В. П. Камша, Л. С. Камша

О ПАРАДИГМЕ КОМПЬЮТЕРНОЙ ЛИНГВИСТИКИ

Демонстрируется нарушение законов Шеннона математической моделью языков, построенной на базе грамматик Хомского. Показано, что предложенное Хомским чисто математическое построение языков и грамматик, по сути, является отказом от системного подхода, от учёта тех законов Природы, на основании которых лингвистическими средствами осуществляются процессы передачи информации. Обосновывается необходимость единообразного системного подхода к исследованию информирования естественными и алгоритмическими языками — разными проявлениями одного и того же объективно существующего явления — с информационно-кибернетических позиций и на базе предложенных авторами распознающих грамматик.

1. ПОСТАНОВКА ВОПРОСА

Данная работа посвящена анализу недостатков, органически присущих ряду аксиом парадигмы компьютерной лингвистики (термин *парадигма* здесь и далее имеет смысл, предложенный Томасом Сэмюэлем Куном [1], — совокупность научных, методологических, философских и иных концепций, положенных в основу некоторой науки, в данном случае — компьютерной лингвистики). Первопричина этих недостатков — применение **порождающих** грамматик Хомского для **распознавания**, что стало возможным из-за ряда неписаных, но, тем не менее, строго соблюдаемых положений современной парадигмы компьютерной лингвистики:

- считается, что для исследования формальных свойств языка достаточно аппарата продукций и отношений (тем самым априорно было наложено табу на исследование информационных аспектов синтаксиса, что и предопределило появление анализируемых далее недостатков);
- грамматики рассматриваются только как способ задания множеств цепочек терминалов, что маскирует принципиальное различие между порождающими и распознающими грамматиками как взаимно обратными исчислениями, каждое из которых имеет свою область применения;
- компьютерная лингвистика как наука давно распалась на две совершенно самостоятельные ветви. Одна из них исследует естественные языки, в том числе и сленги всех профессий, за исключением одной — программистов: исследование и реализация алгоритмических языков образуют другую ветвь.

Вызвано это деление не спецификой объектов исследования. Напротив, в обоих случаях мы наблюдаем одно и то же явление Природы: передачу информации от автора к адресату посредством системы условных языковых знаков. Кардинально различаются только подходы к ним. Если естественные языки рассматриваются как средство общения, а сам процесс передачи информации языковыми средствами — лингвоинформирование — как процесс, объективно происходящий в Природе, то к алгоритмическим языкам относятся как к множествам, порождённым некоторыми математическими абстракциями — грамматиками. Такое разделение введено несмотря на то, что передача информации алгоритмическими языками — такое же объективное явление, как и передача информации естественными языками, а грамматика необходима в равной степени всем языкам.

Методика исследования языков соответствует упомянутым подходам к ним. Естественные языки исследуются естественно-научными методами (сравнением, экспериментами, накоплением и обобщением фактов), а алгоритмические — математическими методами (доказательством теорем и лемм, преобразованиями и другими аналогичными приёмами).

Вся противоречивость такого разделения языков становится очевидной, если вспомнить, что уже начат переход к использованию на ЭВМ естественных языков. Да и существовало это разделение не всегда. Длительное время все языки исследовали, базируясь на порождающих грамматиках Хомского. Они дали возможность компактно и наглядно описывать способы порождения всех цепочек терминалов для алгоритмических языков, а также значительно упростить контроль и трансляцию цепочек терминалов.

Предполагалось, что порождающие грамматики позволят решать аналогичным способом и проблему распознавания естественных языков. Однако все усилия достичь успеха в этом направлении оказались напрасны, что и вызвало в конце 1970-х гг. замену генеративно-трансформационной парадигмы новой, но только для естественных языков [2, 3]. Алгоритмические языки до сих пор реализуются на базе теории формальных грамматик, хотя накоплено достаточно много фактов, свидетельствующих, что и для них такой подход даёт отнюдь не оптимальные результаты. Логичнее, следовательно, было бы принять альтернативную оценку неудач: грамматики Хомского слишком грубо моделируют реалии лингвоинформирования. Их возможностей еще хватает для распознавания сравнительно простых алгоритмических языков, но недостаточно для распознавания естественных.

Поскольку реализация алгоритмических языков до сих пор базируется только на порождающих грамматиках, то сравнивать её просто не с чем. Существует, однако, эталон для программ, обрабатывающих тексты на естественных языках. Им является аппарат распознавания речи, дарованный нам Природой, — наш мозг.

Результаты сравнения оказываются обескураживающими. Не говоря уже о качестве распознавания, эффективность человеческого мозга несоизмеримо выше. Действительно, достаточно сопоставить рабочие частоты человеческого мозга (150 — 200 Гц) с мегагерцами современных ЭВМ, чтобы убедиться, как чудовищно низка эффективность распознавания речи современными лингвистическими процессорами естественных языков. Мозг человека, уступая ЭВМ несколько десятичных порядков по частоте, распознаёт тексты естественных языков в реальном масштабе времени и даже имеет солидный резерв, о чем свидетельствует возможность скорочтения. Обычная ЭВМ этого делать не может. Такое разительное несоответствие эффективности распознавания текстов на естественных языках человеком и современными ЭВМ неопровержимо свидетельствует о использовании ими совершенно разных технологий.

Еще одним красноречивым фактом, свидетельствующим не в пользу современных технологий распознавания текстов, может служить уникальность способа применения грамматик Хомского: из всех математических аппаратов только они, будучи ориентированы на один вид обработки — порождение, используются в основном для обратного действия — распознавания. Подавляющая часть работ по теории формальных грамматик также посвящена приспособлению порождающих грамматик для целей распознавания. Но если математический аппарат, ориентированный на некоторое действие, позволяет моделировать ему обратное, то можно рассчитывать на создание аппарата, непосредственно выполняющего это обратное действие значительно более эффективно.

Заметим, что к моменту смены парадигмы исследования естественных языков не только было известно, что существует альтернатива порождающим грамматикам — распознающие грамматики [4], но уже работал первый компилятор, базирующийся на них. Так как публикации по этой тематике имели малый тираж, перечислим вкратце основные этапы работы над распознающими грамматиками.

2. К ИСТОРИИ ВОПРОСА

Распознающие грамматики создавались в процессе конкретных прикладных разработок, что позволяло сразу же проверять на практике все полученные результаты, однако сильно замедлило темпы развития теории. Идея принципиально иной системы управления семантической обработкой текстов, позднее давшая начало распознающим грамматикам, была предложена для языка АЛГОЛ-60 во второй половине 1960-х гг. [5]. На её основе в Научно-

исследовательском институте управляющих вычислительных машин (НИИУВМ) был разработан первый компилятор, непосредственно (не применяя грамматики Хомского и деревья вывода) распознающий программы языка АЛГАМС — версии АЛГОЛ-60. Комплекс из этого компилятора и отладчика, также базирующегося на распознающих грамматиках, позволил совместить в одном компиляторе быстродействие скоростного с качеством объектной программы оптимизирующего, при значительном превышении диагностических возможностей их обоих [6].

Стало очевидным, что влияние анализаторов на скорость компиляции выходит далеко за рамки затрат времени на собственную работу. Определяя порядок и характер семантической обработки, анализатор решающим образом воздействует на скорость компиляции, однако близость результатов всех видов синтаксического анализа при одной и той же форме их представления не позволяют установить его истинную роль.

Действительно, при любом синтаксическом анализе его результат представлен деревом разбора полученной цепочки терминалов. Различие двух стратегий, в которые группируются все виды синтаксического анализа, сводится к способу построения дерева разбора: свёртыванием анализируемой цепочки в аксиому с помощью правил подстановки, или наоборот, порождением этими правилами цепочки терминалов, тождественной анализируемой. Изменение стратегии синтаксического анализа влияет на конечный результат — дерево разбора — лишь в той мере, в какой меняются используемые при этом грамматики, а они должны быть эквивалентны.

Обе стратегии синтаксического анализа фактически моделируют процесс распознавания путем вывода цепочки терминалов, тождественной заданной, что уже само по себе предполагает дополнительные затраты памяти и времени на организацию моделирования. Переход к распознающим грамматикам позволяет не только в несколько раз сократить затраты памяти и времени на управление семантической обработкой, но и оптимально её организовать, что даёт значительно больший эффект.

Приведенные общие соображения о природе неэффективности грамматик Хомского не могли, конечно, заменить анализа её конкретных причин, но в описываемый период авторы только начинали исследование распознающих грамматик, а в рамках парадигмы порождающих грамматик Хомского это невозможно было сделать. Первая версия аппарата непосредственного распознавания оказалась достаточно сложной, работа с ней плохо формализуемой, а область применения существенно ограниченной. Всё это не давало основания рекомендовать распознающие грамматики для широкого внедрения.

Следующий шаг в становлении распознающих грамматик был сделан в 1980-х гг. при разработке и реализации языков, ориентированных на конвейерную обработку данных, поступающих на вход в виде непрерывных потоков информации [7].

На этом этапе удалось упростить и формализовать распознающие грамматики, одновременно сняв ограничения на область их применения. Оговоримся, однако, что простота исчисления — понятие относительное. Проблемы распознавания, которые необходимо решить с их помощью, остались те же. Поэтому для систематического изложения результатов проделанных исследований, наиболее подходящей формой является монография, работа над которой ведётся. В процессе работы над ней было установлено, что совместить распознающие грамматики и новое понимание лингвоинформирования с парадигмой компьютерной лингвистики невозможно [8]. Изменение парадигмы любой науки всегда требует тщательной проверки и всестороннего обсуждения. Поэтому целесообразно вынести на всеобщее обсуждение и факты, противоречащие современной парадигме компьютерной лингвистики, а предлагаемые альтернативные подходы. В данной работе рассмотрим лишь корректность самой идеи формального определения языков и порождающих грамматик.

3. НЕФОРМАЛЬНЫЕ АСПЕКТЫ ФОРМАЛЬНЫХ ГРАММАТИК

Формальные грамматики Хомского определены как четвёрки, состоящие из терминального и нетерминального алфавитов, аксиомы и множества правил подстановок [9]. Выбор этих правил не имеет ограничений, если считать, что предварительно выбираются именно

они, а класс формируемой грамматики, терминальный и нетерминальный алфавиты определяются по уже сформированному множеству правил подстановки.

Однако продуцирование — это иерархия подстановок. Каждый последующий этап подстановок ограничен теми нетерминалами, которые порождены на предыдущих этапах. И если эта связь не находит своего отражения в определении порождающих грамматик, основной функцией которых является продуцирование, то теряется гарантия продуцирования ею цепочек терминалов. Что может, например породить «грамматика», включающая следующее множество правил:

$A => N_1$,

$N_2 => T$.

Она не порождает ни единой цепочки, и ей соответствует «язык», на котором ничего нельзя сказать. Таких пустых «языков» бесконечное множество, что явно много для вырожденной ситуации. В теоретических работах приходится учитывать возможность получения пустых «языков», несмотря на очевидную абсурдность этого понятия в лингвистике.

Таков первый результат формального определения порождающих грамматик.

Если формальный язык не пуст, это не означает, что в породившей его грамматике нет правил, на применение которых никогда нельзя выйти, начав продуцирование с аксиомы. Более того, формальную грамматику, не содержащую правил, которые не достижимы из аксиомы, можно преобразовывать в новые грамматики, добавляя сколько угодно таких правил. Следовательно, на каждую грамматику Хомского, не содержащую недостижимых правил, будет приходиться бесконечное множество грамматик с недостижимыми правилами, порождающие этот же формальный язык.

Когда среди недостижимых правил встречаются содержащие терминал, не используемый в достижимых правилах, то такой терминал никогда не сможет появиться в цепочках этого языка, хотя формально он и будет входить в терминальный алфавит, как принадлежащий одному из правил подстановки. Однако число терминальных символов алфавита входит в формулы Шеннона [10]. Поэтому расчёт по этим формулам количества информации для тех цепочек терминалов, которые порождены грамматиками Хомского, содержащими недостижимые терминалы, будет давать завышенный результат.

Таков второй результат формального определения порождающих грамматик.

На практике недостижимые правила удаляются, что рассматривается как оптимизирующие преобразования, осуществляющие переход к эквивалентной грамматике. Эквивалентность, как известно, отличается от тождественности тем, что совпадение гарантируется не по всем, а лишь по некоторым параметрам. В данном случае полностью совпадают множества генерируемых цепочек терминалов, но заведомо различны породившие их грамматики Хомского. И различаются они не только правилами подстановки, но в алфавитами. В частности, терминальный алфавит грамматики без недостижимых правил всегда позволяет точно рассчитывать по формулам Шеннона количество информации, передаваемое любой цепочкой терминалов.

Изменения, устраняющие неправильные результаты, принято называть не оптимизацией, а исправлением, а заменяемое понятие — ошибочным. Тем самым рассмотренные примеры дают основание считать формальное определение грамматик Хомским некорректным, так как оно включает и четвёрки, которые вообще неспособны породить хотя бы одну цепочку терминалов, и четвёрки, приводящие к ошибочным результатам при определении количества информации, передаваемой цепочками терминалов.

Однако недостижимые символы хотя бы не нарушают сам процесс генерации цепочек терминалов из аксиомы, в то время как непродуктивные символы делают этот процесс практически невозможным. Действительно, рассмотрим порождающую грамматику Хомского, содержащую следующие правила подстановки:

(1) $A => N_1$,

(2) $A => N_2$,

(3) $N_1 => T\ N_1$,

(4) N_1 => T,

(5) N_2 => T N_2.

Если первой будет использована продукция (1), то применение продукций (3) и (4) даст бесконечное множество различных цепочек терминалов. Однако стоит применить вторую продукцию, и процесс генерации уже никогда не сможет завершиться: нетерминал N_2 при его замене порождает такой же новый нетерминал, и избавиться от него нет никакой возможности. Поэтому, если продуцирование не самоцель, а аппарат для генерации цепочек терминалов, то требование продуктивности также должно найти свое отображение в определении порождающей грамматики, иначе возникает угроза потери работоспособности грамматики из-за зацикливания.

Иными словами, предварительная «оптимизация», а если называть вещи своими именами — исправление ошибок в определении порождающих грамматик по Хомскому — исключает de facto из множества порождающих грамматик неработоспособные четвёрки. Множество формальных языков от этого не меняется, так как исключаемые четвёрки либо вообще ничего не могут породить, либо существует грамматика, в точности содержащая те правила подстановки, которые могут использоваться в рассматриваемой четвёрке для генерации цепочек терминалов. Именно ею заменяется неработоспособная четвёрка, что гарантирует сохранение дерева вывода для каждой цепочки терминалов. Избежать такого предварительного отсева нельзя без угрозы зацикливания и даже полной неработоспособности анализатора. Удалим, например, правило (1) или (4) из рассматриваемого множества подстановок, и зацикливание станет неизбежным.

Не решает проблемы и введение ограничения на применение продукций тех нетерминалов, для которых отсутствует хотя бы одно правило подстановки, правая часть которого не содержит этот же нетерминал. Действительно, добавим в рассматриваемую грамматику еще пару правил подстановки:

(6) N_2 => T N_3,

(7) N_3 => T N_2.

Теперь правило (6) не содержит в правой своей части нетерминала N_2, но тем не менее оно остается непродуктивным. Очевидно существование четвёрок Хомского, для которых установление непродуктивности того или иного правила подстановки может потребовать анализа сколь угодно большого числа других правил. Только установив все недостижимые и непродуктивные правила подстановки, можно обеспечить безотказную генерацию цепочек терминалов, исключив их еще до начала использования грамматики, либо устранив каким-либо образом возможность их применения в процессе генерации цепочки терминалов. В обоих случаях мы переходим к неформальным ограничениям и новым множествам порождающих грамматик, так как и в последнем случае мы их также исключили, но только на этапе использования грамматики. Альтернативой этих ограничений является неработоспособность значительной части грамматик Хомского.

Таков третий результат формального определения порождающих грамматик.

Грамматики Хомского могут содержать тривиальные продукции, априорно не меняющие цепочку, и, в силу этого, абсолютно бесполезные даже при наличии альтернативных правил подстановки. Как и в предыдущем случае, эту ситуацию можно обобщить на любую последовательность продукций вида:

N => N_1 => N_2 => … => N.

Осуществляемая на практике проверка грамматик на отсутствие какого-либо из рассмотренных выше случаев является признанием того факта, что далеко не каждая совокупность правил подстановки пригодна для генерации цепочек терминалов. Исключение избыточных для генерации правил подстановок устраняет также и проблему грамматик, порождающих пустые языки, так как из всего этого обширного множества остаётся единственная грамматика с пустым терминальным алфавитом и только одним правилом: .A => *e*, где *e* — пустая цепочка. Такая действительно вырожденная ситуация может быть исключена из множества порождающих грамматик.

Все рассмотренные факты давно известны. Поэтому возникает ряд закономерных вопросов: почему им не дана надлежащая оценка, в чем причина непригодности значительной

части формальных грамматик Хомского даже для выполнения своей прямой функции — генерации цепочек терминалов, наконец, зачем понадобилось использовать их для распознавания, если они ориентированы на совершенно другую — порождающую функцию? Ответ на все эти вопросы можно получить из следующей цитаты:

«При определении и реализации переводов часто бывает удобнее рассматривать перевод как композицию двух и более простых отображений. Первое из них, называемое синтаксическим отображением, связывает с каждым входом (программой в исходном языке) некоторую структуру, которая служит аргументом второго отображения, называемого семантическим. Сразу не ясно, что должна существовать какая-то структура, помогающая осуществить перевод, но почти всегда той структурой, которую полезно придать входной программе, оказывается помеченное дерево. Не углубляясь в философию о том, почему это так, мы посвятим большую часть книги алгоритмам эффективного построения подходящих деревьев для входных программ.» [11].

Эта цитата, во-первых, демонстрирует уровень обоснованности базовых положений формальных грамматик: во-вторых, она объясняет, что формальные грамматики Хомского привлечены для целей распознавания как аппарат структурирования цепочки терминалов; в-третьих, содержит в неявной форме и ответ на причину непригодности формального определения грамматик: ведь нельзя наделить тем, чем сам не обладаешь.

Поскольку процесс порождения цепочки терминалов самим фактом реализации придаёт ей иерархическую структуру в форме дерева вывода, то продуцирование не может быть осуществлено, если множество правил подстановки само не образует иерархическую систему, которая гарантирует возможность начала и завершения процесса порождения и допустимость применения каждого из правил. Продуктивность, достижимость и неизбыточность всех правил и являются теми неформальными требованиями, которые выделяют из множества четвёрок Хомского подмножество, пригодное для генерации цепочек терминалов, которое и используется в алгоритмических языках. Тем самым практика вступает в непримиримое противоречие с концепцией теории, декларирующей полную формализуемость синтаксиса. Что же касается естественных языков, то грамматики Хомского в принципе для этого непригодны, так как они порождают не меняющиеся во времени множества цепочек терминалов, а основным свойством естественных языков является их непрерывное изменение во времени. Именно за это их называют живыми. Изменение языков имеет два аспекта:

- меняется запас слов и уровень владения правилами грамматики каждого носителя данного языка;
- меняется запас слов и правила грамматики всех пользователей.

Ребенок начинает с применения отдельных слов, запас нарастает, слова начинают объединяться в пары, затем в группы. Усваиваются правила грамматики. Этот процесс продолжается в течение всей жизни.

Развиваются и сами естественные языки. Появляются всё новые и новые слова, а некоторые старые меняют смысл. Так, всего лишь одно столетие назад могла ли быть понятна фраза *Механизатор МТС радиофицировал колхоз*? Меняются и грамматические правила. Но если грамматики Хомского в принципе непригодны для моделирования таких изменений при генерации текстов или предложений, то чего же можно ожидать от их применения для распознавания — функции, на реализацию которой они и не рассчитаны. Ведь грамматики Хомского не могут на практике обеспечить распознавание даже жёстко фиксированных алгоритмических языков, цепочки которых они генерируют.

Действительно, распознавание цепочек терминалов только с помощью одних грамматик Хомского требует перебора правил подстановки на каждом шаге анализа с возвратами при неудачах [11]. Время распознавания возрастает гораздо быстрее, чем длина цепочки терминалов, и становится неприемлемо большим даже при относительно коротких цепочках. Кроме того, грамматики Хомского не содержат понятия «ошибка», и её появление приводит к зацикливанию синтаксических анализаторов такого типа. Поэтому на практике грамматики Хомского применяют для анализа только вместе с матрицами отношений разного вида. Но в состав

порождающих грамматик Хомского никакие отношения не входят. Это уже новый аппарат, построенный специально для распознавания. Другой вопрос, насколько целесообразно включать сами порождающие грамматики в состав распознающего аппарата.

4. НЕДОСТАТКИ СИНТАКСИЧЕСКОГО АНАЛИЗА

Лингвоинформирование осуществляется между двумя участниками этого процесса — автором цепочки терминалов и адресатом. Чтобы оно состоялось, недостаточно передать адресату эту цепочку: необходимо, чтобы по ней можно было синтезировать сообщение. На ограничивающую роль распознавания указывал и сам Хомский [12].

Синтез сообщения в последовательности, строго обратной порождению цепочек терминалов по дереву вывода, логически наиболее прост, но требует восстановления дерева вывода в форме дерева разбора. Именно этот способ и положен в основу синтаксического анализа, который можно интерпретировать как ретроградный анализ со всеми свойственными ему переборами бесчисленных вариантов. Избежать их можно, только опираясь на дополнительную информацию. Её получают, используя таблицы отношений разного вида. Однако часто эти отношения оказываются неоднозначными. Возврат к перебору вариантов генерации цепочки терминалов грамматикой, как сказано выше, потребует слишком много времени. Поэтому на практике пытаются устранить неоднозначность, применяя такие преобразования грамматик, которые априорно сохраняют не только цепочки, но и сообщения. Именно их далее будем называть эквивалентными.

Так, чтобы установить различные отношения предшествования, применяют эквивалентные преобразования, вводящие новые нетерминалы (стратификация) [13]. Стратификация усложняет грамматику, но априорно не влияет на сообщения цепочек терминалов и даёт возможность в ряде случаев добиться однозначности отношений какого-либо вида. Тем самым стратификация иногда позволяет получить эквивалентную грамматику, пригодную для применения конкретного метода синтаксического анализа.

Если таким способом не удается построить анализатор, не требующий перебора вариантов, пытаются применить иные виды отношений и другие эквивалентные преобразования. Но нет универсального метода, позволяющего определить, разрешима ли эта конкретная задача. Если для нее решение имеется, то априорно нельзя установить, какой вид следует придать грамматике.

Существуют десятки вариантов синтаксического анализа, и для каждого языка можно построить бесконечное множество эквивалентных грамматик. Но если подбор эквивалентной грамматики, позволяющий применить один из этих синтаксических методов, увенчался успехом, то деревья разбора цепочек терминалов будут отличаться от деревьев вывода в исходной грамматике.

Сказанное свидетельствует о том, что информация, получаемая в результате синтаксического анализа, избыточна по крайней мере в той своей части, которая отличает полученное дерево разбора от всех деревьев вывода этой же цепочки, порожденных грамматиками, эквивалентными данной, но синтаксический анализ невозможен без какого-либо её варианта. Однако получить фактически ненужную для синтеза сообщения информацию не всегда возможно, что и приводит к множественности способов синтаксического анализа.

Заметим, что на эффективность лингвистической обработки в значительно большей степени влияет форма представления результатов анализа деревом разбора, заимствованным из аппарата генерации. С одной стороны, дерево разбора включает информацию, необходимую для процесса генерации цепочек терминалов, но избыточную, как мы видели, при распознавании, а с другой — не содержит информации, позволяющей упростить семантическую обработку при распознавании, так как она не требуется при генерации. Например, не включено понятие «ошибка». Поэтому частично нарушения синтаксиса устанавливаются отношениями, а окончательно — поиском соответствующего правила подстановки.

Возможность эквивалентных преобразований грамматики лишь несколько смягчает, но отнюдь не устраняет последствия избыточности. Аппарат порождающих формальных грамматик в принципе не позволяет отделить действительно необходимую последовательность синтеза сообщения от остальной, избыточной информации, входящей в дерево разбора. Их можно получить только вместе или не получить ничего. А это ведет к ряду отрицательных последствий, из которых наиболее очевидны следующие:

- получение информации, не нужной для синтеза сообщения, требует дополнительного расхода памяти и времени;
- избыточная информация может вызывать на семантическом уровне только избыточные действия:
- невозможность установить именно избыточную информацию сокращает область применения конкретного метода распознавания цепочек терминалов и побуждает создавать все новые варианты анализаторов.

По мере усложнения грамматик Хомского эффективность синтаксического анализа уменьшается, и это можно показать на примере грамматики, порождающей простейшие арифметические выражения из шести правил [11]. Если расширить её до уровня, характерного для алгоритмических языков (например, АЛГОЛ-60), то число правил удвоится и, соответственно, увеличится дерево вывода, что видно из рис. 1 и рис. 2, изображающих дерево вывода одного и того же выражения *I + I * I*, где под каждым *I* подразумевается идентификатор любой простой переменной. На рис. 2 приведены только правила подстановки, необходимые для построения дерева вывода. Нетерминалы в этих правилах изображены буквами, совпадающими с обозначениями, принятыми для грамматики на рис 1, а именно:

арифметическое выражение — A,
терм — T.
первичное выражение — F,
идентификатор — I.

Нетерминалы, существующие только в языке АЛГОЛ-60, обозначим:

простое арифметическое выражение — S,
множитель — M,
переменная — U.
простая переменная — W,
идентификатор переменной — X.

```
A ::  = A + T
A ::  = T
T ::  = T * F
T ::  = F
F ::  = (A)
F ::  = I
```

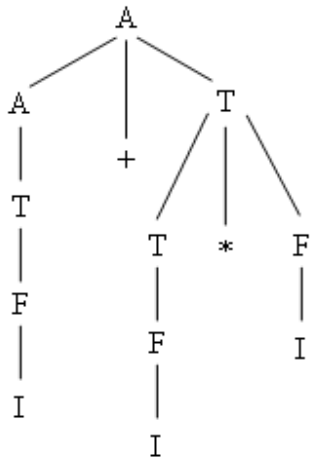

Рис. 1. Дерево вывода выражения *I + I * I* и простейшая грамматика, его порождающая

```
A ::= S
S ::= S + T
S ::= T
T ::= T * M
T ::= M
M ::= M ** F
M ::= F
F ::= (A)
F ::= U
U ::= W
W ::= X
X ::= I
```

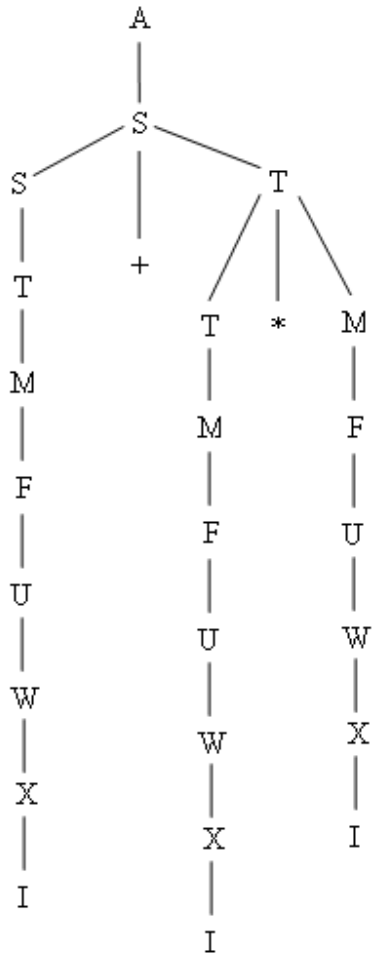

Рис. 2. Дерево вывода выражения *I* + *I* * *I* и фрагмент грамматики языка АЛГОЛ-60, используемый при его выведении

Как видно из рисунков, расширение возможностей языка приводит к усложнению его грамматики и деревьев вывода. И хотя все цепочки терминалов, порождаемые грамматикой, изображенной на рис. 1, вошли со своими сообщениями в язык, порождаемый грамматикой, изображенной на рис. 2, их деревья вывода значительно усложнились, а следовательно, увеличилось число шагов синтаксического анализа и количество избыточной информации, выдаваемой анализатором.

Преобразование грамматики к виду, пригодному для распознавания цепочек терминалов данного языка, обычно ведет к дальнейшему усложнению деревьев вывода, так как их основная цель — однозначность применяемых отношений. Устранение неоднозначностей требует дополнительной информации, которую, согласно формулам Шеннона [10], можно получить, увеличив нетерминальный алфавит, а значит, и количество правил подстановки в грамматике. Если привести, например, грамматики, изображенные на рис. 1 и рис. 2, к виду, обеспечивающему однозначные отношения простого предшествования, то можно убедиться, что грамматика на рис 2 требует включения большего числа дополнительных правил.

Таким образом, синтаксическому анализу свойственны принципиально неустранимые недостатки, которые проявляются тем сильнее, чем сложнее язык. Если сравнить распознавание сообщений человеком и существующими лингвистическими процессорами, обнаруживается колоссальное падение эффективности при переходе от алгоритмических языков к несравнимо более сложным естественным.

5. О СИСТЕМНОМ ПОДХОДЕ В ЛИНГВОИНФОРМИРОВАНИИ

Рассмотрим теперь причину неудач чисто математического подхода к языкам. Само собой разумеется, что математика может исследовать своими методами любые объекты, в том числе и четвёрки Хомского. Однако при каждом прикладном использовании полученных результатов необходимо установить границы их корректного применения.

Математика всегда идеализирует объекты своего исследования, абстрагируясь от ряда их конкретных свойств. Но и в чисто математическом плане реальные объекты часто не совпадают с идеальными. Математический аппарат иногда описывает некоторый реальный объект лишь частично, а иногда рассматривает его как часть исследуемого множества. Если в первом случае приходится следить за корректностью применения математического аппарата к конкретной задаче, то во втором необходимо учитывать последствия абстрагирования от индивидуальных свойств используемого математического аналога. Оба этих случая могут происходить с одним и тем же объектом исследования, что мы и наблюдаем на примере теории формальных грамматик, применяемых к исследованию языков.

Действительно, как было показано выше, формальные грамматики содержат четвёрки, которые не пригодны даже для выполнения своей основной функции — порождения цепочек терминалов. Тем более они не пригодны для распознавания, на которое и не были рассчитаны. Как уже упоминалось, раз и навсегда зафиксированные четвёрки Хомского определяют неизменные множества цепочек терминалов, и поэтому в принципе не способны порождать все возможные предложения живых естественных языков с их семантикой и словарным запасом, находящимися в постоянном развитии. Таким образом, теория формальных грамматик исследует объекты, которые далеко не идентичны реальным объектам лингвоинформирования. Некоторые последствия таких несоответствий реального и идеального объектов мы уже рассмотрели.

Однако подмена объекта исследования этим не ограничивается: известно, что изменение даже единственного правила грамматики может оказаться достаточным, чтобы цепочка, сгенерированная из тех же самых терминалов, стала передавать совершенно другое сообщение [8]. Именно сообщение, а не цепочка терминалов является конечной целью лингвоинформирования. Поэтому исследование цепочек терминалов и порождающих их грамматик в отрыве от передаваемого этими цепочками сообщения — путь, заведомо игнорирующий реалии лингвоинформирования. Действительно, как мы видели, и самый беглый анализ показывает несовместимость принятого определения формальных грамматик с законами Шеннона.

На примере лингвоинформирования мы убедились, что исследование даже чисто формальных аспектов процессов и сущностей реального мира без системного подхода во-первых, приводит к игнорированию уже установленных законов Природы и, следовательно, просто к ошибочным результатам, а, во-вторых, — к чрезвычайно резкому ограничению области исследования, так как эти законы должны выполнять роль дополнительных независимых аксиом, которые и позволяют развить теорию значительно глубже.

Наконец, попытка заменить объективное исследование процессов и сущностей Природы «вычислением» её законов только чисто математическими средствами приводит к нарушению законов и логики самой математики из-за явной недостаточности исходных посылок. Действительно, современная теория формальных грамматик допускает два способа определения бесконечных множеств цепочек терминалов, образующих некоторый формальный язык:

- способ систематического порождения цепочек терминалов заданного языка и только их;
- способ распознавания принадлежности любой цепочки терминалов конкретному языку [4].

Первому способу соответствуют порождающие грамматики, а второй так и остался теоретической возможностью под названием «распознающие грамматики» [4]. Отчасти это можно объяснить предвзятым отношением к ним, как к еще одному способу определения формальных языков, не обещающему никаких интересных результатов ни в практическом, ни в теоретическом плане.

Иную оценку подсказывает нам математика — официальный инструментарий исследования теории формальных грамматик. С позиций математики передача информации языковыми средствами распадается на две взаимно обратные задачи:

- по информации, подлежащей передаче, сгенерировать цепочку терминалов;
- по полученной цепочке терминалов синтезировать сообщение.

Сам факт решения проблем компьютерной лингвистики парой взаимно обратных задач типичен для математики, в которой основная часть операций, функций или исчислений группируется в пары:

- сложение и вычитание;
- логарифмирование и потенцирование;
- дифференцирование и интегрирование.

Таким образом, исключительным является не само существование пары взаимно обратных задач в компьютерной лингвистике, а способ, которым они решаются:

- грамматики Хомского не ориентированы ни на одну из упомянутых выше задач передачи информации;
- являясь порождающими, они применяются, в основном, для решения обратной задачи — распознавания.

Постановка двух задач — порождения и распознавания цепочек терминалов — позволяет рассматривать распознающие грамматики не как еще один способ определения цепочек терминалов, а как исчисление, обратное порождающим грамматикам. Подобный подход продуктивен в двух отношениях:

- во-первых, он позволяет получить универсальный и значительно более эффективный и компактный аппарат распознавания, который непосредственно реализует именно эту функцию, и поэтому наилучшим образом ориентирован на нее и только на нее;
- во-вторых, он выводит нас на новый уровень понимания сущности лингвоинформирования, что даёт новые возможности для развития ряда отраслей науки и производства.

Вся история математики свидетельствует, что именно переход к решению обратных задач способствовал развитию математических понятий. Так, поиск неизвестного слагаемого привел нас от натуральных чисел к целым, деление — от целых к рациональным, извлечение корня и логарифмирование — к иррациональным и комплексным числам. Это требовало усложнения и уточнения алгоритмов сложения, умножения, возведения в степень и расширяло область их применения.

Исследование переменных поставило вопросы о скорости их изменения, максимумах и минимумах, что породило дифференциальное исчисление. Решение обратных задач привело к интегральному исчислению, которое обогатило математику понятием предела и новой методологией, известной как язык эпсилон-дельта.

Переход к распознающим грамматикам также невозможен без коренного пересмотра понятия «порождающая грамматика». Даже поставить в этой статье задачу перехода на распознающие грамматики оказалось невозможно без существенной корректировки этого понятия.

Существует аналогия между проблемами современной компьютерной лингвистики и геометрии накануне появления интегрального исчисления. Она не случайна. Как в наше время для распознавания цепочек терминалов применяется множество различных приемов, так и в те времена существовали десятки методов, позволяющих получать формулы для подсчета площадей, поверхностей и объемов криволинейных фигур и тел. В одних случаях они приводили к успеху, в других — оказывались бесполезными. Приступая к определению площади или объема нового объекта, нельзя было заранее сказать, какой из приемов приведет к успеху и возможен ли он вообще. Кардинальным решением не только этих, но и многих других вопросов, поставленных наукой и техникой эпохи НТР, был переход к интегральному исчислению.

Сейчас на передний план выдвигаются задачи совершенно иного типа — автоматизации управления. Допустимое множество последовательностей управляющих сигналов образует язык.

Контроль правильности последовательностей таких сигналов, их распознавание и реализация — чисто языковые проблемы, которые могут решаться надежно и по единой методике только после отказа от применения порождавших грамматик для реализации несвойственной им функции распознавания цепочек терминалов.

6. ВЫВОДЫ

Информирование как естественными, так и любыми искусственными языками — объективный процесс, происходящий в Природе в соответствии с её законами. Управлять этим процессом можно лишь опираясь на эти законы, а не игнорируя их.

Порождающие грамматики Хомского — чисто математическое построение, не учитывающее как законы передачи информации, так и свойства живых языков, и поэтому эти грамматики определены некорректно: многие из них не могут порождать ничего, применение других угрожает зацикливанием, третьи приводят к неправильному определению количества информации, передаваемого генерируемыми цепочками терминалов. Избавиться от этих недостатков можно, только вводя неформальные ограничения, обеспечивающие достижимость, продуктивность и неизбыточность множества правил подстановки.

Применение для распознавания порождающих грамматик Хомского требует избыточной информации, что является причиной множественности методов синтаксического анализа и их ограниченности, а представление результата в форме дерева разбора, ориентированного на порождение, ведет к большим дополнительным затратам памяти и машинного времени как на синтаксическом, так и на семантическом уровне.

Вынужденный отказ от генеративно-трансформационной парадигмы для естественных языков и сохранение традиционного синтаксического анализа для алгоритмических — основная причина деления компьютерной лингвистики на две несовместимые ветви.

Непригодность порождающих грамматик Хомского для естественных языков и их низкая эффективность при исследовании и реализации алгоритмических языков вызвана недостаточно хорошим соответствием теории формальных грамматик реальным процессам передачи информации средствами языка, объективно происходящими в Природе, а также отсутствием системного подхода при разработке математического аппарата лингвоинформирования.

Избавиться от перечисленных недостатков можно, только пересмотрев парадигму лингвистики. При разработке новой необходимо опираться на системный подход и методологию, базирующуюся на информационно-кибернетическом анализе лингвоинформирования и его компонентов, а также на распознающих грамматиках, как исчислении, ориентированном на синтез сообщений.

СПИСОК ЛИТЕРАТУРЫ

1. Кун Т. С. Структура научных революций. – М.:Прогресс, 1977.— 300 с. (Kuhn T. C. The structure of scientific revolutions. — Chicago, 1970.)
2. DeBeavgrande R. Text, Discourse and Process. Hillsdale. – N.1: Erlbaum, 1980.
3. DeBeavgrande R. and Dressler W. Introduction to Text Linguistics. – London: Longman, 1981.
4. Алферова Э. В. Теория алгоритмов.— М: Статистика, 1973.— 164 с.
5. Камша В. П. Структура транслятора с языка АЛГОЛ-60 и контроль исходной программы // Агрегатная система средств вычислительной техники.— Киев: УкрНИИНТИ, 1969.— С. 46 — 52.
6. Труды Научно-исследовательского института управляющих вычислительных машин.— Северодонецк, 1971. Вып. 3. — С. 80 — 138.
7. Камша Л. С. Язык конвейерной обработки данных // Программирование- 1990.— N. 2. — С. 55 — 66.

8. Камша В. П., Камша Л. С. Некоторые вопросы, парадигмы компьютерной лингвистики // Всесоюз. конф. «Искусственный интеллект-90». Круглые столы.— Минск, 1990. — С. 33 — 35.
9. Хомский Н. О некоторых формальных свойствах грамматик // Кибернетический сборник. — М: ИЛ, 1962. — Вып. 5. — С. 279 — 311. (Chomsky N. On certain formal properties of Grammars // Inform. and Control. 1959.)
10. Shannon C. E., Weawer W. The Mathematical Theory of Communication. – Urbana: U. of Illinois Press, 1949. – P. 3 – 28.
11. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. — М: Мир, 1978. — Т. 1. — 612 с. (Aho A., Ullman J.D. The Theory of Parsing, Translation and Compiling. Vol. 1, N. J. Prentice — Holl. inc., Englewood Cliffs, 1972.)
12. Хомский Н. Три модели для описания языка. Кибернетический сборник. — М: ИЛ, 1961. — Вып. 2. — С. 237 — 266. (Chomsky N. Three models for the description of language // IEEE Trans. Inform. Theory. 1956. 2:3. — P. 113 — 124.)
13. Грис Д. Конструирование компиляторов для цифровых вычислительных машин. — М: Мир, 1975 .— 544 с. (Gries D. Compiler *Construction* for Digital Computers. — New York; Londons; Sydney; Toronto, 1971.)