWR : LVWsMss36s4(L')(s6ML's64MW)R.
```

4. Интерпретации

Двухуровневая грамматика определяет некоторую формальную игру с буквами, не требуя для нее какой-либо интерпретации. Поэтому мы могли бы оставить читателя со списками метаправил и гиперправил, не давая больше никаких объяснений. Однако читателю могло бы быть неясно, для чего эти правила нужны, и он мог бы испытать трудности при рассмотрении их действия. Мы поэтому поможем ему "интерпретациями" (в этом разделе) и "разъяснениями" (в разделе 5), но он должен иметь в виду, что эти интерпретации и разъяснения не имеют определяющей силы и служат всего лишь комментариями.

Терминальные порождения метабуквы W называются "слогами". Слог - это либо "значение", либо "операция". Значения суть терминальные порождения метабуквы V, то есть

```
...,snnn, snn, sn, s, sp, spp, sppp,...,
```

так что операциями будут, например

```
s23, s3n, sp25, spppnn, (sn s64 s10).
```

Каждая (возможно, пустая) последовательность слогов есть терминальное порождение метабуквы R (от right - правый), а если брать более частный случай - каждая (возможно, пустая) последовательность значений есть терминальное порождение метабуквы L (от left - левый). Идея заключается в том, что "слово" состоит, как правило, из некоторой, быть может пус-

той, последовательности слотов. Гиперправила определят тогда, как слово – посредством первой операции в своем контексте – порождает другое слово. Положение первой операции задается "именем" $\mathcal{F}$ положение любого значения слева от первой операции задается именем, которое "указывает на" значение, занимающее это положение. Имена эти, если считать их слева направо, суть

$$S_p,\ S_{pp},\ S_{ppp},\ldots,\mathcal{F}$$

и в то же время, считая справа налево,–

$$\ldots,S_{nnn},\ S_{nn},\ S_n,\ S,\ \mathcal{F}.$$

На каждое значение, находящееся слева от первой операции, указывают, таким образом, два имени – "положительное" и "неположительное". Пара, образованная именем и значением, на которое это имя указывает, называется "переменной". Переменная с положительным именем есть "описанная" переменная, то есть такая переменная, имя которой не зависит от положения, задаваемого именем $\mathcal{F}$; переменная с неположительным именем есть "стековая" переменная, то есть такая переменная, имя которой зависит от этого положения.

Значения можно "интерпретировать", например, в смысле Алгола 68, то есть как целые числа, вещественные числа, истинностные значения, имена и т.д. Интерпретация приведенных выше значений как целых чисел дает

..., -3,-2,-I,0,I,2,3,...

Интерпретация их как вещественных чисел дает

..., -3/B, -2/B, -I/B, 0/B, I/B, 2/B, 3/B,...,

где "база" B обозначает некоторое целое число, большее единицы.

Интерпретация тех же значений как истинностных значений в многозначной логике дает

...,ллл, лл, л, н, и, ии, иии,...,

где л означает 'ложь', и – 'истина', а н – 'неопределено'.

Интерпретация этих значений как имен (при заданном слове)

дает имена S_{nnn}, S_{nn}, S_n, S, S_p, S_{pp}, S_{ppp},... ,
которые были определены выше.

Какая из этих интерпретаций более подходит в каждом данном случае,- это зависит от первой операции и ее контекста.

Терминальные буквы грамматики становятся определенными значениями, если они стоят сами по себе, то есть не как слоги в более длинном слове. Строго говоря, интерпретировать терминальные буквы необязательно, так как грамматик никогда не использует представления терминальных букв;напротив того, программист пишет программу на некотором языке программирования, используя только представления терминальных букв грамматики, определяющей этот язык. Полезнее, однако, будет дать список представлений, показывающий некоторую интерпретацию терминальных букв. В списке этом мы вместо терминальной буквы, состоящей из s, за которым следует, например, 25 раз p, пишем просто sp25, а вместо терминальной буквы, состоящей из s, за которым следует, например, 8 раз n, - просто sn8. Это отнюдь не небрежность, как могло бы показаться на первый взгляд. Согласно гиперправилам ГП4-ГП8, операция sp25 действительно порождает s, за которым следует 25 раз p, а операция sn8 действительно порождает s, за которым следует 8 раз n. Список наш может быть таким:

sn42	×	sn41	—	sn40	+	sn36	=	sn34	⩾
sn32	⩽	sn29	}	sn28	{	sn26	≠	sn24	<
sn22	>	sn19	]	sn18	[	sn17	\	sn16	|
sn15	/	sn13	←	sn12	→	sn11	↑	sn10	↓
sn9	)	sn8	(	sn7	:	sn6	,	sn5	.
sn4	(новая страница)			sn3	(новая строчка)				
snn	(пробел)			sn	(невидимый)				
s	0	sp	1	spp	2	sp3	3	sp4	4
sp5	5	sp6	6	sp7	7	sp8	8	sp9	9
sp10	a	sp11	b	sp12	c	sp13	d	sp14	e
sp15	f	sp16	g	sp17	h	sp18	i	sp19	j
sp20	k	sp21	l	sp22	m	sp23	n	sp24	o
sp25	p	sp26	q	sp27	r	sp28	s	sp29	t
sp30	u	sp31	v	sp32	w	sp33	x	sp34	y
sp35	z	sp36	A	sp37	B	sp38	C	sp39	D
sp40	E	sp41	F	sp42	G	sp43	H	sp44	I
sp45	J	sp46	K	sp47	L	sp48	M	sp49	N
sp50	O	sp51	P	sp52	Q	sp53	R	sp54	S
sp55	T	sp56	U	sp57	V	sp58	W	sp59	X
sp60	Y	sp61	Z						

Выбор представлений для терминальных букв от s до sp 61 очевиден. Представления остальных терминальных букв частично подобраны из мнемонических соображений. Так как, например, s40 есть операция сложения двух целых чисел, то знак + выбран как представление для sn 40.

5. Разъяснения

Теперь подробнее объясним, как работают гиперправила для операций. Гиперправила будут рассматриваться не в том порядке, в каком они были даны в разделе 3, а в таком, чтобы всегда в правой части оказывались лишь уже рассмотренные операции или операция, рассматриваемая в данный момент. Сам факт, что это возможно, доказывает отсутствие в определении грамматики порочного круга.

Частично бывает так, что в некотором применении гиперправила, содержащего W, следовало бы заменить W не на один слог, а на последовательность слогов, например, на sns64s10. Мы заменяем его в таком случае на "составную операцию", в нашем примере на (sns64s10). Составная операция, будучи терминальным порождением метабуквы W, а значит слогом, не является в то же время терминальным порождением метабуквы V, а значит не является значением. Следовательно, она - операция. Если эта операция оказалась первой операцией, то общее правило

ГП 1 L(R')R:LR'R.

разлагает ее на составляющие слоги, после чего управление берет на себя гиперправило для новой первой операции.

В контексте LVs4WW'R операция s4 "выбирает" между W и W', то есть, грубо говоря, работает так же, как (V|W|W'):

ГП 15 LsPs4WW'R:LWR.

ГП 16 LsNs4WW'R:LW'R.

Здесь очевидной интерпретацией для V является истинностное значение. Поскольку наша логика многозначная, порожден-

ем V может быть s, интерпретируемое как 'неопределено'; тогда применимы оба гиперправила и остается неопределенным, произошел выбор в пользу W или в пользу W'.

В контексте LVs3R операция s3 "отрицает" V, то есть превращает s, за которым следует некоторое число букв p, соответственно n, в s, за которым следует такое же число букв n, соответственно p. Если V интерпретируется как целое или вещественное число, то отрицание означает изменение знака V, а если V интерпретируется как истинностное значение, то оно означает логическое отрицание. Определение этой операции сводится к применению операции s4:

ГП12 LsKMs3M'R:LsMsKs4s3nM's3pM'R.

ГП13 Lss3M'R:LsM'R.

Операцию s3 можно было бы определить и без обращения к s4, но это обошлось бы в четыре гиперправила вместо двух, а наша стратегия состоит в том, чтобы минимизировать число гиперправил.

В контексте LVs03KR операция s03p выдает abc V, а операция s03n выдает - abc V

ГП14 LVs03KR:LVVs4( )s3sKs4( )s3R.

В контексте LVs9WR операция s9 "переставляет" V и W:

ГП26 LVs9WR:LWVR.

В контексте LVs10R операция s10 "уничтожает" V:

ГП27 LVs10R:LR.

В контексте LVs11R операция s11 "удваивает" V:

ГП28 LVs11R:LVVR.

В контексте LsMs12WR операция s12 "украшает" W:

ГП29 LsMs12WR:LWWR.

В контексте LVs13R операция s13 "проецирует" V на sK:

ГП30 LVs13R:LVs4spsnR.

Одним из применений этой операции является преобразование многозначной логики в булеву логику; см. также ГП 50.

Обратимся теперь к простейшим арифметическим действиям над целыми числами.

В контексте LVVs40R операция s40 выдает V + V':

ГП42 LsMsM's40R:LsMM'R.

ГП43 LsnNpPR:LsNPR.

ГП44 LspPnNR:LsPNR.

В контексте LVV's41R операция s41 выдает V - V':

ГП45 LVV's41R:LVV's3s40R.

Имея теперь в своем распоряжении вычитание целых чисел, обратимся к операциям, вычисляющим отношения между значениями; значения эти можно интерпретировать по произволу как истинностные значения, как целые числа или как вещественные числа. Для некоторых из них естественнее первая интерпретация, для других - вторая или третья. Начнем с шестнадцати булевых операций, отбросив при этом ограничение, требующее двузначности аргументов и результата. Операции пронумерованы с s20 по s27 и с s30 по s37, но мы будем рассматривать их в таком порядке, чтобы читателю не приходилось заглядывать вперед. Контекстом здесь служит всегда LVV's2HR или, соответственно, LVV's3HR.

Операция s22 выдает V > V':

ГП33 LVV's22R:LVV's41s022R.

ГП34 LspPs022R:LspPR.

ГП35 LsNs022R:LsnNR.

Операция s24 выдает V < V':

ГП37 LVV's24R:LV'Vs22R.

Операция s21 выдает V & V' или min(V, V'):

ГП32 LVV's21R:LVV's24s4VV'R.

Операция s27 выдает V ∨ V' или max(V, V'):

ГП40 LVV's27R:LVV's22s4VV'R.

Операция s20 выдает (V & V')&¬(V& V'),то есть 'ложь' некоторого размера:

ГП31 LVV's20R:LVV's21s11s3s21R.

Операция s23 выдает свой первый операнд V:

ГП36 LVV's23R:LVR.

Операция s25 выдает овой второй операнд V':

ГП38 LVV's25R:LV'R.

Операция s26 выдает V ≠ V':

ГП39 LVV's26R:LVV's22VV's24s27R.

Операция s3H выдает ¬(VV's2H):

ГП41 LVV's3HR:LVV's2Hs3R.

Операции s20, s2I, s23, s25, s27, s30, s3I, s33, s35 и s37 могут выдавать s, то есть 'неопределено', а операции s22, s24, s26, s32, s34 и s36 не могут.

Рассмотрим теперь операции, имеющие дело с именами. В контексте LsM's6HMWR, где слог sM' лучше всего интерпретировать как имя SM' операция s6HM создает операцию s6M' и вычеркивает из слова слог sM' (но не самое себя), операция s6pP вставляется слева от $S\rho$, операция s6N вставляется справа от S:

ГП58 LsM's6HMWR:sM'ss22s4(s6M'L)(Ls6M')s6HMWR.

Вставленная операция s6pP перескакивает через значение, находящееся справа от нее, теряя при этом одно p, а вставленная операция s6nN перескакивает через значение, находящееся слева от нее, теряя при этом одно n. Поступая так, вставленная операция s6M' в конце концов либо остановится справа от SM' как s6, либо, если будет грозить выход ее за пределы L, исчезнет вместе с s6HMW.(См. ГП19 - ГП-22.) Действия, происходящие после того, как операция s6 остановилась справа от SM', зависят от H и M.

В контексте LsM's60R операция s60 "разыменовывает" sM':

ГП59 LVs6L's60R:LVL'VR.

В контексте LV'sM's61R операция s61 "присваивает" значение V' имени sM':

ГП60 LVs6L'V's61R:L'V'L'R.

В контексте LV'sM's62R операция s62 "вставляет" V' справа от SM':

ГП61 Ls6L'V's62R:LV'L'R.

В контексте LsM's63KR операция s63p удваивает ту часть L, которая находится справа от SM', а операция s63n удваивает ту часть L, которая заканчивается на SM':

ГП62 Ls6L's63KR:LsKs4L'LL'R.

В контексте LsM's64MWR операция s64 вызывает применение операции W к SM', а операция s64p, соответственно s64n, вызывает применение операции W к SM' и ко всем слогам L, расположенным справа, соответственно слева, от SM':

ГП63 LVs6L's64MWR:LVWsMss36s4(L')(s6ML's64MW)R.

Имея теперь в своем распоряжении отношения и действия над именами, мы можем определить дальнейшие действия над целыми числами:

В контексте LVV's42Rоперация s42 выдает V×V':

ГП46 LVV's42R:LV'ss36s4s(V's4(VV'sps41s42Vs40)(VV's3s42s3))R.

В контексте LVV's43R операция s43 выдает частное V÷V' и - вслед за ним-остаток V÷×V':

ГП47 LVV's43R:LsVs03pV's03ps043Vs4(V's4()(sns64s3))(V's4(sns64ps3)s3)R.

ГП48 LsP"sPspP's043R:LsPspP's34s4(spP"sPspP's41spP's043)(sP"sP)R.

В контексте LVsPs44R операция s44 выдает V ↑ sP:

ГП49 LVsPs44R:LsPss36s4sp(VsPsps41s44Vs42)R.

В контексте LVV'spPs45R операция s45 "округляет" V, если 2 абс V' > spP:

ГП50 LVV'spPs45R:LVV'V's40s03pspPs22s4(V's13s40)()R.

Рассмотрим действия над вещественными числами. Математическая интерпретация значения как вещественного числа состоит в интерпретации его как целого числа, деленного на некоторую "базу", то есть положительное число, большее единицы, например I0 ↑ I8. Операция s4HpP интерпретирует слог $S_P P$ к.к такую базу.

В контексте LVV's40pPR операция s40pP выдает V + V':

ГП51 LVV's40pPR:LVV's40R.

В контексте LVV's41pPR операция s41pP выдает V - V':

ГП52 LVV's41pPR:LVV's41R.

В контексте LVV's42pPR операция s42pP выдает V×V':

ГП53 LVV's42pPR:LVV's42spPs60s43spPs60s45R.

В контексте LVV's43pPR операция s43pP выдает V/V':

ГП54 LVV's43pPR:LVspPs60s42V's43spPs60s45R.

В контексте LVV's44pPR операция s44pP выдает V ↑ V', где V' интерпретируется как целое число:

```
ГП55 LVV's44pPR:LV'ss36s4(spPs60)(V's4(VV'sps41s44pPVs42pP)
     (spPs60VV's3s44pPs43pP))R.
```

До сих пор мы рассматривали только общие, широко применимые операции, подобные тем, которые входят в стандартное вступление Алгола 68. Дадим теперь пример весьма специфичной операции, подобную которой можно было бы встретить в некотором собственном вступлении Алгола 68.

В контексте LspPs50R операция s50 выдает spP-е по счету простое число:

```
ГП56 LspPs50R:LspPsppspps050R.
ГП57 LspPsppP'sppP"s050R:LsppP'sppP"s43sns64s10ss22s4
     (spPsppP'spppP"s050)

     (sppP'sppP"s22s4(spPspppP'spps050)(spPsps22s4
     (sPspppP'spps050)sppP'))R.
```

Правило ГП56 вводит два новых значения, первоначально равных 2. Первое из этих значений есть кандидат на роль простого числа, а второе – пробный делитель для проверки простоты первого значения. Вводится также новая операция s050. Правило ГП57 определяет остаток от деления кандидата на пробный делитель. Если этот остаток положителен, то пробный делитель не является настоящим делителем и испытывается следующий пробный делитель. Если остаток есть нуль, то правило выясняет, больше ли кандидат, чем пробный делитель. Если это так, то кандидат – составное число и испытывается следующий кандидат, причем пробный делитель вновь полагается равным 2. В противном случае кандидат есть простое число и правило рассматривает число простых чисел, которые еще надо найти. Если это число больше единицы, то оно уменьшается на единицу и та же работа начинает производиться со следующим кандидатом. В противном случае кандидат есть искомое простое число.

Определим теперь некоторые конструкции, подобные тем, которые имеются в Алголе 68.

В контексте LVs5(WR')R операция s5 "выбирает" слог из

последовательности WR' грубо говоря так же, как это делает оператор (V|W, R'):

ГП17 LVs5(WW'R')R:LVss32s4W(Vsps41s5(W'R'))R.

ГП18 LVs5(W)R:LWR.

Естественно здесь интерпретировать V как целое число. Слоги в WR' нумеруются числами от 0 до, например, k. Если $V \leq 0$, то выбирается W; если $V \geq k$, то выбирается k слог. В остальных случаях выбирается V-й слог.

В контексте Ls7WW'R операция s7 "повторяет" W' под контролем условия W, так же, как это делает оператор

пока W цк W' кц:

ГП23 Ls7WW'R:LWs4(W's7WW')( )R.

В контексте LVV'V"s8WW'R операция s8 повторяет W' под контролем граничного условия и условия W, так же как это делает оператор от V шаг V' до V" пока W цк W' кц :

ГП24 LVV'V"s8WW'R:LV'ss36s4sp(VV"V's4s32s34)VV'V"s8pWW'R.

ГП25 LsMVV'V"s8KWW'R:LsMs4(sKs4(WVV'V"s8n)(W'VV's40V'V"s8)
WW')sKR.

В отличие от цикла Алгола 68 здесь выдается непустой результат: sp, если цикл прерван условием по V, и sn - если условием по W.

Рассмотрим далее операции, переводящие из обычных десятичных обозначений в наши унарные обозначения и наоборот.

В контексте LsKHJR операция spHJ, соответственно snHJ выдает s, за которым следует HJ раз p, соответственно n:

ГП4 LsKHJR:LssOKHJR.

ГП5 LVsOKHJR:LVsppppppppppps42sOOKH0123456789s40sOKJR.

ГП6 LVsOKR:LVR.

ГП7 LsMOOKHH'JR:LsKMOOKHJR.

ГП8 LsMOOKHHJR:LsMR.

Правило ГП4 вводит s, то есть нуль, как предварительный результат, а также вводит операцию sOKHJ. Правило ГП5 умножает предварительнный результат на десять, с помощью новой операции sOOKH0123456789 прибавляет к нему число, "изображаемое" старшей десятичной цифрой H с учетом знака, задаваемого K, а затем повторно применяет ГП5 для обработки следующей цифры из J, если есть такая. ГП7 и ГП8 имеют дело с этой новой операцией. Заметим, что в ГП5 она не содержит букв p и n между s и OO, и что H стоит перед цифрой 0. Применение правила ГП7 вставляет после s букву p или n, так что теперь между s и OO находится одна такая буква, и изымает цифру 0, следующую за H, так что теперь H стоит перед цифрой I. Вообще же, после некоторого числа применений правила ГП7 число букв p или n между s и OO будет равно числу, изображаемому цифрой H', перед которой стоит H. Дальнейшее применение правила ГП7 приводит в конечном счете в тупик, единственный в нашей грамматике, за отсутствием цифры H' после H. Однако, в ходе этого процесса H обязательно один раз оказывается перед самим собой, и тогда применимо правило ГП8, выдающее нужный результат, то есть значение, изображаемое цифрой H. После обработки всех цифр из HJ правило ГП6 выдает окончательный результат.

Операция sKHJ позволяет вводить в грамматику такие константы, как spI25 или sn256, подобные константам I25 и -256 в десятичных обозначениях.

В контексте LVs1MR операция s1M преобразует V, то есть унарное представление числа, в последовательность следующих значений: знак V, последовательные десятичные цифры V и указание на число этих цифр:

```
ГП9 LVs1MR:LVss24s4sn41sMVs03ps9s01s01R.

ГП10 Ls01PsPL's01NR:LsP'sp10s34s4(sP'sp10s41s9(s01pP)L's01N)
     (sPss22s4(s01sPsP'L's01nN)(sPL'snN))R.
```

Если V отрицательно, то знак есть sn4I; если V неотрицательно, то знак есть sM. Благодаря этому грамматик может осуществить, например, выбор между знаком плюс sn40, знаком

пробела snn и невидимым знаком sn, написав, соответственно, sn40s12sI, s1nn или s1n. Правило ГП9 вначале определяет знак, а затем помещает абсолютную величину V между операциями s01 и s01. Затем правило ГП10 постепенно разлагает V в последовательность десятичных цифр, разумеется представленных в унарной форме, используя при этом первое s01 как хранилище для промежуточных значений частного от деления на десять, а второе s01 как хранилище для промежуточных значений указания на число полученных десятичных цифр. Окончательное указание есть отрицание этого числа. Вскоре станет ясно, почему мы не строим само это число. Коли грамматик не желает иметь это указание, то он может тут же уничтожить его с помощью s10.

В контексте LVs2R операция s2 "выводит" V:

```
ГП11 LVs2R:V, LR.
```

Это единственное наше гиперправило, в которое входит запятая. V, являясь членом терминального словаря V_t, не играет больше роли в процессе порождения и может быть заменен на свое представление. Это наш эквивалент того,что в языках программирования называется выводом.

Так как LVs1MR порождает знак, за которым следует некоторое число десятичных цифр, за которыми следует отрицание этого числа, то LVs1Ms64ps2R выводит V в десятичном представлении; таким образом, Lsp25s2R выводит

P (ж)

а Lsp25s1ns64ps2R выводит

25 (ж)

В контексте Ls00R операция s00 "заканчивает" процесс порождения:

```
ГП3 Ls00:.
```

Наконец, в контексте s0 операция s0, то есть исходное слово, порождает ту конкретную задачу, которую рассматривает грамматик. Весьма конкретным гиперправилом будет, в виде примера:

```
ГП2 s0:sp25sn8spPs1ns10sn9sn2sn41sn2sp25s1ns10sn2sn36sn2spP
    s50sp25s41s1ns10sps64ps2.
```

Заметим сперва, что в правую часть входит метабуква P, не входящая в левую часть; ее терминальное метапорождение можно поэтому выбрать произвольно. Это наш эквивалент того, что в языках программирования называется вводом. Допустим, что для этого терминального порождения мы выбрали последовательность из ста двадцати четырех букв p. Правая часть начинается с таких безобидных операций как sp25, выдающей s с двадцатью пятью буквами p и sn 8, выдающей s с восемью буквами n. Первая интересная комбинация - это spPs1ns10, порождающая sn spsppspppppp. Вторая интересная комбинация - это spPs50 ,порождающая s с шестьюстами девяносто одним p,так как сто двадцать пятым простым числом является число 691, что следует из ГП56. Далее, комбинация spPs50sp25s441s1ns10, порождает spppppppspppppppspppppp, так как 691-25=666. Наконец, комбинация sps64ps2, выбирает самый левый слог и выводит его вместе со всеми слогами, находящимися справа от него. Таким образом порождено

p (I 2 5) - 2 5 = 6 6 6 (ж)

- тот же результат, что и в собственно программе Алгола 68 из Введения, но конечно, на этот раз строго выведенный.

Литература

1. Wijngaarden, Van A. Orthogonal design and description of a formal language, MathematicalCentre, Amsterdam, MR76 (1965).
2. Пересмотренное сообщение об Алголе 68.-М.: Мир, 1979.
3. Wijngaarden, Van A. Thinking on two levels, in Proc. Bicentennial Congress of the Wiskundig Genootschap, part 2, P.C. Baayen, D. van Dulst & J. Oosterhoff, eds., Mathematical Centre, Amsterdam, Mathematical Centre Tracts, Vol. 101 (1979) 417-428.