

# Синтаксический анализ в терминах деревьев зависимостей

Использовались слайды из презентации Joakim Nivre (ACL Tutorial)

# Теоретические предпосылки

✓ линейной цепочки словоформ в  
предложении соответствует нелинейная  
иерархическая структура

# Теоретические предпосылки

Отражение иерархической структуры в синтаксическом представлении:

- синтаксические единицы
- синтаксические связи
- направленные синтаксические связи
- набор правил, устанавливающих связи между синтаксическими единицами;
- набор правил, устанавливающих направление связи;
- набор правил, устанавливающих тип связи (ярлыки)
- правила проверки грамматической правильности

# Предыдущие лекции про синтаксис

- Базовые системы формального представления синтаксической структуры:
  - деревья непосредственных составляющих
  - деревья зависимостей
  - Head Driven Phrase Structure Grammar (HDPSG)
  - категориальные грамматики
  - LFG (Lexical functional grammar)
  - и др.

# Основные подходы

- КС-грамматики (CFG context-free grammar)
- вероятностные контекстно-свободные грамматики (PCFG – probabilistic context-free grammar)
- унификационные грамматики
- «зависимостный» анализ (dependency parsing)

# Унификационные грамматики. Пример

$$\begin{array}{c} \text{Verb} \\ \left[ \begin{array}{l} \text{arg0} \left( \boxed{1} \left[ \begin{array}{l} \text{NP} \\ \text{case = nom} \end{array} \right] \right) \\ \text{syn} \\ \text{sub} = \boxed{1} \end{array} \right] \end{array} \quad \begin{array}{c} \text{TransVerb} \\ \left[ \begin{array}{l} \text{arg1} \left( \boxed{2} \left[ \begin{array}{l} \text{NP} \\ \text{case = acc} \end{array} \right] \right) \\ \text{syn} \\ \text{ARG-S} < \boxed{1}, \boxed{2} > \end{array} \right] \end{array}$$

TransVerb < Verb

$$\begin{array}{c} \text{LexEntry} \\ \left[ \begin{array}{l} \text{TransVerb} \\ \text{VerbMorph} \\ \text{syn} \\ \text{m or} \\ \left[ \begin{array}{l} \text{root = give} \\ \text{form3 = gives} \\ \text{form4|stem = gave} \end{array} \right] \\ \text{sem = give2a} \end{array} \right] \end{array}$$
$$\begin{array}{c} \text{Verb} \\ \left[ \begin{array}{l} \text{Agr} \\ \left\{ \begin{array}{l} \text{per} \\ \left\{ \begin{array}{l} \text{1st} \\ \text{2nd} \end{array} \right\} \\ \text{num sing} \end{array} \right\} \\ \left[ \begin{array}{l} \text{Agr} \\ \text{num plur} \end{array} \right] \end{array} \right] \\ \text{agr} \boxed{1} \end{array} \\ \text{subj} \left[ \begin{array}{l} \text{Noun} \\ \text{agr} \boxed{1} \end{array} \right]$$

# Роль синтаксического компонента

- Вспомогательный компонент автоматической обработки; нужен для улучшения качества решения конечных задач систем автоматического анализа текста (МП, извлечение информации из текста)

# Проблемы автоматического синтаксического анализа

- «дальние» связи:
  - например, согласование,
  - модели управления
- «непроективность»:
  - перемещения, свободный порядок слов, разрывные составляющие
- нули:
  - элипсис,
  - нулевые связи и местоименные элементы
  - (PRO – ноль при инфинитиве, pro – восстановимое опущение анафорического местоимения)
- омонимия:
  - предложные группы, приложения и т.п. – необходимость распознавания главных актантов глагола для целевой задачи системы
- принципиально большое количество переборов и допустимых синтаксических структур при автоматическом анализе (необходимо учитывать соседние уровни: морфологический анализ и семантику)

# Способы борьбы

- согласование:
  - унификации, ограничения
- множество переборов
  - вероятностные грамматики, лексикализованные вероятностные контекстно-свободные грамматики (машинное обучение по трибандкам)
  - ограничения
  - унификационные грамматики
- нули, непроективность и т.п.
  - модели с перемещением категорий и следами, частичный синтаксический анализ: shallow parsing, chunking

# КС-грамматики. Алгоритмы анализа

- Метод спуска (top-down)
- Восходящий анализ
- Алгоритм Кока-Янгера-Касами (CYK)
- Алгоритм Эрли (EARLEY)

# Решения в рамках подхода НС-структур

## Изменение порядка слов.

### Непроективность

- Изменение порядка слов:

Смещение категорий – один из способов описания разрывных составляющих.

Используется идея представления разрывных составляющих через указание условного „канонического“ места категории и места, куда категория была смещена.

(Данный подход используется в Универсальной грамматике [Chomsky 1995].)

Название смещённой категории начинается с символа „\$“, а название следа смещённой категории – с символа подчёркивания Так, предложения типа рус.

*Дом я вижу* описывается следующим множеством правил (упрощённо):

$S \rightarrow <\$NP> NP VP$

$VP \rightarrow Verb \underline{NP}$

Цитируется по [Перекрестенко А. 2003] Перекрестенко, Александр. Создание парсера для некоторых классов контекстно-зависимых языков для задач автоматического синтаксического анализа. // Труды международной конференции Диалог'2003 (Протвино, 11-16 июня, 2003 г.). URL: <http://www.dialog-21.ru/Archive/2003/Perekrestenko.htm>

# Деревья зависимости

## План

- Методы «зависимостного» анализа
  - синтаксическое представление в терминах зависимостей
  - идеи базовых алгоритмов
  - машинное обучение
- Доступные ресурсы для разных языков
  - Парсеры
  - Трибанки

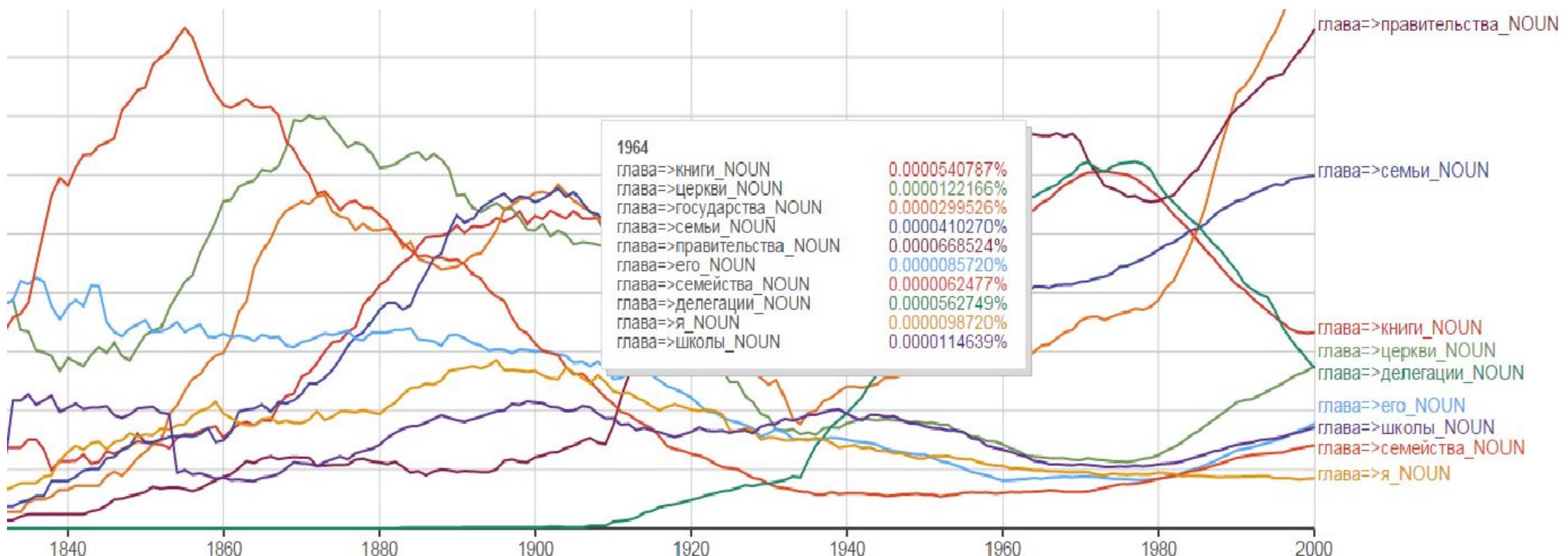
# Синтаксическое представление в терминах зависимостей

# Мотивация

- Повышенный интерес к анализу в терминах грамматики зависимостей
- ▲ CoNLL-X shared task (June, 2006)
- Google N-grams  
(<http://googleresearch.blogspot.ru/2013/05/syntactic-ngrams-over-time.html>, <https://books.google.com/ngrams/>)
- <http://nlp.stanford.edu/software/stanford-dependencies.shtml>
- Проект по разработке универсальных принципов аннотации в терминах зависимостей

# Google N-Grams с зависимостями

Глава => Noun

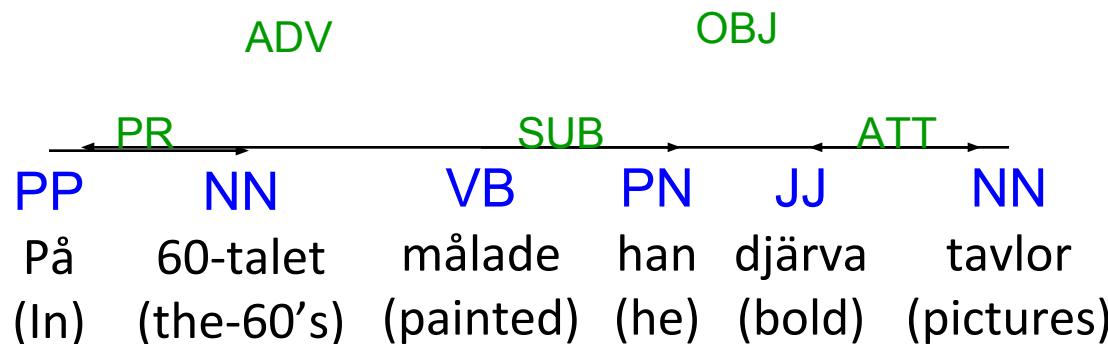


# Синтаксис зависимостей (dependency parsing)

- Синтаксическое представление:
  - лексические узлы, связанные бинарным асимметричным отношением
- Теньер (Lucien Tesni`ere [Tesni`ere 1959])
  - «Основы структурного синтаксиса» ([фр.](#) *Eléments de syntaxe structurale*)
    - вербоцентричность, синтаксис зависимостей:
- бинарные направленные отношения между элементами предложения
- предикат в качестве единственной вершины графической «стеммы»

# Представление синтаксической структуры в терминах зависимостей

- Directed graphs:
  - $V$  is a set of nodes (tokens)
  - $E$  is a set of arcs (dependency relations)
  - $L$  is a labeling function on  $E$  (dependency types)
- Example:



# Ограничения

- единственность вершины
- связанность
- ацикличность
- проективность

# Dependency Parsing

- Dependency-based grammar parsing:
  - Given a dependency grammar  $G$  and an input string  $x \in \Sigma^*$ , derive some or all of the dependency graphs  $y$  assigned to  $x$  by  $G$ .
- Dependency-based text parsing:
  - Given a text  $T = (x_1, \dots, x_n)$ , derive the correct dependency graph  $y_i$  for every sentence  $x_i \in T$ .
- Text parsing may be grammar-driven or not.

# Примеры



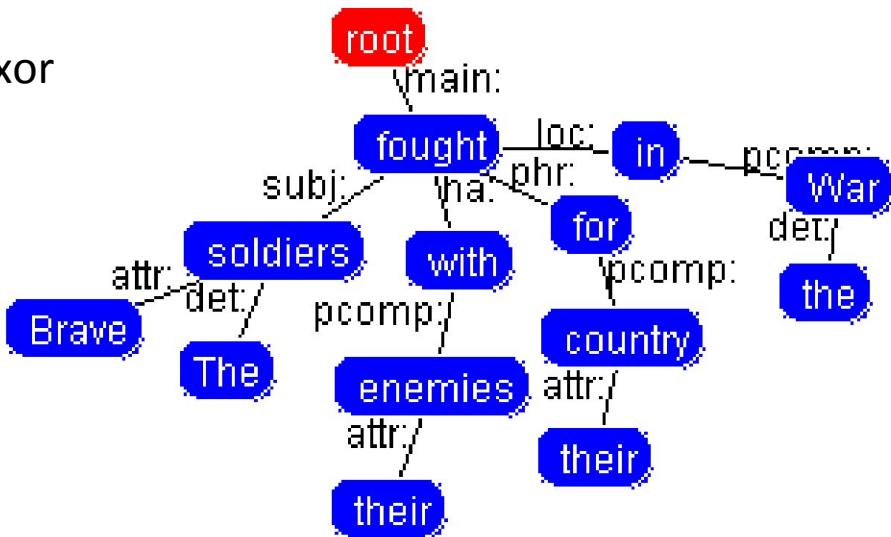
СинтагРус

GS

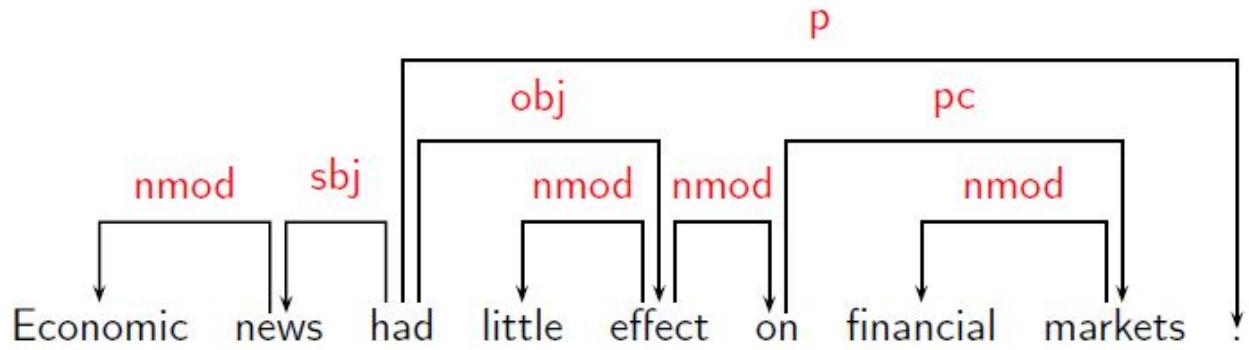
Группы существуют столько же лет, сколько и само человечество, если не дольше.

Rus-TreeBank

Connexor



MaltParser



# Сравнение зависимостей и НС

- Разметка дерева зависимостей включает
  - ✓ связи между парами слов: главное слово–зависимое (head-dependent relation) – направленные стрелки (**directed arcs**);
  - ✓ именование разных типов синтаксических отношений (например, определительное отношение, обстоятельственное отношение и т.п.) – метки (**arc labels**);
  - ✓ грамматические категории (structural categories) – теги частей речи
- ✓ Разметка дерева непосредственных составляющих подразумевает:
  - нетерминальные составляющие или фразовые категории (**non-terminal nodes**);
  - дополнительное именование возникающих в процессе деривации сложных синтаксических объектов (именная группа, предложная группа и т.п.) (**non-terminal labels**);
  - функциональные категории (грамматические функции).
- Гибридные системы могут включать все элементы

# Немного теории

- Достаточно ли только указания на зависимостные отношения между двумя элементами? Необходимо ли указывать на направленное отношение между элементами?
- Многоуровневое синтаксическое представление или нет?
- Какие элементы являются узлами:
  - Морфемы?
  - Словоформы?
  - Словосочетания?

# Немного теории

- Какова природа зависимостных отношений? Как устроены синтаксические метки?
  - грамматические функции?
  - семантические роли?
- На основании каких критериев выделять главное и зависимое?
- Какими структурными свойствами должны обладать деревья зависимостей (ДЗ)?

# Критерии выделения главного vs. зависимого

- Критерий установления связи между  $H$  и  $D$  в конструкции  $C$  [Zwicky 1985, Hudson 1990]
1.  $H$  определяет синтаксическую категорию  $C$ .  $H$  может замещать  $C$
  2.  $H$  определяет семантическую категорию  $C$ ,  $D$  уточняет  $H$
  3.  $H$  обязательно,  $D$  может быть optional
  4.  $H$  накладывает селективные ограничения на  $D$  и определяет, обязательно ли  $D$
  5. Грамматические характеристики  $D$  зависят от свойств  $H$  (согласование или управление)
  6. Линейная позиция  $D$  определяется по отношению к  $H$

# Критерии выделения главного vs. зависимого

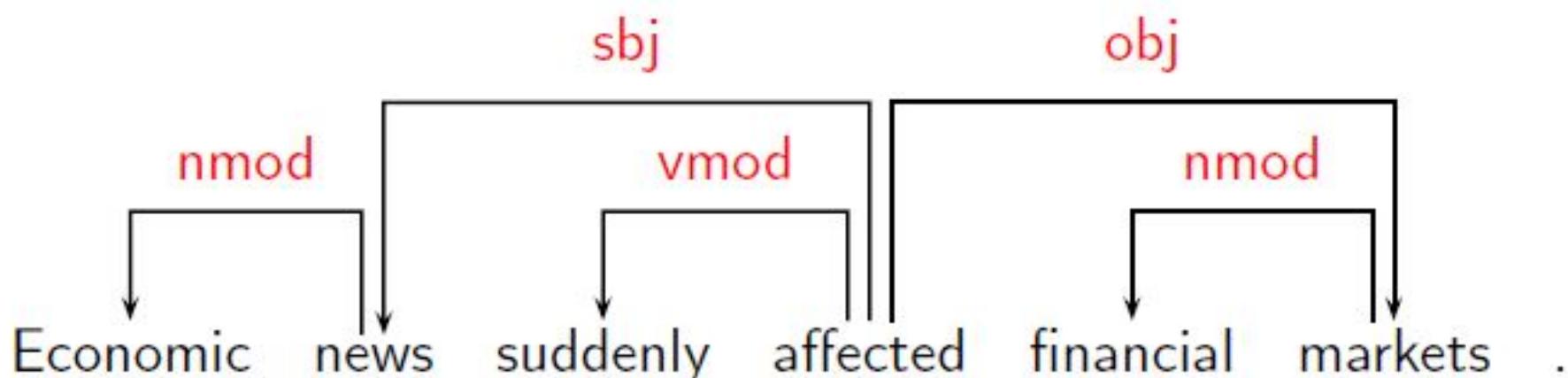


• Я вижу два стола Num N-GEN Num-НеодушN-Неодуш

- Я вижу двух человек *Num* *N-GEN* *Num-одуш* *N-одуш*

## Some Clear Cases

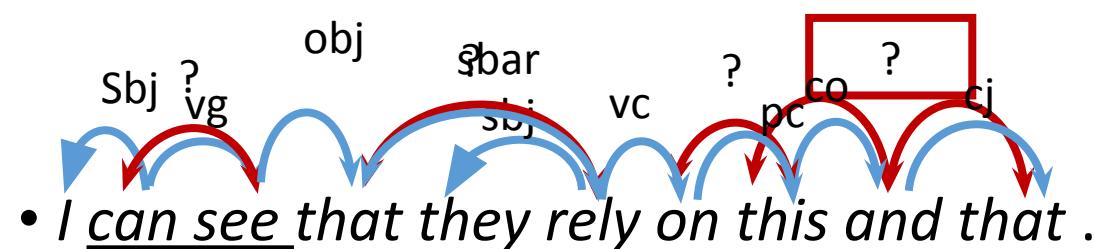
Construction	Head	Dependent
Exocentric	Verb	Subject ( <i>sbj</i> )
	Verb	Object ( <i>obj</i> )
Endocentric	Verb	Adverbial ( <i>vmod</i> )
	Noun	Attribute ( <i>nmod</i> )



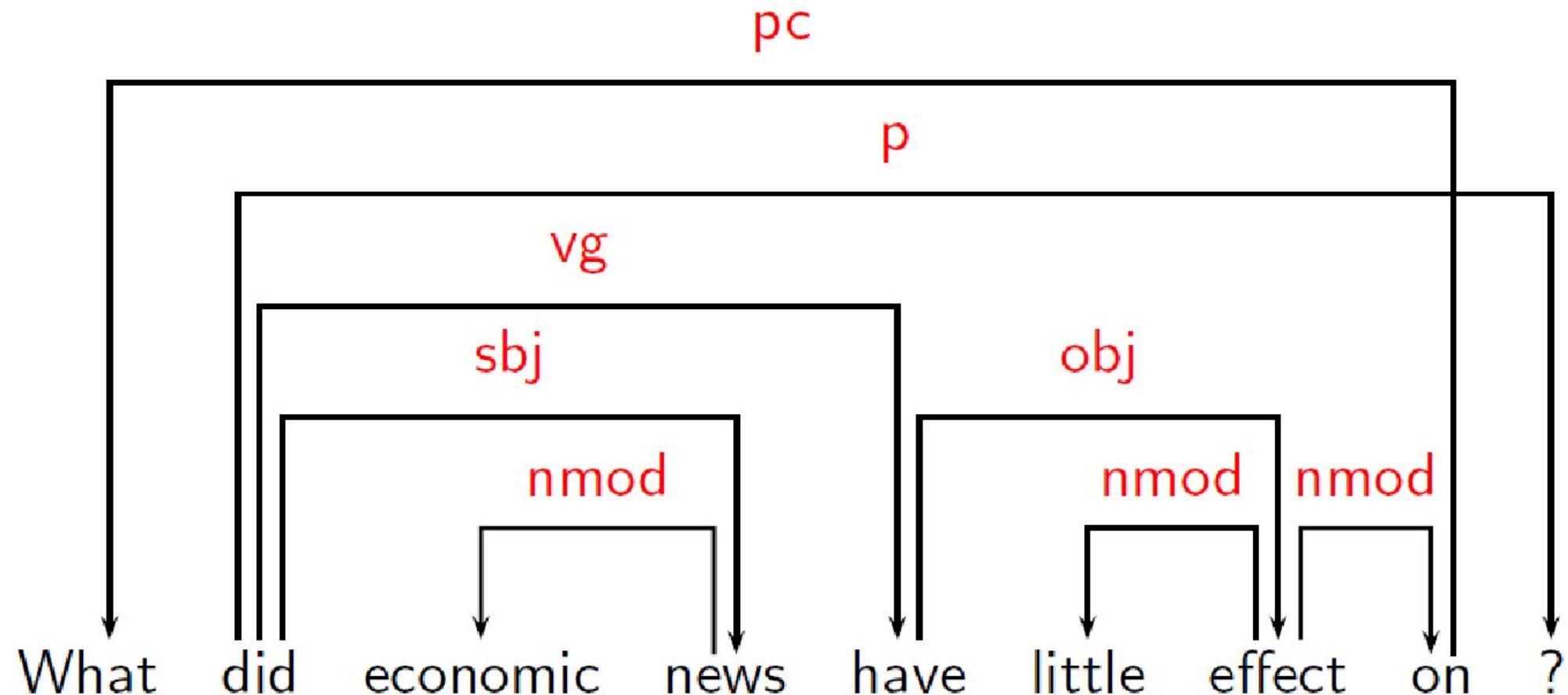
# Сложные случаи

## Some Tricky Cases

- ▲ Complex verb groups (auxiliary ↔ main verb)
- ▲ Subordinate clauses (complementizer ↔ verb)
- ▲ Coordination (coordinator ↔ conjuncts)
- ▲ Prepositional phrases (preposition ↔ nominal)
- ▲ Punctuation



# Сложные случаи: непроективность



# Pros and Cons of Dependency Parsing

- ▶ What are the advantages of dependency-based methods?
- ▶ What are the disadvantages?
- ▶ Four types of considerations:
  - ▶ Complexity
  - ▶ Transparency
  - ▶ Word order
  - ▶ Expressivity

# Дерево зависимостей

- ▶ A dependency structure can be defined as a directed graph  $G$ , consisting of
  - ▶ a set  $V$  of nodes,
  - ▶ a set  $E$  of arcs (edges),
  - ▶ a linear precedence order  $<$  on  $V$ .
- ▶ Labeled graphs:
  - ▶ Nodes in  $V$  are labeled with word forms (and annotation).
  - ▶ Arcs in  $E$  are labeled with dependency types.
- ▶ Notational conventions ( $i, j \in V$ ):
  - ▶  $i \rightarrow j \equiv (i, j) \in E$
  - ▶  $i \rightarrow^* j \equiv i = j \vee \exists k : i \rightarrow k, k \rightarrow^* j$

# Дерево зависимостей

- ▶  $G$  is (weakly) **connected**:
  - ▶ For every node  $i$  there is a node  $j$  such that  $i \rightarrow j$  or  $j \rightarrow i$ .
- ▶  $G$  is **acyclic**:
  - ▶ If  $i \rightarrow j$  then not  $j \rightarrow^* i$ .
- ▶  $G$  obeys the **single-head** constraint:
  - ▶ If  $i \rightarrow j$ , then not  $k \rightarrow j$ , for any  $k \neq i$ .
- ▶  $G$  is **projective**:
  - ▶ If  $i \rightarrow j$  then  $i \rightarrow^* k$ , for any  $k$  such that  $i < k < j$  or  $j < k < i$ .

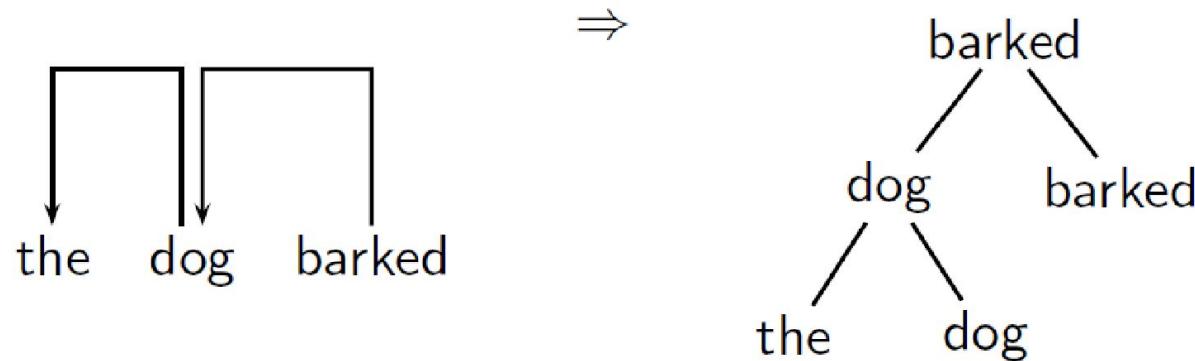
# Методы

# Методы

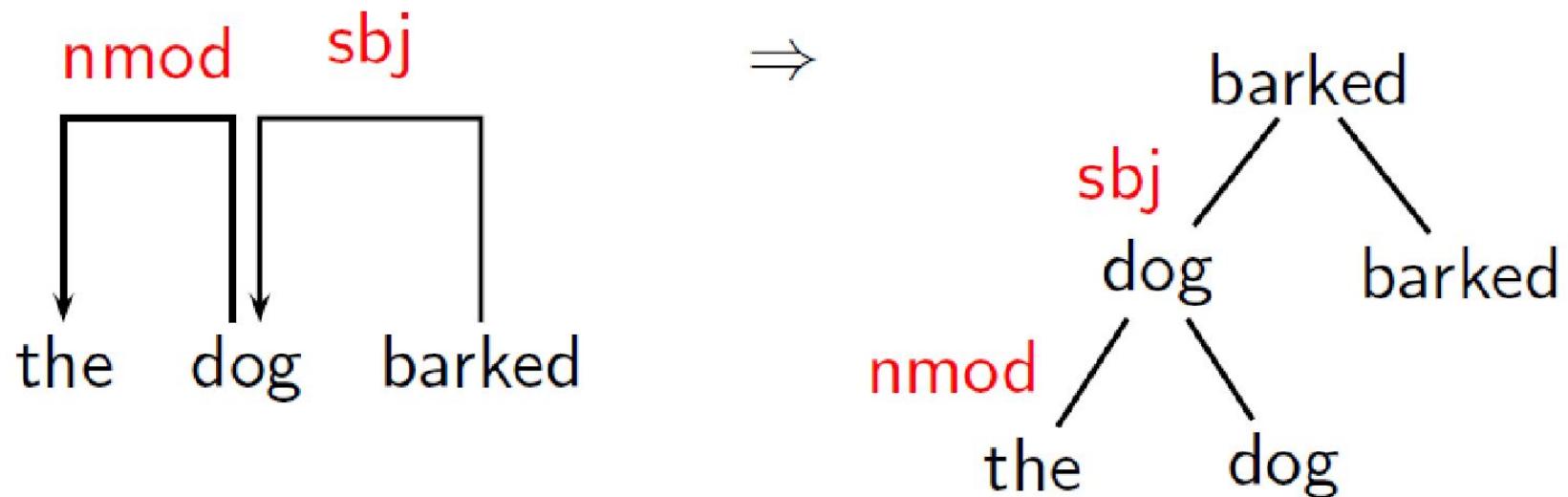
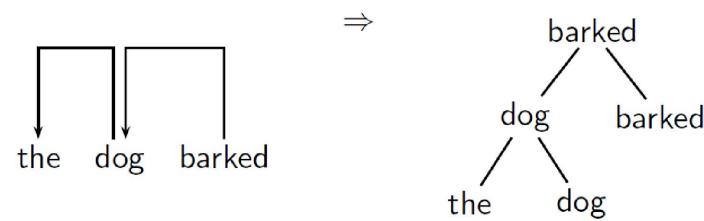
- Три базовых подхода:
  - Dynamic programming algorithms applied to context-free (projective) dependency grammars (методы анализа для КСГ (динамическое программирование), применимые к проективным зависимостным грамматикам)
  - Eliminative parsing techniques applied to constraint-based formulations of (non-projective) dependency grammar (методы, основанные на удалении связей, не удовлетворяющих ограничениям)
  - Deterministic parsing algorithms combined with weak grammars or data-driven methods

# Переход к НС-представлению (динамическое программирование)

- Основная идея: перейти от зависимостей к непосредственным составляющим
- Применить стандартный метод для синтаксического анализа в терминах КС-грамматик (например, алгоритм Кока – Янгера – Касами)



# Переход к НС-представлению



# Переход к НС-грамматикам

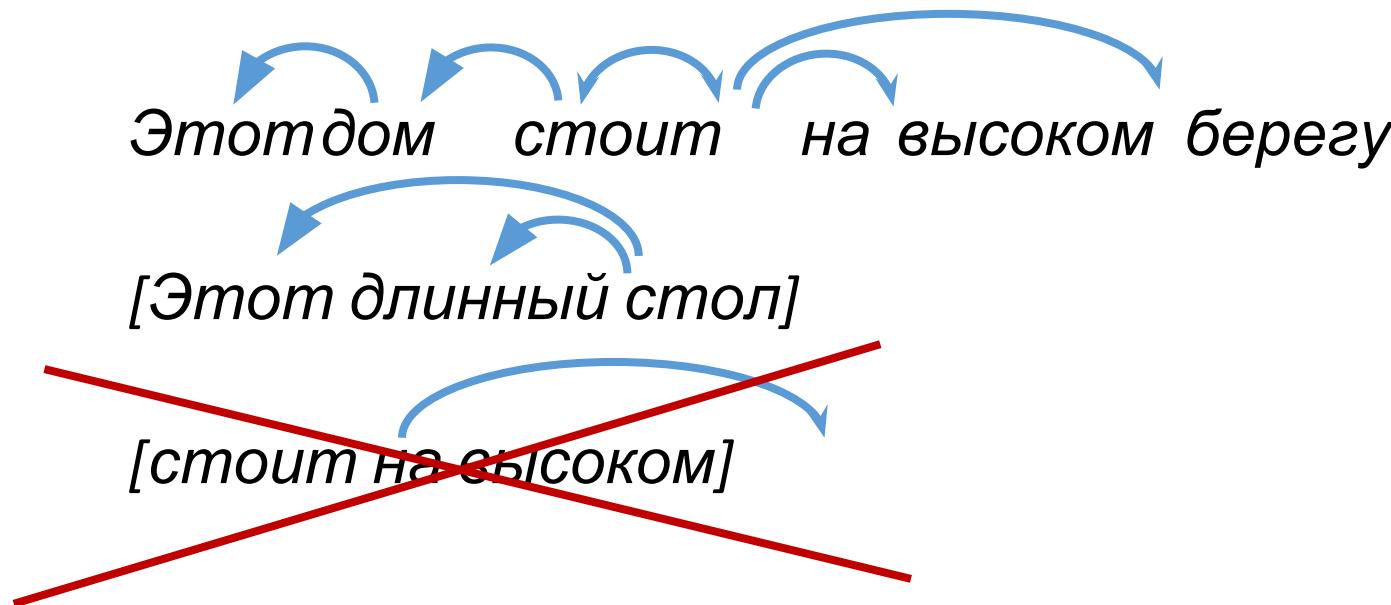
- Грамматика – контекстно-свободная грамматика, каждый узел – лексикализован
- ▲ Слот диаграммы – поддерево, слова со всеми их зависимыми (левыми и правыми).
- ▲ Проблемы: цепочки могут различаться вершинами (разные поддервья в зависимости от выделяемой вершины)
- ▲ Вычислительная сложность:  $O(n^5)$ .

# Переход к НС-грамматикам: Eisner's Bilexical Algorithm

- Направления изменений
  - Модификация алгоритма
  - Вероятностный анализ
- Вычислительная сложность  $O(n^3)$
- Модификация: вместо поддеревьев – отрезки словоформ определенного вида (spans)
- Определение: никакая «внутренняя» словоформа из отрезка не может быть связаны ни с какой словоформой извне
- (только крайние словоформы могут быть связаны со словоформами вне отрезка)
- Основная идея: только крайние словоформы могут быть “активны”, т.е. требовать хозяина (вершины)

# Переход к НС-грамматикам: Eisner's Bilexical Algorithm

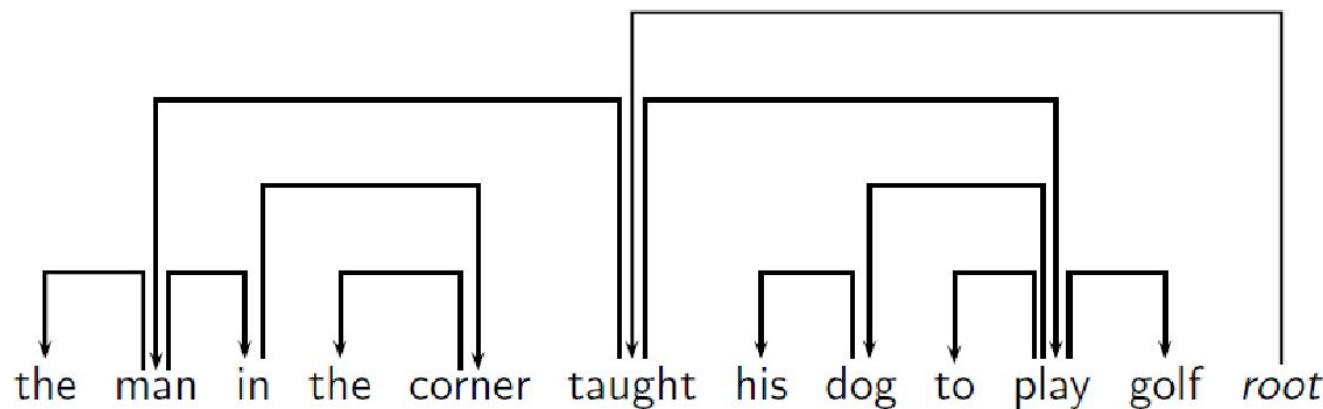
Пример:



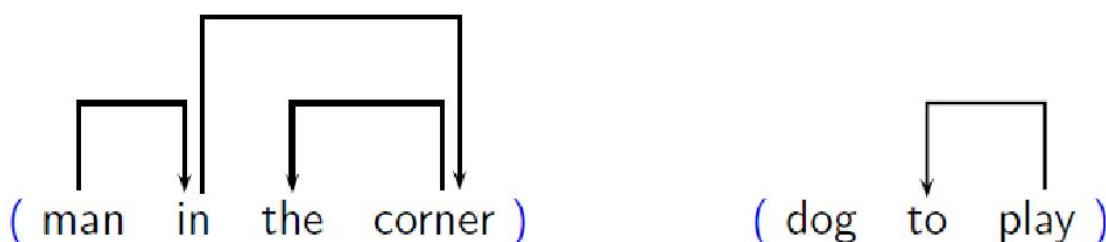
# Пример

Начинаем с объединения в отрезки (spans) смежные слова

**Example**

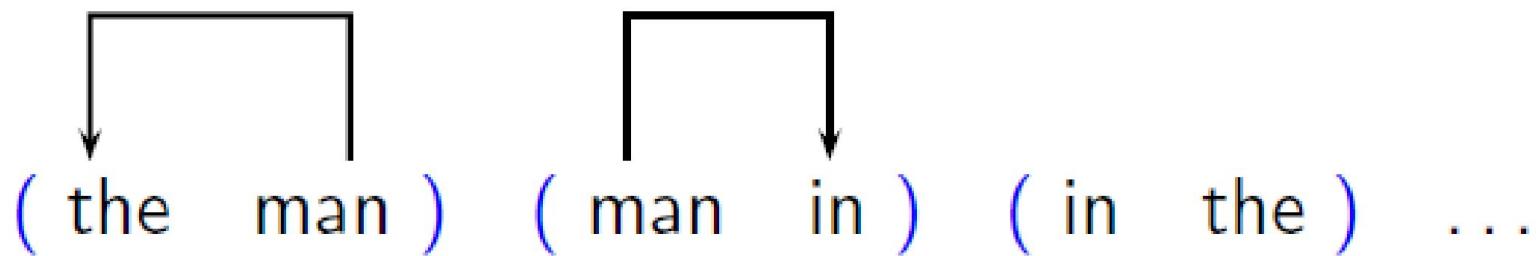


Spans:



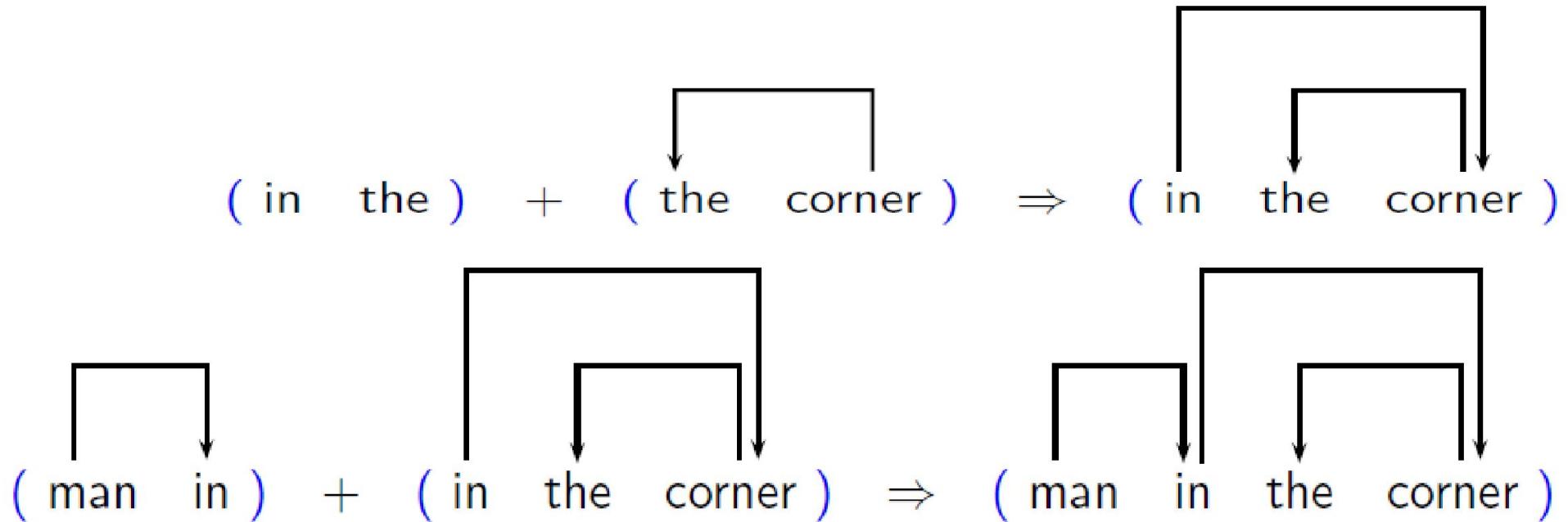
# Переход к НС-грамматикам: Eisner's Bilexical Algorithm

Начинаем с объединения в отрезки (spans) смежные слова



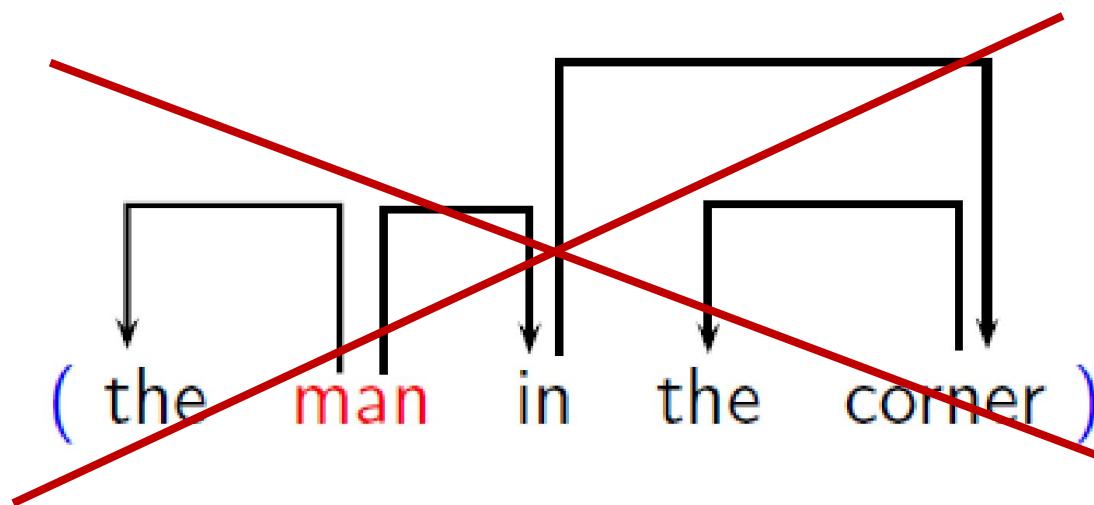
# Переход к НС-грамматикам: Eisner's Bilexical Algorithm

Объединяем в отрезки (spans) отрезки, имеющих словоформу в пересечении; это слово должно иметь «хозяина» либо в левом, либо в правом отрезке



# Переход к НС-грамматикам: Eisner's Bilexical Algorythm

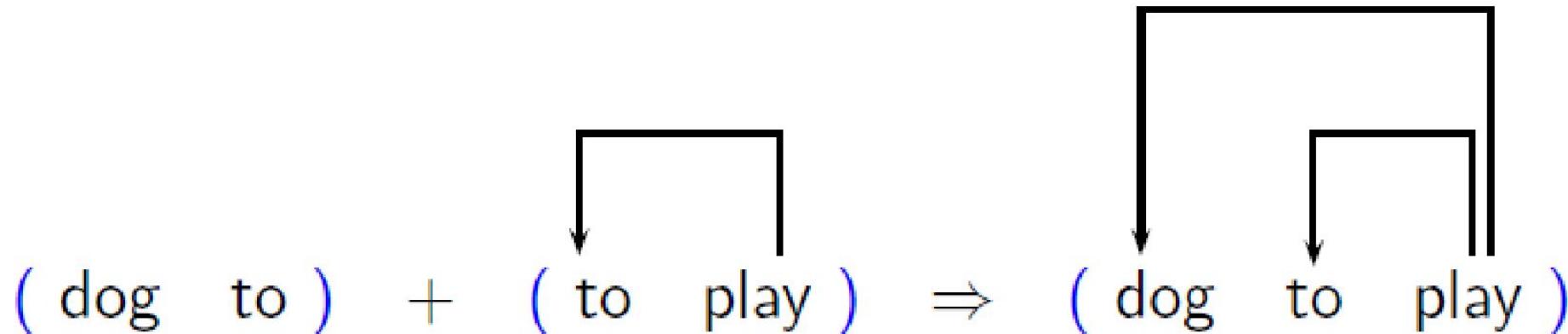
Объединяя в отрезки (spans) отрезки, имеющих словоформу в пересечении; это слово должно иметь «хозяина» либо в левом, либо в правом отрезке



«хозяин» словоформы *man* не находится ни в одном из объединяемых отрезков (*the man* и *man in the corner*)

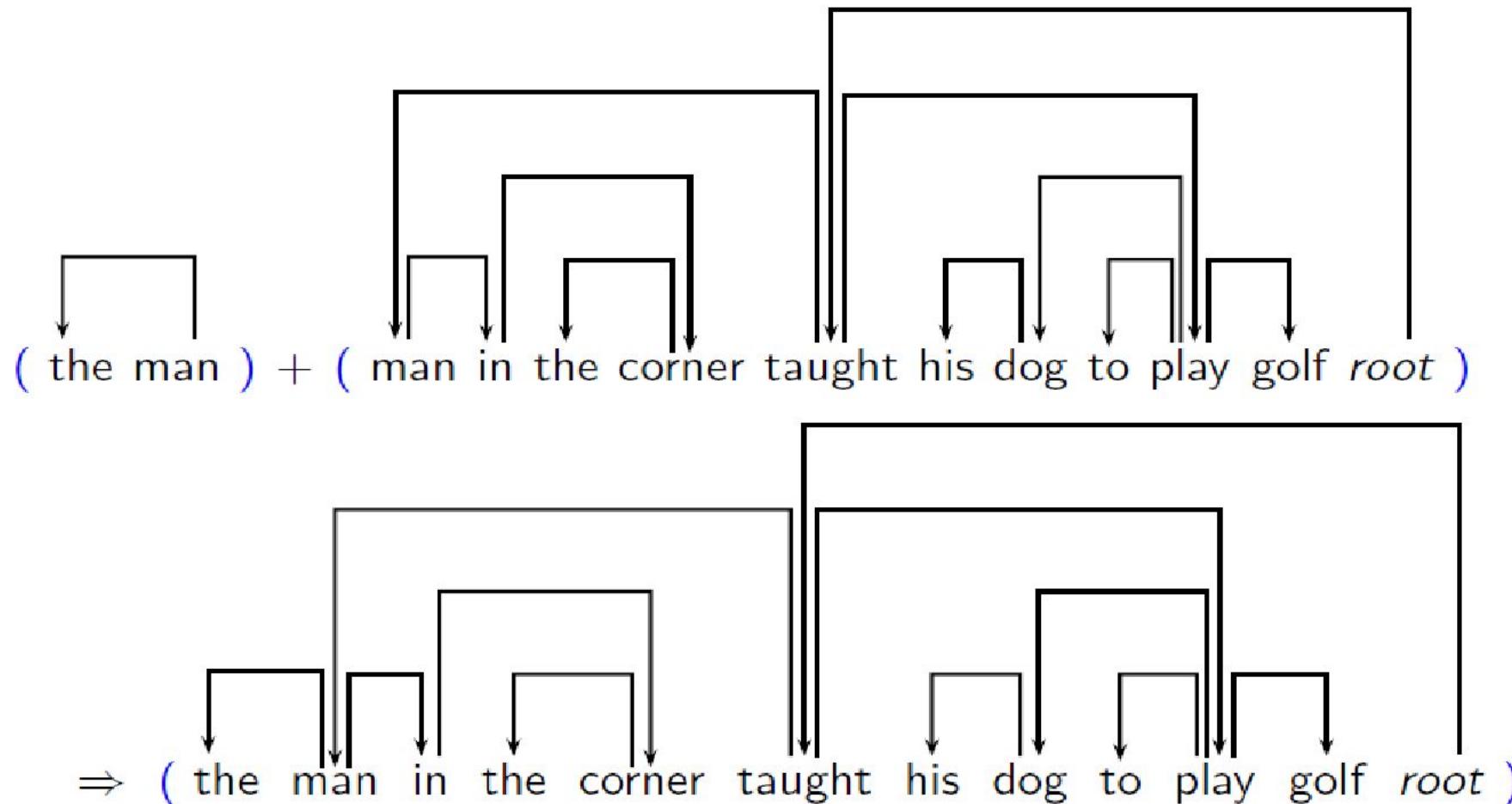
# Переход к НС-грамматикам: Eisner's Bilexical Algorythm

Объединяя в отрезки (spans) отрезки, имеющих словоформу в пересечении; это слово должно иметь «хозяина» либо в левом, либо в правом отрезке



# Переход к НС-грамматикам: Eisner's Bilexical Algorithm

Объединяем в отрезки (spans) отрезки, имеющих словоформу в пересечении; это слово должно иметь «хозяина» либо в левом, либо в правом отрезке



# Переход к НС

- Original version: [Hays 1964]
- ▲ Link Grammar: [Sleator and Temperley 1991]
- ▲ Earley-style parser with left-corner filtering:
- [Lombardo and Lesmo 1996]
- ▲ Bilexical grammar: [Eisner 1996a, Eisner 1996b, Eisner 2000]
- ▲ Bilexical grammar with discriminative estimation methods:
- [McDonald et al. 2005a, McDonald et al. 2005b]

# Методы

- НС-представление <-> зависимости
- Constraint satisfaction
- Deterministic parsing

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

- Зависимостные грамматики с ограничениями
- ▲ грамматика содержит множество ограничений: логические формулы, задающие “правильное” (well-formed) дерево
- ▲ Ограничение: логическое выражение, содержащее переменные с заданным множеством значений.
- ▲ Синтаксический анализ определяется как задача проверки удовлетворения ограничениям (a constraint satisfaction problem).
- ▲ Синтаксический анализ заключается в исключении связей, а не в их установлении
- ▲ В результате проверки удаляются те значения, которые противоречат ограничениям

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

Based on [Maruyama 1990]

- ▲ Example 1:
  - ▲  $\text{word}(\text{pos}(x)) = \text{DET} \Rightarrow$   
( $\text{label}(X) = \text{NMOD}$ ,  $\text{word}(\text{mod}(x)) = \text{NN}$ ,  $\text{pos}(x) < \text{mod}(x)$ )
  - ▲ A determiner (DET) modifies a noun (NN) on the right with the label NMOD.
- ▲ Example 2:
  - ▲  $\text{word}(\text{pos}(x)) = \text{NN} \Rightarrow$   
( $\text{label}(x) = \text{SBJ}$ ,  $\text{word}(\text{mod}(x)) = \text{VB}$ ,  $\text{pos}(x) < \text{mod}(x)$ )
  - ▲ A noun modifies a verb (VB) on the right with the label SBJ.
- ▲ Example 3:
  - ▲  $\text{word}(\text{pos}(x)) = \text{VB} \Rightarrow$   
( $\text{label}(x) = \text{ROOT}$ ,  $\text{mod}(x) = \text{nil}$ )
  - ▲ A verb modifies nothing, its label is ROOT.

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

Based on [Maruyama 1990]

△ Example 1:

△  $\text{word}(\text{pos}(x)) = \text{DET} \Rightarrow$   
( $\text{label}(X) = \text{NMOD}$ ,  $\text{word}(\text{mod}(x)) = \text{NN}$ ,  $\text{pos}(x) < \text{mod}(x)$ )

△ A determiner (DET) modifies a noun (NN) on the right with the label NMOD.

△ Example 2:

△  $\text{word}(\text{pos}(x)) = \text{NN} \Rightarrow$   
( $\text{label}(x) = \text{SBJ}$ ,  $\text{word}(\text{mod}(x)) = \text{VB}$ ,  $\text{pos}(x) < \text{mod}(x)$ )

△ A noun modifies a verb (VB) on the right with the label SBJ.

△ Example 3:

△  $\text{word}(\text{pos}(x)) = \text{VB} \Rightarrow$   
( $\text{label}(x) = \text{ROOT}$ ,  $\text{mod}(x) = \text{nil}$ )

△ A verb modifies nothing, its label is ROOT.

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

Constraint Grammar: [Karlsson 1990, Karlsson et al. 1995]

△ Constraint Dependency Grammar:

[Maruyama 1990, Harper and Helzerman 1995]

△ Functional Dependency Grammar: [Järvinen and Tapanainen 1998]

△ Topological Dependency Grammar: [Duchier 1999, Duchier 2003]

△ Extensible Dependency Grammar: [Debusmann et al. 2004]

△ Constraint Dependency Grammar with defeasible constraints:

[Foth et al. 2000, Foth et al. 2004, Menzel and Schröder 1998,  
Schröder 2002]

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

## Maruyama's Constraint Propagation

Three steps:

1. Form initial constraint network using a “core” grammar.
2. Remove local inconsistencies.
3. If ambiguity remains, add new constraints and repeat step 2.

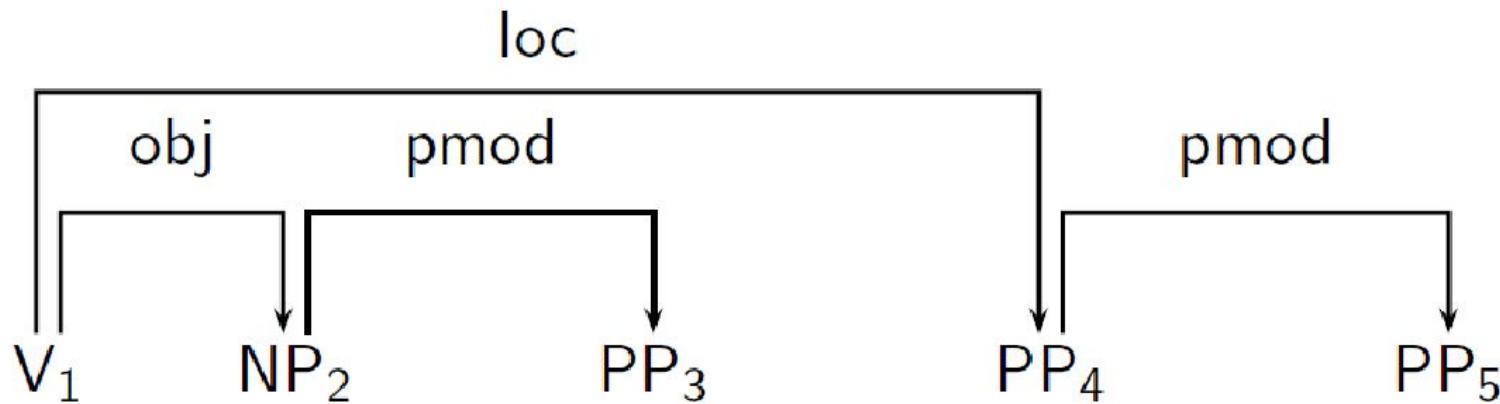
# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

## Constraint Propagation Example

- ▶ Problem: PP attachment
- ▶ Sentence: *Put the block on the floor on the table in the room*
- ▶ Simplified representation:  $V_1 \ NP_2 \ PP_3 \ PP_4 \ PP_5$

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

- ▶ Problem: PP attachment
- ▶ Sentence: *Put the block on the floor on the table in the room*
- ▶ Simplified representation:  $V_1 \ NP_2 \ PP_3 \ PP_4 \ PP_5$
- ▶ Correct analysis:



Put the block on the floor on the table in the room

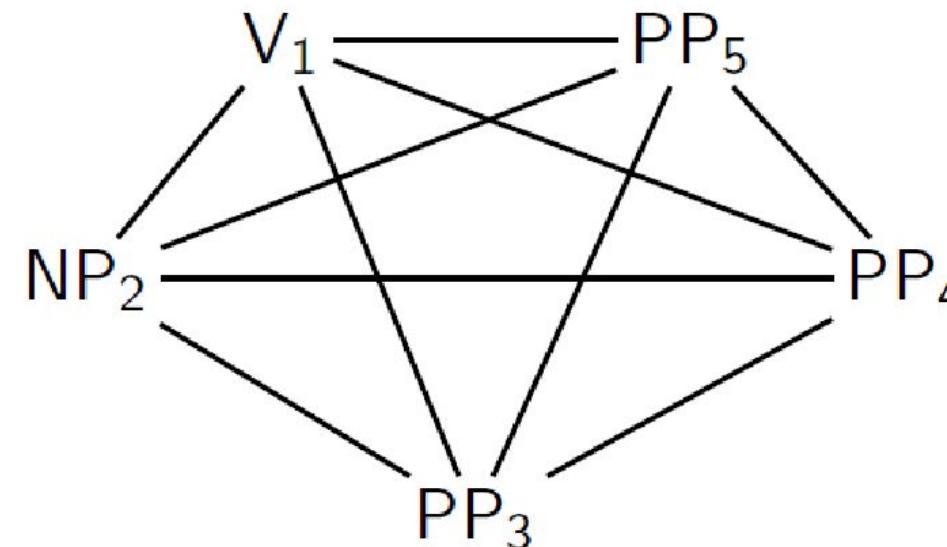
# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

## Initial Constraints

- ▶ ▶  $\text{word}(\text{pos}(x)) = \text{PP}$   
 $\Rightarrow (\text{word}(\text{mod}(x)) \in \{\text{PP}, \text{NP}, \text{V}\}, \text{mod}(x) < \text{pos}(x))$   
▶ A PP modifies a PP, an NP, or a V on the left.
- ▶ ▶  $\text{word}(\text{pos}(x)) = \text{PP}, \text{word}(\text{mod}(x)) \in \{\text{PP}, \text{NP}\}$   
 $\Rightarrow \text{label}(x) = \text{pmod}$   
▶ If a PP modifies a PP or an NP, its label is pmod.
- ▶ ▶  $\text{word}(\text{pos}(x)) = \text{PP}, \text{word}(\text{mod}(x)) = \text{V} \Rightarrow \text{label}(x) = \text{loc}$   
▶ If a PP modifies a V, its label is loc.
- ▶ ▶  $\text{word}(\text{pos}(x)) = \text{NP}$   
 $\Rightarrow (\text{word}(\text{mod}(x)) = \text{V}, \text{label}(x) = \text{obj}, \text{mod}(x) < \text{pos}(x))$   
▶ An NP modifies a V on the left with the label obj.
- ▶ ▶  $\text{word}(\text{pos}(x)) = \text{V} \Rightarrow (\text{mod}(x) = \text{nil}, \text{label}(x) = \text{root})$   
▶ A V modifies nothing with the label root.
- ▶ ▶  $\text{mod}(x) < \text{pos}(y) < \text{pos}(x) \Rightarrow \text{mod}(x) \leq \text{mod}(y) \leq \text{pos}(x)$   
▶ Modification links do not cross.

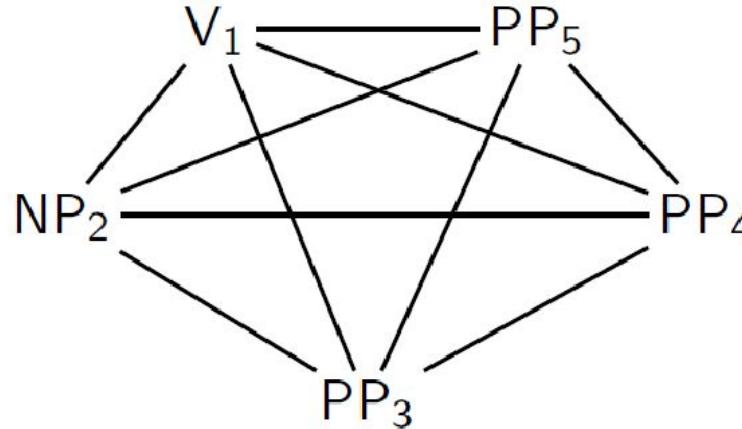
Проверка связей на соответствие  
ограничениям (a constraint satisfaction  
problem)

## Initial Constraint Network



# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

## Initial Constraint Network



Possible values  $\Leftarrow$  unary constraints:

$V_1$ : <root, nil>

$NP_2$ : <obj, 1>

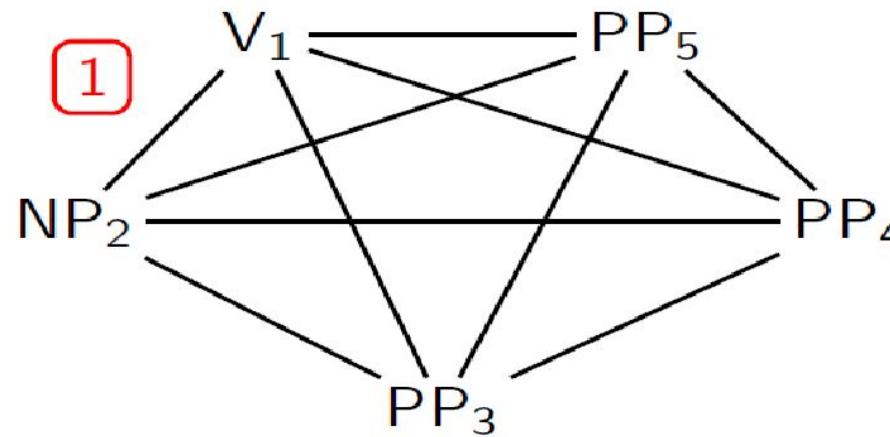
$PP_3$ : <loc, 1>, <pmod, 2>

$PP_4$ : <loc, 1>, <pmod, 2>, <pmod, 3>

$PP_5$ : <loc, 1>, <pmod, 2>, <pmod, 3>, <pmod, 4>

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

## Initial Constraint Network



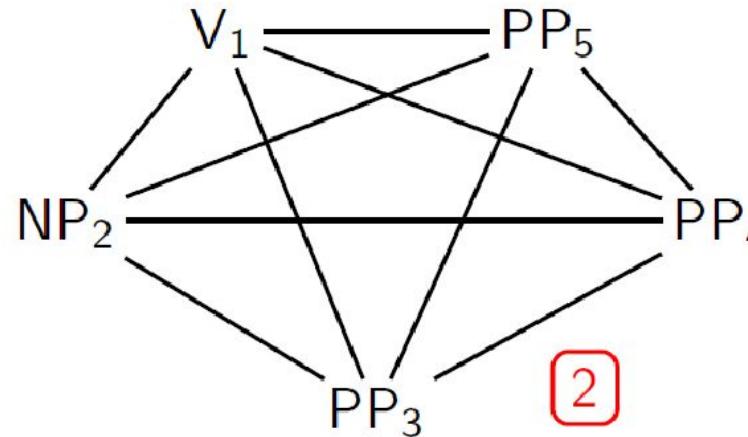
Each arc has a constraint matrix:

For arc 1:

$\downarrow V_1 \setminus NP_2 \rightarrow$	$\langle obj, 1 \rangle$
$\langle root, nil \rangle$	1

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

## Initial Constraint Network



Each arc has a constraint matrix:  
For arc 2:

$\downarrow PP_3 \setminus PP_4 \rightarrow$	$<loc, 1>$	$<pmod, 2>$	$<pmod, 3>$
$<loc, 1>$	1	0	1
$<pmod, 2>$	1	1	1

# Проверка связей на соответствие

## С ~~Adding New Constraints~~

- ▶ Still 14 possible analyses.
- ▶ Filtering with binary constraints does not reduce ambiguity.
- ▶ Introduce more constraints:
  - ▶  $\text{word}(\text{pos}(x)) = PP, \text{on\_table} \in \text{sem}(\text{pos}(x))$   
 $\Rightarrow \neg(\text{floor} \in \text{sem}(\text{mod}(x)))$
  - ▶ A floor is not on the table.
- ▶  $\text{label}(x) = \text{loc}, \text{label}(y) = \text{loc}, \text{mod}(x) = \text{mod}(y), \text{word}(\text{mod}(x)) = V$   
 $\Rightarrow x = y$
- ▶ No verb can take two locatives.

Each value in the domains of nodes is tested against the new constraints.

Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

## Modified Tables

Old:

$\downarrow \text{PP}_3 \setminus \text{PP}_4 \rightarrow$	$\langle \text{loc}, 1 \rangle$	$\langle \text{pmod}, 2 \rangle$	$\langle \text{pmod}, 3 \rangle$
$\langle \text{loc}, 1 \rangle$	1	0	1
$\langle \text{pmod}, 2 \rangle$	1	1	1

violates first constraint

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

## Modified Tables

Old:

$\downarrow PP_3 \setminus PP_4 \rightarrow$	$<loc, 1>$	$<pmod, 2>$	$<pmod, 3>$
$<loc, 1>$	1	0	1
$<pmod, 2>$	1	1	1

After applying first new constraint:

$\downarrow PP_3 \setminus PP_4 \rightarrow$	$<loc, 1>$	$<pmod, 2>$
$<loc, 1>$	1	0
$<pmod, 2>$	1	1

violates second constraint

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

## Modified Tables

Old:

$\downarrow PP_3 \setminus PP_4 \rightarrow$	$<loc, 1>$	$<pmod, 2>$	$<pmod, 3>$
$<loc, 1>$	1	0	1
$<pmod, 2>$	1	1	1

After applying first new constraint:

$\downarrow PP_3 \setminus PP_4 \rightarrow$	$<loc, 1>$	$<pmod, 2>$
$<loc, 1>$	1	0
$<pmod, 2>$	1	1

violates second constraint

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

## Modified Tables

Old:

$\downarrow PP_3 \setminus PP_4 \rightarrow$	$<loc, 1>$	$<pmod, 2>$	$<pmod, 3>$
$<loc, 1>$	1	0	1
$<pmod, 2>$	1	1	1

After applying first new constraint:

$\downarrow PP_3 \setminus PP_4 \rightarrow$	$<loc, 1>$	$<pmod, 2>$
$<loc, 1>$	0	0
$<pmod, 2>$	1	1

After applying second new constraint:

$\downarrow PP_3 \setminus PP_4 \rightarrow$	$<loc, 1>$	$<pmod, 2>$
$<pmod, 2>$	1	1

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

## Weighted Constraint Parsing

- ▶ Approach by [Foth et al. 2004, Foth et al. 2000, Menzel and Schröder 1998, Schröder 2002]
- ▶ Robust parser, which uses soft constraints
- ▶ Each constraint is assigned a weight between 0.0 and 1.0
- ▶ Weight 0.0: hard constraint, can only be violated when no other parse is possible
- ▶ Constraints assigned manually (or estimated from treebank)
- ▶ Efficiency: uses a heuristic transformation-based constraint resolution method

# Transformation-Based Constraint Resolution

- ▶ Heuristic search
- ▶ Very efficient
- ▶ Idea: first construct arbitrary dependency structure, then try to correct errors
- ▶ Error correction by transformations
- ▶ Selection of transformations based on constraints that cause conflicts
- ▶ Anytime property: parser maintains a complete analysis at any time ⇒ can be stopped at any time and return a complete analysis

# Проверка связей на соответствие ограничениям (a constraint satisfaction problem)

## Menzel et al.'s Results

- ▶ Evaluation on NEGRA treebank for German
- ▶ German more difficult to parse than English (free word order)
- ▶ Constituent-based parsing: labeled F measure including grammatical functions: 53.4 [Kübler et al. 2006], labeled F measure: 73.1 [Dubey 2005].
- ▶ Best CoNLL-X results: unlabeled: 90.4, labeled: 87.3 [McDonald et al. 2006].

Data	Unlabeled	Labeled
1000 sentences	89.0	87.0
< 40 words	89.7	87.7

# Методы

- НС-представление <-> зависимости
- Constraint satisfaction
- Deterministic parsing

# Deterministic Parsing

- ▶ Basic idea:
  - ▶ Derive a single syntactic representation (dependency graph) through a deterministic sequence of elementary parsing actions
  - ▶ Sometimes combined with backtracking or repair
- ▶ Motivation:
  - ▶ Psycholinguistic modeling
  - ▶ Efficiency
  - ▶ Simplicity

# Covington's Incremental Algorithm

- ▶ Deterministic incremental parsing in  $O(n^2)$  time by trying to link each new word to each preceding one [Covington 2001]:

```
PARSE( $x = (w_1, \dots, w_n)$ )
1   for  $i = 1$  up to  $n$ 
2     for  $j = i - 1$  down to 1
3       LINK( $w_i, w_j$ )
```

$$\text{LINK}(w_i, w_j) = \begin{cases} E \leftarrow E \cup (i, j) & \text{if } w_j \text{ is a dependent of } w_i; \\ E \leftarrow E \cup (j, i) & \text{if } w_i \text{ is a dependent of } w_j \\ E \leftarrow E & \text{otherwise} \end{cases}$$

- ▶ Different conditions, such as Single-Head and Projectivity, can be incorporated into the LINK operation.

# Анализ методом “сдвиг-свертка” (Shift-Reduce Algorithm)

- Структура данных:
  - Стек (Stack):  $[..., w_i]_S$  частично обработанных токенов
  - Очередь (Queue):  $[w_j ...]_q$  оставшихся токенов из анализируемого предложения (input tokens)
- Шаги анализа строятся из атомарных действий:
  - Добавление стрелки ( $w_i \rightarrow w_j, w_i <- w_j$ )
  - Операции над стеком и очередью
- Анализ слева на право ( $O(n)$ )
- Ограничение на проективность

# Deterministic parsing: Yamada Algorithm

- Три типа действий

$$\text{Shift} \quad \frac{[\dots]_S \quad [w_i, \dots]_Q}{[\dots, w_i]_S \quad [\dots]_Q}$$

$$\text{Left} \quad \frac{[\dots, w_i, w_j]_S \quad [\dots]_Q}{[\dots, w_i]_S \quad [\dots]_Q \quad w_i \rightarrow w_j}$$

$$\text{Right} \quad \frac{[\dots, w_i, w_j]_S \quad [\dots]_Q}{[\dots, w_j]_S \quad [\dots]_Q \quad w_i \leftarrow w_j}$$

- Алгоритм первоначально разрабатывался для Японского (имеющего строго левое ветвление), имел только два типа действий: сдвиг (**Shift**) и присоединение элемента справа (**Right**) [Kudo and Matsumoto 2002]
- Для английского добавили операцию **Left** [Yamada и Matsumoto 2003]
- Неоднократное «перемещение» по исходному предложению дает вычислительную сложность  $O(n^2)$

# Nivre's Algorithm

- ▶ Four parsing actions:

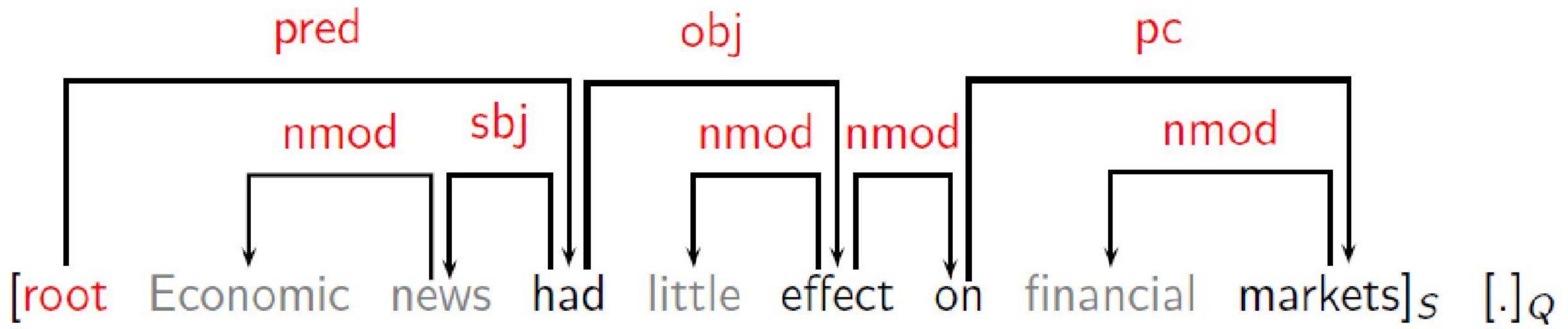
$$\begin{array}{ll} \text{Shift} & \frac{[\dots]s \quad [w_i, \dots]_Q}{[\dots, w_i]s \quad [\dots]_Q} \\ \\ \text{Reduce} & \frac{[\dots, w_i]s \quad [\dots]_Q \quad \exists w_k : w_k \rightarrow w_i}{[\dots]s \quad [\dots]_Q} \\ \\ \text{Left-Arc}_r & \frac{[\dots, w_i]s \quad [w_j, \dots]_Q \quad \neg \exists w_k : w_k \rightarrow w_i}{[\dots]s \quad [w_j, \dots]_Q \quad w_i \xleftarrow{r} w_j} \\ \\ \text{Right-Arc}_r & \frac{[\dots, w_i]s \quad [w_j, \dots]_Q \quad \neg \exists w_k : w_k \rightarrow w_j}{[\dots, w_i, w_j]s \quad [\dots]_Q \quad w_i \xrightarrow{r} w_j} \end{array}$$

Пояснение к Reduce:  
если для  $w_i$  находится  
хозяин внутри стека,  
то этот элемент  
окончательно обработан  
(хозяин может быть  
только один)  
и дальше в анализе не  
участвует

- ▶ Characteristics:

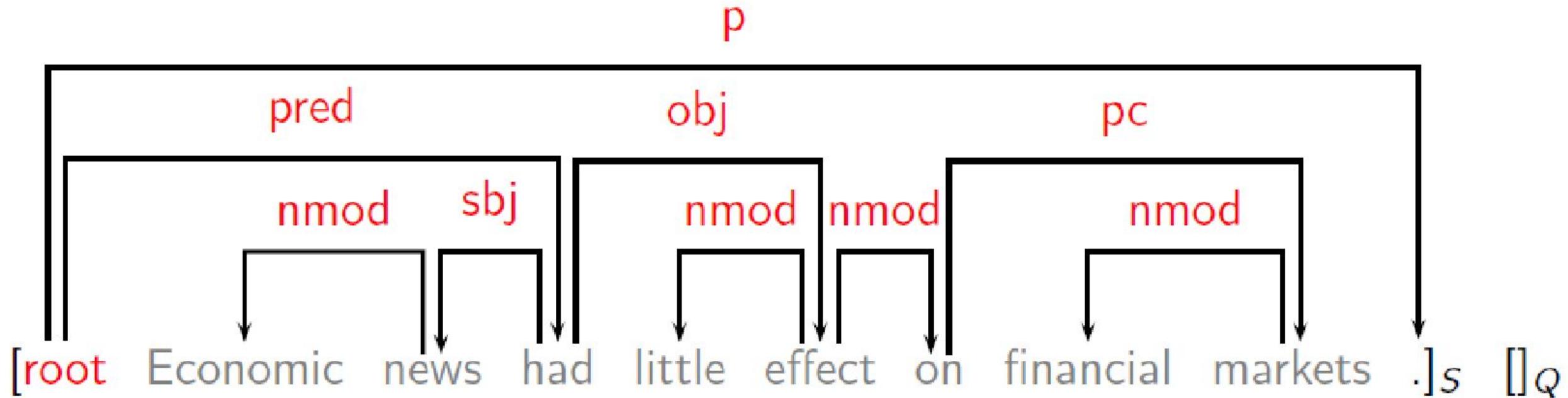
- ▶ Integrated labeled dependency parsing
- ▶ Arc-eager processing of right-dependents
- ▶ Single pass over the input gives time complexity  $O(n)$

# Пример



Right-Arc<sub>pc</sub>

# Пример



Right-Arc<sub>p</sub>

# Classifier-Based Parsing

- ▶ Data-driven deterministic parsing:
  - ▶ Deterministic parsing requires an **oracle**.
  - ▶ An oracle can be approximated by a **classifier**.
  - ▶ A classifier can be trained using **treebank** data.
- ▶ Learning methods:
  - ▶ Support vector machines (SVM)  
[Kudo and Matsumoto 2002, Yamada and Matsumoto 2003,  
Isozaki et al. 2004, Cheng et al. 2004, Nivre et al. 2006]
  - ▶ Memory-based learning (MBL)  
[Nivre et al. 2004, Nivre and Scholz 2004]
  - ▶ Maximum entropy modeling (MaxEnt)  
[Cheng et al. 2005]

# Feature Models

- ▶ Learning problem:
  - ▶ Approximate a function from **parser states**, represented by feature vectors to **parser actions**, given a training set of gold standard derivations.
- ▶ Typical features:
  - ▶ Tokens:
    - ▶ Target tokens
    - ▶ Linear context (neighbors in  $S$  and  $Q$ )
    - ▶ Structural context (parents, children, siblings in  $G$ )
  - ▶ Attributes:
    - ▶ Word form (and lemma)
    - ▶ Part-of-speech (and morpho-syntactic features)
    - ▶ Dependency type (if labeled)
    - ▶ Distance (between target tokens)

# State of the Art – English

- ▶ Evaluation:
  - ▶ Penn Treebank (WSJ) converted to dependency graphs
  - ▶ Unlabeled accuracy per word (W) and per sentence (S)
    - ▶ Deterministic classifier-based parsers  
[Yamada and Matsumoto 2003, Isozaki et al. 2004]
    - ▶ Spanning tree parsers with online training  
[McDonald et al. 2005a, McDonald and Pereira 2006]
    - ▶ Collins and Charniak parsers with same conversion

Parser	W	S
Charniak	92.2	45.2
Collins	91.7	43.3
McDonald and Pereira	91.5	42.1
Isozaki et al.	91.4	40.7
McDonald et al.	91.0	37.5
Yamada and Matsumoto	90.4	38.4

# Non-Projective Dependency Parsing

- ▶ Many parsing algorithms are restricted to projective dependency graphs.
- ▶ Is this a problem?
- ▶ Statistics from CoNLL-X Shared Task [Buchholz and Marsi 2006]
  - ▶ NPD = Non-projective dependencies
  - ▶ NPS = Non-projective sentences

Language	%NPD	%NPS
Dutch	5.4	36.4
German	2.3	27.8
Czech	1.9	23.2
Slovene	1.9	22.2
Portuguese	1.3	18.9
Danish	1.0	15.6

# Trainable Parsers

- ▶ Jason Eisner's **probabilistic dependency parser**
  - ▶ Based on bilexical grammar
  - ▶ Contact Jason Eisner: jason@cs.jhu.edu
  - ▶ Written in LISP
- ▶ Ryan McDonald's **MSTParser**
  - ▶ Based on the algorithms of  
[McDonald et al. 2005a, McDonald et al. 2005b]
  - ▶ URL:  
<http://www.seas.upenn.edu/~ryantm/software/MSTParser/>
  - ▶ Written in JAVA

# Trainable Parsers (2)

- ▶ Joakim Nivre's **MaltParser**
  - ▶ Inductive dependency parser with memory-based learning and SVMs
  - ▶ URL:  
<http://w3.msi.vxu.se/~nivre/research/MaltParser.html>
  - ▶ Executable versions are available for Solaris, Linux, Windows, and MacOS (open source version planned for fall 2006)

# Синтаксический анализ русского языка (без аннотации типа связи)

	P	R	F1	
Trieste	0,952	0,983	0,967	Compreno
Marceille	0,933	0,981	0,956	ЭТАП–3
Barcelona	0,895	0,980	0,935	SyntAutom
Toulon	0,889	0,947	0,917	SemSyn
Brega	0,863	0,980	0,917	Dictum
				Semantic analyzer
Nice	0,856	0,860	0,858	group
Napoli	0,789	0,975	0,872	AotSoft

# Трибанки для русского языка

- <http://www.ruscorpora.ru/search-syntax.html> - СинТагРус
- лингвистическая модели «Смысл ⇔ Текст» И.А.Мельчука и А.К. Жолковского; Лаборатория компьютерной лингвистики Института проблем передачи информации РАН (<http://cl.iitp.ru/ru/about/staff>)
- <http://otipl.philol.msu.ru/~soiza/testsynt/> - корпус с параллельной автоматической синтаксической разметкой (4 системы)
- RTB (Russian Treebank) - <http://rtb.maimbava.net/res01/rtb.php>  
(частичная ручная разметка отдельных типов связей:  
подлежащее, прямое дополнение, согласованное отименное  
зависимое) – link, reflexive12  
<http://corpus-i.compling.net/res01/rtb.php> - корпус с автоматической  
синтаксической разметкой (20 млн.), MaltParser

# Трибанки для русского языка

- <http://www.ruscorpora.ru/search-syntax.html> - СинТагРус
- лингвистическая модели «Смысл ⇔ Текст» И.А.Мельчука и А.К. Жолковского; Лаборатория компьютерной лингвистики Института проблем передачи информации РАН (<http://cl.iitp.ru/ru/about/staff>)
- <http://otipl.philol.msu.ru/~soiza/testsynt/> - корпус с параллельной автоматической синтаксической разметкой (4 системы)
- RTB (Russian Treebank) - <http://rtb.maimbava.net/res01/rtb.php>  
(частичная ручная разметка отдельных типов связей:  
подлежащее, прямое дополнение, согласованное отименное  
зависимое) – link, reflexive12

<http://corpus-i.compling.net/res01/rtb.php> - корпус с автоматической синтаксической разметкой (20 млн.), MaltParser

Link Grammar Parser -  
<http://www.abisource.com/projects/link-grammar/russian/>

# Доступные анализаторы для русского языка

- Malt Parser - <http://corpus.leeds.ac.uk/mocky/>
- <http://web-corpora.net/wsgi3/ru-syntax/>
- <https://github.com/tensorflow/models/tree/master/syntaxnet>
-

# Трибанки и полезные ссылки

- <http://otipl.philol.msu.ru/~soiza/testsynt/files/info-parse.htm>
  - ссылки на трибанки, синтаксические анализаторы и т.п.
- <http://nlpub.ru/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B>
- <http://nlpub.ru/>

# Трибанки и полезные ссылки

Mate

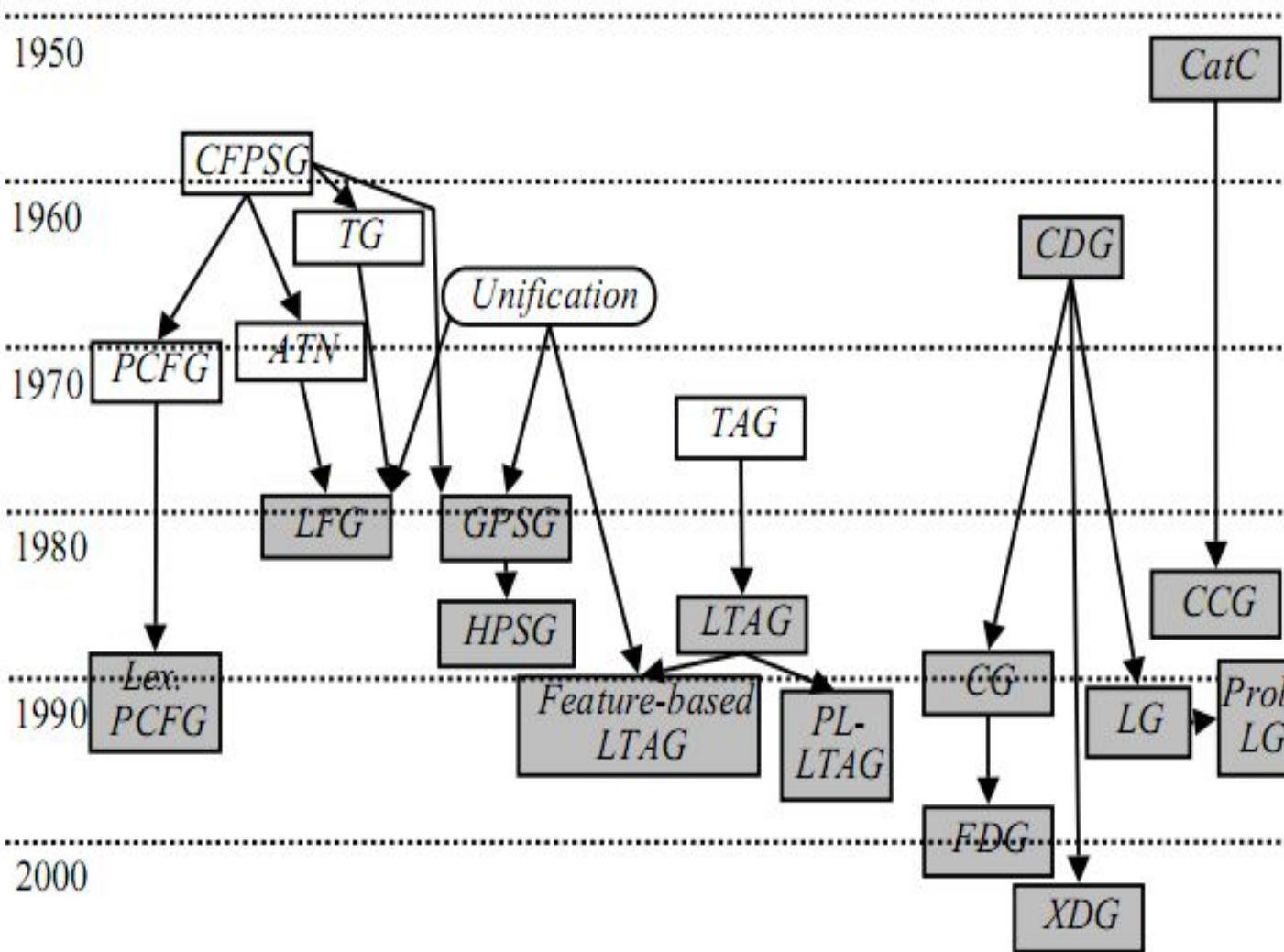
TurboParser

MSTParser (McDonald and Pereira, 2006)

MaltOptimizer

MaltParser

Wehrli, Eric, Yves Scherrer, and Luka Nerima. "On-line Multilingual Linguistic Services." (2016).



CatC, CCG – комбинаторно-категориальные грамматики

CFPSG – Context-Free Phrase Structure Grammar (CFG)  
PCFG – Probabilistic Context-Free Grammar

LexPCFG – Lexicalized Context-Free Grammar  
ATN - Augmented Transition Network – грамматика расширенных сетей переходов

LFG – лексико-функциональная грамматика

HPSG - Head-Driven Phrase Structure Grammar

TAG – Tree-adjoining grammar