**Порождение текстов на естественном языке**

Порождение текстов на естественном языке - процесс преднамеренного построения текста на естественном языке с целью решать определенные коммуникативные задачи. Термин "текст" рассматривается как общий, рекурсивный термин, который может относится к письменному или устному высказыванию, или к отдельным частям высказывания. При порождении текстов, в устной или письменной форме, человеку важно обдумать и отредактировать производимое высказывание. Едва ли можно сказать, что большинство программ может “говорить” сегодня, в основном все они лишь выводят слова на экран. Так как для программы порождения текстов на сегодняшний день не стоит вопрос конструирования фразы, эти детали принимаются во внимание только тогда, когда они задействованы в создании программы.

Цели исходят из другой программы, возможно экспертной рассуждающей системы или ICAI обучающей программы, которая общается с пользователем на естественном языке. Произведенные тексты могут быть различной длины: от одиночной фразы, данной в ответ на вопрос, до диалогов с большим количеством предложений или толкований на целую страницу. Порождение текстов на естественном языке отличается от программ, просто использующих естественный язык. Программы, печатающие сообщения на естественном языке, существуют со времен появления компьютеров, но сейчас, например, никто не хочет разбираться, каким образом построены сообщения об ошибках при компиляции на ФОРТРАНе, как бы правильно они не были написаны. Сообщение об ошибках ничего не "означает" для программы, которая печатает их: связь между цепочкой слов и работой программы создается программистом. Даже использование утверждений с параметром, где зафиксированная цепочка слов может быть увеличена именами или простыми описаниями, заменяющими переменные, не является собственно порождением текстов на естественном языке. Успех таких приемов как “заполнить пробелы” или “шаблон” зависит от количества и сложности ситуаций, в которых программа должна использовать их. То, что они были адекватны до сих пор для работы программы, объясняется, по большей части, относительной простотой сегодняшних программ, чем возможностями порождения с использованием метода “шаблона”.

В отличие от таких "инженерных разработок", исследование порождения текстов на естественном языке, подобно другим областям вычислительной лингвистики (qv), имеет своей целью компьютерное моделирование человеческой способности к порождению высказываний. Основное внимание при этом сосредотачивается на объяснении двух ключевых вопросов: многосторонность и творческий потенциал. Что люди знают относительно их языка, какие процессы они при этом используют, что дает возможность им быть универсальным, изменяя тексты в форме и акцентировании, чтобы покрыть огромный диапазон языковых ситуаций?

В этой статье описываетcя исследование в области ИИ по порождению естественных языков, при этом особое внимание уделяется конкретным проблемам, которые требуют разрешения. Статья начинается с противопоставления порождения пониманию, чтобы установить базисные понятия разложения процесса на компоненты. Далее приводятся примеры, показывающие работу некоторых порождающих систем, их возможности и трудности, с которыми они сталкиваются.

В оставшейся части статьи рассматриваются общие подходы к порождению речи, включая характерные описания порождающего словаря. Отдельный раздел продолжает обзор альтернативных подходов к представлению и использованию грамматики.

**Характер процесса порождения.** В отличие от организации процесса понимания, который, на первый взгляд, может следовать традиционным стадиям лингвистического анализа: морфология, синтаксис, семантика, прагматика /дискурс¦ процесс порождения имеет существенно отличный характер. Этот факт следует непосредственно из присущих различий в информационном потоке в двух процессах. Понимание осуществляется от формы к содержанию; порождение есть совершенно противоположный процесс. При понимании, формулировка текста (и, возможно, интонация) - "известны". Из формулировки процесс создает и выводит примерное содержание, переданное текстом и, вероятно, усилиями диктора в создании текста. Первым делом следует просмотреть слова текста последовательно, в течение чего форма текста постепенно разворачивается. Главные проблемы вызваны неоднозначностью¦ одна форма может содержать диапазон альтернативных значений, и аудитория получает большее количество информации из ситуационных заключений, чем это может быть фактически передано текстом. Кроме того, несоответствия у диктора и аудитории модели ситуации ведут к непредсказуемым заключениям.

Порождение имеет противоположный информационный поток. Оно переходит от содержания к форме, от целей и перспектив к линейно упорядоченным словам и синтаксическим маркерам. Модель ситуации и дискурс обеспечивают основу для создания выбора среди альтернативных формулировок и конструкций, которые производит язык: первое в построении заранее обдуманного текста. Большинство систем порождения производит поверхностные тексты последовательно слева направо, но только приняв решение сверху-вниз по содержанию и форме текста в целом. Проблема генератора состоит в том, чтобы выбрать из поставленных источников, как правильно сообщить о желаемых умозаключениях аудитории и какую информацию опустить из явного упоминания в тексте.

Можно вообразить, что процесс порождение также организован, как и процесс понимания, только в противоположном порядке. К некотором смысле это верно: идентификация намерения (цели) в значительной степени предшествует любой детализации информация, которая предназначается для аудитории: планирование риторической структуры, например, в значительной степени, предшествует любой синтаксической структуре, а синтаксический контекст слова должен быть зафиксирован, прежде чем будут известны морфологическая и суперсегментная формы, которые примет слово.

Синтаксис и словарь языка становится как ресурсами, так и ограничениями, определяя элементы, доступные для создания текста, а также зависимости между ними, которые определяют возможные правильные комбинации. Эти зависимости, и тот факт, что они по умолчанию управляют, когда информация, от которой зависит каждое решение, становится доступной, - основная причина, почему программы порождения в значительной степени следуют стандартным стадиям, определенными лингвистами. Идентификация цели предшествует выбору содержания и риторическому планированию, которое предшествует синтаксической конструкции, только потому что это - естественный порядок принятия решения; проще следовать потоку зависимостей, чем перепрыгивать и принимать случайное решение, которое может оказаться преждевременным и несостоятельным. Сегодняшнее исследование сосредоточено как на понимании, как лучше представить решения, которые являются возможными, и зависимости среди них, так и на том, как представить ограничения и возможности раньше решений, которые встанут на место последних во время процесса порождения.

**Стандартные Компоненты и Терминология.** Компоненты порождения естественного языка не существуют сами по себе. Они расположены внутри человеко-машинного интерфейса, который также используют и компоненты понимания естественного языка, - ВВОД в систему. В хорошем человеко-машинном интерфейсе сегодня также хотелось бы видеть координированную графическую поддержку ввода и вывода, дополняя систему ВВОДа-ВЫВОДа естественного языка. Интерфейс может закончиться здесь, а может также включать в себя другие общедоступные компоненты, типа контроллера дискурса, который указывает генератору, какие действия нужно предпринять, а также координирует интерпретации, сделанные компонентом понимания. За интерфейсом следует ­нелингвистическое рассуждение (qv) или программа базы данных, которую пользователи используют в качестве речевого интерфейса. Эта программа будет упоминаться в этой статье как основная программа; ею может оказаться любая система ИИ: совместная база данных, экспертная диагностическая система, ICAI обучающая программа, комментатор, программа-консультант, машинный переводчик. Тип основной программы теперь не имеет никакого значения для самой порождающей системы (генератора естественного языка).

Сегодня большинство исследователей в этой области работает, в основном, с экспертными системами, где процесс общения контролируется программой, а не пользователем. Кроме того, ЭС и интеллектуальные машинные обучающие программы, вероятно, способны понимать довольно сложные тексты, что делает их привлекательными для специалистов, готовых работать с уже разработанными системами.

Процесс порождения начинается внутри основной программы, в случае, когда, например, необходимо ответить на вопрос пользователя; или во время беседы может возникнуть потребность прервать действия пользователя, чтобы указать надвигающуюся проблему. Как только процесс инициализирован, три вида действий должны быть выполнены:

1. Идентификация целей высказывания,

2. Планирование, как эти цели могут быть достигнуты, включая оценку ситуации и доступных коммуникативных ресурсов,

3. Реализация планов в текст.

Цели должны обычно передавать некоторую информацию аудитории или побуждать их к действиям или рассуждениям. Социальные и психологические, а также практические мотивы, побуждающие человека к общению, естественно, неприменимы для сегодняшних компьютерных программ. Планирование включает в себя отбор (преднамеренное вычеркивание) информационных модулей, которые появляются в тексте (например, концепции, отношения, индивидуальность).

Реализация зависит от знания грамматики языка и правил связности дискурса, и дает синтаксическое описание текста как промежуточное представление. При этом выделяется не только лингвистическая форма, но также знание относительно критериев, которые показывают, как используются эти формы. В многих исследованиях процесс, который проводит грамматическую реализацию, называется лингвистическим компонентом(10), а иногда планирование и вместе с процессом идентификации цели называется стратегическим компонентом (13). Обычно это - только лингвистический компонент, который имеет любое прямое знание относительно грамматики производимого языка. Какую форму эта грамматика принимает - один из самых больших различий среди проектов порождения.

Традиционно для лингвиста, грамматика - костяк в отрезке утверждения/ высказывания. Содержание утверждений - специфические факты данного естественного языка - не представляет такого интереса для лингвиста.

Аналогичная ситуация с порождением текстов, за исключением того, что запись - процедурная и декларативная - разработана, чтобы обеспечивать очень специфическую функцию, с которой традиционный лингвист не сталкивается, а именно: вести и сдерживать процесс порождения текста со специфическим содержанием и целями в присутствии специфической аудитории. Грамматика теперь ответственна за наличие выбора, который язык предоставляет для формы и словаря. Исследователи порождения должны сделать верный выбор, чтобы, используя функции различных конструкций для достижения конкретной цели. Другая функция грамматики - следить за грамматичностью текста, т. е. определение зависимостей и ограничивая решения.

Технический уровень

Разноплановое развитие и творческий потенциал в порождении текстов является возможным при следующих условиях:

1. Генератор включает в себя весь объем основной грамматики;

2. Основная программа имеет сложное, разносторонее, концептуальное представление(вид);

3. Текстовый планировщик может использовать модели аудитории и дискурса.

К сожалению, такие генераторы - все еще только предмет исследования сегодня, т. к. техническая сторона остается на уровне программы SHRDLU Винограда в 1970 (17), которая порождала предложения в процессе ответа на вопросы, система “непосредственной замены”, порождающая простые грамматические глагольные корректировки в целях достижения удобочитаемого текста.

When did you pick up [the green pyramid]?

While I was stacking up yhe red cube, a large red block, and a large green cube.

К концу 1970-ых такие системы стали достаточно популярны в работе ЭС: для перевода многочисленных правил в этих системах. Необходимость программ порождения текстов в системах с составной структурой и коммуникативным контекстом была очевидной.

Исследователи заинтересованы в более сложных текстах, нежели в контекстно-свободных представлениях, которые требуются правилами системы. В качестве примера приводится простое описание из программы Сигурда, чья цель была выяснить, как в помощью интонации выявляется группировка:

The submarine is to the south of the port. It is approaching the port, but is not close to it. The destroyer is approaching the port too.

Использование слов-ссылок “but” “too” является большим прогрессом в структурировании системы. Предложение, которое является источником в базе данных ЭС , рассуждающее о субмаринах и эсминцах, не будет обрамлено концептуальными эквивалентами таких функциональных слов, и может быть прочтено простым шаблоном, потому что ссылки специфичны и могут быть употреблены только в отдельном конкретном случае.

Еще одна техническая, пока не разрешенная, проблема - “последующая ссылка”. Какими должны быть слова-заменители, если предмет появляется больше, чем один раз в тексте? Постоянное употребление местоимений может привести к неоднозначности. В качестве примера приводится отрывок из исследований Гранвилле, который классифицирует отношения между референтом и предметом и разрабатывает правила, по которым бы могли строиться последующие ссылки.

Pogo cares for Hepzibah. Churchy likes her, too. Pogo gives a rose to her, which pleases her. She does not want Churchy’s rose. He is jealous. He punches Pogo. He gives a rose to Hebzibah. The petals drop off. This upsets her. She cries.

Неудивительно, что у исследователей, разрабатывающих основную программу, генераторы обладают наибольшей эффективностью, что дает уверенность в том, что имеется концептуальная основа для группирования отдельных предложений/ утверждений в тексте. Важным моментом на этом этапе является программа PROTEUS, разработанная Дэйви в 1974. Программа дает описание игры крестики-нолики и считается одной из программ, наиболее свободно владеющей естественным языком. PROTEUS имеет модель толкования конкретных шагов: нападение, встречное нападение, включает в себя риторический принцип, что в текст нужно помещать только наиболее существенную информацию в ситуации. Грамматика и средства реализации выбирают описанные и сгруппированные шаги, исправляют формы, так чтобы они были грамматичны в английских предложениях, и порождают собственно текст.

Следует упомянуть и программу ERMA Клиппенгера (1974)- единственная программа на тот момент, работающая со спонтанной речью. Как люди размышляют о том, что они говорят, как они динамически планируют или меняют свои намерения относительно того, что они хотят сказать в разговоре? В целях моделирования этого процесса, Клиппенгер анализировал стенограмму речи пациента по психоанализу с тем, чтобы понять рассуждения пациента, дающие объяснение одному из параграфов стенограммы, который ERMA могла подробно воспроизвести. Клиппенгер разработал структуру из пяти основных взаимосвязанных компонентов, участвующих в порождении спонтанного текста. Но для компьютерного программирования в 1974 реализовать этот план было не под силу, вследствие чего проект был оставлен.

**Исторический обзор проблемы.** По сути дела, программы PROTEUS Дэйви и ERMA Клиппенгера являются самыми старшими в этой области. Во-первых, потому что до начала 80-ых сравнительно мало людей работало над проблемой порождения , во-вторых, сама проблема достаточно сложна, по мнению авторов статьи, намного сложнее проблемы понимания речи. На самом деле, проблемой серьезно занимались в начале 1970-ых. Но справедливо отметить, что на важной конференции по данной проблеме в 1975г представленные отчеты о проделанной работе не нашли должного отклика, после чего исследования по порождению естественного языка были почти приостановлены до начала 1980-ых.

До 80-ых специалисты в области ИИ склонны были считать проблему порождения достаточно легкой. В самом деле, разве трудно взять к-л утверждение из некоторого речевого фрагмента, связать его с определениями, хранящимися отдельно, и произвести, например, следующее “The big black block supports a green one”. Это было под силу SHRDLU Винограда уже в 1970г. Если бы можно было ограничиться этими знаниями, то, на самом деле, не возникало бы проблем. Но вариативность языка не давала такой возможности. Каким образом человек представляет грамматические знания, которые позволяют генератору использовать синтаксическую структуру предложения в целях cоздания соответствующего относительного предложения (“the green block that’s supported by the big red one”, “a green one”, а не “a green block”), а также вообще иметь представление о возможности таких относительных предложений и подобных замен.

**Общие подходы к проблеме.** Трудно идентифицировать общие элементы в различных проектах исследования по порождению естественного языка. Напротив, в исследованиях по пониманию речи можно выделить несколько основных подходов к проблеме: использование расширенных сетей переходов, семантические грамматики (qv), рабочие системы, основанные на представлении концептуальной зависимости, процедурная семантика и многое другое. Исследование порождения не может дать подобной классификации, поскольку очень мало специалистов ставили эту проблему во главу угла. Большие исследовательские группы, полностью сконцентрировавшиеся на вопросе порождения естественного языка, начали создаваться в последние два года. Основная проблема состоит в отсутствии общего отправного пункта, конкретной основы для сравнения, что осложняет работу, не дает возможности для взаимопомощи между исследователями: практически невозможно проверить свои эксперименты на системе другого разработчика. Однако имеются общие нити, связывающие различные проекты: похожие подходы, похожие представления, похожие грамматики.

Существует два вопроса, представляющих общий интерес. Первый вопрос: как сопоставить многообразие форм в естественных языках, чтобы разработать их функциональное использование, ответить на вопрос, почему человек использует одну форму, а не другую, а далее формализовать этот процесс.

Второй вопрос - это контроль над процессом порождения. Что определяет выбор говорящего в данной языковой ситуации? Как человек организовывает и представляет промежуточные результаты? Какими знаниями о зависимостях между вариантами выбора должна обладать система? Как представлены эти зависимости и как они могут влиять на алгоритмы управления? Ответы на поставленные вопросы будут рассмотрены в этой статье.

**Контроль над постепенной обработкой сообщения.** Среди порождающих систем, которые были специально построены для работы в основных системах, преобладающий подход контроля состоит в обработке сообщений как определенного вида программ. Эти "сообщения" не просто выражения, чьи контекст и форма изоморфны по отношению к конечному тексту. “Сообщения” могут быть закодированы на компьютерном языке. Их нельзя просто перевести. Конечно, при самой простой обработке порождения, перевода было бы достаточно (как почти во всех существующих ЭС), но в обработке, которая сосредоточена на порождении текстов на естественном языке, отношения и содержание в сообщении лучше всего просматриваются в виде команд для достижения определенного эффекта лингвистическими средствами. Оценка происходит при постепенной обработке от внешних команд к внутренним. Эта методика контроля естественна для разработчиков систем, так как она имитирует стиль языков программирования, которые они используют.

Наиболее общие сообщения сегодня не создаются планировщиком, а являются просто структурами данных, которые извлекаются из основной программы и которым генератор дает особую интерпретацию. Подобная практика распространена в программах, которым необходимо объяснять свои рассуждения, заключенные в доказательстве дедуктивным методом исчисления предиката. Ниже приводится такого рода доказательство.

**На входе**

Line 1: premis

Exists(x) [barber(x) and

Forall(y)..shaves(x,y) iff not.shaves(y,y)l

Line 2: existential instantiation (1)

barber(g)and Forall(y)..shaves(g,y) iff not.shaves(y,y)

Line 3: conjunction reduction (2)

Forall (y)..shaves(g,y) iff not.shaves(y,y)

Line 4: universal instantiation (3)

shaves(g,g) iff not.shaves(g,g)

Line 5: tautology (4)

shaves(g,g) and not.shaves(g,g)

Line 6: conditionalization (5,1)

(Exists(x) [barber(x) and

Forall(y)..shaves (x,y) iff not.shaves(y,y)]

implies (shave(g,g) and not.shaves(g,g))

Line 7: reductio-ad-absurdum (6)

not(Exists(x) barber(x) and

Forall(y)..shaves (x,y)

iff not.shaves(y,y))

**На выходе**

Assume that there is some barber who shaves everyone who doesn’t shave himself (and no one else). Call him Giuseppe. Now, anyone who doesn’t shave himself would be shaved by Giuseppe. This would include Guiseppe himself. That is, he would shave himself, if and only if he did not shave himself, which is a conradiction. Therefore it is false, there is no such barber.

Модель дает объяснение действиям автора доказательства в выборе, какое правило применять, например, что цель правой части условия в первой строке наложить ограничение на переменную Y ("... Кто не бреет себя "). Это дает право воспринимать доказательство особым образом. Эти действия, однако, нигде в доказательстве (которое было единственным входом в программу) не появляется. Они только предполагаются и, таким образом, имеют силу только для нескольких примеров доказательств, произведенных естественным дедуктивным методом.

Недостаток информации в сообщениях основной программы - постоянная проблема в работе с порождением текстов. Специалисты по вычислительной лингвистике вынуждены вчитываться в структуры данных основных программ, потому что последние уже не включают те виды риторических команд, которые необходимы генератору, если следовать синтаксическим конструкциям языка, которые использует человек. Без “дополнительной” информации связность произносимого - особенно для длинных текстов - будет зависеть от того, насколько непротиворечиво и полно авторы основных программ представили информацию: каждый раз, когда генератор встречает к-л символ, ему ничего не остается как обрабатывать его как "посылку" или как условие одним и тем же способом, если он встречает их в одинаковом контексте. Если поддерживается непротиворечивость, проектировщик может восполнять неточности, усовершенствуя структуры данных, как только они оказываются внутри лингвистического компонента.

Средства, направленные на достижение беглости и преднамеренной детализации формы, объясняют использование фразовых словарей и промежуточного лингвистического представления. Простой пример показывает, почему это необходимо. Рассмотрим логическую формулу, которую программа обычно использовала бы внутренне. В этом примере обработка проводится тем же методом, что описан выше. Пример представляет из себя наиболее общий вид сообщения: выражение прямо из модели основной программы (система доказательства естественным дедуктивным методом), которому теперь дается особая интерпретация, так как это выражение служит для анализа текста.

(exists x

(and barber(x)

(forall y

(if-and-only-if shaves(x,y)

(not shaves(y, y) )))))

В этой формуле генератор одновременно сопоставляется с выбором реализации. Должно ли навешивание кванторов выражаться буквально ("Существует такой X, что ..."), или должно быть свернутым внутри основной части как определяющая информация относительно реализации переменных ("...some barber”)? Должно ли условие if-and-only-if реализовываться буквально как конъюнкция подчинения или может быть интерпретировано как ограничение диапазона переменной? Утверждение типа barber(x), по-видимому, всегда должно декодироваться и преобразовываться в детальное описание переменной. Остальное реализуется независимым образом, однако, после тщательного обдумывания.

Объекты, которые заполняют "мозг" основной программы, в данном случае - логические связки, предикаты, и переменные, полностью связаны со словами и грамматическими конструкциями, которые подлежат обработке "специальными процедурами/ процедурами знаний" поддерживаемыми внутри генератора. Эти процедуры - эквивалент словаря в понимающей системе. Специалисты строят фразу для понимания, используя лексическую информацию, связанную непосредственно с индивидуальными логическими объектами. Каждый объект обычно ассоциируется с к-л лексическими единицами: константа может иметь имя; предикат может иметь прилагательное или глагол. Специалист помещает их во фразовый контекст, который будет дополнен рекурсивной прикладной программой других специалистов, например, двуместный предикат "shaves(x,y)" становится шаблоном предложения "x shaves y."

Таким образом, лингвистические шаблоны обеспечивают упорядоченную реализацию параметров, что поддерживает эффективное функционирование с наименьшим количеством блокирований, ускоряя процесс порождения в целом, избегая необходимость "резервировать" преждевременные решения, которые могут оказаться несовместимыми с грамматическим контекстом, определенным более высоким шаблоном.

**Лексический Выбор.** Некоторые подходы к машинному пониманию основываются на небольшом наборе базисных элементов (qv) и, формулируют знания программы в виде набора выражений к базисным элементам, что упрощает работу программы: становится легче выводить умозаключения, потому что при помощи базисных элементов они распределяются в естественные группы. Однако, сведение диапазона человеческих действий к определенному набору, например, лишь к 13 концептуальным базисным элементам, означает, что специфика значений распределяется в выражениях и извлекается оттуда каждый раз, если во время порождения необходимо использовать глаголы со специфическим значением. Голдман первый провел исследования по использованию сетей распознавания. Он показал, как производится выбор слова, в отрыве от основных базисных элементов. Например, из базисного элемента действия "глотать" можно получить глаголы "пить", "есть", "вдыхать", "дышать", "курить", или *"проглотить"*, как бы проверяя при этом, был ли проглоченный объект жидкостью или дымом.

Проект сети распознавания заставляет исследователя порождения выходить за рамки основных различий типов объектов и включать контекстные факторы, напр., эмоциональные рассуждения говорящего. Ниже - выборка из работы Хови, цель которой состояла в том, чтобы сместить текст, чтобы подчеркнуть желаемую точку зрения (в данном случае сообщить в февральских первичных выборах так, чтобы результаты понравились Картеру, даже если он проиграл.

Kennedy only got a small number of delegates in the elections on 20 February. Cater just lost by a small number of votes. He has several delegates more than Kennedy in total.

**Фразовые словари.** Какое слово ассоциируется с простыми понятиями, типа "парикмахер" или "брить", является очевидным; однако, для объектов в комплексных основных программах, лексический выбор может оказаться более проблематичным. Помощь в этой ситуации может оказать использование фразового словаря. Это понятие было введено в 1975 Бекером и с тех пор стало важным инструментом систем порождения. С лингвистической точки зрения, "фразовый" словарь - ­концептуальное расширение стандартного словаря, включающее все непроанализированные фразы, - на той же самой семантической основе, что и словарь отдельных слов. Это обеспечивает фиксацию незаконсервированных идиом и различных речевых способов, которые люди используют каждый день. Так как люди используют эти " фиксированные фразы " как нерасчленимое целое, программы должны научиться делать то же самое. Пример ниже - из работы Кукича.

Wall Street securities markets meandered upward through most of the motning, before being pushed downhill late in the day yesterday. The stock market closed out the day with a small loss and turned in mixed showing in moderate trading.

Это информационное объявление было вычислено непосредственно из анализа данных по поведению рынка в течение дня. Качественные моменты в сообщении были соединены непосредственно со стереотипными фразами подобного рода объявлений: "a small loss", "a mixed showing", "in moderate trading". Объекты, действия и указатели времени были отображены непосредственно в соответствующих цепочках слов: "Wall Street securities markets", "meandered upward ", "be pushed downhill", "late in the day". Композиционный шаблон состоит из предложений, сформированных на основе S-V-Advp фразы: (рынок) (действие) (указатель времени).

**Обработка Грамматики**

В изучении порождения выбор формализации представления грамматики языка всегда связывался с выбором протокола контроля. Известны три основных подхода к решению этого вопроса:

1. грамматика как независимый корпус предложений и фильтр к ним (например, *объединенная функциональная грамматика*);

2. использование грамматики с целью выявления всех возможных поверхностных структур, доступных для языка; затем проведение выбора и реализации среди данных поверхностных структур (смысловые подходы);

3. грамматика как структура пересеченного графа, который контролирует весь процесс, как только создается план текста (план выражения) (грамматика расширенных сетей переходов, а также систематическая грамматика).

В этой статье не оказывается предпочтение ни одному из трех подходов. Однако каждый из них будет рассмотрен в соответствие с поставленной задачей, которая мотивирует использование этих подходов.

**Объединенная Функциональная Грамматика (ОФГ) в порождении.**

Объединенная Функциональная Грамматика была разработана Кейем, является “реверсивной” грамматикой, т. е. может использоваться как при порождении, так и при понимании речи.

Термин "функциональный", по мнению разработчиков, говорит о том, что следует оттолкнуться от описания структуры лингвистических форм, чтобы обратиться к причинам, почему используется язык. В отличие от систематических грамматик, *функциональные элементы* в ОФГ представляют к настоящему времени лишь минимальное расширение стандартного категориального лингвистического словаря, используемого традиционно, чтобы описать синтаксическую форму (например, "clause", "noun phrase", "adjective"), и имеют много общего с "лексико-функциональной грамматикой", (стоящей в той же парадигме грамматик). Классическое *функциональное* значение, типа различие между "уже имеющейся" и "новой" информацией в предложении, подобно различию между "темой" и "ремой”, еще не включено в ОФГ. ОФГ использует “telegram” грамматику, разработанную Аппельтом, понимающий компонент, написанный Босси.

Первый пример (из Аппельта) описывает одну из составляющих ролей, которые сопровождают фразовую категорию, именную фразу.

ОФГ используют, чтобы изложить в деталях минимальные, концептуально полученные функциональные описания, например, что главным словом к-л именной фразы должно быть слово "отвертка". Недавняя работа Паттена использует систематическую грамматику в очень схожим образом. Операции такого типа на семантическом уровне, выполняемые в других подходах путем планирования уровня, специалисты определяют как набор особенностей *вывода* внутри систематической грамматики, эквивалент начального функционального описания, которое управляет ОФГ. Обратное и прямое формирование цепочки перемещается через систематическую грамматику, затем определяет, какие дополнительные лингвистические особенности должны быть добавлены к грамматической спецификации текста.

ОФГ используются в процессе последовательных объединений, ограниченных правилами, которые следят за тем, как два описания могут быть объединены. Ключевая идея состоит в том, что планировщик первоначально создает минимальное описание фразы, что можно делать и стандартным способом. Чтобы излагать в деталях описание к пунктам, где это было бы грамматически верно, оно затем объединяется с грамматикой: описание фразы и спецификация грамматики успешно объединены. Конкретизация понятий прежде не определенных особенностей описания константами, снабженными грамматикой, вызывает эффект ряби во всей системе: решения, которые зависят от только что конкретизированных особенностей, провоцируют дальнейшее циклическое объединение, пока не будет сформулировано грамматически полное описание высказывания. Кроме того, элементы в описании планировщика побуждают к отбору среди дизъюнктивных спецификаций в грамматике. Например, определение глагола приводит к выбору грамматической подклассификации.

Полное описание составляет дерево подописаний (составляющих) как определено "стандартом" (образцом), который предписывает последовательный порядок на каждом уровне. Фактически текст создается при просмотре этого дерева и чтении слов с лексическими особенностями каждой составляющей. Ограничения накладываются в процессе объединения: только совместимые частичные описания присутствуют в конечном результате. Это имеет большое значение, так как планировщику не нужно разбираться с грамматическими ограничениями и зависимостями, что, с другой стороны ограничивает его потенциал: он не может пользоваться знаниями по грамматическим ограничениям, даже когда ему это понадобится.

С точки зрения разработки грамматики, ОФГ является вполне удовлетворительной, так как данный подход позволяет компактно формулировать языковые факты, то есть необязательно расшифровывать взаимосвязь между предложениями, так как это происходит автоматически во время объединения.

**Прямой Контроль Грамматики при Понимании: Систематическая Грамматика и Грамматика Расширенных Сетей Переходов (РСП).** Расширенная сеть переходов используется в порождении почти с момента своего определения. РСП использовали сначала Симмонс и Слокум в 1970, чью систему затем использовал Голдман. РСП также применял Шапиро, чей генератор, в этой группе, является наиболее продуманным. Все системы имеют схожую структуру. Они просматривают структуру данных, которую поддерживает основная программа. Сети поддерживают формат сверху-вниз, как обычно у всех РСП-парсеров (синтаксических анализаторов). Для ранних РСП подобная структура являлась семантической сетью, основанной на теории фреймов с глаголом в центральной части (еще одна "функциональная" лингвистическая система). Специальный узел в сети, "вектор модальности", определяет информацию на корневом уровне, например, время и вид; является предложение активным или пассивным. Первичная функция РСП в ранних системах состояла в линейном упорядочении сетевой структуры, которая была главным образом уже закодирована в лингвистическом словаре.

РСП, по существу, представляет из себя процедурное кодирование порождающей грамматики. Регистры, которые дают сетям "расширенное" влияние, используются как представление грамматических отношений с глубинной структурой, и пути в сетях кодируют все составные поверхностные альтернативные последовательности. Ограничения распространяются по дереву сверху-вниз (то есть к рекурсивным подсетям РСП) через значения в обозначенных регистров, приводя в действие подсети при контекстном управлении. Проект РСП Шапиро особенно впечатляет, поскольку его структура управления данных занимает весь вычислительный режим основной программы.

Дальнейший аспект проекта РСП - тот факт, что средства создания слов текста являются выполнением побочного эффекта по прохождению ребра графа, что приводит генератор к действию почти в тот момент, когда ситуация воспринимается. Особенно впечатляет то, что оценивает, что РСП Шапиро никогда не пользуется резервированием. Это - совершенно необычное поведение для РСП, так как порождение является в сущности процессом планирования.

Наиболее значительной проблемой для проектов РСП - ­трудность выделения понимания из действия. Генераторы, основанные на систематической грамматике, имеют дело с этой проблемой, непосредственно представляя срединную репрезентацию в форме набора характерных признаков, что позволяет спецификации текста постепенно накапливаться, предоставляя ограничениям возможность распространяться и влиять на более поздние решения.

Две важных системы порождения были основаны на систематической грамматике: PROTEUS Дэйви(обсуждали ранее) и NIGEL Манна и Маттхиссена. NIGEL - самая большая систематическая грамматика в мире и, очень вероятно, одна из самых больших машинных грамматик любого сорта.