TALLINNA POLUTEHNILISE
INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО
ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 386

ТРУДЫ ЭКОНОМИЧЕСКОГО ФАКУЛЬТЕТА

XX

ТАЛЛИН 1975

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
№ 386 1975

УДК 681.3

ТРУДЫ ЭКОНОМИЧЕСКОГО ФАКУЛЬТЕТА

XX

Обработка информации

Таллин 1975

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

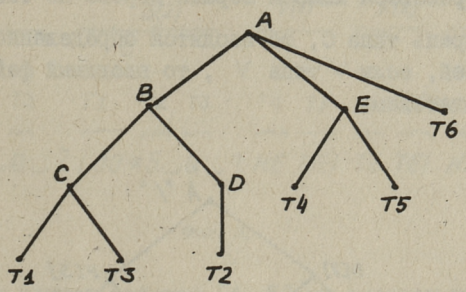
№ 386 1975

УДК 681.3.01

Т.И.Микли, Х.Ф.Симонова, Э.Т.Нунапуу

ВВОД ДАННЫХ В ИНФОРМАЦИОННЫЙ БАНК И ЯЗЫК ГЕНЕРАТОРА ВВОДА ИЕРАРХИЧЕСКИХ ДАННЫХ В ЭВМ "МИНСК-32"

В данной работе рассматривается генератор ввода данных. При этом считается, что структура данных представлена орграфом [3], т.е. конечным, ориентированным графом, вершины которого имеют метки. Не рассматривая формальное представление подобных графов, считаем, что на каком-то носителе информации структура данных представляет собой ориентированное дерево (см. фиг. 1).



Фиг. 1.

Вершины подобного дерева можно распределить на три класса:

а) корень,

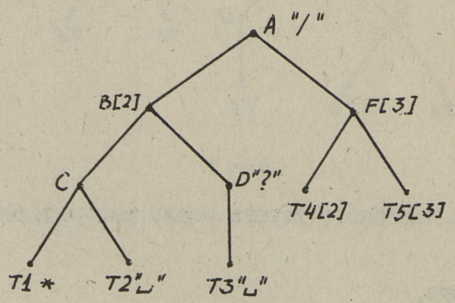
б) промежуточные вершины,

в) висячие вершины.

Используя классификацию структур данных, предложенную Кнутом [I], предлагается рассмотреть структуру данных, где сами данные расположены в атомах. Висячие вершины являются представителями атомов, а все промежуточные вершины, соответственно, представляют собой какие-то порции данных, т.е. каждая промежуточная вершина представляет собой множество всех подчиненных ей атомов. Корень дерева представляет собой запись, расположенную на каком-то носителе информации. В статье [2] ввод данных рассматривается примерно следующим образом. Структура данных на носителе информации описывается приведенным выше деревом. Каждая вершина этого дерева может быть типа C (константной) или V (переменной). Кроме того, в каждой вершине возможно повторение данных. Такую вершину мы называем рекурсивной. Если дана рекурсивная вершина типа C, то дано точно известное число повторений данных. Если вершина типа V , число повторений не известно. В этом случае, во избежание неопределенности, необходимо иметь разделитель, который указывает на конец повторений. Отметим, что разделитель можно связать с какой-либо вершиной в любом случае (число повторений $n \geq 0$).

Рассматриваются классы вершин дерева по типам.

Если корень типа C, то вводится определенное число исходных записей, если - типа V , то вводимый файл заканчивается разделителем.



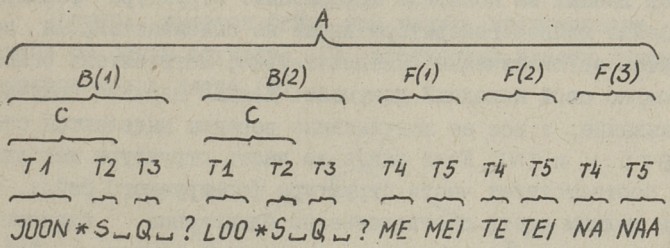
Фиг. 2.

Константа при промежуточной вершине типа C определяет массив порций, представленных ее поддеревом. При промежуточной вершине типа V разделитель заканчивает такую порцию.

Висячие вершины. В случае типа C константа является длиной поля атома, в случае типа V данные заканчиваются разделителем.

При вершинах типа C можно использовать разделители для дополнительного контроля.

На фиг. 2 изображена структура данных какого-то носителя информации с описанием вершин. Данные этого носителя оканчиваются разделителем "/". Признаки T1, T2, T3 заканчиваются соответственно разделителями "*, ␣, ␣". Вершина D отмечена разделителем "?". Это означает, что признак T3 может повторяться и при этом неопределенное число раз. Совокупность признаков T1, T2, T3 повторена дважды, и затем следуют значения признаков T4 и T5, которые имеют длину поля соответственно 2 и 3 единиц, а их совокупность повторяется 3 раза. Пример перфорации записи приведен на фиг. 3.



Фиг. 3.

В своих ранних работах [4, 5] мы следуем концепции того, что выбор носителя информации для ввода данных должен быть решен потребителем. Кроме того, данные можно вводить по частям с различных носителей информации, собирая в один файл на магнитной ленте. Это позволяет вводить данные, отперфорированные до описания их структуры.

По результатам настоящей работы составлен генератор ввода данных и язык генератора, с использованием указанных выше методов.

Структурой исходных данных для генератора ввода данных является вышеописанное дерево. Банк данных как система файлов еще окончательно не разработан. Здесь рассматривается только структура записи файла, которая представлена как дерево и хранится вместе с данными на магнитной ленте. Структура записи файла отличается от структуры вводимых данных уже тем, что в ней нет разделителей и все висячие вершины принадлежат типу С, т.е. имеют определенную длину поля.

В обязанности генератора ввода входит составление программы для ввода данных по приведенной схеме, так чтобы:

а) осуществлялся контроль соответствия структуре,

б) производилась перестановка и перекодировка данных в зависимости от требований формируемого файла,

в) осуществлялся пропуск данных, которые не относятся к файлу,

г) производилось добавление, стирание и замена данных.

Язык генератора ввода в основном описывает структуру исходных данных на носителе информации. Структура формируемого файла языком генератора ввода не описывается, а используется автоматически. Следовательно, потребитель описывает только свой исходный документ. Каждая вершина получает свое название, а все ее подчиненные вершины выделяются скобками BEGIN и END. Если какая-то часть структуры исходных данных соответствует части структуры формируемого файла, то вершины должны иметь соответственно одинаковые названия. Вершины, не имеющие соответствия в описании записи файла, должны иметь другие названия, для того, чтобы можно было производить изменения структуры.

Отметим следующие правила преобразования данных при вводе.

I. Все разделители в процессе обработки данных удаляются.

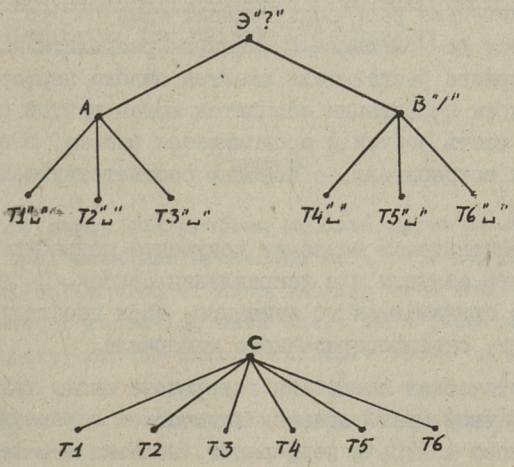
2. Если висячая вершина имеет название, которое имеется также в структуре формируемого файла, то соответствующее значение вводится.

3. Если висячая вершина не имеет соответствия в файле, то данное значение пропускается.

4. Если промежуточная вершина соответствует вершине в формируемом файле, то данные, подчиненные этой вершине, должны быть в файле без изменения структуры.

5. Если промежуточная вершина не имеет соответствия в структуре файла, то при распознавании данных она исключается. Если вершины, подлежащие исключению, не рекурсивны, то эти вершины включены в структуру для указания разделителя или просто потребитель захотел сделать структуру более наглядной. При исключении рекурсивных вершин производится операция объединения. Объединением (конкатенацией) называется операция, которая добавляет всем рекурсиям (повторениям) данной вершины все те данные, которые предшествовали данным, подчиненным этой вершине. Операция объединения может быть произведена при описании структуры исходных данных несколько раз. Она прекращается при появлении промежуточной вершины с именем, которое имеется в формируемом файле или по окончании вводимой записи. Благодаря этой операции мы получаем иерархический ввод данных, формируя из одной вводимой записи больше, чем одну выходную запись.

Пример: имеется файл, где запись представляет собой данные об одном студенте (С). Необходимо ввести данные экзаменационного листа (Э).



Фиг. 4.

Экзаменационный лист представляет собой документ вида

T1 T2 T3
T4 T5 T6
T4 T5 T6
.....
T4 T5 T6 /

Результатом операции объединения является

T1 T2 T3 T4 T5 T6
T1 T2 T3 T4 T5 T6
.....
T1 T2 T3 T4 T5 T6

Далее рассмотрим как производится контроль на правильность перфорации (контроль структуры вводимой записи). Для распознавания элементов записи используем дерево описания и алгоритм изменения порядка прохождения вершин дерева (см. [1, 3]) (образование предпорядка вершин дерева). Например, предпорядок вершин дерева, указанного на фиг. 2, следующий: T1, T2, C, T3, D, B, T4, T5, F, A.

В рекурсивных вершинах прохождение поддерева модифицируется так, чтобы обеспечить прохождение (возврат) поддерева столько раз, сколько требует рекурсия.

Если вершина типа C, то возврат производится определенное количество раз; если вершина типа V, то возврат производится до появления указанного разделителя. При появлении неверного разделителя имеется ошибка перфорации. В случае ошибки перфорации находится конец данной записи или следующая часть записи и продолжается работа, в зависимости от желания потребителя, с помощью соответствующей реакции на ошибку.

При составлении описания документа потребитель может сам выбирать символы для исправления ошибок. Не все вводимые данные принимаются во внимание, если необходимо ввести только часть отперфорированного материала.

Теоретическая концепция генератора ввода такова, что с каждой вершиной можно связать логическое выражение для содержательного контроля перфорации, но пока это не реализовано.

Литература

I. D.E. Knuth. The Art of Computer Programming. Vol. 1. Addison-Wesley, Mass. 1969.

2. A. Padgett. Tree driven data input and its validation. Computer Journal, 1973, 16, N 4.

3. А.Т. Берзтисс. Структуры данных. М., 1974.

4. Т.И. Микли, М.О. Томбак. Принципы организации больших массивов в системе "СОДИ". "Тр. Таллинского политехнического института", серия А, 1971, № 313.

5. Л.К. Выханду. Об интегрированных системах обработки дискретной информации "Тр. Таллинского политехнического института", серия А, 1971, № 313.

T. Mikli, H. Simonova, E.Винарин

Data Base Input and Language of Hierarchical Data Input Generator for "Minsk-32" Computer

Summary

In this paper the problem of data input is considered. We assume that data are presented as a tree structure, in which every collection of data belongs to a vertex.

In case of repeating data there is a possibility to use a delimiter or the number of repetitions on every level of hierarchy.

An algorithm to check the punched data and to change the structure of data is given.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 386 1975

УДК 681.3.01

Т.И. Микли, М.О. Томбак

О РАСШИРЕННОМ ГЕНЕРАТОРЕ ОТЧЕТОВ ДЛЯ ЭВМ "МИНСК-32", ПОЗВОЛЯЮЩЕМ ОПЕРИРОВАТЬ СО СТРУКТУРАМИ, ПОДОБНЫМИ СТРУКТУРАМ ЯЗЫКА ПЛ/I

Мы рассматриваем структуру данных, описание которой есть дерево с отмеченными вершинами. Каждая вершина этого дерева представляет какую-нибудь порцию данных или массив таких порций, куда входят все данные подчиненных ей вершин (см.фиг.I). Такую структуру в литературе часто называют подобной структуре языка ПЛ/I [3]. В рассматриваемой нами структуре размерность массива практически не ограничена, также рассматриваются динамические массивы, т.е. массивы, границы индексов которых есть переменные. Сами данные (атомы) этой структуры могут быть следующих типов: действительные (R), целые (I), десятичные (D), символы (C), натуральные числа (N). Каждый атом обнаруживается шаблоном, который указывает длину поля для размещения данных в памяти и выбора формата для печати.

Мы выделили тип данных "натуральные числа", так как используем плотную упаковку данных. Метод упаковки данных используется для увеличения емкости информации банка данных [4].

Вершины вышеназванного дерева распределяем в три класса:

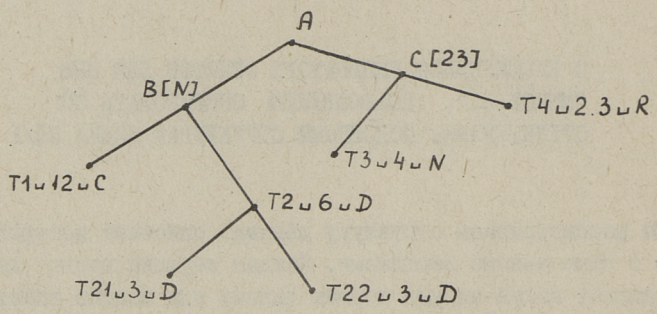
а) корень,

б) промежуточные вершины,

в) вершины, представляющие атомы.

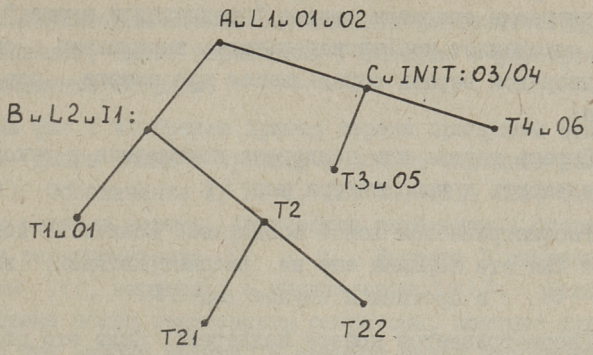
Данные типов "символы" и "десятичные числа" могут не быть представлены висячими вершинами, так как эти данные могут иметь свою структуру. Следовательно, вершины третьего

класса могут не быть висячими вершинами. Пример такой структуры приведен на фиг. I. Вершина A является корнем дерева. В вершинах B и C заданы одномерные массивы, верхние границы которых соответственно N и 23. Вершины T1, T2, T3, T4, T2I и T22 принадлежат к третьему классу. Например, T1 ␣ I2 ␣ C описывает поле из I2 символов, T4 ␣ 2.3 ␣ R описывает действительное число с плавающей запятой вида 99.999, и т.д. для вывода.



Фиг. 1.

Рассмотрим генератор данных, который позволяет преобразовать информацию банка данных в выходную информацию. Примером такого генератора является генератор данных РПГ [5]. Исходными данными могут быть один или несколько файлов, структура записи каждого из них представлена деревом. Структура выходных данных также представляет собой дерево. В программе на языке генератора не надо приводить описания исходных данных, достаточно указать названия используемых файлов. Мы считаем естественным, что потребитель системы представляет структуру своего выходного документа (отчета) вместе с правилами вычисления выходных переменных. Использование ПЛ/I подобных структур позволяет ему одновременно описать один или группу выходных документов. Для этого в языке генератора предвидена возможность в каждой вершине задать предписание вычисления значения этой переменной или в более общем варианте целые алгоритмы или их части. Следовательно, в структуре данных в каждой вершине может быть представлена какая-то последовательность операторов 0,I, 0,2, ... 0 N (см. фиг. 2).



Фиг. 2.

Рассматриваемая структура выходной информации позволяет выделить класс отчетов, который не требует сортировки данных. Предположив, что если выводимая запись (отчет)представляет собой матрицу или соединение нескольких матриц, которая помещается в оперативной памяти, то ее можно сформировать, просматривая исходные файлы только один раз. Отметим, что класс генераторов, не требующих сортировки данных, рассматривается в работе [2].

Данный генератор отчетов позволяет обрабатывать сортированные и несортированные исходные данные. В первом случае, следуя примеру генератора РПГ, для управления выводом используем индикаторы, т.е. метки – логические переменные. Индикатор можно связать с любой вершиной дерева и при значении TRUE все данные, подчиненные этой вершине выводятся. Значение TRUE можно присвоить индикатору при помощи управляющего поля или присвоением значения логического выражения. В первом случае предполагается, что сортировка исходных данных проведена в порядке задания управляющих полей. Приняты к использованию два системных индикатора INIT и FIN. Первый получает значение TRUE после каждого такта генератора, т.е. после смены исходной записи, второй получает значение TRUE только по окончании исходного файла.

Структура выходных данных – дерево с отмеченными вершинами. Это дерево является одновременно схемой программы,

т.е. определяет действительный порядок ее выполнения. Точнее, схемой программы является дерево,"предпорядок вершин" [I,3] которого определяет последовательность выполнения операторов. Предпорядок вершин определяется при помощи следующего алгоритма:

S1. Корень дерева или поддерева помещается в выходную последовательность и выполняется шаг S2.

S2. Рассматривается левое поддерево. Если это поддерево не пустое или эта вершина еще не рассматривалась, выполняется шаг S1, в противном случае шаг S3.

S3. Рассматривается правое поддерево. Если это поддерево не пустое, выполняется шаг S1, иначе рассматривается вершина, соответствующая началу дуги, ведущей в данное поддерево, и выполняется шаг S2.

Работа алгоритма заканчивается, когда пройдены все вершины дерева.

Например, предпорядок вершин, приведенных на фиг. I,следующий: A, B, TI, T2, T2I, T22, C, T3, T4.

Пусть дано дерево, с каждой вершины которого можно связать логическое выражение L. Предположим, что значение L можно вычислить. Теперь модифицируем алгоритм таким образом, что при L = FALSE поддерево данной вершины не рассматривается. Шаг S1 теперь будет следующий:

S1. Корень дерева или поддерева помещается в выводную последовательность. При L=TRUE выполняется S2, при L=FALSE выполняется S3.

Допустим, что с вершинами A и B дерева на фиг. 2 связаны соответственно логические выражения L1 и L2. Если логическое выражение L1 = FALSE, то выходной последовательности принадлежит лишь вершина A. Если же L2 = FALSE, то последовательность вершин A, B, C, T3, T4.

Язык описываемого здесь генератора отчетов позволяет связывать с вершинами дерева выходного документа логические выражения. При помощи логических выражений можно определить, какие исходные данные преобразуются в выходные в определенной части поддерева. Методика такой сортировки данных при помощи логических выражений дана в статье [2]. В ней рас-

сматриваются отчеты типа матриц, где каждой строке и каждому столбцу матрицы можно сопоставить логическое выражение, определяющее, какие переменные участвуют в формировании данного элемента матрицы.

На фиг.2 изображен пример дерева программы нашего генератора (длины полей и границы индексов массивов указаны на фиг. I). Обозначения A, B, C, TI, T2, T2I, T22, T3, T4 являются метками вершин. Логические выражения и операторы отмечены соответственно Li и Oi. Вершина B связана с индикатором I1:, вершина C с индикатором INIT:. Вместе с индикаторами можно представить операторы, которые выполняются только при выводе данных. В примере таким оператором является O3 (назовем его оператором для таблицы).

Далее представляем функциональный алгоритм работы генератора, считая, что схемой программы является вышеописанное дерево (фиг. 2) и последовательность вершин схемы образовывается описанным алгоритмом. Обозначим добавление новой вершины в предпорядок оператором NODE = f(NODE).

TI. Выполнить операторы T2 ÷ T5, пока имеются исходные записи.

T2. Ввести очередную исходную запись и, если надо, изменить значение индикатора.

T3. NODE = f(NODE). Если с вершиной связано логическое выражение, вычислить его значение. При значении TRUE выполнить шаг T4, при значении FALSE – шаг T3 снова.

T4. Если с вершиной связан индикатор, то проверяется его значение. При значении TRUE выполнить операторы для таблиц, если они имеются, и вывести данные, представленные поддеревом.

T5. Если с вершиной связаны операторы, выполнить их и, если схема не закончена, выполнить шаг T3.

Для такого генератора отсчетов составлены язык и транслятор. Аналогично генератору ввода данных вершины дерева задаются названиями. Подчиненные какой-то вершине операторы выделяются скобками BEGIN и END. Язык имеет понятия арифметического, логического выражения и оператор присваивания. Их представление аналогично АЛГОЛу.

Литература

I. D.E. Knuth. The Art of Computer Programming. Vol. 1. Addison-Wesley, Mass., 1969.

2. Т.А. Аус, М.Г. Рябовнйтра, М.О. Томбак. Генератор матричных отчетов. "Тр. ВЦ Тартуского гос. ун-та", 1974, 30, с. 23-30.

3. А.Т. Берзтиссс. Структуры данных. М., 1974.

4. Т.И. Микли, М.О. Томбак. Принципы организации больших массивов в системе "СОДИ". "Тр. Таллинск. политехн. ин-та", серия А, № 1971, № 313, с. 21-30.

5. Ф.П. Фишер, Д.Ф. Суиддл. Системы программирования. М., 1971.

T. Mikli, M. Tombak

Report Program Generator Using Data as PL/I-like Structures

Summary

A model of report program generator using data as PL/I-like structures is described. Logical expressions for data sorting, arithmetical operators and indicators for output control can be connected with all names in the structure. An algorithm to describe the functioning of such generator is given.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 386 1975

УДК 5I9.I

Г.А. Вейнер

ДВЕ ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ ОПТИМАЛЬНОГО ПРИБЛИЖЕНИЯ ГРАФА ПОЛНЫМИ ПОДГРАФАМИ (дополнение)*

В настоящей краткой статье структура графа G и остальные необходимые понятия определены так, как в статье [I]. В результате следующих далее рассуждений обобщается теорема, рассмотренная в статье [I], и уменьшается объем необходимых вычислений.

Теорема I. Если векторы функции оценки графа G направлены в G_0, то не найдется оптимального разбиения, которое имело бы классы V_i, $i \in I$, $I \subseteq \{1,2,\ldots,\kappa\}$ и $V_0 - \bigcup_{i \in I} V_i$.

Доказательство. Обозначим $\sum_{i=1}^{\kappa} m_i = M$. Вначале заметим, что для функции оценки в виде $f(G_i, G_0)$ выполняется условие

$$(v_0 - v_i) : \left[m_i + \frac{v_i}{v_0 - v_i}(M - m_i) \right] > 1, \qquad (P)$$

$i = 1,2,\ldots,\kappa$. Умножим обе части неравенства на величину v_i и перепишем неравенство следующим образом:

$$(v_0 - v_i) v_i > m_i v_i + \frac{v_i^2}{v_0 - v_i}(M - m_i). \qquad (I)$$

Второе слагаемое правой части неравенства (I) не отрицательное. Поэтому $(v_0 - v_i)v_i > m_i v_i$ и не существует оптимального разбиения точно с одним классом V_i.

---

* См. статью [I].

Покажем теперь, что не найдется и такого оптимального разбиения, которое содержит все классы V_i, $i = 1, 2, \ldots, k$. Пусть существует однако такое разбиение и пусть оно обозначено E'.

Суммируем неравенства (I), $i = 1, 2, \ldots, k$ и сложим с результатом неотрицательное число

$$\left(v_0 - \sum_{i=1}^{k} v_i\right) \sum_{i=1}^{k} v_i .$$

Получим

$$\sum_{i=1}^{k} (v_0 - v_i) v_i + \left(v_0 - \sum_{i=1}^{k} v_i\right) \sum_{i=1}^{k} v_i > \sum_{i=1}^{k} m_i v_i +$$

$$+ \sum_{i=1}^{k} \frac{v_i^2}{v_0 - v_i} (M - m_i) + \left(v_0 - \sum_{i=1}^{k} v_i\right) \sum_{i=1}^{k} v_i . \qquad (2)$$

Легко видеть, что левая часть неравенства (2) равна $2\rho(E')$. Предполагаем, что выполняется условие

$$M > \frac{m_i v_0}{v_i}, \quad i = 1, 2, \ldots, k. \qquad (3)$$

Отсюда следует, что $m_i v_i < \frac{v_i^2}{v_0 - v_i} (M - m_i)$ и поэтому

$$\sum_{i=1}^{k} m_i v_i < \sum_{i=1}^{k} \frac{v_i^2}{v_0 - v_i} (M - m_i). \qquad (4)$$

Справедливо неравенство

$$\sum_{i=1}^{k} m_i v_i + \sum_{i=1}^{k} \frac{v_i^2}{v_0 - v_i} (M - m_i) + \left(v_0 - \sum_{i=1}^{k} v_i\right) \sum_{i=1}^{k} v_i >$$

$$> \sum_{i=1}^{k} m_i v_i + \sum_{i=1}^{k} m_i v_i ,$$

откуда ввиду транзитивности отношения $>$ получим

$$\rho(E') > \sum_{i=1}^{k} m_i v_i . \qquad (5)$$

Это отношение (5) выполняется и тогда, когда значение M меньше требуемого условием (3). При уменьшении M правая часть неравенства (5) может только уменьшиться. Значит, разбиение E' не оптимальное разбиение.

Рассмотрим теперь подробнее функцию

$$\varphi(q) = \frac{\sum_{i=1}^{q}\left(v_0 - \sum_{i=1}^{i} v_i\right)v_i}{\sum_{i=1}^{q} m_i v_i}, \qquad (6)$$

числитель которой есть число удаленных ребер при образовании классов V_i, $i = 1,2,\ldots,q$ и знаменатель – число удаленных ребер при создании классов $V_i - V_0$. Допустим еще, что индексы удовлетворяют условиям

$$m_k \geq m_{k-1} \geq \ldots \geq m_1 .$$

$$\varphi(q) - \varphi(q+1) =$$

$$= \frac{v_{q+1}\left\{\left[(v_0 - v_1)v_1 + \ldots + (v_0 - v_1 - \ldots - v_q)v_q\right]m_{q+1} - (v_0 - v_1 - \ldots - v_{q+1})(m_1 v_1 + \ldots + m_q v_q)\right\}}{\sum_{i=1}^{q} m_i v_i \sum_{i=1}^{q+1} m_i v_i} \qquad (7)$$

Покажем, что числитель дроби (7) больше или равен нулю. Для этого заметим, что $v_0 - v_1 - \ldots - v_{q+1} > 0$. Разделим числитель на v_{q+1} и на $(v_0 - v_1 - \ldots - v_{q+1})\,m_{q+1} > 0$*. В уменьшаемом числителе для каждого коэффициента v_i выполняется условие

$$\frac{v_0 - v_1 - \ldots - v_i}{v_0 - v_1 - \ldots - v_q} \geq 1 .$$

Соответствующий коэффициент вычитаемого $\frac{m_i}{m_{q+1}} \leq 1$. Отсюда следует, что $\varphi(q) - \varphi(q+1) \geq 0$ и функция $\varphi(q)$ монотонная, что и доказывает теорему.

Следствие I. Теорему можем обобщать на случай, когда $f(G_i, G_0) \geq 1$, $i = 1,2,\ldots,k$.

Следствие 2. Если векторы функции оценки графа G направлены в G_0, то не найдется оптимального разбиения, которое имело бы классы $V_i' \subset V_i$ и $V_0 - \bigcup_{i\in I} V_i'$, $I \subseteq \{1,2,\ldots,k\}$ при условиях $V_i - V_i' \neq \emptyset$, $V_i' - V_0 \neq \emptyset$ и $V_i' \cap V_0 \neq \emptyset$.

Доказательство. Доказательство ведем от противного. Обозначим $v_i' = |V_i' \cap V_0|$, $m_i' = |V_i' - V_0|$.

* Если $v_0 - v_1 - \ldots - v_{q+1} = 0$, то дробь (7) положительная.

Рассмотрим такой частичный граф G' графа G, где подграфы V_i заменены подграфами V_i'. Из выражения функции оценки (I') легко видеть, что $f(V_i', G_0') > 1$.

При удалении одного класса V_i' от графа G необходимо удалить

$$|V_i - V_i'|\, m_i' + v_i'(v_0 - v_i')$$

ребер. Теорема I утверждает, что удаление класса V_i' относительно G_0 не оптимально и можно записать

$$m_i' v_i' < v_i'(v_0 - v_i').$$

Следовательно, замена класса V_i' классом $V_i' - V_0$ уменьшила бы число удаленных ребер.

Данное рассуждение мы можем обобщать на случай, если число классов V_i' больше одного. Следствие доказано.

Теорема 2. Граф G не имеет оптимального разбиения, которое имело бы классы $\bigcup_{j\in I} V_j$, $I \subseteq \{1, 2, \ldots, \kappa\}$, $|I| \geq 2$.

Доказательство. Рассмотрим подмножество вершин $V_s \cup V_t \subseteq \subseteq \bigcup_{j\in I} V_j$ графа G. Допустим, что для индексов s и t выполняется условие $v_s \leq v_t$. Если класс $\bigcup_{j\in I} V_j$ принадлежал бы оптимальному разбиению, то надо добавить, по крайней мере $m_s(m_t + v_t)$ ребер для того, чтобы соединить вершины множества $V_s - V_0$ с остальными вершинами этого класса. При разбиении множества $V_s - V_0$ в отдельный класс надо удалить $m_s v_s$ ребер. Это "улучшит" разбиение потому, что $m_s(m_t + v_t) > m_s v_s$. Получили противоречие.

Следствие 3. Последнюю теорему можно тривиально расширить на классы:

I) $\bigcup_{j\in I} V_j'$, где $I \subseteq \{1, 2, \ldots, \kappa\}$, $|I| \geq 2$ и каждый $V_j' \subseteq V_j$, $V_j' \cap V_0 \neq \emptyset$ и $V_j' - V_0 \neq \emptyset$,

2) $V_0' \cup (\bigcup_{j\in I} V_j')$, где $\bigcup_{j\in I} V_j'$ определен так, как в пункте I, но $V_0' \subseteq V_0$ и $V_0' - \bigcup_{j\in I} V_j' \neq \emptyset$.

Следствие 4. Если векторы функции оценки графа G направлены в подграф G_0, то не найдется оптимального разбиения, которое имело бы классы в виде $V_i \cup (V_j \cap V_0)$ и

$$V_0 - \bigcup_{i\in I} [V_i \cup (V_j \cap V_0)], \quad \text{где } I \subset \{1, 2, \ldots, \kappa\}, |I| \leq \frac{\kappa}{2}.$$

На основании теоремы I доказательство элементарное и поэтому оно опускается. Заметим еще, что следствие справедливо и тогда, когда рассмотренные классы имеют вид

$$V_i' \subseteq V_i \cup (V_j \cap V_0), \text{ где } V_i' - V_0 \neq \emptyset,\ V_i' \cap V_0 \neq \emptyset \text{ и } V_0 - \bigcup_{i \in I} V_i'.$$

Следствие 5. Если векторы функции оценки графа G направлены в подграф G_0 и $\frac{v_0 - v_i}{v_i} > 1$, $i = 1,2,\ldots,k$, то разбиение с классами V_0 и $V_i - V_0$, $i = 1,2,\ldots,k$, есть единственное оптимальное разбиение.

Доказательство. Теоремы и следствия, рассмотренные выше, позволяют утверждать, что не существует такого оптимального разбиения, которое образует разбиение для множества V_0. Допустим теперь, что класс $V_0 \cup V_i$ принадлежит какому-то оптимальному разбиению. Следовательно, необходимо добавить $m_i(v_0 - v_i)$ ребер. Но $v_i < v_0 - v_i$, и поэтому при отделении множества $V_i - V_0$ следует удалить $m_i v_i < m_i(v_0 - v_i)$ ребер. Ввиду этого класс $V_0 \cup V_i$ не принадлежит оптимальному разбиению, что и доказывает следствие.

Литература

I. Г.А. Вейнер. Две вспомогательные функции оптимального приближения графа полными подграфами. "Тр. Таллинск. политехн. ин-та", 1974, № 366.

G. Veiner

Two Auxiliary Functions for Optimal Approximating Graph by Complete Subgraphs (Additional Results)

Summary

This paper, as the first one [1], deals with the problem of finding an optimal partition E_0, which best approximates a given symmetric relation R in a special case. The main result of the paper [1] is generalised.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 386 1975

УДК 51:801

А.О. Вооглайд, М.О. Томбак

О ПРОБЛЕМАХ РЕДУЦИРОВАНИЯ В ГРАММАТИКАХ ПРЕДШЕСТВОВАНИЯ

I. Введение

В настоящей статье рассматриваются некоторые методы синтаксического анализа типа "снизу вверх". Уточняются некоторые возможности подхода, данного в [3], где Грэй и Гаррисон разбивают шаг анализа на две фазы – детектирование и редуцирование. Точнее – исследуются все возможности для редуцирования в грамматиках предшествования при помощи односимвольного контекста слева и справа от основы. Используя один результат из [1] показывается, что односимвольный контекст слева от основы для редуцирования позволяет анализировать все детерминированные языки.

Для полноты изложения приведем некоторые общие понятия из теории формальных грамматик.

Обозначим через A^* множество всех слов в алфавите A и $A^+ = A^* \setminus \{\Lambda\}$, где Λ – пустое слово. Если $x \in A^*$, то $|x|$ – длина слова x. Контекстно-свободная грамматика (КСГ) – это упорядоченная четверка $G = (V_N, V_T, P, S)$, где V_N – конечный алфавит нетерминальных символов, V_T – конечный алфавит терминальных символов, такой, что $V_N \cap V_T = \emptyset$, P – множество правил подстановки вида $A \to x$, где $A \in V_N$, $x \in V_T^*$ и S – начальный символ. Обозначим $V = V_N \cup V_T$, пусть $x, y \in V^*$. Слово y непосредственно выводимо из слова x (обозначим через $x \Rightarrow y$), если $x = z_1 A z_2$, $y = z_1 z z_2$ и $A \to z \in P$ ($z_1, z_2, z \in V^*$, $A \in V_N$). Если $z_2 \in V_T^*$, то слово y непосредственно кано-

нически выводимо из слова x ($x \underset{k}{\Rightarrow} y$). Слово y выводимо из слова x ($x \overset{*}{\Rightarrow} y$), если существует последовательность слов $z_0, z_1, \ldots, z_k$ ($k \geq 0$), такая, что $x = z_0$, $z_k = y$, $z_i \Rightarrow z_{i+1}$ ($0 \leq i < k$). Последовательность $z_0, z_1, \ldots, z_k$ называется выводом слова y из слова x и k – длиной вывода. Если $z_i \underset{k}{\Rightarrow} z_{i+1}$ ($0 \leq i < k$), то слово y канонически выводимо из слова x и последовательность $z_0, z_1, \ldots, z_k$ называется каноническим выводом. Пусть $G = (V_N, V_T, P, S)$ – некоторая КС-грамматика. Язык, определяемый грамматикой G – это множество слов $\mathcal{L}(G) : \mathcal{L}(G) = \{x \mid x \in V_T^*, S \overset{*}{\Rightarrow} x\}$. Любое слово $x \in \mathcal{L}(G)$ называется предложением языка $\mathcal{L}(G)$. Если $S \overset{*}{\Rightarrow} x$ ($x \in V^*$), то x называется сентенциальной формой. Если $S \overset{*}{\underset{k}{\Rightarrow}} x$ ($x \in V^*$), то x называется канонической сентенциальной формой. Если $S \overset{*}{\underset{k}{\Rightarrow}} z_1 A z_2 \underset{k}{\Rightarrow} z_1 z z_2$ ($z_1, z \in V^*$; $z_2 \in V_T^*$; $A \in V_N$), то z называется основой канонической сентенциальной формы $z_1 z z_2$.

Разбором предложения $x \in \mathcal{L}(G)$ называется последовательность слов из V^*: $z_k, \ldots, z_1, z_0$, такая, что $z_k = x$, $z_0 = S$ и последовательность $z_0, z_1, \ldots, z_k$ являются выводом предложения x. Если $z_0, z_2, \ldots, z_k$ – канонический вывод, то $z_k, \ldots, z_1, z_0$ называется каноническим разбором предложения x. Можно показать, что для каждого предложения $x \in \mathcal{L}(G)$ существует канонический разбор. Если для любого предложения $x \in \mathcal{L}(G)$ существует единственный канонический разбор, то КСГ G называется однозначной.

КСГ $G = (V_N, V_T, P, S)$ называется Λ – свободной, если все правила подстановки из P имеют вид $A \to x$, $x \in V^+$. КСГ G называется приведенной, если для каждого нетерминального символа A существуют слова $x, y \in V^*$ и $z \in V_T^*$, такие, что $S \overset{*}{\Rightarrow} xAy$ и $A \overset{*}{\Rightarrow} z$. КСГ G называется обратимой, если для любых двух правил подстановки $A \to x$ и $B \to y$ следует $x \neq y$.

В дальнейшем используем следующую систему обозначений. Символы алфавита V_N обозначим через A, B, D, символы алфавита V_T – через T, символы алфавита V – через X, Y, Z.

Прописные буквы латинского алфавита будут обозначать слова произвольного алфавита.

2. Схема редуцирования

Для описания некоторого метода синтаксического анализа (типа "снизу вверх") для класса КС-грамматик K, достаточно дать алгоритм, который по любой грамматике $G \in K$ и любой канонической сентенциальной форме x находит элемент, непосредственно следующий за x в каноническом разборе КС-формы x, т.е. алгоритм, реализующий шаг анализа. Следуя [3] разобьем шаг анализа на два действия:

I) нахождение основы $X_i \dots X_j$ $(1 \leqslant i \leqslant j \leqslant \kappa)$ сентенциальной формы $X_1 \dots X_\kappa$ (детектирование).

2) нахождение такого правила подстановки $A \to X_i \dots X_j \in P$, что $S \overset{*}{\underset{\kappa}{\Rightarrow}} X_1 \dots X_{i-1} A X_{j+1} \dots X_\kappa \underset{\kappa}{\Rightarrow} X_1 \dots X_{i-1} X_i \dots X_j X_{j+1} \dots X_\kappa$ (редуцирование).

Очевидно, что КСГ G является однозначной тогда и только тогда, когда для любой сентенциальной формы в G детектирование и редуцирование определены однозначно, т.е. если

$$S \overset{*}{\underset{\kappa}{\Rightarrow}} x_1 A_1 y_1 \underset{\kappa}{\Rightarrow} x_1 z_1 y_1$$

и

$$S \overset{*}{\underset{\kappa}{\Rightarrow}} x_2 A_2 y_2 \underset{\kappa}{\Rightarrow} x_2 z_2 y_2 = x_1 z_1 y_1,$$

то всегда

$$x_1 = x_2, \quad y_1 = y_2, \quad z_1 = z_2$$

(условие однозначности детектирования) и

$$A_1 = A_2$$

(условие однозначности редуцирования).

Определение. КС-грамматика $G = (V_N, V_T, P, S)$ называется грамматикой $C(m,\kappa)$ - ограниченным каноническим контекстом (сокращенно (m,κ) ОКК), если из условий

$$\#^m S \#^\kappa \overset{*}{\underset{\kappa}{\Rightarrow}} u_1 x_1 A_1 y_1 v_1 \underset{\kappa}{\Rightarrow} u_1 x_1 z_1 y_1 v_1$$

и

$$\#^m S \#^\kappa \overset{*}{\underset{\kappa}{\Rightarrow}} u_2 x_2 A_2 y_2 v_2 \underset{\kappa}{\Rightarrow} u_2 x_2 z_2 y_2 v_2 = u_3 x_1 z_1 y_1 v_3,$$

где

$$u_i, x_j, z_j \in V^*; \quad A_j \in V_N; \quad v_i, y_j \in V_T^*; \quad |x_j| = m; \quad |y_j| = \kappa$$
$$(i = 1, 2, 3; \ j = 1, 2),$$

всегда следует, что

$$u_2 = u_3,\ x_1 = x_2,\ z_1 = z_2,\ y_1 = y_2,\ v_2 = v_3, \qquad (I)$$

$$A_1 = A_2. \qquad (2)$$

Если соблюдаются только условия (I), то грамматика G называется (m,k) OKK детектируемой грамматикой. Если из предпосылок определения I и равенств (I) следует равенство (2), то G называется (m,k) OKK редуцируемой грамматикой.

Одна и та же КСГ G может быть (m_1,k_1) OKK редуцируемой и (m_2,k_2) OKK детектируемой грамматикой для некоторых $m_1, k_1, m_2, k_2 \geq 0$. Ясно, что при этом грамматика $G-(m,k)$OKK, где $m = \max\{m_1,m_2\}$ и $k = \max\{k_1, k_2\}$. Если нам необходимо различать фазы редуцирования и детектирования, то мы будем писать, что $G-(m_1,k_1;\ m_2,k_2)$ OKK (такое обозначение взято из [I]).

Универсальным методом детектирования для всех КС-языков является метод предшествования [6], суть которого в следующем.

Пусть $G=(V_н, V_т, P, S)$ – КС – грамматика. Определим следующие бинарные отношения на множестве $V \cup \{\#\}$ $(\# \notin V)$:

$$\lambda = \{(A,X) \mid A \to Xv \in P,\ v \in V^*\};$$

$$\rho = \{(X,A) \mid A \to uX \in P,\ u \in V^*\};$$

$$\alpha = \{(X,Y) \mid A \to uXYv \in P,\ u \in V^*;\ v \in V^*\} \cup \{(\#,S),(S,\#)\};$$

$$\lessdot = \alpha\lambda^+;$$

$$\doteq = \alpha;$$

$$\gtrdot = (\rho^+\alpha\lambda^*) \cap (V \times V_т).$$

Отношения $\lessdot$, $\doteq$, $\gtrdot$ называются отношениями предшествования.

Λ – свободная приведенная КС-грамматика G называется грамматикой предшествования, если отношения $\lessdot, \doteq, \gtrdot$ в G попарно непересекающиеся.

Имеет место следующая теорема [I].

Если G –грамматика предшествования и

$$\#S\# \overset{*}{\underset{к}{\Rightarrow}} X_1 X_2 \dots X_к A T_1 T_2 \dots T_l \underset{к}{\Rightarrow} X_1 X_2 \dots X_к Y_1 Y_2 \dots Y_m T_1 T_2 \dots T_l$$

в G, то

I) $X_i \lessdot X_{i+1} \vee X_i \doteq X_{i+1} \quad (1 \leq i < k)$;

2) $X_k \lessdot Y_1$;

3) $Y_j \doteq Y_{j+1} \quad (1 \leq j < m)$

4) $Y_m \gtrdot T_1$;

5) эти символы не находятся ни в каких других отношениях предшествования, кроме указанных в пп. I-4 отношений.

Эта теорема показывает, что любая грамматика предшествования является $(1,1)$ ОКК детектируемой.

Кроме того, имеет место и следующая теорема [2,3,5].

Для любой Λ -свободной КС-грамматики G можно построить грамматику предшествования G', такую, что $\mathcal{L}(G)=\mathcal{L}(G')$.

Алгоритм для конструирования G', приведенный в [5], вполне пригоден для практического использования.

Наложим на грамматики предшествования дополнительные ограничения, чтобы гарантировать однозначность редуцирования.

Пусть G - обратимая грамматика предшествования. Фаза редуцирования в том случае тривиальная, так как для любой основы в любой канонической сентенциальной форме существует в точности одно правило подстановки из P, при помощи которого можно осуществить редуцирование. Это - классический метод анализа грамматик предшествования [6].

Так как для редуцирования требуется информация только об основе, то можно сказать, что обратимые грамматики предшествования принадлежат к классу (0,0, I,I) ОККГ.

Определение 2. Пусть K - некоторый класс КС-грамматик. Мы будем говорить, что язык L принадлежит к классу K, если $L=\mathcal{L}(G)$ для некоторого $G \in K$.

Известно, что существуют языки типа (m,k) ОКК, для которых не существует обратимой грамматики предшествования [2]. Поэтому будем рассматривать необратимые грамматики предшествования, т.е. такие грамматики, в которых по крайней мере два правила с одинаковой правой частью.

Приведем общую схему для анализа (m,k) ОКК редуцируемых грамматик предшествования.

Определение 3. Назовем каноническим (m,k) -контекстом для нетерминала A множество:

$$C_A^{m,k}=\{(x,y)\mid \#^m S \#^k \overset{*}{\underset{k}{\Rightarrow}} uxAyv,\ |x|=m,\ |y|=k,\ yv\in V_T^*\}.$$

Из определений 1 и 3 видно, что грамматика предшествования является (m,k) OKK редуцируемой тогда и только тогда, если для любых двух нетерминалов A и B таких, что правила $A\to z$, $B\to z$ принадлежат множеству P, имеет место условие $C_A^{m,k}\cap C_B^{m,k}=\emptyset$. В этом случае можно решить вопрос редуцирования в канонической сентенциальной форме $uxzyv$ $(|x|=m,|y|=k)$ с основой z в зависимости от того, принадлежит ли (x,y) множеству $C_A^{m,k}$ или $C_B^{m,k}$. Методы для нахождения $C_A^{m,k}$ хорошо известны [4], но довольно трудоемкие и соответствующие анализаторы для $m,k>1$ весьма неэффективны. Если же $m,k\leqslant 1$, то эффективность соответствующего анализатора сравнима с эффективностью анализатора обратимого предшествования. Индукцией по длине вывода можно доказать следующую теорему.

Теорема 1. Если G -грамматика предшествования, и

$$\#S\# \overset{*}{\underset{k}{\Rightarrow}} X_1X_2\dots X_kAT_1T_2\dots T_l \underset{k}{\Rightarrow} X_1X_2\dots X_kY_1Y_2\dots Y_mT_1T_2\dots T_l$$

в G, то

1) $X_k \lessdot A \vee X_k \doteq A$;

2) $A \lessdot T_1 \vee A \doteq T_1 \vee A \gtrdot T_1$;

3) никакие другие отношения предшествования кроме указанных в пунктах 1 и 2 между этими символами места не имеют.

Из этой теоремы следует, что

$$(X,\Lambda)\in C_A^{1,0} \equiv X \lessdot A \vee X \doteq A,$$

$$(\Lambda,X)\in C_A^{0,1} \equiv [A \lessdot T \vee A \doteq T \vee A \gtrdot T] \,\&\, \in V_T \cup \{\#\}.$$

Введем следующие обозначения

$$LC(A)=\{X\mid (X,\Lambda)\in C_A^{1,0}\},$$

$$RC(A)=\{T\mid (\Lambda,T)\in C_A^{0,1}\},$$

$$C_A^{1|1}=LC(A)\times RC(A).$$

Множество $C_A^{1|1}$ назовем (1,1) независимым каноническим контекстом нетерминала A.

$C_A^{1,1} \subseteq C_A^{1|1}$, так как если $(X,T) \in C_A^{1,1}$, то $X \in LC(A)$ и $T \in RC(A)$. Поэтому, если в грамматике предшествования G для любых двух правил с одинаковой правой частью $A \to x, B \to x$ имеет место

$$C_A^{1|1} \cap C_B^{1|1} = \emptyset,$$

то

$$C_A^{1,1} \cap C_B^{1,1} = \emptyset$$

и G является (1,1)ОКК редуцируемой грамматикой. Это дает нам возможность применять вышеуказанную общую схему синтаксического анализа.

Заметим, что в некоторых случаях $C_A^{1,1} \subset C_A^{1|1}$, т.е. включение строгое. Это видно из следующего примера.

Пример I.

$G_1 = (V_N, V_T, P, S)$;

$V_N = \{S, A, B\}$; $V_T = \{a, b, c\}$;

$P = \{S \to aAa,\ S \to bAb,\ S \to aBb,\ S \to bBa,$

$A \to c,\ B \to c\}$.

$C^{1,0} = \{a, b\}$; $C_B^{1,0} = \{a, b\}$;

$C_A^{0,1} = \{a, b\}$; $C_B^{0,1} = \{a, b\}$;

$C_A^{1|1} = C_B^{1|1} = \{(a,a), (a,b), (b,a), (b,b)\}$

и G_1 не является (1|1)ОКК редуцируемой грамматикой. Выпишем все канонические выводы

$$\#S\# \underset{к}{\Rightarrow} \#aAa\# \underset{к}{\Rightarrow} \#aca\#;$$

$$\#S\# \underset{к}{\Rightarrow} \#bAb\# \underset{к}{\Rightarrow} \#bcb\#;$$

$$\#S\# \underset{к}{\Rightarrow} \#aBb\# \underset{к}{\Rightarrow} \#acb\#;$$

$$\#S\# \underset{к}{\Rightarrow} \#bBa\# \underset{к}{\Rightarrow} \#bca\#.$$

$C_A^{1,1} = \{(a,a), (b,b)\}$

и

$C_B^{1,1} = \{(a,b), (b,a)\}$;

$C_A^{1,1} \cap C_B^{1,1} = \emptyset$

и G_1 является (1,1)ОКК редуцируемой грамматикой. Пусть $x \in V^*$. Обозначим $H'(x) = \{T \mid x \underset{к}{\overset{*}{\Rightarrow}} Tz,\ T \in V_T,\ z \in V_T^*\}$.

Рассмотрим нахождение (I,I) ограниченного канонического

контекста. Алгоритм для конструирования множества $C_A^{1,1}$ дает следующая теорема.

Теорема 2. Пусть $G=(V_N,V_T,P,S)$ грамматика предшествования и $A\in V_N$. Тогда

$$C_A^{1,1}=C_1(A)\cup C_2(A)\cup C_3(A)\cup(C_4)(A),$$

где

$$C_1(A)=\{(X,T)\mid\exists\varphi\in P:\ B\to uXAv,\ T\in H^1(v)\},$$

$$C_2(A)=\{(X,T)\mid\exists\varphi\in P:\ B\to uXA,\ T\in RC(B)\},$$

$$C_3(A)=\{(X,T)\mid\exists\varphi\in P:\ B\to Av,\ X\in LC(B),T\in H^1(v)\},$$

$$C_4(A)=\{(X,T)\mid\exists\varphi\in P:\ B\to A,(X,T)\in C_B^{1,1}\}.$$

Доказательство.

I. Докажем, что $C_1(A)\cup C_2(A)\cup C_3(A)\cup C_4(A)\subseteq C_A^{1,1}$.

Возьмем произвольную (X,T) из множества

$$C_1(A)\cup C_2(A)\cup C_3(A)\cup C_4(A).$$

Имеется четыре возможности:

а) $(X,T)\in C_1(A)$.

Так как по определению множества $C_1(A)$ в P есть правило $B\to uXAv$, такое, что $v\overset{*}{\underset{K}{\Rightarrow}}Tz$ для некоторого $z\in V_T^*$ и так как G – приведенная грамматика, то существует вывод

$$\#S\#\overset{*}{\underset{K}{\Rightarrow}}xBy\underset{K}{\Rightarrow}xuXAvy\overset{*}{\underset{K}{\Rightarrow}}xuXATzy,\quad Tzy\in V_T^*\{\#\}$$

и $(X,T)\in C_A^{1,1}$.

б) $(X,T)\in C_2(A)$. Так как по определению $C_2(A)$ в P имеется правило $B\to uXA$, такое, что $T\in RC(B)$, то существует вывод

$$\#S\#\overset{*}{\underset{K}{\Rightarrow}}xBTy\underset{K}{\Rightarrow}xuXATy,\ Ty\in V_T^*\{\#\}$$

и $(X,T)\in C_A^{1,1}$.

в) $(X,T)\in C_3(A)$. Так как по определению $C_3(A)$ в P имеется правило $B\to Av$, такое, что $X\in LC(B)$ и $v\overset{*}{\underset{K}{\Rightarrow}}Tz$ для некоторого $z\in V_T^*$, то существует вывод

$$\#S\#\overset{*}{\underset{K}{\Rightarrow}}xXBy\underset{K}{\Rightarrow}xXAvy\overset{*}{\underset{K}{\Rightarrow}}xXATzy,\quad Tzy\in V_T^*\{\#\}$$

и $(X,T)\in C_A^{1,1}$.

г) $(X,T) \in C_4(A)$. Так как по определению $C_4(A)$ в P имеется правило $B \to A$, такое, что $(X,T) \in C_B^{1,1}$, то существует вывод

$$\#S\# \overset{*}{\underset{K}{\Rightarrow}} xXBTy \underset{K}{\Rightarrow} xXATy, \quad Ty \in V_T^* \{\#\},$$

и $(X,T) \in C_A^{1,1}$.

2. Докажем, что

$$C_A^{1,1} \subseteq C_1(A) \cup C_2(A) \cup C_3(A) \cup C_4(A).$$

Возьмем произвольную $(X,T) \in C_A^{1,1}$.

По определению $C_A^{1,1}$ существует канонический вывод

$$\#S\# \overset{*}{\Rightarrow} uXATv, \quad Tv \in V_T^* \{\#\}. \qquad (3)$$

Из этого следует, что на некотором шаге вывода (3) использовано правило $B \to xAy$, где $x, y \in V^*$. Имеются следующие возможности:

а) $x \neq \Lambda$, $y \neq \Lambda$,

б) $x \neq \Lambda$, $y = \Lambda$,

в) $x = \Lambda$, $y \neq \Lambda$,

г) $x = \Lambda$, $y = \Lambda$.

Эти возможности соответствуют определениям множеств $C_1(A)$, $C_2(A)$, $C_3(A)$ и $C_4(A)$ и, следовательно,

$$(X,T) \in C_1(A) \cup C_2(A) \cup C_3(A) \cup C_4(A).$$

Теорема доказана.

3. Сравнение классов грамматик и языков

В пункте 2 настоящей статьи определены следующие классы грамматик предшествования: (I,0) – ограниченным каноническим контекстом редуцируемые грамматики (сокращенно (1,0) ОКР; (0,1) ОКР; (1|1) ОКР и (1,1) ОКР грамматики). Контекст для редуцирования для классов (1,0) ОКР, (0,1) ОКР и (1|1) ОКР можно найти непосредственно из матрицы предшествования. Для построения анализатора для грамматик этих классов необходимо только незначительно модифицировать анализирующий алгоритм обратимого предшествования. Класс (1,1) ОКР грамматик требует уже специальных таблиц для контекста нетерминалов, и поэтому реализовать соответствующий анализатор несколько сложнее.

Рассмотрим соотношения данных классов грамматик и соответствующих им классов языков. Обозначим класс обратимых грамматик предшествования через $(0,0)\,ОКР$. Из соответствующих определений вытекают следующие отношения между классами грамматик:

$$(0,0)\,ОКР \subseteq (1,0)\,ОКР, \tag{4}$$

$$(0,0)\,ОКР \subseteq (0,1)\,ОКР, \tag{5}$$

$$(1,0)\,ОКР \subseteq (1|1)\,ОКР, \tag{6}$$

$$(0,1)\,ОКР \subseteq (1|1)\,ОКР, \tag{7}$$

$$(1|1)\,ОКР \subseteq (1,1)\,ОКР, \tag{8}$$

$$(0,1)\,ОКР \cap (1,0)\,ОКР = (0,0)\,ОКР. \tag{9}$$

В самом деле все включения в выражениях (4)–(8) строгие, что видно из следующих примеров.

Пример 2. $G_2 = (V_N, V_T, P, S)$,

$V_N = \{S, A, B\}$; $V_T = \{a, b, c\}$,

$P = \{S \to aA,\ S \to bB,\ A \to c,\ B \to c\}$

$G_2 \in (1,0)\,ОКР$, $G_2 \notin (0,0)\,ОКР$.

Пример 3. $G_3 = (V_N, V_T, P, S)$,

$V_N = \{S, A, B\}$, $V_T = \{a, b, c\}$,

$P = \{S \to Aa,\ S \to Bb,\ A \to c,\ B \to c\}$,

$G_3 \in (0,1)\,ОКР$, $G_3 \notin (0,0)\,ОКР$.

Пример 4. $G_4 = (V_N, V_T, P, S)$,

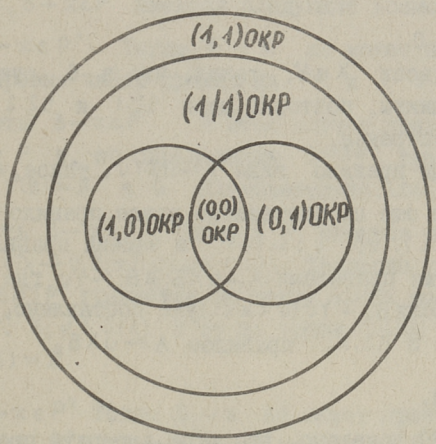
$V_N = \{S, A, B, D\}$, $V_T = \{a, b, c\}$,

$P = \{S \to aAb,\ S \to aBa,\ S \to bBa,\ S \to bDb,$

$A \to c,\ B \to c,\ D \to c\}$.

$G_4 \in (1|1)\,ОКР$, $G_4 \notin (1,0)\,ОКР$, $G_4 \notin (0,1)\,ОКР$.

Грамматика G_1 из примера I указывает на строгость включения (8). Результаты установленных соотношений изображены на фиг. I. Для изучения соответствующих классов языков введем понятие грамматик простого предшествования со смешанной стратегией [I].



Фиг.1.

Определение 4. Λ - свободная приведенная КС-грамматика называется грамматикой слабого предшествования, если:

I. Отношения $\lessdot \cup \doteq$ и $\gtrdot$ попарно не пересекаются в G.

2. Если $A \to uXYv$ и $B \to Yv \in P$, то не имеют места $X \lessdot B$ или $X \doteq B$.

Определение 5. Грамматика слабого предшествования G называется грамматикой простого предшествования со смешанной стратегией (SMSP), если из того, что $A \to x$ и $B \to x$ принадлежат P, следует, что $L_C(A) \cap L_C(B) = \emptyset$.

В статье [I] доказывается, что для любого детерминированного языка L существует SMSP грамматика G, такая, что $L = \mathcal{L}(G)$.

Теорема 3. Пусть G - SMSP грамматика. Тогда существует (1,0)ОКР грамматика G', такая, что $\alpha(G) = \alpha(G')$.

Доказательство. Для преобразования G в G' используем вторую часть алгоритма Макафе и Прессера [5]. Пусть

γ – некоторое бинарное отношение. Обозначим $X\gamma = \{Y \mid (X,Y) \in \gamma\}$, $\gamma X = \{Y \mid (Y,X) \in \gamma\}$.

Будем говорить, что в грамматике G существуют конфликты типа P_2 для $X \in V$ ($P_2(X)$ – конфликты), если $(X \doteq) \cap (X \lessdot) \neq \emptyset$. Приведем алгоритм для устранения конфликтов типа P_2.

S1. $i := 0$.

S2. Для всех $X \in V$, таких, что в G существуют $P_2(X)$ – конфликты, повторяй шаги S2.1 и S2.2, пока конфликт не будет устранен.

S2.1. Найди правило вида $A \to uXYv$, такое, что $X \lessdot Y$.

S2.2. Если еще не составлено новое правило $B_K \to Yv\ (0 < K \leqslant i)$, то $i := i+1$ и замени правило, найденное в S2.1, двумя правилами $A \to uXB_i$ и $B_i \to Yv$, где B_i – новый символ в V_N. Если $B_K \to Yv\ (0 < K \leqslant i)$ уже составлено, замени правило, найденное в S2.1, правилом $A \to uXB_K$.

S3. Стоп.

В статье [5] доказано, что этот алгоритм преобразует любую грамматику, в которой имеются конфликты только типа P_2, в грамматику предшествования.

Пусть $G' = (V'_N, V_T, P', S)$ – грамматика, полученная из SMSP грамматики $G = (V_N, V_T, P, S)$ при помощи вышеуказанного алгоритма устранения P_2-конфликтов. Докажем, что G' – $(1,0)$ OKK – грамматика. По определению SMSP грамматик в G имеются конфликты только типа P_2. Следовательно, G' – грамматика предшествования.

Покажем, что для всех $A \in V_N$ множества $LC(A)$ в грамматиках G и G' совпадают. В самом деле $LC(A) = (\lessdot A) \cup (\doteq A)$. Множества $\lessdot A, \doteq A$ могут изменяться лишь в том случае, если для ликвидации некоторого P_2 – конфликта между символами X и A сделана замена правила $D \to uXAv$ на $C \to uXB$, $B \to Av$. Но такая замена не изменяет множества $\lessdot A$, $a \doteq_G A = \doteq_{G'} A \cup \{X\}$.

Отметим, что, если в правых частях правил из P' встречается некоторый новый нетерминал из $V'_N \setminus V_N$, то этот нетерминал может быть только крайним правым символом в этой части правила. Поэтому мы можем разбить P' на следующие взаимно непересекающиеся классы:

$$P^1 = \{A \to x \mid A \to x \in P',\ A \in V_N,\ x \in V^*\},$$

$$P^2 = \{A \to x \mid A \to x \in P',\ A \in V_N,\ x \in V^*(V'_N \setminus V_N)\},$$

$$P^3 = \{A \to x \mid A \to x \in P',\ A \in V'_N \setminus V_N,\ x \in V^*\},$$

$$P^4 = \{A \to x \mid A \to x \in P',\ A \in V'_N \setminus V_N,\ x \in V^*(V'_N \setminus V_N)\}.$$

Пусть $A \to x, B \to x \in P'$. Имеются следующие возможности.

I) $A \to x \in P^1$. Тогда $B \to x$ не может принадлежать P_2 или P_4 (правые части правил из P^1 и $P^2 \cup P^4$ различны).

а) Пусть $B \to x \in P^1$. В этом случае правила $A \to x$ и $B \to x$ принадлежат P и, по определению SMSP грамматик, $LC(A) \cap LC(B) = \emptyset$ в G. Следовательно, $LC(A) \cap LC(B) = \emptyset$ в G'.

б) Пусть $B \to x \in P^3$. Тогда в P имеется некоторое множество правил $\{D_i \to u_i X_i x \mid 1 \leqslant i \leqslant K, K > 0\}$. Согласно условию 2) в определении SMSP грамматик не могут иметь места $X_i \lessdot A$ или $X_i \doteq A$, т.е. $X_i \notin LC(A)$. В то же время $LC(B) = \{X_i \mid 1 \leqslant i \leqslant K\}$ и $LC(A) \cap LC(B) = \emptyset$.

2) $A \to x \in P^2$. Тогда $B \to x$ не может принадлежать множествам P^1 или P^3 (правые части правил из P^2 и $P^1 \cup P^3$ различны).

а) Пусть $B \to x \in P^2$. Тогда в P имеются два правила с одинаковой правой частью $A \to z$ и $B \to z$ и, аналогично Iа) мы можем заключить, что $LC(A) \cap LC(B) = \emptyset$ в G'.

б) Пусть $B \to x \in P^4$. Тогда в P имеются правило $A \to z$ для некоторого z и множество правил $\{D_i \to u_i X_i z \mid 1 \leqslant i \leqslant K, K > 0\}$.

Аналогично Iб), мы можем заключить, что $LC(A) \cap LC(B) = \emptyset$.

3) $A \to x \in P^3$. Тогда $B \to x$ не может принадлежать множествам P^3 или P^4, ввиду того, что $P^3 \cup P^4$ не содержит правил с одинаковой правой частью (см. алгоритм для устранения P_2 —конфликтов). $B \to x$ не может принадлежать множеству P^2 (правые части правил из P^3 и P^2 различны). Последний вариант $B \to x \in P^1$ симметричен варианту Iб).

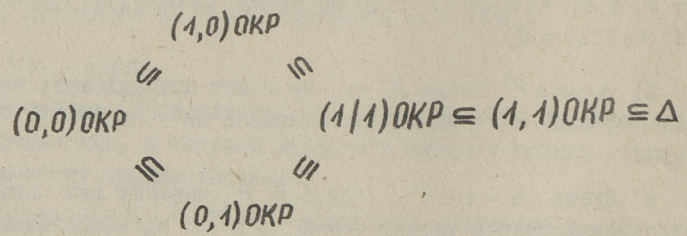
4) $A \to x \in P^4$. Тогда $B \to x$ не может принадлежать множествам P^3 или P^4 (аналогично 3), $B \to x$ не может принадлежать множеству P^1 (правые части правил из P^1 и P^4 различны. Вариант $B \to x \in P^2$ симметричен варианту 2б).

Итак, мы показали, что если $A \to x \in P'$, $B \to x \in P'$, то $LC(A) \cap LC(B) = \emptyset$. Теорема доказана.

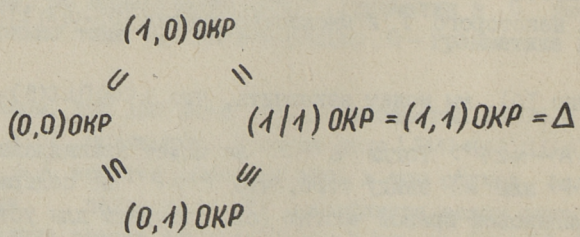
Следствие I. Класс (I,0)ОКР языков совпадает с классом детерминированных языков.

Доказательство. Теорема 3 в [I] утверждает, что для любого детерминированного языка существует SMSP грамматика. Этот результат и теорема 3 в настоящей статье показывают, что для любого детерминированного языка существует (I,0)ОКР грамматика. С другой стороны, известно, что любая LR(1) грамматика генерирует детерминированный язык ([4]), а все грамматики, определенные в настоящей статье, являются LR(1) грамматиками. Из этого следует, что любой (I,0)ОКР язык является детерминированным языком.

Учитывая схему включения классов грамматик (фиг. I), мы получаем следующую схему соотношений классов языков (фиг.2)



Фиг. 2.



Фиг. 3.

(через Δ обозначим класс детерминированных языков). Используя следствие I и результат М. Фишера [2] о существовании детерминированного языка, для которого не существует

обратимой грамматики предшествования, получаем более точную схему соотношений классов языков (фиг. 3)

Точные соотношения класса (0,I)ОКР языков с другими классами остаются пока нерешенными.

Литература

1. A.V. Aho, P.I. Denning, J.D. Ullman. Weak and mixed strategy precedence parsing. J. ACM. 1972. 19 N 2, 225-243.

2. M.J. Fischer. Some properties of precedence languages. Proc. ACM Symp. on Theory of Computing. May 1969, pp. 181-190.

3. J.N. Gray, M.A. Harrison. On the covering and reduction problems for context-free grammars. J. ACM. 1972. 19 N 4, pp. 675-698.

4. D.E. Knuth. On the translation of language from left to right. Inform. Contr. 1965. 8, pp. 607-639.

5. J. McAfee, L. Presser. An algorithm for the design of simple precedence grammars. J. ACM. 1972, 19 N 3, pp. 385-395.

6. N. Wirth, H. Weber, EULER - a generalization of ALGOL, and its formal definition. Pt. I. Comm. ACM. 1966, 9 N 1, pp. 13-25.

A. Vooglaid, M. Tombak

On Reducing Problems in Precedence Grammars

Summary

Reducing problems in precedence grammars some "top-down" syntax analysis methods are considered in this paper. Refinements are given to the approach from paper [3] where Gray and Harrison split the analysis into 2 phases: detection and reduction. More exactly, we analyse all possibilities to reduce precedence grammars using one character neighbourhood of the handle.

Following a result from [1] we prove that one character left context of the handle enables us to analyse all deterministic languages.

Содержание

Таллинский политехнический институт. Труды ТПИ № 386.
ТРУДЫ ЭКОНОМИЧЕСКОГО ФАКУЛЬТЕТА XX. Редактор Ю. Л а м п .
Техн. редактор В. Р а н н и к. Сборник утвержден коллегией Трудов ТПИ 8 мая 1975 г. Подписано к печати 20 ноября 1975 г. Бумага 60x90/16. Печ. л. 2,5 + 0,25 приложение. Уч.-изд. л. 1,71. Тираж 350. МВ-07865. Ротапринт ТПИ, Таллин, ул. Коскла, 2/9. Зак. №813.

Ц е н а 17 к о п.

Цена 17 коп.