УВАЖАЕМЫЕ СТУДЕНТЫ!

ВАМ НЕОБХОДИМО ВЫПОЛНИТЬ СЛЕДУЮЩЕЕ:

1. Ознакомиться с лекцией и законспектируйте

2. Предоставит отчет конспекта лекции прислать в виде скриншото в течении трех дней.

3. Отправить преподавателю на почту v.vika2014@mail.ru и указать свою Ф.И.О, группу, и название дисциплины тел 072-17-44-9-22

Тема: Нормальные алгоритмы Маркова

Для формализации понятия алгоритма российский математик А.А. Марков предложил использовать ассоциативные исчисления.

Рассмотрим некоторые понятия ассоциативного исчисления. Пусть имеется алфавит (конечный набор различных символов). Составляющие его символы будем называть буквами. Любая конечная последовательность букв алфавита (линейный их ряд) называется словом в этом алфавите.

Рассмотрим два слова N и M в некотором алфавите A. Если N является частью M, то говорят, что N входит в M.

Зададим в некотором алфавите конечную систему подстановок $N - M$, $S - T$,..., где N, M, S, T,... -слова в этом алфавите. Любую подстановку N-M можно применить к некоторому слову K следующим способом: если в K имеется одно или несколько вхождений слова N, то любое из них может быть заменено словом M, и, наоборот, если имеется вхождение M, то его можно заменить словомN.

Например, в алфавите $A = \{a, b, c\}$имеются слова $N=ab$, $M = bcb$, $K = abcbcbab$, Заменив в слове K слово N на M, получим $bcbcbcbab$ или $abcbcbbcb$, и, наоборот, заменив M на N, получим $aabcbab$ или $abcabab$.

Подстановка ab – bcb недопустима к слову $bacb$, так как ни ab, ни bcb не входит в это слово. К полученным с помощью допустимых подстановок словам можно снова применить допустимые подстановки и т.д. Совокупность

всех слов в данном алфавите вместе с системой допустимых подстановок называют ***ассоциативным исчислением.*** Что бы задать ассоциативное исчисление, достаточно задать алфавит и систему подстановок.

Слова P_1иP_2в некотором ассоциативном исчислении называются смежными, если одно из них может быть преобразовано в другое однократным применением допустимой подстановки.

Последовательность слов $P,\ P_1,\ P_2,\ \ldots,\ M$ называется дедуктивной цепочкой, ведущей от слова *P к* слову *M,* если каждое из двух рядом стоящих слов этой цепочки - смежное.

Слова *P* и *M* называют эквивалентными, если существует цепочка от *P* к *M* и обратно.

Пример

Алфавит Подстановки

{*a, b, c, d, e*} *ac - ca, abac - abace*

ad - da; eca - ae

bc - cb; eda - be

bd - db; edb - be

Слова *abcde* и *acbde* - смежные (подстановка *bc - cb*). Слова *abcde-cadbe* эквивалентны.

Может быть рассмотрен специальный вид ассоциативного исчисления, в котором подстановки являются ориентированными: N → M (стрелка означает, что подстановку разрешается производить лишь слева направо). Для каждого ассоциативного исчисления существует задача: для любых двух слов определить, являются ли они эквивалентными или нет.

Любой процесс вывода формул, математические выкладки и преобразования также являются дедуктивными цепочками в некотором ассоциативном исчислении. Построение ассоциативных исчислений является универсальным методом детерминированной переработки информации и позволяет формализовать понятие алгоритма.

Введем понятие алгоритма на основе ассоциативного исчисления: алгоритмом в алфавите А называется понятное точное предписание, определяющее процесс над словами из А и допускающее любое слово в качестве исходного. Алгоритм в алфавите А задается в виде системы допустимых подстановок, дополненной точным предписанием о том, в каком порядке нужно применять допустимые подстановки и когда наступает остановка.

Пример

Алфавит: Система подстановок *B:*

А = {*a, b, c*}*cb* - *cc*

cca - *ab*

ab – *bca*

Предписание о применении подстановок: в произвольном слове *P*надо сделать возможные подстановки, заменив левую часть подстановок на правую; повторить процесс с вновь полученным словом.

Так, применяя систему подстановок *B*из рассмотренного примера к словам *babaac* и *bcacabc* получаем:

babaac→*bbcaaac*→ остановка

bcacabc → *bcacbcac*→*bcacccac*→*bcacabc*→ бесконечные процесс (остановки нет), так как мы получили исходное слово.

Предложенный А.А. Марковым способ уточнения понятия алгоритма основан на понятии нормального алгоритма, который определяется следующим образом. Пусть задан алфавит *A*и система подстановок *B.* Для произвольного слова *P* подстановки из *B* подбираются в том же порядке, в каком они следуют в *B*. Если подходящей подстановки нет, то процесс останавливается. В противном случае берется первая из подходящих подстановок и производится замена ее правой частью первого вхождения ее левой части в *P.* Затем все действия повторяются для получившегося словаP_1. Если применяется последняя подстановка из системы *B,* процесс останавливается.

Такой набор предписаний вместе с алфавитом *A* и набором подстановок *B* определяют нормальный алгоритм. Процесс останавливается только в двух случаях: 1) когда подходящая подстановка не найдена; 2) когда применена последняя подстановка из их набора. Различные нормальные алгоритмы отличаются друг от друга алфавитами и системами подстановок.

Приведем пример нормального алгоритма, описывающего сложение - натуральных чисел (представленных наборами единиц).

Пример

Алфавит: Система подстановок B:

A = (+, 1) 1 + → + 1

+ 1 → 1

1 → 1

Слово P: 11+11+111

Последовательная переработка слова *P* с помощью нормального алгоритма Маркова проходит через следующие этапы:

P = 11 + 11 + 111 P_5= + 1 + 111111

P_1= 1 + 111 + 111 P_6= ++ 1111111

P_2= + 1111 + 111 P_7= + 1111111

P_3= + 111 + 1111 P_8= 1111111

P_4= + 11 + 11111 P_9= 1111111

Нормальный алгоритм Маркова можно рассматривать как универсальную форму задания любого алгоритма. Универсальность нормальных алгоритмов декларируется принципом нормализации: для любого алгоритма в произвольном конечном алфавите *A* можно построить эквивалентный ему нормальный алгоритм над алфавитом *A,*

Разъясним последнее утверждение. В некоторых случаях не удается построить нормальный алгоритм, эквивалентный данному в алфавите *A,* если использовать в подстановках алгоритма только буквы этого алфавита. Однако, можно построить требуемый нормальный алгоритм, производя расширение алфавита *A* (добавляя к нему некоторое число новых букв). В этом случае

говорят, что построенный алгоритм является алгоритмом над алфавитом *A,* хотя он будет применяться лишь к словам в исходном алфавите *A*.

Если алгоритм *N* задан в некотором расширении алфавита *A,* то говорят, что *N* есть нормальный алгоритм над алфавитом *A*.

Условимся называть тот или иной алгоритм нормализуемым, если можно построить эквивалентный ему нормальный алгоритм, и не нормализуемым в противном случае. Принцип нормализации теперь может быть высказан в видоизмененной форме: все алгоритмы нормализуемы.

Данный принцип не может быть строго доказан, поскольку понятие произвольного алгоритма не является строго определенным и основывается на том, что все Известные в настоящее время алгоритмы являются нормализуемыми, а способы ромпозиции алгоритмов, позволяющие строить новые алгоритмы из уже известных, не выводят за пределы класса нормализуемых алгоритмов. Ниже перечислены способы композиции нормальных алгоритмов.

I. *Суперпозиция алгоритмов*. При суперпозиции двух алгоритмов *A* и *B* выходное слово первого алгоритма рассматривается как входное слово второго алгоритма *B*. Результат суперпозиции *C* может быть представлен в виде *C(p)=B(A(p)),*

II. *Объединение алгоритмов*. Объединением алгоритмов *A* и *B* в одном и том же алфавите называется алгоритм С в том же алфавите, преобразующий любое слово *p,* содержащееся в пересечении областей определения алгоритмов *A* и *B,* в записанные рядом слова *A (p)* и *B(p).*

III. *Разветвление алгоритмов*. Разветвление алгоритмов представляет собой композицию *D* трех алгоритмов *A, B* и *C,* причем область определения алгоритма *D* является пересечением областей определения всех трех алгоритмов *A, B*и С, а для любого слова *p* из этого пересечения *D(p)=A(p),*если *C(p)=e, D(p)=B(p),*если *C(p) = e,* где *e* -пустая строка.

IV. *Итерация алгоритмов*. Итерация (повторение) представляет собой такую композицию С двух алгоритмов *A* и *B,* что для любого входного слова

p соответствующее слово *C(p)*получается в результате последовательного многократного применения алгоритма *A* до тех пор, пока не получится слово, преобразуемое алгоритмом *B*.

Нормальные алгоритмы Маркова являются не только средством теоретических построений, но и основой специализированного языка программирования, применяемого как язык символьных преобразований при разработке систем искусственного интеллекта. Это один из немногих языков, разработанных в России и получивших известность во всем мире.

Существует строгое доказательство того, что по возможностям преобразования нормальные алгоритмы Маркова эквивалентны машинам Тьюринга.5