УДК 681.322.06+681.322.04:800.92

ЭКВИВАЛЕНТНЫЕ ПРЕОБРАЗОВАНИЯ ПРОГРАММ НА РЕФАЛе

Турчин В. Ф.

Достоинства всякого формализованного языка определяются не только тем, сколь удобен он для непосредственного использования человеком, но и тем, в какой степени тексты на этом языке поддаются формальным преобразованиям. Эта вторая сторона важна потому, что в системе формализованных языков, образующей каркас математизированного знания, ни один язык (во всяком случае, если иметь в виду не только формальную схему, но и парадигму использования этой схемы) не может считаться верхней ступенькой иерархии.

Важнейший шаг в развитии математики – это *метасистемный переход*, то есть переход от некоторой формальной системы S к метасистеме S^*, для которой система S и аналогичные системы являются предметом исследования, преобразования, порождения. Таковы, например, переходы от арифметики к алгебре, от конкретных алгоритмов к алгоритмическим языкам, от словесных доказательств к математике. Это примеры наиболее крупномасштабных переходов. Но и в рамках более узкой дисциплины движение происходит подобным же образом. В программировании и теории алгоритмов переходы от машинных программ к языку ассемблера и от языка ассемблера к машинно-независимым языкам также представляют создание метасистем (хотя и с более узким кругом задач, чем в предыдущих примерах). Научившись писать алгоритмы определенным образом, мы тут же начинаем задумываться, нельзя ли этот образ действия формализовать и передать машине. Первым шагом на этом пути является разработка системы эквивалентных преобразований.

Чрезвычайно мощную систему эквивалентных преобразований можно построить, если наложить на язык РЕФАЛ [1] определенные ограничения, а именно:

1. Исключаются символы обмена и связанные с ними возможности менять состояние поля памяти рефал-машины в процессе ее работы, включая процедуры *закапывания* и *выкапывания*.

2. Исключаются свободные переменные терма.

3. Исключаются рекурсивные переменные выражения. Из рекурсивных переменных символа разрешается использование лишь переменных с детерминативом 'ИЗ' в спецификаторе.

4. Запрещается использование открытых и повторных свободных переменных выражения.

Мы будем называть этот подъязык ограниченным РЕФАЛом. Его синтаксис будет слегка модифицирован в соответствии с введенными ограничениями. Хотя во многих случаях использование ограниченного РЕФАЛа, вместо полного заметно удлиняет запись, она все же продолжает оставаться достаточно компактной и ясной. Впрочем, для практического применения эквивалентных преобразований не обязательно отказываться от изобразительных средств полного РЕФАЛа, ибо перевод текста с полного РЕФАЛа в ограниченный может быть выполнен автоматически.

Так как ограничения 1-4 существенно упрощают семантику языка, а также учитывая, что нам понадобится более высокий уровень строгости, мы дадим новое описание ограниченного РЕФАЛа, независимое от описания полного РЕФАЛа.

1. Описание ограниченного РЕФАЛа

Синтаксис

Первое понятие, которое нам необходимо ввести для описания языка РЕФАЛ, - понятие о знаке. Оно является абстрактным, или общим, т.е. утверждение "это знак" может быть верным для нескольких различных предметов. Чтобы решить, является ли данный предмет знаком, надо прежде всего указать некоторое число предметов, называемых эталонными знаками. Например, набором эталонных знаков могут служить следующие пять предметов:

$$А \quad Б \quad В \quad = \quad *$$

Предмет является знаком в том и только в том случае, если может быть отождествлен как один из эталонных знаков. Понятие отождествления по отношению к знакам опирается непосредственно на распознавательные способности человека и не требует словесного определения. Далее, говоря о знаках, мы будем указывать, с каким именно эталонным знаком отождествляется данный предмет. Это достигается либо помещением непосредственно за словом "знак" еще одного предмета, отождествляемого, как тот же эталонный знак, либо использованием словесного наименования эталонного знака. Например, в конструкции "Б" "=" предмет, заключенный в первую пару кавычек, есть знак Б , а предмет из второй пары кавычек - знак равенства. Так как отношение отождествления между предметами является рефлексивным, симметричным и транзитивным, в качестве эталонного знака может использоваться любой из предметов, отождествляемых, как один и тот же знак.

Наша терминология и стоящая за ней система понятий близка к терминологии А.А.Маркова в его "Теории алгоритмов" [2]. Однако мы не говорим об абстрактном знаке (или букве), так как обычное понятие знака (как и все, впрочем, понятия языка, выражаемые именами нарицательными) уже абстрактно, и чтобы сконструировать конкретное понятие приходится пользоваться (абстрактным же) понятием предмета в сочетании с указательными местоимениями: "тот предмет, который ..." и т.д.

Знаки интересуют нас потому, что они используются в знаковых машинах. Под машиной вообще мы понимаем полностью детерминированную материальную систему, т.е. такую систему, состояние которой в данный момент времени однозначно определяет состояние в любой из следующих моментов времени. Некоторые части машины, на которые в первую очередь обращено наше внимание, мы называем наблюдаемыми частями; все остальное называем устройством машины. Если наблюдаемые части машины представляют собой последовательности (упорядоченные множества) знаков, мы называем машину знаковой.

Знаковая машина может быть выполнена из различных материалов и основана на различных принципах. Устройство ее может быть, например, механическим или электрическим, наблюдаемые части могут состоять из барабанов с нарисованными на них знаками или из неоновых лампочек. Наконец, в качестве устройства знаковой машины может выступать человек, делающий формальные выкладки. Наблюдаемыми частями являются здесь листы бумаги, на которых он пишет знаки. Именно на этот случай мы будем ориентироваться, описывая знаковые машины. Мы будем описывать действия машины таким образом, чтобы человек мог их в точности воспроизвести.

Если задан набор знаков и описана знаковая машина, у которой в наблюдаемых частях появляются только эти знаки, то говорят, что описан полностью формализованный язык. Чаще всего в наблюдаемых частях машины появляются комбинации знаков языка не произвольные, а построенные определенным образом. Эти комбинации мы будем называть языковыми объектами. Правила построения и распознавания языковых объектов называют синтаксисом данного языка. В рамках одного языка обычно различают несколько типов языковых объектов, они называются синтаксическими типами или конструкциями.

Знаки ограниченного РЕФАЛа делятся на собственные и объектные. Собственных знаков девять:

- рефал-кавычка (или символьная скобка),

( ) – круглые (или структурные) скобки,
k – знак конкретизации,
$\perp$ – конкретизационная точка,
§ – параграф,
$\sim$ – тильда,
s – признак свободной переменной символа,
e – признак свободной переменной выражения.

Объектные знаки определяются следующим эталонным набором:

А Б В Г Д Е Ж З И К Л М Н О П Р С Т

У Ф Х Ц Ч Ш Щ Ь Ы Ъ Э Ю Я

D F G I J L N Q R S U V W Z

0 1 2 3 4 5 6 7 8 9

. , ; : ! ? *

\+ − × / = ≠ < ≤ > ≥ *V* & *U* ≡

Сюда входят заглавные буквы русского и латинского алфавита, цифры и обычные математические знаки (набор которых может быть расширен). Заметим, что латинские и русские буквы, совпадающие по начертанию, не различаются.

Знаковую машину, определяющую язык РЕФАЛ, мы будем называть рефал-машиной. При описании синтаксиса мы в качестве метаобъектов, изображающих различные последовательности знаков, будем использовать заглавные рукописные буквы латинского алфавита. Пустой объект будем иногда обозначать <пусто>.

Символом в языке РЕФАЛ мы называем объектный знак или конструкцию '$\mathcal{P}$', где $\mathcal{P}$ – непустая последовательность объектных знаков. В процессе работы рефал-машины символ рассматривается как нечто целое и не может быть разложен на составные части.

Пусть $\mathcal{Z}$ – произвольный объектный знак, а $\mathcal{R}$ – непустая последовательность символов. Введем следующие три конструкции:

$e\,\mathcal{Z}$ – свободная переменная выражения,
$s\,\mathcal{Z}$ – неспецифицированная свободная переменная символа,
$s(\mathcal{R})\mathcal{Z}$ – специфицированная свободная переменная символа.

Каждая из этих конструкций может быть упомянута как свободная переменная, или просто переменная. Знак $\mathcal{Z}$ называется индексом переменной, конструкция $(\mathcal{R})$ – ее спецификатором.

Синтаксические понятия терма и выражения введем с помощью следующих опирающихся друг на друга опреде-

лений. Терм – это либо символ, либо свободная переменная, либо конструкция $(\mathcal{E})$, либо конструкция $k\,\mathcal{E}\perp$, где $\mathcal{E}$ – произвольное выражение. Выражение – это последовательность термов (быть может, пустая), на которую накладывается одно ограничение: она не должна содержать двух свободных переменных, имеющих одинаковые индексы, но разные признаки.

Выражение $\mathcal{E}'$ называется подвыражением выражения $\mathcal{E}$, если $\mathcal{E}$ можно представить в виде $\mathcal{C}_1\,\mathcal{E}'\,\mathcal{C}_2$, где $\mathcal{C}_1$ и $\mathcal{C}_2$ – последовательности знаков (не обязательно являющиеся выражениями).

Выражение, не содержащее свободных переменных, называется рабочим выражением. Выражение, не содержащее знаков k, называется типовым выражением. Выражение, не содержащее ни свободных переменных, ни знаков k, называется объектным выражением.

Типовое выражение называется L – выражением, если:
1) ни одно его подвыражение не содержит более чем одну свободную переменную выражения, не входящую в скобки;
2) ни одна свободная переменная выражения не входит в него более одного раза.

Примеры L –выражений:

$A\ B\ C$

$'RS'(s(+-)1e2(Ae3B))s1$

$(e1)(e2)(e3)(e4)\,e5$

Примеры типовых выражений, не являющихся L–выражениями:

$$e1\ +\ e2$$
$$s1\ e2(s3\,e2)$$

Из определения L –выражения сразу следует:

Теорема 1. Каждое подвыражение L –выражения является L –выражением.

Областью действия знака k называется выражение, начинающееся непосредственно за этим знаком k и кончающееся перед парным знаком $\perp$. Ведущим знаком k в некотором выражении называется первый (слева) знак k, в области действия которого не содержится ни одного знака k.

Предложением ограниченного РЕФАЛа называется конструкция

$$\S\ \mathcal{C}\ k\,\mathcal{L}\ \sim\mathcal{R}$$

где $\mathcal{C}$ – последовательность (возможно, пустая) объектных знаков, называемая комментарием;

$\mathcal{L}$ – L – выражение; называемое левой частью;

$\mathscr{R}$ – выражение, называемое правой частью предложения.

Кроме того, $\mathscr{L}$ и $\mathscr{R}$ должны удовлетворять следующим условиям:

А1. Левая часть должна начинаться с символа. Он называется детерминативом предложения.

А2. В предложении не должно встречаться двух свободных переменных с различными признаками, но совпадающими индексами. Если в пределах одного предложения встречаются несколько вхождений свободных переменных, у которых совпадают признаки и индексы, то мы будем говорить о них, как о различных вхождениях одной и той же переменной.

А3. В правую часть предложения могут входить только те переменные, которые входят в левую часть.

Синтаксическое отождествление

Важную роль в языке РЕФАЛ играет понятие синтаксического отождествления. Мы говорим, что объектное выражение $\mathscr{E}_o$ может быть отождествлено как типовое выражение $\mathscr{E}_t$, если свободные переменные в $\mathscr{E}_t$ могут быть заменены с соблюдением указанных ниже правил на такие выражения, называемые их значениями, что $\mathscr{E}_t$ совпадает с $\mathscr{E}_o$. Правила таковы:

В1. Значением свободной переменной выражения может быть любое выражение.

В2. Значением неспецифицированной свободной переменной символа может быть любой символ, а для специфицированной переменной – символ, входящий в спецификатор.

В3. Значения всех вхождений одной и той же переменной должны совпадать.

Если типовое выражение является L-выражением, то синтаксическое отождествление, когда оно возможно, осуществляется лишь при единственном наборе значений свободных переменных, о чем свидетельствует следующая теорема.

Теорема 2. Пусть $\mathscr{E}_\ell$ – L-выражение, которое при замене вхождений свободных переменных на некоторые значения, удовлетворяющие условиям *В1-В3*, переходит в объектное выражение $\mathscr{E}_o$. Тогда не существует другого набора значений, удовлетворяющего условиям *В1-В3*,, используя который можно перевести $\mathscr{E}_\ell$ в $\mathscr{E}_o$.

Эту теорему мы докажем путем построения алгоритма, который по заданным $\mathscr{E}_\ell$ и $\mathscr{E}_o$ определяет, возможно ли отождествление $\mathscr{E}_o$ как $\mathscr{E}_\ell$, и в случае положительного ответа однозначно определяет значения свободных переменных в $\mathscr{E}_\ell$.

Те термы, последовательностью которых является, по определению, данное выражение, будем называть его 0-терма-

ми. Если выражения $\mathcal{E}_1$ и $\mathcal{E}_2$ совпадают, то первый 0-терм выражения $\mathcal{E}_1$ должен совпадать с первым 0-термом выражения $\mathcal{E}_2$, а последний 0-терм $\mathcal{E}_1$ – с последним 0-термом $\mathcal{E}_2$. Мы будем пользоваться этим для сравнения всевозможных объектных выражений $\mathcal{E}_\ell^o$, получающихся из $\mathcal{E}_\ell$ заменой свободных переменных на некоторые значения, с выражением $\mathcal{E}_o$. Из 0-термов, входящих в $\mathcal{E}_\ell$, только свободная переменная выражения, т.е. терм вида $e\mathcal{Z}$, может при замене переменных на их значения перейти в произвольное выражение; каждый из остальных термов переходит в терм. Поэтому мы будем сначала просматривать 0-термы выражения $\mathcal{E}_\ell$ слева направо, сравнивая их с соответствующими 0-термами выражения $\mathcal{E}_o$, а если дойдем до терма $e\mathcal{Z}$, то будем двигаться от конца выражения $\mathcal{E}_\ell$ справа налево, пока снова не дойдем до того же терма $e\mathcal{Z}$ (ибо он, по определению L-выражения, может быть лишь в единственном числе. Сравнение терма $\mathcal{T}_\ell$ из $\mathcal{E}_\ell$ с термом $\mathcal{T}_o$ из $\mathcal{E}_o$ и попытка придания свободным переменным в $\mathcal{T}_\ell$ таких значений, чтобы $\mathcal{T}_\ell$ совпадал с $\mathcal{T}_o$, будет называться *проектированием* $\mathcal{T}_\ell$ на $\mathcal{T}_o$. Распространим это понятие и на выражения. Проектирование $\mathcal{E}_\ell$ на $\mathcal{E}_o$ – это в точности то же самое, что отождествление $\mathcal{E}_o$ как $\mathcal{E}_\ell$; в одних случаях удобнее пользоваться термином "отождествление", в других – термином "проектирование".

Опишем теперь более подробно процесс проектирования. Рассмотрим сначала случай, когда $\mathcal{E}_\ell$ не содержит скобок, если не считать скобок, ограничивающих спецификатор. В дальнейшем под скобками, входящими в $\mathcal{E}_\ell$, мы всюду будем понимать скобки, ограничивающие терм, но не спецификатор. Если $\mathcal{E}_\ell$ пусто, то вопрос об отождествлении решается тривиально: оно возможно тогда и только тогда, когда $\mathcal{E}_o$ тоже пусто. Если $\mathcal{E}_\ell$ не пусто, то его можно представить в виде $\mathcal{T}_\ell\,\mathcal{E}_{\ell 1}$, где $\mathcal{T}_\ell$ – первый 0-терм выражения $\mathcal{E}_\ell$. Допустим, что $\mathcal{T}_\ell$ не есть $e\mathcal{Z}$. Тогда, если $\mathcal{E}_o$ пусто, отождествление невозможно. Если оно не пусто, его можно представить в виде $\mathcal{T}_o\,\mathcal{E}_{o1}$, где $\mathcal{T}_o$ – первый 0-терм выражения $\mathcal{E}_o$. Теперь продолжение проектирования возможно только при условии, что можно спроектировать $\mathcal{T}_\ell$ на $\mathcal{T}_o$. Здесь могут представиться следующие три случая.

С1. $\mathcal{T}_\ell$ есть символ $\mathcal{S}$. Проектирование возможно только если $\mathcal{T}_o$ также есть $\mathcal{S}$.

С2. $\mathcal{T}_\ell$ имеет вид $s\mathcal{Z}$, где $\mathcal{Z}$ – объектный знак. Проектирование возможно только, если терм $\mathcal{T}_o$ есть символ $\mathcal{S}$ (а не выражение в скобках) и только при значении переменной $s\mathcal{Z}$, тождественном $\mathcal{S}$.

С3. $\mathcal{T}_\ell$ имеет вид $s(\mathcal{P})\mathcal{Z}$, где $\mathcal{P}$ – список символов.

Проектирование возможно только, если терм $\mathcal{T}_0$ есть символ $\mathcal{S}$, входящий в список $\mathcal{P}$, и только при значении переменной $s\mathcal{Z}$, тождественном $\mathcal{S}$.

Если проектирование $\mathcal{T}_\ell$ на $\mathcal{T}_0$ возможно, то проблема отождествления $\mathcal{E}_0$ как $\mathcal{E}_\ell$ сводится к проблеме отождествления $\mathcal{E}_{01}$ как $\mathcal{E}_{\ell 1}$ с учетом значения $\mathcal{S}$, принятого переменной $s\mathcal{Z}$ в случаях *C2* и *C3*. Чтобы учесть это последнее обстоятельство, заменим в выражении $\mathcal{E}_{\ell 1}$ все вхождения вида $s\mathcal{Z}$ и вида $s(\mathcal{P}_1)\mathcal{Z}$, в том случае, если список $\mathcal{P}_1$ включает $\mathcal{S}$, на $\mathcal{S}$. Если в $\mathcal{E}_{\ell 1}$ есть хотя бы одно вхождение вида $s(\mathcal{P}_1)\mathcal{Z}$, где список $\mathcal{P}_1$ не включает $\mathcal{S}$, отождествление невозможно. Если таких вхождений нет, то выражение, полученное из $\mathcal{E}_{\ell 1}$ в результате указанной замены, обозначим через $\mathcal{E}'_{\ell 1}$, и проблема сводится к отождествлению $\mathcal{E}_{01}$ как $\mathcal{E}'_{\ell 1}$.

Если, отщепляя таким образом терм за термом, мы приходим к ситуации, когда первый 0-терм $\mathcal{T}_\ell$ имеет вид $e\mathcal{Z}$, мы приступаем к совершенно аналогичному движению справа налево, представляя $\mathcal{E}_\ell$ и $\mathcal{E}_0$ в виде $\mathcal{E}_{\ell 1}\mathcal{T}_\ell$ и $\mathcal{E}_{01}\mathcal{T}_0$. Если выражение $\mathcal{E}_\ell$ сводится к одному терму $e\mathcal{Z}$, то оно спроектируется на оставшееся выражение и переменная $e\mathcal{Z}$ примет, таким образом, единственно возможное значение.

Итак, для L-выражения $\mathcal{E}_\ell$, не содержащего скобок, наша теорема доказана. Докажем ее индукцией по числу пар скобок, встречающихся в $\mathcal{E}_\ell$. Допустим, что для выражения с числом пар скобок, не превышающим n, теорема доказана. Только что рассмотренный случай $n = 0$ будет служить нам базисом для индукции. Рассмотрим L-выражение $\mathcal{E}_\ell$, содержащее $n+1$ пару скобок.

Наличие скобок в $\mathcal{E}_\ell$ требует рассмотрения еще одного случая, в дополнение к случаям *C1*–*C3*, который может возникнуть при проектировании терма $\mathcal{T}_\ell$ на терм $\mathcal{T}_0$:

C4. $\mathcal{T}_\ell$ имеет вид $(\mathcal{E}_{\ell i})$, где $\mathcal{E}_{\ell i}$ – некоторое выражение. Проектирование возможно только, если терм $\mathcal{T}_0$ имеет вид $(\mathcal{E}_{oi})$, где $\mathcal{E}_{oi}$ – выражение, которое может быть отождествлено как $\mathcal{E}_{\ell i}$.

По теореме 1 выражение $\mathcal{E}_{\ell i}$ является L-выражением. Кроме того, оно содержит по крайней мере на одну пару скобок меньше, чем выражение $\mathcal{E}_\ell$. Следовательно, по индукционному предположению алгоритм проектирования $\mathcal{E}_{\ell i}$ на $\mathcal{E}_{oi}$ дает ответ о возможности отождествления и, в случае положительного ответа, припишет каждой переменной из $\mathcal{E}_{\ell i}$ одно вполне определенное значение. Теперь нам остается только, подобно тому, как это делалась в случаях *C2* и *C3*, заменить $\mathcal{E}_{\ell 1}$ на $\mathcal{E}'_{\ell 1}$ путем подстановки значений всех переменных,

встречавшихся в $\mathcal{E}_{\ell i}$ (если такая подстановка возможна), и приступить к проектированию $\mathcal{E}'_{\ell 1}$ на $\mathcal{E}'_{o1}$, которое также даст нам однозначный ответ, ибо $\mathcal{E}'_{\ell 1}$ тоже содержит не более n пар скобок.

Это завершает описание алгоритма проектирования и доказательство теоремы 2. Кроме того, мы доказали следующее утверждение:

Теорема 3. Алгоритм проектирования L-выражения $\mathcal{E}_\ell$ на объектное выражение $\mathcal{E}_o$ всегда дает ответ о возможности отождествления $\mathcal{E}_o$ как $\mathcal{E}_\ell$, а при положительном ответе дает также соответствующий набор значений свободных переменных.

Рассмотрим все вхождения свободной переменной символа с индексом $\mathcal{Z}$ в типовое выражение $\mathcal{E}_t$. Допустим, что некоторые из них имеют спецификаторы. Составим список, являющийся пересечением всех списков, входящих в эти спецификаторы. Пусть этот список есть $\mathcal{P}$. Заменим все вхождения переменной $s\mathcal{Z}$ в $\mathcal{E}_t$ на $s(\mathcal{P})\mathcal{Z}$. Эту процедуру назовем унификацией спецификаторов. Очевидно, что если $\mathcal{E}_t$ и $\mathcal{E}'_t$ различаются лишь таким образом, что это различие исчезает после унификации спецификаторов некоторых переменных, они эквивалентны в том смысле, что любое $\mathcal{E}_o$ отождествимо как $\mathcal{E}_t$ тогда и только тогда, когда оно отождествимо как $\mathcal{E}'_t$. Если в L-выражении $\mathcal{E}_\ell$ спецификаторы всех переменных унифицированы, то алгоритм отождествления упрощается: подстановка значения свободной переменной в оставшуюся часть $\mathcal{E}_{\ell 1}$ оказывается всегда возможной.

Рефал-машина

Рефал-машина имеет три наблюдаемые части: поле памяти, представляющее собой последовательность предложений, которая не меняется в процессе работы машины; поле зрения, являющееся в каждый момент времени некоторым рабочим выражением, преобразуемым рефал-машиной; указатель состояния, который может принимать значения: "работа", "аварийная остановка" и "нормальная остановка". Мы будем также говорить, когда это удобно, что поле памяти содержит последовательность предложений, а поле зрения содержит то или иное выражение.

Работа рефал-машины складывается из ряда однотипных действий, называемых шагами. Выполнение шага начинается с поиска в поле зрения ведущего знака k. Если в поле зрения нет знаков k, рефал-машина приходит в состояние нормальной остановки. Найдя ведущий знак k, машина сравнивает его область действия, которую мы будем называть

конкретизируемым выражением, с левыми частями предложений, составляющих поле памяти, пытаясь с помощью описанного выше алгоритма отождествить конкретизируемое выражение как левую часть одного из предложений.

Найдя в поле памяти первое такое предложение, что конкретизируемое выражение может быть отождествлено как его левая часть, рефал-машина заменяет в поле зрения конкретизируемое выражение вместе с ограничивающими его знаками k и $\perp$ на правую часть предложения, в которой все свободные переменные заменены на те значения, которые они получили в процессе отождествления. Этим выполнение шага заканчивается, и машина переходит к выполнению следующего шага.

Если рефал-машина не находит подходящего выражения, она приходит в состояние аварийной остановки.

Употребляя термин алгоритм в обычном значении, можно сказать, что последовательность предложений есть описание алгоритма преобразования выражений на языке РЕФАЛ. Поэтому мы будем также называть последовательность предложений алгоритмом. Процесс преобразования рефал-машиной поля зрения или его подвыражения будем называть конкретизацией данного выражения. Пусть задан алгоритм $\mathcal{A}$ и рабочее выражение $\mathcal{E}$. Приведем в действие рефал-машину с полем памяти $\mathcal{A}$ и полем зрения $\mathcal{E}$. Если после совершения некоторого числа шагов машина придет к состоянию нормальной остановки, то ее поле зрения будем называть результатом конкретизации $\mathcal{E}$. Если машина приходит в состояние аварийной остановки или не останавливается вообще, будем говорить, что конкретизация $\mathcal{E}$ невозможна. Иногда мы будем для краткости называть результат конкретизации $\mathcal{E}$ просто конкретизацией $\mathcal{E}$.

Пусть в алгоритме $\mathcal{A}$ есть хотя бы одно предложение с детерминативом $\mathcal{F}$. Тогда можно построить знаковую машину, называемую функцией $\mathcal{F}$. Она имеет две наблюдаемые части: поле аргумента-значения, содержащее некоторое объектное выражение, и указатель результата, который может находиться в одном из двух состояний: "нет результата" и "есть результат". Устройство функции $\mathcal{F}$ включает в себя рефал-машину, поле памяти которой содержит алгоритм $\mathcal{A}$. Перед началом работы функции $\mathcal{F}$ указатель приводится в состояние "нет результата", а в поле аргумента - значения вводится некоторое объектное выражение $\mathcal{E}$, называемое аргументом. Когда мы включаем функцию $\mathcal{F}$, ее устройство вводит в поле зрения рефал-машины выражение $k\mathcal{F}\mathcal{E}\perp$ и запускает ее. Если конкретизация этого выражения возможна, то ее результат называют значением функции $\mathcal{F}$ от аргумента $\mathcal{E}$. Он вводится в поле аргумента-значения, вместо $\mathcal{E}$, а указатель результата приходит в

состояние "есть результат". Если конкретизация невозможна, указатель остается в прежнем состоянии, и говорят, что для данного аргумента значение функции $\mathcal{F}$ не определено.

Отличие функции от алгоритма (точнее от рефал-машины, в поле зрения которой введен алгоритм) состоит, во-первых, в том, что преобразуется не рабочее, а объектное выражение, а во-вторых, в том, что наблюдаемым является лишь результат преобразования, а не весь процесс. Поэтому две функции, которые при одинаковых аргументах всегда имеют одинаковые значения, рассматриваются как тождественные, если даже они определяются различными алгоритмами.

Последовательность предложений с детерминативом $\mathcal{F}$ будем называть описанием функции $\mathcal{F}$.

Чтобы сократить и сделать более удобочитаемой запись, мы будем пользоваться следующими правилами скорописи:

1) индекс свободной переменной набирается шрифтом меньшего размера и помещается в нижнюю позицию; заглавные буквы при этом заменяются на строчные;

2) знак $\S$ при пустом комментарии может опускаться. Предложение всегда начинается в определенной позиции с новой строки;

3) в качестве детерминативов функций могут использоваться греческие буквы, которые формально рассматриваются как составные символы (например α есть сокращение для *'ALPHA'* и т.п.).

2. Классы и подклассы

Каждое типовое выражение $\mathcal{E}_t$ можно сопоставить с множеством всех тех объективных выражений, которые могут быть отождествлены как $\mathcal{E}_t$. Это множество мы назовем класс о м, изображаемым выражением $\mathcal{E}_t$ или просто класс о м $\mathcal{E}_t$. Класс, изображаемый L-выражением, назовем L-класс о м. Дополним наш метаязык знаками, выражающими отношения и операции на множестве множеств объектных выражений. (Тот факт, что эти знаки используются как элементы метаязыка, а не как объектные знаки РЕФАЛа, будет всегда ясен из контекста). С помощью знаков $\subset$ и $=$ будем записывать отношения включения и равенства, а с помощью $\cap$ и $\cup$ – операции пересечения и объединения. Знак $\varnothing$ будет изображать пустое множество. Мы не будем отличать объектное выражение от класса, изображаемого этим выражением (и состоящего, очевидно, из единственного элемента). Поэтому запись $\mathcal{E}_o \subset \mathcal{E}_t$, где $\mathcal{E}_o$ – объектное выражение, означает отождествимость $\mathcal{E}_o$ как $\mathcal{E}_t$.

При определении отождествления были введены правила В1–В3 подстановки вместо свободных переменных их значений.

Теперь мы обобщим понятие подстановки на случай, когда значениями переменных могут быть типовые выражения. Будем называть правильной подстановкой в $\mathcal{E}_t$ замену всех свободных переменных, входящих в $\mathcal{E}_t$, на некоторые выражения, называемые их значениями, при соблюдении следующих правил:

B'1. Значением свободной переменной выражения может быть любое типовое выражение.

B'2. Значением неспецифицированной переменной символа может быть любой символ или любая свободная переменная символа. Значением специфицированной переменной символа может быть символ, входящий в ее спецификатор, или специфицированная переменная символа, спецификатор которой содержит подмножество символов, входящих в спецификатор заменяемой переменной.

B'3. Значения всех вхождений одной переменной должны совпадать.

B'4. Совокупность всех значений не должна содержать двух переменных с одинаковыми индексами, но разными признаками.

Так как правила разрешают замену переменной на себя, подстановка может затрагивать лишь часть переменных (но обязательно все вхождения этих переменных). Результат применения правильной подстановки Δ к выражению $\mathcal{E}$ будем изображать в виде $\Delta // \mathcal{E}$.

Теорема 4. Пусть даны типовое выражение $\mathcal{E}_t$ и типовое выражение $\mathcal{E}'_t = \Delta // \mathcal{E}_t$, полученное из $\mathcal{E}_t$ правильной подстановкой. Тогда $\mathcal{E}'_t \subset \mathcal{E}_t$.

Доказательство. Пусть объектное выражение $\mathcal{E}_o$ отождествимо как $\mathcal{E}'_t$. Значит, существует такая подстановка Δ_0, что $\mathcal{E}_o = \Delta_0 // \mathcal{E}'_t$. Отсюда $\mathcal{E}_o = \Delta_0 // [\Delta // \mathcal{E}_t]$. Легко видеть что композиция подстановок Δ и Δ_0 является правильной подстановкой. Следовательно, $\mathcal{E}_o$ отождествимо как $\mathcal{E}_t$. Итак, $\mathcal{E}_o \subset \mathcal{E}'_t$ влечет $\mathcal{E}_o \subset \mathcal{E}_t$, то есть $\mathcal{E}'_t \subset \mathcal{E}_t$.

Класс $\mathcal{E}'_t$, образованный из $\mathcal{E}_t$ путем подстановки, будем называть подклассом класса $\mathcal{E}_t$, или его сужением. Процедуру выделения подкласса из класса $\mathcal{E}_t$ мы также будем называть сужением класса $\mathcal{E}_t$. Заметим, что подкласс L-класса не является, вообще говоря, L-классом. Так, любой класс есть подкласс L-класса e_1.

Рассмотрим несколько примеров.

Класс, состоящий из всех выражений, содержащих ровно два символа, изображается типовым выражением $s_1\ s_2$

Классу, включающему все выражения, начинающиеся и кончающиеся одной и той же цифрой, соответствует типовое выражение

$$s\,(0123456789)_1\ e_2\ s_1$$

С помощью подстановки

$$s_1 \to s(13579)_1$$

$$e_2 \to =$$

выделяем из этого класса подкласс

$$s(13579)_1 = s(13579)_1$$

Обозначим через $\mathcal{E}_1$ типовое выражение

$$s_1 e_2 (s_3 + s_1 e_4) e_5$$

Класс $\mathcal{E}_1$ включает, в частности, выражения

$$A(B + A^*) C$$

$$АГРО(НОМ)(Х + АГРОНОМ)?$$

и другие. Подстановка

$$s_1 \to s_1$$

$$e_2 \to e_2 (137\, s_1 e_x)$$

$$s_3 \to s(AB)_1,$$

$$e_4 \to$$

$$s_5 \to 6$$

переводит класс $\mathcal{E}_1$ в класс $\mathcal{E}_2$:

$$s_1 e_2 (137\, s_1 e_x)\,(s(AB)_1 + s_1)\,6$$

который является его подклассом.

Займемся объединением и пересечением классов. Объединение двух классов может быть классом, а может и не быть им. Например, объединение классов $s(12)_a$ и $s(34)_b$ есть класс $s(1234)_a$. С другой стороны, множество всех термов есть объединение двух классов – s_1 и (e_1) , и оно никак не может быть представлено в виде одного класса. Наилучшим представлением для этого множества является объединение $s_1 \cup (e_1)$ двух классов. Представление множества в виде объединения некоторого числа классов мы будем считать удовлетворительным и поставим перед собой задачу выразить в этом виде пересечение двух произвольных классов.

Мы дадим решение этой задачи в виде алгоритма, ограничившись, однако, тем случаем, когда один из рассматриваемых классов является L-классом. Для выполнения эквивалентных преобразований этого случая оказывается достаточно. Это связано с тем, что в левые части предложений могут входить только L-выражения.

Итак, пусть дано L-выражение $\mathcal{E}_\ell$ и произвольное типовое выражение $\mathcal{E}_t$. Не ограничивая общности, мы можем считать, что в обоих выражениях спецификаторы всех переменных унифицированы. Найти пересечение $\mathcal{E}_t \cap \mathcal{E}_\ell$ – значит найти множество всех таких объектных выражений $\mathcal{E}_o$, которые входят в класс $\mathcal{E}_t$ и в то же время отождествимы как $\mathcal{E}_\ell$. Таким образом, наша задача является обобщением задачи о синтанксическом отождествлении. Допустим, что $\mathcal{E}_t$ есть объектное выражение; тогда $\mathcal{E}_t \cap \mathcal{E}_\ell$ совпадает с $\mathcal{E}_t$, если отождествление возможно, и пусто, есть отождествление невозможно. Вопрос однозначно решается описанным выше алгоритмом проектирования $\mathcal{E}_\ell$ на $\mathcal{E}_t$. В общем случае мы тоже будем решать вопрос путем проектирования $\mathcal{E}_\ell$ на $\mathcal{E}_t$, однако будем помнить, что $\mathcal{E}_t$ теперь представляет не одно объектное выражение, а множество их, и в процессе проектирования мы будем сужать это множество, выделяя из него то подмножество, на которое проектирование возможно.

Обращаясь к описанному выше алгоритму проектирования, мы обобщим его, учитывая, что в $\mathcal{E}_t$ могут появиться три дополнительных класса термов: $s\mathcal{Y}, s(\mathcal{R})\mathcal{Y}$ и $e\mathcal{Y}$ (мы используем новые метасимволы, чтобы не смешивать свободные переменные в $\mathcal{E}_t$ со свободными переменными в $\mathcal{E}_\ell$). Подклассы класса $\mathcal{E}_t$ мы будем описывать указанием на те подстановки, с помощью которых они получаются. Подстановки будут изображаться в виде $s\mathcal{Y} \to \mathcal{E}$ и т.п. Тот факт, что свободная переменная из $\mathcal{E}_\ell$ принимает определенное значение, будем записывать кратко, используя стрелку, направленную в другую сторону $s\mathcal{Z} \leftarrow \mathcal{E}$, и т.п.

В самом начале применения алгоритма возникает новая возможность, которую мы опишем, введя дополнительный пункт *CO*:

CO. Если $\mathcal{E}_\ell$ пусто, то отождествление возможно, либо если $\mathcal{E}_t$ пусто, либо если $\mathcal{E}_t$ есть последовательность некоторого числа свободных переменных $e\mathcal{Y}_1 \ldots e\mathcal{Y}_n$ (среди которых могут быть и одинаковые). В последнем случае необходимо сужение $\mathcal{E}_t$ подстановкой $e\mathcal{Y}_1 \to \langle\text{пусто}\rangle, \ldots, e\mathcal{Y}_n \to \langle\text{пусто}\rangle$.

Теперь будем двигаться слева направо по выражению $\mathcal{E}_\ell$, отщепляя, как и раньше, терм $\mathcal{T}_\ell$ от $\mathcal{E}_\ell$ и терм $\mathcal{T}_t$ от $\mathcal{E}_t$. Отличие от прежнего будет иметь место, если терм $\mathcal{T}_t$ есть свободная переменная. Опишем возможные случаи, введя по три дополнительных пункта к каждому из пунктов *C1-C4*.

Напоминаем, что случай *C1* имеет место, когда $\mathcal{T}_\ell$ есть символ $\mathcal{S}$.

C1.1. $\mathcal{T}_1$ есть $s\mathcal{Y}$. Отождествление возможно только при сужении $s\mathcal{Y} \to \mathcal{S}$. Следовательно, мы выполняем эту подстановку в $\mathcal{E}_t$ и продолжаем проектирование.

C1.2. $\mathcal{T}_t$ есть $s(\mathcal{R})\mathcal{Y}$. Если $\mathcal{S}$ входит в $\mathcal{R}$, поступаем как в п. *C1.1*. В противном случае отождествление невозможно, то есть $\mathcal{E}_o \cap \mathcal{E}_t = \emptyset$.

C1.3. $\mathcal{T}_t$ есть $e\mathcal{Y}$. Разобъем класс $\mathcal{E}_t$ на два подмножества: первое образуется как сужение класса $\mathcal{E}_t$ подстановкой $e\mathcal{Y} \to$ <пусто>, во второе войдут все остальные объектные выражения. Относительно пересечения первого подмножества с классом $\mathcal{E}_\ell$ мы пока сказать ничего не можем, надо просто продолжать проектирование. Во втором подмножестве, чтобы было возможно проектирование, значение $e\mathcal{Y}$ должно начинаться с символа $\mathcal{S}$. Множество всех выражений есть подкласс, образований подстановкой $e\mathcal{Y} \to \mathcal{S}e\mathcal{Y}$. Итак, в данном случае мы выделяем из класса $\mathcal{E}_t$ два подкласса, в которых возможно отождествление, и в каждом подклассе продолжаем проектирование независимо друг от друга.

Случай *C2* имеет место, когда $\mathcal{T}_\ell$ есть $s\mathcal{Z}$.

C2.1. $\mathcal{T}_t$ есть $s\mathcal{Y}$. Какое бы значение ни приняла переменная $s\mathcal{Y}$ в объектном выражении $\mathcal{E}_o \subset \mathcal{E}_t$, переменную $s\mathcal{Z}$ можно будет спроектировать на это значение. Поэтому мы положим $s\mathcal{Z} \leftarrow s\mathcal{Y}$ ($s\mathcal{Z}$ принимает значение $s\mathcal{Y}$) и продолжим проектирование. Однако мы должны учесть, что переменная $s\mathcal{Z}$ уже приняла значение. Это значение не есть какой-либо определенный символ, который мы могли бы подставить вместо $s\mathcal{Z}$ в оставшуюся часть $\mathcal{E}_t$. О нем известно только, что оно должно совпадать со значением $s\mathcal{Y}$. Мы учтем это, заменив в оставшейся части $\mathcal{E}_\ell$ все вхождения $s\mathcal{Z}$ на $a\mathcal{Y}$ (читать: чужая переменная $\mathcal{Y}$). Проектирование такого терма мы опишем в пункте *C5*. Так как выражение $\mathcal{E}_\ell$ содержит теперь ссылки на свободные переменные из $\mathcal{E}_t$, сужение в $\mathcal{E}_t$ должно сопровождаться некоторой модификацией $\mathcal{E}_\ell$. А именно, при сужении $s\mathcal{Y} \leftarrow \mathcal{E}$ мы должны чужие переменные $\mathcal{Y}$ заменить на $\mathcal{E}$ (с заменой s на a). Это замечание надо рассматривать, в частности, как поправку к уже описанным случаям *C1.1* и *C1.2*.

C2.2. $\mathcal{T}_t$ есть $s(Q)\mathcal{Y}$. Положим $s\mathcal{Z} \leftarrow s(Q)\mathcal{Y}$ и в оставшейся части $\mathcal{E}_\ell$ заменим $s\mathcal{Z}$ на $a(Q)\mathcal{Y}$.

C2.3. $\mathcal{T}_t$ есть $e\mathcal{Y}$. Аналогично пункту *C1.3* выделяем два подкласса, в которых можно продолжать проектирование:

$$e\mathcal{Y} \to \text{<пусто>}$$
$$e\mathcal{Y} \to s\mathcal{Y}_1\, e\mathcal{Y}$$

Во второй подстановке $\mathcal{Y}_1$ – объектный знак, отличный от всех индексов переменных в $\mathcal{E}_t$ и переменных, введенных ранее в процесс сужения $\mathcal{E}_t$.

Случай *C3* имеет место, когда $\mathcal{T}_\ell$ есть $s(\mathcal{P})\mathcal{Z}$.

C3.1. $\mathcal{T}_t$ есть $s\mathcal{Y}$. Производим сужение $s\mathcal{Y} \to s(\mathcal{P})\mathcal{Y}$.

Переменной $s\mathcal{Z}$ придаем значение $s\mathcal{Y}$. Вхождения $s(\mathcal{P})\mathcal{Z}$ в $\mathcal{E}_\ell$ заменяем на $a(\mathcal{P})\mathcal{Y}$.

C3.2. $\mathcal{T}_t$ есть $s(Q)\mathcal{Y}$. Производим сужение $s(Q)\mathcal{Y} \to s(\mathcal{R})\mathcal{Y}$, где $\mathcal{R}$ – пересечение списков $\mathcal{P}$ и Q. (Если $\mathcal{R}$ пусто, отождествление невозможно). Переменной $s\mathcal{Z}$ придаем значение $s(\mathcal{R})\mathcal{Y}$. Вхождения $s(\mathcal{P})\mathcal{Z}$ в $\mathcal{E}_\ell$ заменяем на $a(\mathcal{R})\mathcal{Y}$.

C3.3. $\mathcal{T}_t$ есть $e\mathcal{Y}$. Выделяем два подкласса:

$$e\mathcal{Y} \to \langle\text{пусто}\rangle$$
$$e\mathcal{Y} \to s(\mathcal{P})\mathcal{Y}_1 e\mathcal{Y}$$

Случай *C4* имеет место, когда $\mathcal{T}_\ell$ есть $(\mathcal{E}_{\ell i})$.

C4.1-2. Если $\mathcal{T}_t$ есть символ или свободная переменная символа, отождествление невозможно.

C4.3. $\mathcal{T}_t$ есть $e\mathcal{Y}$. Отождествление возможно в двух подклассах:

$$e\mathcal{Y} \to \langle\text{пусто}\rangle$$
$$e\mathcal{Y} \to (e\mathcal{Y}_1)\, e\mathcal{Y}$$

Теперь нам надо еще расмотреть случай, которого не было при отождествлении объектных выражений, а именно, случай, когда $\mathcal{T}_\ell$ есть чужая переменная символа. Будем считать, что она имеет вид $a(\mathcal{P})\mathcal{Y}$. Случай отсутствия спецификатора включим сюда, введя воображаемый $\mathcal{P}$, содержащий все мыслимые символы, и опуская в конечных выражениях $(\mathcal{P})$, если $\mathcal{P}$ – такой список.

C5. Рассмотрим сначала случай, когда терм $\mathcal{T}_t$ не является свободной переменной. Пусть $\mathcal{T}_t$ есть некоторый символ $\mathcal{S}$. Если $\mathcal{S}$ входит в $\mathcal{P}$, то производим сужение класса $\mathcal{E}_t$ подстановкой $s\mathcal{Y} \to \mathcal{S}$ и одновременно в $\mathcal{E}_\ell$ заменяем всюду $a(\mathcal{P})\mathcal{Y}$ на $\mathcal{S}$. Если $\mathcal{S}$ не входит в $\mathcal{P}$, отождествление невозможно. Если $\mathcal{T}_t$ есть $(\mathcal{E}_{ti})$, то отождествление невозможно.

Теперь обратимся к свободным переменным.

C5.1. $\mathcal{T}_t$ есть $s\mathcal{X}$. Производим сужение $s\mathcal{X} \to s(\mathcal{P})\mathcal{Y}$ и считаем что терм $\mathcal{T}_\ell$ спроектирован на терм $\mathcal{T}_t$.

C5.2. $\mathcal{T}_t$ есть $s(Q)\mathcal{X}$. Обозначим через $\mathcal{R}$ пересечение $\mathcal{P}$ и Q. Если оно пусто, отождествление невозможно. Если оно не пусто, производим два сужения: $s(Q)\mathcal{X} \to s(\mathcal{R})\mathcal{Y}$ и $s(\mathcal{P})\mathcal{Y} \to s(\mathcal{R})\mathcal{Y}$ и одновременно всюду в $\mathcal{E}_\ell$ заменяем $a(\mathcal{P})\mathcal{Y}$ на $a(\mathcal{R})\mathcal{Y}$. Заметим, что в пп. *C5.1* и *C5.2* индекс $\mathcal{X}$ может, в частности, совпадать с индексом $\mathcal{Y}$.

C5.3. $\mathcal{T}_t$ есть $e\mathcal{X}$. В этом случае производим выделение двух подклассов:

$$e\mathcal{X} \to \langle\text{пусто}\rangle$$
$$e\mathcal{X} \to s(\mathcal{P})\mathcal{Y} e\mathcal{X}$$

Этим мы заканчиваем описание проектирования термов, отличных от $e\mathcal{Z}$. Движение справа налево происходит совершенно аналогично, только при выделении подклассов по зна-

чению переменной $e\mathcal{Y}$ терм отщепляется не слева, а справа, например в п. *C1.3.*

$$e\mathcal{Y} \rightarrow \text{<пусто>}$$
$$e\mathcal{Y} \rightarrow e\mathcal{Y}\mathcal{S}$$

в п. *C2.3*

$$e\mathcal{Y} \rightarrow \text{<пусто>}$$
$$e\mathcal{Y} \rightarrow e\mathcal{Y}\ s\mathcal{Y}_1$$

и т.д.

Когда $\mathcal{E}_\ell$ превращается в $e\mathcal{Z}$, оно проектируется на любое $\mathcal{E}_t$, принимая его в качестве значения. На это правило мы будем ссылаться как на правило *C6*.

В процессе проектирования свободные переменные из $\mathcal{E}_\ell$ принимают значения, в которые могут входить свободные переменные из $\mathcal{E}_t$, и которые, следовательно, могут меняться в результате сужений $\mathcal{E}_t$. Поэтому, чтобы получить в конце проектирования правильный список значений переменных, надо либо подновлять этот список при каждом сужении $\mathcal{E}_t$, либо провести повторное отождествление выделенного подкласса (при котором, очевидно, сужений уже не будет).

Результат рассмотрения обобщенного алгоритма проектирования можно сформулировать следующим образом.

Теорема 5. Алгоритм проектирования L-выражения $\mathcal{E}_\ell$ на типовое выражение $\mathcal{E}_t$ дает представление множества $\mathcal{E}_\ell \cap \mathcal{E}_t$ в виде $\mathcal{E}_t^1 \cup \mathcal{E}_t^2 \ldots \cup \mathcal{E}_t^n$, где $\mathcal{E}_t^i$ – подклассы класса $\mathcal{E}_t$ а n – число, которое может быть, в частности, равно нулю (тогда $\mathcal{E}_\ell \cap \mathcal{E}_t = \varnothing$).

Рассмотрим несколько примеров проектирования $\mathcal{E}_\ell$ на $\mathcal{E}_t$. Для краткой записи хода проектирования примем несколько соглашений. Правило, используемое на данном шаге алгоритма, будем указывать, помещая в начале строки буквенно-цифровое обозначение соответствующего случая (пункта), отделенное от остальной части двоеточием, например: *C2.1:*. Если движение происходит справа налево, номер пункта помечается звездочкой: *C2.1**. Знаки равенства и запятую будем использовать как разделители; смысл их ясен из контекста. Во избежание путаницы эти знаки не будут входить в рефал-объекты в качества объектных знаков. Когда используется одно из правил, выделяющих из $\mathcal{E}_t$ два подкласса, возникает разветвление. Сначала мы всегда будем исследовать ту ветвь, которая получается подстановкой $e\mathcal{Y}$ <пусто>. После завершения исследования этого подкласса и всех порожденных им подклассов (с учетом возможных разветвлений) мы вернемся к исследованию второй ветви. Для этого мы будем метить начала разветвлений, помещая непосредственно за знаком

двоеточия порядковый номер разветвления, заключенный в скобки.

1. Найдем пересечение следующих классов:

$$\mathscr{E}_\ell = A\, s_1\, (e_2)\, e_3\, s_1\, A$$

$$\mathscr{E}_t = A\, s_a\ \ (C + e_b)\, e_c$$

C1: Удаляем первый знак A b $\mathscr{E}_\ell$ и $\mathscr{E}_t$.

C2.1: $s_1 \leftarrow s_a$, $\mathscr{E}_\ell = (e_2)\, e_3\, a_a\, A$

C4: $\mathscr{E}_{\ell i} = e_2$, $\mathscr{E}_{ti} = C + e_b$

C6: $e_2 \leftarrow C + e_b$

$\mathscr{E}_\ell = e_3\, a_a\, A$ $\quad$ $\mathscr{E}_t = e_c$

C1.3: (1)* $\cdot$ $\quad e_c \rightarrow$ <пусто>

Отождествление невозможно. Возвращаемся на **(1)**.

C1.3: (1)* $e_c \rightarrow e_c\, A$

C1: $\mathscr{E}_\ell = e_3\, a_a$ $\quad$ $\mathscr{E}_t = e_c$

C5.3: (2)* $e_c \rightarrow$ <пусто>

Отождествление невозможно. Возвращаемся на **(2)**.

C5.3: (2)* $e_c \rightarrow e_c\, s_a$

$\mathscr{E}_\ell = e_3$, $\quad \mathscr{E}_t = e_c$

C6: $e_3 \leftarrow e_c$

Отождествление закончено. Собирая все подстановки, находим, что $\mathscr{E}_\ell \cap \mathscr{E}_t$ есть подкласс $\mathscr{E}_t$, получаемый подстановкой:

$$e_c \rightarrow e_c\, s_a\, A$$

Он изображается выражением

$$A\, s_a\, (C + e_b)\, e_c\, s_a\, A$$

которое отождествляется как $\mathscr{E}_\ell$ при значениях переменных

$$s_1 \leftarrow s_a,\ e_2 \leftarrow C + e_b,\ \ e_3 \leftarrow e_c$$

2. Пусть

$$\mathscr{E}_\ell = s_1 (e_2)\, e_3\, s_4\, A, \quad \mathscr{E}_t = e_1 + e_2$$

(вместо "Отождествление невозможно, Возвращаемся на *(n)* будем писать "× *(n)*".)

C2.3: (1) $e_1 \rightarrow$ <пусто>

C2: $s_1 \leftarrow +$, $\mathscr{E}_\ell = (e_2)\, e_3\, s_4\, A$, $\mathscr{E}_t = e_2$

C4.3:(2) $e_2 \rightarrow$ <пусто>, × (2)

C4.3:(2) $e_2 \rightarrow (e_3)\, e_2$

C6: $e_2 \leftarrow e_3$

$\mathscr{E}_\ell = e_3\, s_4\, A, \quad \mathscr{E}_t = e_2$

C1.3:(3)* $e_2 \to$ <пусто>, $\times$ (3)

C1.3:(3)* $e_2 \to e_2\, A$

C2.3:(4)* $e_2 \to$ <пусто>, $\times$ (4)

C2.3:(4)* $e_2 \to e_2\, s_4$

C2.1:* $s_4 \leftarrow s_4$

C6: $e_3 \leftarrow e_2$

Отождествление закончено. Получен подкласс

$$\mathscr{E}_t' = +(e_3)\; e_2\, s_4\, A$$

Возвращаемся на (1).

C2.3:(1) $e_1 \to s_3\, e_1$

C2,1: $s_1 \leftarrow s_3$

$\mathscr{E}_\ell = (e_2)\, e_3\, s_4\, A, \quad \mathscr{E}_t = e_1 + e_2$

C4.3:(5) $e_1 \to$ <пусто>, $\times$ (5)

C4.3:(5) $e_1 \to (e_4)\, e_1$

C6: $e_2 \leftarrow e_4$

C1.3:(6)* $e_2 \to$ <пусто>, $\times$ (6)

C1.3:(6)* $e_2 \to e_2\, A$

$\mathscr{E}_\ell = e_3\, s_4, \quad \mathscr{E}_t = e_1 + e_2$

C2.3:(7)* $e_2 \to$ <пусто>

C2: $s_4 \leftarrow +$

C6: $e_3 \leftarrow e_1$

Отождествление закончено. Получен подкласс

$$\mathscr{E}_t^2 = s_3(e_4)\; e_1 + A$$

Возвращаемся на (7).

C2.3:(7)* $e_2 \to e_2\, s_5$

C2.1 $s_4 \leftarrow s_5$

C6: $e_3 \leftarrow e_1 + e_2$

Отождествление закончено. Получен подкласс

$$\mathscr{E}_t^3 = s_3(e_4)\; e_1 + e_2\, s_5\, A$$

Таким образом,

$$\mathscr{E}_\ell \cap \mathscr{E}_t = \mathscr{E}_t' \cup \mathscr{E}_t^2 \cup \mathscr{E}_t^3$$

Как легко увидеть, классы, на которые мы разложили $\mathscr{E}_\ell \cap \mathscr{E}_t$, пересекаются. Например, объектное выражение $+(A) + A$

входит в первые два подкласса, а выражение $+(A)++A$ входит во все три подкласса.

3. Найдем пересечение классов:

$$\mathscr{E}_\ell = s_4(e_2\, s(ABC)_1)\; s_4\, s(ABC)_1$$

$$\mathscr{E}_t = s(AB)_1(BC\, e_x\, s(AB)_1)\; e_z\, s(BC)_2$$

C2.2: $s_4 \leftarrow s(AB)_1$

$$\mathscr{E}_\ell = (e_2\, s(ABC)_1)\; a\; (AB)_1\, s(AB)_1$$

C4: $\mathscr{E}_{\ell i} = e_2\, s(ABC)_1$, $\mathscr{E}_{ti} = BC\, e_x\, s(AB)_1$

C3.2:* $s(AB)_1 \to s(AB)_1$, $s(ABC)_1 \leftarrow s(AB)_1$

C6: $e_2 \leftarrow BC\, e_x$

Отождествление в скобках закончено. Значения переменных, которые были найдены, должны быть подставлены в оставшуюся часть $\mathscr{E}_\ell$, а подстановка переменных в $\mathscr{E}_t$ (в данном случае тривиальная) должна быть распространена на оставшуюся часть $\mathscr{E}_t$.

Получаем:

$$\mathscr{E}_\ell = a(AB)_1\, a(AB)_1, \quad \mathscr{E}_t = e_z\, s(BC)_2$$

C5.5:(1) $e_z \to$ <пусто>

C5.2: $s(BC)_2 \to s(B)_1$

$$\mathscr{E}_\ell = a(AB)_1, \mathscr{E}_t = \text{<пусто>}, \times \text{ (1)}$$

C5.3:(1) $e_z \to s(AB)_1\, e_z$

$$\mathscr{E}_\ell = a(AB)_1, \quad \mathscr{E}_t = e_z\, s(BC)_2$$

C5.3:(2) $e_z \to$ <пусто>

C5.2: $s(BC)_2 \to s(B)_1$, $s(AB)_1 \to s(B)_1$

Отождествление закончено. Получаем подкласс

$$\mathscr{E}'_t = s(B)_1(BC\, e_x\, s(B)_1)\; s(B)_1\, s(B)_1$$

Заметим, что если список символов в спецификаторе содержит один символ, переменную можно заменить на этот символ. Итак,

$$\mathscr{E}'_t = B(BC\, e_x\, B)\; BB$$

Однако, мы еще не исследовали все возможные подклассы. Возвратимся на разветвление (2)

C5.3:(2) $e_z \to s(AB)_1\, e_z$,

$$\mathscr{E}_\ell = \text{<пусто>}, \quad \mathscr{E}_t = e_z\, s(BC)_2.$$

Отождествление невозможно. Следовательно, подкласс $\mathscr{E}'_t$ исчерпывает пересечение $\mathscr{E}_\ell \cap \mathscr{E}_t$.

Если выражение $\mathscr{E}_t$, на которое происходит проектирование, является, как и проектируемое выражение $\mathscr{E}_\ell$, L-выражением,

то совокупность подклассов, получаемая в результате, обладает следующим важным свойством.

Теорема 6. Пересечение двух L-классов, построенное с помощью алгоритма проектирования, является объединением попарно непересекающихся L-классов.

Доказательство. Выделяемые подклассы образуются из L-выражения $\mathscr{E}_t$ путем подстановки на место свободных переменных некоторых типовых выражений. Возможные подстановки таковы, что свободные переменные символа никогда не порождают новых переменных вида $e\mathscr{Z}$, а переменная вида $e\mathscr{Z}$ если и порождает новую переменную вида $e\mathscr{Z}$, то только взятую в скобки (правило *C4.3*). Следовательно, L-выражение $\mathscr{E}_t$ может породить только L-выражение.

Увеличение числа подклассов происходит при применении одного из правил *C1.3*, *C2.3*, *C3.3*, *C4.3*, *C5.3*, которые мы будем называть ветвящими. Первую альтернативу в этих правилах назовем обрывающей подстановкой, вторую-удлиняющей подстановкой.

Если $\mathscr{E}_t$ не содержит переменных вида $e\mathscr{Z}$, то алгоритм проектирования может дать не более одного подкласса.

Рассмотрим разветвление, порождаемое переменной $e\mathscr{Z}$ на основном уровне скобочной структуры. Обозначим через n число 0-термов в $\mathscr{E}_t$, не считая $e\mathscr{Z}$, и заметим, что при использовании всех правил, кроме ветвящих, число термов не меняется. Если при первом разветвлении мы совершаем обрывающую подстановку, то получаем подкласс, каждый элемент которого состоит равно из n 0-термов, а совершая удлиняющую подстановку, получаем подкласс, где каждый элемент содержит не менее чем $n+1$ 0-терм. Следовательно, эти подклассы не пересекаются. Второй подкласс может снова расщепиться, благодаря применению ветвящего правила к той же переменной, но это расщепление снова дает два непересекающихся подкласса и т.д. Сколько бы раз ни происходило ветвление по переменной $e\mathscr{Z}$, мы будем получать только попарно непересекающиеся подклассы. Следовательно, два подкласса могут пересекаться только в том случае, если при всех разветвлениях по переменной $e\mathscr{Z}$, расположенной на основном уровне скобочной структуры, каждый раз выбиралась одна и та же альтернатива. Таким образом, на основном уровне скобочной структуры эти подклассы (точнее, изображающие их L-выражения) должны совпадать, и получить два различных и пересекающихся подкласса мы можем только в том случае, если одно из выражений, входящих в скобки, может породить такие подклассы. Ясно, что это возвращает нас к исходной точке, и невозможность получения таких подклассов можно доказать путем индукции по глубине скобочной структуры.

В дальнейшем нам понадобится небольшое обобщение теоремы 6. Назовем конориginальными те подстановки Δ_j свободных переменных в типовом выражении $\mathscr{E}_t$, которые получаются в результате проектирования на $\mathscr{E}_t$ некоторого L-выражения $\mathscr{E}_t$ (так что $\Delta_j // \mathscr{E}_t$ суть сужения $\mathscr{E}_t$, отождествимые как $\mathscr{E}_t$). Подстановки Δ_j могут быть применены не только к $\mathscr{E}_t$, но и к любому другому выражению, содержащему лишь те переменные, которые входят в $\mathscr{E}_t$. Если это выражение является L-выражением, то имеет место следующая теорема.

Теорема 7. Пусть Δ_j при $j=1,2,\ldots,r$ – конориginальные подстановки, а $\mathscr{E}$ – некоторое L-выражение. Тогда $\Delta_j // \mathscr{E}$ – попарно непересекающиеся L-классы.

При доказательстве теоремы 6 мы опирались исключительно на конориginальность используемых подстановок. Отсюда следует и теорема 7.

3. Эквивалентные преобразования алгоритмов

Преобразование алгоритма $\mathscr{A}$ в алгоритм $\mathscr{A}'$ мы назовем строго эквивалентным, если замена $\mathscr{A}$ на $\mathscr{A}'$ в поле памяти рефал-машины не меняет наблюдаемых действий рефал-машины при любом поле зрения. Под наблюдаемыми действиями мы понимаем изменение состояния наблюдаемых частей машины, т.е. в данном случае – содержимого поля зрения и указателя состояния.

Если $\mathscr{A}'$ строго эквивалентен $\mathscr{A}$, то области определения этих алгоритмов совпадают, т.е. $\mathscr{A}'$ приводит к нормальной остановке тогда и только тогда, когда $\mathscr{A}$ приводит к нормальной остановке. Целесообразно несколько ослабить это требование. Мы будем называть просто эквивалентным преобразованием алгоритма $\mathscr{A}$ замену его на такой алгоритм $\mathscr{A}'$, что при любом поле зрения результаты выполнения одного шага рефал-машиной, загруженной алгоритмом $\mathscr{A}$, и рефал-машиной, загруженной алгоритмом $\mathscr{A}'$, связаны следующим образом. Если $\mathscr{A}$ не приводит к аварийной остановке, то $\mathscr{A}'$ дает в точности то же новое поле зрения, что и $\mathscr{A}$. Если $\mathscr{A}$ приводит к аварийной остановке, то результат выполнения шага при алгоритме $\mathscr{A}'$ может быть любым.

Для алгоритмов, которые никогда не приводят к аварийным остановкам, понятия простой и строгой эквивалентности совпадают. На РЕФАЛе нетрудно описывать алгоритмы таким образом, чтобы аварийные остановки были заведомо невозможны. Для этого достаточно следить за тем, чтобы для всякой функции $\mathscr{F}$ объединение всех классов, представляющих аргументы левых частей описания этой функции, было бы тождественно множеству всех выражений. В частности, к описанию

функции $\mathscr{F}$ можно добавить в конце предложения с левой частью $\mathscr{F}\ e_1$ и доопределить тем самым функцию $\mathscr{F}$ так, как представляется удобным. Однако это зачастую приводит к удлинению программы без всякой необходимости. Пусть, например, нам нужна функция $\propto$, которая уничтожает первый символ выражения, и мы всегда обращаемся к этой функции таким образом, что ее аргумент действительно начинается с символа. Тогда достаточно одного предложения, чтобы описать функцию $\propto$:

$$k \propto s_1 e_2 \sim s_1$$

Алгоритм, использующий функцию $\propto$, может быть всюду определенным, хотя сама функция $\propto$ таковой не является. Как бы мы ни доопределяли функцию $\propto$, это не повлияет на алгоритм в целом. Итак, даже для строго эквивалентного преобразования алгоритмов может оказаться полезным просто эквивалентное преобразование его составных частей.

Первое правило эквивалентного преобразования алгоритмов непосредственно вытекает из его определения:

Правило *EA1*. К алгоритму $\mathscr{A}$ можно приписать (в конце) любое предложение.

Пользуясь алгоритмом проектирования, мы можем по данной паре левых частей $\mathscr{L}_1$ и $\mathscr{L}_2$ определить их пересечение, что дает возможность совершить некоторые эквивалентные преобразования над алгоритмом, содержащим предложения с этими левыми частями. Три частных случая пересечения двух L - выражений, а именно, когда пересечение пусто или совпадает с первым или вторым выражением, дают нам еще три правила эквивалентного преобразования алгоритмов.

Правило *EA2* (перестановка). Пусть два предложения с левыми частями $\mathscr{L}_1$ и $\mathscr{L}_2$ такими, что $\mathscr{L}_1 \cap \mathscr{L}_2 = \varnothing$ стоят в алгоритме рядом. Тогда их можно поменять местами.

Правило *EA3* (экранировка). Пусть предложение с левой частью $\mathscr{L}_1$ предшествует предложению с левой частью $\mathscr{L}_2$, причем $\mathscr{L}_1 \cap \mathscr{L}_2 = \mathscr{L}_2$ (т.е. $\mathscr{L}_2 \subset \mathscr{L}_1$). Тогда второе предложение можно удалить.

Правило *EA4* (поглощение). Пусть в алгоритме стоят рядом два предложения

$$k\mathscr{L}_1 \sim \mathscr{R}_1$$
$$k\mathscr{L}_2 \sim \mathscr{R}_2$$

причем $\mathscr{L}_1 \cap \mathscr{L}_2 = \mathscr{L}_1$ (т.е. $\mathscr{L}_1 \subset \mathscr{L}_2$) и если в $\mathscr{R}_2$ заменить все свободные переменные теми значениями, которые они принимают при отождествлении $\mathscr{L}_1$ как $\mathscr{L}_2$, то $\mathscr{R}_2$ совпадает с $\mathscr{R}_1$. Тогда первое предложение можно удалить.

Доказательство справедливости этих правил тривиально, поэтому мы ограничимся тем, что приведем пример, в котором используются все правила *EA1-EA4*.

Большую роль в различных приложениях эквивалентных преобразований рефал-программ играют предикаты, то есть функции, которые принимают только два значения: T и F. Предикаты как раз и дают наиболее богатый материал для применения правила *EA4*. Допустим, что предикат $\propto$ описывается следующими предложениями

$$\S.1\ k \propto s_1 \sim F$$
$$\S.2\ k \propto s_1 s(+-)_2 \sim F$$
$$\S.3\ k \propto s_1 s_2 \sim F$$
$$\S.4\ k \propto s(+-)_1 s_2 \sim T$$
$$\S.5\ k \propto s_1 s_2 s_3 \sim F$$
$$\S.6\ k \propto s_1 s_2 s_3 s_4 s_5 \sim k_\beta\ e_5 \perp .$$

Предложение §.4 не будет работать никогда, так как оно экранируется предложением §.3. Поэтому мы отбрасываем §.4 (правило *EA3*). Предложение §.2 поглощается предложением §.3, поэтому мы и его отбрасываем (правило *EA4*). Легко видеть, что все оставшиеся предложения перестановочны (правило *EA2*), поэтому мы перепишем алгоритм следующим образом:

$$\S.1\ k \propto s_1 s_2 s_3 s_4\ e_5 \sim k_\beta\ e_5 \perp .$$
$$\S.2\ k \propto s_1 \sim F$$
$$\S.3\ k \propto s_1 s_2 \sim F$$
$$\S.4\ k \propto s_1 s_2 s_3 \sim F$$

Пользуясь правилом *EA1*, допишем еще одно предложение:

$$\S.7\ k \propto e_1 \sim F$$

в результате чего функция $\propto$ будет доопределена для случая, когда ее аргумент пуст или содержит скобки. Однако не доопределение является нашей целью, а то, что теперь стало возможным сократить запись, поглотив предложением §7 три предыдущих предложения. В результате получаем алгоритм:

$$k \propto s_1 s_2 s_3 s_4\ e_5 \sim k_\beta\ e_5 \perp .$$
$$k \propto e_1 \sim F$$

Подчеркнем, что правила *EA2* и *EA4* применимы только тогда, когда предложения непосредственно следуют друг за другом. Например, в алгоритме

$$k\ \phi\ A \sim T$$
$$k\ \phi\ s_1 \sim F$$
$$k\ \phi\ s_1 e_2 \sim T$$

никак нельзя поглотить первое предложение третьим.

Сформулируем еще одно довольно очевидное правило, которое понадобится нам в дальнейшем.

Правило *EA5* (расщепление). Пусть алгоритм содержит предложение

$$\S\, k\mathcal{L} \sim \mathcal{R}$$

и пусть $\mathcal{L}_1 = \Delta // \mathcal{L}$ есть L-класс, являющийся сужением $\mathcal{L}$ путем правильной подстановки Δ. Тогда указанное предложение можно заменить на пару предложений

$$\S\ \ k\mathcal{L}_1 \sim \Delta // \mathcal{R}$$
$$\S\ \ k\mathcal{L} \sim \mathcal{R}$$

4. Эквивалентные преобразования функций

Пусть $\mathcal{A}$ – алгоритм, описывающий, в частности, функцию $\mathcal{F}$. Обозначим через $\mathcal{M}$ область определения $\mathcal{F}$, то есть множество объектных выражений $\mathcal{E}_o$, при которых выполнение конкретизации $k\mathcal{F}\mathcal{E}_o\perp$ приводит к нормальной остановке. Назовем эквивалентным преобразованием функции $\mathcal{F}$ замену алгоритма $\mathcal{A}$ на такой алгоритм $\mathcal{A}'$, что для всех объектных выражений $\mathcal{E}_o$, входящих в $\mathcal{M}$, выполнение конкретизации $k\mathcal{F}\mathcal{E}_o\perp$ при алгоритме $\mathcal{A}'$ приводит к тому же результату, что и при алгоритме $\mathcal{A}$. Если это соотношение имеет место при любых $\mathcal{E}_o$, то преобразование $\mathcal{A} \to \mathcal{A}'$ будем называть строго эквивалентным преобразованием функции $\mathcal{F}$; как и в случае алгоритмов, мы будем формулировать правила для просто эквивалентных преобразований.

При эквивалентном преобразовании функции промежуточные состояния поля зрения и число шагов, необходимое для конкретизации значения функции, меняются. Основным инструментом таких преобразований является прогонка, то есть выполнение шага рефал-машины в условиях, когда ее поле зрения предполагается загруженным не каким-то определенным рабочим выражением, а любым из некоторого класса рабочих выражений, который мы будем изображать путем использования свободных переменных. Это не значит, что мы модифицируем рефал-машину, разрешая появление в поле зрения свободных переменных. Формально, прогонка есть рассмотрение множества рефал-машин, у каждой из которых в поле зрения содержится одно из рабочих выражений, принадлежащих к данному классу.

Кроме того, мы будем, естественно, пользоваться эквивалентными преобразованиями алгоритмов, ибо они представляют частный случай эквивалентных преобразований функций.

Так как мы будем иметь дело с классами, описываемыми подвыражениями правой части, мы отныне будем всегда считать, что свободные переменные символа в правой части снабжены теми спецификаторами, которые они получили при унификации спецификаторов в левой части.

Правило *EF1* (прогонка). Пусть одно из предложений, описывающих функцию $\mathcal{F}$, имеет вид:

$$\S\ F.X\ k\mathcal{L}_f \sim\ \mathcal{C}_1\ k\mathcal{G}\mathcal{E}_t \perp \mathcal{C}_2$$

где $\mathcal{G}$ – функция с описанием

$$\begin{aligned}
&\S\ G.1\ k\mathcal{G}\mathcal{L}_g^1 \sim \mathcal{R}_g^1\\
&\S\ G.2\ k\mathcal{G}\mathcal{L}_g^2 \sim \mathcal{R}_g^2\\
&\quad\ldots\ldots\ldots\ldots\\
&\S\ G.n\ k\mathcal{G}\mathcal{L}_g^n \sim \mathcal{R}_g^n
\end{aligned}$$

$\mathcal{E}_t$ – типовое выражение,

$\mathcal{C}_1$ и $\mathcal{C}_2$ – некоторые последовательности знаков.

В результате применения предложения $\S\ F.X$ в поле зрения войдет в качестве одного из подвыражений терм

$$k\mathcal{G}\mathcal{E}_o \perp$$

где $\mathcal{E}_o$ – объектное выражение из класса $\mathcal{E}_t$. Вид этого терма не будет влиять на работу рефал-машины до тех пор, пока знак k, с которого он начинается, не станет ведущим. Когда же этот знак k станет ведущим, машина будет применять описание функции $\mathcal{G}$, отождествляя $\mathcal{E}_o$ сначала как $\mathcal{L}_g^1$, а в случае неудачи – как $\mathcal{L}_g^2$ и т.д. до $\mathcal{L}_g^n$. Нам известно, что $\mathcal{E}_o \subset \mathcal{E}_t$, и из этого можно сделать кое-какие выводы. С помощью алгоритма проектирования определим пересечение $\mathcal{E}_t \cap \mathcal{L}_g^1$. Алгоритм проектирования может привести к одному из следующих трех результатов.

EF1.1. $\mathcal{E}_t \cap \mathcal{E}_g^1 = \mathcal{E}_t$, то есть $\mathcal{E}_t \subset \mathcal{L}_g^1$, следовательно, отождествление любого $\mathcal{E}_o \subset \mathcal{E}_t$ как $\mathcal{L}_g^1$ заведомо возможно. Поэтому мы можем, предвосхищая действия рефал-машины, заменить в правой части предложения $\S\ F.X$ терм $k\mathcal{G}\mathcal{E}_\perp$ на выражение $\mathcal{R}_g^1$, полученное из $\mathcal{R}_g^1$ заменой свободных переменных на те значения (являющиеся типовыми выражениями со свободными переменными из $\mathcal{L}_f$), которые они приняли при отождествлении $\mathcal{E}_t$ как $\mathcal{L}_g^1$.

EF1.2. $\mathcal{E}_t \cap \mathcal{L}_g^1 = \varnothing$ Значит, отождествление $\mathcal{E}_o \subset \mathcal{E}_t$ как $\mathcal{L}_g^1$ заведомо невозможно, и мы можем игнорировать предложение $\S\ G.1$ и перейти к отождествлению $\mathcal{E}_t$ как $\mathcal{L}_g^2$.

EF1.3. $\mathcal{E}_t \cap \mathcal{L}_g^1$ есть объединение r подклассов $\mathcal{E}_t$,

образуемых r коноригинальными подстановками Δ_j свободных переменных, входящих в $\mathcal{E}_t$. Эти подклассы будем называть отождествляемыми. Так как все свободные переменные входящие в $\mathcal{E}_t$, встречаются в $\mathcal{L}_f$, мы можем использовать теорему 7; для этого достаточно при проектировании $\mathcal{L}_g^1$ на $\mathcal{E}_t$ вводить такие новые идентификаторы переменных, которые не входят не только в $\mathcal{E}_t$, но и в $\mathcal{L}_f$, и дополнить подстановку в $\mathcal{E}_t$ тождественной подстановкой тех переменных которые входят в $\mathcal{L}_f$, но не входят в $\mathcal{E}_t$ (если таковые существуют). По теореме 7 подстановки Δ_j порождают r сужений класса $\mathcal{L}_f$, являющихся попарно неперсекающимися L-классами. Применяя r раз правило *EA5*, заменим $F.X$ на $r+1$ предложений

$$\S\ k\mathcal{F}\Delta_1 /\!/ \mathcal{L}_f \sim \Delta_1 /\!/ \mathcal{C}\ k\mathcal{G}\mathcal{E}_t \perp \mathcal{C}_2$$
$$\cdots\cdots\cdots\cdots\cdots\cdots$$
$$\S\ k\mathcal{F}\Delta_r /\!/ \mathcal{L}_f \sim \Delta_r /\!/ k\mathcal{G}\mathcal{E}_t \perp \mathcal{C}_2$$
$$\S\ F.X\ k\mathcal{F}\mathcal{L}_f \sim \mathcal{C}_1\ k\mathcal{G}\mathcal{E}_t \perp \mathcal{C}_2$$

Теперь, поскольку $\Delta_j /\!/ \mathcal{E}_t \subset \mathcal{L}_g^1$, мы можем к каждому из первых r предложений применить правило *EF1.1*.

Так как мы выделили из $\mathcal{L}_f$ все подклассы, для которых $\mathcal{E}_o \subset \mathcal{E}_t$ может быть отождествлено как $\mathcal{L}_g^1$, предложение $\S\ F.X$ будет теперь использоваться рефал-машиной лишь для таких $\mathcal{E}_o$, когда $\S G.1$ неприменим. Поэтому мы можем продолжить преобразование предложения $\S\ F.X$, игнорируя $\S G.1$ и отождествляя $\mathcal{E}_t$ как $\mathcal{L}_g^2$. Повторяя этот процесс, мы либо придем в конце концов к случаю *EF1.1* (которым прогонка завершается), либо исчерпаем все n предложений описания функции $\mathcal{G}$, применяя правила *EF1.2*, или *EF1.3*. В последнем случае группа предложений, являющаяся трансформацией исходного предложения $\S\ F.X$, будет заканчиваться этим предложением в его неизменном виде. Тогда это предложение можно отбросить. Действительно, оно будет использовано при трансформированном алгоритме $\mathcal{A}'$ лишь в том случае, когда $\mathcal{E}_o \subset \mathcal{E}_t$ таково, что при исходном алгоритме $\mathcal{A}$ рефал-машина для конкретизации $k\mathcal{G}\mathcal{E}_o \perp$ не найдет подходящего предложения. Значит, она либо не дойдет до того момента, когда этот знак станет ведущим, то есть будет работать до бесконечности, либо испытает аварийную остановку. Согласно нашему определению эквивалентного преобразования функций, алгоритм $\mathcal{A}'$ в этом случае может давать любой результат.

Этим завершается описание прогонки (правило *EF1*).

Примеры.

1. Пусть функция α описана предложениями

$k \propto \sim k'$ печ' ошибка $\perp$

$k \propto s_1 e_2 s_3(e_a) \sim k \propto e_2(e_a) \perp$

$k \propto s_1 e_2 (s_1 e_3) \sim e_2$

$k \propto e_1 \sim e_1$

а описание функции ϕ содержит предложение

$$\S\ X\ k\ \phi\ e_1(\ ) \sim k \beta\ k \propto +(e_1)(e_1)(+-) \perp \perp$$

Здесь возможна прогонка обращения к функции $\propto$. Первые два предложения в описании $\propto$ показываются заведомо неприменимыми ($\mathscr{E}_t \cap \mathscr{L}_\alpha^i = \varnothing$), а третье – заведомо применимым ($\mathscr{E}_t \subset \mathscr{L}_\alpha^3$), причем свободные переменные принимают следующие значения:

$s_1 \leftarrow +$

$e_2 \leftarrow (e_1)(e_1)$

$e_3 \leftarrow -$

Применяя правило *EF1*, преобразуем предложение $\S X$ к виду

$$k\ \phi\ e_1(\ ) \sim k\ \beta(e_1)(e_1) \perp$$

2. Пусть функция $\propto$ определяется алгоритмом

$\S$ *A.1* $k \propto s_1 \sim s_1$

$\S$ *A.2* $k \propto e_1 \sim k \propto e_1 A \perp\ k\ B\ e_1 \perp$

$\S$ *B.1* $k \beta \sim$

$\S$ *B.2* $k \beta s_1 e_2 \sim e_2$

Сделаем прогонку обращения к β в правой части $\S$ *A.2*. Рассматривая $\S$ *B.1*, выделяем подкласс, отождествляемый с помощью подстановки $e_1 \rightarrow$ <пусто>. В результате $\S$ *A.2* расщепляется на два предложения

$\S$ *A.2.1* $k \propto \sim k \propto A \perp$

$\S$ *A.2* $k \propto e_1 \sim k \propto e_1 A \perp\ k \beta e_1 \perp$

Предложение $\S$ *A.2.1* допускает простую (без расщепления) прогонку с использованием $\S$ *A.1*:

$\S$ *A.2.1* $k \propto \sim A$

Продолжая прогонку *§ A.2* , мы теперь принимаем во внимание *§ B.2* . Отождествляемый подкласс образуется подстановкой $e_1 \to s_2 e_1$. Так как *§ B.2* – последнее предложение в описании функции β , исходное предложение *§ A.2* мы опускаем, (правило *EF1.3*). В результате получаем

$$\S\ A.2.2 \quad k \propto s_2 e_1 \sim k \propto s_2 e_1 A \perp e_1$$

Теперь описание функции $\propto$ составляют предложения *§ A.1, § A.2, § A.2.2.* Отметим, что разделение множества аргументов функции $\propto$ на два подмножества, сообразно с причиной неопределенности (аварийная остановка или отсутствие остановки), изменилось вследствие нашего преобразования. Аргумент, начинающийся со скобки, приводил к бесконечной работе при прежнем описании, но приводит к аварийной остановке при новом описании.

3. Пусть предикат ϕ описан следующими предложениями:

$$k\phi \sim T$$
$$k\,\phi\, s_1 \sim T$$
$$k\,\phi\, s_1 e_2 s_1 \sim k\,\phi\, e_2$$
$$k\,\phi\, e_1 \sim F$$

Функция ϕ принимает значение T для симметричных цепочек символов. Поставим вопрос: при каких e_1 цепочка $AB e_1 B e_1$ будет симметричной? Для этого введем функцию $\propto$, описанную одним предложением

$$\S\ A \quad k \propto e_1 \sim k\,\phi\ \mathrm{AB}\, e_1\, \mathrm{B}\, e_1$$

и будем преобразовывать ее путем прогонки.

Предложения §1 и §2 оказываются непременимыми. Применяем §3. Опишем процесс проектирования (номер используемого пункта опускаем):

$$\mathscr{E}_\ell = s_1 e_2 s_1\,, \qquad \mathscr{E}_t = AB e_1\ B e_1$$
$$s \leftarrow A$$

(1) $\quad e_1 \to$ <пусто>, $\quad \mathscr{E}_t = ABB, \quad \times$ *(1)*

(1) $\quad e_1 \to e_1 A\,, \quad e_2 \leftarrow B e_1\ AB e_1$

Отождествление закончено. Выделен один подкласс. По правилу *EF1.3* предваряем *§ A* предложением:

$$\S\ A.1 \quad k \propto e_1 A \sim k\,\phi\ AB e_1\ AB\ e_1 A \sim k\phi\ B e_1 AB\, e_1$$

(Вторая правая часть, отделенная знаком $\sim$, означает, что произведен еще один шаг прогонки, не требующий расщепления). Продолжая прогонку в предложении § A , применяем §4:

$$\S\, A \quad k \propto e_1 \sim F$$

Итак, мы установили, что e_1 должно кончаться на символ A. Теперь будем прогонять правую часть § *A.1.* . Применяем §3.

$$\mathscr{E}_\ell = s_1 e_2 s_1 , \qquad \mathscr{E}_t = B\, e_1\, AB\, e_1$$
$$s_1 \leftarrow B$$
(1) $$e_1 \to \langle \text{пусто} \rangle , \qquad \mathscr{E}_t = AB, \quad e_2 \leftarrow A$$

Отождествление возможно.

(1) $$e_1 \to e_1 B, \qquad \mathscr{E}_t = e_1 BAB\, e_1, \quad e_2 \leftarrow e_1\, BAB\, e_1$$

Снова отождествление возможно. Мы выделели, таким образом, два отождествляемых подкласса:

$$\S\; k \propto A \sim k\phi\, BAB \sim k\phi\, A \sim T$$

$$\S\; A.1.1 \quad k \propto e_1 BA \sim k\phi\, B\, e_1\, BAB\, e_1 B \sim k\phi\, e_1 BAB\, e_1 \perp$$

Продолжаем прогонку в предложении § *A.1:*

$$\S \quad A.1 \quad k \propto e_1 A \sim F$$

Теперь § *A.1* мы можем поглотить (правило *EA4*) предложением § A . Итак, описание $\propto$ имеет вид:

$$\S \quad k \propto A \sim T$$
$$\S \quad A.1.1 \quad k \propto e_1 BA \sim k\phi\, e_1 BAB\, e_1 \perp$$
$$\S \quad k \propto e_1 \sim F$$

Прогоняем обращение к ϕ в § *A.1.1.*

$$\mathscr{E}_\ell = s_1\, e_2\, s_1 , \qquad \mathscr{E}_t = e_1\, BAB\, e_1$$
(1) $$e_1 \to \langle \text{пусто} \rangle \qquad \mathscr{E}_t = BAB, \quad s_1 \leftarrow B, \quad e_2 \leftarrow A$$

Отождествление возможно

(1) $$e_1 \to s_2\, e_1 , \quad s_1 \leftarrow s_2 , \quad \mathscr{E}_\ell = e_2 a_2 , \qquad \mathscr{E}_t = e_1\, BAB\, s_2\, e_1$$

(2) $$e_1 \to \langle \text{пусто} \rangle , \quad \mathscr{E}_t = BAB\, s_2 , \quad e_2 \leftarrow BAB$$

Отождествление возможно

(2) $$e_1 \to e_1 s_2 , \quad e_2 \leftarrow e_1 s_2\; BAB\, s_2 e_1$$

Итак, мы получили три отождествляемых подкласса. После очевидных выкладок описание функции $\propto$ принимает вид:

$$k \propto A \sim T$$
$$k \propto BA \sim T$$
$$k \propto s_2 BA \sim T$$
$$k \propto s_2 e_1 s_2 BA \sim k\phi\, e_1 s_2 BA s_2 e_1 \perp$$
$$k \propto e_1 \sim F$$

Единственное предложение, содержащее в правой части знак конкретизации, можно подвергнуть дальнейшему преобразованию. На следующем шаге оно даст три предложения:

$$k \propto s_2 s_2 BA \sim T$$
$$k \propto s_2 s_3 s_2 BA \sim T$$
$$k \propto s_2 s_3 e_1 s_3 s_2 BA \sim k\phi\, e_1 s_3 s_2 BAB s_2 s_3 e_1, \perp$$

Этот процесс можно продолжать сколько угодно, и он будет давать все новые классы аргументов, порождающих симметричные цепочки. Множество всех выражений $\mathcal{E}$, для которых $k \propto \mathcal{E} \perp$ есть T, не является классом. С помощью описанного процесса мы получаем разложение этого множества на бесконечное число непересекающихся классов. Если ограничиться выражениями $\mathcal{E}$, содержащими не более чем заданное число символов, то мы получим исчерпывающий ответ на поставленный вопрос.

Правило *EF2* (приостановка конкретизации). Пусть одно из предложений, описывающих функцию $\mathcal{F}$, имеет вид:

$$\S\, F.X \quad k\mathcal{F}\mathcal{L}_1 \sim \mathcal{C}_1 k\mathcal{G}\mathcal{E}\perp \mathcal{C}_2$$

где $\mathcal{E}$ – произвольное (общее) выражение. Заменим в выражении $\mathcal{E}$ все термы, начинающиеся со знака k на различные свободные переменные выражения $e\mathcal{Z}_1$, ..., $e\mathcal{Z}_m$, которые назовем неразменными переменными. Если предложение, видоизмененное таким образом, допускает прогонку с тем ограничением, что не требуется сужений, затрагивающих неразменные переменные, то и исходное предложение $\S\, F.X$ может быть подвергнуто аналогичному преобразованию. Для этого надо преобразовать видоизмененное предложение, а затем снова заменить неразменные переменные на те термы, которые они представляют.

Пример. Определим функцию сложения натуральных чисел следующим образом:

$$k + (e_1)(0) \sim e_1$$
$$k + (e_1)(e_2 1) \sim k + (e_1)(e_2) \perp 1$$

Поставим вопрос: если первый аргумент есть 011, то каков должен быть второй аргумент, чтобы в сумме получилось 01111? Введем предикат равенства:

$$k = (0)\,(0) \sim T$$
$$k = (e_1 1)(e_2 1) \sim k = (e_1)(e_2)\perp$$
$$k = e_x \sim F$$

и предикат $\propto$, отвечающий на вопрос, обладает ли аргумент e_x нужным свойством:

$$k \propto e_x \sim k = (k + (011)(e_x)\perp)\ (0\,1111)\perp$$

Начинаем прогонку обращения к функции +. Предложение $\S A$ расщепляется на два:

$$k \propto 0 \sim k = (011)\,(01111) \sim F,$$
$$k \propto e_x 1 \sim k = (k + (011)(e_x)\perp 1)\,(01111)\perp$$

Во втором предложении мы можем приостановить конкретизацию и прогнать обращение к функции = (правило *EF2*). Получаем

$$k \propto e_x 1 \sim k = (k + (011)(e_x)\perp)(0111)\perp$$

Повторяя эту процедуру, преобразуем это предложение в группу

$$k \propto 01 \sim F$$
$$k \propto 011 \sim T$$
$$k \propto e_x 111 \sim k = (k + (011)(e_x)\perp)\,(01)\perp$$

Мы уже нашли решение. Однако продолжим прогонку последнего предложения:

$$k \propto 0111 \sim F$$
$$k \propto 01111 \sim F$$
$$k \propto 011111 \sim k = (k + (011)(e_x)\perp 1)\ (0)\perp$$

Снова применяя правило *EF2*, преобразуем последнее предложение:

$$k \propto e_x\ 11111 \sim F$$

Пользуясь тем, что все левые части не пересекаются, переставляем предложение с правой частью T на первое место, добавляем в конце

$$k \propto e_x \sim F$$

(правило *EA1*), и поглощаем этим предложением все предложения, кроме первого *(EA4)*. . Окончательно получаем:

$$k \propto 011 \sim T$$
$$k \propto e_1 \sim F$$

что дает полный формальный ответ не только о существовании и виде решения, но и его единственности.

В последнем примере мы с помощью эквивалентных преобразований алгоритмов произвели, в сущности, вычитание натуральных чисел, представленных в системе счисления с основанием единица. Эта идея более подробно рассматривается в работе [3], где показано, как, используя эквивалентные преобразования, можно построить алгоритм обращения заданной функции, то есть алгоритм, определяющий множество аргументов, при которых функция имеет заданное значение. В частности, этим способом из алгоритма сложения двоичных чисел столбиком получается алгоритм вычитания столбиком.

Литература

1. Турчин В.Ф. Программирование на языке РЕФАЛ. Ч.2. Формальное описание и принципы реализации РЕФАЛа. Препринт №43 Института прикладной математики АН СССР. М., 1971.
2. Марков А.А. Теория алгорифмов. – "Труды МИАН им.В.А.Стеклова". Т.42, М. Изд-во Акад. наук СССР, 1954.
3. Турчин В.Ф. Эквивалентные преобразования рекурсивных функций, описанных на языке РЕФАЛ. – В сб.: Теория языков и методы построения систем программирования. Труды симпозиума в Алуште. Киев – Алушта, 1972, с.31.

ВЫПУСК 6

ТРУДЫ ИНСТИТУТА

ГОССТРОЙ СССР • ЦНИПИАСС

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ СТРОИТЕЛЬСТВОМ

ГОССТРОЙ СССР
Центральный научно-исследовательский и проектно-экспериментальный институт автоматизированных систем в строительстве
ЦНИПИАСС

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ СТРОИТЕЛЬСТВОМ

ВЫПУСК 6 | ТРУДЫ ИНСТИТУТА

Под общей редакцией канд. экон. наук
О. А. ОВСЯННИКОВА

МОСКВА 1974

Труды института

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ СТРОИТЕЛЬСТВОМ

Выпуск 6

Редактор Е.Я.Назарова
Художник И.И.Шляндина
Макет-оригинал З.А.Александровой
Технический редактор Г.И.Блаженкова
Корректор А.К.Блажкова

Л-53291. Подписано к печати 31/X-74 г.
Формат 60×90/16. Объем 12 печ.л.
Тир. 1000 Зак. 446 Цена 1 р. 25 к.

ОТРД ЦНИПИАСС
117393, ГСП-1, Москва, В-393,
Новые Черемушки, квартал 28,
корпус 3