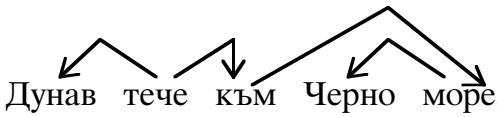
Синтаксис

I. Граматики на зависимостите (dependency grammars)

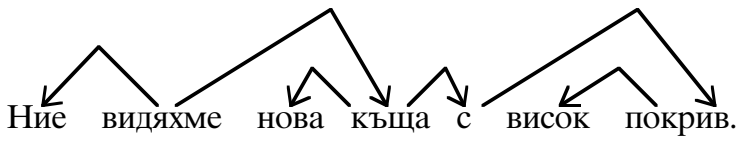
За първи път терминът е споменат от френския лингвист L. Tesnier през 1959. Стават основен инструмент за моделиране на синтаксиса на славянски езици (руски, чешки) и се прилагат за немския в Института за немски език в Източен Берлин (ГДР). Днес повечето системи за машинен превод, особено създаваните в Япония, използват този вид граматики при описание на синтактичните структури. Въпреки че липсват редица привлекателни страни на конституентните граматики (например алгоритмите за анализ на безконтекстни граматики), изглежда за редица ЕЕ е по-удобно синтактичните данни да се описват чрез граматики на зависимостите. Да видим защо.

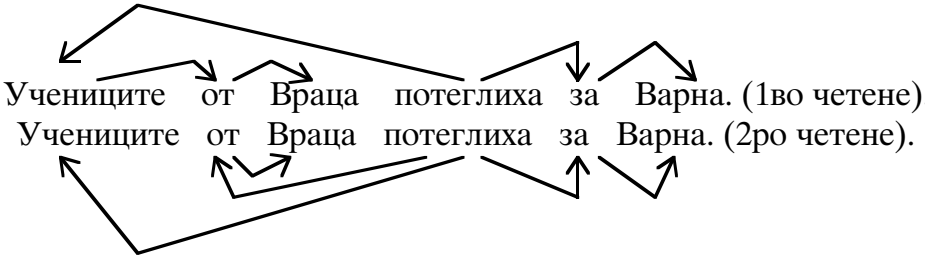
Синтактичната структура на изречението се представя чрез дърво, като върховете са думите от изречението. Дъгите съответстват на отношенията на синтактична подчиненост между думите, например:



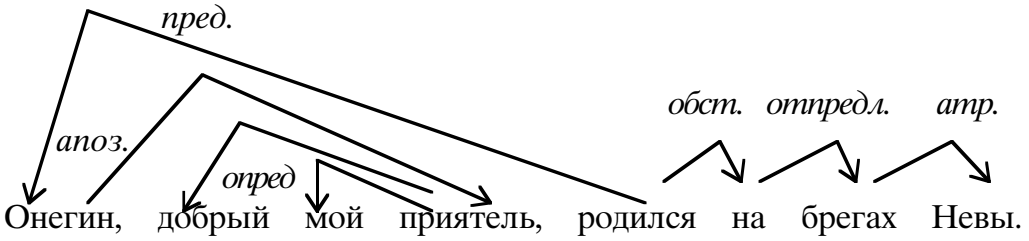
Структурата е йерархична, с корен сказуемо. *Целта е на всяка дума да се намери точно една друга дума (или група думи), която я "подчинява".* Този модел ни напомня по простотата си за училищната граматика, с тази разлика, че тук стигаме до ориентиран граф (дърво) и има лингвистична теория за обяснение на насочеността на отношението "синтактична зависимост". Фактически, когато във върха на дървото стои сказуемото, при просто сказуемо от един глагол веднага се вижда, че то подчинява "актантите" на глагола, т.е. тези части на изречението, които "запълват валентностите на глагола" (валентност /valency/ се използва в лингвистиката по аналогия с химията - това са тези позиции, в които например един глагол трябва или може да се допълни в изречението с подлог, пряко или непряко допълнения, обстоятелства и т.н.; актанти са участниците, запълващи валентните роли на глагола). Ако сказуемото е сложно, напр. "момчета и момичета пеят и танцуват", са нужни конвенции за трактуване на сложни сказуемни групи (но и конституентните граматики имат такива проблеми).

Да разгледаме още няколко примера:

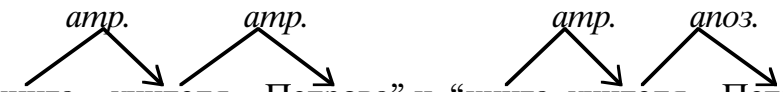




Внимателно разглеждане на дърветата ни показва, че те изразяват не само синтактично подчинение, а и семантично. Следователно, ако се научим да поставяме маркери (етикети) по дъгите на дървото (което е възможно, ако в лексикона си имаме етикети на всички възможни потенциални дъги-валентности на отделните лексеми), тогава бихме могли да получаваме дървета на зависимостите с маркери: напр.,



Имаме следните маркери на отношения:
пред(икативно): между сказуемо и подлог;
апоз(итивно): между съществително и негово приложение;
опред(елително): м-у съществително и негово съгласувано определение;
атр(ибутивно): между съществително и негово несъгласувано определение;
обст(оятелствено): между глагол и негово обстоятелство;
отпредл(ожно): между предлог и управлявано от него съществително;
агентивно: между сказуемо и име на деятеля (на български със страдате-лен залог, а на руски с творителен падеж /“дом построен рабочими”/);
първо_обектно: между сказуемо и пряко допълнение;
второ_обектно: между сказуемо и непряко допълнение;
и др.



Получаваме: “книга учителя Петрова” и “книга учителя Петрова”, т.е. “книгата на учителя на Петров” и “книгата на учителя Петров”. Така заедно с построяването на синтактичната структура сме направили и семантичен разбор. (Това не е възможно при конституентните граматики). Въвеждат се и синтактически групи, например:
“Утре тук ще играят деца” аналогично на “Днес тук играят деца”.

Да навлезем в математическия апарат, който се прилага за моделиране на “хубави” изречения. От всички възможни дървета на зависимостите отделяме подклас на “естествените” дървета, които съответстват на “хубави” изречения. Това става чрез понятието “проективност”:

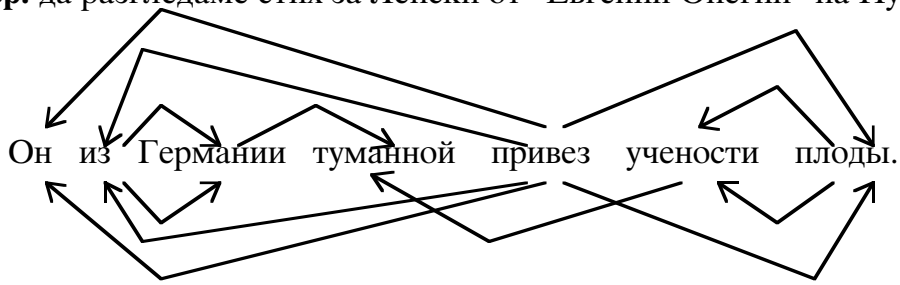
Определение. Едно дърво на зависимостите е проективно, ако за всеки три негови възела α, β, γ от

$\alpha \rightarrow \beta$ (α подчинява β) и

γ като дума в изречението на дървото лежи между α и β

следва, че $\alpha \rightarrow \gamma$ (α подчинява γ).

Пример: да разгледаме стих за Ленски от “Евгений Онегин” на Пушкин:



Първото четене (горното дърво), съответства на смисъла “Он привез из туманной Германии плоды учености”. Но Германия, за разлика от Англия, не се счита за мъглива страна, а пък учеността на Ленски може да се разглежда като мъглива. Поради особеностите на поетическата реч е възможно още едно четене на руски (долното дърво), а именно: “Он привез из Германии плоды туманной учености”. Счита се, че Пушкин е целял втория смисъл, но “психолингвистичната” трудност за възприемане на такъв прочит произтича от факта, че горното дърво е проективно, а долното - не.

Преди да преминем към конституентните граматики, да резюмираме положителните и отрицателни качества на граматиките на зависимостите:

+ Ако имаме

а) добър лексикон (а речници и граматики за подготовка на такава информация има за всеки език, по-добри или по-лоши),

б) добра морфология, която със сведена до минимум многозначност разпознава основните лексеми на всички форми,

то виждаме, че анализ с граматика на зависимостите не е толкова трудно постижим, поне за прости изречения. Особено при език с падежи (ако си мислим за руски или немски), където падежът е сам по себе си знак за запълване на определена валентност на разпознатия глагол - започваме веднага да си представяме реализацията на синтактико-семантичен анализ от описания по-горе тип.

+ Данните в граматиката на зависимостите се описват чрез много по-близки и понятни абстракции, ако ги сравним с умозрителните граматически категории, които при конституентните граматики обслужват само целите на записване на правила и анализ;

– Липсват стандартните процедури за анализ, не са развити истински унификационни формализми за граматики на зависимостите (понеже не се създаде стандарт как да изглеждат данните за такива граматика).

Известните софтуерни среди за граматическа информация са разработвани в САЩ и Зап. Европа и изцяло прилагат различни диалекти на конституентните граматики (вж. по-долу).

Литература: Гладкий, А.В. *Синтаксические структуры естественного языка в автоматизированных системах общения*. Серия "Проблемы исскуственного интеллекта", Москва, "Наука", 1985.

II. Конституентни граматики (граматики на съставящите)

(Constituent grammars) При този вид синтактичен анализ, във върховете на получената синтактична структура стоят граматически категории (а думите на изречението са листа). *Така изречението се обяснява като йерархия от съставящи го синтактични конституенти.*

Американски структурализъм в лингвистиката (напр. Блумфийлд, 1933). Ноъм Чомски (Noam Chomsky) въвежда понятието "**безконтекстна граматика**" през 1956 година, в контектса на дефинираната от него йерархия на формални езици, разпознавани от пораждащи граматики (вж. примера *Времето лети като стрела* в лекция 1). Днес конституентните граматики в различни техни диалекти са почти единствения формализъм, който се използва в учебниците по КЛ за описание на допустимите ЕЕ структури; с тях се свързва и понятието **граматически разбор** (parsing) - това е метод за анализ на изречението с цел определяне на неговата структура съгласно дадена граматика. Структурата се изразява чрез **дърво на разбора** (parsing tree). Ще използваме известните ни *пораждащи граматики*, процедурите за анализ *top-down* и *bottom-up* и граматическите категории, въведени в първите две лекции. Целта ни е да разгледаме най-общо основните постановки на проблемите и най-известните решения.

1. Chart Parsing

Главната разлика между top-down и bottom-up анализа е начинът, по който се избира правило за опитване. Нека имаме безконтекстна граматика (БКГ):

1. S —> NP VP
2. NP —> ART ADJ N
3. NP —> ART N
4. NP —> ADJ N
5. VP —> AUX VP
6. VP —> V NP

Да вземем например правило 2: в top-down анализ това правило ще се използва за заместване на NP-листо в текущо построеното дърво и така това NP ще се замести с три наследника ART, ADJ, N, което обаче може да се окаже ненужно, ако няма подходящи думи в анализираното изречение. При bottom-up анализ това правило ще бъде опитано, когато в изречението стои дума с категория ART. Лесно се вижда, че пълното претърсване на всички възможности е много неефективно, и затова се

въвежда структура от данни, наречена таблица или чертеж (**chart**), като идеята е да се запомнят междинните резултати и така да се редуцират част от повтарящите се опити. Ще разгледаме първо един алгоритъм за bottom-up chart parsing.

Правило за прилагане се търси винаги от гледна точка на една конституента-ключ (key). За да намерим правило, подходящо за прилагане над низ съдържащ ключа, търсим

- или правила започващи с ключа като първа конституента отдясно на стрелката,

- или правила, които са били прилагани над предишни ключове и съдържат настоящия ключ за завършване или разширения на дясната си част от нетерминали.

Например, да анализираме изречение, започващо с дума с категория ART. Подходящи са правила 2 и 3, но ние искаме да запишем, че след тяхното прилагане ART е вече анализирана категория. Слагаме знак след нея и преписваме правила 2 и 3 като:

2'. NP —> ART ° ADJ N

3'. NP —> ART ° N

Ако следващата дума в изречението е ADJ, можем да приложим правило 4, но тогава ще препишем 2' като

2". NP —> ART ADJ ° N

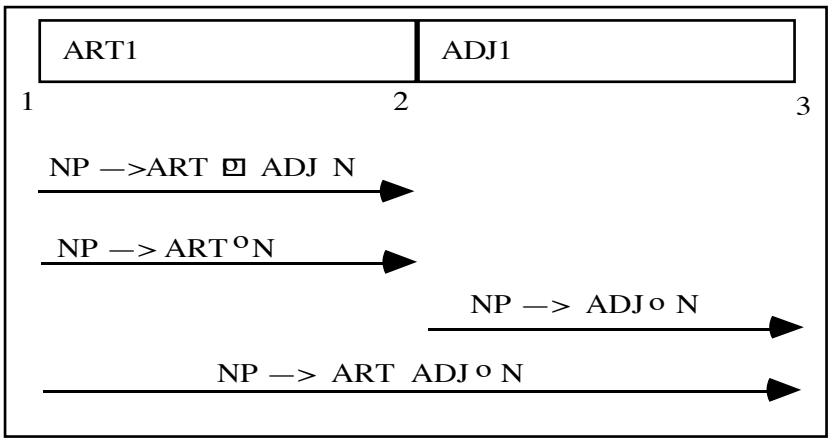
за да отразим факта, че то е все още актуално за прилагане, но "от дадено място нататък" на десния низ от нетерминали.

На всяка стъпка от анализа се поддържа

- запис на всички конституенти, породени досега, както и

- запис на всички правила, които са частично подходящи, но изборът все още не е приключил (наричат се active arcs).

Фигура 1 показва чертежа за стъпката ART+ADJ:



Фигура 1 се интерпретира както следва: има две разпознати (породени) конституети, ART1 на позиция 1-2 и ADJ1 на позиция 2-3. Има 4 активни дъги, които изискват различни продължения: например 1-вата изисква N

в позиция 3, 2-рата изисква N в позиция 2, и все още не сме сигурни че това е невъзможно. Има и 3та, и 4та активни дъги и т.н.

Основната операция е комбиниране на активни дъги с окомплектовани конституенти. Резултатът е или нова окомплектована конституента, или нова активна дъга, която е разширение на оригиналната активна дъга. Новите конституенти се записват в т. нар. agenda (дневен ред), докато дойде момент да се прехвърлят в чертежа. Процесът се дефинира чрез:

Алгоритъм 1 за разширяване на дъгите: за да прибавим конституента C от позиция p_1 към p_2:
1. Включваме C в чертежа от позиция p_1 към p_2.
2. За всяка активна дъга от вида X—> X_1 ... ° C ... X_n от позиция p_0 към p_1, включваме нова активна дъга от вида X—> X_1 ... C ° ... X_n от позиция p_0 към p_2.
3. За всяка активна дъга от вида X—> X_1 ... X_n° C от позиция p_0 към p_1, включваме нова конституента от вида X в "дневния ред" от p_0 към p_2.

Алгоритъм 2 за bottom-up chart parsing: Докато входното изречение се анализира до последната си дума:
1. Ако "дневният ред" е празен, търси интерпретациите на следващата дума в изречението и ги добави към "дневния ред".
2. Избери конституента от "дневния ред" (да я наречем C от позиция p_1 към p_2).
3. За всяко правило в граматиката от вида X—> C X_1 ... X_n, прибави активна дъга от вида X—> ° C X_1 ... X_n от позиция p_1 към p_2.
4. Прибави C към чертежа по алгоритъм 1.

Ще предполагаме стратегия depth-first и ще проследим детайлно анализа на изречението "*The large can can hold the water*" с лексикона *the* - ART; *large* - ADJ; *can* - N, AUX, V; *hold* - N, V; *water* - N, V.

В началото дневният ред е празен, чете се "the" и ART се записва в дневния ред. Имаме:

Включване на ART1 (the от позиция 1 към 2):
Добави активна дъга NP—> ART ° ADJ N от 1 към 2
Добави активна дъга NP—> ART ° N от 1 към 2

Дъгите са добавени по стъпка 3 и 4 на алгоритъм 2, съгласно правила 2 и 3 на граматиката. Чете се думата large и се създава конституента ADJ1.

Включване на ADJ1 (large от позиция 2 към 3):
Добави активна дъга NP—> ADJ ° N от 2 към 3
Добави активна дъга NP—> ART ADJ ° N от 1 към 3

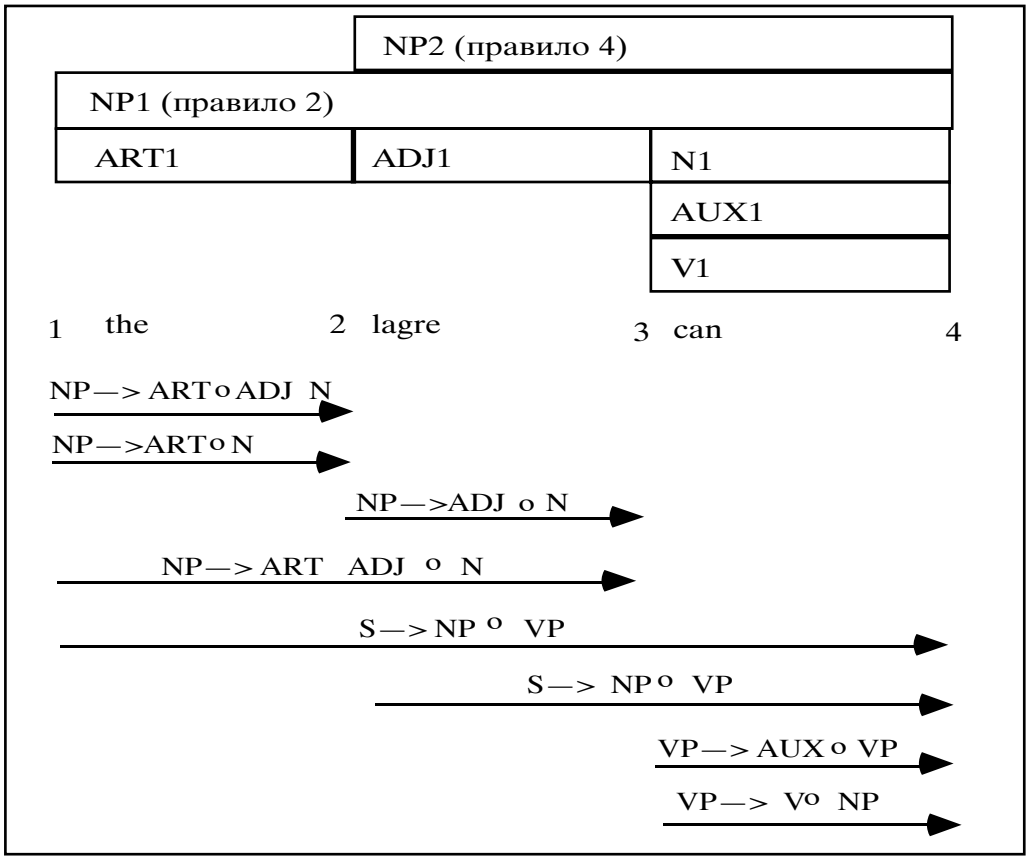
Първата дъга се добавя по стъпка 3 на алгоритъм 2. Втората дъга е разши-рение по стъпка 2 на алгоритъм 1 на първата активна дъга за ART1. Чертежът в този момент е показан на фиг. 1. Активните дъги не се изтриват от чертежа, например при разширение на дъга стоят и старата,

и новата версия. Това се налага поради възможността стара дъга да се използва отново при друга интерпретация.

При следващата дума *can* се създават три конституенти: N1, AUX1 и V1. Стъпка 2 на алгоритъм 1 не внася нови активни дъги, но две дъги се окомплектоват според стъпка 4 на алгоритъм 1: едната, по правило 2, е NP (от ART, ADJ и N) от позиция 1 до позиция 4; другата, NP от 2 до 4, се конструира по правило 4 (ADJ, N). Сега тези две NP (NP1 и NP2) са на върха на дневния ред, над AUX1 и V1. Почваме да ги вкарваме в чертежа едно след друго.

Включване на NP1 - едно NP (the large can - от 1 до 4):
Добави активна дъга S—> NP ° VP от 1 до 4
Включване на NP2 - едно NP (large can - от 2 до 4):
Добави активна дъга S—> NP ° VP от 2 до 4
Включване на AUX1 (can - от 3 до 4):
Добави активна дъга VP—> AUX ° VP от 3 до 4
Включване на V1 (can - от 3 до 4):
Добави активна дъга VP—> V ° NP от 3 до 4

Фигура 2 по-долу показва състоянието на чертежа в този момент:



Фигура 2: чертежът след анализа на “the lagre can”. Разпознати са конституентите NP2, NP1, ART1, ADJ1, N1, AUX1, V1 и са показани незавършените активни дъги. Забележете, че всяка нова конституента се въвежда със свой собствен индекс и така различаваме NP1, NP2, ...

При повторното срещане на следващата дума *can* се създават отново три конституенти: N2, AUX2 и V2:

Включване на NP2 (2ро can - от 4 до 5):

Не се добавят активни дъги

Включване на AUX2 (2ро can - от 4 до 5):

Добави активна дъга VP—> AUX ° VP от 4 до 5

Включване на V2 (2ро can - от 4 до 5):

Добави активна дъга VP—> V ° NP от 4 до 5

Прочита се *hold*, и се добавят N3 и V3.

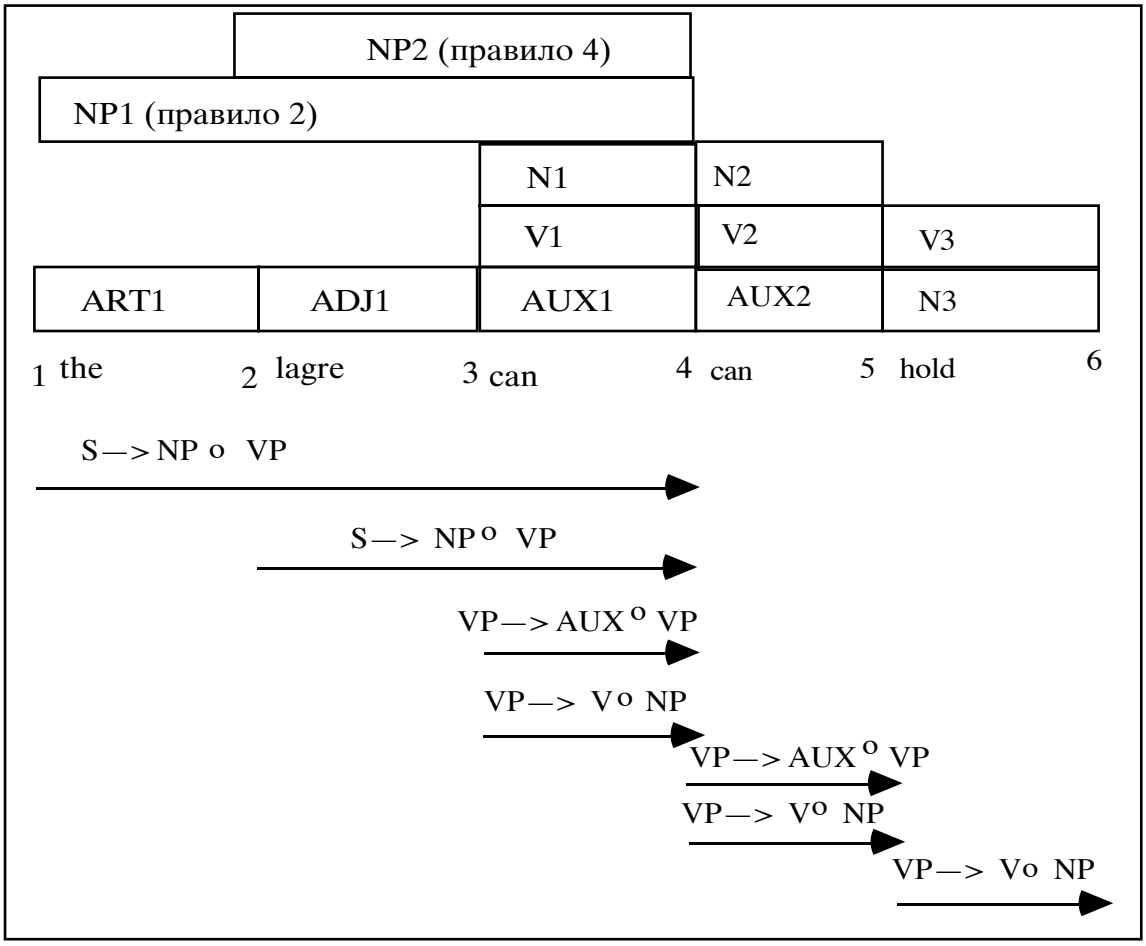
Включване на N3 (hold - от 5 до 6):

Не се добавят активни дъги

Включване на V3 (hold - от 5 до 6):

Добави активна дъга VP—> V ° NP от 5 до 6

Фигура 3 по-долу показва състоянието на чертежа в този момент, като са пропуснати дъгите, създадени за първото NP:



Фигура 3.

Прочита се ART2 *the*, от 6 до 7.

Добави активна дъга NP—> ART ° ADJ N от 6 до 7

Добави активна дъга NP—> ART ° N от 6 до 7

Прочита се N4 *water*, от 7 до 8.

Не се добавят активни дъги

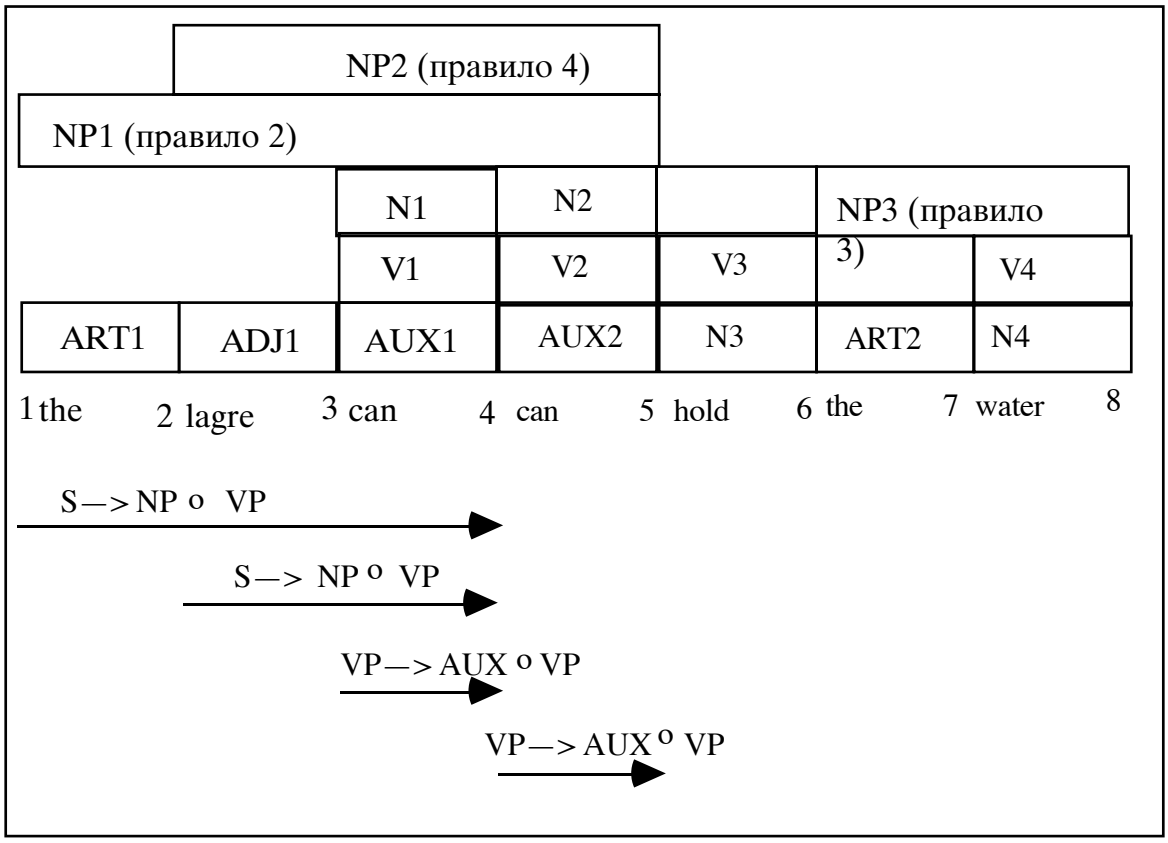
Създава се ново NP, NP3, от 6 до 8 чрез окомплектоване на активната дъга NP—>ART ° N от 6 до 7

Включване на NP3 (the water - от 6 до 8):

Създава се ново VP, VP1, от 5 до 8 чрез окомплектоване на активната дъга VP—>V ° NP от 5 до 8

Добави активна дъга S—> NP ° VP от 6 до 8

Фигура 4 по-долу показва състоянието на чертежа в този момент, като са показани само дъгите, които ще се използват до края на анализа:



Фигура 4.

Включване на VP1 (hold the water - от 5 до 8):

В дневния ред се създава ново VP, VP2, от 4 до 8 чрез завърш-

ване на активната дъга VP—>AUX ° VP от 4 до 8

Включване на VP2 (can hold the water - от 4 до 8):

В дневния ред се създава ново S, S1, от 1 до 8 чрез завърш-

ване на активната дъга S—>NP ° VP от 1 до 8

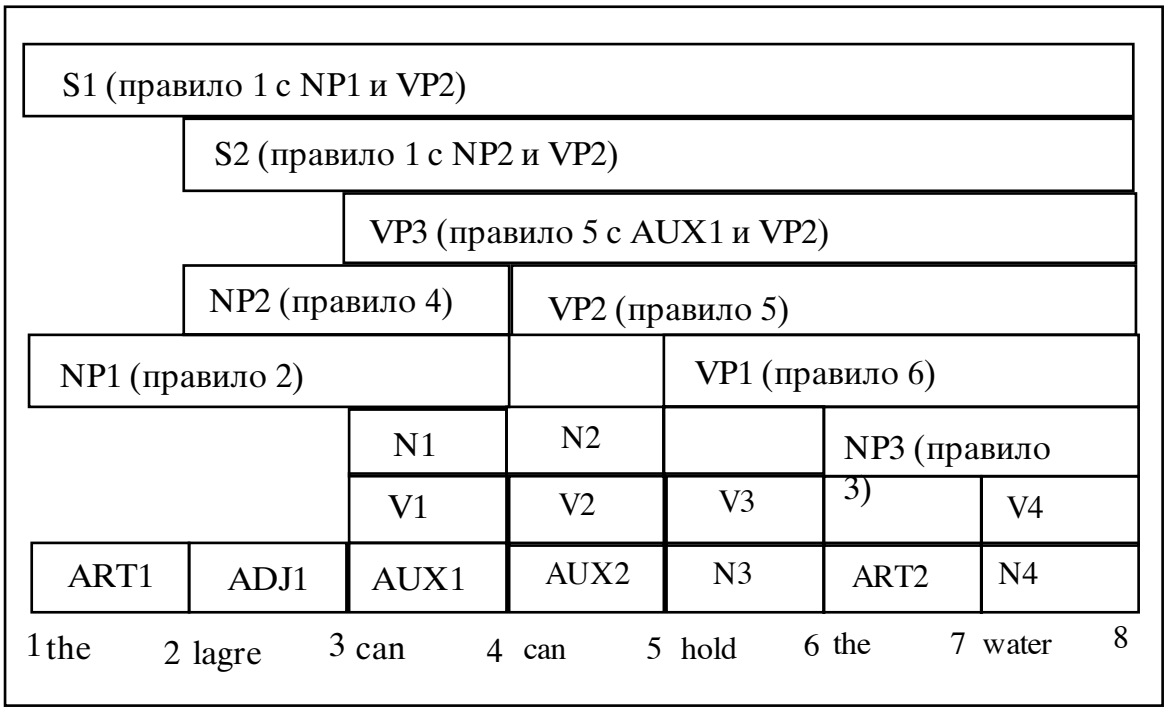
Създава се ново VP, VP3, от 3 до 8 чрез завърш-

ване на активната дъга VP—>AUX ° VP от 3 до 4

Създава се ново S, S2, от 2 до 8 чрез завърш-

ване на активната дъга S—>NP ° VP от 2 до 4

Фигура 5 е финалът на анализа, след получаване на първото S, което покрива цялото входно изречение:



Фигура 5.

Понеже вече сме получили едно S, със сигурност анализът е успешен. Можем да продължим до края на "дневния ред", и така ще се получат всички възможни S - които съответстват на различните разбори. Този вид анализ е по-ефикасен от пълното претърсване, понеже една и съща конституента не се конструира повече от един път.

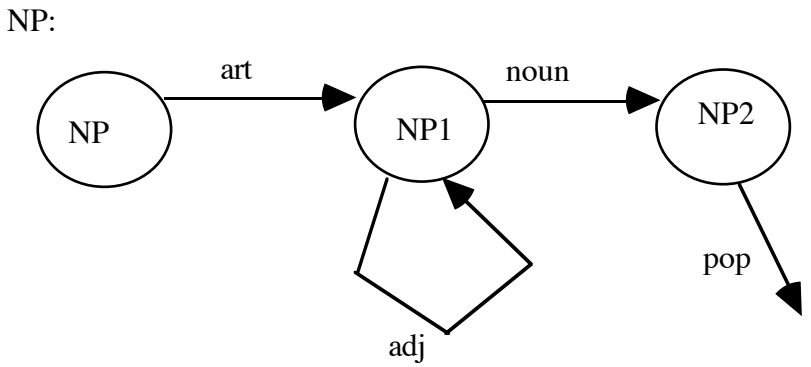
Пълното претърсване top-down или bottom-up става с C^n операции, където C е константа, зависеща от метода, и n е дължината на изречението в думи. При chart-parsing е $K.n^3$, където K е константа зависеща от метода, и очевидно K>C. Нека си представим разликата в порядъка на двете оценки: ако C е 10, а K - 1000 (100 пъти по-бавно), то за n=12 имаме: C^n=1 000 000 000 000, а $K.n^3$=1 728 000 операции. При това предположение за константите, което е съвсем реално, анализът с чертеж е до 500 000 пъти по-ефективен.

III. Граматики с мрежи на преходите (Transtion Network Grammars)

Досега сме говорили само за класическите БКГ. Сега се спираме на

III.1. Transition networks

Мрежа от върхове (nodes) и наименовани дъги (labeled arcs). Един връх е определен за начално състояние (initial state или start state). Например мрежата по-долу на Фиг. 6



Фиг.6.

съответства на граматиката

NP—> art NP1
NP1—> adj NP1
NP1—> N

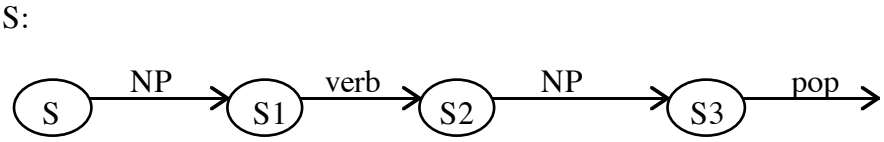
Мрежата се използва за анализ както следва: започваме от началното състояние и преминаваме към следващо по тази дъга, която е наименована с категория съвпадаща с категорията на текущата дума в анализираното изречение. Когато се премине по една дъга, текущата дума се обновява със следващата. За мрежата от Фиг. 6, една фраза е правилно NP, ако има път от началното състояние до стрелка, наименована pop. Така “a purple cow” е правилно NP, понеже успяваме да го анализираме по мрежата (този анализ също се нарича parsing). Простите мрежи на преходите са еквивалентни на крайните автомати (finite state machines, FSM) и пораждат регулярните езици. Тука ще се спрем по-подробно на рекурсивните мрежи на преходите.

III.2. Мрежи с Рекурсивни Преходи МРП (recursive transition network, RTN)

Дефинират се като разширение на обикновените мрежи: *маркираните дъги могат да сочат към други мрежи или към граматически категории*. МРП имат изразителната сила на БКГ, както (лесно) ще видим по-късно.

Ще използваме NP-мрежата от Фиг. 6 по-горе и нейната граматика; забележете, че граматиката еквивалентна на мрежата от Фиг. 6 е тип 3 (обикновените мрежи разпознават и генерират регулярните езици).

Можем да съставим МРП за прости английски изречения както следва на фиг. 7. Етикетите с главни букви сочат към мрежи; така ще се премине по дъгата от S към S1 само когато NP-мрежата от Фиг.6 е успешно обходена от началото до някоя нейна pop-дъга. Възможно е да се дефинира истинска рекурсия в RTN: т.е., в мрежата да имаме дъга с етикет самата нея (но такъв случай няма на фиг.7).



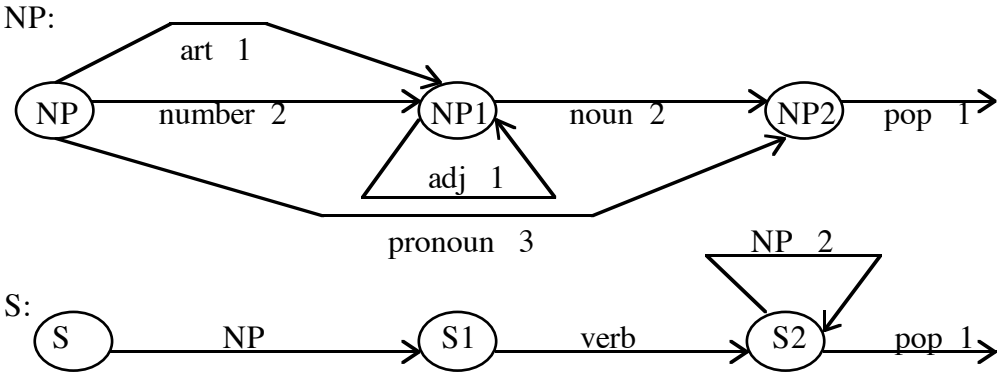
Фигура 7. S-мрежа, "цитираща" NP-мрежата от фиг. 6

Нека имаме изречението "The purple cow ate the grass". След началния връх S влизаме в дъга NP, където по мрежата от фиг. 6 трасираме успешно the purple cow. Дъгата pop на NP мрежата ни връща към върха S1, откъдето с вход ate по дъгата verb достигаме S2. Отново по NP-мрежата от фиг. 6 трасираме the grass до върха S3. Следва pop-дъга при празен вход, следователно анализът е успешен и изречението е допустимо.

Долната таблица резюмира видовете дъги, допустими в МРП:

Вид дъга	Пример	Начин на използване
CAT	noun	успява при текуща дума с категория noun
WRD	of	успява при текуща дума, идентична с етикета
PUSH	NP	успява при успешно трасиране на мрежа с начало (или име) NP
JUMP	jump	винаги успява
POP	pop	успява и показва успешен край на мрежата

Друга примерна мрежа, която ще бъде разгледана в детайли по-нататък:



Фиг. 8. Рекурсивна мрежа на преходите, кодираща синтактично знание

Анализ отгоре-надолу с МРП

Аналогичен на анализ с БКГ (ние вече разгледахме анализ отдолу-нагоре с чертеж). Състоянието на анализа във всеки момент ще бъде представяно чрез променливите:

текуща_позиция - указател към следващата дума за анализ;

текущ_връх - връх в мрежата, където се намираме;

точки_за_възврат - стек от върхове в други мрежи, където ще се премине, ако в текущия връх има **pop**-дъга.

Ще разгледаме алгоритъм за анализ, при който имаме 4 случая за напускане на текущия връх по излизаща от него дъга:

Случай 1: При CAT-дъга и следваща входна дума с тази категория:

(1) премести **текуща_позиция** на следващата дума;

(2) обнови т**екущ_връх** с върха, към който е насочена CAT-дъгата.

Случай 2: При PUSH-дъга към мрежа X:

(1) добави към **точки_за_възврат** върха, в който води PUSH-дъгата;

(2) обнови т**екущ_връх** с началния връх на мрежата X.

Случай 3: При POP-дъга и непразен стек **точки_за_възврат**:

(1) изтрий най-горния елемент на **точки_за_възврат** и го запиши в т**екущ_връх**.

Случай 4: При POP-дъга, празен стек **точки_за_възврат** и край на входното изречение:

(1) успешен анализ.

Алгоритъмът ще бъде илюстриран чрез мрежата - тоест граматиката - на фиг. 8. Числата показват коя дъга трябва да се избере първа, при наличие на много избори. Долната таблица показва как граматиката на фиг. 8 анализира изречението (1) The (2) old (3) man (4) cried (5). Показана е редицата на състояния на анализатора (trace), като всяка дъга се идентифицира с върха, който напуска, и нейния номер на фиг. 8. Дадени са само изборите, които водят към успех:

Стъпка	Текущ връх	Текуща позиция	Точки възврат	По коя дъга	Коментар
1.	(S,	1,	NIL)	S/1	начална позиция
2.	(NP,	1,	(S1))	NP/1	по PUSH-дъга към NP-мрежата, възврат в S1
3.	(NP1,	2,	(S1))	NP/1	дъга NP/1 с “the”
4.	(NP1,	3,	(S1))	NP1/1	дъга NP1/1 с “old”
5.	(NP2,	4,	(S1))	NP1/2	последвана дъга NP1/2 (NP1/1 е невъзможна)
6.	(S1,	4,	NIL)	S1/1	pop-дъга ни води обратно към S1
7.	(S2,	5,	NIL)	S2/1	последвана S2/1
8.					успех

Този пример (The old man cried) успява, понеже на всяка стъпка се избира необходимата дъга. За други изречения обаче е нужно да се прави backtracking, като се връщаме към стари "backup" състояния и избираме нови дъги. Така се налага разширение на възможните стъпки с оглед осигуряване на възврат. По-долу е дадена таблица с анализа на "(1) one (2) saw (3) the (4) man (5)". В началото се прави опит да се разпознае "one saw" като NP, след неуспеха само "one" е анализирано като NP:

Стъпка	Текущо състояние	Дъга за следване	Backup - състояния
1.	(S,1,NIL)	S/1	NIL
2.	(NP,1,(S1))	NP/2 (& NP/3 за backup)	(NP,1,(S1)) с NP/3
3.	(NP1,2,(S1))	NP1/2	(NP,1,(S1)) с NP/3
4.	(NP2,3,(S1))	NP2/1	(NP,1,(S1)) с NP/3
5.	(S1,3,NIL)	няма дъга	(NP,1,(S1)) с NP/3
6.	(NP,1,(S1))	NP/3	NIL
7.	(NP2,2,(S1))	NP2/1	NIL
8.	(S1,2,NIL)	S1/1	NIL
9.	(S2,3,NIL)	S2/2	NIL
10.	(NP,3,(S2))	NP/1	NIL
11.	(NP1,4,(S2))	NP1/2	NIL
12.	(NP2,5,(S2))	NP2/1	NIL
13.	(S2,5,NIL)	S2/1	NIL
14.	успех		NIL

Опитайте се да докажете сами, че МРП са еквивалентни на БКГ. (Към тази лекция има приложени задачи с основната идея, че по дадена МРП се строят правила с по една променлива отляво на стрелката).

До този момент в тази лекция за синтаксис сме се фокусирали върху БКГ и алгоритмите за анализ отдолу-нагоре и отгоре-надолу. Забележете, че графическото представяне (т.е. записването на граматиката като мрежа) не променя същината на обработката, а само визуализира по удобен (за нас) начин граматическите правила.

IV. Признакови структури и системи (Feature Systems, Feature Structures) и Разширени граматики (Augmented Grammars)

БКГ са основа на (десетина) различни формализма за все по-пълно и удобно представяне на лингвистична информация. (Не бива да си мислим, че КЛ днес прилага БКГ в чист вид). Почваме с едно разширение на БКГ, при което граматическите категории са множество от **признаци** (features). Така се постига управление например на:

- съгласуването по род и число между подлог и сказуемо;

- съгласуване между глагола и неговите допълнения в момента, *когато глаголът си "взима" допълненията* си (*subcategorization*).

Лесно се вижда, че БКГ са твърде тромави за описание на ЕЕ. Нека имаме например правило NP —> ART N. Всъщност това правило трябва да е приложимо, когато числото на ART (напр. единствено) съвпада с числото на N (за да не се позволява например "a trees"). Значи или ще "размножим" променливите и правилата (не особено елегантно решение):

NP-SING —> ART-SING N-SING
NP-PL —> ART-PL N-PL,

или ще разглеждаме категориите като набор от стойности. Така NP ще има признака (**feature**) NUMBER със стойности (**values**) или *s* (единствено), или *p* (множествено). Почваме да използваме в лексикона си запис от рода на:

ART1: (**CAT** ART
ROOT a
NUMBER s)

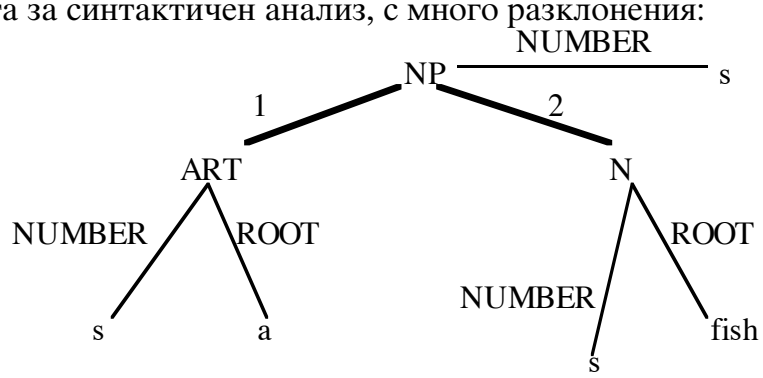
или по-простото

ART1: (ART **ROOT** a **NUMBER** s)

Позволяваме вграждането на feature-структури в други:

NP1: (NP **NUMBER** s
1 (ART **ROOT** a
NUMBER s) (1)
2 (N **ROOT** fish
NUMBER s))

и тогава feature-структурите вече изглеждат като "спагето-подобни" дървета за синтактичен анализ, с много разклонения:



Граматиките, в които граматическите категории се задават чрез *множество от стойности,* се наричат **разширени** (augmented). Така броят на правилата се пази сравнително малък - което осигурява ефективен анализ, за сметка на усложняване на вътрешната структура на възлите (категориите), с което се осигурява контрол върху елементите на конституентите. Може да има променливи в правилата, като стойности на признаковите структурите, и така правилото става приложимо към повече ситуации:

(NP **NUMBER** ?n) —> (ART **NUMBER** ?n) (N **NUMBER** ?n)

Сега съгласуването по число е задължително условие за прилагане на правилото. По това правило, NP1 в (1) е допустима конституента. Но (2) и (3) не са допустими:

```
(NP:   1 (ART      NUMBER  s)                    (2)
       2 (N        NUMBER  s))
```

(тук категорията NP няма NUMBER) и

```
(NP    NUMBER  s
       1 (ART      NUMBER  s)                    (3)
       2 (N        NUMBER  pl))
```

(ART и N не са съгласувани по число).

Чрез променливите се задава и многозначност, например в лексикона на синтактичния анализатор: ето думата “fish" в английския се използва както в единствено, така и в множествено число и можем да имаме
(N **ROOT** fish **NUMBER** ?n) или
(N **ROOT** fish **NUMBER** ?n {s p}) (множество допустими стойности) или
(N **ROOT** fish **NUMBER** {s p}) за по-просто.
Когато на променлива се задава списък от допустими стойности, тя се нарича ограничена (**constrained variable**).

Интересен е въпросът, дали разширените БКГ описват същите класове езици както обикновените БКГ. Отговорът зависи от допустимите стойности на feature-структурите и вложеността на feature-структурите. Ако стойностите са крайно множество, без влагане на структури, винаги може да се конструират нови категории и множество от БКГ-правила, които да описват допустимите комбинации от стойности. Тогава имаме разширени граматики с изразителността на БКГ. Иначе се получават граматики, които описват рекурсивно-изброимите езици (общия тип).

IV.1. Някои основни признаци (basic features)

Видяхме вече **NUMBER**. Имаме и **PERSON**. Глаголите имат PERSON в техните форми, но същото може да се каже и за NP-групите. Например,

PERSON 1: NP-групи, които означават или говорителя, или групи хора вкл. говорителя: “аз”, “ние”, “вие и аз”, “аз и Иван”.

PERSON 2: NP-групи, които означават слушателя или групи хора вкл. слушателя: “ти”, “вие”, “всички вие”, “ти и Иван”.

PERSON 3: други NP-групи, за един или повече обекти, но без участие на говорителя и слушателя.

Когато NUMBER и PERSON се срещат заедно, можем да ги комбинираме в едно: 1s, 2s, 3s, 1p, 2p, 3p. Така самият избор на свойствата и техните стойности е важна част от дизайна на граматиката.

Глаголите са разбира се най-трудната лексика за описване, особено при subcategorization. Да навлезем малко в тази материя, която на пръв поглед може да ни се стори шокиращо неприятна. Първо, очевидно имаме признака (feature) **VFORM** за формата на глагола. За простота да разгледаме **VFORM** и негови възможни 7 стойности за английския език:

base - основна форма, напр. go, be, ...
pres - сегашно просто време, напр. във формите goes, go, am, ...
past - минало просто, напр. went, was, said, decided, ...
fin - finite (т.е. лична форма с време, значи *fin* е име на {*pres past*}
ing - сег. причастие, напр. being, going, ... (тези форми нямат време)
pastprt - минало причастие, напр. gone, been, said, decided, ...
inf - специална стойност на **VFORM**, за инфинитив след "to"

Минаваме към категорията **SUBCAT**, с която се описва поведението на глагола относно неговото обкръжение в изречението (но не спрямо подлога). За разлика например от признака **NUMBER** с двете му възможни стойности *s* и *p*, **SUBCAT** има десетки възможни стойности. Долната таблица дава някои възможни стойности на **SUBCAT**, *а за да се подсещаме*, тези стойности са наименовани със съкращения на главните синтактични групи на глаголното обкръжение (*! стойностите са толкова много, че ги забравяме*). Ако някоя категория е допълнително ограничена с друга стойност, след категорията се слага ":" и се задава необходимото ограничение (виж Jack told the man to go).

Стойност	Глагол	Примерно изречение с този глагол
_none	laugh	Jack laughed.
_np	find	Jack found a key.
_np_np	give	Jack gave Sue the paper.
_vp:inf	want	Jack wants to fly.
_np_vp:inf	tell	Jack told the man to go.
_np_vp:ing	catch	Jack caught Sam looking at his desk.
_vp:ing	keep	Jack keeps hoping for the best.

Сега вече сме готови да покажем как може да изглежда правило, по което към глаголите да се присъединява друго изречение с инфинитив:

(VP) —> (V **SUBCAT** _np_vp:inf)
(NP)
(VP **VFORM** inf)

Много глаголи имат допълнения с предлози (непреки допълнения, indirect objects). Въвеждаме признака (feature) **PFORM**, и групираме предлозите в няколко вида стойности - както е показано на следващата таблица. В нея различаваме например LOC за място и MOT(ion) за глаголи като "вървя", за да опишем някои аспекти на пътя. Част от стойностите са трудно разграничими при отделните думи; но както вече казахме, изборът им е произволен и от неговия успех и адакватност зависи дизайнът на граматиката.

Стойност	Примерен предлог	Примерно изречение
TO	to	I gave it to the bank.
LOC	in, on, by, inside, on top of, ...	I put it on the desk.
MOT	to, from, along, ...	I walked to the store.

Ето и други **SUBCAT** стойности:

Стойност	Глагол	Примерно изречение с този глагол
_np_pp:to	give	Jack gave the key to the man.
_pp:loc	be	Jack is in the store.
_np_pp:loc	put	Jack put the box in the corner.
_pp:mot	go	Jack went to the store.
_np_pp:mot	take	Jack took the hat to the party.
_adjp	be, seem	Jack is happy.
_np_adjp	keep	Jack kept the dinner hot.
_s:for	hope	Jack hoped for the man to win the prize.
_s:that	believe	Jack believed that the world was flat.

Сега имаме напр. (VP) —> (V **SUBCAT** _np_pp:loc)
(NP)
(VP **PFORM** LOC)

Други свойства: например двоични, такова е **INV**, което показва дали изречението има инвертирана структура, например S="Jack laughed" има **INV** -, а S="Did Jack laught?" има **INV** +.

Със свойствата често е полезно да свързваме *стойност по премълчаване* (default value), например може по премълчаване **INV** -.

IV.2. Какво става с лексикона при такъв подход?

Почваме да слагаме признакови структури и в лексикона (тоест всичко - и правилата, и лексикона - става признакови структури), и съответно да прилагаме правила за лексически изводи (а не само за синтактични). В по-далечен план, feature-системите се превръщат във feature-логики, и така третираме комплексно лексиката, синтаксиса и тая част от семантиката, която можем да си представим описана във вид на feature-структури.

Примери за лексически правила (*лексически* значи, че работят над думи): напр. как формата 3s се генерира от основната с добяване на "S"

(V **ROOT** ?r **SUBCAT** ?s **VFORM** pres **AGR** 3s) —>
(V **ROOT** ?r **SUBCAT** ?s **VFORM** base) (+S)

където (+S) е нова лексикална категория, съдържаща само морфемата -s. Това правило може да се комбинира с лексикона

want: (V **ROOT** want
SUBCAT {_np_vp:inf _vp:inf}
VFORM base)

и при даден вход "want-s" в някой момент на извода ще се получи следната конституента:

```
want: (V    ROOT  want
            SUBCAT    {_np_vp:inf _vp:inf}
            VFORM     pres
            AGR  3s)
```

Друг пример: правило което показва, че всички форми освен в 3л. ед.число са еквивалентни на основната при правилни глаголи със свойството IRREG-PRES **–** (т.е. липсва нерегулярност в сегашно време):

(V **ROOT** ?r **SUBCAT** ?s **VFORM** pres **AGR** {1s 2s 1p 2p 3p}) —>
(V **ROOT** ?r **SUBCAT** ?s **VFORM** base **IRREG-PRES –**)

Да обобщим няколко правила за лексически изводи на общи наставки в английския език в следната граматика от 7 правила:

Present Tense	1. (V **ROOT** ?r **SUBCAT** ?s **VFORM** pres **AGR** 3s) —> (V **ROOT** ?r **SUBCAT**?s **VFORM** base **IRREG-PRES –**)) (+S) 2. (V **ROOT** ?r **SUBCAT** ?s **VFORM** pres **AGR** {1s 2s 1p 2p 3p}) —> (V **ROOT** ?r **SUBCAT** ?s **VFORM** base **IRREG-PRES –**)
Past Tense	3. (V **ROOT** ?r **SUBCAT** ?s **VFORM** past **AGR** {1s 2s 3s 1p 2p 3p}) —> (V **ROOT** ?r **SUBCAT** ?s **VFORM** base **IRREG-PRES –**) (+ED)
Past Parti-ciple	4. (V **ROOT** ?r **SUBCAT** ?s **VFORM** pastprt) —> (V **ROOT** ?r **SUBCAT** ?s **VFORM** base **EN-PASTPRT –**) (+ED) 5. (V **ROOT** ?r **SUBCAT** ?s **VFORM** pastprt) —> (V **ROOT** ?r **SUBCAT** ?s **VFORM** base **EN-PASTPRT +**) (+EN)
Present Partic.	6. (V **ROOT** ?r **SUBCAT** ?s **VFORM** ing) —> (V **ROOT** ?r **SUBCAT** ?s **VFORM** base) (+ING)
Plural Nouns	7. (N **ROOT** ?r **AGR** 3p) —> (N **ROOT** ?r **AGR** 3p **IRREG-PL –**) (+S)

Сега вече можем да напишем един примерен малък лексикон:

```
a:    (CAT ART
      ROOT A1
      AGR  3s)
be:   (CAT V
      ROOT BE1
      VFORM       base
      IRREG-PRES  +
      IRREG-PAST  +
      SUBCAT      {_adjp_np} )
cry:  (CAT V
      ROOT CRY1)
      VFORM       base
      SUBCAT      none )
dog:  (CAT N
      ROOT DOG1
      AGR  3s)
fish: (CAT N
      ROOT FISH1
saw:  (CAT N        трион
      ROOT SAW1
      AGR  3s)
saw:  (CAT V        сея
      ROOT SAW2
      VFORM base
      SUBCAT  _np)
saw:  (CAT  V
      ROOT SEE1
      VFORM past
      SUBCAT  _np)
see:  (CAT V
      ROOT SEE1
      VFORM base
      SUBCAT  _np
      IRREG-PAST +
      EN-PASTPRT +)
seed: (CAT N
```

AGR {3s 3p}
IRREG-PL {+ -})
happy: (**CAT** ADJ
ROOT HAPPY1
SUBCAT _vp:inf)
he: (**CAT** PRO
ROOT HE1
AGR 3s)
is: (**CAT** V
ROOT BE1
VFORM pres
SUBCAT{_adjp_np}
AGR 3s)
Jack: (**CAT** NAME
AGR 3s)
man: (**CAT** N
ROOT MAN1
AGR 3s)
men: (**CAT** N
ROOT MAN1
AGR 3p)
ROOT SEED1
AGR {3s 3p})
the: (**CAT** ART
ROOT THE1
AGR {3s 3p})
to: (**CAT** PREP
ROOT TO1)
want: (**CAT** V
ROOT WANT1
VFORM base
SUBCAT{_np_vp:inf, _vp:inf})
was: (**CAT** V
ROOT BE1
VFORM past
SUBCAT{_adjp_np}
AGR {1s 3s})
were: (**CAT** V
ROOT BE
VFORM past
SUBCAT{_adjp_np}
AGR {2s 1p 2p 3p})

Забележете, че за всяка дума е показано как тя се събкатегоризира в изреченията, т.е. какви думи могат да се долепят до нея с цел изграждане на правилни конституенти. Напр. глаголът want може да участва в два вида изречения: Jack wants Sam to go (при **SUBCAT** =_np_vp:inf) и Jack wants to go (при **SUBCAT** =_vp:inf}). На практика в лексикона се описва синтактичното поведение на отделните думи, докато в граматиката се кодира знание как в изреченията се строят конституенти (групи от думи). И двете описания са много обемни и се съставят за десетки и стотици човекогодини (морфологичното знание е сравнително по-просто).

Много признаци се конструират така, че да има съвпадение на признаците на конституентата-"връх" и на главната конституента на правилото (глава), например във всички VP-правила **VFORM** и **AGR** са еднакви за VP-категорията вляво на —> и за главния глагол (глава - head):

(VP **VFORM** ?v **AGR** ?a) —>
(V **VFORM** ?v **AGR** ?a **SUBCAT** {_np _vp:inf})
(NP)
(VP **VFORM** inf)

Това е запис на факта, че някои VP-та се състоят от линейно съчетани V NP VP, като се задават условията на съгласуване между елементите им.

IV.3. Можем вече да дадем **пример на БКГ, разширена с признаци:**

1. (S **INV** - **VFORM** ?v {pres past} **AGR** ?a) —>
(NP **AGR** ?a) (VP **VFORM** ?v {pres past} **AGR** ?a)

2. (NP **AGR** ?a) —> (ART **AGR** ?a) (N **AGR** ?a)
3. (NP **AGR** ?a) —> (PRO **AGR** ?a)
4. (VP **AGR** ?a **VFORM** ?v) —> (V **SUBCAT** _none **AGR** ?a **VFORM** ?v)

5. (VP **AGR** ?a **VFORM** ?v) —> (V **SUBCAT** _np **AGR** ?a **VFORM** ?v) NP

6. (VP **AGR** ?a **VFORM** ?v) —>
   (V **SUBCAT** _vp:inf **AGR** ?a **VFORM** ?v) (VP **VFORM** inf)

7. (VP **AGR** ?a **VFORM** ?v) —>
   (V **SUBCAT** _np_vp:inf **AGR** ?a **VFORM** ?v) NP (VP **VFORM** inf)

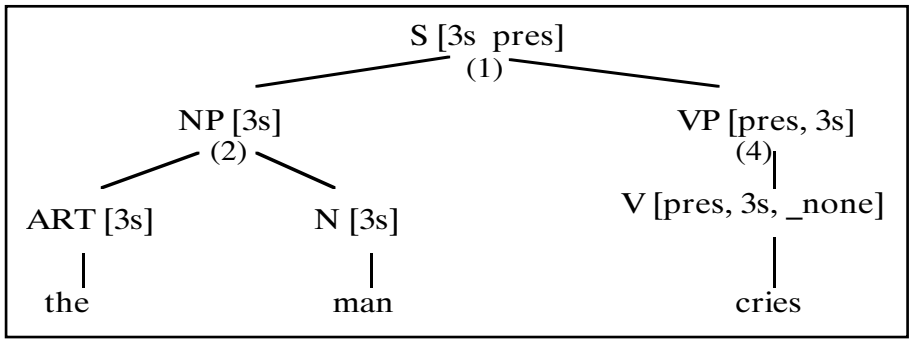
8. (VP **AGR** ?a **VFORM** ?v) —> (V **SUBCAT** _adjp **AGR** ?a **VFORM** ?v) **ADJP**

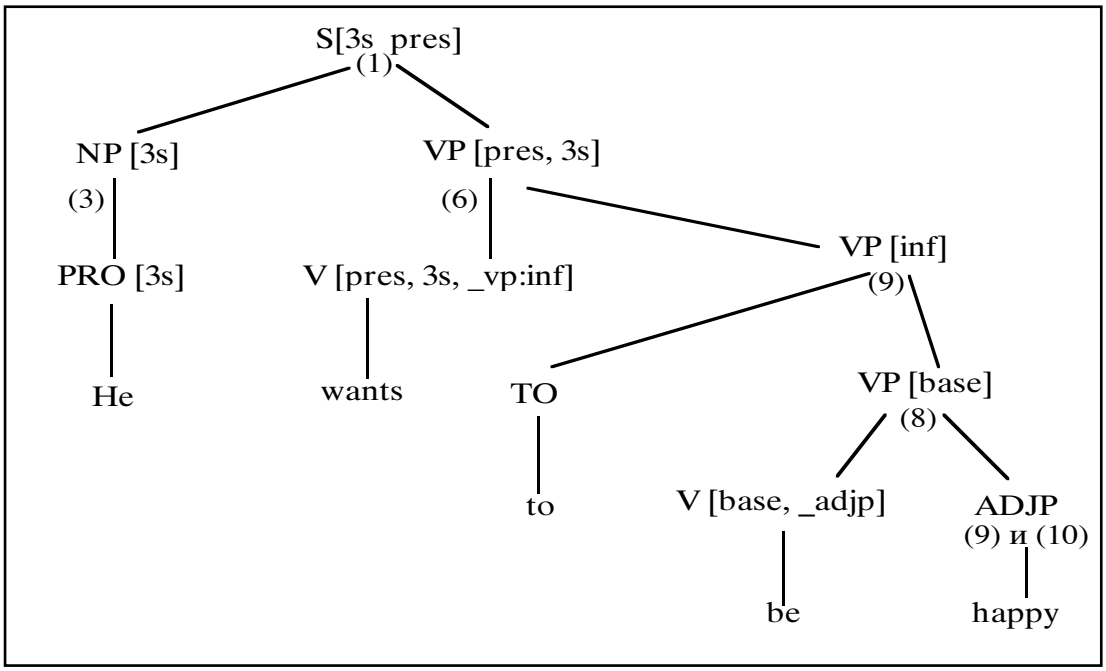
9. (VP **SUBCAT** inf **AGR** ?a **VFORM** inf) —>
   (TO **AGR** ?a **VFORM** inf) (VP **VFORM** base)

10. ADJP —> ADJ

Тоест, правило (1) показва, че S се състои от NP и VP. По правила (2) и (3), NP е или ART N, или PROnoun. Правила (4)-(9) описват 6 вида конституети VP, с различни конфигурации, които се управляват от признака SUBCAT и различни негови стойности.

Ето и съответни дървета на извода в граматиката, с отбелязани във върха правила за извод (цифра - пореден No. на правило). Единствените стойности на признаците, получени при извода, са изброени по върховете в скоби [...]:





В резюме: накарали сме безконтекстните граматики в процеса на извод да "контролират" зависимости между категориите, които участват в един и същи контекст (или в една и съща конструкция, в една и съща конституента). Така сме разширили изразителната им сила (или поне горните примери дават идея как може да се постигне това). Забележете колко по-мощна е последната граматика с 10-те си правила и колко пълно описва граматическото знание в разпознаваните изречения. Но придобивките са за сметка на увеличаване сложността на правилата, поради което такива граматики се пишат и тестват по-трудно.

ЗА САМОСТОЯТЕЛНИ УПРАЖНЕНИЯ:

Задача 1: Начертайте всички (според вас) дървета на синтактични зависимости за изречението:
Иван пише бързо писмо за разрушаването на Мавзолея тази седмица.
Опитайте се да начертаете някакви конституентни дървета за тези значения (редно е да начертаете по едно дърво за всяко четене, както с "Времето лети като стрела" в лекция 1).

Задача 2: Напишете мрежа с рекурсивни преходи, състоящата се от три мрежи S, NP и PP, която разпознава езика на следната граматика:

S —> NP VP	NP —> ART NP2	NP2 —> NP3 PP
VP —> V	NP —> NP2	NP3 —> N
VP —> V NP	NP2 —> N	PP —> PREP NP
VP —> V PP	NP2 —> ADJ NP2	

Упътване: използвайте дъги от всички типове.

Задача 3: Разпознава ли граматиката от част IV.3 изречения от типа

John saw the Grand Canion flying to Boston

и ако да (или не), защо? Можете ли да добавите *точно* едно правило, за да осигурите разпознаването на това изречение *само* в четенето "Джон видя Големия Каньон, докато каньонът летеше към Бостън"?