

## Algoritmy syntaktické analýzy (pomocí CFG)

Vladimír Kadlec, Aleš Horák

E-mail: [hales@fi.muni.cz](mailto:hales@fi.muni.cz)  
[http://nlp.fi.muni.cz/nlp\\_intro/](http://nlp.fi.muni.cz/nlp_intro/)

Obsah:

- ▶ Základní postupy pro syntaktickou analýzu obecných CFG
- ▶ Tomitův zobecněný analyzátor LR
- ▶ Algoritmus CYK
- ▶ Tabulkové analyzátoři
- ▶ Porovnání jednotlivých algoritmů
- ▶ Syntaktická analýza s využitím strojového učení

## Syntaktická analýza

- ▶ Vstupy:
  - řetězec lexikálních kategorií (preterminálních symbolů)  $a_1 a_2 \dots a_n$   
 např.: ADJ CONJ ADJ N V PREP N '.'
  - bezkontextová gramatika  $G = \langle N, \Sigma, P, S \rangle$ .
- ▶ Výstup:
  - efektivní reprezentace derivačních stromů.

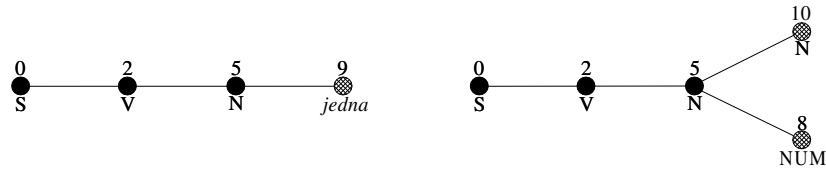
## Základní postupy pro syntaktickou analýzu obecných bezkontextových gramatik

- ▶ **obecná CFG** – rozsáhlá, (silně) víceznačná, s  $\epsilon$ -pravidly
- ▶ všechny uvedené algoritmy pracují s *polynomiální časovou a prostorovou složitostí*
- ▶ Tomitův zobecněný algoritmus LR (*generalized LR*)
- ▶ algoritmus CYK – Cocke, Younger, Kasami;
- ▶ tabulková (chart) analýza (*Chart Parsing*):
  - shora dolů (*top-down*);
  - zdola nahoru (*bottom-up*);
  - analýza řízená hlavou pravidla (*head-driven*);

## Tomitův zobecněný analyzátor LR

- ▶ **generalized LR parser (GLR)**
- ▶ Masaru Tomita: *Efficient parsing for natural language*, 1986
- ▶ standardní LR tabulka, která může obsahovat konflikty;
- ▶ zásobník je reprezentován **acyklickým orientovaným grafem (DAG)**
- ▶ derivační stromy jsou uloženy ve **sbaleném "lese" stromů**
- ▶ v podstatě stejný jako algoritmus LR
- ▶ udržujeme si **seznam aktivních uzlů** zásobníku (grafu)
- ▶ akce **redukce** provádíme vždy před akcemi čtení
- ▶ akci **čtení** provádíme pro všechny aktivní uzly najednou
- ▶ kde je to možné, tam uzly **slučujeme**

### Příklad konfliktu redukce/redukce



stav	položka	akce	symbol	další stav
5	$CLAUSE \rightarrow V N \bullet NUM$	shift	NUM	8
	$NN \rightarrow N \bullet N$		N	10
	$NUM \rightarrow \bullet jedna$		jedna	9
	$N \rightarrow \bullet tramvaj$		tramvaj	7
	$N \rightarrow \bullet jedna$			
9	$NUM \rightarrow jedna \bullet$	reduce (6)		
	$N \rightarrow jedna \bullet$	reduce (5)		

### Algoritmus CYK

- ▶ Gramatika musí být v **Chomského normální formě**

$$CNF: \begin{matrix} A \rightarrow BC \\ D \rightarrow 'd' \end{matrix}$$

- ▶ Převod libovolné CFG do CNF:

1. přidáme **nový kořen**  $S_0$ :  $S_0 \rightarrow S$
2. eliminujeme  $\epsilon$ -pravidla:

$$\begin{matrix} S \rightarrow Ab \mid B \\ A \rightarrow a \mid \epsilon \end{matrix} \rightarrow \begin{matrix} S \rightarrow Ab \mid b \mid B \\ A \rightarrow a \end{matrix}$$

3. eliminujeme **jednoduchá pravidla**:

$$\begin{matrix} A \rightarrow B \\ B \rightarrow a \mid CD \end{matrix} \rightarrow \begin{matrix} A \rightarrow a \mid CD \\ B \rightarrow a \mid CD \end{matrix}$$

4. rozgenerujeme **dlouhá pravidla**:

$$A \rightarrow BCD \rightarrow \begin{matrix} A \rightarrow BA_1 \\ A_1 \rightarrow CD \end{matrix}$$

### Algoritmus CYK, příklad – zadání

- ▶ vstupní gramatika je:

$$\begin{matrix} S \rightarrow AA|BB|AX|BY|a|b \\ X \rightarrow SA \\ Y \rightarrow SB \\ A \rightarrow a \\ B \rightarrow b \end{matrix}$$

- ▶ vstupní řetězec je  $w = abaaba$ .

### Algoritmus CYK, příklad – řešení (matice V)

abaaba

$$\begin{matrix} S \rightarrow AA|BB|AX|BY|a|b \\ X \rightarrow SA \\ Y \rightarrow SB \\ A \rightarrow a \\ B \rightarrow b \end{matrix}$$

$p$  – pozice,  $q$  – délka

$q \backslash p$	1	2	3	4	5	6
1	S, A	S, B	S, A	S, A	S, B	S, A
2	Y	X	S, X	Y	X	
3	S	$\emptyset$	Y	S		
4	X	S	$\emptyset$			
5	$\emptyset$	X				
6	S					

## Algoritmus CYK pokrač.

- ▶ Gramatika musí být v **Chomského normální formě**.
- ▶ Pro daný vstup délky  $n$  derivujeme **podřetězce** symbolů **délky  $q$**  na **pozici  $p$** , značíme  $w_{p,q}$ ,  $1 \leq p, q \leq n$ .
- ▶ Derivace **řetězců délky 1**,  $A \Rightarrow w_{p,1}$ , je prováděno prohledáváním terminálních pravidel.
- ▶ Derivace **delších řetězců**  $A \Rightarrow^* w_{p,q}$ ,  $q \geq 2$  vyžaduje aby platilo  $A \Rightarrow BC \Rightarrow^* w_{p,q}$ . Tedy z  $B$  derivujeme řetězec délky  $k$ ,  $1 \leq k \leq q$ , a z  $C$  derivujeme zbytek, řetězec délky  $q - k$ . Tzn.  $B \Rightarrow^* w_{p,k}$  a  $C \Rightarrow^* w_{p+k,q-k}$ . Kratší řetězce máme tedy vždy “předpočítané.”

## Algoritmus CYK pokrač.

```

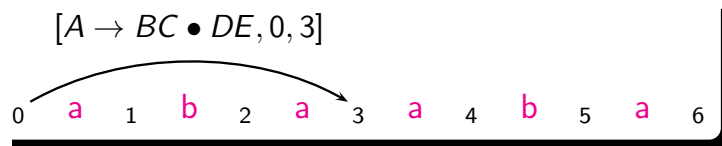
program CYK Parser;
begin
  for p := 1 to n do V[p, 1] := {A | A → ap ∈ P };
  for q := 2 to n do
    for p := 1 to n - q + 1 do
      V[p, q] = ∅;
      for k := 1 to q - 1 do
        V[p, q] =
          V[p, k] ∪
          ∪ {A | A → BC ∈ P, B ∈ V[p, k], C ∈ V[p + k, q - k]};
      od
    od
  od
end
    
```

složitost CYK je  $O(n^3)$

## Tabulkové (chart) analyzátoři

- ▶ Rozlišujeme tři základní typy **tabulkových analyzátorů**:
  - shora dolů;
  - zdola nahoru;
  - analýza řízená hlavou pravidla.
- ▶ Mnoho dalších variant je popsáno v:
 

Sikkel Klaas: *Parsing Schemata: A Framework for Specification and Analysis of Parsing Algorithm*, 1997.
- ▶ Neklade se žádné omezení na gramatiku.
- ▶ Analyzátoři typu “chart” v sobě většinou obsahují dvě datové struktury **chart** a **agendu**. Chart a agenda obsahují tzv. **hrany**.
- ▶ **Hrana** je trojice  $[A \rightarrow \alpha \bullet \beta, i, j]$ , kde:
  - $i, j$  jsou celá čísla,  $0 \leq i \leq j \leq n$  pro  $n$  slov ve vstupní větě
  - a  $A \rightarrow \alpha \beta$  je pravidlem vstupní gramatiky.



## Obecný analyzátor typu “chart”

```

program Chart Parser;
begin
  inicializuj (CHART);
  inicializuj (AGENDA);
  while (AGENDA ≠ ∅) do
    E := vezmi hranu z AGENDA;
    for each (hrana F, která může být vytvořena pomocí
      hrany E a nějaké jiné hrany z CHART) do
      if F ∉ AGENDA and F ∉ CHART and F ≠ E
        then přidej F do AGENDA;
      fi;
    od;
    přidej E do CHART;
  od;
end;
    
```

složitost tabulkové analýzy je  $O(n^3)$  ( $|Pravidla|$  bereme jako konstantu)

## Varianta shora dolů

### Inicializace:

- ▶  $\forall p \in P \mid p = S \rightarrow \alpha$  přidej hranu  $[S \rightarrow \bullet \alpha, 0, 0]$  do agendy.
- ▶ počáteční chart je prázdný.

### Iterace – vezmi hranu $E$ z agendy a pak:

- (*fundamentální pravidlo*) pokud je  $E$  ve tvaru  $[A \rightarrow \alpha \bullet, j, k]$ , potom pro každou hranu  $[B \rightarrow \gamma \bullet A \beta, i, j]$  v chartu vytvoř hranu  $[B \rightarrow \gamma A \bullet \beta, i, k]$ .
- (*uzavřené hrany*) pokud je  $E$  ve tvaru  $[B \rightarrow \gamma \bullet A \beta, i, j]$ , potom pro každou hranu  $[A \rightarrow \alpha \bullet, j, k]$  v chartu vytvoř hranu  $[B \rightarrow \gamma A \bullet \beta, i, k]$ .
- (*terminál na vstupu*) pokud je  $E$  ve tvaru  $[A \rightarrow \alpha \bullet a_{j+1} \beta, i, j]$ , vytvoř hranu  $[A \rightarrow \alpha a_{j+1} \bullet \beta, i, j+1]$ .
- (*predikce*) pokud je  $E$  ve tvaru  $[A \rightarrow \alpha \bullet B \beta, i, j]$  potom pro každé pravidlo  $B \rightarrow \gamma \in P$ , vytvoř hranu  $[B \rightarrow \bullet \gamma, j, j]$ .

## Příklad – tabulkové analýzy (typu chart)

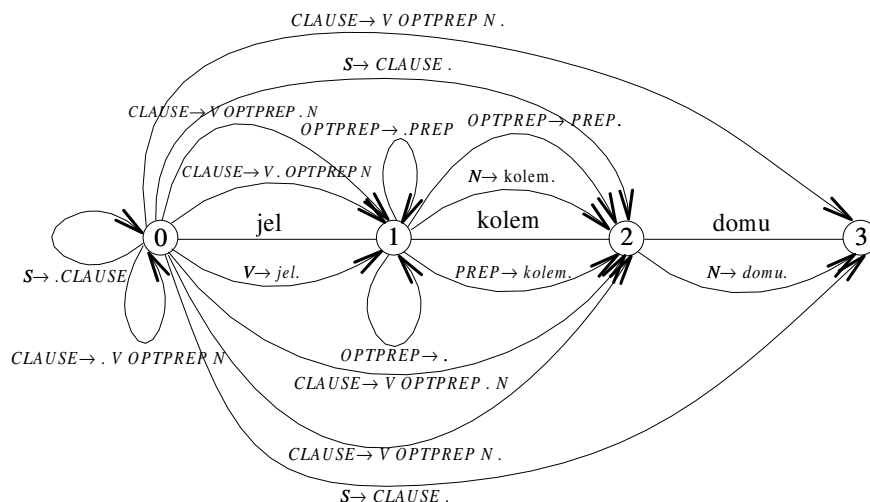
### Gramatika:

$S$	$\rightarrow$	$CLAUSE$
$CLAUSE$	$\rightarrow$	$V OPTPREP N$
$OPTPREP$	$\rightarrow$	$\epsilon$
$OPTPREP$	$\rightarrow$	$PREP$
$V$	$\rightarrow$	$jel$
$PREP$	$\rightarrow$	$kolem$
$N$	$\rightarrow$	$domu$
$N$	$\rightarrow$	$kolem$

### Věta:

"jel kolem domu" ( $a_1=jel, a_2=kolem, a_3=domu$ ).

## Příklad – chart po analýze shora dolů



## Varianta zdola nahoru

### Inicializace:

- ▶  $\forall p \in P \mid p = A \rightarrow \epsilon$  přidej hrany  $[A \rightarrow \bullet, 0, 0], [A \rightarrow \bullet, 1, 1], \dots, [A \rightarrow \bullet, n, n]$  do agendy.
- ▶  $\forall p \in P \mid p = A \rightarrow a_i \alpha$  přidej hranu  $[A \rightarrow \bullet a_i \alpha, i-1, i-1]$  do agendy.
- ▶ počáteční chart je prázdný.

### Iterace – vezmi hranu $E$ z agendy a pak:

- (*fundamentální pravidlo*) pokud je  $E$  ve tvaru  $[A \rightarrow \alpha \bullet, j, k]$ , potom pro každou hranu  $[B \rightarrow \gamma \bullet A \beta, i, j]$  v chartu vytvoř hranu  $[B \rightarrow \gamma A \bullet \beta, i, k]$ .
- (*uzavřené hrany*) pokud je  $E$  ve tvaru  $[B \rightarrow \gamma \bullet A \beta, i, j]$ , potom pro každou hranu  $[A \rightarrow \alpha \bullet, j, k]$  v chartu vytvoř hranu  $[B \rightarrow \gamma A \bullet \beta, i, k]$ .
- (*terminál na vstupu*) pokud je  $E$  ve tvaru  $[A \rightarrow \alpha \bullet a_{j+1} \beta, i, j]$ , potom vytvoř hranu  $[A \rightarrow \alpha a_{j+1} \bullet \beta, i, j+1]$ .
- (*predikce*) pokud je  $E$  ve tvaru  $[A \rightarrow \alpha \bullet, i, j]$ , potom pro každé pravidlo  $B \rightarrow A \gamma$  vstupní gramatiky vytvoř hranu  $[B \rightarrow \bullet A \gamma, i, j]$ .

## Analýza řízená hlavou pravidla

- ▶ *head-driven chart parsing*
- ▶ **Hlava pravidla** – libovolný (určený) symbol z pravé strany pravidla.  
Například pravidlo  $CLAUSE \rightarrow V \underline{PREP} N$  může mít hlavy  $V$ ,  $PREP$ ,  $N$ .
- ▶ Epsilon pravidlo má hlavu  $\epsilon$ .
- ▶ Hrana v analyzátoru řízeném hlavou pravidla – trojice  $[A \rightarrow \alpha \bullet \beta \bullet \gamma, i, j]$ , kde  $i, j$  jsou celá čísla,  $0 \leq i \leq j \leq n$  pro  $n$  slov ve vstupní větě a  $A \rightarrow \alpha \beta \gamma$  je pravidlo vstupní gramatiky a hlava je v  $\beta$ .
- ▶ Algoritmus vlastní analýzy (varianta zdola nahoru) je podobný jednoduchému přístupu. Analýza neprobíhá zleva doprava, ale **začíná na hlavě** daného pravidla.

## Analyzátor řízený hlavou pravidla

### Inicializace:

- ▶  $\forall p \in P \mid p = A \rightarrow \epsilon$  přidej hrany  $[A \rightarrow \bullet \bullet, 0, 0]$ ,  $[A \rightarrow \bullet \bullet, 1, 1]$ , ...,  $[A \rightarrow \bullet \bullet, n, n]$  do agendy.
- ▶  $\forall p \in P \mid p = A \rightarrow \alpha \underline{a_i} \beta$  ( $a_i$  je hlavou pravidla) přidej hranu  $[A \rightarrow \alpha \bullet a_i \bullet \beta, i-1, i]$  do agendy.
- ▶ počáteční chart je prázdný.

*Je tato inicializace v pořádku?*

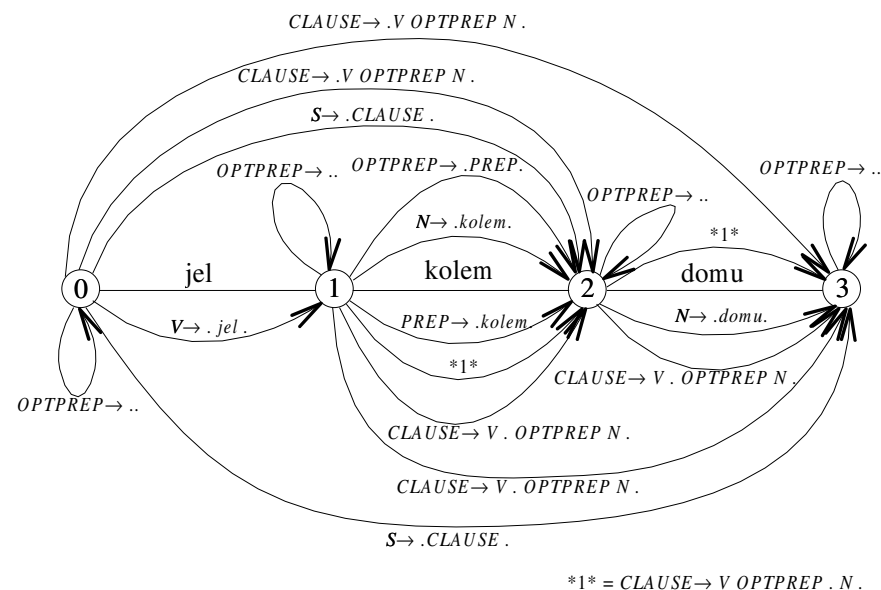
Co když inicializace nic nepřidá? (žadne  $\epsilon$  ani žádný terminál jako hlava)  
Odpověď: taková gramatika by generovala prázdný jazyk.

## Analyzátor řízený hlavou pravidla pokrač.

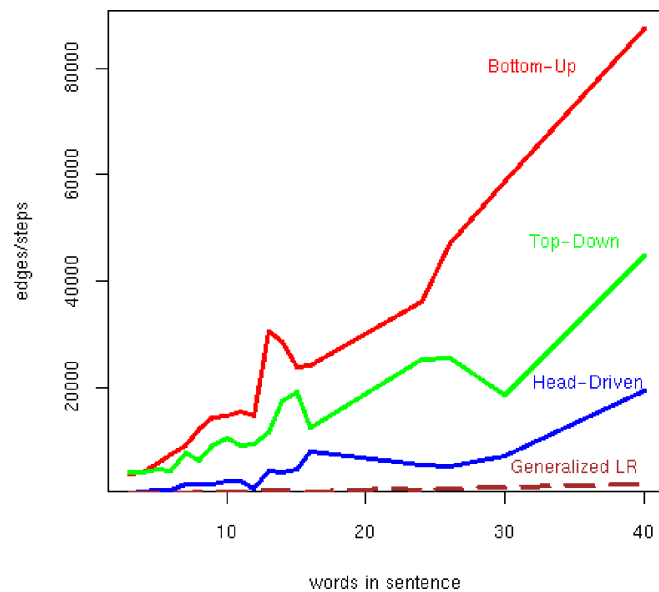
**Iterace** – vezmi hranu  $E$  z agendy a pak:

- ▶ pokud je  $E$  ve tvaru  $[A \rightarrow \bullet \alpha \bullet, j, k]$ , potom pro každou hranu:  $[B \rightarrow \beta \bullet \gamma \bullet A \delta, i, j]$  v chartu vytvoř hranu  $[B \rightarrow \beta \bullet \gamma A \bullet \delta, i, k]$ ,
- ▶ pro  $[B \rightarrow \beta A \bullet \gamma \bullet \delta, k, l]$  v chartu vytvoř hranu  $[B \rightarrow \beta \bullet A \gamma \bullet \delta, j, l]$ .
- ▶ pokud je  $E$  ve tvaru  $[B \rightarrow \beta \bullet \gamma \bullet A \delta, i, j]$ , potom pro každou hranu  $[A \rightarrow \bullet \alpha \bullet, j, k]$  v chartu vytvoř hranu  $[B \rightarrow \beta \bullet \gamma A \bullet \delta, i, k]$ .
- ▶ pokud je  $E$  ve tvaru  $[B \rightarrow \beta A \bullet \gamma \bullet \delta, k, l]$ , potom pro každou hranu  $[A \rightarrow \bullet \alpha \bullet, j, k]$  v chartu vytvoř hranu  $[B \rightarrow \beta \bullet A \gamma \bullet \delta, j, l]$ .
- ▶ pokud je  $E$  ve tvaru  $[A \rightarrow \beta a_i \bullet \gamma \bullet \delta, i, j]$ , potom vytvoř hranu  $[A \rightarrow \beta \bullet a_i \gamma \bullet \delta, i-1, j]$ .
- ▶ pokud je  $E$  ve tvaru  $[A \rightarrow \beta \bullet \gamma \bullet a_{j+1} \bullet \delta, i, j]$ , potom vytvoř hranu  $[A \rightarrow \beta \bullet \gamma a_{j+1} \bullet \delta, i, j+1]$ .
- ▶ pokud je  $E$  ve tvaru  $[A \rightarrow \bullet \alpha \bullet, i, j]$ , potom pro každé pravidlo  $B \rightarrow \beta \underline{A} \gamma$  ve vstupní gramatice vytvoř hranu  $[B \rightarrow \beta \bullet A \bullet \gamma, i, j]$  (symbol  $A$  je hlavou pravidla).

## Příklad – chart po analýze řízené hlavou pravidla

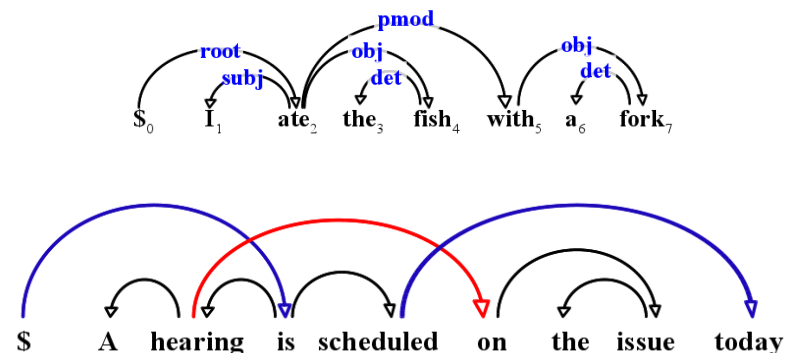


## Porovnání jednotlivých algoritmů



## Syntaktická analýza s využitím strojového učení

- ▶ nejčastěji pro **závislostní formalismy**
- ▶ **jedna hrana** pro každé slovo
- ▶ složitější pro **neprojektivní stromy**

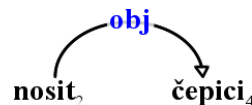


Example from "Dependency Parsing" by Kübler, Nivre, and McDonald, 2009

## Hodnocení úspěšnosti

### základní informace:

- ▶ **hlava** – které slovo je **řídící**
- ▶ **potomek** – které slovo je **závislé**
- ▶ **typ** – označení **typu hrany (label)**



### metriky (vždy procentuálně):

- ▶ **Unlabeled attachment score (UAS)** – slova, která mají **správnou hlavu**
- ▶ **Labeled attachment score (LAS)** – slova, která mají **správnou hlavu a typ**
- ▶ **Root Accuracy (RA)** – analýzy, které mají **správný kořen**
- ▶ **Complete Match rate (CM)** – zcela **správné analýzy**

## Formalizace závislostní analýzy pro učení

$$Tree^{best} = \arg \max_{Tree \in \Phi(Sentence)} score(Sentence, Tree)$$

- ▶ **Sentence** =  $x_1 x_2 \dots x_n$  – vstupní věta
- ▶  $(h, p)$  – hrana mezi **hlavou**  $x_h$  a **potomkem**  $x_p$
- ▶ **Tree** =  $\{(h, p) : 0 \leq h \leq n, 0 < p \leq n\}$  – potenciální **strom**
- ▶  $\Phi(Sentence)$  – množina všech možných závislostních **stromů nad Sentence**
- ▶ **score(Sentence, Tree)** – závisí na algoritmu, např.

$$score(Sentence, Tree) = \sum_{(h,p) \in Tree} score(Sentence, h, p)$$

## Způsob řešení závislostní analýzy

základní přístupy:

- ▶ **řešení pomocí přechodových akcí** (*transition-based*) – sekvence akcí přiřazujících závislostní hrany, využívá zásobník *arc-standard* akce – *shift*, *leftarc\_type*, *rightarc\_type* takto odpovídá shift-reduce analýze, s dalšími akcemi (*swap*, *reduce*) zvládne i neprojektivní analýzy např. *MaltParser* (Nivre et al, 2006)
- ▶ **grafové řešení** (*graph-based*) – tvorba stromu z ohodnoceného seznamu hran grafové řešení je **přesnější** na delších větách, klade ale větší **požadavky** na vlastnosti *score()*

## Závislostní analýza pomocí přechodových akcí – příklad

krok	zásobník	vstup	akce	hrana
0	[root]	[dej, Karlovi, tu, novou, knihu]	SHIFT	
1	[root, dej]	[Karlovi, tu, novou, knihu]	SHIFT	
2	[root, dej, Karlovi]	[tu, novou, knihu]	RIGHTARC	(dej → Karlovi)
3	[root, dej]	[tu, novou, knihu]	SHIFT	
4	[root, dej, tu]	[novou, knihu]	SHIFT	
5	[root, dej, tu, novou]	[knihu]	SHIFT	
6	[root, dej, tu, novou, knihu]	[]	LEFTARC	(novou ← knihu)
7	[root, dej, tu, knihu]	[]	LEFTARC	(tu ← knihu)
8	[root, dej, knihu]	[]	RIGHTARC	(dej → knihu)
9	[root, dej]	[]	RIGHTARC	(root → dej)
10	[root]	[]	—	

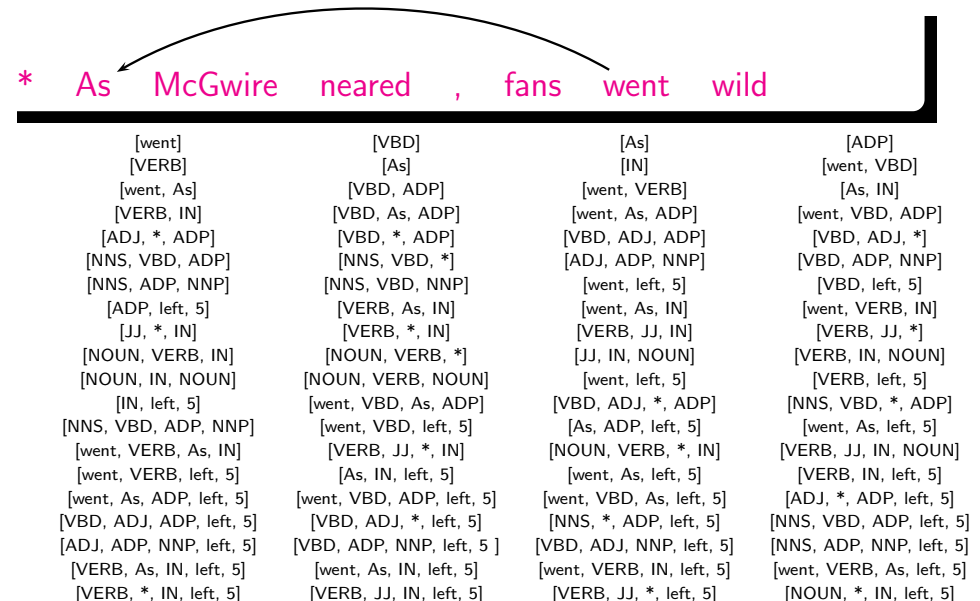
## Grafové řešení závislostní analýzy

2 úkoly:

- ▶ **nalezení stromu** (*search problem*)
  - známe **skóre** hran, jak najdeme *Tree<sup>best</sup>*
  - např. *Maximum Spanning Tree* (McDonald et al, 2005)
- ▶ **učení** (*learning problem*)
  - máme zadané **věty a stromy**, jak určíme **skóre hran**
  - pomocí **rysů hran** a **online učení**

UDPipe 2 implementuje grafové řešení analýzy

## Rysy závislostních hran



(příklad z Rush and Petrov, 2012)

